

An Intrusion Diagnosis Perspective on Cloud Computing

Junaid Arshad, Paul Townend, Jie Xu

School of Computing, University of Leeds, Leeds. LS2 9JT

Abstract

Cloud computing is an emerging paradigm with virtual machine as its enabling technology. As with any other internet based technology, security underpins widespread success of cloud computing. However, cloud computing introduces new challenges with respect to security mainly due to the unique characteristics inherited via virtual machine technology. In this chapter, we focus on the challenges imposed on intrusion diagnosis for clouds due to these characteristics. In particular, we identify the importance of intrusion diagnosis problem for clouds and the novel challenges for intrusion diagnosis for clouds. Also, we propose a solution to address these challenges and demonstrate the effectiveness of the proposed solution with empirical evaluation.

1. Cloud Computing

The advent of internet technologies has directed the revival of e-Science. It has significantly changed the methods used in e-Science along with the emergence of new computing paradigms to facilitate e-Science research. Cloud computing is one of such emerging paradigms which makes use of the contemporary virtual machine technology. The consonance between internet and virtual machine technologies enable Cloud computing to emerge as a paradigm with promising prospects to

facilitate the development of large scale, flexible computing infrastructures, available on-demand to meet the computational requirements of e-Science applications. Cloud computing has witnessed widespread acceptance mainly due to compelling characteristics such as; Live Migration, Isolation, Customization and Portability, thereby increasing the value attached with such infrastructures. The virtual machine technology has profound role in it. Amazon [2], Google [8] and GoGrid [7] represent some of commercial Cloud computing initiatives whereas Nimbus [13] and OpenNebula [14] represent academic efforts to establish a Cloud.

Cloud Computing has been defined in different ways by different sources however, for the purpose of our research, we define Clouds as *a high performance computing infrastructure based on system virtual machines to provide on-demand resource provision according to the service level agreements established between a consumer and a resource provider.*

A system virtual machine, as described in this definition, serves as the fundamental unit for the realization of a Cloud infrastructure and emulates a complete and independent operating environment. Within the scope of this chapter, we define the cloud platforms focused at satisfying computation requirements of compute intensive workloads as *Compute Clouds* whereas those facilitating large scale data storage as *Storage or Data Clouds*. For the rest of this chapter, we use terms *cloud computing* and *clouds* interchangeably to refer to our definition of compute clouds. As described in the above definition, cloud computing involves on-demand provision of virtualized resources based on Service Level Agreements (SLA) [26] thereby facilitating the user to acquire resources at runtime by defining the specifications of the resource required. The user and the resource provider are expected to negotiate the terms and conditions of the resource usage through SLAs so as to protect the quality of service being committed at resource acquisition stage.

2. Intrusion Diagnosis for Clouds

In order to stimulate extensive adoption of clouds, there is need to develop mechanisms focused at improving the security of such infrastructures. This is aggravated by the adoption of *pay-per-use* model by Cloud providers whereby customers usually pay for the resources they use. Related to this, clouds inherit unique characteristics such as diversity, mobility and flexibility from virtual machines, which present novel security challenges and, therefore, require dedicated efforts to address them [6]. In this chapter, we focus on the challenges due to one of these characteristics i.e. diversity. As described in figure 1, virtual machines provide the ability to host multiple different execution environments on a single physical machine, which enables a cloud provider to be able to address diverse user requirements with same physical resources. However, it also poses a number of novel security challenges such as; evaluating the impact of an intrusion on the guest virtual machines [3]. From figure 1, a security module residing in the domain 0 of a virtualized resource has to evaluate the impact of an intrusion on the guest virtual machines. This process becomes non-trivial given the potentially different security requirements of the guest virtual machines. We define the impact of an intrusion on a virtual machine as the *Level of Severity* (LoS) of the intrusion. Also, intrusion diagnosis is traditionally defined as the process to discover the cause of an intrusion [32]. For the purpose of our research, we define intrusion diagnosis to be *the process to evaluate the level of severity of an intrusion for a virtual machine*. For the rest of this chapter, we use terms intrusion diagnosis and intrusion severity analysis interchangeably to refer to process defined above.

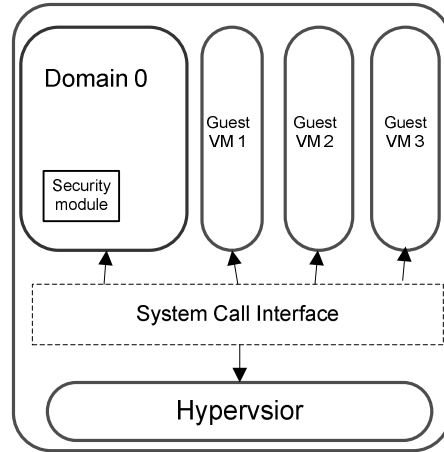


Figure 1: A virtualized resource

Intrusion diagnosis, as defined above, has a significant impact on the overall security of a system such as; selecting appropriate response mechanism, depending on the mechanisms used to achieve this evaluation. This will enable provision of an intelligent response selection mechanism, which facilitates triggering response mechanisms based on the severity of an intrusion for the victim application. Furthermore, a customized severity analysis also facilitates delivery of virtual machine specific quality of service with respect to security, which is vital for a user-oriented environment of clouds computing. This is envisioned to enable the cloud providers to devise Quality of Service (QoS) based pricing strategies, taking into account the quality of security service delivered in relation with the security requirements of virtual machines. To the best of our knowledge, we are the first to identify the intrusion diagnosis problem, highlight its importance and propose a solution to address it for virtual machine based systems in general and clouds in particular [3].

2.1 Requirements of Clouds for Intrusion Diagnosis

As described earlier in this chapter, cloud computing inherits unique characteristics from virtual machines, which present novel challenges for security in general and intrusion diagnosis in particular. In relation to this, we describe the requirements of clouds with respect to intrusion diagnosis and their comparison with contemporary efforts.

- **Comprehensive Severity Evaluation**

One of the defining characteristics of virtual machines is their ability to facilitate hosting multiple execution environments on a single physical machine. This governs diversity in the infrastructure, which demands a comprehensive severity evaluation. Furthermore, diversity also requires comprehensive representation of security characteristics of the guest virtual machines whereas contemporary efforts lack this comprehensiveness as described in the previous section.

- **Real-time Operation**

Clouds support flexible infrastructure where guest virtual machines can be created, migrated and deleted at runtime. This requires a security module to be considerate of the dynamic nature of the infrastructure. Traditional severity evaluation methods are static by nature mainly due to the static nature of the infrastructures. Furthermore, addition, deletion and migration of nodes in traditional infrastructures is not particularly frequent. Therefore, traditional methods do not meet this requirement of Clouds.

- **Automatic Approach**

With regards to flexibility and dynamic nature of clouds, an automated approach is mandatory. However, the existing approaches are largely manual due to their dependence on a human administrator. As described earlier in this chapter, the

automatic existing approaches lack mechanisms to automate the severity evaluation, which hampers their ability to match this requirement of clouds.

- Customization

As described earlier, clouds inherit diversity from virtual machines. Now, each guest virtual machine can potentially have different security characteristics. For example, a physics workload can have different priorities for security attributes as compared to a social science workload. Furthermore, we hold that security characteristics of an application dictate the severity of an intrusion on that application [4]. Therefore, it is necessary for a severity evaluation approach for clouds to enhance customization to a virtual machine level. On the contrary, existing methods for severity evaluation lack customization as policy definition is mostly done at group or domain level comprising a number of nodes.

- Minimized Response Time

Due to the runtime behavior of a cloud computing infrastructure, intrusion response time becomes critical. It is, therefore, required for a severity evaluation method to be integrated with other security modules such as intrusion detection and response systems. Traditional methods do not fulfill this requirement as most of the security modules are considered to be isolated.

As explained by the comparison above, we conclude that existing efforts to evaluate intrusion severity do not fulfill the requirements of clouds. Therefore, an intrusion severity analysis method is required which meets the requirements described above. In [3], we have proposed such a method and we intend to explain it in detail in the following sections of this chapter.

2.2 Contemporary Efforts for Intrusion Diagnosis

With respect to traditional distributed systems such as Grids, the intrusion diagnosis problem is not relevant at host level. This is because Grids allocate a complete physical node to a particular workload. Therefore, the security module in this case can be implemented as traditional host or network based system. However, as described in the previous section, both these approaches introduce a tradeoff between visibility and isolation.

Related to this, intrusion severity analysis has been studied in contemporary literature particularly in network-based systems where a security module is usually installed at the border node and is responsible for the whole network. The state of the art in this regard can be divided into two types i.e. administrator dependent and Intrusion Response Systems (IRS). With respect to intrusion response systems, most of the existing systems are based on the alerts from an Intrusion Detection System (IDS) or other information about an attack. There are some IRSs which do take into account the level of severity when selecting a response mechanism, however, this metric is again supposed to be decided by an administrator at an initial stage of site policy definition. Therefore, both types of contemporary systems involve a decisive human factor, which can prove to be the weak link in the severity evaluation process.

With respect to severity evaluation, few mechanisms have been proposed to facilitate a human administrator. In this regard, [24] and [25] present more formal methods to evaluate the severity or impact of intrusions for different applications. In [24], the severity evaluation is proposed to be a function of *Criticality*, *Lethality*, *Countermeasures* and *Net Countermeasures*. As can be inferred from their names, the subjective nature of these terms hampers their applicability in a flexible, diverse and user-driven system such as clouds. Furthermore, the analysis is assumed to be performed by a human administrator, which leads to manifold problems. Firstly, metrics such as Criticality and le-

thality are relative rather than absolute. Therefore, these require an in depth knowledge of the system under attack, the attack itself and also parameters defining the current status of the victim. Secondly, the metrics such as Countermeasures and Net Countermeasures are only applicable for well-known intrusions. Finally, the manual analysis also deteriorates the response time for an intrusion.

Common Vulnerability Scoring System (CVSS) [25] defines three metric groups and formula to calculate the impact of a vulnerability. The objective of this method is to facilitate a system administrator to perform the manual analysis so as to designate impact factor of a vulnerability before it is exploited. It does take into account custom security requirements by the notion of *Environmental Metrics* but these are optional and do not affect the score unless explicitly included by the user. This approach has several limitations. Firstly, it assumes manual execution of the whole process i.e. a representative of user has to decide on the values of different metrics such as the Availability, Confidentiality and Integrity Impacts. These metrics then contribute to the resultant impact metric. Secondly, the metrics are overly abstract which impede a human's ability to clearly specify the application specific metrics. For instance, availability, integrity and confidentiality are proposed to have three levels of impact i.e. *none*, *partial* or *complete*. These terms are too vague to accurately express the impact of a vulnerability on a particular attribute of security.

From the above discussion, we conclude that none of the contemporary mechanisms address the requirements of clouds with respect to intrusion diagnosis as established in the previous section. Therefore, we hold that a more fine-grained analysis is required, facilitated by comprehensive representation of user requirements.

3. An Automatic Intrusion Diagnosis Approach for Clouds

As highlighted in the previous section, there is a need to establish a dedicated intrusion diagnosis mechanism for clouds. We address the requirements described in previous section as follows: firstly, we achieve comprehensive and virtual machine specific severity evaluation by quantifying security at the virtual machine level, thereby, improving the representation of virtual machine security characteristics in our evaluation. With respect to real-time behavior, we incorporate SLA state and attack frequency in our analysis that makes it state or time-aware. Also, we propose to establish our system in the hypervisor of a virtualized resource, which enables our approach to be considerate of the creation, migration or deletion of guest virtual machines efficiently. In order to facilitate automation, we minimize the human factor by incorporating security requirements with SLA, thereby allowing applications to interact with virtualized resources without any human intervention. Furthermore, the severity evaluation process is triggered automatically without any human input, again eliminating the human factor. Finally, we have proposed an abstract model for integrated intrusion detection and diagnosis system in [3]. By integrating the severity evaluation with intrusion detection system, we envisage to reduce the intrusion response system and eliminate the human factor as well.

We establish our solution on the assumption that the severity of an intrusion for a particular virtual machine depends on a number of factors including; security requirements of the application hosted by the virtual machine, the state of any Service Level Agreement (SLA) negotiated beforehand and, the frequency of attack on a security requirement. There can be other parameters, however, we hold that these are the most important of the factors and therefore restrict our research to these factors. We also believe that the severity problem can be treated as a special case of traditional classification problem as it essentially involves segregating intrusion trails into different categories. Finally, we

use both supervised and unsupervised learning algorithms to implement our proposed solution.

With respect to virtual machine specific security requirements, one option can be to render the security policy definition and management a responsibility of the virtual machine itself. This can be achieved by instrumenting a policy engine within each virtual machine, which will coordinate with detection and diagnosis modules in the privileged virtual machine. This approach is attractive due to the ease of implementation and simplicity of the resultant system. However, it breaks the isolation property as, in the event of a successful attack, an attacker can modify the security policies to facilitate its malicious objectives. Furthermore, a guest virtual machine needs to be trustworthy to be delegated such responsibility which is contradictory to our assumption that, all guest virtual machines are treated as compromised. Due to these limitations of this approach, we adopt an approach that guarantees isolation while ensuring customization with respect to security policies. We propose using service level agreements to negotiate security requirements for a virtual machine. Following this approach, a customer is envisaged to specify the security requirements as part of the service level agreement at the resource acquisition phase. It requires quantification of security as explained in [4] and summarized in a following section.

With respect to SLA state, we designate the time remaining for completion of job as the SLA state. This is because of our understanding that severity of an intrusion is also affected by the time available for response. Ideally, the SLA state would be calculated by using different parameters such as; quality of service metrics and available resources etc. This requires establishment of complete monitoring infrastructure because of which we render this as out of the scope of our research. However, we assume that SLA state is available to us as an aggregate metric that can be used for formal analysis such as the one described in this document. Finally, the frequency of attack attempts on a particular

security requirement depicts either the value of the target or likelihood of success of attack attempt against the security requirement under attack. This therefore requires relatively immediate and more effective response mechanism to avoid recurrence of such attack attempts. For this reason, we designate the frequency of attacks on a security requirement as an important factor to dictate the severity of an intrusion.

As stated earlier, we propose to solve the severity problem by treating it as a classification problem. Related to this, a characteristic of supervised learning techniques is that they involve an initial training or learning phase to serve as a basis for online classification. However, with the problem focused in our research, no previous knowledge of severity of intrusions for applications is maintained which makes it difficult to use supervised learning techniques. Furthermore, most of the unsupervised learning techniques are more suitable for offline analysis as the classifications tend to change over the length of analysis datasets. This characteristic makes them inappropriate for systems that require real-time classification such as the one under consideration in our research. We therefore, decided to use both supervised and unsupervised classification techniques to achieve our objectives. We use an unsupervised classification technique i.e. K-means to prepare the training datasets for further analysis and use supervised classification technique i.e. Decision Trees for real-time severity analysis. The motivation of our choice for these techniques is explained in a later section. We describe both these learning techniques along with a formal description of our solution in the following sections.

3.1 Fault Model

The system proposed for our research resides in a system-level virtual machine (VM) under a Type1 Hypervisor [11]. The VM is a part of a platform virtualization based high performance computing infrastructure called a Cloud as defined earlier in this document. As the VM is a

system-level virtual machine, it is envisaged to emulate an independent node in traditional high performance computing infrastructures such as Compute Grids. The applications running on the VM and the processing being performed at the VM are assumed to be a part of a compute intensive job and will consequently have limited, presumably predefined, interactions with resources outside the VM such as interaction with a remote data sources. As cloud computing promotes the idea of on-demand computing whereby a customized VM is created for each resource request, a VM will generally have very limited number of users, for instance, members of a particular research project or even a single user for whom the VM has been created. The proposed system is envisaged to be a part of the local administrative components of a hypervisor in the domain 0 to achieve isolation from the host VMs. However, this limits the visibility of the system to the system calls executed by the VM being monitored. Due to this fact, we assume the hypervisor to be trustworthy whereas the virtual machines hosted by a hypervisor are assumed to be compromised.

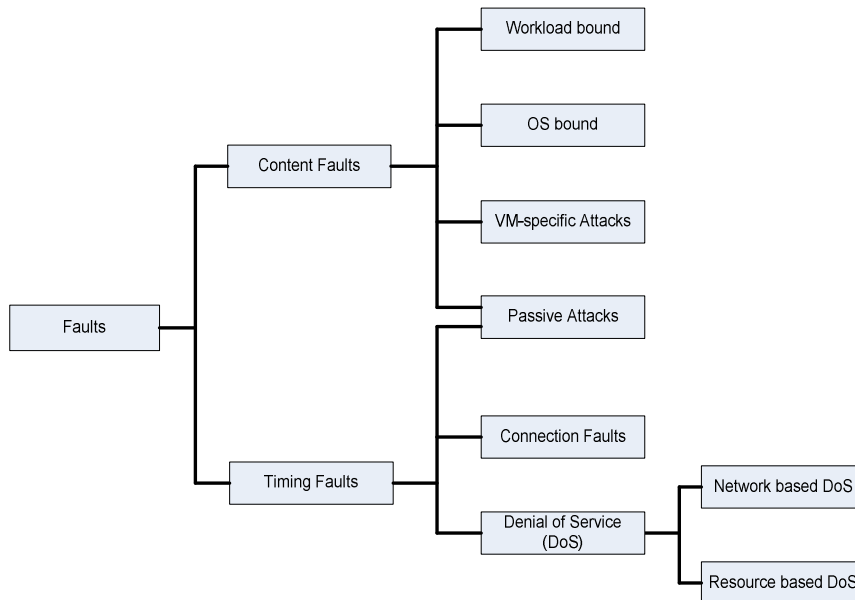


Figure2: The proposed fault model

The fault model for the proposed system consists of arbitrary intrusions that can occur in an intermittent fashion and can be categorized as external malicious interaction software faults, which can be sub-divided into content or timing faults as described by [1]. Figure 2 describes the fault model proposed for our research categorized into timing and content faults. The faults listed in the diagram have been identified as important from the perspective of a compute intensive workload being executed in a Cloud resource. Additionally, we only take into account the faults that can be mitigated at the system call level. As is evident from figure 2, the fault model excludes the faults that can occur at “site level” i.e. the Hypervisor or the Dom 0 being compromised is not included in the fault model for this research. The source of faults for our research is considered to be application-specific vulnerabilities that allow an intruder to compromise the VM and possibly use the resources allocated to the VM to accomplish malicious motives, and operational mistakes i.e. mistakes in configuration of VMs which can be exploited to enable a VM to infect a neighbor VM.

3.2 System Definition and Architecture

As described earlier, we use security requirements, service level agreement state and frequency of attack on a security requirement as parameters to evaluate the severity of an intrusion. The resultant metric of severity or impact factor is envisioned to be used by an intrusion response system to trigger appropriate response mechanism towards achieving end-to-end operation. We present the formal description of our system below followed by the architecture representation.

Let $C = \{s1, s2, s3, \dots, sm\}$ be a set of system calls where $m = |C|$ is the number of system calls. Data set D can be defined as a set of labeled sequences $\{ \langle Z_i, R_i \rangle \mid Z_i \in C^*, R_i \in \{1,2,3,4,5\} \}$ where Z_i

is an input sequence and R_i is a corresponding class label denoting one of the five possible severity levels i.e. “Minimal”, “Medium” “Serious”, “Critical” and “Urgent”. Given the data set D , the goal of the learning algorithm is to find a classifier $h: \mathcal{C}^* \rightarrow \{1,2,3,4,5\}$ that maximizes given criteria.

As part of the resource acquisition phase, security requirements are collected from the user along with other resource requirements. These security requirements are then used to formulate the diagnosis policy which lists the priorities of the different security requirements for a particular virtual machine. Let P_{ds} be such a diagnosis policy for a VM. It can be described as $P_{ds} = \{P_{dsi}, P_{dsi+1}, \dots, P_{dsn}\}$ where P_{dsi} represents a particular policy statement in the diagnosis policy.

Also, each system call SC_i can be mapped to one of the three attributes of security i.e. if A_{type} is a set of the three security attributes i.e. $A_{type} = \{\text{Availability, Confidentiality, Integrity}\}$, then $SC_i \in A_{type}$ where SC_i represents the type of attack attempted by the system call SC_i with respect to the three attributes of security. Furthermore, given the granularity of our data to be system calls, a system call can be mapped to one or more security requirements and vice versa. This requires mapping to be performed between the system calls and the security requirements. Let a security requirement be represented by set R described as $R = \{R_1, R_{i+1}, \dots, R_n\}$ and let $R_{i,type}$ denote the type of security attribute represented by R_i . Now, if DnS_i represents a diagnosis signature with respective diagnosis policy P_{dsi} and the attack type attempted by current system call is denoted by SC_i , then we can say that

$$SC_i \subset R_{i,type}$$

Therefore, if R_i denotes the requirement affected by SC_i , we can write the mapping function as below. We also describe the results of this mapping in a later section.

$$R_i = f(R_{type}, SC_i, P_{dai})$$

Finally, if S_t denotes the SLA state, $freq$ represents the frequency of the attack on the security requirement and Pr_i denotes the priority of the affected security requirement for the VM, we can write the impact factor or the level of severity as:

Level of Severity = $f(R_i, Pr_i, S_t, freq, DnS_i)$ where DnS_i is the diagnosis signature and is defined as $DnS_i = \{DS_i, D_i, TIMESTAMP\}$ where DS_i represents the detection signature i.e. the information passed on from the detection engine to the diagnosis module. This information is implementation dependent and is envisaged to vary across different implementations, however, we assume that it includes at least SC_i , VMID and Detection decision.

As described earlier, the result of this activity is an aggregate metric, which is envisioned to be used in cooperation with an intrusion response system to select an appropriate response to the intrusion under consideration.

We present the architectural representation of our system in Figure3. In this representation, we focus on $dom0$, the most privileged domain of a virtualized resource. As described in the diagram, we envisage our system to be established in $dom0$ to achieve maximum isolation from the monitored hosts. The system is also envisaged to interact with other components both within the virtualized resource and also with the global administrative components such as a resource manager. We have performed rigorous evaluation of the presented architecture using Architectural Trade-off Analysis Method (ATAM) [27], the

details of this evaluation are not presented here to preserve the focus of this chapter.

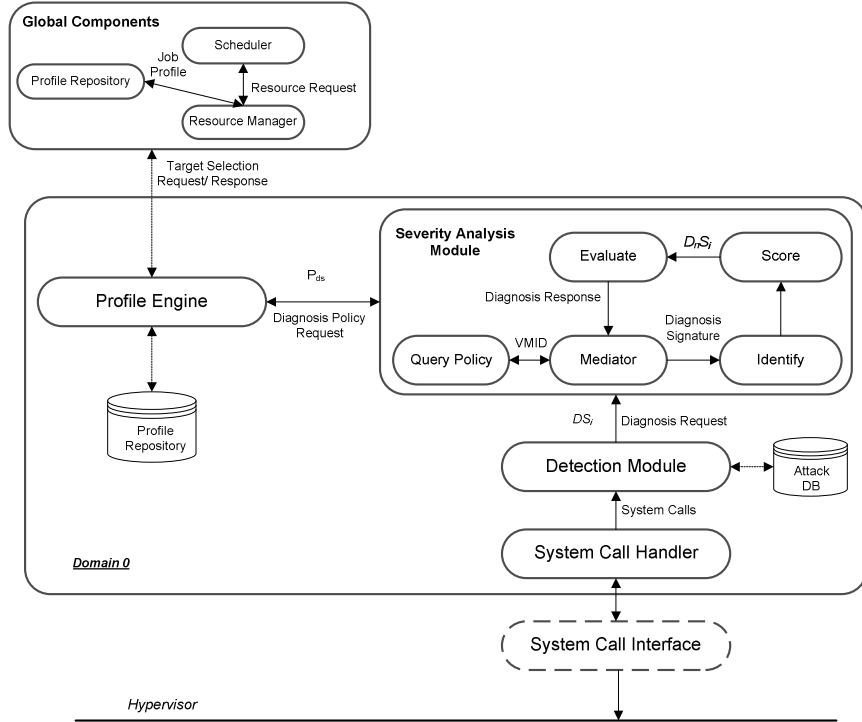


Figure 3: Architectural representation of the system

3.3 Quantification of Security

We hold that the severity of a security attack for a particular virtual machine depends upon different attributes including the security characteristics of the virtual machine. One approach to capture security characteristics of virtual machines can be heuristic method whereby applications hosted in virtual machines can be assumed to have certain security attributes. These assumptions are usually based on the experience of security expert with applications under consideration. However, this approach is limited with respect to identifying security

attributes for the substantially diverse nature of candidate applications for virtual machine based systems in general and clouds in particular. An alternate approach is to encourage the users of virtual machines to explicitly specify the corresponding security attributes. A feasible implementation of this approach is to enable the user to choose/prioritize security attributes from a set of given attributes. This enables both the user of the virtual machine and the governor of the cloud system to be synchronized regarding the security requirements of the virtual machine. We adopt this approach due to the degree of customization offered and the potentially comprehensive representation of a virtual machine's security characteristics. In order to ensure minimal human intervention and seamless end-to-end operation, we envisage gathering security characteristics of a virtual machine during the resource acquisition phase, as part of the service level agreement, in the form of security requirements. However, incorporation of security as part of a service level agreement requires its quantification so as to achieve an appropriate level of granularity to facilitate comprehensive representation of the security characteristics of a virtual machine. In this section, we summarize our efforts to quantify security into seven security requirements whereas detailed explanation and evaluation is described elsewhere [3,4].

The security requirements are described from the perspective of a workload running in a virtual machine. As a virtual machine can act as an independent node on the network, the hypervisor is not concerned about the message level security parameters such as; key size, encryption algorithm etc. Therefore, we limit our research to the metrics which can be monitored by a hypervisor using system calls. This has advantage of being agnostic of the applications running in a virtual machine. Furthermore, the security requirements described here can be defined as high-level primarily because each of the requirements can be further divided into more fine-grained metrics. However, it is intentional to group similar metrics under a common classification so as to minimize the complexity governed by the proposed

approach. Having said that, the security requirements described here are not meant to be exhaustive and only represent an effort as a proof of concept.

We have used the fault model described in the previous section as the basis for our study and have formulated seven security requirements. The preferences for these can be specified as “High”, “Medium” and “Low” by the user of a service as part of the resource request. A “low” preference for a certain security requirement, therefore, means that the impact of a successful attack breaching that security requirement is assumed to have low impact on the expected behaviour of the particular workload. For example, e-social science applications usually deal with confidential data processing where the confidentiality of data is rendered more important than on-time delivery of results [29]. In this case, the owner of an e-social science application may wish to designate a “low” or “medium” preference for denial of service attacks and a “high” preference for backdoor protection. In relation with our overall research objectives, we envisage to use these preferences to evaluate the level of severity of an intrusion for a particular workload.

Table 1: Proposed Security Requirements

Security Attributes	Requirements
<i>Integrity</i>	Workload State Integrity
	Guest OS Integrity
<i>Availability</i>	Zombie Protection
	Denial of Service Attacks
	Malicious Resource Exhaustion
	Platform Attacks
<i>Confidentiality</i>	Backdoor Protection

We also assume a resource provider to use these security requirements to possibly group its resources based on their capabilities to fulfil these requirements to a certain level. This can be achieved by the resource provider by using appropriate infrastructures or technologies to guarantee the committed security attributes. For example, if a resource provider has committed to provide assurance for protection against denial of service attacks, it is expected that appropriate mechanisms have been installed to protect against such attacks.

The security requirements described above are listed in Table 1 in accordance with the three attributes of security i.e. Integrity, Availability and Confidentiality as described by [1]. These are meant to be a subset of the faults covered by the fault model described in an earlier section and are envisaged to be specified as part of a resource request along with their priorities by the consumer of the service.

3.4 Classification

Classification is a popular machine learning technique regarded as the process of segregating different items of a dataset into multiple groups or classes based on their characteristics or traits. Given a particular dataset, the goal of a classifier is to build a model of class distribution in terms of the quantified characteristics of the constituent objects of the dataset. In more formal terms; let $Z = \{ (d_1, c_1), (d_2, c_2), \dots, (d_n, c_n) \}$ be a dataset where $d_i \in \mathcal{D}$ which represents the individual data items, and $c_i \in \mathcal{C}$ which represents the class to which the particular data item belongs. In this case, a classifier h is a function such that $\square : \mathcal{D} \rightarrow \mathcal{Y}$ i.e. it defines a mapping between a data item and its class based on some attributes.

In relation to this, the severity of a security attack on a system is also dependent upon certain attributes including the security requirements of the processes affected by the attack, the frequency of attack and the

time available for reaction. Also, the severity of a security attack is usually described in terms of different levels such as “*High*”, “*Medium*” and “*Low*” which can also be treated as different classes of the potential effects of a security attack. We use these generalizations as our motivation to use classification techniques to solve the problem of severity analysis for virtual machine based systems in general and clouds in particular.

With respect to learning and classification, there are two major categories of techniques i.e. supervised and unsupervised. As suggested by their names, supervised techniques require an initial training phase where the algorithm is trained using existing dataset with appropriate classification. The algorithm then uses this knowledge to perform real time classification on test data. Conversely, unsupervised techniques do not require any existing classification examples and usually use multiple runs to fine tune the classification patterns. K means, Expected Maximization (EM), Gaussian Mixture and Self-Organization Map (SOM) are some examples of unsupervised learning techniques whereas Decision Trees, Support Vector Machines and Regression Analysis are some example of supervised learning techniques. We now describe our selected unsupervised and supervised techniques i.e. K Means and Decision Trees respectively.

3.4.1 K Means Clustering

K-means clustering [15] is an algorithm to classify objects based on attributes/features into K number of classes. The classification is done by minimizing the sum of squares of distances between data and the corresponding cluster centroid. Formally, for a given dataset $S = \{s_1, s_2, s_3, \dots, s_n\}$, K-means classifies data items into k classes based on the attributes of data items, where $n < k > 1$. The objective of K-means is to minimize the distance between the items surrounding each centroid which is formally represented as:

$$J = \sum_{j=1}^k \sum_{i=1}^n \|x_i^{(j)} - c_j\|^2$$

where $\|x_i^{(j)} - c_j\|^2$ is a chosen distance measure between a data point $x_i^{(j)}$ and the cluster centre c_j , is an indicator of the distance of the n data points from their respective cluster centers. Our selection of K-means for our research was motivated by the simplicity of the technique and its ability to support more than two classes for classification purpose.

3.4.2 Decision Trees

Decision trees are a popular supervised classification technique that uses a series of questions or rules about the attributes of the class to classify datasets into classes. As characterized by their supervised nature, they require training datasets to establish a classifier which is then used for test data. Decision tree algorithms are based on the concept of *measures of impurity* or heterogeneity of the data i.e. the metrics which demonstrate the presence of heterogeneous groups or classes of data in a given dataset. These metrics can be one of *Entropy*, *Gini Index* or *Classification Error*. There are a number of different algorithms to implement decision trees, however, C4.5 [17] and its improved version C5.0 are the most popular ones. In both C4.5 and C5.0, *Information Gain* is used to decide the class of each data item which can be describe as under:

Given a set R of objects, and an attribute A ,

$$Gain(R,A) = Entropy(R) - \sum (|R_v|/|R|)Entropy(R_v)$$

where R_v is the subset of R that has the attribute value v , the sum \sum is over each attribute value of A , and $|R_v|$ is the number of elements in the set R_v

Another important supervised classification technique is Neural Networks. We have selected decision trees for our research because of the following reasons:

- i) Decision trees are simplistic as compared to neural networks and hence easier to manipulate
- ii) Neural networks historically require more training data as compared to decision trees.
- iii) A critical phase in constructing neural networks is to choose the number of neurons and layers. This process is complex and skewed towards the experience of the designer. Also, the construction is based on trial and error method rather than any formula.
- iv) Historically, decision trees are proven to be better classifiers of unknown data as compared to neural networks.
- v) An added benefit of using decision trees is the rule generation which can be used for subsequent rule-based modelling.

4. Evaluation

In order to evaluate our approach to predict severity of an attack for a particular virtual machine, we conducted several experiments based on experimental datasets. As we use both unsupervised and supervised learning techniques, we present our experiences with both preparing training datasets using K-means and building and testing classifier using decision trees. Furthermore, we assume a user of virtual machine to have maximum privileges for the virtual machine. Therefore, exploits targeted at exceeding privileges are out of scope of our research. For now, each system call is treated as a separate event and is classified independent of preceding or following system calls. Considering chains of system calls to detect attacks has been proposed by [27], however, we render it as an opportunity for future work. Fur-

thermore, as there will be a single diagnosis system per virtualized resource, we assume that all the system calls being analysed are executed by the same virtual machine.

4.1 Datasets

For experiments, we choose publicly available system call sequences from the University of New Mexico. Our choice of system call dataset is motivated by the fact that our proposed system is envisaged to be instrumented in the dom0 and the granularity of data available to our system is assumed to be system calls. The University of New Mexico (UNM) provides a number of system call data sets. Each data set corresponds to a specific attack or exploit. However, given the age of these datasets and the system used for their generation, they do not contain virtual machine specific exploits. This limits the coverage of our evaluation to the remaining six security requirements, however, as all of the traditional exploits still apply to virtual machine based systems, we do not envisage any major deviations in the results. Furthermore, the datasets are also limited in the sense that they only list the names of the system calls executed by a process without its parameters. This can have an effect on the mappings between the system calls and the security requirements. Finally, one of the assumptions of our research is that the diagnosis module only deals with system calls identified as malicious by an intrusion detection system. However, the datasets do not acknowledge this assumption and consequently limits the effectiveness of our results. We envisage rectifying these problems with more rigorous experiments and evaluations with real datasets facilitated by implementation of our proposed solution with one of the existing hypervisors.

As we assume a user of virtual machine to have maximum privileges for the virtual machine, exploits targeted at exceeding privileges are out of scope of our research. Also, security attacks which exploit network

vulnerabilities to take control of the victim are not applicable because proposed is not assumed to monitor network traffic. Consequently, The datasets we have used are synthetic FTP and INETD data. We divide our test dataset into two sub datasets without compromising the integrity. This is because our proposed system is meant to deal with individual system calls and does not take into account the contextual information contained in these datasets. Therefore, each system call is treated as a separate event and is classified independent of preceding or following system calls. Furthermore, as there will be a single diagnosis system per virtualized resource, we assume that all the system calls being analyzed are executed by the same virtual machine.

4.2 Data Cleansing and Harmonization

In order to perform experiments to evaluate our approach, the datasets needed to be cleansed to rectify the discrepancies described in the earlier section and mould them to match our requirements. The first step towards this process was the mapping between system calls and the security requirements with the objective to know which security requirements are affected by a particular system call. We used the classification of system calls proposed by REMUS [22] facilitating grouping of system calls based on threat levels. We only focus on system calls which pose threat levels 2 and 3 because threat level 1 is defined as full access to system which is one of the assumptions of our research, whereas threat level 4 is harmless. The mapping has been performed by manual analysis of each system call description and results are presented in Table 2. As is described by the table, there are system calls which can potentially affect more than one security requirements and therefore have been mapped against multiple system calls. Also, for Platform Attacks i.e. attacks which exploit characteristics of virtual machines, mapping is performed based on existing literature.

The second step in data preparation stage is to harmonize the frequencies of system calls. As described earlier, the available datasets contain both malicious and non-malicious system calls. The affect of this problem with data is aggravated by the fact that frequency of an attack on a security requirement is an important parameter in our analysis. Therefore, the initial experiments performed revealed the results being skewed towards patterns with higher frequency. We address this problem by normalizing the frequency attribute of data based on the standard score method with custom parameters as given below.

$$\gamma = 3 * \omega - 0.060 * \frac{\mu(\omega)}{3.77 * \sigma(\omega)}$$

where,

γ = normalized frequency

ω = original frequency

$\mu(\omega)$ = mean of original frequency and,

$\sigma(\omega)$ = standard deviation of original frequency

This normalization process limits the value of frequency in the range of 0 and 3 which is synchronous to the priorities of each security requirement. With regards to the service level agreement state, we envisage this to be an aggregate metric generated as a function of multiple attributes including quality of service attributes, resource availability and performance. However, this detailed analysis is out of scope of our research and therefore, we generate service level agreement state using the rand()function in MATLAB [16] to represent an aggregate metric.

The final step towards data preparation is to use unsupervised learning technique i.e. K-means [15] to predict the severity of each system call event. This is to facilitate the training phase of our supervised classification technique i.e. decision trees. We have used MATLAB to per-

form this step. As with any unsupervised learning technique, K-means requires multiple runs to achieve a mature classification. For our experiments, we performed various runs with different parameters and a stable classification is determined by manually verifying the classes assigned to randomly chosen events. The final configuration used to achieve a stable output is given below.

$$(\delta, \alpha) = \text{kmeans}(\rho, k, \theta)$$

where, δ represents the predicted severity for a particular system call event, α represents the centroids calculated for the given data, ρ represents the dataset being analyzed which in our case is a matrix prepared using the steps described earlier, k represents the number of classes or clusters to be generated which in our case is 5 and, θ represents the number of replications performed to get the output which in our case was 100. As k-means uses distance to segregate different classes, the distance used in our case is the Euclidean Distance. We should emphasize here that, the data prepared as a result of this process is not envisaged to be 100% error free both because of the crudeness of the data and the simplicity of the approach.

4.3 Evaluation and Discussion

In order to evaluate our proposed approach, we used decision trees as a supervised classification technique for experiments on the data prepared as a result of the process described in the previous section. There are a number of algorithms to perform decision tree analysis, however, we have used C5.0 [21] because of its proven efficiency in different applications. We use See5 software [21] to perform our experiments with the C5.0 algorithm. Given the training data, we first constructed different classifiers i.e. decision trees by using different permutations of the attributes involved and compared the predicted classes against the

classes assigned during the data preparation stage. This process reveals different error rates for different classifiers. It also generates a tree classifier and a set of rules for each classifier. The rules generated as a result of this analysis can be used to model rule-based systems. Figure 4 presents an example of such rules whereas Figure 5 presents an example decision tree generated by C5.0. As shown in the decision tree, the decision nodes represent the attributes i.e. security requirements whereas the boxes present the predicted severity classes. As described earlier, we use five classes or levels of severity proposed by [5] i.e. *Urgent*, *Critical*, *Serious*, *Medium* and *Minimal*.

Table 2: Mappings between system calls and security requirements

Workload State Integrity	<i>Truncate, ftruncate, dup2, flock, ioctl, write, close, lseek, fcntl, umask, select, _llseek, _newselect, writev, poll, pwrite, mprotect, msync, mmap, munmap, mremap, signal, setpgid, uselib, sigreturn,</i>
Guest OS Integrity	<i>rmdir,ioctl,truncate,ftruncate,brk,delete_module,write,close,umask,setpgid,uselib,capset,</i>
Zombie Protection	<i>nfsservctl,ioperm,iopl,socketcall</i>
DoS	<i>Umount,mkdir,rmdir,umount2,ioctl,nfsservctl, truncate, ftruncate, quotactl, dup, dup2, flock,fork,kill,iopl,reboot,ioperm,clone,modify_ldt,adjtimex,vhangup, vm86, delete_module, stime, settimeofday, socketcall, sethostname, syslog, setdomainname, _sysctl,exit,ptrace</i>
Malicious Resource Exhaustion	<i>creat,fork,flock,setrlimit,setpriority,clone,sched_setparam,vfork,sched_setscheduler,swapon,swapoff,mlock,mlockall,nice,ipc,mlock,mlockall,sigsuspend,pause,waitpid,wait4,sched_yield</i>
Platform Attacks	<i>Ptrace</i>
Backdoor Protection	<i>Nfsservctl,dup,dup2,flock,ioperm,iopl,socketcall,read,readv,fcntl,select,fsync,poll,pread,sendfile</i>

We use these classifiers on our tests datasets to evaluate accuracy of each classifier against unseen data. We have compiled the results of this evaluation in Table 3. As part of the prediction of class for unseen

or test data, C5.0 also generates a confidence metric called Confidence, with a value of between 0-1, which facilitates rigorous analysis by revealing the assurance of the predicted value. Analysis of predicted values revealed that there are predictions with low confidence value which should be adjusted. In accordance with this, we consider a prediction to replace the original value if the confidence is more than 80%. This helps us remove inconsistencies where the confidence value is low. As shown in the table, error rate for classifiers differ with and without considering confidence metric which helps us improve our understanding of the classifiers.

```

Rule 1: (53, lift 7.4)
  req1 = 0
  req4 = 0
  -- class1 (0.902)
Rule 2: (88, lift 4.5)
  req1 = 0
  req4 = 0
  -- class2 (0.909)
Rule 3: (159, lift 2.5)
  req1 = 0
  req4 = 0
  req5 = 0
  -- class3 (0.994)
Rule 4: (81, lift 4.9)
  req4 = 0
  req5 = 0
  -- class4 (0.980)
Rule 5: (19, lift 20.1)
  req1 = 0
  req4 = 0
  -- class5 (0.952)

```

Figure 4: An example decision tree by C5.0

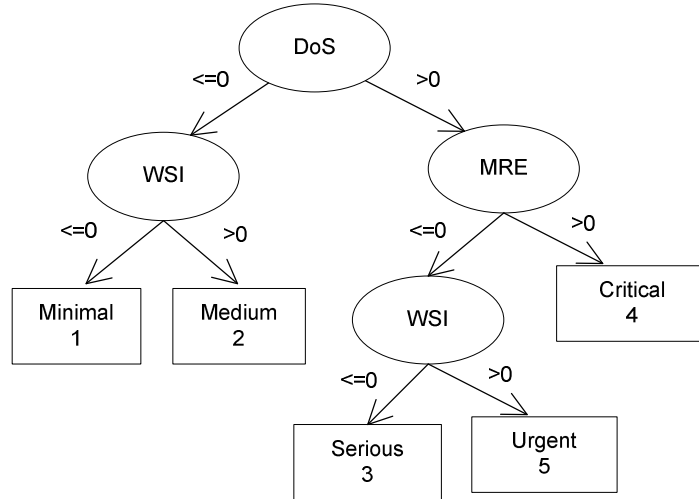


Figure 5: Rules generated for decision tree

An interesting observation from our experiments is the unexpected behavior of classifiers to test data. Table 3 presents the results of our experiments with the training and the two test datasets. It lists the error rates for each classifier when evaluated against test datasets both with and without consideration of the confidence value of the prediction. As can be seen in the table, the most consistent classifier of all is the one which gives maximum error rate for the training data i.e. 25.6%. The decision tree for this classifier has not been presented here because of limited space. However, the classifier with 0% error rate for the training data presents a high error rate of 77.1% for both analyses, with and without confidence metric. As can be seen, the results of the best performing classifier are not ideal and, to our best understanding this is largely due to the quality of data we used for our experiments. As explained in the earlier sections, we have made efforts to normalize and clean data using different techniques, however, discrepancies are still assumed to be present in the data. This limits our ability to perform a rigorous evaluation of our approach. Given this unsatisfactory quality of data, we are motivated to render these results as encouraging to carry out further research with better data. As stated earlier in this text, ef-

forts are underway to implement the proposed system with one of the existing hypervisors. This will enable us to address the problem of data quality and cleansing and facilitate a more rigorous evaluation of our approach.

Table 3: Evaluation of classifiers with training and test data

Classifier	Training Data	Test Data1		Test Data2	
		Without confidence	With confidence	Without confidence	With confidence
1	0.0%	77.1%	77.1%	79.8%	79.8%
2	0.2%	77.1%	52.5%	79.8%	73.6%
3	4.2%	90.9%	38.7%	87.9%	39.14%
4	13.4%	56.2%	31.6%	57.8%	6.2%
5	25.6%	97.9%	24.5%	98.5%	6.2%
6	27.7%	77.1%	77.1%	79.8%	79.8%

On the contrary, based on our experience with the use of machine learning techniques to problems such as intrusion diagnosis, we hold that machine learning techniques have promising prospects in this respect. Although our experiments involved rather simple machine learning algorithms, however, the results produced demonstrate the effectiveness of the approach even with imperfect data. Furthermore, the results also highlight the need to address the Data Problem for computer security research such as the one described in this chapter.

5. Conclusions and Future Work

We hold that security underpins extensive adoption of cloud computing. However, there are novel challenges which need to be addressed in order to improve the security of such infrastructures. The focus of our research is to investigate issues introduced by the unique characteristics of virtual machine. In this chapter, we have presented our efforts to identify and address one such problem- evaluating the level of attack severity for virtual machines on a same physical machine. To

the best of our knowledge, we are the first to identify and propose a solution to this problem for clouds. As described in the chapter, we propose using machine learning technologies facilitated by service level agreements to achieve our objectives. The evaluation conducted as part of our research reveals challenges and opportunities for further research in this domain. In particular, we highlight the *Data Problem* i.e. the unavailability of appropriate datasets to conduct effective research. Although the results of our research were marred by lack of appropriate data, we render them encouraging. Our efforts also demonstrate the challenges involved in optimizing machine learning techniques and highlight opportunities for their use in related research.

We intend to extend our research to implementation with real cloud environments. In particular, we plan to contribute our research to improve the dependability of iVIC cloud infrastructure [12]. In relation to this, efforts are underway to implement our system with Xen [10] hypervisor. Furthermore, as has been identified during evaluation, we intend to consider chains of system calls for our analysis as well. This will improve the application of our proposed solution to a wider domain.

References

- [1] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell and Carl Landwehr: *Basic Concepts and Taxonomy of Dependable and Secure Computing*, IEEE Transaction on Dependable And Secure Computing, Vol. 1, No. 1, January-March 2004
- [2] *Amazon Elastic Computing Cloud* Available at: aws.amazon.com/ec2
- [3] Junaid Arshad, *Integrated Intrusion Detection and Diagnosis for Clouds*. In the proceedings of Dependable Systems and Networks (DSN), Student Forum 2009
- [4] Junaid Arshad, Paul Townend, *Quantification of Security for Compute Intensive workloads in Clouds*. Submitted to International Conference on Parallel and Distributed Systems (ICPADS) 2009.
- [5] Severity Levels: http://www.internetbankingaudits.com/severity_levels.htm
- [6] Tal Garfinkel, Mendel Rosenblum: *When Virtual is Harder than Real: Security Challenges in Virtual Machine Based Computing Environments* In the Proceedings of 10th Workshop on Hot Topics in Operating Systems, 2005 - usenix.org
- [7] GoGrid: *Scalable Load-Balanced Windows and Linux Cloud-Server Hosting*. Available at: <http://www.gogrid.com/>

- [8] Google Cloud. Available at: www.googlecloud.com
- [9] Goldberg, R.P.: A survey of virtual machine research. *IEEE Computer*. 7, 34–45 (1974)
- [10] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, Andrew Warfield; *Xen and the Art of Virtualization* in the Proceedings of *SOSP'03*, October 19.22, 2003
- [11] IBM Systems, Virtualization version 2, release 1 available at: publib.boulder.ibm.com/infocenter/eserver/v1r2/topic/eicay/eicay.pdf
- [12] Jinpeng Huai, Qin Li, Chunming Hu; *CIVIC: A Hypervisor based Computing Environment* in the Proceedings of the 2007 International Conference on Parallel Processing Workshops
- [13] Nimbus. Available at: www.workspace.globus.org
- [14] OpenNebula Project. <http://www.opennebula.org>
- [15] J. MacQueen. Some methods for classification and analysis of multivariate observations, volume 1 of *Proceedings of the Fifth Berkeley Symposium on Mathematical statistics and probability*, pages 281-297, Berkeley, 1967. University of California Press.
- [16] The MathWorks - MATLAB and Simulink for Technical Computing. <http://www.mathworks.com>
- [17] Quinlan, J. R. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, 1993
- [18] I. Foster, and C. Kesselman; *The Anatomy of the Grid: Enabling Scalable Virtual Organizations*, *International Journal of High Performance Computing Applications*. Vol 15, p. 200-222, 2001
- [19] A. Weiss: *Computing in the Clouds*, *netWorker*, Volume 11, Issue 4, Pg 16-25, Dec. 2007
- [20] Ravi Subramaniam; *The Siamese Twins of IT Infrastructure: Grid and Virtualization*, *Open Grid Forum* 2007.
- [21] Information on See5/C5.0 www.rulequest.com/see5-info.html
- [22] Massimo Bernaschi, Emnuale Gabrieli, Luigi V. Mancini; *Remus: a security-enhanced operating system* in the proceedings of *ACM Transactions on Information and System Security* 2002.
- [23] Phillip A. Porras, Martin W. Fong, and Alfonso Valdes *A Mission-Impact-Based Approach to INFOSEC Alarm Correlation* in the Proceedings of *RAID 2002*: 95-114.
- [24] Stephen Northcutt and Judy Novak; *Network Intrusion Detection: An Analyst's Handbook*, 3rd edition New Riders Publishing Thousand Oaks, CA, USA ISBN:0735712654
- [25] Peter Mell and Karen Scarfone *A Complete Guide to the Common Vulnerability Scoring System* Version 2.0 www.first.org/cvss/cvss-guide.html
- [26] Burchard, L., M. Hovestadt, O. Kao, A. Keller, and B. Linnert: *The Virtual Resource Manager: An Architecture for SLA-aware Resource Management*, in the *IEEE International Symposium on Cluster Computing and the Grid*. 2004. p. 126-133.
- [27] Rick Kazman, Mark Klein, Mario Barbacci, Tom Longstaff, Howard Lipson and Jeromy Carriere *The Architecture Tradeoff Analysis Method* Technical Report, CMU/SEI-98-TR-008 ESC-TR-98-008 available at <http://www.pst.ifi.lmu.de/lehre/WS0102/architektur/VL9/ATAM.pdf>
- [28] Community Emergency Response Team <http://www.cert.org>
- [29] Wei Jie, Junaid Arshad, Richard Sinnott and Paul Townend; *Towards Shibboleth based Security for Grids - A State-of-art Review on Grid Authentication and Authorization Technology*. Accepted for *ACM Computing Surveys*. Association for Computing Machinery 2009.
- [30] R. Kazman, G. Abowd, and M. Webb. SAAM: A Method for Analyzing the Properties of Software Architectures. In the Proceedings on 16th International Conference on Software Engineering, pp. 81-90, 1994.
- [31] P. Bengtsson, N. Lassing, J. Bosch, and H. V. Vliet. Architecture-Level Modifiability Analysis. *Journal of Systems and Software*, vol. 69, 2004.

- [32] John D. Strunk, Garth R. Goodson, Adam G. Pennington, Craig A. N. Soules, Gregory R. Ganger. Intrusion detection, diagnosis, and recovery with self-securing storage. Technical report CMU-CS-02-140. May 2002.