

A Novel Intrusion Severity Analysis Approach for Clouds

Junaid Arshad, Paul Townend, Jie Xu
School of Computing
University of Leeds,
Leeds, UK. LS2 9JT
Corresponding Author: Junaid Arshad
sc06ja@leeds.ac.uk
Tel: +44 113 343 1707
Fax: +44 113 343 5468

Abstract

Cloud computing presents exciting opportunities to foster research for scientific communities; virtual machine technology has a profound role in this. Among other benefits, virtual machine technology enables Clouds to offer large scale and flexible computing infrastructures that are available on demand to address the diverse requirements of scientific research. However, Clouds introduce novel security challenges which need to be addressed to facilitate widespread adoption. This paper is focused on one such challenge - intrusion severity analysis. In particular, we highlight the significance of intrusion severity analysis for the overall security of Clouds. Additionally, we present a context-aware method to address this challenge in accordance with the specific requirements of Clouds for intrusion severity analysis. We also present rigorous evaluation performed to assess the effectiveness and feasibility of the proposed method to address this challenge for Clouds.

Keywords: Cloud computing, Cloud security, Intrusion Severity Analysis, Machine Learning.

1. Introduction

The advent of internet technologies such as; Service Oriented Architectures (SOAs) has significantly influenced the methods used in e-Science along with the emergence of new computing paradigms to facilitate e-Science research. Cloud computing is one such emerging paradigm which makes use of contemporary virtual machine technology. This collaboration between internet and virtual machine technologies enables Cloud computing to emerge as a paradigm with promising prospects to facilitate development of large scale, flexible computing infrastructures that are available on-demand to meet the computational requirements of e-Science applications. Related to this, Cloud computing has witnessed widespread acceptance mainly due to compelling characteristics such as; Live Migration, Isolation, Customization and Portability thereby increasing the significance of such infrastructures. The virtual machine technology has had profound role in it. Amazon [1], Google [2] and GoGrid [3] represent some of commercial Cloud computing initiatives whereas Nimbus [4] and OpenNebula [5] represent academic efforts to establish a Cloud.

Cloud computing has been defined in a number of different ways by different sources however, for the purpose of the research described in this paper, Clouds have been defined 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 Cloud computing system representing the above definition has been presented in Figure 1. 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 paper, the Cloud platforms focused at satisfying computation requirements of compute intensive workloads have been defined as *Compute Clouds* whereas those facilitating large scale data storage as *Storage or Data Clouds*. For the rest of this paper, the terms *Cloud computing* and *Clouds* have been used interchangeably to refer to our definition of compute Clouds.

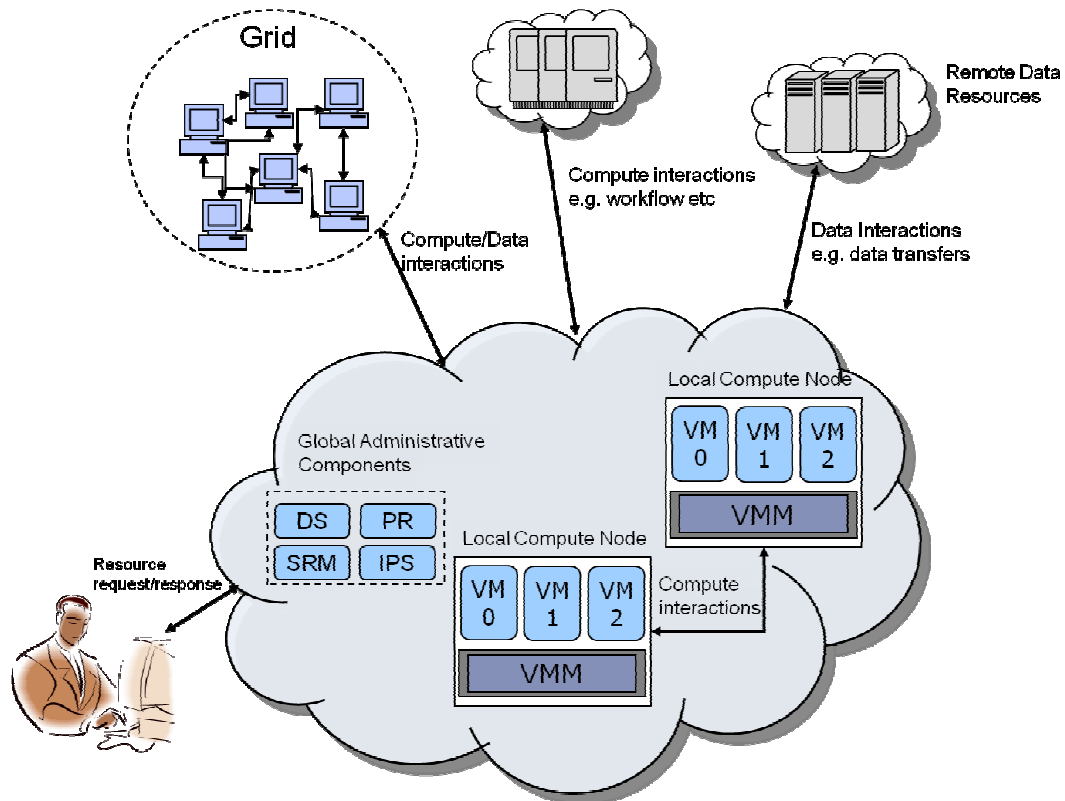


Figure 1: A Cloud computing system

As with any other emerging paradigm, different models of Cloud computing have been proposed to harvest its benefits. These are *Infrastructure as a Service (IaaS)*, *Software as a Service (SaaS)* and *Platform as a Service (PaaS)* [7]. With regards to these models, the Cloud computing system defined earlier and illustrated in Figure 1 resembles IaaS and therefore, inherits the characteristics of this model of Clouds. From the definition of Cloud computing presented earlier, the term Cloud computing has been used to refer to IaaS model of Cloud computing for the rest of this paper.

However, security underpins the extensive adoption of Cloud computing. Related to this, virtualized computing systems such as Clouds, introduce novel challenges for the development of enabling mechanisms in general and security in particular which require dedicated efforts for their solution [8]. The emphasis of this research is to investigate security issues due to the ability of virtualization to host multiple different computing environments on a single physical resource. Therefore, an intrusion detection and severity analysis system, residing in the most privileged domain, is required to monitor multiple virtual machines

having diverse security requirements. Depending on the security requirements of a virtual machine, a particular malicious attempt can have different degree of impact on different virtual machines. Within the context of this research, this has been defined as the *Level of Severity (LoS)* of a security breach for a particular application. There can be different implications of this concept such as the invoking of an appropriate recovery mechanism based on the severity of a particular attack. To the best of our knowledge, this is *the first effort to identify the significance and application of this concept* for Cloud computing.

The efforts described in this paper are a part of the system described in [9] and are focused at establishing a severity analysis scheme as part of the overall architecture. As part of the methodology to address the severity analysis problem, we hold that the severity problem can be treated as a special case of the traditional classification problem as it essentially involves segregating intrusion trails into different categories based on different parameters such as the security characteristics of the victim machine. Furthermore, machine learning based classification techniques have been used to perform the intrusion severity analysis. The parameters used for this analysis include security requirements for guest virtual machines along with Service Level Agreement (SLA) state and the frequency of attack on a particular security requirement.

This paper is organised as follows. The section 2 describes the problem and the state-of-the-art related to it. This is followed by a description of system and fault models for the proposed system in sections 3 and 4 respectively. Section 5 describes the proposed methodology to address the intrusion severity problem. This is followed by two critical aspects of the proposed method i.e. classification techniques used as part of the proposed solution and the security quantification to achieve customized operation in sections 6 and 7 respectively. Section 8 presents a detailed explanation of the various aspects of evaluation followed by a mention of the conclusions and potential future work in section 9.

2. Problem Definition and Related Work

The research presented in this paper is considered to be related to severity analysis of intrusions in general and for Cloud computing in particular. Furthermore, similarities can be held with traditional network based systems as well. Therefore, the existing literature in these domains has been explored to draw a comparative analysis of the proposed approach with contemporary approaches.

With respect to Cloud computing, to the best of our knowledge, we are the first to identify the intrusion severity analysis problem for such systems. However, there have been efforts in traditional systems to address this problem. An obvious example of such systems is the Network Intrusion Detection Systems (NIDS) where an intrusion detection system is usually deployed at a border node to look after a whole network of computers. As part of the network, different sub-domains or clusters can have varying security requirements. Existing research suggests that the problem of potential varying impact of an intrusion for such systems is addressed by defining customized security policies for such groups of nodes. However, there are certain defining differences between such systems and the virtual machine based systems such as Clouds. Firstly, the policies for traditional network based systems tend to be static, largely due to the static nature of the monitored systems. This is because the groups of nodes tend to have stable security requirements which have been established overtime based on experience with such systems. However, with Clouds, the monitored virtual machines are added and removed dynamically. Furthermore, the security requirements of individual virtual machines are also envisaged to be diverse, aggravating the problem. Secondly, the security policies in traditional systems are designed and managed by a system security administrator, with some input from the users, who is responsible for the security of the whole system. This

human intervention can become the weak link to realize customization and on-demand operation offered by Clouds. Additionally, it also affects the intrusion response time thereby affecting the overall security of the system.

With respect to the use of intrusion severity to select optimal response, intrusion response systems [10] use different metrics to achieve this objective. [11] and [12] represent two such approaches which use a severity metric. However, this metric is assumed to be calculated by a human administrator through an offline analysis at the policy definition stage. In this case, severity is usually calculated based on the administrator's experience and other resources such as Community Emergency Response Team (CERT) [13].

However, [14] and [15] present more formal methods to evaluate the severity or impact of intrusions for different applications. These methods are envisaged to facilitate a human administrator. In [14], 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 Lethality are relative rather than absolute. Therefore, these require an in-depth knowledge of the system under attack, the attack itself, and the 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, manual analysis also deteriorates the response time for an intrusion. Common Vulnerability Scoring System (CVSS) [15] defines three metric groups and a formula to calculate the impact of a vulnerability. The objective of this method is to facilitate a system administrator to perform manual analysis so as to designate the 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 a 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. Therefore, it has been concluded that a more fine-grained analysis is required, facilitated by comprehensive representation of user requirements.

As compared to the approaches described above, the approach proposed in this paper is fundamentally different in that it is envisaged to form a part of an integrated intrusion detection and severity analysis system as explained in [9]. Due to this, our approach also takes into account the effects of this analysis on the intrusion response time. Therefore, an attempt has been made to minimize disruption in the normal execution of system as part of our approach. Furthermore, our approach is envisaged to incorporate user input via SLAs which enables our method to be virtual machine specific. As this user input is restricted to the resource acquisition phase, it is considered that severity analysis process does not require any human intervention, whereas, the above explained approaches are agnostic of this fact and therefore, are exposed to the limitations described earlier in this section. In addition to these approaches, there are other approaches for alert correlation. However, they are focused on probabilistic models for predicting the likelihood of a successful intrusion by aggregating alerts from different sources of intrusion detection [16].

3. System Model

In order to facilitate understanding, figure 2 presents a lower level model of the proposed system. As illustrated in this figure, the proposed system is envisaged to reside in the domain 0, the most privileged virtual machine, of a virtualized resource in a Cloud. Therefore, in a Cloud infrastructure, each virtualized resource is envisioned to implement this system as part of other local administrative modules in the domain 0. By developing the proposed system as part of domain 0, the system can assume maximum isolation from the monitored virtual machines [17]. Furthermore, the visibility of activities performed by monitored virtual machines is enhanced to the system call level which encapsulates all the activities performed by a guest virtual machine.

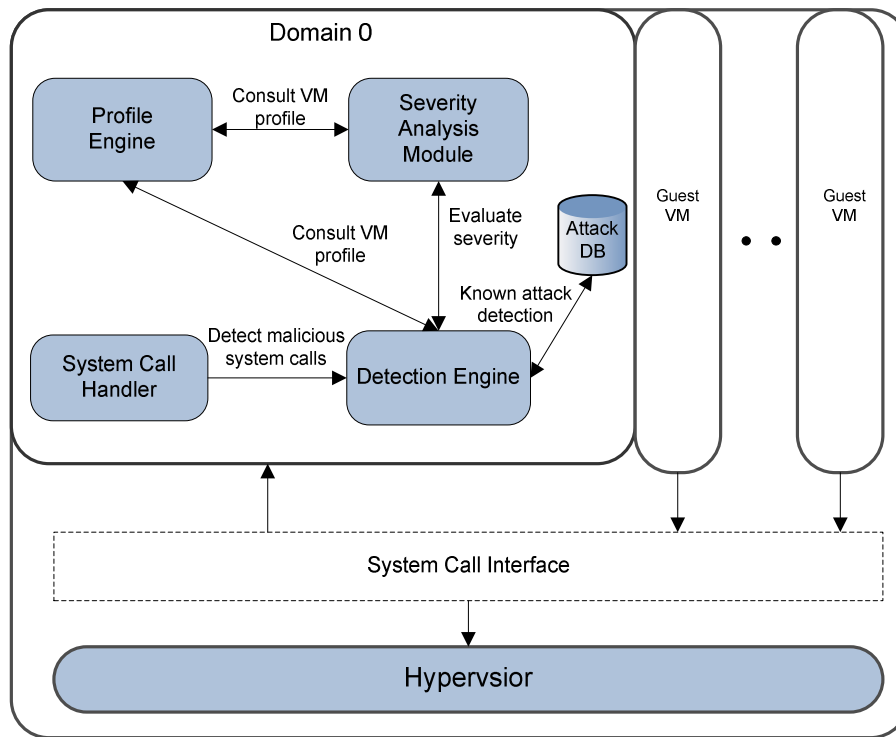


Figure 2: Model for the proposed system

The proposed system consists of the following components:

- A module to handle system calls executed by virtual machines - System Call Handler
- An intrusion detection system – Detection Engine
- A database of known attack signatures – Attack DB
- An intrusion severity analysis module – Severity Analysis Module
- A module to create and manage virtual machine profiles – Profile Engine

A System Call Handler (SCH) is envisaged to receive system calls executed by monitored virtual machines via the system call interface and transfers it to the Detection Engine (DE) to detect any possible malicious operations intended to be performed by the system call. The DE can be either an anomaly based or misuse intrusion detection system. In case of a misuse intrusion detection system, the DE can consult the Attack DB for signatures of known attacks. Otherwise, the DE can consult a Profile Engine (PE) to perform anomaly based intrusion detection. The PE is envisioned to create and manage security profiles for monitored virtual machines and can consult external components such as a global resource

manager for the cloud. These profiles include security characteristics of a virtual machine along with other attributes. In the event that a system call is identified as malicious by the DE, the system call is transferred to the Severity Analysis Module (SAM) to evaluate severity of the intrusion identified by the DE. As with the DE, the SAM can also consult with a PE to obtain virtual machine specific information such as security characteristics of the victim virtual machine. Finally, SAM is envisioned to evaluate the severity of an intrusion for a malicious event and communicate with an Intrusion Response System (IRS) to convey the result of severity analysis. The intrusion response process is, therefore, completed by the IRS by executing a response in accordance with the severity analysis.

4. Fault Model

The fault model for the proposed system consists of arbitrary intrusions that can occur in an intermittent fashion. Figure 3 presents the fault model for the proposed system whereas a description of the faults included in this model is presented below.

The faults included as part of this fault model have been identified as important from the perspective of a compute intensive workload in a cloud. Additionally, the proposed system has to deal with system calls from within the domain 0. Therefore, this fault model takes into account faults that can be mitigated at system call level from within the domain 0. Furthermore, the fault model excludes faults that can occur at *site level* i.e. the Hypervisor or the domain 0 being compromised is not included in the fault model for this research. The faults considered for the proposed system are external software faults alone and hardware is assumed to be error free. Furthermore, only malicious faults are considered a part of this fault model. The source of faults 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 neighbour VM.

As illustrated by figure 3, the faults can be categorized into *Timing* and *Content* faults. *Timing faults* are said to occur when a process or a service is not delivered within a specified time interval [18]. For the proposed system, timing faults generally represent Denial of Service (DoS) attack attempts which cause unavailability of the victim. Timing faults considered in this fault model are connection faults and denial of service attack attempts. *Connection faults* encompass malicious attempts to initiate illegitimate network connections with a malicious objective. These include opening a backdoor channel with the attacker's machine, opening network connection with attacker's machine to transport vital data and also to connect with arbitrary locations over the internet to use the victim as a zombie to launch a Distributed DoS (DDoS) attack. *DoS faults* represent malicious attempts to exhaust network or local resources [19]. DoS attack attempts at network resources intend to initiate enormous volume of illegitimate network connections at the victim virtual machine thereby consuming buffer memory. This could lead to a situation where the victim virtual machine runs out of memory and cannot host any more network connections. DoS attack attempts to exhaust local resource represent malicious attempts to consume all the available resources such as processor, memory and disk space. These attacks are explicitly targeted at exhausting local resources so as to deny the legitimate process to be executed.

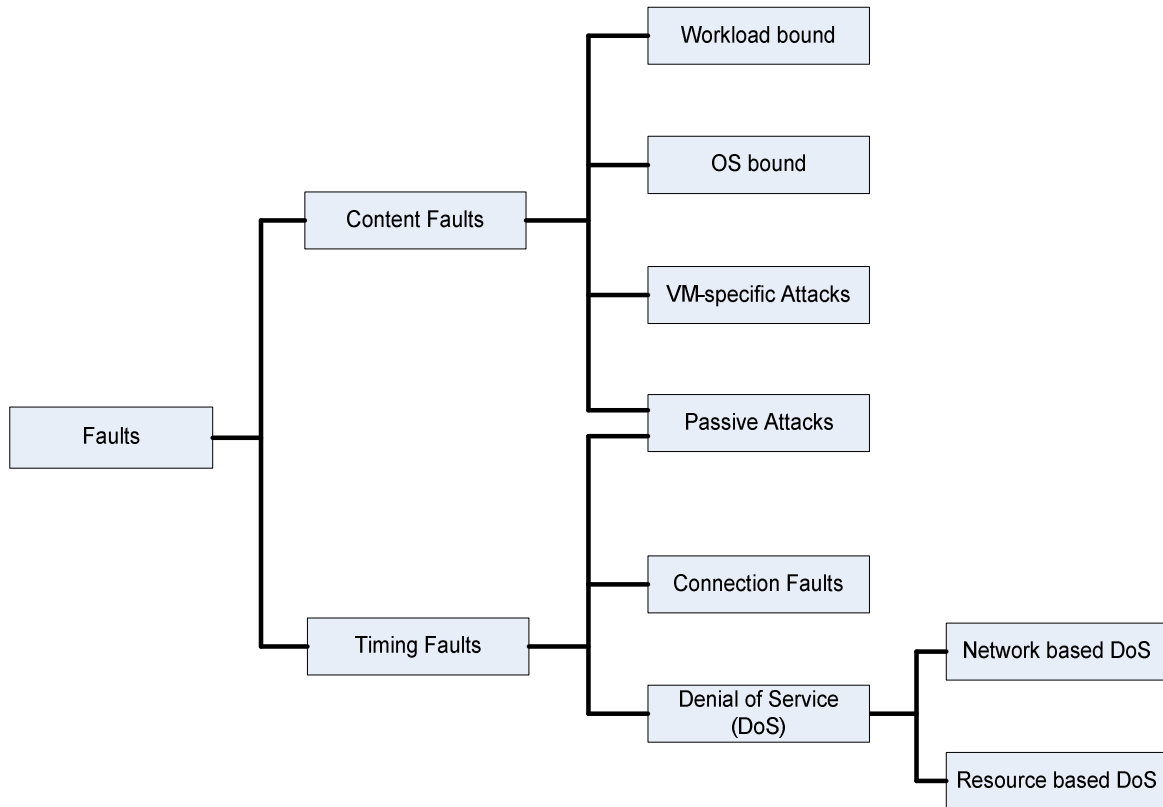


Figure 3: The proposed fault model

Content faults are considered to occur in the event of an undesired change to the information delivered or its state by a service. With regards to the proposed system, these encompass malicious attempts targeted to compromise the integrity and confidentiality of the data used by a workload. *Workload bound* content faults refer to malicious attempts to compromise the integrity of the data used and generated by the workload. *OS bound* content faults include malicious attempts targeted at compromising the sensitive data or memory locations for the operating system hosting the workload. These can include attacks such as *buffer overflow attacks*. *VM-specific attacks* incorporate malicious attempts specific to virtual machines. These include attack attempts to break the isolation between guest virtual machines and attempts to use virtualization for malicious purposes. *Passive attacks* include malicious attempts to sabotage a workload without being visible to normal system user. These include *sniffer* attacks which are only meant to record data of interest to the malicious user and, malicious backdoor channels. Among other possible threats, a backdoor channel can be used to control use of victim's resources and potential use of victim as zombie in a DDoS attack attempt.

5. Methodology

The proposed solution has been established 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 the SLA negotiated beforehand, and the frequency of attack on the victim machine. There can be other parameters, however, it is assumed that these are the most important factors.

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 establishing a policy engine within each virtual machine which will

coordinate with detection and severity analysis 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 malicious. Due to these limitations of this approach, an approach has been adopted which guarantees isolation while ensuring customization with respect to security policies. We propose to use SLAs 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 [20] and our efforts in this regard have been summarized in section 7.

With respect to the SLA state, the time remaining for completion of a workload has been designated 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. This requires establishment of complete monitoring infrastructure due to which it has been rendered as out of the scope of this research. However, it is assumed that SLA state is available as an aggregate metric which can be used for formal analysis such as the one described in this paper. 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 effective response mechanism to avoid recurrence of such attack attempts. For this reason, the frequency of attacks on a security requirement has been designated as an important factor to dictate the severity of an intrusion.

As stated earlier, it has been proposed to solve the severity problem by treating it as a classification problem. Within this context, machine learning techniques have been used to perform this classification. Related to this, the unsupervised learning techniques are usually more suitable for offline analysis as the classifications tend to change over the length of analysis datasets. This characteristic makes them inappropriate for systems which require real-time classification such as the one under consideration in this research. However, a characteristic of supervised learning techniques is that they involve an initial training or leaning phase to serve as a basis for real-time classification. However, with the problem focused in this research, no previous knowledge of severity of intrusions for virtual machines is maintained which makes it difficult to use supervised learning techniques. In order to mitigate this challenge, a dynamic weighted scheme was developed to facilitate training phase for the selected supervised classification technique i.e. decision trees. The motivation for the use of classification techniques has been described in the following section along with a description of decision trees classification technique.

6. Classification and Machine Learning

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 $d_i \in D$ which represents the individual data items, and $c_i \in C$ which represents the class to which the particular data item belongs. In this case, a classifier h is a function such that $h: D \rightarrow V$ i.e. it defines a mapping between a data item and its class based on some attributes.

6.1 Use of classification for intrusion severity analysis

Consider an application Z with certain security characteristics denoted by X . Now, the severity S of an intrusion I on the application Z is a function of the intrusion and the security characteristics of the victim application. This can be formally described as $S = f(I, X)$

Now, consider a dataset D containing record of intrusions on the victim application. Each element d_i in this dataset can have a different severity for the victim application; earlier defined as the level of severity of the intrusion for victim. Assuming a limited set of entries representing the possible levels of severity, the output for function S can be defined as under:

$$S = f(I, X) \{ : c \}; \text{ where } c \text{ is an entity in set } C \text{ representing possible levels of severity}$$

From the above definitions, severity analysis of an intrusion can result into one of the possible levels of severity. These levels of severity can be regarded as different classes whereby a class is distinguished by values of different attributes such as; intrusion and security characteristics of the victim application. Also, the severity of a security attack for traditional computing systems 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.

Furthermore, the severity of an intrusion depends on a number of factors described earlier. This increases the complexity of the process and therefore, makes it extremely difficult to evaluate severity for all known intrusions against all possible application types. It is non-trivial to perform such process for unknown intrusions. Therefore, a solution for intrusion severity problem should incorporate the ability to predict severity for unknown attack patterns. Machine learning techniques present an opportunity to address these issues efficiently and have been used to perform effective intrusion detection. All these factors provided the motivation to investigate the use of classification and machine learning techniques to address the intrusion severity analysis problem for Clouds.

6.2 Machine Learning

With respect to machine learning based classification, there are two major categories of such 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. Expected Maximization (EM) [23], Gaussian Mixture [24] and Self-Organization Map (SOM) [25] are some examples of unsupervised learning techniques whereas Decision Trees, Support Vector Machines [26] and Regression Analysis [27] are some example of supervised learning techniques. The selected supervised technique i.e. Decision Trees has been described below.

6.2.1 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 [28] 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 ,

$$Gain(R,A) = Entropy(R) - \sum ((|R_v|/|R|)Entropy(R_v)) \quad (4)$$

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 [29]. Decision trees have been selected for this research because of the following reasons:

- i) 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.
- ii) Neural networks historically require more training data as compared to decision trees.
- iii) Historically, decision trees are proven to be better classifiers of unknown data as compared to neural networks.
- iv) An added benefit of using decision trees is the rule generation which can be used for subsequent rule-based modelling.
- v) Decision trees are simplistic as compared to neural networks and hence easier to manipulate

7. Quantification of Security Requirements

As described in section 5, the severity of a security attack for a particular virtual machine is envisaged to depend upon different attributes including the security characteristics of the victim 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 administrator 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. This approach has been adopted 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, it is envisaged to gather 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, our efforts to quantify security into seven security requirements have been summarized whereas detailed explanation and evaluation is described in [9, and 20].

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, this research has been focused at 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 mean to be exhaustive and only represent an effort as a proof of concept.

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

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 [30]. 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 the overall research objectives, it is envisaged to use these preferences to evaluate the level of severity of an intrusion for a particular workload.

Furthermore, a resource provider is assumed to use these security requirements to possibly group its resources based on their capabilities to fulfil these requirements to 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 are 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 [31]. These are meant to be a subset of the faults covered by the fault model described in section 4 and are envisaged to be specified as part of a resource request along with their priorities by the consumer of the service.

8. Experiments and Evaluation

The overall evaluation of the proposed system comprised of i) experiments to evaluate the feasibility and effectiveness of the security quantification described in section 7 and ii) experiments to evaluate the intrusion severity analysis method described in this section. The

evaluation of security quantification has been explained in detail in [20] whereas the preliminary experimentation for intrusion severity analysis method has been explained in [32]. The overall objective of these experiments is to investigate the effectiveness and feasibility of machine learning techniques for intrusion severity analysis for Clouds. In this section, the advanced experimentation for proposed method has been explained along with their respective results and analysis

8.1 System call to security requirements mappings

Within the context of the system model described in section 3, the proposed system is envisaged to reside in the most privileged virtual machine or *domain 0* of a virtualized resource. This improves overall security of the system by isolating the proposed system from the monitored virtual machines. However, it limits the visibility of the monitoring system to the system calls executed by the monitored hosts. This is because the interaction between a guest virtual machine and the proposed system is performed using a system call interface as illustrated by figure 2. Therefore, the data available to the proposed system is the system calls executed by the virtual machines hosted on the virtualized resource.

The parameters proposed in section 5 to evaluate the severity of an intrusion include security requirements S_i , frequency of attack f , on a specific security requirement and the state of the SLA, S_t . Among these parameters, f and S_t are envisaged to be evaluated at real time and do not depend on the type or granularity of the monitored data. However, in order to identify the security requirement affected by a particular system call, mappings between system calls and security requirements are required.

Table 2: Mappings between system calls and security requirements

Security Requirements	System calls
Workload State Integrity	Truncate, ftruncate, dup2, flock, ioctl, write, close, lseek, fcntl, umask, select, _lseek, _newselect, writev, poll, pwrite, mprotect, msync, mmap, munmap, mremap, signal, setpgid, uselib, sigreturn,
Guest OS Integrity	rmdir, ioctl, truncate, ftruncate, brk, delete_module, write, close, umask, setpgid, uselib, capset,
Zombie Protection	nfsservctl, ioperm, iopl, socketcall
DoS	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
Malicious Resource Exhaustion	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
Platform Attacks	Ptrace
Backdoor Protection	Nfsservctl, dup, dup2, flock, ioperm, iopl, socketcall, read, readv, fcntl, select, fsync, poll, pread, sendfile

In order to create these mappings, each system call is treated as a separate event and is classified independent of preceding or following system calls. Therefore, the mappings are created considering each system call as an independent event. Related to this, considering chains of system calls for intrusion detection has also been proposed in existing literature such as [33, 34]. However, considering chains of system calls remains confined for intrusion

detection. This is because leading efforts for intrusion detection based on chains of system calls strive to identify a malicious system call from a sequence or chain of system calls [34]. For instance, consider a non-malicious sequence of system calls $\{open, read, mmap\}$ used to perform a write on a specific system file. Now, if $\{open, mmap, mmap\}$ represents a malicious sequence of system calls, an intrusion detection system which considers sequence of system calls will identify the later sequence as malicious and highlight *mmap* as the culprit system call. Therefore, a single system call will be transported to the intrusion severity evaluation module as described by figure 2. The mappings between system calls and security requirements have been presented by table 2

The mappings described in table 2 have been developed based on the classification of system calls proposed by REMUS [35] facilitating grouping of system calls based on threat levels. As part of this classification, REMUS proposed four different threat levels i.e. level 1, 2, 3 and 4. Here, threat level 1 represents intrusions focused at attaining complete control of the victim system. As one of the assumptions of the proposed system, the monitored virtual machines are assumed to be compromised. Therefore, this group of system calls is out of scope of this research. Furthermore, level 4 presents system calls which do not present any harmful threat to the system. Therefore, this group of system calls is also ignored for the purpose of this research. Due to these observations, the system calls considered in this research represent threat levels 2 and 3 as classified by REMUS. These system calls are focused at subverting the execution of a victim process within the target system.

The mappings described in table 2 have been performed by manual analysis of each system call description for Unix kernel. As can be seen from the mappings table, there are system calls which can potentially affect more than one security requirements and therefore have been mapped against multiple system calls. For instance, *socketcall* system call allows a process to create a new internet socket which is eventually used to create a network connection. From the perspective of this research, a network connection can be used to open a backdoor channel to the victim host after it has been compromised. Furthermore, this system call can also be used to create dormant or *half-open* [36] network connections at an exponential rate to achieve exhaustion of network connection buffers, resulting in a network based DoS attack. Therefore, *socketcall* has been mapped to both *DoS* and *Backdoor Protection* security requirements.

Additionally, the system call mappings also include virtual machine specific attacks. In this respect, *Platform Attacks* represents the security requirement which takes into account virtual machine specific attacks i.e. attacks which exploit characteristics of virtual machines. For these attacks, mapping is performed based on existing literature. Finally, the mappings presented in table 2 are envisaged to be static. This is due to the fact that the system calls for a specific kernel are not envisaged to change frequently, however, in the event of modifications to the system call table for a kernel these mappings do not present a hindrance with respect to maintenance.

8.2 Data preparation

In order to acquire data aligned with the system model and assumptions of the proposed method, implementation of various components of the detailed system model is required. However, due to complexity involved in this process and the limited time-span of our research, it was demand infeasible to establish the entire infrastructure. Therefore, alternate methods were required to generate datasets capable of representing the real datasets. In this respect, it was decided to generate data using a computer program, however, it should be emphasized here that every possible measure was taken to preserve the objectivity and impartiality of this process. The details of this process are described below.

As described earlier, each event for the proposed intrusion severity analysis module comprises of ten different variables i.e. the system call, seven security requirements, frequency of attack on the security requirements and the state of the SLA for the victim workload. All of these variables except the system call are used to evaluate the severity of the intrusion. However, the characteristic of a system call is represented in the data using the system call mappings described in the section 8.1. Now, a malicious event for the proposed system can be described as:

$$E_i = \{s_1, \dots, s_7, f, S_t\}$$

From the above representation, it can be concluded that each event represents a distinct point in a nine dimensional space where each of the nine variables represents a dimension of this space. Therefore, in order for a dataset to assume maximum coverage for the proposed system, it must contain all possible points of the nine dimensional space. Following this principle, the data contains all possible events which can occur in any real system. Therefore, if this data can be denoted by D_a and similar data for a real Cloud system can be denoted by D_r , the following can be derived

$$D_r \subseteq D_a$$

The above principle verifies the extensive coverage of the programmatically generated data with respect to the requirements of the proposed intrusion severity analysis approach.

Given the programmatically prepared data, devising an evaluation methodology for the advanced experiments was a non-trivial challenge. Although a number of different experimental settings can be carried out to achieve rigorous evaluation of the proposed method, a method is required to assess its effectiveness for the ISA problem. Commonly used strategies to perform evaluation in the related field of intrusion detection compare the results of proposed methods against the existing performance [33,34,35]. However, the challenge encountered in this research was the novelty of the research because of which no existing datasets could be found which can be used for evaluation. Additionally, it threatened to hamper the training of classifiers as the training required datasets with known severity for the events represented in the training data.

Due to these challenges, the only possible option left to the best of our understanding was to manually examine the results generated from the experiments. In a way, this process would resemble the current practice in real systems i.e. security administrators use expert judgement to decide the severity of a malicious event. However, as the number of malicious events in the datasets exceeds one million records, manual examination of a significant subset of this data was considered extremely difficult if not impossible. Therefore, it was decided to automate the reasoning process to facilitate the intended experimentation methodology described in section 5. Throughout this process, careful efforts were made to design the reasoning to avoid any prospect of biased evaluation.

The resultant process is a dynamic weighted scheme where weights are allocated based on the relative importance of each parameter. In particular, security requirements have been allocated the highest weight with SLA state as the second highest and frequency as the lowest relative weight. This is to ensure a customized evaluation based on the argument that the security requirements of an application are influential in deciding the impact of an intrusion on the victim application. Furthermore, the dynamic nature of weights facilitates adjustment of the parameters on severity evaluation. This is to comprehend with scenarios where there is no high prioritized security requirement under threat by a particular attack. The details of this scheme are presented elsewhere to preserve the focus of this paper.

8.3 Experimentation

The data prepared as a result of the process described in the section 8.2 has been used to conduct advanced experiments. The experiments were conducted on a general purpose workstation with Intel Core 2 Quad CPU 2.83GHz and 4.0 GB memory. The machine learning algorithm used for these experiments was C4.5 [28] however, the software used to perform these experiments was the Weka machine learning software [37]. Weka is a machine learning software platform which provides the ability to perform rigorous experiments using various machine learning techniques. The software has a very rich library with respect to machine learning techniques and provides variety of tools to support both graphical user interface and programmable Application Programming Interface (API).

For these experiments, random subsets of data produced as part of the process described in the section 8.3 were used. The two types of experiments performed are described below.

- *Cross Validation.* Cross validation is a statistical technique for evaluation of machine learning techniques with respect to prediction. As part of these experiments, 10-fold cross validation has been used. In 'x'-fold cross validation the data set is divided into x subsets of approximately equal size. One of the subsets is picked for testing and the rest subsets are used for training. In other words, a classifier is generated from 'x-1' subsets and the classifier is tested over the rest subset. This routine is applied for each of x different subsets, and then the results are averaged respectively over each of x different subsets tested. This is to ensure that no information used for classifier generation is reused as test data.
- *Combination of different datasets.* With cross validation, the results of a test run are compared against the training run using the same dataset. As part of these set of experiments, different combination of test and training data were used. Specifically, four different subsets of data were randomly selected from the artificial data. For each of these four subsets of data, one subset was used as training data to build a classifier. After the training phase, each classifier was tested against the remaining three subsets of data. This ensures that the test data is essentially unseen for the classifier and different from the training data. Furthermore, Weka provides the ability to specify different confidence values for each classifier. This value is similar to the confidence metric for See5 described as part of the preliminary experiments. Experiments with varying confidence values were also performed to ensure rigorous evaluation of the classifiers against the desired objectives.

The above described experiments provided interesting insights into the proposed method for intrusion severity for Clouds. The results of these experiments along with their evaluation have been described in the next section.

8.4 Results and Analysis

As described in the section 8.3, two types of experiments were conducted i.e. cross validation and combination of different datasets. In this section, the results of both these types of experiments have been presented along with their analysis.

8.4.1 Cross validation

For these experiments, the statistical technique of cross validation was used. With respect to 'x'-fold cross validation, the data set is divided into x subsets of approximately equal size. One of the subsets is picked for testing and the rest subsets are used for training. For these experiments, 10-fold cross validation was used with training data forming the 66% of each dataset. Therefore, for each dataset used in these experiments, 66% is used for training

whereas 34% is used for test purposes. Additionally, as this is a 10-fold cross validation, this process is repeated 10 times for each dataset.

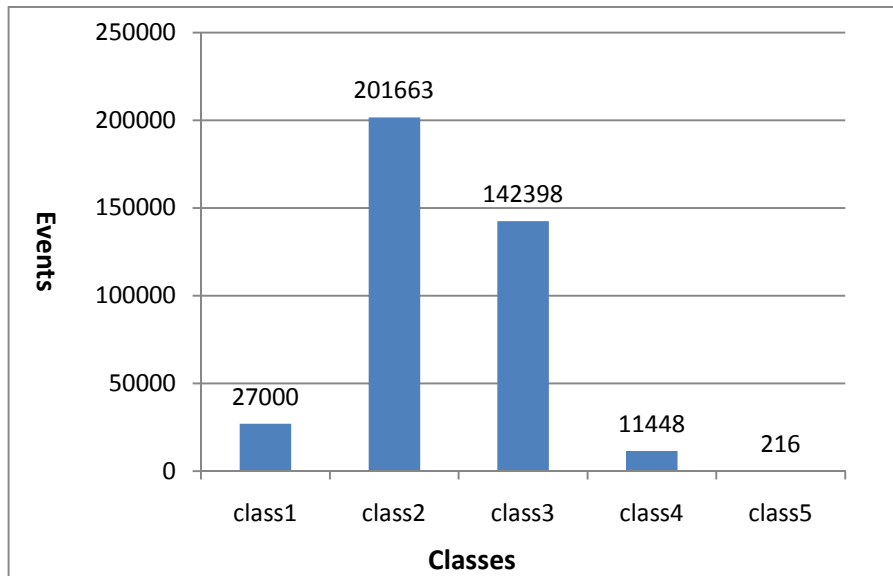


Figure 4: Class distribution for a dataset

For the experiments conducted in this research, the cross-validation experiments were conducted on four different randomly generated datasets as described in section 8.2. Figure 4 presents class distribution for one of the datasets used in these experiments. Furthermore, all these experiments were repeated four times and results were averaged to ensure the repeatability of the experiments. Each of these runs of experiments is regarded as a *round* throughout the rest of this paper.

Table 3 presents the percentage success rates for 10-fold cross validation for all sixteen randomly generated datasets along with the averaged results at the bottom. Additionally, figure 5 presents a graph with success rates for all sixteen datasets whereas figure 6 presents the success rates of cross validation for a single round. As can be seen from the statistics presented in these figures, the lowest success rate for these experiments is 99.6% which is very encouraging. However, the lowest average success rate is 99.85% which again represents a very healthy success rates. An example decision tree from these experiments has been presented in figure 9. Due to the number of datasets used and the classifiers created during this evaluation process, the decision trees for all the classifiers have not been presented here to conserve space.

Table 3: Percentage success rate for cross validation

	Round 1	Round 2	Round 3	Round 4
Dataset 1	99.793	99.877	99.917	99.812
Dataset 2	99.607	100	99.938	99.963
Dataset 3	100	100	99.921	100
Dataset 4	100	99.807	99.781	100
Average	99.850	99.921	99.889	99.944

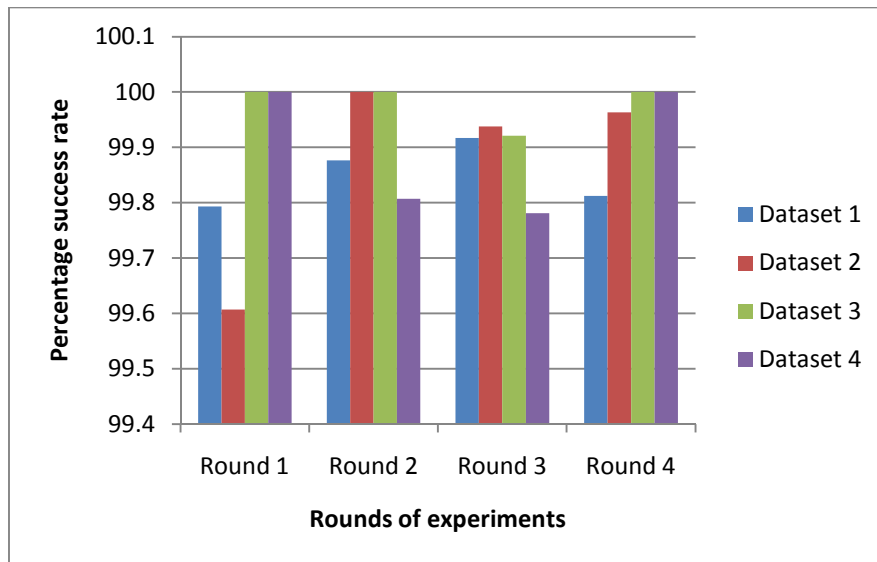


Figure 5: Results of cross validation for all rounds

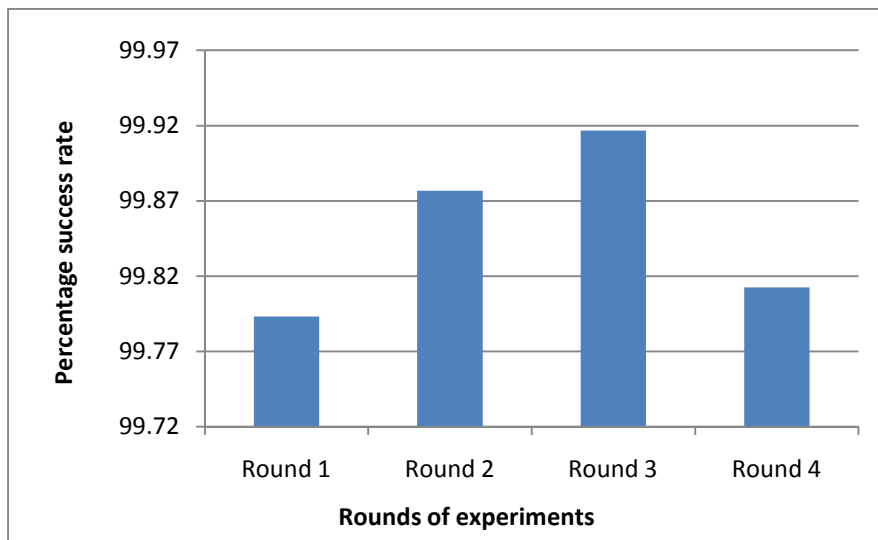


Figure 6: Results of cross validation for a single dataset

As can be seen from these results, the accuracy of prediction is very high as compared to the results of preliminary experiments presented in [32]. This is partly due to the comparatively improved datasets which facilitate better training of the classifiers thereby improving their performance during the test runs. An argument against these experiments can be that both the training and the test data are taken from the same dataset which can result in inherent similarities between the test and training datasets. However, conscious efforts were made to ensure the objectivity of the evaluation process by configuring the classifiers to clearly distinguish between the training and test datasets. This was performed by using configuration options in Weka which allow clear separation between test and training datasets instead of random selection. Finally, further experiments were performed with different datasets as test and training sets as explained in the next section.

8.4.2 Combination of different datasets

As described in the previous section, cross validation used both training and test datasets from the same dataset. This can raise questions about the accuracy of the experimental

results. In order to assess the proposed method further, a set of experiments were performed using different training and test datasets. Specifically, the experiments involved four different datasets randomly generated using the process described in section 8.2. Additionally, the elements of these datasets were labelled with appropriate classes using the dynamic weighted scheme described in section 8.2. For each of these four subsets of data, one subset was used as training data to build a classifier. After the training phase, the classifier was tested against the remaining three subsets of data. This ensures that the test data is essentially unseen for the classifier and is different from the training data. Furthermore, these experiments were repeated four times and results were averaged to preserve the objectivity of the experimental results. This effectively means that the experiments were conducted sixteen different randomly generated datasets and results were averaged.

Table 4: Averaged percentage success rates of evaluation with different test and training sets

Training Datasets	Dataset 1	Dataset 2	Dataset 3	Dataset 4
Dataset 1	x	89.959	86.824	92.176
Dataset 2	90.578	x	88.912	94.665
Dataset 3	91.171	84.031	x	90.440
Dataset 4	94.273	94.275	92.242	x

Table 4 presents the results of these experiments where each row in this table represents one round of experiments and each round of experiments is conducted with four different datasets. The entries in this table present the percentage of correctly classified elements from for each experiment. Furthermore, figure 7 presents the results of these experiments with dataset 1 as the training dataset and rest as the test datasets.

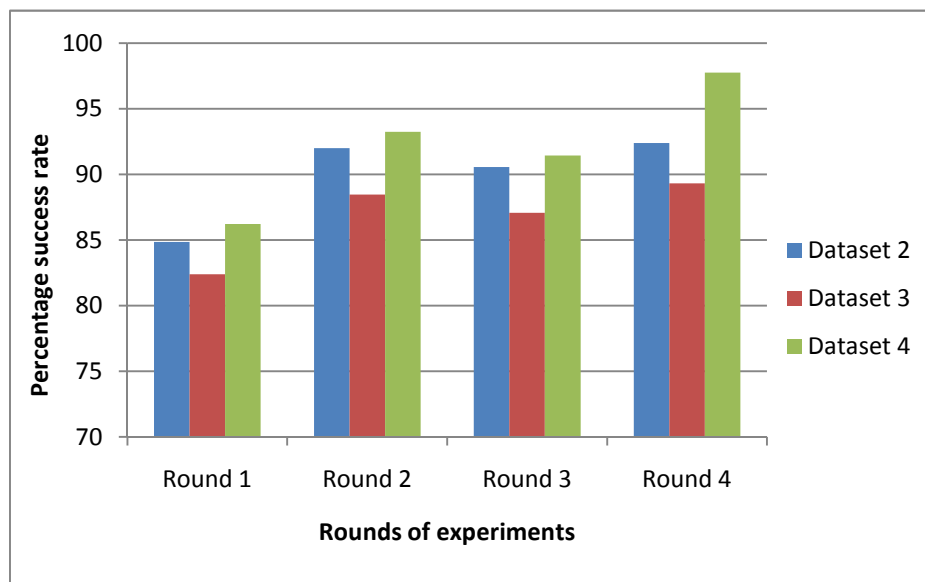


Figure 7: Results of experiments with dataset 1 as training and rest as test sets

As can be seen from both the table 4 and figure 7, the results present a very encouraging picture as compared to the initial experiments. Although these results present lower success rates than the results of cross validation, still the lowest average success rate is 84.03% which is very promising with respect to the feasibility of using machine learning based classification techniques for the severity evaluation problem. Furthermore, figure 8 presents the average

success rates for these experiments. As can be seen from figure 8, the lowest average success rate is 84.03 % whereas most of the remaining results have success rates higher than 90%. This is very encouraging as compared to the initial experiments and therefore, demonstrates the effectiveness of the proposed approach for intrusion severity analysis for Clouds.

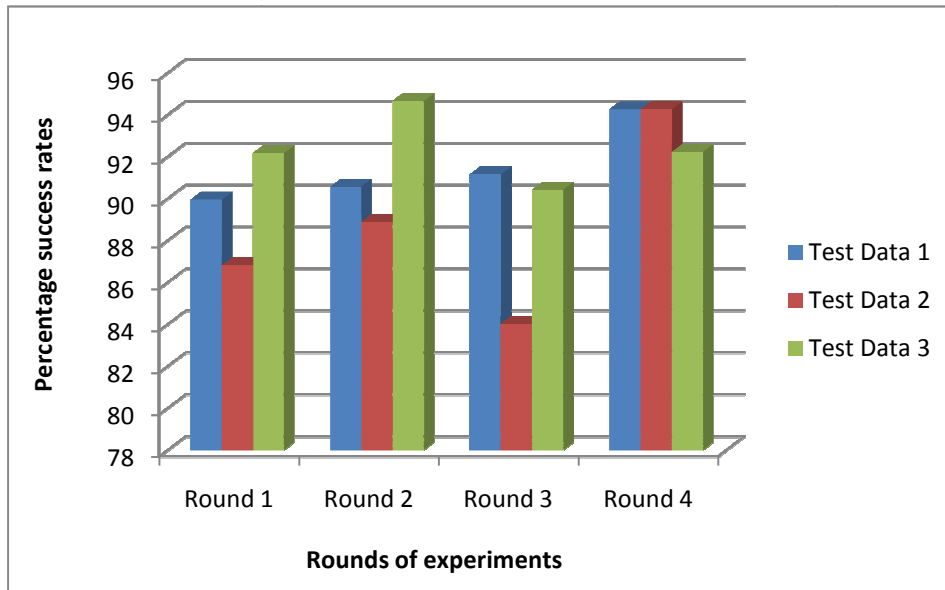


Figure 8: Average success rates for all rounds

8.5 Discussion of results

As described in section 8.3, the experiments presented in this paper were comprised of two types of experiments i.e. cross validation and combination of separate training and test datasets. Related to this the primary objective of these set of experiments was to assess the effectiveness of the proposed method with improved quality of data. The results obtained from these experiments as presented in the previous section successfully demonstrate the effectiveness of the intrusion severity analysis method for Clouds proposed in this paper.

The cross validation experiments involved testing four rounds of four datasets each with both training and test datasets taken from the same set. As described in figures 5 and 6, the results of these experiments represent close to ideal success rates. It can be seen that more than one of these classifiers achieved 100% success rate whereas the lowest success rate achieved during all sixteen rounds was 99.6%. These results clearly demonstrate the effectiveness of using machine learning techniques for intrusion severity analysis. The second type of experiments used different randomly generated datasets as training and test sets. Although these experiments did not result in as big success as the cross validation experiments, however, these do mark substantial improvements as compared to the initial experiments. As can be seen in figures 7 and 8, the success rates for most of the experiments has been more than 90%. Furthermore, the aggregate average of all the experimental results presented in table 4 is 90.7954% which is very promising given the austerity of the experimental process. These results further establish the effectiveness of the proposed approach for intrusion severity for Clouds in general and feasibility of using machine learning based classification approaches for problem addressed in this paper in particular.

However, the size of the tree created during these experiments is considerably large as can be seen from the figure 9. This is regarded as a result of the relatively complex classification procedure adopted for training datasets. Additionally, as the rules represent standard *if-else* statements and the primary objective of this research is to achieve best possible results with

respect to severity analysis, the performance overhead due to the size of tree is not considered. In particular the average decision tree size is recorded to be more than 100 which can indicate a performance overhead. The assessment of this overhead along with possibilities to compact the tree whilst preserving the decision tree remains an area of future work.

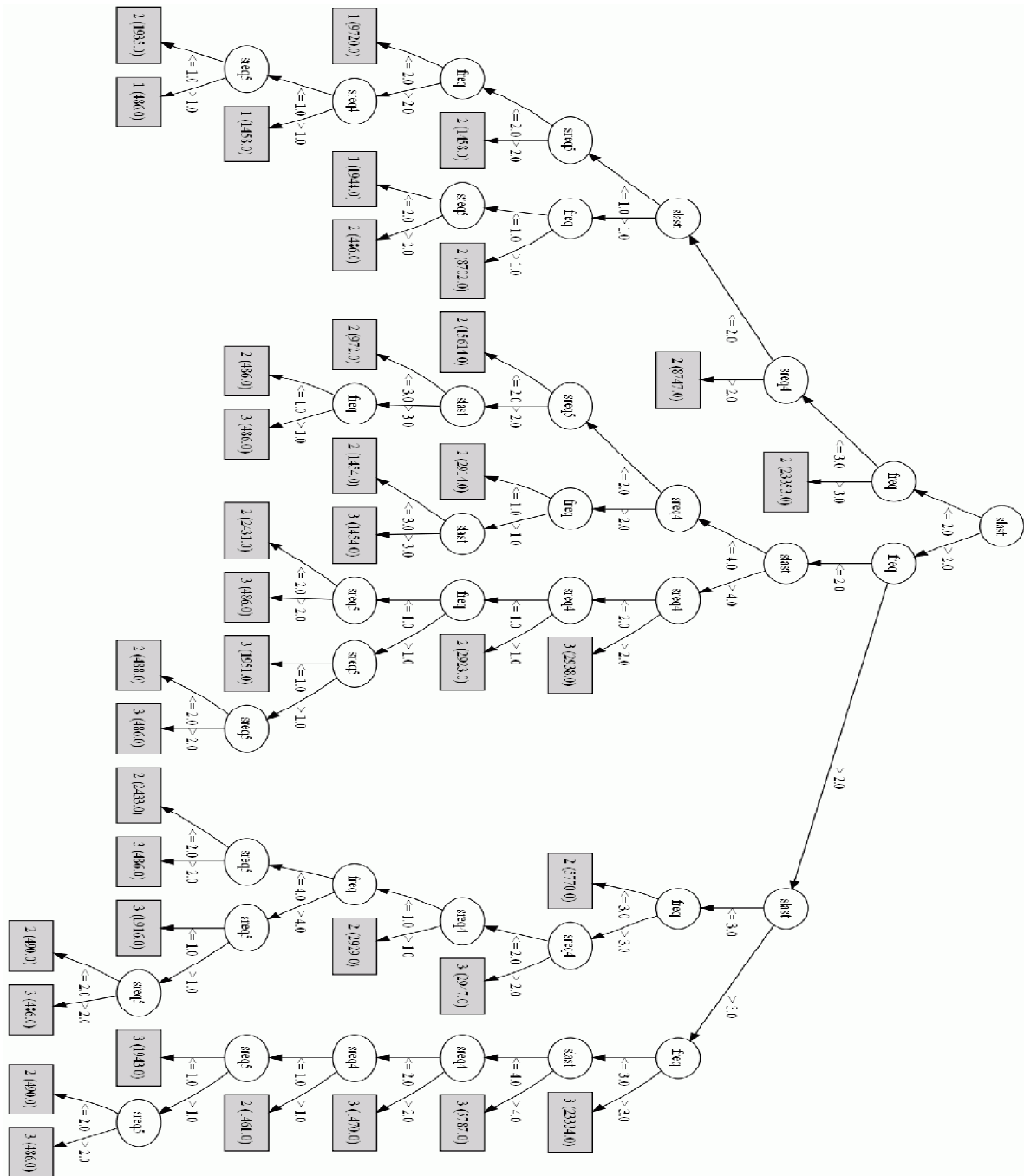


Figure 9: an example decision tree for the experiments

9. Advantages of the Proposed Method

This paper has presented a method to address the ISA problem for Clouds along with rigorous evaluation to determine its effectiveness and feasibility for Clouds. This section presents the benefits of this method and how it helps address the ISA problem for Clouds.

- *Comprehensive representation of security characteristics.* As discussed earlier, security characteristics of a workload have significant effect on the severity of an intrusion for the workload. In order to address the diversity of security characteristics, the method proposed in this paper quantifies security into a set of security requirements. By doing this, it improves the representation of security characteristics of a workload and therefore, contributes to the accuracy of the intrusion severity analysis process.
- *Automated approach.* Clouds are large-scale VM based system where VMs are created, migrated and deleted at runtime on demand of a user. In order to mitigate this dynamic nature of Clouds, one objective of the proposed approach is to achieve maximum automation. This is achieved by two factors; firstly, SAM is proposed to be integrated with other modules such as IDS, and potential human intervention is limited to specifying security requirements at resource acquisition phase. Finally, automation also facilitates seamless operation transparent to the users of the VMs.
- *Ability to determine severity for unknown events.* The proposed method uses machine learning techniques to evaluate severity of an intrusion for a particular GVM. This is beneficial with respect to zero-day malicious attempts i.e. previously unknown intrusion attempts. As the proposed method does not depend on a set of existing values of severity evaluation, previously unknown malicious events are not ignored.
- *Dynamic behaviour.* Among other parameters, the proposed method also takes into account the SLA state and the frequency of attack on a particular security requirement. These parameters help achieve runtime behaviour of the severity analysis process. This is because both these parameters can change according to changes in the environment such as; increase or decrease in the number of malicious attempts on a security requirement for a VM, and change in the SLA state with respect to time and other QoS attributes.
- *Significant reduction in intrusion response time.* One of the implications of the automated nature of the proposed scheme is its effects on the intrusion response time. Intrusion response time is crucial to deliver required level of quality for a particular service. For instance, in the absence of an automated mechanism, the IDS will generate alerts to be responded by a human system administrator. During the time elapsed between the alert raised and the time when a response is initiated, the security of victim service is compromised. Therefore, automated mechanism facilitates reduced response time.
- *Implications for QoS delivery.* Another benefit of the proposed method is that it enables a CSP to offer different QoS with respect to security at the level of VM with varying security requirements. This can be achieved by using the security requirements quantified as part of the proposed method to differentiate between different types of quality of security service available to the customers.
- *Trustworthiness.* As the proposed technique is envisaged to be implemented as part of the domain 0, maximum isolation is guaranteed. This helps to improve the trustworthiness of the analysis performed by the proposed method.

10. Conclusions and Future Work

Cloud computing presents exciting opportunities to establish large scale computing infrastructures which are available on demand to fulfil customized user requirements. However, as with any other emerging paradigm, security underpins extensive adoption of

Clouds. In addition to the contemporary security issues, Clouds present novel security challenges which require dedicated efforts for their solution. This paper has focused on one such challenge i.e. intrusion severity analysis problem for Clouds. In an effort to address this challenge, the paper presents a machine learning based approach which makes use of virtual machine specific parameters such as security requirements, SLA state and frequency of attack. In order to assess the effectiveness of this approach for Clouds, the paper presents rigorous experimentation along with a detailed discussion of the results of this experimentation.

As an outcome of the evaluation, the results show majority of success rates to be over 90% with the lowest averaged success rate of 84.03%. Therefore, these results successfully demonstrate the proposed approach to be effective to address the intrusion severity analysis problem for Clouds. Furthermore, the results also demonstrate the feasibility of use of machine learning techniques to address the intrusion severity analysis problem. However, one opportunity of future improvement with respect to the tree size has also been highlighted. In particular the average decision tree size is recorded to be more than 100 which can indicate a performance overhead. The assessment of this overhead along with possibilities to compact the tree whilst preserving the decision tree remains an area of future work.

10. References

- [1] *Amazon Elastic Computing Cloud*, available at: aws.amazon.com/ec2
- [2] *Google Cloud*, available at: <http://www.googlecloud.com>
- [3] *GoGrid: Scalable Load-Balanced Windows and Linux Cloud-Server Hosting*. available at: <http://www.gogrid.com/>
- [4] *Nimbus Cloud*, available at: www.workspace.globus.org
- [5] *OpenNebula Project*, available at: <http://www.opennebula.org>
- [6] L. Burchard, M. Hovestadt, O. Kao, A. Keller, B. Linnert; *The Virtual Resource Manager: An Architecture for SLA-aware Resource Management*, in the IEEE International Symposium on Cluster Computing and the Grid. pg. 126-133.
- [7] P. Mell, T. Grance; *A NIST National Definition of Cloud Computing*, available at: <http://csrc.nist.gov/groups/SNS/cloud-computing/cloud-def-v15.doc>.
- [8] T. Garfinkel, M. 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](http://www.usenix.org)
- [9] J. Arshad; *Integrated Intrusion Detection and Diagnosis for Clouds*, in the proceedings of Dependable Systems and Networks (DSN), Student Forum 2009
- [10] N. Stakhanova, S. Basu, J. Wong; *A Taxonomy of Intrusion Response Systems*, International Journal of Information and Security. Inderscience Publishers.
- [11] D. Schnackenberg, H. Holliday, R. Smith, et al; *Cooperative Intrusion Traceback and Response Architecture (CITRA)*, in the Proceedings, IEEE DARPA Information Survivability Conference and Exposition (DISCEX I), 2001.
- [12] P. Porras, P. Neumann; *EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances*, in the Proceedings of the 1997 National Information Systems Security Conference, 1997
- [13] *Community Emergency Response Team*, available at: <http://www.cert.org>
- [14] S. Northcutt, J. Novak; *Network Intrusion Detection: An Analyst's Handbook*, 3rd edition New Riders Publishing Thousand Oaks, CA, USA ISBN:0735712654
- [15] P. Mell, K. Scarfone; *A Complete Guide to the Common Vulnerability Scoring System Version 2.0* available at: www.first.org/cvss/cvss-guide.html
- [16] P. A. Porras, M. W. Fong, A. Valdes; *A Mission-Impact-Based Approach to INFOSEC Alarm Correlation*, in the Proceedings of RAID 2002: 95-114.
- [17] P. Barham, B. Dragovic, K. Fraser, et al; *Xen and the Art of Virtualization* in the Proceedings of SOSP'03, October 19.22, 2003

- [19] A. Hussain, J. Heidemann, C. Papadopolous; *A Framework for Classifying Denial of Service Attacks*, in the Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications Pg 99-110 2003.
- [20] J. Arshad, P. Townend, J. Xu; *Quantification of Security for Compute Intensive workloads in Clouds*, in the Proceedings of the International Conference on Parallel and Distributed Systems (ICPADS) 2009.
- [21] J. MacQueen; *Some Methods for Classification and Analysis of Multivariate Observations*, in the Proceedings of the Fifth Berkeley Symposium on Mathematical statistics and probability, pages 281-297, Berkeley, 1967. University of California Press.
- [22] J. R. Quinlan; *Introduction to Decision Trees*, Machine Learning Volume 1, Number 1, 81-106, DOI: 10.1023/A:1022643204877
- [23] F. Dellaert; *The Expectation Maximization Algorithm* College of Computing, Georgia Institute of Technology Technical Report number GIT-GVU-02-20 February 2002.
- [24] J. J. Verbeek et al.; *Efficient Greedy Learning of Gaussian Mixture Models*, Neural Computation, 5(2), pp. 469-485, Feb 2003
- [25] T. Kohonen; *Self-Organizing Maps*, Springer-Verlag Berlin Heidelberg New York 2001 ISBN 3540679219
- [26] J. Shawe-Taylor, N. Cristianini; *Support Vector Machines and Other Kernel-based Learning Methods*, Cambridge University Press, 2000
- [27] A. K. Sen, M. S. Sirivastava; *Regression Analysis: Theory, Methods and Applications*, Springer Publisher 1990 ISBN 0387972110
- [28] J. R. Quinlan; *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, 1993
- [29] C. M. Bishop; *Neural Networks for Pattern Recognition*, Oxford University Press first published 1995. ISBN 0198538642
- [30] W. Jie, J. Arshad, R. Sinnott, P. 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.
- [31] A. Avizienis, J. C. Laprie, B. Randell, C. Landwehr; *Basic Concepts and Taxonomy of Dependable and Secure Computing*, IEEE Transaction on Dependable And Secure Computing, Vol. 1, No. 1, January-March 2004
- [32] J. Arshad, P. Townend, J. Xu; *An Automatic Approach to Intrusion Detection and Diagnosis for Clouds*, Accepted for the International Journal of Automation and Computing special issue for Cloud Computing, Springer Publisher.
- [33] D. Kang, D. Fuller, V. Honavar; *Learning Classifiers for Misuse and Anomaly Detection Using a Bag of System Calls*, In the proceedings of the 2005 IEEE workshop on Information Assurance and Security.
- [34] A. Somayaji, S. Forrest; *Automated Response Using System Call Delays*, in the Proceedings of the 9th conference on USENIX Security Symposium – Volume 9. 2000.
- [35] M. Bernaschi, E. Gabrieli, L. V. Mancini; *Remus: A Security-enhanced Operating System* in the proceedings of ACM Transactions on Information and System Security 2002.
- [36] R. K. C. Chang; *Defending Against Flooding-based Distributed Denial-of-Service Attacks: A Tutorial*, in the IEEE Communications Magazine Oct 2002, Volume 40 Issue 10 Page 42-51.
- [37] Weka – Data Mining with Open Source Machine Learning Software in Java. Available at: <http://www.cs.waikato.ac.nz/ml/weka/>