Watson, M, Shirazi, N-U-H, Marnerides, A, Mauthe, A and Hutchison, D

Malware Detection in Cloud Computing Infrastructures

Article

Malware Detection in Cloud Computing Infrastructures

Michael R. Watson, Noor-ul-hassan Shirazi, Angelos K. Marnerides, Andreas Mauthe and David Hutchison

Abstract—Cloud services are prominent within the private, public and commercial domains. Many of these services are expected to be always on and have a critical nature; therefore, security and resilience are increasingly important aspects. In order to remain resilient, a cloud needs to possess the ability to react not only to known threats, but also to new challenges that target cloud infrastructures. In this paper we introduce and discuss an online cloud anomaly detection approach, comprising dedicated detection components of our cloud resilience architecture. More specifically, we exhibit the applicability of novelty detection under the one-class support Vector Machine (SVM) formulation at the hypervisor level, through the utilisation of features gathered at the system and network levels of a cloud node. We demonstrate that our scheme can reach a high detection accuracy of over 90% whilst detecting various types of malware and DoS attacks. Furthermore, we evaluate the merits of considering not only system-level data, but also network-level data depending on the attack type. Finally, the paper shows that our approach to detection using dedicated monitoring components per VM is particularly applicable to cloud scenarios and leads to a flexible detection system capable of detecting new malware strains with no prior knowledge of their functionality or their underlying instructions.

Index Terms—Security, resilience, invasive software, multi-agent systems, network-level security and protection.

1 Introduction

CLOUD datacenters are beginning to be used for a range of always-on services across private, public and commercial domains. These need to be secure and resilient in the face of challenges that include cyber attacks as well as component failures and mis-configurations. However, clouds have characteristics and intrinsic internal operational structures that impair the use of traditional detection systems. In particular, the range of beneficial properties offered by the cloud, such as service transparency and elasticity, introduce a number of vulnerabilities which are the outcome of its underlying virtualised nature. Moreover, an indirect problem lies with the cloud's external dependency on IP networks, where their resilience and security has been extensively studied, but nevertheless remains an issue [1].

The approach taken in this paper relies on the principles and guidelines provided by an existing resilience framework [2]. The underlying assumption is that in the near future, cloud infrastructures will be increasingly subjected to novel attacks and other anomalies, for which conventional signature based detection systems will be insufficiently equipped and therefore ineffective. Moreover, the majority of current signature-based schemes employ resource-intensive deep packet inspection (DPI) that relies heavily on payload information where in many cases this payload can be encrypted, thus extra decryption cost is incurred. Our proposed scheme goes beyond these limitations since its operation does not depend on a-priori attack signatures and it does not consider payload information, but rather depends on per-flow meta-statistics as derived from packet header and volumetric information (i.e. counts of packets, bytes, etc.). Nonetheless, we argue that our scheme can synergistically operate with signature-based approaches on an online basis in scenarios were decryption is feasible and cost-effective. Overall, it is our goal to develop detection techniques that are specifically targeted at the cloud and integrate with the infrastructure itself in order to, not only detect, but also provide resilience through remediation.

At the infrastructure level we consider: the elements that make up a cloud datacentre, i.e. cloud nodes, which are hardware servers that run a hypervisor in order to host a number of Virtual Machines (VMs); and network infrastructure elements that provide the connectivity within the cloud and connectivity to external service users. A cloud service is provided through one or more interconnected VMs that offer access to the outside world. Cloud services can be divided into three categories based on the amount of control retained by the cloud providers. Software as a Service (SaaS) retains the most control and allows customers to access software functionality on demand, but little else. Platform as a Service (PaaS) provides customers with a choice of execution environment, development tools, etc., but not the ability to administer their own Operating System (OS). Infrastructure as a Service (IaaS) relinquishes the most control by providing customers with the ability to install and administer their own choice of OS and install and run anything on the provided virtualised hardware; as such, IaaS clouds present the most challenges in terms of maintaining a properly functioning system. Such a system would ideally be free from malware and from vulnerabilities that could lead to an attack. It is for this reason that we focus on this type of cloud since security measures applicable to IaaS clouds will also be relevant for other cloud types.

In order to increase the resilience of cloud infrastructures we have already defined a resilience architecture in our previous works [3], [4] that comprises anomaly detection, remediation and also coordination elements. However, this paper discusses two particular components within this architecture that deal with anomaly detection at the system and network level.

The elements presented here form the basis in which different detection techniques can be hosted and further allow the identification and attribution of anomalies. In this paper we discuss the detection of anomalies using a novelty detection approach that employs the one-class Support Vector Machine (SVM) algorithm and demonstrate the effectiveness of detection under different anomaly types. More specifically, we evaluate our approach using malware and Denial of Service (DoS) attacks as emulated within a controlled experimental testbed. The malware samples used are Kelihos and multiple variants of Zeus. We have selected these particular malware samples and their variants since they have been identified as posing recent and evolving threats for a range of Windows OS flavors that have already compromised more than 3.6 million machines worldwide between 2010 and 2014; mainly due to their varying and sophisticated evasion techniques, as well as their stealthy propagation[1]. Our contributions are as follows:

- Experiments carried out in this work are done so in the context of an overall cloud resilience architecture under the implementation of one-class Support Vector Machines (SVMs). The resulting experimental findings show that anomalies can be effectively detected online, with minimal time cost for reasonably realistic data samples per Virtual Machine (VM), using the one-class SVM approach, with an overall accuracy of greater than 90% in most cases.
- Our work is the first to explicitly address the aspect of malware detection in pragmatic cloud-oriented scenarios as performed by cloud providers, such as VM live-migration.
- We provide an online novelty detection implementation that allows the adaptive SVM-specific parameter estimation for providing better detection accuracy benefits.
- This work assesses the VM-based feature selection spectrum (i.e. system, network-based or joint datasets) with respect to the detection performance benefits on two distinct network-wise attacks (malware and DDoS) under novelty detection.

The remainder of this paper is structured as follows: in Section 2 the relevant background is introduced as well as our cloud resilience architecture, into which the detection components we evaluate in this paper are to be placed. Section 3 is dedicated to describing the data and the methodology used in this work. In Section 4 we provide information on our particular evaluation approach setup and a description of the malware samples we have used. Section 5 provides the results of the experimentation conducted in this work. Finally, Section 6 summarizes and concludes this paper.

1. The Kelihos malware was first detected in 2010 and has since been developed into new variants that perform a range of attacks such as phishing and spamming [5]. Zeus was first detected in 2010 [6], but since then there has been a plethora of new variants that even recently (July 2014) compromised millions of machines and gave rise to a botnet that could steal sensitive banking information [7].

2 Background & Related Work

The intrinsic properties of virtualised infrastructures (such as elasticity, dynamic resource allocation, service co-hosting and migration) make clouds attractive as service platforms. Though, at the same time they create a new set of security challenges. These have to be understood in order to better protect such systems and make them more secure. A number of studies have addressed aspects of cloud security from different viewpoints (e.g. the network, hypervisor, guest VM and Operating System (OS)) under various approaches derived either from traditional rule-based Intrusion Detection Systems (IDSs) or statistical anomaly detection models. This paper presents a cloud security solution derived from a sub-domain of anomaly detection, viz. novelty detection. In this section we firstly review the challenges arising from the virtualisation embedded within cloud technologies and further discuss background and related work with respect to anomaly detection in cloud environments. We also present the architectural context, within which the research presented in this paper is carried out.

2.1 Virtualisation & Cloud Technologies

In [3], [8], [9] the specific security threats and challenges introduced into clouds through the use of core virtualisation technologies are discussed. Despite the end-user benefits gained by virtualisation it also comes with a range of threats that include: exploits to security holes on virtual machines (e.g. rootkit attacks on virtual machines [10]); mutated cloud-specific Internet-based attacks that aim to compromise cloud networks (e.g. malware [11], [3]; and DDoS attacks on cloud services [11]). According to [12] blackhat hackers have already identified the potential of the cloud since the instantiation, maintenance and continued operation of botnets seems to be much more effective under a cloud paradigm.

In parallel, co-residence as a security concern has been explored in [10] and is the result of VMs belonging to different customers being hosted on the same cloud node. It was revealed that the outcome of co-residence is to enable shared memory attacks that, at their most benign, are capable of leaking sensitive information, and at their most destructive are capable of taking control of the entire node. Moreover, the aspect of VM migration is also a possible enabler of malicious side effects in cases where infected VMs are migrated around the cloud to a number of nodes. The cause of migration could be as a result of the provider's load balancing policy, but as an unwanted side-effect the result is to place malware in contact with a larger number of potential targets throughout the cloud infrastructure.

Additionally, automation is becoming an increasingly integral part of computer system configuration through the use of dedicated tools (e.g. Ansible[2]) or simply by creating new VMs from clones or snapshots. This results in a collection of servers, all with the same functionality, being configured in precisely the same way. Hence, vulnerabilities and threats are being repeatedly instantiated across large portions of the cloud and malware can more easily propagate and exploit said vulnerabilities.

2. Ansible configuration management software: http://www.ansible.com/home

2.2 Malware & Detection Methods

One of the biggest challenges within the development of resilient and secure cloud-oriented mechanisms is related to the adequate identification and detection of malware. This is due to the fact that, in the majority of cases, malware is the first point of initiation for large-scale Distributed Denial of Service (DDoS) attacks, phishing and email spamming [3], [8], mainly through the deployment of botware.

Current methods of detecting attacks on cloud infrastructures or the VMs resident within them do not sufficiently address cloud specific issues. Despite the huge efforts employed in past studies regarding the behaviour of certain types of malware in the Internet [13], [14], so far little has been done to tackle malware presence in clouds. In particular, the studies in [15], [16] aimed to adjust the performance of traditional Intrusion Detection Systems (IDS) under signature-based techniques that employ Deep Packet Inspection (DPI) on network packets. Moreover, work in [17], [18] studied system-related features on monitored VMs by employing Virtual Machine Introspection (VMI) methods in order to detect threats on a given VM's Operating System (OS).

Nevertheless, despite the important lessons learned from these studies they do not develop an overall online detection strategy that considers real-time measurement samples from each VM. Further, these approaches are purely signature-based, and as such are not in a position to provide a robust scheme for any future threats posed by novel malware strains due to their simplistic rule-based nature.

Each solution to detection is performed in an isolated manner and neglects to consider the unique topology of the cloud, which is at its heart a network of interconnected nodes, each with their own isolated execution environments. If a detection system is to perform effectively within a cloud it is required to possess the capability of communicating detected faults and challenges across the whole infrastructure, especially if it is to perform as part of a larger, autonomous and self-organising, cloud resilience system.

2.3 Anomaly Detection in Clouds

Anomaly detection has been an active research area for a number of years. Numerous techniques for different scenarios and application domains have been developed. Chandola et al. show in their survey [19] the prediction, detection and forecasting accuracy of anomaly detection in a number of disciplines, whereas the work in [20] thoroughly surveys the use of several anomaly detection schemes in the context of IP backbone networks. Within this paper the focus is on anomaly detection in the cloud.

A number of anomaly detection techniques [21], [22], [23], [24], [25], [26] aim to proactively and reactively detect cloud-specific threats, but due to their complex statistical measures they mostly lack scalability and often require prior knowledge, thus making them unsuitable for online detection in cloud infrastructures.

The work by Wang et al. [27] produced the EbAT system that allowed the online analysis of multiple metrics obtained from system-level components (e.g. CPU utilization on rack servers, memory utilization, read/write counts of the OS, etc.). The proposed system showed potential in the areas of detection accuracy and monitoring scalability, however it's evaluation did not adequately emphasise pragmatic cloud scenarios.

In [28] an anomaly detection technique to detect intrusions at different layers of the cloud was proposed. However, the technique appears to lack the flexibility required by dynamic cloud environments. It is also not sufficiently demonstrated how such techniques can be operationally applied. In [29] the authors propose a multi-level approach, which provides fast detection of anomalies discovered in the system logs of each guest OS. One of its disadvantageous is the apparent lack of scalability since it requires increasingly more resources under high system workload. Further, it is designed to classify text based log data, which may not manifest the effects of malware.

The work in [24] provided a novel prototype that enabled an online spatio-temporal anomaly detection scheme in a cloud scenario. Thus, the authors were able to initially formulate and further implement a wavelet-based multi-scale anomaly detection system. The system relies on measured cloud performance metrics (e.g. CPU utilization, memory) gathered by multiple components (e.g. hardware, software, system) within the examined institution-wide cloud environment. The resulting experimental outcomes were quite promising since the proposed approach reached a 93.3% of sensitivity on detecting anomalous events with only just a 6.1% of the reported events to be false alarms.

The only study that has some similarities to what we propose in this paper is the approach by Pannu et al. in [30]. In particular, the authors in [30] instrumented an online adaptive anomaly detection (AAD) framework that was able to detect failures through the analysis of execution and runtime metrics using the traditional two-class Support Vector Machine (SVM) algorithm. Under a real experimentation, over a 362-node cloud computing environment in a university campus, the produced results were extremely promising since they exhibited the efficiency of the proposed scheme, which reached an overall of over 87% of anomaly detection sensitivity. However, the main issue raised by this study was that the formulation of the two-class SVM algorithm suffered from the data imbalance problem [31], which affected the training phase, and consequently led to several mis-classifications of newly tested anomalies. Moreover, in contrast with our work the proposed approach did not explicitly address the aspect of early attack detection, but rather was mainly aimed at various faults in the cloud infrastructure.

Therefore, apart from providing an online anomaly detection approach, our work is also aimed at confronting an algorithmic constraint that is inherited in most of the traditional two-class on n-class Machine-Learning based techniques (e.g. two-class SVMs, Artificial Neural Networks, Bayesian Classifiers) when applied to cloud environments (e.g. [30], [32]); data imbalance. As indicated in [31], [33] a dataset is imbalanced if the classification labels are not approximately equally represented. In simple terms, the imbalanced nature of training datasets[3] invoke high clas-

3. Having a tremendously large training dataset for DDoS attacks versus a comparatively small one for malware instances, for example

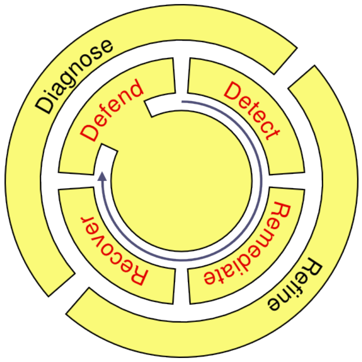

Fig. 1. A high level overview of the $D^2R^2 + DR$ network resilience framework [2]

sification errors and problematic labelling of the training instances that naturally engage high rates of misclassification throughout the testing phase of n-*class* classifiers (e.g. traditional SVMs). Hence, in this work we are inspired by the findings in [31], where one-class SVMs perform much better than two-class SVMs, as well as Artificial Neural Networks (ANNs), in the context of classifying DSL-level faults. Here we employ them explicitly for the detection of anomalous events in cloud environments, in particular those resulting from the execution of malware. One further reason to use one-class SVMs in the context of our research is the lack of dependence on prior knowledge regarding a particular cause for anomalous behaviour and the ability to detect new types of anomalous events as "novelties". As a result, it is possible to detect anomalies that are not well understood (i.e. no prior models) under the concept of novelty detection since they were not experienced throughout the training phase of a one-class SVM[4].

2.4 Cloud Resilience Architecture

The research introduced in this paper is part of a larger international research initiative on network and system resilience. It is based on the $D^2R^2 + DR$ network resilience framework [2]. This framework comprises two nested modes of operation. An inner real-time control loop comprising *Defending* the system, *Detecting* faults and anomalies, *Remediating* against them, and finally *Recovering* from any detected faults. And an outer loop that *Diagnoses* weaknesses in the current configuration and *Refinines* the overall system and resilience strategy. Whilst the inner control loop aims at protection in real-time, the outer control loop is conducted over a longer period of time (see Figure 1).

In order to realise the $D^2R^2 + DR$ strategy, network and system specific resilience architectures have been developed with the aim of providing interoperable resilience infrastructures that host the components necessary to enable various resilience methods and techniques. In [4] we introduced a cloud resilience architecture that specifies the components through which detection and remediation in the cloud is realised. The resilience system is distributed and self-organising, and is composed of individual software instances, known as Cloud Resilience Managers (CRMs). Each CRM is composed of four software components, or engines, which are shown in Figure 2[5].

The software components within each CRM are: the System Analysis Engine (SAE), the Network Analysis Engine (NAE), the System Resilience Engine (SRE) and the Coordination and Organisation Engine (COE). The CRM on each node performs local anomaly detection based on features gathered from its node's VMs and its local network view, where those features are handled by the SAE and NAE components respectively. The SRE component is in charge of remediation and recovery actions based on the output from the analysis engines (i.e. the NAE and SAE), which is conveyed to it by the COE. Finally, the COE component coordinates and disseminates information between other instances and the components within its own node. It is the COE that is ultimately in charge of the maintenance of the connections between its CRM peers and embodies the self-organising aspect of the overall system.

In addition to node level resilience, the detection system is capable of gathering and analysing data at the network component level through the deployment of network CRMs as shown by **C** in Figure 2. Network level CRMs operate in exactly the same manner as the CRMs deployed within the cloud, but are able to observe network traffic from a unique vantage point not available to the inner network. For example, a CRM deployed on an ingress/egress router (i.e. **D** in the figure) is able to observe traffic before it is firewalled, enabling it to communicate valuable information back into the cloud. An ingress/egress CRM is also able to analyse the traffic from multiple nodes, allowing the presence of a botnet to be detected, communicated to each internal CRM, and thwarted by the SREs on each node. However, the research presented in this paper is concerned with the online detection component within the System Analysis Engine (SAE) and Network Analysis Engine (NAE), hence further details about the overall resilience architecture can be found in [4], [3], [8].

Based on features gathered from each individual VM, the SAE and NAE are designed to enforce algorithms that are capable of building models for normal VM operation. These are then used to pinpoint anomalous events. In our implementation, features are extracted from the virtual memory of each VM (e.g. process memory usage) as well as from the network interface of each VM and are combined to form a feature vector for each measurement interval. Under normal operation (i.e. with no malware injected)[6] all of the feature vectors are combined into a training dataset for the one-class SVM formulation. Conversely, under detection conditions each newly monitored and post-processed feature vector is tested against the training data in order to determine whether it is anomalous or normal. The following section

4. For example, in our work we train the classifier to label feature vectors that strictly represent normal behaviour. Thus, malware instances, which consequently change the statistical properties of newly tested feature vectors, are labelled as "novelties" because they represent deviations from the normal operation of the cloud.

5. Element **A** in Figure 2 represents a single hardware node in the cloud, while **B** represents the CRM, along with its associated software engines.

6. Normal operation was determined based on the knowledge that our VMs were not infected or attacked within our controlled experimental testbed. This process is also quite common within large organizations as revealed by discussions with cloud providers and collaborators under the IU-ATC project [34].

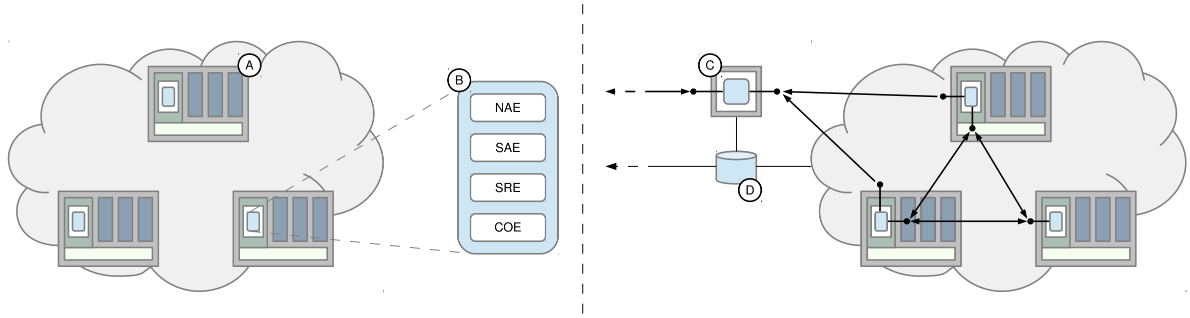

Fig. 2. An overview of the detection system architecture

is dedicated to describing the steps involved in our methodology in order to conduct the above operations in an online mode.

3 METHODOLOGY

The cloud testbed used in this work is based on KVM hypervisors under Linux (which in turn use Qemu for hardware emulation). The testbed comprises two compute nodes, one of which also acts as the storage server for VM images, and a separate controller server. The management software is Virtual Machine Manager (sometimes referred to as virt-manager), which interfaces with libvirt daemons on the compute nodes. Cloud orchestration software (such as OpenStack) is not deemed necessary for our particular experiments since we are concerned solely with direct data acquisition from VMs and not the interaction of the detection system with management software. However, the tools used in this work are compatible with any cloud orchestration software that uses either Xen or KVM as a hypervisor and the approach we take here could therefore be applied to such an environment. In general, our testbed is capable of many of the functions associated with cloud computing such as flexible provisioning of VMs, cloning and snapshotting VM images, and offline and online[7] migration.

3.1 Data Collection & Feature Extraction

The data collection and analysis tools installed on each compute node in the described testbed include libVMI[8] and Volatility[9] for real-time Virtual Machine Introspection (VMI), tcpdump[10] and CAIDA's CoralReef[11] for packet capturing and network flow summarisation. Overall, the data acquisition, feature extraction and anomaly detection performed by both the SAE and NAE components of our resilience architecture are achieved through custom software that operates on VMs in real-time at the hypervisor level of the cloud node.

Based on the monitoring and measurement tools described above, the collection of training data into a training dataset is achieved through the monitoring of a VM that has been created from a known-to-be-clean disk image. Each VM snapshot that is collected is stored in a single file that represents the normal behaviour of that VM image. At 8 second intervals the Volatility tool is invoked with our custom plugin that crawls VM memory for every resident process structure. From each process we extract the following raw features per process:

- memory usage (i.e. actual size of the process in memory)
- peak memory usage (i.e. the requested memory allocation)
- number of threads
- number of handles (resources the process has open, e.g. files)

As mentioned, the raw features are per process, which is not useful if we are to consider each sample, or snapshot, as a single feature vector. Therefore a subsequent step is dedicated to building statistical meta-features such as the mean, variance and standard deviation of each feature across all processes. This results in a final feature vector for the snapshot of the form $x = (x_1, x_2, \ldots, x_{n-1}, x_n)$, where $n = 12$ due to the three groups of four meta-features. At this stage, the snapshot feature vector is either appended to a file that represents the training dataset for normal operation, or is classified through online anomaly detection.

At the network level the NAE gathers data through tcpdump, which separates packets into 8 second time bins. Features are then extracted using the CAIDA CoralReef suite of tools, which provides the capability to generate statistics per uni-directional TCP and UDP flow. The raw features include:

- packets per address pair
- bytes per address pair
- flows per address pair

The raw features are then used to produce meta-features in a similar manner to the functionality of the SAE. The resulting feature vector therefore has dimension $n = 9$ and, in experiments where the NAE and SAE feature sets are combined into one, the resulting feature vector has dimensionality $n = 21$.

7. Also referred to as cold and live migration respectively.
8. libVMI: https://code.google.com/p/vmitools/
9. Volatility framework: https://code.google.com/p/volatility/
10. tcpdump/libpcap: http://www.tcpdump.org/
11. CoralReef software suite: http://www.caida.org/tools/measurement/coralreef/

3.2 One-Class SVM

The core of our online detection methodology within the SAE and NAE lies with the implementation of the supervised one-class SVM algorithm, which is an extension of traditional two-class SVM, and was proposed by Scholkopf et al. in [35]. In practise, the one-class SVM formulation handles cases using unlabelled data (i.e. novelty detection), the main goal of which is to produce a decision function that is able to return a class vector y given an input matrix x based on the distribution of a training dataset. The class y is a binary class where one outcome is the known class, which in our case is the normal VM behaviour, and the other is the novel class, which represents any testing instances that are unknown to the classifier. If we let $x = (x_1, x_2, \ldots, x_{n-1}, x_n)$ represent a feature vector, which contains all of the VM-related features described earlier (section 3.1), then the decision function $f(x)$ takes the form:

$$f(x) = \sum_{i=1}^{N} \alpha_i k(x, x_i) - \rho \tag{1}$$

However, in order to achieve $f(x)$ and attain the α_i multipier over the kernel function $k(x, x_i)$ it is firstly required to solve the optimisation problem in Equation 2 using Lagrange multipliers, as follows:

$$\begin{gathered} \min_{w,\xi_i,\rho} \tfrac{1}{2}\|w\|^2 + \tfrac{1}{\nu n}\textstyle\sum_{i=1}^{n} \xi_n - \rho \\ \text{subject to:} \\ (w \cdot \phi(x_i)) \geq \rho - \xi_i \quad \text{for all } i = 1, \ldots, n \\ \xi_i \geq 0 \quad \text{for all } i = 1, \ldots, n \end{gathered} \tag{2}$$

The parameter ν is extremely critical and characterises the solution by setting an upper bound on the fraction of outliers, and a lower bound on the number of support vectors. Increasing ν results in a wider soft margin, meaning there is a higher probability that the training data will fall outside the normal frontier, thus identifying legitimate VM behaviour as anomalous in our case. With reference to Equation 1, the function $k(x, x_i)$ denotes the kernel function and can be chosen to suit a particular problem. In our implementation we employed the Radial Basis Function (RBF) kernel function, which is defined as:

$$k(x, x_i) = \exp(-\gamma\|x - x_i\|^2) \tag{3}$$

The kernel parameter γ is sometimes expressed as $1/\sigma^2$ and a reduction in σ results in an decrease in the smoothness of the frontier between normal data and outliers. It is therefore possible to produce a decision function which approximates a nearest neighbour classifier by increasing the value of γ. As we explain next, both γ and ν parameters are quite critical and require some tuning in order to avoid missclassifications of abnormal behaviour to normal and vice versa.

3.3 SAE & NAE One-Class SVM Tuning

Prior to the training process, the SAE & NAE engines automatically transform the initial gathered dataset by scaling them towards a Gaussian distribution. This is due to a requirement of the RBF kernel that the data be centred on zero and have unit variance. Thus the tuning process embedded in the SAE and NAE removes the mean from each feature and divides the feature vector by the standard deviation. The training process subsequently involves passing the scaled training dataset as an input to the one-class SVM algorithm, which produces a decision function that is able to classify new feature vectors.

In general, the training process is determined by four factors: the size and content of the training dataset and the two parameters ν and γ. The training dataset size is determined by the length of time over which VM monitoring is conducted, after which it is possible to select subsets of the available data resulting in a refinement of training data and a reduction in dataset size if required. Dataset content is determined by the behaviour of the processes in the VM and is not accurately controllable, hence the only influence that can be imposed on the data is by varying the applications and the loads on each of them. In contrast, the parameters ν and γ can be finely controlled and are chosen at training time to alter the accuracy of the classifier with respect to the available training data.

The choice of algorithm parameters is not obvious *a priori* and a small change of ν or γ either way can result in a less accurate detector. However, by choosing the parameters based on how accurately the classifier classifies its own training dataset it is possible to optimise the detector for a particular server profile.

The process of parameter selection is conducted in an incremental manner by selecting the lowest reasonable values for ν and γ and incrementing the values of first ν and then γ in a pair of nested loops[12]. The increment for γ need not be as fine as ν because, within our experimentation, we have found it to have much less influence on the accuracy of the detector. At each step the training data is reclassified using the new values of ν and γ and the False Positive Rate (FPR) is calculated for the pair of parameter values according to the formula in Equation 4. This search allows us to select the values that produce a minimum FPR.

Overall, by conducting this iterative process we have found that once a minimum is reached there may be some parameter pairs that yield the same minimum, after which the FPR will rise again for all subsequent pairs of values. This is to be expected due to the fact that increasing both parameters past a certain point results in a frontier that fits too tightly to close neighbours in the training data and does not generalise well. Thus, a compromise needs to be reached between fitting the training data loosely with low values of the algorithm parameters, and being too restrictive with high values. Hence, with empirical experience of search times it is possible to stop the procedure long before the end of the exhaustive search and therefore reach an optimised set of parameters in reasonable time[13].

3.4 SAE & NAE Online Detection Process

As described in the previous subsections, the one-class SVM classifier within our SAE and NAE implementations is trained to identify anomalies by training it on a dataset

12. Since it is impossible for ν to be equal to 0 we begin the search with a value of 0.0001 and also increment in intervals of 0.0001. Since γ can be any non negative real number we begin at 0 and increment in intervals of 0.01

13. Within our experimentation we found that this iterative process takes no more than 10 seconds on an average machine and need only be carried out once per training dataset

of normal VM behaviour. This is embodied in a dataset comprising features obtained during normal operation and is used to generate a decision function that is capable of classifying novel samples (i.e. anomalous behaviour). Once trained, the classifier operates on feature vectors in an online capacity in order to produce a classification in real-time. The evaluation of the classifier within the SAE is conducted experimentally through the following procedure:

- A clean VM is created from a known-to-be-clean disk image
- The VM is monitored for a period of 10 minutes in what we refer to as the "normal phase"
- Malware is injected and a further 10 minutes of monitoring follows in what we refer to as the "anomalous phase"

The output of the detector component is a vector y with an n dimension equal to the m dimension of the input matrix, which in the case of online detection yields a single value of $y \in \{-1, 1\}$ for each snapshot vector x. This means it is possible to infer the success of the detector from its output relative to the phase in which the output was produced.

3.5 Classification Performance Metrics

The detection performance of the classifier can be assessed by determining the difference between the class it produces for a given input and the class it should produce. For example, if a sample of data contains no anomalies due to a malware strain, and the classifier produces an output of 1 for that data point, it is a correct classification. In order to quantify the classification performance we consult a confusion matrix that describes all possible outcomes of a prediction and has the form:

		Actual Class 1	**Actual Class** −1
Predicted Class	**1**	TN	FN
	−1	FP	TP

In our experiments a "positive" outcome is one in which the detector detects an anomaly, i.e. produces a class of -1. From this we can conclude that a True Positive (TP) is possible when the classifier produces a -1 during malware execution, otherwise it is treated as a False Positive (FP). Similarly, negative results occur when the detector detects normal operation. As such, if malware is not executing, an output of 1 is a True Negative (TN), otherwise it is treated as a False Negative (FN). From the confusion matrix it is possible to derive a number of performance metrics which are shown in Equation 4 below.

$$
\begin{aligned}
FPR &= \frac{FP}{FP + TN} \\
Accuracy &= \frac{TP + TN}{TP + TN + FP + FN} \\
Precision &= \frac{TP}{TP + FP} \\
Recall &= \frac{TP}{TP + FN} \\
F\ score &= 2 \times \left(\frac{Precision \times Recall}{Precision + Recall} \right) \\
G\ mean &= \sqrt{Precision \times Recall}
\end{aligned}
\tag{4}
$$

Accuracy is the degree to which the detector classifies any newly tested data samples correctly whereas precision is a measure of how many of the positive classifications are correct, i.e. the probability that a detected anomaly has been correctly classified. The recall metric is a measure of the detector's ability to correctly identify an anomaly, i.e. the probability that an anomalous sample will be correctly detected. The final two metrics are the harmonic mean (F score) and geometric mean (G mean), which provide a more rounded measure of the performance of a particular detector by accounting for all of the outcomes to some degree.

4 Experimental Scenarios & Malware Description

4.1 Malware Analysis on Static VMs

An initial concern of any cloud provider should be the aspect of VM screening; the process of profiling the system and network features of a running VM and subsequently confirming that it is not infected with malware. Thus, our first experiment as illustrated via Figure 3 utilised the testbed configuration described earlier and aimed to evaluate our screening process by injecting malware and also emulating a DDoS attack (as described in section 5.6) on a given VM. The VM in our experimentation hosts a simple web server that provides an HTTP service to multiple client requests. The experiment lasted for 20 minutes, with malware injection (using Kelihos and Zeus malware strains serparately) on the 10^{th} minute. In order to generate some realistic background traffic we developed some custom scripts on other hosts within the same LAN that enabled the random generation of HTTP requests to the target server[14]. The choice of HTTP for traffic generation is typical of many cloud servers that host web servers or related REST based applications. In addition, these types of server are among the most targeted by malware due to them being very public facing, and therefore require the most monitoring.

The experiment duration was chosen based on empirical experience of the behaviour of our chosen malware strains. Since detection is conducted in real-time it is necessary

14. These scripts were based on the implementation of iperf clients and they included random bursty and "lightweight" requests with varying content and flow size.

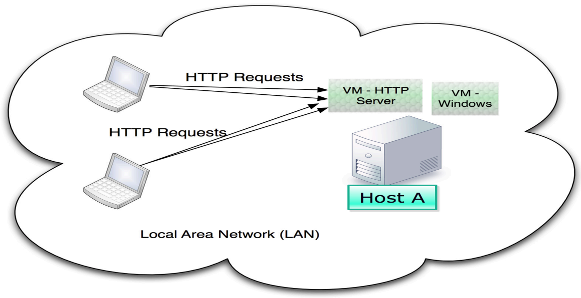

Fig. 3. Visualization for the experimental setup for static malware analysis.

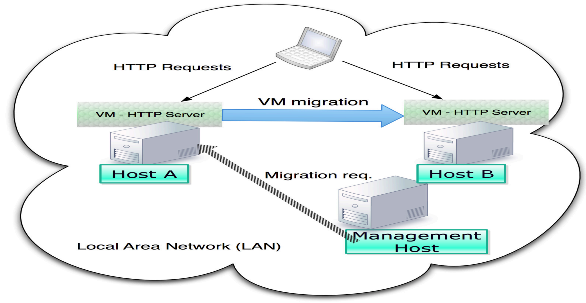

Fig. 4. Visualization for the experimental setup for malware analysis under VM migration.

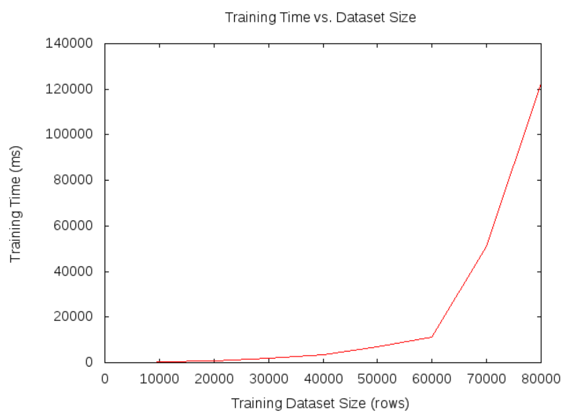

Fig. 5. Time taken to train the classifier vs. training dataset size

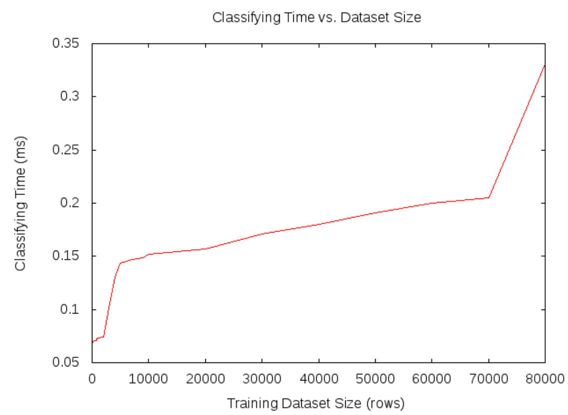

Fig. 6. Time taken to output a class vs. training dataset size

to have the experiment run uninterrupted. This places a constraint on the format of the experiment whereby the malware needs to be detectable for the duration of the anomalous period, with the only valid outcomes being TPs and FNs during this time. Through experimentation it was observed that the Zeus samples we obtained have a tendency to cease execution beyond 15 minutes; we do not, therefore, need to continue the experiment beyond this boundary. Moreover, we have found that 10 minutes of malware execution is more than sufficient to characterise the detection performance of the detector under the parameters of our experimentation.

4.2 Malware Analysis During Live-Migration

Cloud providers are also heavily concerned with the security implications associated with the scenario of VM/service migration from one physical host to another. Thus, in this work we have explicitly targeted live migration for experimentation, since the greatest majority of commercial cloud management software (e.g. VMWare VSphere[15]) employ this functionality by default. Therefore, the objectives of our second experiment were: to firstly determine whether the malware resident on an infected VM would remain operational post-migration; secondly, we aimed to address the actual detection of the malware from data gathered at the hypervisor level of the nodes that hosted the VM.

15. VMWare VSphere: http://www.vmware.com/uk/products/vsphere

When investigating the effects of migration each experimental run had a total duration of 20 minutes. The experiment was divided into two scenarios: one in which the malware was active during the migration, and one in which the malware was injected after migration. In the first scenario the malware was injected on the 10^{th} minute, with migration occuring after injection on the 15^{th} minute. The second scenario involved migration on the 5^{th} minute and injection of the malware on the 10^{th} minute, as before. As Fig. 4 demonstrates, the testbed for the migration scenario consists of four physical machines, where one machine acts as the management entity (in charge of regulating the migration activities between Host A and Host B), one provides the HTTP client connections, and the other two host the infected VM. Throughout the experiment the HTTP sessions remained active despite the migration of the VM, which is precisely the behaviour expected of webservers in the cloud.

4.3 Malware Samples

This work could have not be evaluated without the ability to generate anomalies within a testing environment. It was therefore essential to utilise appropriate samples of genuine malware in our experiments[16]. As already mentioned (see Section 1), both malware strains have been reported to

16. The specific samples of malware used under experimental conditions are: *Trojan.Kelihos-5*, *Trojan.Zbot-1433*, *Trojan.Zbot-1023*, *Trojan.Zbot-18* and *Trojan.Zbot-385*, which were obtained from `offensivecomputing.net`

exhibit sophisticated evasive and propagation properties and they have compromised million of Windows OS-based machines since 2010 until recently, hence we consider their selection and analysis as timely and at the same time necessary.

In particular, the Kelihos malware spawns many child processes and subsequently exits from its main process. This is likely an obfuscation method to avoid detection, but has the effect of skewing system level features resulting in an obvious anomaly. The main purposes of these child processes are to monitor user activity and contact a Command and Control server (C&C) in order to join a botnet. At the same time, the Zeus malware and its variants, exhibit obfuscation techniques that tamper with security software installed on a given host. Its first action is to inject itself into one of the main system processes and to subsequently disable antivirus and security center applications. This behaviour leads to any attempt to detect it from within the OS futile and makes detection systems that exist outside the execution environment of the malware (such as the method used in this work) particularly applicable.

The choice of Windows as the subject of experimentation is largely due to the fact that a range of IaaS clouds do demonstrate a higher need for Windows-based VMs as mentioned by cloud operators within the IU-ATC project [34]. In addition, most of the malware available in binary form have been compiled as Windows executables, thus we chose a compatible target on which to unleash them.

5 Results

The experiments we present in this section test the detection aspects of the System and Network Analysis Engines (SAE and NAE respectively). Given the fact that both engines perform online anomaly detection under the one-class SVM formulation we initially present our results related to the computational cost of the online training and testing of the algorithm, since they affects the overall response of the real-time detection process. We subsequently present our assessment on detecting the Kelihos and Zeus malware strains as well as the DDoS attacks. In addition, we further present a comparison between the detection accuracy obtained when using a joint dataset (i.e. composed of both system and network features) with a featureset that strictly considers network-based features.

The experiments that focus on the SAE functionality involve the detection of Kelihos and Zeus under static analysis and live-migration using a 12 dimensional system-level dataset. NAE performance is tested under static analysis against DoS using a 9 dimensional network-level dataset and against Zeus using the 9 dimensional network dataset and a 21 dimensional joint-level dataset (i.e. system and network).

5.1 Training and Classification Cost Analysis

Figure 5 illustrates the required time for training the one-class SVM classifier on various sizes of training datasets. For the sake of completeness we have experimented with a range of sizes having, as a maximum, a large dataset consisting up to 80000 rows. This was in order to demonstrate the extremely small impact that training and classification have in our actual experimental conditions. The dataset used in the experiments was around 200 samples, which resulted in a training time of between 2 and $10ms$, which is not possible to measure reliably using our tools. Hence, the dataset was extrapolated up to 80000 entries in order to produce an observable trend.

Considering feature extraction takes in the order of seconds to complete[17], the time taken to train the classifier is negligible, especially since it is only required to take place once during the lifetime of the classifier. In scenarios where the role of a server changes significantly and frequently the classifier would need to be retrained in order to produce a model of normal behaviour that sufficiently characterises the new normal behaviour patterns. Though, in our experience, in such cases it is more usual to replace a VM with the new version by swapping one for the other, rather than altering it in place. This allows the new image to be profiled and a more complete model of the new normal to be established before deployment.

Classification could also potentially hold up the process of obtaining a class for a particular vector and, like training, is dependent on dataset size. However, as Figure 6 shows, the time taken to produce a class is also negligible with respect to the time taken to obtain the feature vector itself, despite the fact that classification is carried out on every sample vector.

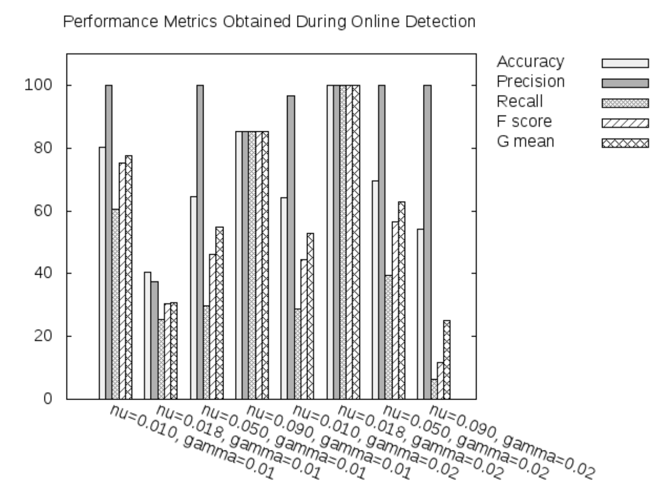

Fig. 7. Results of detection for Kelihos-5 using end-system features and a variety of kernel parameters

5.2 SAE Kelihos Detection

As already mentioned and described in Section 4.3, the first sample of malware used to test the performance of the SAE component was Kelihos (*Trojan.Kelihos-5*), which due to its nature as a trojan can be directly executed on the target VM without the need to explicitly alter the Windows registry.

Our trained and tuned one-class SVM implementation was used in an online mode to classify feature vectors

17. 8 seconds in our experiments, however this is without any optimisations and could be reduced significantly through natively compiled code rather than interpreted scripts.

as they were collected from the test VM. The classifier was tuned according to the methods described in Section 3.3 and was trained using a dataset consisting of around 200 samples of normal behaviour gathered during normal server operation.

The output class produced by the detector for each input vector was determined to be either correct or incorrect depending on the state of the malware sample at the time of feature extraction.

In particular, the timeline for the experiment consisted of two phases: 10 minutes of normal activity, followed by 10 minutes of malware infection where any positive detection classifications in the first phase were therefore false positives, whereas positive results in the second phase were true positives.

The results of this experiment can be seen in Figure 7 where all bar charts shown in the figure were produced by calculating the various performance metrics for each set of the SVM-specific parameters according to the formulae in Equation 4. In our case, the tuned classifier can be identified by the kernel parameters $nu = 0.018$ and $gamma = 0.02$. The other sub-optimal parameter pairs were chosen through empirical experience of the tuning process in order to represent the surrounding parameter space close to the optimal pair.

Based on the generated results it is shown that tuning an SVM classfier according to the method in Section 3.3 results in a more reliable detector for our particular scenario. In addition, the results show that it is possible to reliably classify feature vectors as they are produced, which enables the algorithm to be used in an online capacity to detect anomalies in a target VM as they occur. Furthermore, the anomalies produced by Kelihos as a result of its execution behaviour were detectable using the features collected by our analysis engine at accuracies nearing 100%.

5.3 SAE Zeus Detection

Experiments using Zeus samples were conducted in the same manner as those using Kelihos. A Zeus sample was executed for the last 10 minutes of a 20 minute experiment, during which results were obtained from the classifier in real time.

The first experiment using one Zeus sample tested the ability of the SAE to detect samples other than Kelihos in order to verify that the method is not limited to one type of malware. As evidenced by Figure 8 the detector performs equally well when detecting either Kelihos or Zeus by reaching overall more than 95% of detection accuracy throughout all the detection performance metrics.

5.4 Detecting Zeus Variants

The experiments thus far have tested the SAE against two strains of malware from different malware families. However we felt it necessary to test against different samples from the same strain in order to determine whether our approach is flexible in its classification of anomalous activity.

Figure 9 shows experiments conducted with the same experimental procedure as the previous two experiments, but with each using a different sample of Zeus. The excellent detection results from each show that the method is suitable not only for detecting multiple strains of malware, but also variants of the same strain with more than 90% of overall detection performance for most of the malware strains.

However we have witnessed a poor result that involved the detection of the *Trojan.Zbot-18* strain. We argue that this is likely due to this particular malware sample's execution pattern and not neccesarily a deficiency of the detector. The *Trojan.Zbot-18* sample does not exhibit anomalous activity when first executed, but rather waits for a period of time before continuing to operate. As such the detector correctly detects normal activity even though the experiment has progressed into the infected, or "anomalous", phase. This skews the results so that the detector appears to be performing less well, when in fact the malware is dormant during a portion of its lifetime.

5.5 SAE Detection During VM "Live" Migration

Clouds are characterised not only by hardware virtualisation, but also by the elasticity that virtualisation enables. As such, and based on discussions with real cloud operators, we considered it important to test the detection performance of our proposed technique in scenarios that utilise the elastic nature of the cloud. One such elasticity measure is VM "live" migration which allows real-time load balancing, failover and other resilience techniques to be employed towards improving server uptime on a physical host, as well as efficient operation of the services hosted on a given VM.

To test the SAE under scenarios with VM "live" migration, we deployed our SAE implementation on two compute nodes and configured them to use the same training dataset and algorithm parameters. The first experiment consisted of a normal phase lasting 10 minutes followed by an anomalous phase of a further 10 minutes, with live-migration scheduled halfway through the anomalous phase. The second experiment was conducted in a similar manner, but with migration scheduled halfway through the normal phase. Figures 10 and 11 exhibit the results for each experiment respectively.

In general, the results show that migration does not affect the performance of the detector at all due to the fact that each SAE is configured in exactly the same way, and thus detects anomalies with the same level of accuracy. The migration itself has no effect because it pauses the VM and reinstates it with exactly the same configuration on the new node; the actual downtime is not noticeable. The VM has fixed hardware parameters that must be satisfied on the new node if migration is to succeed, therefore since the migration occurred without issue the detection is able to continue, also without issue.

In Figure 11 the *pre-migration* results are missing values for precision, f-score and g-mean. This is simply a matter of divide-by-zero errors, caused by the detector not producing any true positives or false positives. The metrics cannot be calculated and are therefore absent from the chart.

No true positives were produced because the detector was in the normal phase and there were no anomalies to detect; no false positives were produced because the detector was performing particularly well. Hence, we can only determine values for the missing metrics *post-migration* when the experiment has entered the anomalous phase and

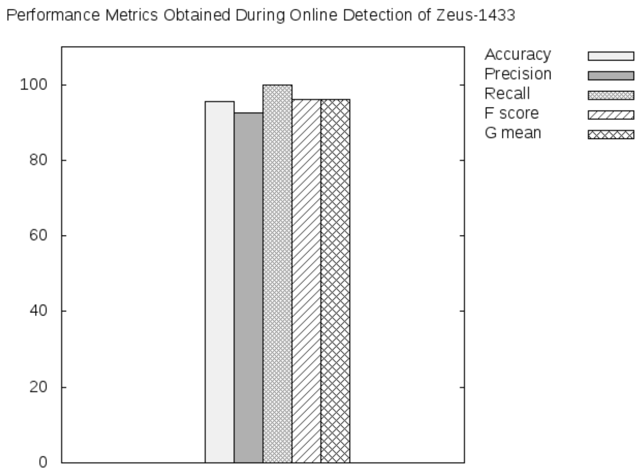

Fig. 8. Detection of Zeus-1433 using end-system features with a tuned classifier

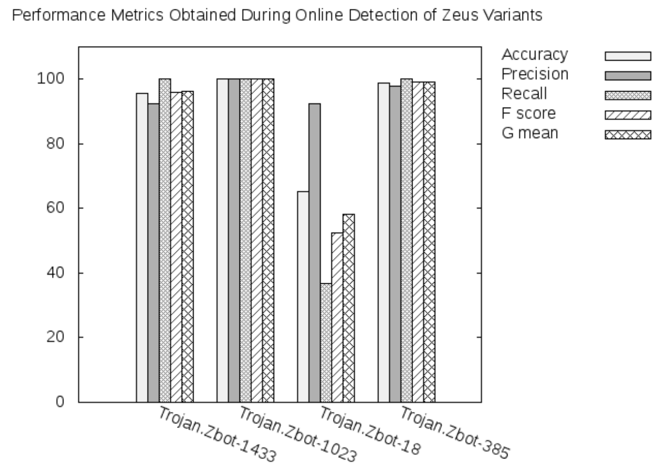

Fig. 9. SAE Detection of various Zeus samples using end-system data and a tuned classifier

true positives are once again a possibility. The *overall* results are the best measure of the performance of the SAE and were calculated by combining the results of both SAE components as if they had been produced by a single detector and, as is evident from the figure, once again reached accuracy levels well above 90%.

5.6 NAE Detection of Volumetric Network Attacks

Using a featureset that is capable of encapsulating changes to the volumetric properties of traffic on the network we were able to detect Denial of Service (DoS) attacks on the HTTP service running on a given VM using our NAE component.

The experiment consisted of the same VM as in previous experiments, running an HTTP server and serving clients with random data. The NAE collected network features in 8 second time bins under a normal period which lasted for 10 minutes. After the normal period the VM was attacked using the DoS traffic generator Low Orbit Ion Cannon (LOIC). The anomalies produced by this tool were detected by the NAE, which used SVM to compare new vectors with a dataset of normal samples. The output of the NAE was used to produce evaluation metrics according to the formulae in Equation 4.

The results in Figure 12 show that our choice of network features is appropriate and sufficient for detecting network based DoS attacks, since the accuracies obtained echo those of the SAE of well above 90%.

Although the SAE was not the subject of this experiment it is likely that the system metrics obtained by the SAE component would be impacted by the attack, if the attack had any impact on the VM's services. This would be the case if the DoS caused the server to spawn more processes/threads in order to meet demand, therefore skewing these particular features and their respective meta-features. Unfortunately, due to time constraints, it was not possible to confirm this.

5.7 Detection Using Joint vs. Network-Only Datasets

The previous experiment using network data involved the use of a volume-based external attack to test the detector. Figure 13 shows the results of two experiments to detect the anomalies produced by Zeus at the network level. As is evident from the figure the detector is not successful, using the features that are more suited to detecting DoS attacks rather than C&C communication. This is embodied in a result of less than 10% for recall, a measure of how well the detector can identify anomalies.

Experiments involving the detection of malware have so far been conducted in each domain separately. It is also possible, using the same techniques of feature extraction, to combine the features into a joint feature vector before analysis; that is, rather than analysing system and network-level features separately it is possible to combine them using a joint analysis approach. The experimental parameters of 10 minutes of normal activity followed by 10 minutes of anomalous activity were carried out as usual. However, the SAE classifier was trained using a dataset composed of vectors that were created from both network and system-level data. The evaluation metrics from Equation 4 were applied to the output in order to determine the detector's reliability under these new experimental conditions.

The results in Figure 13 show that overall detection is no more effective when system and network data are analysed together, in fact the performance is almost as poor as if the system-level data had been left out. The result for recall was improved to just over 10%, however this is still unacceptable. This is due to the fact that the network features chosen were not sufficient to detect network anomalies on their own, indicating that a larger number of statistical meta-features are needed.

Another possibility is the fact that the separate featuresets do not correlate sufficiently in order to be used effectively together. In parallel, this outcome also indicates that a joint dataset may not be useful in general due to the explicit algorithmic formulation of one-class SVMs. As we show in our other study in [8], the joint dataset was appropriate for use with the Empirical Mode Decomposition (EMD) algorithm, which performed well under the same constraints. This may be due to the formulation of EMD as a signal processing-based solution, which differs significantly to the

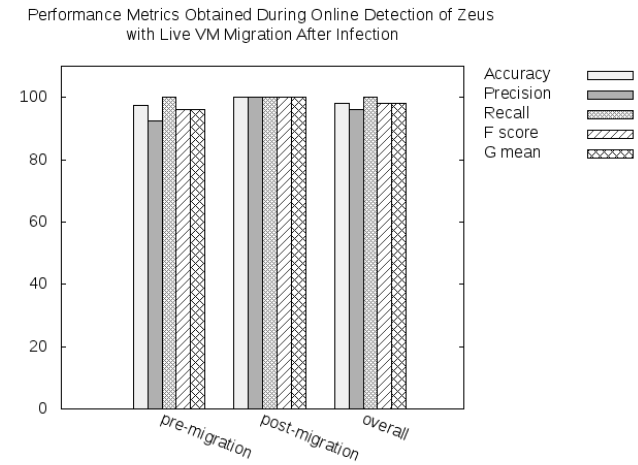

Fig. 10. Detection of Zeus-1433 with live migration occuring 5 minutes after the infection

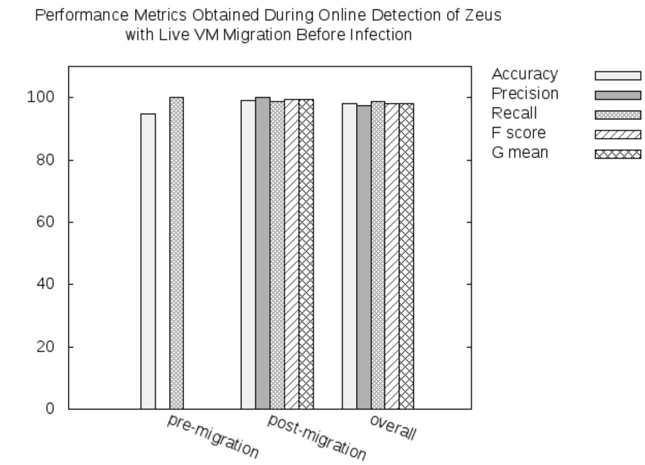

Fig. 11. Detection of Zeus-1433 with live migration occuring 5 minutes before the infection

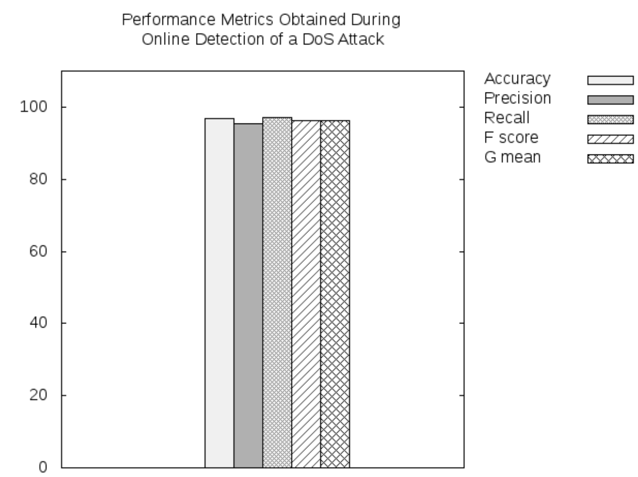

Fig. 12. Detection of a DoS attack by a tuned NAE component

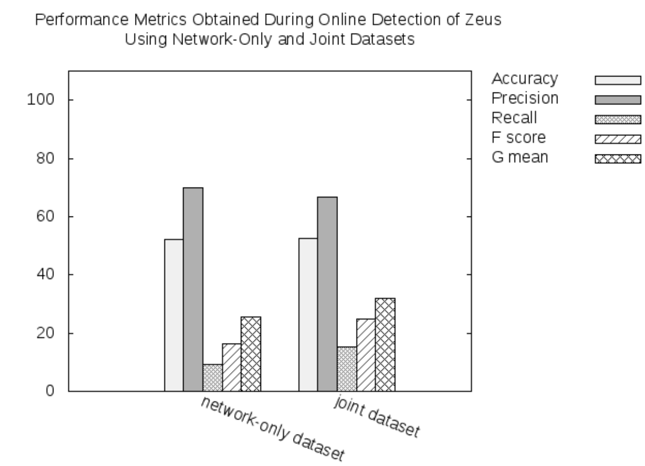

Fig. 13. Detection of Zeus-1433 using network-only and joint datasets with tuned classifiers

machine-learning approach of one-class SVM.

6 Conclusions

In this paper we introduce an online anomaly detection method that can be applied at the hypervisor level of the cloud infrastructure. The method is embodied by a resilience architecture that was initially defined in [4], further explored in [36], [37] and which comprises the System Analysis Engine (SAE) and Network Analysis Engine (NAE) components. These exist as submodules of the architecture's Cloud Resilience Managers (CRMs), which perform detection at the end-system, and in the network respectively. Our evaluation focused on detecting anomalies as produced by a variety of malware strains from the Kelihos and Zeus samples under the formulation of a novelty detector that employs the one-class Support Vector Machine (SVM) algorithm. Moreover, in order to empower the generic properties of our detection approach we also assess the detection of anomalies by the SAE and NAE during the onset of DoS attacks.

Overall, this work performs online anomaly detection under two pragmatic cloud scenarios, based on suggestions by cloud operators, which emulate "static" detection as well as detection under the scenario of VM "live" migration. The results obtained by strictly utilizing system-level data in our SAE detection, which was supported by an automatic SVM-specific parameter selection process, have shown excellent detection for all samples of malware under a variety of conditions (i.e. static and migration analysis) with an overall detection accuracy rate of well above 90%. Hence, we have demonstrated that the extracted features for classifier training were appropriate for our purposes and aided towards the detection of the investigated anomalies under minimal time cost throughout the training and testing phase. Nonetheless, in order to further the investigation, this featureset can easily be expanded to include statistics derived from vCPU usage and a deeper introspection of process handles, which could be beneficial for the detection of highly stealthy malware. However, the consideration of new features would naturally invoke a computational tradeoff, since deeper introspection will require more system resources.

The results derived from the experiments based on network-level detection of DoS attacks have also justified that the network features used were sufficient for the detection of such challenges, since the detection accuracy rate also reached well above 90%. However, when attempting to detect the examined Zeus and Kelihos malware samples using a strictly network-based featureset the gained results were inconclusive with low detection accuracy rates and unacceptable recall. In parallel, we have also observed minimal improvement in the evaluation metrics when considering a joint dataset, which was composed of both end-system and network level data. Hence, despite experiencing good results from the detection conducted using system-based features in the SAE we concluded that is not possible to improve the results obtained from the NAE through the combination of feature sets. Therefore, we demonstrate that by extending the featureset explicitly under the one-class SVM formulation would not necessarily lead to higher detection accuracy rates. However, as we show in our other work using the Ensemble Empirical Mode Decomposition (E-EMD) algorithm [8], a joint dataset could lead to good detection accuracy levels, thus we argue that the effectiveness of a featureset is strongly related with the exact mathematical formulation of a given detection algorithm.

In general, the detection approach presented in this paper is designed to be adaptive and respond to new threats and challenges online and in real time under minimal computational cost. Given the promising results presented through this work, we argue that our novel solution can overcome the commonly used signature-based intrusion detection solutions that are currently governing the domain of cloud security and further benefit cloud datacenter operations where security and resilience are of paramount importance.

Acknowledgements

This work is sponsored by UK-EPSRC funded EPSRC IU-ATC project, grant agreement no. EP/J016675/1 in collaboration with British Telecommunications (BT); and the EU FP7 Project SECCRIT (Secure Cloud Computing for Critical Infrastructure IT), grant agreement no. 312758.

References

[1] A. Marnerides, C. James, A. Schaeffer, S. Sait, A. Mauthe, and H. Murthy, "Multi-level network resilience: Traffic analysis, anomaly detection and simulation," ICTACT Journal on Communication Technology, Special Issue on Next Generation Wireless Networks and Applications, vol. 2, pp. 345–356, June 2011.

[2] J. P. G. Sterbenz, D. Hutchison, E. K. Çetinkaya, A. Jabbar, J. P. Rohrer, M. Schöller, and P. Smith, "Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines," Comput. Netw., vol. 54, no. 8, pp. 1245–1265, Jun. 2010. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2010.03.005

[3] A. K. Marnerides, M. R. Watson, N. Shirazi, A. Mauthe, and D. Hutchison, "Malware analysis in cloud computing: Network and system characteristics," IEEE Globecom 2013, 2013.

[4] M. R. Watson, N. Shirazi, A. K. Marnerides, A. Mauthe, and D. Hutchison, "Towards a distributed, self-organizing approach to malware detection in cloud computing," 7th IFIP/IFISC IWSOS, 2013.

[5] M. Garnaeva, "Kelihos/Hlux Botnet Returns with New Techniques." Securelist http://www.securelist.com/en/blog/655/Kelihos_Hlux_botnet_returns_with_new_techniques.

[6] H. Binsalleeh, T. Ormerod, A. Boukhtouta, P. Sinha, A. Youssef, M. Debbabi, and L. Wang, "On the analysis of the zeus botnet crimeware toolkit," in Privacy Security and Trust (PST), 2010 Eighth Annual International Conference on, Aug 2010, pp. 31–38.

[7] T. Brewster, "GameOver Zeus returns: thieving malware rises a month after police actions," Guardian Newspaper, 11, July, 2014, http://www.theguardian.com/technology/2014/jul/11/gameover-zeus-criminal-malware-police-hacking.

[8] A. K. Marnerides, P. Spachos, P. Chatzimisios, and A. Mauthe, "Malware detection in the cloud under ensemble empirical model decomposition," in Proceedings of the 6th IEEE International Conference on Networking and Computing, 2015.

[9] L. Kaufman, "Data security in the world of cloud computing," Security Privacy, IEEE, vol. 7, no. 4, pp. 61–64, July 2009.

[10] M. Christodorescu, R. Sailer, D. L. Schales, D. Sgandurra, and D. Zamboni, "Cloud security is not (just) virtualization security: A short paper," in Proceedings of the 2009 ACM Workshop on Cloud Computing Security, ser. CCSW '09. New York, NY, USA: ACM, 2009, pp. 97–102. [Online]. Available: http://doi.acm.org/10.1145/1655008.1655022

[11] N. Gruschka and M. Jensen, "Attack surfaces: A taxonomy for attacks on cloud services," in Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on, July 2010, pp. 276–279.

[12] Y. Chen, V. Paxson, and R. H. Katz, "Whats new about cloud computing security?" EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2010-5, Jan 2010. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-5.html

[13] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee, "Bothunter: Detecting malware infection through ids-driven dialog correlation," in Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium, ser. SS'07. Berkeley, CA, USA: USENIX Association, 2007, pp. 12:1–12:16. [Online]. Available: http://dl.acm.org/citation.cfm?id=1362903.1362915

[14] M. Bailey, J. Oberheide, J. Andersen, Z. Mao, F. Jahanian, and J. Nazario, "Automated classification and analysis of internet malware," in Recent Advances in Intrusion Detection, ser. Lecture Notes in Computer Science, C. Kruegel, R. Lippmann, and A. Clark, Eds. Springer Berlin Heidelberg, 2007, vol. 4637, pp. 178–197. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-74320-0_10

[15] C. Mazzariello, R. Bifulco, and R. Canonico, "Integrating a network ids into an open source cloud computing environment," in Information Assurance and Security (IAS), 2010 Sixth International Conference on, Aug 2010, pp. 265–270.

[16] S. Roschke, F. Cheng, and C. Meinel, "Intrusion detection in the cloud," in Dependable, Autonomic and Secure Computing, 2009. DASC '09. Eighth IEEE International Conference on, Dec 2009, pp. 729–734.

[17] A. Ibrahim, J. Hamlyn-Harris, J. Grundy, and M. Almorsy, "Cloudsec: A security monitoring appliance for virtual machines in the iaas cloud model," in Network and System Security (NSS), 2011 5th International Conference on, Sept 2011, pp. 113–120.

[18] B. Hay and K. Nance, "Forensics examination of volatile system data using virtual introspection," SIGOPS Oper. Syst. Rev., vol. 42, no. 3, pp. 74–82, Apr. 2008. [Online]. Available: http://doi.acm.org/10.1145/1368506.1368517

[19] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," ACM Computing Surveys (CSUR), vol. 41, no. 3, p. 15, 2009.

[20] A. Marnerides, A. Schaeffer-Filho, and A. Mauthe, "Traffic anomaly diagnosis in internet backbone networks: a survey," Computer Networks, vol. 73, pp. 224–243, 2014.

[21] C. Wang, K. Viswanathan, L. Choudur, V. Talwar, W. Satterfield, and K. Schwan, "Statistical techniques for online anomaly detection in data centers," in Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on. IEEE, 2011, pp. 385–392.

[22] C. Wang, V. Talwar, K. Schwan, and P. Ranganathan, "Online detection of utility cloud anomalies using metric distributions," in Network Operations and Management Symposium (NOMS), 2010 IEEE. IEEE, 2010, pp. 96–103.

[23] Q. Guan and S. Fu, "Adaptive anomaly identification by exploring metric subspace in cloud computing infrastructures," in Reliable Distributed Systems (SRDS), 2013 IEEE 32nd International Symposium on. IEEE, 2013, pp. 205–214.

[24] Q. Guan, S. Fu, N. DeBardeleben, and S. Blanchard, "Exploring time and frequency domains for accurate and automated anomaly detection in cloud computing systems," in Dependable Computing (PRDC), 2013 IEEE 19th Pacific Rim International Symposium on. IEEE, 2013, pp. 196–205.

[25] I. Cohen, J. S. Chase, M. Goldszmidt, T. Kelly, and J. Symons, "Correlating instrumentation data to system states: A building block for automated diagnosis and control." in OSDI, vol. 4, 2004, pp. 16–16.

[26] P. Bahl, R. Chandra, A. Greenberg, S. Kandula, D. A. Maltz, and M. Zhang, "Towards highly reliable enterprise network services via inference of multi-level dependencies," in ACM SIGCOMM Computer Communication Review, vol. 37, no. 4. ACM, 2007, pp. 13–24.

[27] C. Wang, "Ebat: online methods for detecting utility cloud anomalies," in Proceedings of the 6th Middleware Doctoral Symposium. ACM, 2009, p. 4.

[28] Y. Guan and J. Bao, "A cp intrusion detection strategy on cloud computing," in International Symposium on Web Information Systems and Applications (WISA), 2009, pp. 84–87.

[29] J.-H. Lee, M.-W. Park, J.-H. Eom, and T.-M. Chung, "Multi-level intrusion detection system and log management in cloud computing," in Advanced Communication Technology (ICACT), 2011 13th International Conference on. IEEE, 2011, pp. 552–555.

[30] H. S. Pannu, J. Liu, and S. Fu, "Aad: Adaptive anomaly detection system for cloud computing infrastructures," Reliable Distributed Systems, IEEE Symposium on, vol. 0, pp. 396–397, 2012.

[31] A. Marnerides, S. Malinowski, R. Morla, and H. Kim, "Fault diagnosis in {DSL} networks using support vector machines," Computer Communications, no. 0, pp. – , 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0140366415000080

[32] W.-H. Chen, S.-H. Hsu, and H.-P. Shen, "Application of svm and ann for intrusion detection," Computers & Operations Research, vol. 32, no. 10, pp. 2617–2634, 2005.

[33] Y. Tang, Y.-Q. Zhang, N. Chawla, and S. Krasser, "Svms modeling for highly imbalanced classification," Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, vol. 39, no. 1, pp. 281–288, Feb 2009.

[34] "India - uk advanced technology centre project," http://www.iu-atc.com/.

[35] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, "Support vector method for novelty detection." in NIPS, vol. 12, 1999, pp. 582–588.

[36] A. Marnerides, M. Watson, N. Shirazi, A. Mauthe, and D. Hutchison, "Malware analysis in cloud computing: Network and system characteristics," in Globecom Workshops (GC Wkshps), 2013 IEEE, Dec 2013, pp. 482–487.

[37] N.-U.-H. Shirazi, S. Simpson, A. Marnerides, M. Watson, A. Mauthe, and D. Hutchison, "Assessing the impact of intra-cloud live migration on anomaly detection," in Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on, Oct 2014, pp. 52–57.

Michael R. Watson is a final year Ph.D. student in the School of Computing and Communications at Lancaster University, UK. His research is mainly in the fields of malware detection and cloud computing and his interests include computer security, networking technologies and open source development and collaboration.

Noor-ul-hassan Shirazi is a research associate in the School of Computing and Communications at Lancaster University, UK. His current research focuses on anomaly based challenge detection techniques in elastic cloud deployment scenarios. His interests include security and resilience of computer networks and networked systems.

Angelos K. Marnerides is a Lecturer (Assistant Professor) in the School of Computing and Mathematical Sciences at Liverpool John Moores University, UK. His research interests include anomaly detection, network security, resilience and cloud computing. Prior to that he was a Research Associate in the Department of Computing and Communications at Lancaster University (2012–2014), a Postdoctoral Research Fellow in the Carnegie Mellon University – Portugal (University of Porto) post-doctoral scheme (2011–2012) and an Honorary Research Associate with the Department of Electronic and Electrical Engineering at University College London (UCL) (2012–2013). He obtained his M.Sc. and Ph.D. in Computer Science from Lancaster University in 2007 and 2011 respectively. For more information on selected papers and up to date activities please visit http://www.staff.ljmu.ac.uk/cmpamarn/.

Andreas Mauthe is Reader in Networked Systems and has worked in the areas of distributed and multimedia systems, and network management for over 20 years. He has lead research and development activities in academia as well as industry. Andreas' current research focus is in the area of autonomic networking, and resilient networks and systems (including Clouds). He has more than 90 peer-reviewed publications, is on the Editorial Board of ACM Multimedia Systems Journal, and serves as an expert and evaluator for the EC.

David Hutchison is Professor of Computing at Lancaster University and founding Director of InfoLab21. He has served on the TPC of top conferences such as ACM SIGCOMM, IEEE Infocom, and served on editorial boards of Springers Lecture Notes in Computer Science, Computer Networks Journal and IEEE TNSM, as well being editor of the Wiley book series in Computer Networks and Distributed Systems. He has helped build a strong research group in computer networks, which is well known internationally for contributions in a range of areas including Quality of Service architecture and mechanisms, multimedia caching and filtering, multicast engineering, active and programmable networking, content distribution networks, mobile IPv6 systems and applications, communications infrastructures for Grid based systems, testbed activities, and Internet Science. He now focuses largely on resilient and secure networking, with interests in Future Internet and also the protection of critical infrastructures including industrial control systems.