Towards a Distributed, Self-Organising Approach to Malware Detection in Cloud Computing

Michael R. Watson, Noor-ul-Hassan Shirazi, Angelos K. Marnerides, Andreas Mauthe and David Hutchison
School of Computing and Communications
Lancaster University
Lancaster, UK, LA1 4WA
Email: {m.watson1,n.shirazi,a.marnerides2,a.mauthe,d.hutchison}@lancs.ac.uk

Abstract — *Cloud computing is an increasingly popular platform for both industry and consumers. The cloud presents a number of unique security issues, such as a high level of distribution or system homogeneity, which require special consideration. In this paper we introduce a resilience architecture consisting of a collection of self-organising resilience managers distributed within the infrastructure of a cloud. More specifically we illustrate the applicability of our proposed architecture under the scenario of malware detection. We describe our multi-layered solution at the hypervisor level of the cloud nodes and consider how malware detection can be distributed to each node, removing the need for wholly centralised detection systems.*

I. Introduction

Cloud computing presents a number of unique security and resilience management issues, while at the same time offering new opportunities for better protection. The security issues are a direct consequence of the virtualisation of hardware within the cloud and include issues associated with coresidence and migration. However, virtualisation also provides the opportunity to observe the processes that execute within a virtual machine (VM) along with the immediate network view of that VM, all from one location in the architecture.

Cloud environments are in general made up of a number of physical machines in a data centre hosting multiple VMs that provide the actual resources for the cloud's services. The data centre has an internal network and is connected through one or more ingress/egress routers to the wider Internet. In order to provide resilience within a cloud environment it is necessary to observe and analyse both system and network behaviour, and to take remedial action in case of any detected anomalies.

Detection in this scenario has to happen at various points throughout the cloud infrastructure and hence the observation instances, or resilience managers, need to exchange information and co-ordinate a reaction to any observed anomalies. Since cloud environments are highly distributed structures with no prescribed hierarchy or fixed configuration the resilience managers need to have the ability to flexibly organise themselves taking into account architectural considerations as well as system state. Each resilience manager therefore needs to be a self-organising entity within a larger resilience management framework. They need the capability of acting autonomously but in a coordinated manner in order to maintain the overall system operability, even during a period of challenges.

In this paper we present the architecture of a Cloud Resilience Manager (CRM) and the overall architecture arising from a network of CRMs under the same conceptual autonomic properties followed by previous work [1, 2]. Overall, we discuss the self-organising aspect of each element and how each CRM interacts to form the overall resilience framework.

The rest of this paper is structured as follows: In Section II we cover the details of the resilience system and its individual components; Section III provides information on the self-organising aspect of the CRMs; we provide a use case in Section IV; and conclude the paper in Section V.

II. System Architecture

The overall system architecture can be seen in Figure 1 with *A* representing a single hardware node in the cloud. For simplicity only three nodes are shown and the network connections between each node have been omitted. Each node has a hypervisor, a host VM (or dom0 under Xen terminology [3]) and a number of guest VMs. Within the host VM of each node there is a dedicated CRM which comprises one part of the wider detection system. The internal structure of the CRM can be seen in more detail in *B*.

The software components within *B* are, in order: the Network Analysis Engine (NAE), the System Analysis Engine (SAE), the System Resilience Engine (SRE) and the Coordination and Organisation Engine (COE). The CRM on each node will perform local malware detection based on the information obtained from its node's VMs and its local network view; this is handled by the SAE and NAE components respectively. The SRE component is in charge of protection and remediation actions based on the output from the analysis engines (i.e. NAE and SAE). Such actions include destroying an infected VM or blocking information destined to a vulnerable VM, however this paper is not concerned with specifics relating to this component. Finally, the COE component coordinates and disseminates information between other instances and, in parallel, controls the components within its own node.

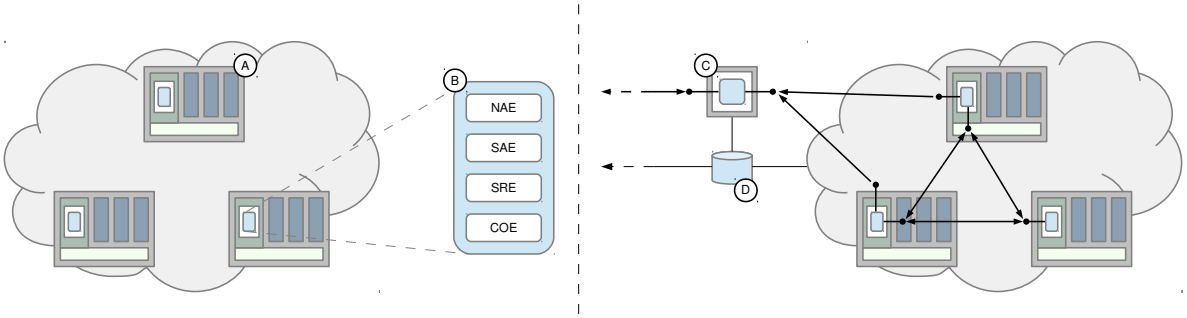

Figure 1: An overview of the detection system architecture

In addition to system level resilience, the detection system is capable of gathering and analysing data at the network level through the deployment of network CRMs as shown by *C* in Figure 1. *D* in the figure represents an ingress/egress router of the cloud; the monitoring system is directly connected to router *D* and as such can gather features from all traffic passing through it. The following subsections explain the operation of each CRM component in more detail.

A. Network Analysis Engine

The purpose of the Network Analysis Engine (NAE) is to categorise normal traffic at a cloud node and to subsequently identify anomalies in the traffic. This is achieved by capturing packets from the network interfaces of each VM and applying feature extraction and offline analysis. The goal is to work towards online feature extraction and analysis which would enable real-time anomaly detection.

The main function of the NAE is to relate observations from lower layers, such as the transport layer, with higher layers, such as the application layer. This approach, which looks at different aspects of network activity across each layer, will reveal commonalities that relate to the specific type of challenge.

A layer-by-layer approach allows the NAE to make use of application layer information to identify threats, such as attempted web application attacks, and subsequently review lower layer information related to the same threat. A correlation of anomalies that occur at different layers will increase the effectiveness of later correlation with system-level data.

The final correlation occurs in the COE by mapping statistical anomalies found in the network to end-system state as reported by the SAE. An example of this is the identification of protocols and ports used by anomalous traffic and the attribution of these to a particular process executing within a guest VM. In this way it is possible to attribute anomalies at disparate locations in the architecture to a single threat.

B. System Analysis Engine

In our architecture the end-system level detection is achieved by the System Analysis Engine (SAE). The SAE detects anomalies through the observation of VM memory from the hypervisor using the libVMI [4] introspection library. By acquiring VM memory at the hypervisor level it is possible to consolidate detection to a single point in the architecture of a cloud node.

The tool currently used to extract features from VM memory samples is Volatility [5], a memory analysis framework. Volatility provides the flexibility to write custom plugins, which allows raw features, such as process memory usage, to be extracted from a VM's processes. These are combined to form a feature vector for each process, which can then be analysed using statistical anomaly detection. The approach is similar to that employed in the NAE, but applied to system-level information. The first step is to determine what is normal behaviour with respect to the operation of processes, after which anomalous behaviour can be identified.

An issue with the current toolchain is its intensive use of resources; also the analysis is conducted offline, however, with further research it is anticipated that the feature extraction, pre-processing and analysis stages can be streamlined in order to speed up the process as a whole.

The next stage in our research regarding the NAE and SAE components is the selection of complimentary anomaly detection methods, such that their outputs can be easily combined in the COE.

C. System Resilience Engine

The System Resilience Engine (SRE) is designed to alleviate the Coordination and Organisation Engine (COE) of any responsibility regarding system state. This is due to the fact that the COE is already required to correlate the output of the NAE and SAE as well as coordinate with the other COEs within the cloud. The SRE therefore manages the resilience aspect of the end-system on behalf of the COE.

The SRE is required to interact with the VMs in order to achieve autonomous resilience. Such interactions include traffic filtering at the hypervisor level, process

sandboxing or termination, service or VM migration and, at the extreme, VM termination. All of these actions can proceed in parallel to anomaly detection and inter-CRM communication due to the separation of these responsibilities into a dedicated component.

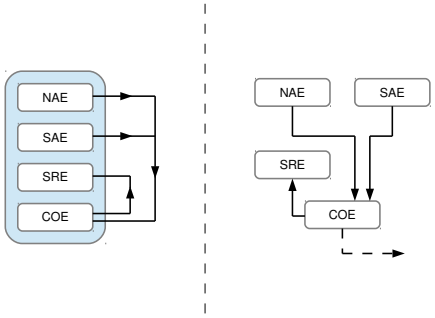

Figure 2: The flow of information and hierarchy of engines within a CRM

D. Coordination and Organisation Engine

The most important component of each CRM is the Coordination and Organisation Engine (COE) since it performs a critical role in the overall system architecture. The COE acts autonomously to control the other components within its CRM while at the same time allowing the system of CRMs as a whole to be self-organizing.

Figure 2 shows the relationship between CRM engines and the coordinating role that the COE plays. The dashed arrow in Figure 2 indicates communication between the local COE and the remote COE of a peer CRM.

Self-organization in a system of CRMs is achieved through the dissemination and exchange of meaningful information with respect to the system and network activities of each VM, as well as with the router(s) that connect the datacenter network to the Internet. In practice, and as depicted in Fig. 1, there are various system/network interfaces that act as information dispatch points [1] in order to allow efficient event dissemination. This concept is expanded on in the next section.

III. Resilience Manager Self-Organisation

A system of Cloud Resilience Managers (CRMs) is required to be self-organising in order to allow the system to make autonomous decisions regarding end-system and overall cloud resilience. This is achieved through an internal peer-to-peer network in combination with hierarchical interactions between internal and external network interfaces.

A. Network Architecture

In Figure 1 the system architecture is shown as consisting of two levels of communication. The interfaces between the CRMs within the cloud correspond to an internal peer-to-peer network where peers can perform push/pull actions with other peers. Peer communication is explained in the next subsection.

The interfaces between system level CRMs and the network CRM correspond to external interfaces that only allow push actions, from low level CRMs (i.e. system level) to higher level CRMs (i.e. network level). The reception of system or network-related events is managed by the COE on each node, which communicates under a common API with the other components of the CRM.

Due to the hierarchical nature of the system the information sent to D is under a filtered, event-based format resulting from the malware analysis and detection achieved internally by the SAE and NAE components. Thus, the COE in C will only receive meaningful input from a remote COE such that it is able to correlate the VM-level anomalous activity with the traffic it captures on the ingress/egress router(s). For instance if a destination within the cloud infrastructure is locally flagged as suspicious by its CRM, traffic to that destination can be thoroughly analysed by a detection component residing on C, as in [2].

The hierarchy between CRMs in the cloud access network and the CRMs within the cloud is intended to reduce complexity by only allowing information to flow from cloud to network. This means that network CRMs will not be in a position to notify individual COE instances; however, this is not anticipated to reduce resilience due to the ability of the network CRM to make decisions on behalf of the peers below it. For example, anomalous traffic destined for a particular VM within the cloud can be filtered out by the network CRM without the need to notify the COE corresponding to that particular VM.

B. Peer Communication

Peer-to-peer communication of data between CRMs enables each individual CRM to make local decisions which are influenced by the activity experienced on remote cloud nodes. This direct communication between peers results in the ability of the CRMs to exchange information with respect to the current health of their respective virtual environments.

The peer network itself is a simple message exchange system, whereby healthy peers advertise their presence in the network and nothing more. Peers experiencing anomalous behaviour exchange the type of anomaly and pertinent information on how this will affect other peers. An example scenario which highlights this behaviour is shown in Figure 3 and described in Section IV. The other COEs in the cloud will receive this data and take action by invoking their local SRE.

The benefits of this information exchange versus a centralised system are the ability to notify vulnerable systems in a single exchange, and the ability to reduce the amount of data flowing over communication channels. In a centralised detection system it would be necessary

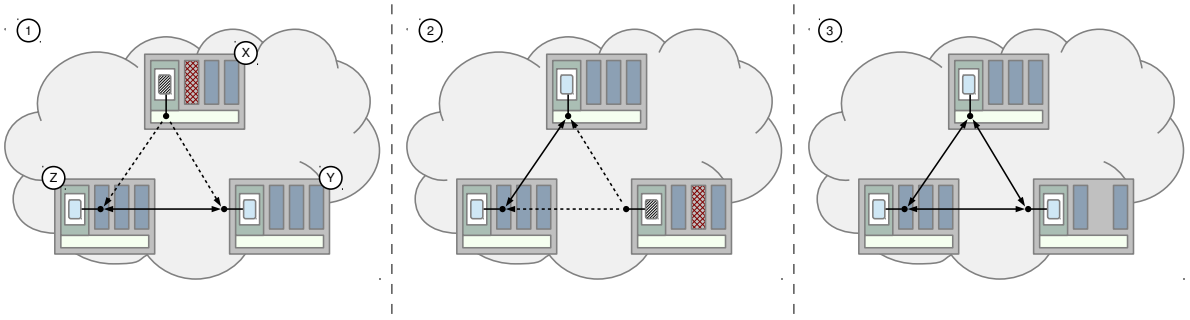

Figure 3: Message exchange and system state in the presence of malware during service migration

to export all data relating to the state of every physical machine in the cloud to a single point. This scenario puts a higher demand on network links than the solution proposed in this paper; only data relating to detected anomalous events needs to be transmitted. Moreover, a single point of analysis reduces the resilience of the cloud due to a single point of failure. This fact, coupled with the typically homogeneous internal topology of clouds indicates that self-organisation is a better fit architecturally.

IV. An Example Scenario

Service migration can have a significant negative effect on anomaly detection in a conventional cloud [6]. Our example scenario involves this type of challenge and is centred around the migration of a bot infected service.

In Figure 3, pane 1 shows node *X* hosting an infected service on one of its VMs. The CRM on *X* has detected this as an anomaly and communicates the type of anomaly to its neighbours, *Y* and *Z*. In this case the behaviour is attributed to a bot, which *X* communicates to its neighbours, along with specific threats posed to them such as target port numbers. The communication between *Y* and *Z* is the standard message exchange of normally operating nodes.

Pane 2 shows the cloud after the migration of the infected service from node *X* to node *Y*. The assumption here is that the SRE on node *X* was unable to respond before the service was migrated. Possible actions available to the SRE include blocking the request to migrate due to the presence of anomalous behaviour. In this case, however, it can be assumed this was not possible and it is now the responsibility of *Y*'s CRM to signal the anomaly to its neighbours. The message exchange between *X* and *Z* has returned to normal.

Between stages 1 and 2 the COE of *X* communicates to the COEs of *Y* and *Z* which specific service was infected. This allows *Y* to anticipate the reception of an infected service and deal with it immediately instead of following an identical detection process to that conducted by *X*.

In pane 3 the SRE of *Y* has determined that the only action suitable in this situation is to terminate the offending VM. The messages exchanged between all nodes has therefore returned to normal.

V. Conclusion

Cloud environments present unique challenges in terms of security and resilience. These challenges need to be confronted through the synergistic analysis of both system and network-level properties in order to more effectively utilise available information. In this paper we have proposed a solution to these challenges by introducing the concept of a Cloud Resilience Manager (CRM) which combines self-organisation with a distributed approach to detection.

Acknowledgments

The authors would like to thank Fadi El-Moussa and Ben Azvine of BT Research for their contribution to this work. This work is sponsored by the IU-ATC (EPSRC grant number EP/J016675/1) and the EU FP7 SECCRIT project (grant agreement no. 312758).

References

[1] A. K. Marnerides, D. P. Pezaros, and D. Hutchison. *Detection and mitigation of abnormal traffic behaviour in autonomic networked environments*. In *Proceedings of ACM SIGCOMM CoNEXT Conference 2008*.

[2] A.K. Marnerides, D.P. Pezaros, and D. Hutchison. *Autonomic diagnosis of anomalous network traffic*. In *Proceedings of IEEE WoWMoM 2010*.

[3] Citrix Systems, Inc. Xen. `http://www.xen.org/`.

[4] Brian D. Payne. LibVMI:. `http://code.google.com/p/vmitools/wiki/LibVMIIntroduction`.

[5] Volatility tool:. The Volatility Framework. `https://www.volatilesystems.com/default/volatility`.

[6] K. Adamova. Anomaly Detection with Virtual Service Migration in Cloud Infrastructures. Master's thesis, Swiss Federal Institute of Technology, Zurich, 2013.