Malware Analysis in Cloud Computing: Network and System Characteristics

Angelos K. Marnerides, Michael R. Watson, Noorulhassan Shirazi, Andreas Mauthe, and David Hutchison
InfoLab21, School of Computing & Communications, Lancaster University, UK
{a.marnerides2,m.watson1,n.shirazi,a.mauthe,d.hutchison}@lancaster.ac.uk

***Abstract*—The deployment of cloud computing environments is increasingly common, and we are implicitly reliant on them for many services. However, their dependence on virtualised computer and network infrastructures introduces risks related to system resilience. In particular, the virtualised nature of the cloud has not yet been thoroughly studied with respect to security issues including vulnerabilities and appropriate anomaly detection. This paper proposes an approach for the investigation and analysis of malware in virtualised environments. We carry out an analysis, on a system and network-wide scale, and further pinpoint some system and network features specifically by studying the example of the Kelihos malware.**

I. Introduction

The cloud presents a number of unique security issues and an emerging critical aspect is related to the adequate identification and detection of malware. Malware is, in the majority of cases, the first point of initiation for larger security threats such as distributed denial of service attacks (e.g. DDoS); thus its immediate detection is of crucial importance. Furthermore, the virtualised nature of cloud environments is a possible additional vulnerability with respect to malware activity, due to the various operations allowed by virtualised systems such as service migration and virtual machine (VM) co-residence. 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 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 the case of any detected anomalies. Particularly for malware detection it is of vital importance to perform a fine-grained observation of system-specific properties. At the same time a careful analysis of the network traffic traversing an individual node is required. Therefore, in case of malware activity it is crucial first to identify the most representative features that may be incorporated within a unified detection scheme that covers both the system and the network view.

Threat analysis and anomaly detection in cloud environments have been addressed in other pieces of work [1], [2], [3], [4], [5], [6]. However, we argue that most of the proposed solutions do not provide any clear understanding with respect to particular types of malware and also do not provide direct relationship of the network traffic and the local system behaviour. In [1] the authors propose a cloud security monitoring system, which automatically detects and makes a response to anomalies that occur inside a cloud infrastructure using the IP Flow information export method. However, this paper purely addresses network security threats and issues that may occur in virtual infrastructure clouds that use shared hardware, network and hypervisor resources. Molna and Schechter in [2] analyse security threats related to hosting applications in a cloud infrastructure. In [5], the authors analyze network traffic and observe specific changes of flow data with the intention to extract anomalies. In particular their work is dependent on a rule-based deep packet inspection (DPI) scheme subject to analysing NetFlow traces using MapReduce.

Overall, most of the detection techniques employed for cloud computing infrastructure are at individual layers and mostly independent of each other [3], [4]. Therefore this paper provides a continuation of our previous work in [7] and illustrates the internal analysis performed within two main components of our proposed resilience approach. We use the example of the Kelihos malware, which is considered as an emerging threat in today's networks. Thus, we pinpoint the most important system features to consider for the detection process of this malware and further illustrate network meta-statistics that can be useful within a detection scheme. Our hypothesis is that the energy computed for the first ten packets of a given unidirectional flow based on the Choi-Williams Time Frequency distribution [8] constitutes a good discriminating feature within a covariance analysis over all the network features that we employ. The intuition behind using the Choi-Williams distribution relates with the fact that it has shown significant accuracy outcomes in the area of Internet traffic classification [12]. In parallel, the explicit observation of the distributional behaviour of the first five to ten packets within a given flow has proven to be extremely beneficial in the area of traffic characterisation as exhibited in [18]. Thus, under our controlled experimental scenario a major decline of the representative energy for the sizes of the first ten packets during Kelihos infection is noticeable in the network trace; and a ramp up of energy is observable during its establishment.

The remainder of this paper is structured as follows: Section II briefly introduces our resilience approach. Section III describes the operations and methods invoked by the two system and network analysis engines. A description of the Kelihos malware is provided at Section IV where also the experimentation setup of this work is presented. Section V is dedicated to illustrating the results attained from our evaluation

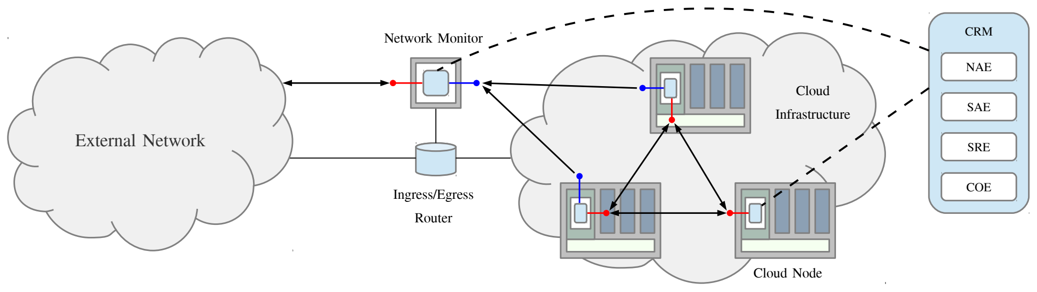

Fig. 1. System Overview

whereas Section VI summarises and concludes this work.

II. Resilience Approach

The overall architecture of our approach can be seen in Figure 1. For simplicity only three nodes are shown and the network connections between nodes are omitted. Each node has a hypervisor, a host VM (or dom0 under Xen terminology [10]) and a number of guest VMs. Within the host VM of each node there is a dedicated Cloud Resilience Manager (CRM) which comprises one part of the wider detection system. The software components within the CRM are 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. Finally, the COE component coordinates and disseminates information between other instances and, in parallel, controls the components within its own node. 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 Monitors each containing a network CRM as shown in Figure 1. This monitoring system is directly connected to each Ingress/Egress Router as shown, and can gather features from all traffic passing through it. As already mentioned, this work is explicitly addressing the operations of the NAE and SAE engines using the example of characterising the Kelihos malware.

III. Methodology & Feature Composition

A. Network Analysis Engine (NAE)

Due to the binary format of the captured packet data, the network analysis engine is composed of several scripts that allow automated post-processing by calling tcpdump filtering routines [11] alongside several precompiled tools that are integrated within the CAIDA's CoralReef toolbox [9]. CoralReef is a software suite developed by CAIDA, which consists of several tools, both command line tools and libraries. These tools provide surplus of information about several properties of traffic with respect to several aspects (e.g. routing, application layer classification etc.) on either a packet, flow or flow aggregate viewpoint. Due to the high-dimensionality of these features we have developed a set of scripts that filter and extract relevant flow-based fields from CoralReef, and further feed the data to MATLAB in order to perform the algorithmic part of the NAE. This suite of shell, Perl and Python scripts perform various operations and they are in a position to ensure an automated feature extraction from the outputs of the CoralReef tools. Their operation spans into several post processing outputs that convert STDIN input into various normalised statistical properties on a per-packet (e.g. inter-arrival times, etc) and per-flow basis (e.g. counts of packets/bytes pert flow, average packet size etc.) which also have an interface with MATLAB functionality.

From an algorithmic point of view the flow-based analysis performed on the captured network data relies on the covariance analysis that has been proven as useful in other anomaly detection frameworks [17]. Nevertheless, in order to ensure an accurate extraction of an anomaly in our datasets it was initially of great importance to identify the most meaningful flow features. Apart from the standardised volume-based traffic statistics (e.g. counts of bytes, packets, etc.) we have also included meta statistics that describe the statistical characteristics of the first ten packet sizes since their applicability has been demonstrated in work related with traffic classification [18]. In particular, we have estimated the overall energy representation derived by the first ten packets and included them in our feature vectors for each TCP or UDP flow since they empower a far more accurate statistical description as presented in [12]. The energy representation is also evident to be extremely useful for characterising the activities of the Kelihos malware as we show in Section V-B2. In the following section we present the definitions and describe how they were incorporated in our covariance-based analysis.

1) Energy estimation under the Choi-Williams Distribution: The energy representation for the first ten packets of a given flow, could have not been easily extracted without the initial characterisation by an appropriate probability distribution function. Given the promising outcomes of the Cohen-based Time-Frequency (TF) Distributions [8] on Internet traffic classification [12], our approach targeted at the estimation of these metrics. Thus, we have initially achieved a general statistical description of TCP and UDP flows based on a special formulation derived by the general Cohen class of energy Time-Frequency distributions, the Choi-Williams distribution since it has shown much better performance with respect to accuracy than other Cohen-based formulations [12]. As already mentioned, we are concerned with the size of the first ten packets on a given TCP or UDP flow since they are proven to be quite good "raw" descriptors for a given flow and their meta-statistics (e.g. sum, standard deviation) have been used with success in [18] for flow characterization purposes. Therefore, from now on we consider the series of the first ten packets for a given flow as a signal $s(t)$ where its analytical form after applying a Hilbert transformation is denoted as $s(u)$.

The generic Cohen distribution alongside its extended formulations [8] enable to represent energy as a joint function of time and frequency. The aim is to obtain energy E under a description of a joint probability function $M(t, \omega)$ that provides the intensity of a signal at time t with frequency ω as follows:

$$E = \int M(t, \omega) d\omega dt \tag{1}$$

Equation 1 provides an intuitive representation of the energy representation that Cohen in [8] tries to interpret with his general formulation that we do not present here. Under the general Cohen distribution [8] that emphasised upon the description of $M(t, \omega)$, there were several efforts that try to eliminate *auto* and *cross-terms* from the energy distribution of each signal on the Time-Frequency (TF) plane. In practise, this additive interference terms reveal high levels of noise that lead to inaccurate transformation and mapping of the signal on the TF plane. In our case, this is manifested as inaccuracies on the mapping of the packet size measurement with respect to time for each flow on the TF domain. The Choi-Williams distribution, tries to confront the artifacts caused by auto and cross-terms by introducing a new kernel function $\phi(\theta, \tau) = e^{-\theta^2 \tau^2 / \epsilon}$ and considers our signal $s(t)$ to be constructed by a number of components such as $s(t) = \sum_{a=1}^{N} s_a(t)$ and is defined as follows:

$$CW(t, \omega) = \frac{1}{4\pi^{\frac{3}{2}}} \iint \frac{1}{\sqrt{\tau^2/\epsilon}} e^{-[(u-\tau)^2/(4\tau^2/\epsilon)] - j\tau\omega} .s^*(u - \frac{1}{2}\tau).s(u + \frac{1}{2}\tau) du d\tau \tag{2}$$

2) Feature Set: As already mentioned the energy computation was one of the features the describe a given unidirectional flow. Overall, the feature set for each flow was determined by the following 13 features:

- Protocol identifier (i.e., TCP, UDP, ICMP)
- Source Port
- Destination Port
- Number of bytes
- Number of packets
- Mean packet size
- Packet size standard deviation
- Mean packet inter-arrival time
- Packet inter-arrival time standard deviation
- Skewness of the first 10 packet sizes
- Flow Duration
- Energy for the size distribution of the 10 first packet sizes
- Sum of the first 10 packet sizes

These features were gathered for all the captured flows and aggregated into 3 equally sized 20 minute time-bins and used within the covariance analysis technique that we following describe.

3) Covariance Analysis: The employed covariance analysis had the intention to pinpoint the most crucial network-oriented features with respect to the initiation and establishment of the Kelihos malware on a given VM. Let p_β be the number of a features that compose a random vector that represents the observations of a flow x in such way that $x = (p_1, \cdots, p_\beta)$. Also, let $x_1, \cdots, x_n$ be the number of flows observed such as $x_i = p_1^i, \cdots, p_\beta^i$ where $p_i^{\tau,j}$ denotes the value of p_i in the j^{th} observation on the time instant τ within the time period T_τ which in our case was the total of 60 minutes. Given these definitions we aim to describe the observation of all the traffic flows in the matrix X_τ as follows:

$$X_\tau = \begin{pmatrix} p_1^{\tau,1} & \cdots & p_\beta^{\tau,1} \\ p_1^{\tau,2} & \cdots & p_\beta^{\tau,2} \\ \vdots & \ddots & \vdots \\ p_1^{\tau,n} & \cdots & p_\beta^{\tau,n} \end{pmatrix} \tag{3}$$

Where X_τ has a covariance matrix Σ_{X_τ}:

$$\Sigma_{X_\tau} = \begin{pmatrix} \sigma_{p_1^\tau p_1^\tau} & \sigma_{p_1^\tau p_2^\tau} & \cdots & \sigma_{p_1^\tau p_\beta^\tau} \\ \sigma_{p_2^\tau p_1^\tau} & \sigma_{p_2^\tau p_2^\tau} & \cdots & \sigma_{p_2^\tau p_\beta^\tau} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{p_\beta^\tau p_1^\tau} & \sigma_{p_\beta^\tau p_2^\tau} & \cdots & \sigma_{p_\beta^\tau p_\beta^\tau} \end{pmatrix} \tag{4}$$

And we further define the distance variable D_τ as the Eucledian distance between two matrices Σ_{X_τ} with the mean of Σ_{X_τ}, $\mu(\Sigma_{X_\tau})$:

$$D_\tau = \| \left(\Sigma_{X_{T_1}} - \Sigma_{X_{T_2}} \right) - \mu(\Sigma_{X_\tau}) \| \tag{5}$$

In practise, D_τ determines the anomalous activity that persists within the gathered flow features. As we present in section V, our experimentation investigated the difference between the covariance matrices produced for each of the 20 minute time bins in order to identify the discriminative network-oriented features that can be exploited for the characterisation and further detection of the Kelihos malware.

B. System Analysis Engine (SAE)

Data gathering from each VM was achieved through a synergistic deployment of several Python scripts that provided automation by invoking the libVMI library [15], the Volatility memory analysis tool [16] and tcpdump [11]. The libVMI introspection library allows access to the virtual memory of the VM during runtime and, through the use of Volatility, features are extracted from the memory relating to processes within the OS, such as process ID (PID), thread count, virtual size, peak virtual size, etc. Finally, tcpdump allows raw network packets to be captured at the virtual interface of the VM. Overall, there is a main monitoring script which subsequently invokes two sub-scripts, one to monitor the network and one to monitor the end-system behaviour through memory extraction.

The network script obtains a timestamp from the dom0 OS and creates a file. The file is further used for the output of tcpdump, which is configured to capture from the virtual interface of the VM with a snaplength of 65535 bytes to ensure full packets are captured. Once tcpdump is running, the script sleeps for 60 minutes after which time the tcpdump process is killed.

In parallel, the system script performs a similar sequence of commands to the network script with the addition of a loop due to the one shot nature of libVMI's memory extraction. The loop is entered, a timestamp is taken from the dom0 OS and a related file is created. Subsequently, volatility is invoked and is pointed at the address space provided by libVMI through its pyVMI Python compatibility layer. Volatility is configured to use our custom plugin for feature extraction and the output is directed at the newly created output file. Moreover, the current time is obtained from the dom0 OS and the loop starts from the beginning. As soon as the difference between the current time and the start time of the experiment is more than 60 minutes any processes relating to memory extraction are killed and the script exits.

IV. Kelihos & Experimental Setup

A. Kelihos malware

The sample used in our work is known as "Kelihos" and is considered to be an emerging threat as highlighted by various researchers [13]. The research community argues that Kelihos (also known and Hlux) is a replacement of the famous Storm worm, which was active in 2007 and replaced by Waledac in 2009. In [14] authors gave an overview of the functionality of Kelihos and exposed its peer-to-peer network protocol. It is highlighted that a new variant is capable of sophisticated evasion techniques both in malware and network components. In particular, it monitors network traffic for HTTP and FTP to steal sensitive information and propagates through TCP port 80 where data exchange is completely encrypted. Moreover, the initial infection vector used by Kelihos is through propagation of malicious links embedded in emails. These links are based on domains controlled by the bot master and generally use fast flux techniques[14].

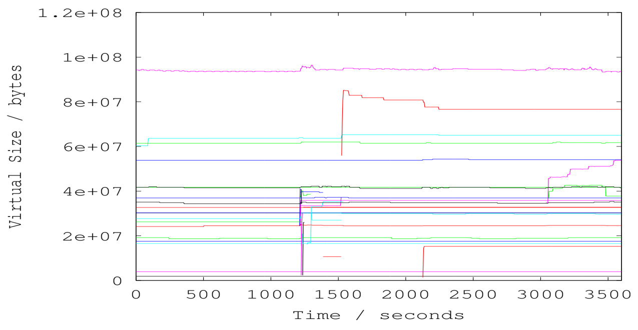

Fig. 2. Virtual Size of VM Processes Over Time

B. Testbed

The testbed is composed of a single physical node running multiple VMs. The host runs Xen 4.1 with the XAPI toolstack and Ubuntu 12.10 Linux as the dom0 operating system. The VMs used for testing were running Windows XP (SP3), two Iperf clients (one TCP, one UDP) and the Kelihos malware strain *Trojan.Kelihos-5*. The experiment lasted for 1 hour and had four separate phases: the set-up phase, normal operation phase, infected phase and tear-down phase. In the set-up phase the VM was started and the Iperf clients were initiated for traffic generation. This involved setting up two server instances on a remote machine and starting the clients, one in TCP mode with default settings and one in UDP mode with a target throughput of 10MB/sec and a packet size of 1470. In parallel the monitoring scripts were readied for execution in dom0.

In the normal operation phase we made a note of the time and executed the monitoring scripts such that the tcpdump captures and libVMI memory dumps were synchronised. This phase lasted for 20 minutes, during which time the network interface was monitored continuously and memory dumps were taken as often as Volatility would allow, which was approximately every 3 seconds. The infected phase began twenty minutes after the noted start time and involved the injection of the Kelihos trojan. This phase lasted for 40 minutes with monitoring continuing as in the previous phase. The monitoring scripts were set to perform a self tear-down after 1 hour, which brought the experiment into the tear-down phase. The VM was forced to shutdown and the data was checked to make sure it had been recorded correctly.

V. Results

A. System Snapshot

Fig. 2 shows the virtual size (or size in virtual memory) of each process within a given Windows VM as produced by the SAE. For the first 20 minutes of feature extraction the only processes running were those started by the OS, along with the two Iperf clients and an *explorer.exe* process. After the first 20 minutes, the Kelihos sample was injected and, as Fig. 2 shows, a number of new processes began to execute. Since the only change in the system is the presence of malware, it can be concluded that all new processes are linked to the activity of

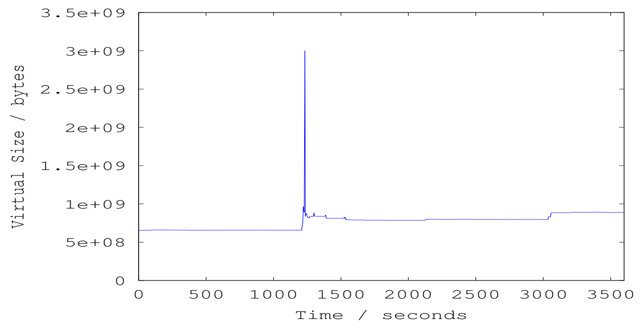

Fig. 3. Total Virtual Size of VM Processes Over Time (each line represents a different VM process).

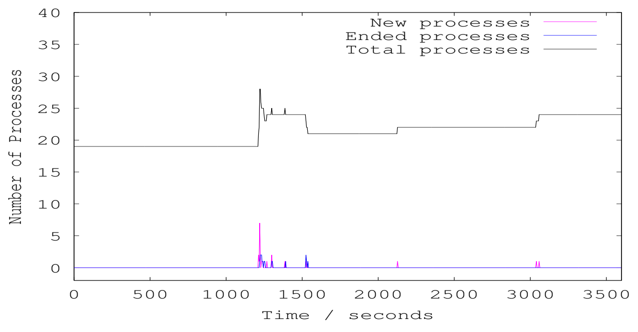

Fig. 4. Creation and Destruction of Processes Over Time

the Kelihos sample. The large number of processes present in the VM makes Fig. 2 difficult to interpret; however, by totalling the virtual size of all processes it is possible to see the effect of malware execution a little more clearly.

Fig. 3 shows a large increase in overall memory usage just a few seconds after Kelihos was injected, followed by a return to lower levels of memory usage. The memory usage before injection was around 700MB with a peak usage after injection of 3GB.

Huge differences like this are not usual and would normally be caused by processes that make extensive use of swap space or have a memory leak due to a bug or configuration error. In both cases the spike would be expected to last for a longer period of time than just a few seconds. By looking more closely at the number of processes over time it is possible to go some way towards explaining the memory usage spike. Fig. 4 shows that at the time of injection the total number of processes within the VM increases from 19 to 28. However, this increase of almost 50% in the number of processes corresponds to a 320% increase in memory usage, which is much higher than would be expected of such short lived processes. Closer inspection of the raw dump files revealed that the memory spike was caused by a single process consuming over 2GB of virtual memory.

B. Network Snapshot

1) NAE Characterization:: The NAE outputs show that for full sixty minutes record, the most common packet size is between 1400 and 1500 bytes, with total number of 14904706 packets, or 80% of the packets overall.

This huge peak was caused due to bulk transfer of fixed length. Thus, over 80% of the traffic in the traffic trace , are more between 1400-1500 bytes ,and the remaining 20% of the packets are between 0-1200 bytes long. Our initial characterization leaned towards the investigation of the packet and flow rate per second for the full duration of the experiment (i.e. 60 minutes). As evidenced by Fig. 5 , there is an initial ramp-up packet rate around the first 100 seconds that is mainly due to the triggering of Iperf which then reaches a stable state up to the end of the fist time bin (i.e. 1200 seconds). At the point where the Kelihos malware is injected we witness a sharp decrease on the packet rate which based on the system analysis provided earlier we assume that this is due to the exhaustion of some core OS processes (e.g. explorer.exe) that partially control networking functionality.

On the other hand and within the same time frame, the per-flow rate is still maintained at a reasonably stable shape. However, after an extremely small amount of time ($\approx 300 seconds$) there is a visible increment on the flow rate. Given the stable characteristics of the packet rates, our cross examination with the raw data revealed that the malware following its initial injection phase was subsequently placing an effort to lookout for nearby peers by sending multiple flows with minimal number of packets. The output of Fig. 5 also shows that this behaviour was persistent up to the end of the second time bin (i.e. 2400 seconds) but due to the controlled experimental environment that we have deployed, the malware could not initiate any connection with nearby VMs. Hence, in the final time bin the flow rate profile had a close to stable behaviour with respect to its mean flow rate.

2) Covariance Analysis & Comparison:: Having as a basis the definitions provided in Section III-A3 and under analysis of the post-processed information provided by the NAE engine we placed an effort at identifying the most discriminant network-based feature with respect to the initiation and establishment of the Kelihos malware. Simply enough, a comparison between the three produced covariance matrices for each time bin would provide a statistical description regarding the impact and correlations of the feature sets gathered for each flow.

As already mentioned, the feature set for each observed flow went beyond the traditional raw flow features but also contained some meta-statistics based on the distribution of the first 10 packet sizes for each unidirectional flow. The purpose of Fig. 6 is to emphasise the direct impact on the energy metric that was derived by the Choi-Williams distribution given the packet size rate of the first 10 packets in a flow. In particular, the comparison between the eigenvectors of the covariance matrices for the first two time bins have shown a constant but distant behaviour between their features. However, the sharp decrease on the energy metric (i.e. 12^{th} feature on the x-axis)

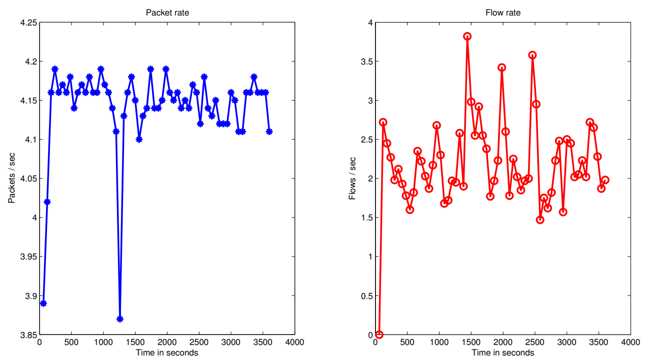

Fig. 5. Packets (left) and Flow (right) per second for the full duration of the experiment

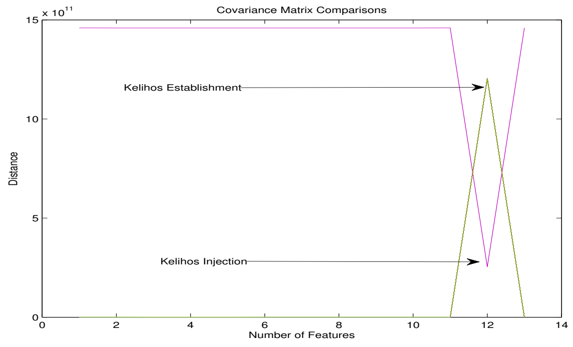

Fig. 6. Results of covariance analysis indicating the behaviour of network flow-based features.

obtained during the phase of Kelihos injection (i.e. the second time bin) indicates an anomalous pattern in comparison with the behaviour of the other compared features. This observation is also justified on a second comparison between the second and the third time bins where a sharp increase is credited again to the energy metric. Overall, the high distance difference denoted on the energy metric is directly related with the establishment of the Kelihos malware.

VI. Conclusion

Virtual infrastructures constitute the basis of cloud environments and due to their intrinsic operations are prone to malware. Hence, this paper focuses on tackling malware characterisation from a systems and network perspective within a controlled and experimental virtual environment. Our analysis specifically looks at Kelihos within the context of a systematic resilience approach. The focus of this work is the main operations and outputs employed by the System Analysis Engine (SAE) and the Network Analysis Engine (NAE) that reside within our resilience architecture. Through their outputs the most important system and network-based features can be pinpointed; these should be considered for the characterisation of this particular malware. Further, we propose the energy metric as a promising feature for the network-based detection of Kelihos. However, this study has produced some interesting outcomes that will stimulate further research on the detection of malware in the cloud, which has not been thoroughly addressed by the research community up to this point.

Acknowledgments

This work is sponsored by the UK-EPSRC funded IU-ATC project (grant number EP/J016675/1) and the EU FP7 SECCRIT project (grant agreement no. 312758).

References

[1] Mukhtarov M., Miloslavskaya N., Tolstoy A., Cloud Network Security Monitoring and Response System, The Third International Conference on Cloud Computing, GRIDs, and Virtualization, 2012
[2] Molnar D., Schechter S., Self Hosting vs. Cloud Hosting: Accounting for the security impact of hosting in the cloud, in WEIS 2010
[3] A. V. Dastjerdi, K. A. Bakar, and S. G. H. Tabatabaei, Distributed Intrusion Detection in Clouds Using Mobile Agents, In 3^{rd} ADVCOMP-09, pp. 175-180, Sliema, October 11-16, 2009.
[4] C. Mazzariello, R. Bifulco, and R. Canonico, Integrating a Network IDS into an Open Source Cloud Computing Environment, in 6th IAS-10, pp. 265-270, Atlanta, GA, August 23-25, 2010.
[5] Zhang L., Wang J., Lin S., Design of the Network Traffic Anomaly Detection System in Cloud Computing Environment, In 4^{th} International Symposium on Information Sciences and Engineering 2012.
[6] J. Xu, J. Yan, L. He, P. Su, and D. Feng, CloudSEC: A Cloud Architecture for Composing Collaborative Security Services, In IEEE CloudCom 2010
[7] Watson, M. R., Shirazi, N., Marnerides, A. K., Mauthe, A., Hutchison, D., Towards a Distributed, Self-Organizing Approach to Malware Detection in Cloud Computing, in 7^{th} IFIP/IFISC IWSOS 2013
[8] Cohen L., Time-Frequency Distributions - A Review, in Proceedings of IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. 77, No 7, pp 941-981, July 1989
[9] CAIDA CoralReef, tool, http://www.caida.org/tools/measurement/coralreef/
[10] Citrix Systems, Inc. Xen. http://www.xen.org/.
[11] Tcpdump, http://www.tcpdump.org/
[12] Marnerides, A. K., Pezaros, D., P., Kim, H., Hutchison, D., Internet Traffic Classification using Energy Time-Frequency Distributions, in IEEE ICC 2013, Budapest, Hungary, 2013
[13] Garnaeva, M. "Kelihos/Hlux Botnet Returns with New Techniques." Securelist, http://www.securelist.com/en/blog/655/Kelihos_Hlux_botnet_returns_with_new_techniques.
[14] Bureau, P., Same botnet, same guys, new code, in proceedings of Virus Bulletin VB2011.
[15] LibVMI: http://code.google.com/p/vmitools/wiki/LibVMIIntroduction
[16] Volatility: https://www.volatilesystems.com/default/volatility.
[17] Yeung D. S., Jin S., Wang X.: Covariance-Matrix Modeling and Detecting Various Flooding Attacks. IEEE Tran. Systems, Man and Cybernetics, Part A, vol.37, no.2, 157-169 (2007)
[18] Bernaille, L., Teixeira, R., Akodkenou, I., Soule, A., Salamatian, K., Traffic Classification on the Fly, in ACM SIGCOMM CCR, vol. 36, pp:23-26, 2006