Analysis of Mobility Algorithms for Forensic Virtual Machine Based Malware Detection

Nada Alruhaily, Behzad Bordbar, and Tom Chothia
School of Computer Science
University of Birmingham
Birmingham, UK B15 2TT
Emails: {Nma012, B.Bordbar, T.P.Chothia}@cs.bham.ac.uk

***Abstract*—Forensic Virtual Machines are a new technology that replaces signature-based malware detection for the cloud. Forensic Virtual Machines are mini-VMs which are used to identify symptoms of malicious behaviour on customer VMs. Scanning using these mini-VMs consumes less resources than a full scan would and their small size reduces the possibility of the FVMs themselves containing vulnerabilities. A mobility algorithm embedded in every FVM specifies how it chooses which customer VM to scan. Although multiple scanning strategies have been introduced, there is no work which provides a comparison of these strategies. In this paper, we develop a probabilistic approach which tells us which strategy is best for a given cloud environment and particular family of malware. Our framework uses Bayesian probability in addition to a malware knowledge base in order to simulate the scanning process of a number of FVMs.**

***Keywords*-Forensic Virtual Machine; Mobility Algorithms; Malware; Behavioural Analysis**

I. Introduction

The massive growth of the cloud usage have led to an increase in the related security concerns. Subsequently, multiple techniques have been proposed to minimise the risk and to detect the abnormal behaviour that might be carried out by malicious activities. *Virtual Machine Introspection* (VMI) was one of the proposed techniques that help to increase the security level of the cloud by allowing monitoring the state of Virtual Machines (VMs) in real time through checking their memory pages and other behaviours such as registries and disk changes [1]. It is a powerful technique that can be used by giving some VMs a special privilege in order to inspect and analyse the target VM from the outside. This makes it hard for the attacker to know that these VMs are being investigated. An improvement to this technique has also been introduced by [2], where they proposed a more robust and secure cloud environment through offering an encrypted Virtual Machine Introspection system referred to as *CryptVMI*. Furthermore, in order to offer this mechanism to cloud users on public cloud platform and to get the desired benefit of it easily, Baek et al. in [3] have made it available as-a-service through the virtualisation of the VMI interface.

Based on the VMI technique, Harrison et al. and Shaw et al., in [4] and [5] respectively, proposed and developed *Forensic virtual Machines* (FVMs) which is an architecture for using VMI to detect symptoms of malicious behaviour. FVMs benefit from a mini virtual machines that are used to detect any abnormal behaviour carried out by the inspected VMs. In fact, making FVMs small with fewer functionality reduces the attack surface. Each FVMs are dedicated to identify the existence of a specific symptom that can be resulted from a malicious activity. When an infected VM identified, a mitigation plan will then be applied by the cloud Command and Control centre (C&C).

However, due to the high cost and the fact that lots of resources need to be used when deploying FVMs [5], a further development have been introduced which allow FVMs to check multiple VMs at the same time using preprogrammed strategies which have been referred to as *Mobility Algorithms*. The purpose of these Algorithms is to arrange the movement of FVMs from one VM to another which help in using a single FVM for checking multiple VMs rather than creating a dedicated FVM per VM. Accordingly, this work address some unanswered questions regarding developing a technique that help to choose the most effective Mobility Algorithm to be used by FVMs.

In this paper, we developed a probabilistic approach to analyse Mobility Algorithms. The framework uses Bayesian probability in addition to our implemented malware knowledge base in order to simulate the scanning process of a number of FVMs. Based on the resources that have been used during the scan, the total cost is calculated in order to identify the suitable Mobility Algorithms. Our framework showed that there is a considerable difference on the scanning performance when using different scanning strategies and it was able to show the optimum strategy given parameters that represent the scanned environment such as the number of VMs, the deployed FVMs and the cost of false positive and false negative. Our framework shows that the optimum strategy, in most of the tested cases, is a strategy which has been produced previously by [4] and it has been described later on this paper in Section V-B. In summary, this paper makes the following contributions:

1) We developed a probabilistic approach that helps to compare the cost and the performance of the FVMs

scanning strategies and identifies the optimum strategy for each given environment to be used in the scan.
2) The framework has been tested and evaluated using real malware data.
3) Based on our data, the framework showed that the optimum strategy for most of the tested Malware families and types is a strategy which has been produced previously by[4].

The rest of this paper is organised as follows: Section II describes preliminary information about Cloud Computing, Forensic Virtual Machines and mobility algorithms. Section III introduces the problem under discussion, while Section IV-A presents the sample preparation and data collection procedure. Then the main probabilistic approach is described in section IV-B. The results of the experiment along with the evaluation are then proposed in section V. The paper is then concluded with Section VI.

II. Preliminaries

Cloud infrastructure relies on virtualisation technology, which allows division of the resources between multiple instances of virtual machines (VMs) that, in turn, resulted in the efficient use of the existing computing resources [6]. VMs were defined by [7] as *"A hardware-software duplicate of a real existing computer system in which a statistically dominant subset of the virtual processors instructions execute on the host processor in a native mode"*.

In the following sections we shall briefly review the idea of Virtual Machine Introspection[1] and its latest improvement and application with regard to cloud security. We shall also describe the notion of Forensic Virtual Machines [4] and how they have been used as a replacement of the signature-based detection systems in the cloud. In Section II-A we will explain the outline of the Introspection approach while in Section II-B we describe the recent developed FVMs and several related notions such as Mobility Algorithms.

A. Virtual Machine Introspection

Virtual Machine Introspection (VMI) is a new technology that enables one VM to scan, monitor and modify the current state, such as memory pages and registries, of another VM - when having the sufficient privilege - while remaining hidden. VMI technology was first introduced by [1]. They suggested that instead of having the (IDS) within the customer's VM, the IDS can be pulled outside the host, which will give it a good view over the addressed VM, in addition to let it remain invulnerable to attacks. The proposed IDS (so-called Livewire) has been built for a customised version of VMware Workstation for Linux x86.

Kourai and Chiba in [8] and Keshavarzi in [9] have also introduced an IDS that targets distributed computer systems. The implemented IDS in both benefits from the VMI technology, where it isolates the server that it monitors from the IDS. This allows to provide a complete isolation between the IDS and the attacker malicious code.In addition to its wide usage in developing cloud's IDS, VMI also have been used in improving other services, such as *digital forensics* [10] [11]. There are also further research which have been carried out in order to provide introspection-as-a service in the cloud through introducing written libraries and tools[12] [13] [3].

B. Forensic Virtual Machines

Harrison et al. in [4] made some improvements to the previous works in the area of introspection. They introduced a technique that makes use of mini virtual machines, known as Forensic Virtual Machines (FVMs), which can be used to inspect the memory pages of other VMs to identify the existence of specific symptoms that arise due to malicious behaviour. FVMs try to identify these symptoms instead of looking at the behaviour itself, which works in a similar way to diagnosing illness the human body. Fig. 1 gives an overview of the implemented approach where it shows a number of mini VMs (FVMs) which have been given the required privilege that let them inspect the other VMs.

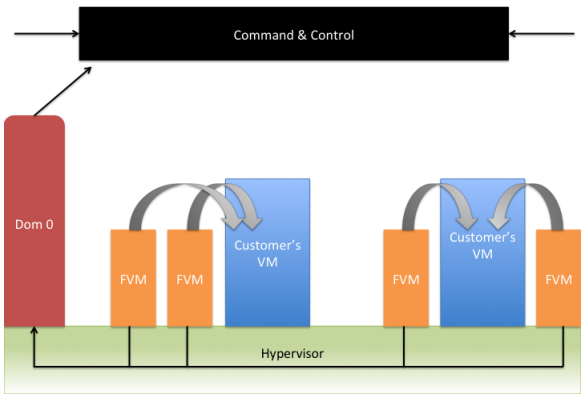

Figure 1: FVMs inspecting customer's VMs.

FVMs can inspect multiple VMs at the same time, which reduces the cost of FVMs created, on one hand, and increases the security, on the other hand [4][5]. This is because the number of FVMs inspecting an exact VM becomes unknown for the attacker. In addition, each of the FVMs are dedicated to identify the existence of an only one symptom, and when a symptom is discovered, Domain 0 (Dom 0) will report it to the C&C centre, which will, in turn, identify the suitable mitigation based on this fact and in relation to other information as well. Shaw et al, have developed these FVMs further in order to reduce the size of these VMs to make verifying the integrity of these special VMs less time consuming [5].

1) Mobility algorithms: After each FVM is created, it is initialised with a special scanning strategy or so- called *Mobility Algorithm* that is responsible for scheduling the

movement of the FVM from one VM to another. Harrison et al. and Shaw et al., in [4] and [5] respectively, proposed three different Mobility Algorithms which vary from simple to dynamic ones such as random movement, scanning according to a determined order, in addition to introducing an advanced scanning strategy or algorithms.

III. Description of the problem

FVM is an innovation technology that can be used to replace the signature-based detection systems on the cloud. It helps to reduce the usage of the resources and the memory consumption, which in turn will boost the scanning performance on the cloud. During the scan, each FVM will be initialised with a chosen Mobility Algorithm, which defines the FVM movement from one VM to another during the scan.

Currently, there are multiple scanning strategies that can be used during the initialisation of FVMs. For example, FVMs can be prompted to scan VMs randomly, according to a specific order or based on a dynamic Mobility Algorithm. During the scan, each scanning strategy may allocate a different number of resources, therefore, the scanning cost can be increased due to various factors such as:

- The type of malware infection or family.
- The state of the scanned environment, such as the number of VMs scanned and the FVM initialised.

Therefore, a further development have been introduced by [5] that allow FVMs to check multiple VMs at the same time using preprogrammed strategies or the Mobility Algorithms. This resulted in an urgent need for an approach that helps in understanding and analysing the current state of the scanned environment in order to nominate a specific scanning strategy based on each scanned environment.

In the following sections we shall present a solution that help to identify and nominate the optimum scanning strategy based on the current state of a given environment.

IV. Sketch of the solution

In this paper, as a solution for the described problem, we present a probabilistic approach that helps in identifying the optimum Mobility Algorithm given parameters that represent the scanned environment such as the number of VMs, the deployed FVMs and the cost of false positive and false negative. The framework can be broken down to three steps as shown in Fig. 2. Section IV-A describes the procedure which has been followed in order to collect the required malware data, whereas section IV-B covers the remaining two steps of the solution.
By using this approach, the scanning strategy with the lowest scanning cost can be identified which, in turn, helps in boosting the scanning time and procedure.

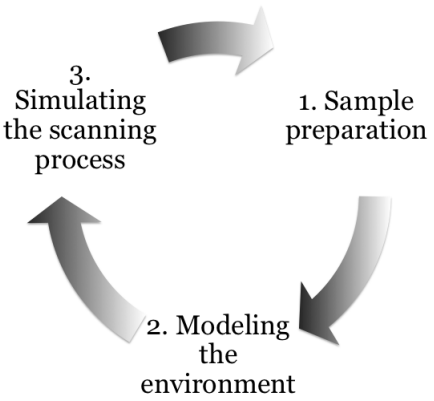

Figure 2: Our proposed solution.

A. Sample preparation and building the knowledge base

In order to achieve the objectives of this research, it was essential to develop a knowledge base, which consists of a large number of malware and benign symptoms. This helps in providing a broad picture of the possible and expected symptoms in terms of both malicious and normal activities. The following section provides an overview of the procedure that have been followed to collect the required malware data. Initially, a python-based tool (maltrieve)[14] was used to parse a list of malware websites and download the latest malicious code uploaded to them. This list includes: Malware Domain List, VX Vault, Malc0de, Malware Black List and Sacour.cn. However, the problem was that downloading malware using such tool only can result in a considerable bias in the database. Therefore, we have combined this method with collecting malware based on their families in order to minimise such a disadvantage. We have used all of the top 10 and 20 malware families recorded on Internet Security Threat reports by Symantec[15] and Microsoft Security Intelligence Reports [16](2006-2013), respectively. For this purpose, a python script was written to pull all samples resulting from the search request for each family, including all known variants, from an open malware source ”Open Malware”[17]. It has been noted that the bias in the database was minimised and more samples from the most common malware families were downloaded, and as a result of the previous step, 14746 malware samples have been collected.

Using the above methods, we have implemented a downloading framework, which is shown in Fig. 3, to help downloading malware automatically from the mentioned different sources, in addition to sending them to Anubis sandbox, an online sandbox [18]. After the analysis is completed, behavioural analysis reports can be automatically retrieved, and parsed so that the recorded features can be added to our knowledge base. Apart from the information parsed, the knowledge base also includes some valuable data such as the approximate discovered date of each malware family and the

risk level, as well as the distribution level of the malware family which has been set by Symantec and Avira websites [19][20].

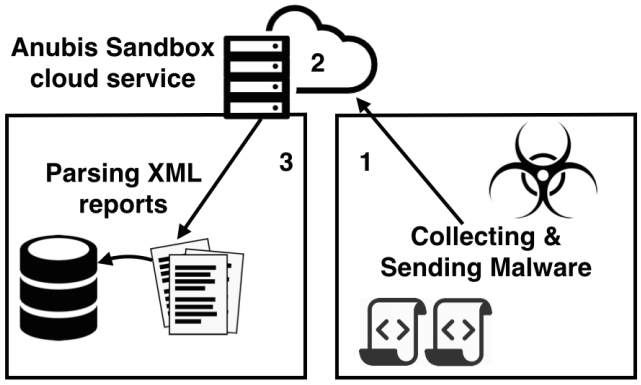

FIGURE 3: OUR DOWNLOADING FRAMEWORK.

B. Identifying the optimum Mobility algorithm to be used in the scan

The main problem that this research addresses is identifying the optimum Mobility Algorithm that can be adopted in the scanning process. We have benefitted from our knowledge base in implementing a framework that simulates the process of FVM scanning. Fig. 4 shows an overview of the implemented simulator.

The simulator is obtained based on probability theory and Bayes theorem; it mainly benefits from information recorded or derived from our knowledge base, such as the probability of seeing an exact symptom given that there is a malware infection. Such information is used to develop a sample environment or a case study with a number of infected or non-infected VMs along with a number of symptoms, that can be seen on them on either case. The scan is then carried out by by multiple FVMs which are initialised with a specific Mobility Algorithm after setting multiple parameters, such as the number of scanned VMs, the cost of false positive FP, the false negative FN and the initial belief of a malware infection $p(Malware)$. This implemented environment with the initialised VMs and FVMs is used then as a case study to simulate the scanning process. Thus, during the scanning process, the framework will be able to update its initial belief of an infection according to the new evidence gathered using Bayes Theorem role. The evidence in this case is like confirming or denying the existence of a symptom on a specific VM. Updating the initial belief after each step of the scan will help the simulator in determining the infected and the non-infected machines. Therefore, in order to nominate the optimum and most beneficial strategy, it is necessary to determine the cost of the scan after each step, in addition to calculating the overall cost of the scan. We shall describe the framework mathematically as follows:

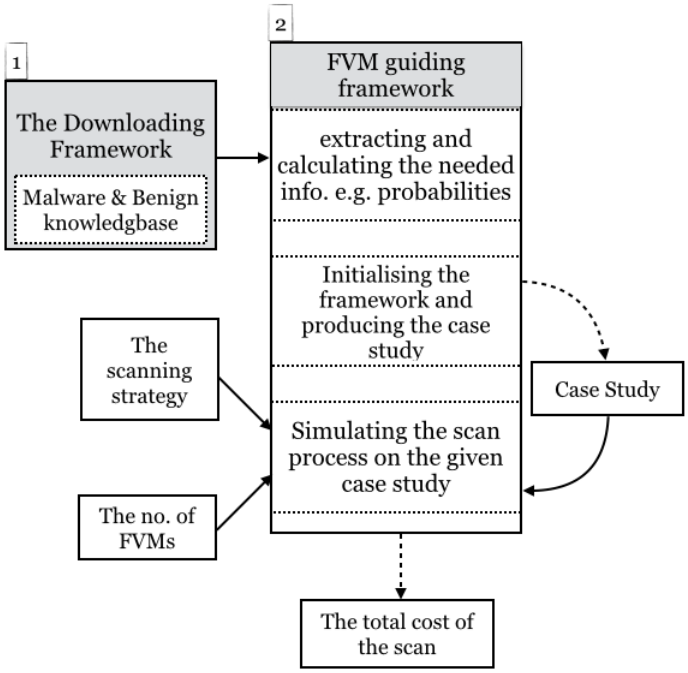

FIGURE 4: FVM GUIDING FRAMEWORK

Definition 1: For set of boolean symptoms: $S_1, \ldots S_n$ (which may be True or False) Set of all symptom sub-assignments is defined as:

$$AS = \{\{S_i = b \mid S_i \in \mathcal{S}, b \in \{True, False\}\} \mid \mathcal{S} \in \mathcal{P}(S)\} \quad (1)$$

an example of set of symptoms can be: $\{S_2 = True, S_5 = False, S_8 = True\}$ which is a member of AS.

From our obtained knowledge base we would like also to derive the following information, for all $S \in AS$:

- $p(S \mid M)$: is the probability of seeing S when there is malware.
- $p(S \mid B)$: is the probability of seeing S when malware is not present

By applying Bayes law, we can then calculate the degree of belief in having a malware infection $p(M|S)$ taking into account the likelihood of the evidence occurring in addition to the initial belief $p(M)$:

$$p(M|S) = p(S \mid M).p(M)p(S) \quad (2)$$

where $p(S)$, the probability of seeing a group of symptoms S, can be calculated as follows:

$$p(S) = p(S \mid M).p(M) + p(S \mid B).(1 - p(M)) \quad (3)$$

As a result, depending on the value of $p(M|S)$ we can claim that there is a malware if $p(M|S) \geq$ *a pre agreed threshold* , as an example, if we assume that the VMs we are scanning are infected with

a probability of 0.4, we can then describe the VMs status as follows:

$$\texttt{p(M|S)} \begin{cases} \geq 0.4, & \text{there is a malware} \\ < 0.4, & \text{there is no malware} \end{cases} \tag{4}$$

Subsequently, we can say that false positive and false negative can be calculated based on $p(M|S)$ and it would be possible to calculate the cost of the overall scan using the proposed framework, therefore, a conclusion can be drawn based on the results such as follows:

Definition 2: Let the cost of false positive represented as $\mathcal{FPC}$, and the cost of false negative as $\mathcal{FNC}$ and the cost of reading S as $\mathcal{SC}$, the cost of scanning for a group of symptoms S in a VM is:

$$\texttt{Cost(S)} = \begin{cases} \mathcal{SC} + (1 - p(M|S)).\mathcal{FPC}, & p(M|S) \geq 0.4 \\ \mathcal{SC} + p(M|S).\mathcal{FNC}, & p(M|S) < 0.4 \end{cases} \tag{5}$$

where

- $p(M|S)$, refers to the degree of belief in having an infection on the scanned machines.
- FNC, refers to the cost of incorrectly identifying a machine as non-infected.
- FPC, refers to the cost of incorrectly identifying a machine as infected.

V. Experiment results and validation of the outcomes

A. Modelling the environment

We shall describe the following assumptions that have been made during the experiment before presenting the experiment procedures and results:

1) Because we assume that we are dealing with only a snapshot of a system, we are not considering the case when a system state changes from infected to non-infected or vice versa during the scan.
2) We assume that when an FVM scans for a specific symptom, it will eventually identify it.
3) We are assuming discrete time steps where the scan of each symptom take the same amount of time.

In addition to the above assumptions, it is important to state that the framework results depend on the malware data examined during the experiment. We have also examined the framework using registry key changes as symptoms, and for each family, we have used the most common changes in the registry keys recorded mostly by Symantec [19] and Microsoft [21].

B. Description of the Experiment and Analysing the Results

The four Mobility Algorithms that have been tested can be described briefly as follows:

1) **Strategy 1:** is a random strategy where the next scanning target is chosen completely randomly at the beginning of each step of the scan.
2) **Strategy 2 and 3:** in both, each FVM has different preprogrammed priorities, and it chooses the next scanning targets based on these priorities. The priorities in both strategies have been set based on the probability of seeing a malware given a group of symptoms S which have been calculated based on our knowledge base. Therefore, when an FVM that scans for symptom s_p is informed that s_m and s_n have been found on two different machines VM_i and VM_j respectively, the FVM has to check their predefined priorities to determine which machine has the symptom that - together with the symptom that it scans for - has the highest probability between the two, $p(Malware \mid s_p \cap\ s_m)$ or $p(Malware \mid s_p \cap\ s_n)$. It will then set this machine as the next scanning target. The only difference between these two strategies is that the former will make the decision based on the probability of having a malware given only two observed symptoms, whereas the latter will be considering three symptoms all together.

3) **Strategy 4:** it has been actually proposed previously by [4], in which they grouped the malicious behaviour into configurations, where $C = c1, c2, ..., cm$ denote the set of all the important configurations. Each of these configurations consists of a subset of symptoms where it can be represented as follows: $S = s1, s2, ..., sn$. Their strategy can be mathematically written as follows:

$$F(v) = \sum_{i=1}^{K} \frac{Disc(c_i, v)}{size(c_i)} val(c_i) + \lambda\ [CurrentTime - LastVisited(v)] \tag{6}$$

where

- $\frac{Disc(c_i,v)}{size(c_i)}$ is the ratio of the number of discovered symptoms that belongs to the configuration c_i, to the total number of symptoms in the same configuration.
- $val(c_i)$ is the severity value of configuration c_i assigned by a security expert.
- $[CurrentTime - LastVisited(v)]$ is the time passed since a VM has been scanned or visited by any FVM.
- λ is the *Loneliness Parameter* where $\lambda > 0$, and the smaller the number we set this parameter to, the less important the time of the last visiting will be.

We have tested the mentioned FVM's strategies on four malware families and three malware types. These families and types have been chosen randomly from our database and the type of each malware family has been identified based on the Symantec online database [19]. Table I shows the tested

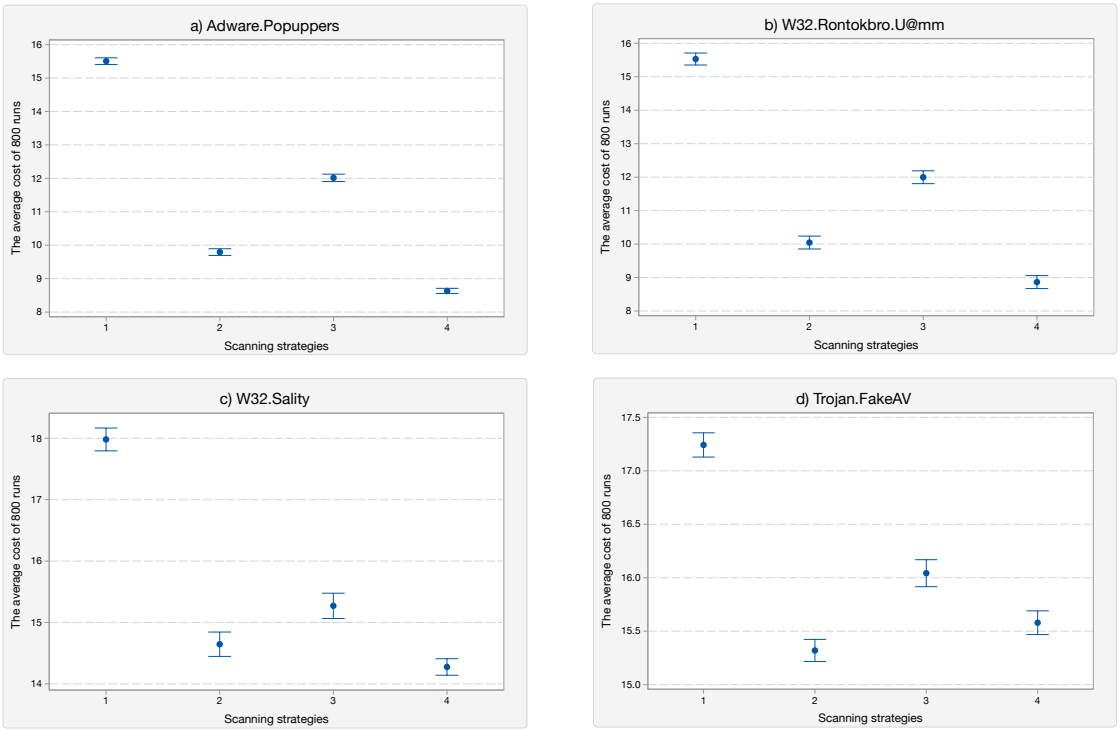

FIGURE 5: THE 95% CI OF THE MEAN COST FOR EACH MALWARE FAMILY

malware families along with the number of malware samples that have been used during the experiment, in addition to the type of malware that each family belongs to.

During the test, we have recorded the results of 20 trial where In each trial we simulate the run of 800 scans and record the average cost of these scans in order to get a more robust results. We have calculated the $Cost$ based on the previously mentioned assumptions as follows:

$$\texttt{Cost(S)} = \begin{cases} \mathcal{SC} + (1 - p(M|S)).\mathcal{FPC}, & p(M|S) \geq 0.4 \\ \mathcal{SC} + p(M|S).\mathcal{FNC}, & p(M|S) < 0.4 \end{cases} \tag{7}$$

Given the following estimated values:
$FNC = 15$ and $FPC = 5$

In each test, FVMs were initialised to scan for a different malware family by identifying their distinctive registry key changes. Then, based on the calculated cost, the simulator nominates the most efficient Mobility Algorithm that reduces the cost in each case. The strategies performance is represented on Fig. 5 and Fig. 6 based on a 95% confidence interval calculation of the mean cost of each strategy. It can be seen from Fig. 5 that none of the strategies' confidence interval, when tested with each family, have shown any overlap. This shows that there is a significant difference in the performance between the tested strategies. Based on Fig. 5, it can be concluded that the optimum strategy for Adware.Popuppers, Rontokbro and Sality is the fourth strategy, where it gives the lowest scanning cost, whereas strategy 2 shows the lowest scanning cost when used to scan for Trojan.FakeAV. Fig. 5 also shows that strategy 1 - the random strategy - gives always the highest cost given the tested malware families.

TABLE I: MALWARE FAMILIES USED ON THE EXPERIMENT

Malware Family	no. of Samples	Malware Type
Adware.Popuppers	167	Adware
W32.Rontokbro@mm	210	Worm
W32.Sality	109	Virus
Trojan.FakeAV	102	Trojan

The strategies also have been tested not only on malware families, but also on their types in order to see if the optimum strategy can vary from family to family within the same malware type. Table II shows the tested malware types, the tested families in each type along with the number of malware samples that have been used during the experiment. The distinctive symptoms that have been used during the FVMs initialisation on family testing have also been used during the initialisation of FVMs when testing the related malware type. This is based on the assumption that all malware families within the same type share the same symptoms. The results are shown on Fig. 6 and it can be seen that the optimum Mobility Algorithm for all tested malware types tends to be the fourth strategy with the highest cost

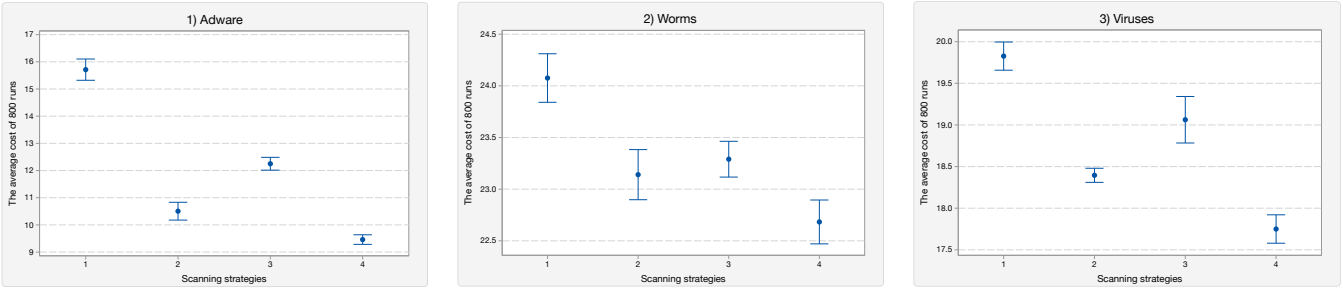

FIGURE 6: THE 95% CI OF THE MEAN COST FOR EACH MALWARE TYPE

given when the scan performed randomly (strategy 1). It can be seen also in the case of Adware that the dependency between the symptoms is relatively strong, which induce the gap between Strategy 1 results and the rest of the strategies , as strategy 1 chooses the next scanning target randomly.

TABLE II: MALWARE TYPES USED ON THE EXPERIMENT

Malware type	families tested within this type	no. of Samples
Adware	Adware.Popuppers	167
	Adware.Clkpotato!gen3	32
	Adware.Istbar	26
	Adware.Slagent	26
	Adware.ZenoSearch	14
Worms	W32.Rontokbro@mm	210
	W32.Spybot.Worm	196
	W32.Benjamin.Worm	88
	W32.Mabezat.B!inf	35
	W32.SillyFDC	25
Viruses	W32.Sality	109
	W32.Virut	97
	W32.Whybo!inf	88
	W32.Licum	27
	W32.Xpaj.C	21

VI. CONCLUSION

Before initialising and using Forensic Virtual Machine Based Malware Detection system, it is important to be able to recognise the most beneficial FVM's Mobility Algorithms which gives the optimum scanning results with the minimum spending cost. This article presented a probabilistic approach in order to improve FVMs scanning performance. We have implemented also a framework that collects malware automatically. We have shown in this paper that some scanning strategies can consume less resources with some malware families and checking which strategy is the optimum before initialising FVMs can improve the scanning procedure.

Now, that we have successfully demonstrated that the proposed technique work, our future goal is to address some limitations on the existing work. For example, we have not consider the case when a system state change from infected to non-infected or vice versa during the scan. We have also, for the simplification purpose, consider only the case when all the VMs on the system current snapshot are infected with the same family or type of malware. However, considering the size of the cloud, a VM can be subject to multiple attacks, therefore, having multiple malware families or type on the same time should be taken into account. Thus, our future goals are to add new extensions that address these current limitations.

REFERENCES

[1] T. Garfinkel, M. Rosenblum *et al.*, "A virtual machine introspection based architecture for intrusion detection." in *NDSS*, 2003.

[2] F. Yao and R. Campbell, "Cryptvmi: Encrypted virtual machine introspection in the cloud," in *Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on*, June 2014, pp. 977–978.

[3] H. W. Baek, A. Srivastava, and J. van der Merwe, "Cloudvmi: Virtual machine introspection as a cloud service," in *Cloud Engineering (IC2E), 2014 IEEE International Conference on*, March 2014, pp. 153–158.

[4] K. Harrison, B. Bordbar, S. T. Ali, C. I. Dalton, and A. Norman, "A framework for detecting malware in cloud by identifying symptoms," in *Enterprise Distributed Object Computing Conference (EDOC), 2012 IEEE 16th International*. IEEE, 2012, pp. 164–172.

[5] A. L. Shaw, B. Bordbar, J. Saxon, K. Harrison, C. Dalton *et al.*, "Forensic virtual machines: dynamic defence in the cloud via introspection," in *Cloud Engineering (IC2E), 2014 IEEE International Conference on*. IEEE, 2014, pp. 303–310.

[6] D. E. Williams, *Virtualization with Xen (tm): Including XenEnterprise, XenServer, and XenExpress: Including XenEnterprise, XenServer, and XenExpress*. Syngress, 2007.

[7] R. P. Goldberg, "Architectural principles for virtual computer systems," DTIC Document, Tech. Rep., 1973.

[8] K. Kourai and S. Chiba, "Hyperspector: virtual distributed monitoring environments for secure intrusion detection," in *Proceedings of the 1st ACM/USENIX international conference on Virtual execution environments*. ACM, 2005, pp. 197–207.

[9] M. Keshavarzi, "Traditional host based intrusion detection systems' challenges in cloud computing," *Advances in Computer Science: an International Journal*, vol. 3, no. 2, pp. 133–138, 2014.

[10] C. Benninger, S. Neville, Y. Yazir, C. Matthews, and Y. Coady, "Maitland: Lighter-weight vm introspection to support cyber-security in the cloud," in *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, June 2012, pp. 471–478.

[11] "V. systems. volatility," https://www.volatilesystems.com, 2015.

[12] F. Westphal, S. Axelsson, C. Neuhaus, and A. Polze, "Vmi-pl: A monitoring language for virtual platforms using virtual machine introspection," *Digital Investigation*, vol. 11, pp. S85–S94, 2014.

[13] B. D. Payne, "Libvmi - virtual machine introspection," http://libvmi.com, 2015.

[14] K. Maxwell, "Maltrieve," https://github.com/technoskald/maltrieve, 2015.

[15] Symantec, "Internet security threat report," http://www.symantec.com/security_response/publications/threatreport.jsp, 2015.

[16] Microsoft, "Microsoft security intelligence report," http://www.microsoft.com/security/sir/default.aspx, 2015.

[17] Open Malware, http://www.offensivecomputing.net, 2015.

[18] International Secure Systems Lab, "Anubis - malware analysis for unknown binaries," https://anubis.iseclab.org, 2014.

[19] Symantec, "Symantec security response," http://www.symantec.com/security_response/landing/azlisting.jsp, 2015.

[20] Avira Virus Lab, http://www.avira.com/en/support-virus-lab, 2015.

[21] Microsoft, "Malware encyclopedia," http://www.microsoft.com/security/portal/threat/Threats.aspx, 2015.