Adaptive Evidence Collection in the Cloud Using Attack Scenarios

Liliana Pasquale[a], Sorren Hanvey[a], Mark Mcgloin[b], Bashar Nuseibeh[a,c]

[a]*Lero - the Irish Software Research Centre, University of Limerick, Ireland*
[b]*IBM Software Labs, Dublin, Ireland*
[c]*The Open University, Milton Keynes, UK*

Abstract

The increase in crimes targeting the cloud is increasing the amount of data that must be analysed during a digital forensic investigation, exacerbating the problem of processing such data in a timely manner. Since collecting all possible evidence proactively could be cumbersome to analyse, evidence collection should mainly focus on gathering the data necessary to investigate potential security breaches that can exploit vulnerabilities present in a particular cloud configuration. Cloud elasticity can also change the attack surface available to an adversary and, consequently, the way potential security breaches can arise. Therefore, evidence collection should be adapted depending on changes in the cloud configuration, such as those determined by allocation/deallocation of virtual machines. In this paper, we propose to use attack scenarios to configure more effective evidence collection for cloud services. In particular, evidence collection activities are targeted to detect potential attack scenarios that can violate existing security policies. These activities also adapt when new/different attacks scenarios can take place due to changes in the cloud configuration. We illustrate our approach by using examples of insider and outsider attacks. Our results demonstrate that using attack scenarios allows us to target evidence collection activities towards those security breaches that are likely, while saving space and time necessary to store and process such data.

Keywords: forensic readiness, cloud computing, adaptive software, attack planning

1. Introduction

Although recent years have seen substantial market growth in the usage of cloud services, such services are increasingly the target of cyber-crimes [1]. For example, a recent attack targeting JP Morgan Chase [2] likely exploited the lack of two factor authentication for gaining access to user information from over 76 millions house holds. To assess how a cyber-crime was perpetrated, what harm was done, and who are

Email addresses: liliana.pasquale@lero.ie (Liliana Pasquale), sorren.hanvey@lero.ie (Sorren Hanvey), mark.mcgloin@ie.ibm.com (Mark Mcgloin), bashar.nuseibeh@lero.ie (Bashar Nuseibeh)

the parties responsible, existing systems must be forensic ready [3] and, in particular, support targeted data collection. This would allow preserving relevant data that might provide evidence necessary for prosecution. Such data can reside at both the customer and the provider premises, and, therefore, the partitioning of forensic responsibilities between the cloud service providers and the customers may depend on the cloud service model used.

The increase in crimes targeting the cloud has also increased the amount of data that must be analysed during a digital forensic investigation, exacerbating the problem of processing such data in a timely manner. Moreover, while cloud computing provides an "elastic" environment for storage and computing resources to be provided and released on demand, the evidence necessary to investigate a cyber-crime can be volatile and may no longer be available after a security breach is perpetrated. Collecting all possible evidence proactively is not always a viable solution, since it can be too voluminous and cumbersome to analyse effectively. Instead, we suggest evidence collection activities should focus on gathering the data necessary to investigate potential security breaches that can exploit vulnerabilities present in a particular cloud configuration.

Cloud elasticity can also change the attack surface available to an adversary and, consequently, the way potential security breaches can arise. Therefore, cloud-related changes, such as those determined by allocation/deallocation of virtual machines (VMs) or the modification of physical/virtual machines configurations, should be detected and monitored, and the corresponding data collection activities should be adapted accordingly.

In this paper, we address the above concerns by proposing the use of attack scenarios [4] to engineer more effective evidence collection for cloud services. Attack scenarios represent a sequence of actions an adversary can perform to achieve her criminal goals (e.g., compromising a physical or virtual machine, or copying a VM image). Automated generation of attack scenarios [5, 4, 6, 7, 8] has been utilised extensively to identify the security breaches that are likely and to focus the security strategy of an organisation on their prevention. However, this might not always be possible; for example, as a security patch might not yet be available or a cloud provider may have to rely on the customer to apply such patches. Our approach, instead, builds on the intuition that attack scenarios can help reduce the amount of data to be preserved in the cloud, by focusing data collection activities on the security breaches that are likely.

Our approach helps configure evidence collection activities for preserving the data necessary to explain how potential attacks are perpetrated. We model explicitly the cloud configuration, the security breaches and the basic actions that an adversary can perform to achieve her criminal goals (i.e. cause a security breach). These basic actions can be legitimate, such as access to a VM, or malicious, i.e. attack modules, that an adversary can execute to exploit existing vulnerabilities. We assume attack modules are defined systematically by a security administrator from vulnerability databases (e.g., CVE[1]) documenting existing vulnerabilities and how they can be exploited. Basing them on the representation of the cloud configuration, the security breaches, and the basic actions of an adversary, we use planning techniques to identify attack scenarios

[1] https://cve.mitre.org/index.html

that achieve the security breaches by exploiting the current cloud configuration. Our technique cannot identify evidence related to attacks exploiting vulnerabilities that are unknown to the software vendor. Nevertheless, identifying attack scenarios exploiting known vulnerabilities is a challenging problem, as the complexity of a cloud computing environment makes it difficult for a security administrator to foresee all possible attack scenarios exploiting vulnerabilities introduced by the software components installed.

We then map each basic action composing the attack scenarios to the data to be collected at the cloud service provider premises for demonstrating their execution. Finally, we adapt the evidence collection activities in reaction to changes of the cloud environment or updates of available attacks modules, which may be the sources of new/different attack scenarios. Such changes are monitored constantly and reflected onto the model representing the cloud configuration and the basic actions that can be perpetrated by an adversary. Note that we proposed the notion of adaptive digital forensics in previous work [9, 10] for adapting the evidence collection and analysis activities of a digital forensic investigation depending on the likelihood of possible hypotheses of a crime. However, unlike our previous work, in this paper we propose to adapt evidence collection activities depending on changes in the cloud configuration and in existing vulnerabilities.

We illustrate our approach by using examples of cyber-crimes targeting cloud customers and providers. We do not consider additional cases [11] such as the use by an adversary of cloud resources to perpetrate a crime (e.g., to build botnets, to store and share illicit material). Our simplified examples are set in an Infrastructure as a Service (IaaS) cloud deployment including one service provider and several customers all sharing that infrastructure and resources. For this reason, in this paper we assume that the collection process does not breach the regulations of the jurisdiction in which the data are collected, and does not compromise the confidentiality of other tenants that share the resources. We evaluated our approach by estimating the percentage of data that it avoids collecting, and measuring the time required to generate attack scenarios for cloud configurations of increasing complexity. Our results demonstrate that using attack scenarios allows us to target evidence collection activities only in those situations in which security breaches are likely, while saving space and time necessary to store and process such data. We believe this is a reasonable simplification for preserving relevant evidence, and issues such as regulatory compliance can be addressed separately. Furthermore, attack scenarios can be generated in a negligible time even for cloud configurations representing realistic data centers.

The rest of the paper is organised as follows. Section 2 motivates our approach by discussing related work. Section 3 describes scenarios supporting the need for adaptive evidence collection in the cloud. Section 4 provides an overview of our approach. Section 5 explains how attack scenarios are generated for cases of insider and outsider attacks. Section 6 illustrates how evidence collection and monitoring of changes in the cloud environment are performed. Section 7 presents our evaluation and Section 8 concludes.

2. Related Work

Recently, researchers have analysed the digital forensic challenges brought by cloud computing [12, 13, 14, 15]. Wolthusen [12] notes that one of the major challenges is the collection of evidence across multiple virtual hosts, physical machines, data centers, and geographical and legal jurisdictions. Taylor et al. [13] emphasise the ephemerality of evidence stored in the cloud, such as registry entries (on Microsoft Windows platforms) or temporary internet files that can be lost when a customer leaves the service. Ruan et al. [14] suggest the need of approaches to continuously preserve volatile data and guarantee data segregation among multiple tenants and in various cloud service models. Grispos et al. [15] highlight the challenges that an investigator may face while analysing an extremely large amount of data placed in the cloud by a customer. Cloud environments have also been suggested as a basis for conducting digital forensic investigations [16]. In particular, cloud virtual instances and storage can be used, respectively, to gather and store evidence relative to potential or detected incidents/crimes.

Existing research has started exploring technical solutions to perform evidence collection in the cloud. For example, Birk and Wegener [17] assess the usability of various sources of evidence for investigative purposes in all three major cloud service models (SaaS, PaaS, IaaS). Dykstra and Sherman [18] expose technical and trust issues that arise in acquiring forensic evidence from IaaS cloud deployments by using existing forensic acquisition tools (EnCase [19] and AccessData Forensic Toolkit [20]). The authors also suggest that evidence should be collected at the virtual machine level, where a web system interfaces with the provider's underlying filesystem and hypervisor. The advantages lie in the fact that evidence can be collected "on-demand" by several parties, including customers, providers and lawyers. A tool providing such functionalities was subsequently developed by Dykstra et al. [21] to support forensic acquisition of virtual disks associated with VMs, logs of all APIs requests made to the cloud provider for administering virtual machines, and OpenStack firewall logs for any of the customers' virtual machines. Shields et al. [22] created a proof-of-concept continuous forensic evidence collection system that could be used in the cloud, for example, to record the deletion and creation of service provisions. However, in large scale environments like the cloud, monitoring all possible evidence is not a viable solution, as it might be cumbersome to analyse. Existing work has mainly focused on how evidence can be collected in the cloud without providing guidance on what data should be preserved. Although triaging techniques have also been proposed as a means of reducing the amount of data to be analysed in conventional investigations [23], they have not been applied to target evidence collection activities towards the preservation of the data necessary to investigate the security breaches that are more likely. Furthermore, none of the existing approaches have focused on how to adapt evidence collection depending on the potential attack scenarios brought about by the current cloud configuration.

Our approach automatically generates attack scenarios through planning in order to identify the security breaches that are likely and the evidence that should be collected to investigate them. Attack planning [5, 4] has mainly been used to anticipate zero-day attacks against the networked computing infrastructures of an organisation and to improve penetration testing tools. Krautsevich et al. [6] propose to generate attack scenarios depending on the knowledge and resources possessed by an adversary. The

authors assume an attacker can recompute her strategy dynamically in case an attack step is unsuccessful. This allows companies to focus their security strategies to prevent the most likely attacks. Similarly, Sarraute et al. [7] generate attack scenarios by taking into account lack of knowledge an adversary has about the network topology. LeMay et al. [8] represent explicitly how an adversary is likely to attack the system depending on her preferences, actions cost, payoff and probability of detection. However, automated attack planning research has not addressed the problem of effective evidence collection in the cloud.

In our approach attack scenarios are generated automatically after the model of the cloud configuration, the security breaches and the attack modules representing existing vulnerabilities are defined by the security administrator. Considering the vulnerabilities associated with specific software products installed allows us to identify concrete attack scenarios and hence specific monitoring activities that collect forensic evidence necessary to demonstrate that these attacks took place. Cloud security reference architectures [24] provide guidance for identifying potential misuse cases [25] arising from the interaction of stakeholders with the system and for detecting a set of security patterns, which include countermeasures to prevent or mitigate those misuse cases. Security patterns also include a general list of components of the cloud architecture, such as VMs and networks, from which forensic evidence regarding a specific misuse pattern can be found. Introducing the usage of a reference architecture in our approach could lead to the identification of a more complete set of security breaches, referred to as threats. However, reference architectures are too general and do not suggest ways to implement monitoring activities to collect forensic evidence of specific attacks. Moreover they do not consider adaptation of security patterns, which might be necessary when the cloud configuration changes.

3. Adaptive Evidence Collection Scenarios

In this section we explain how cloud-related changes affecting virtual and physical machines and jurisdictions can require modifying the evidence collection strategy at the cloud service provider.

Virtual machine changes are related to the allocation/deallocation of new or existing VMs or to the modification of their software configuration. In particular, changes a customer performs on the VMs she is authorised to use, such as installation of third party applications, might introduce vulnerabilities that can be exploited by an adversary. The introduction of new vulnerabilities may require a change in the evidence collection strategy to be enacted since new or different attack scenarios might be likely. Privileges over a VM determine whether a customer is authorised to use a specific VM, deploy third party applications within the VM, or make a copy of the present state of the VMs, including all the data contained within or associated with the VM. Privileges that are currently granted and revoked to customers over specific VMs can affect how possible attacks can be perpetrated, and, therefore, may require modification of the evidence collection strategy depending on their current state.

Similarly, *physical machine changes* are related to the acquisition/dismissal of new/existing physical machines or to the modification of their software configuration. These changes can affect how insider attacks [26] and intrusions are performed. For

example, the physical machine on which a VM is hosted can affect potential attacks scenarios. For instance, if a VM is hosted on a physical machine an administrator can login to and make copies of VM images, confidential data could be stolen easily by a malicious administrator. In this case, it will be necessary to collect information related to VM copy operations and logins on the physical machine. However, in another case, if a VM is installed on a physical machine an adversary cannot login to, she must exploit existing vulnerabilities of that physical machine (e.g., perform a buffer overflow attack) to gain the root privileges and copy the image of the target VM. In this case, different evidence should be collected in addition to the evidence specified in the previous case (e.g., networks connections and open ports of the machine, executing processes).

*Jurisdiction change*s are related to the modification of the privacy or security regulations a cloud provider should comply with. Indeed some evidence collection activities might no longer be legal in the new jurisdiction and might be restrained. Cloud provider merges (e.g., cloud provider outsources part of its services to IaaS offered by other providers) are also relevant as they might require to include evidence collection activities to comply with the SLAs negotiated with external providers.

4. Overall Approach

An overview of our approach is shown in Figure 1. A *Planning* activity generates potential attack scenarios by using the information coming from potential *Security Breaches*, the *Cloud Configuration*, and the possible *Attack Modules*. We assume that these models are initially created and periodically updated by the system administrators at the provider premises.

[Figure 1 about here.]

Security breaches represent possible criminal goals of an adversary, which may violate the policies of an organisation or the regulations of a specific jurisdiction in which the violation takes place. For example, an adversary can aim to steal customers' sensitive data stored on a target VM, or compromise a physical/virtual host (e.g., by installing untrusted software that can escalate her privileges). An attack can be perpetrated by individuals assuming different roles. An adversary can be an administrator appointed by the host company, a customer with allocated virtual machines or an external individual who is not authorised to use or access the cloud infrastructure.

The cloud configuration models object types, objects instantiations and their states. Object types can represent, for example, physical and virtual machines, authoritative domains, installed software components, network connections, customers or system administrators at the hosting company. Object instantiations represent concrete objects of a specific type. Initial states can be used to identify, for example, the location of physical and virtual machines and their software configuration, network connections, open ports of physical and virtual hosts, or the authorisations granted to customers and administrators.

The cloud configuration also includes the legitimate actions that can be performed by the customers and system administrators within the cloud deployment. These are expressed in terms of pre- and post-conditions (effects) on the state of some of the

objects represented in the cloud configuration. For example, some actions can be used to allocate/deallocate a VM, perform login/logout, or copy the image of a VM hosted on a physical machine. In this example, the pre-condition necessary to perform a copy of a VM image is that the user is an administrator logged on the physical machine hosting the target VM and s/he is authorised to perform a copy of the VM image or the user is a customer that has been allocated the VM and s/he is authorised to use it.

Attack modules represent malicious actions that can be performed by an adversary to compromise a virtual or physical host. These actions can also be expressed in terms of pre- and post-conditions on the state of the objects belonging to the cloud configuration. Attack modules leverage vulnerabilities that are present in existing hosts. For example, the following action (`untrusted-search-path`) represents a local exploit that uses a vulnerability [2] of VMware Workstation (versions 5.x and 6.x), which allows a local user to gain root privileges by installing a library path option in a configuration file. In this action parameter `pm` is used to identify a physical machine (`pmachine`). The precondition requires either the existence of a user (`u`) administering a physical machine to be logged on to it or the physical machine to be compromised. The physical machine should also run VMWare Workstation (versions 5.x or 6.x). The post-condition represents the fact that a local exploit is installed and has assigned root privileges.

```
(:action untrusted-search-path
 :parameters (?pm - pmachine)
 :pre (and (or (exists (?u - user)(and (admin ?u ?pm)(logged ?u ?pm)))
               (compromised ?pm))
           (has_VM ?pm VMWare_WS)
           (or (has_VM_version ?pm ver5) (has_VM_version ?pm ver6)))
 :effect (installed untrusted_searchPathAgent ?pm high_privilege))
```

We recognise that many of the vulnerabilities exploited by available attack modules can be patched by cloud providers. However, for some exploits a patch might not be available yet or the cloud provider may be at the mercy of the customer when it comes to applying patches (especially for IaaS cloud deployments).

The *Attack Scenarios* generated by the planning activity represent a sequence of legitimate or malicious actions that an attacker can perform to achieve a specific goal. We express the planning problem by using PDDL (Planning Domain Definition Language)[3]. A PDDL planning problem is expressed in terms of a *domain* and a *problem definition*. Different problem definitions may be connected to the same domain description just like several instances may exist of a class in Object Oriented Programming. The domain definition includes a representation of the objects types, constants, predicates expressed on objects and constants, and a set of actions. Each action has a set of parameters (variables that may be instantiated with objects), preconditions and effects. The problem definition includes the definition of all the possible objects, the initial conditions of the planning environment (a conjunction of true/false facts), and the definition of goal states (a logical expression over facts that should be true/false in

[2] `http://www.cvedetails.com/cve/CVE-2008-0967/`

[3] `http://en.wikipedia.org/wiki/Planning_Domain_Definition_Language`

a goal-state of the planning environment). The output of the planner is usually a totally or partially ordered plan (a sequence of actions) necessary to achieve the goal specified in the problem definition from the initial conditions of the planning environment.

In our approach we use the domain definition to model object types, such as physical and virtual machines, and their states (i.e. predicates expressed on these objects). We also use the domain definition to represent the legitimate actions that can be performed by cloud users and administrators as well as malicious actions performed by an adversary (attack modules). The problem definition is used to define concrete object instantiations of a specific type in the cloud environment and their initial state. In our case, the goal state refers to a specific security breach that the attack scenario aims to achieve. The output of the planner returns a sequence of legitimate or malicious actions that an offender can perform to achieve a specific goal. We chose to use PDDL as it is widely supported by several planners and allows ranking potential attack scenarios depending on specific metrics, such as number of actions in the scenario, or estimated harm caused by the attack, which are fundamental features to prioritise evidence collection strategies.

The *Evidence Collection* activity can be configured systematically after possible attack scenarios are generated. The evidence that should be collected is that necessary to verify whether the post-conditions of each (legitimate or malicious) action within the attack scenario are satisfied. This information can be extracted from the events recorded in the log files associated with the software installed in the cloud deployment. We expect that collected evidence will be useful during future investigations to reconstruct the events leading to a security breach and locate the vulnerabilities that were exploited.

The elasticity of a cloud environment requires continuous monitoring of cloud related changes (e.g., location of physical and virtual machines, software configuration of physical and virtual hosts). Such changes must be reflected on the domain and problem definition used during planning. To achieve this aim, we also configure a *Monitoring* activity which identifies potential changes that may affect the state of the objects instantiated in the problem definition, update the problem definition accordingly, and trigger a re-planning of the attacks scenarios. Updates of existing attack modules are also taken into account, as they can modify potential attack scenarios and, consequently, the evidence collection activities. We assume security administrators are notified when vulnerability databases contain new entries for the specific software versions installed in the current cloud deployment. Finally, we assume that modifications of the type of the objects belonging to the cloud configuration (e.g., a new software can be installed) are performed manually by the security administrator and may trigger updates when the new software is bringing known vulnerabilities.

5. Attack Scenarios Generation

In this section, we describe how the cloud configuration, the security breaches, and the attack modules are represented in PDDL, by using a simplified example of an Infrastructure as a Service (IaaS) cloud deployment. Subsequently, we explain how these models are used during planning to generate the attack scenarios. In particular, we consider two sets of scenarios, *Insider attacks*, where the adversary is an authorised

individual at the cloud service provider premises, such as a malicious administrator, and *Outsider attacks*, where the attacks are perpetrated by a malicious customer or an external adversary. We also explain how changes in the cloud configuration trigger modifications in the attack scenarios (and, therefore, in the evidence collection strategy described in Section 6.1).

[Figure 2 about here.]

5.1. Cloud Configuration, Attack Modules, and Security Breaches

The initial *Cloud Configuration* of our example cloud deployment is presented in Figure 2. It comprises three physical machines (`m1`, `m2`, and `m3`) located within two different domains (`a` and `b`). All machines can accept network connections from the others. `m1` and `m2` belong to domain `a` while `m3` belongs to domain `b`. Each physical machine can host VMs; for example, `m1` hosts virtual machines `v1-v5`. Each physical and virtual machine is characterised by a specific software and network configuration. Although we identified the VMs (in grey) hosted on each physical machine, for reasons of simplicity in this initial scenario we only consider the configuration of the physical machines. We subsequently introduce some of the configuration of the virtual machines when describing further attack scenarios.

Each physical machine is described by its operating system, virtualisation software, open ports and offered services. For example, as shown in Figure 2, the operating system (`os`) of `m1` and `m2` is `VMWare WorkStation 5.x`, while the operating system of `m3` is `Win XP SP2`. The virtualisation software (`v_sw`) of `m1`, `m2`, and `m3` is `VMWare vCenter Server Appliance 5`, `VMWare ESXi`, and `HP OpenStack Overview 7.5`, respectively. Ports `5053` and `1559` are used for TCP networking and Web2Host connections, respectively. The services offered by each machine are Ruby vSphere Console (`rcv`) on `m1`, Application Lifecycle Service (`als`) on `m2`, and Shared Trace Service (`ovtrcd`) on `m3`. The `rvc` service is a Linux console UI for the VMWare vCenter Server Appliance v5. Access to this console allows users authenticated remotely to execute arbitrary commands as root, potentially granting themselves administrative privileges. The `als` service offered by the HP Helion Cloud Development Platform allows users to create VMs using a seed node image, where all VMs sharing the same node image also share the same security key. Therefore a user having access to one of these VMs can also access the other ones created with the same seed node image. The `ovtrcd` service of HP OpenStack Overview 7.5 is used to log the actions performed by the OpenStack components for debugging purposes.

We also represent cloud users (administrators and customers) and their privileges, explicitly. In this initial scenario we have one administrator (`admin1`) who is authorised to login to `m1` and `m2` and perform copies of the images of the virtual machines hosted on `m1`.

The planning problem can be expressed in terms of a *domain* and a *problem definition*. The domain definition comprises a representation of the objects types, the legitimate and the malicious actions (attack modules) that can be performed, and the predicates adopted in the actions specification. The problem definition includes the ob-

jects instantiations in the cloud environment, their initial state and the potential crime goals.

The object types are defined using the `:types` keyword. Examples of object types representing virtual machines, physical machines, users, services and ports are expressed as follows.

```
(:types vmachine pmachine user service port - object)
```

Some object types have constant values and are specified using the `:constants` keyword. In particular, a set of constant names is followed by its type (preceded by symbol '-'). The following constants identify port `5053` and the `ovtrcd`, `rvc`, and `als` services.

```
(:constants port5053 - port
            ovtrcd rvc als - service)
```

The domain model also includes predicate definitions (keyword `:predicates`). The predicates are used to define the required pre- and post-condition of the possible actions. Each predicate is specified by a name and a set of parameters (`?<parameter name> - <parameter_type>`. Some of the predicates are presented below and allow identifying the physical machine hosting a VM (`allocated`), the operations an administrator is allowed to execute on a physical machine (`authorised_admin`) and the user authorised to use a VM (`authorised_cust`).

```
(:predicates (allocated ?vm - vMachine ?pm - pMachine)
             (authorised_admin ?a - user ?o - operation ?pm - pmachine)
             (authorised_cust ?c - user ?vm - vmachine))
```

The following example describes a legitimate action (`copy-vm`) that can be performed to copy a virtual machine. Actions are expressed in PDDL by using the `:action` keyword. An action is characterised by a name, pre- and post-conditions (effects), and a set of parameters used in the definition of such conditions. Pre-conditions and effects are expressed as a set of predicates in conjunction or disjunction. The pre-condition for copying a VM is that either a) a user is an administrator that is logged on the physical machine on which the VM is hosted and has the right to copy the VMs hosted on it, or b) the user is a customer who is logged an on the VM allocated to her and is authorised to perform a copy, or c) the VM is allocated on a compromised physical machine. If either of these conditions is satisfied the VM image can be copied.

```
(:action copy-vm
 :parameters (?vm - vmachine ?pm - pmachine ?d - jdomain ?a - user)
 :precondition (or (and (allocated ?vm ?pm) (logged ?a ?pm)
                        (admin_dom ?a ?d)(in_dom ?pm ?d)
                        (authorised_admin ?a copyvm ?pm))
                   (and (logged_cust ?a ?vm) (authorised_cust ?a ?vm)
                        (allocated ?vm ?pm))
                   (and (compromised ?pm) (allocated ?vm ?pm)))
 :effect (vm-copied ?vm ?a))
```

Attack modules describe the malicious actions that can be performed by an adversary to exploit the vulnerabilities brought by the software installed in the physical and

virtual machines that are present in the cloud configuration. We use the Common Vulnerability Exposure database (CVE) to identify the vulnerabilities associated with each software version and assess how they can be exploited. From this information the security administrator can define the action describing how the attack modules can exploit the vulnerabilities. Such actions are included in the domain definition.

For example, `VMWare vCenter Server Appliance` virtualisation software introduces a new vulnerability (`CVE-2014-3790`[4]). In particular, this vulnerability can be exploited by the `run-service-rvc` attack module, which can be performed if a user is logged on a VM allocated on a physical machine running VMWAre vCenter version 5.x. The effect of this attack module is to allow a VM user to run the rvc service on the physical machine in which the vm is allocated. This enables the precondition of `make-admin-local` attack module, and allows a user to grant herself the administrator privileges on the physical machine hosting the vm and the right to perform logins and copies of the VMs allocated on that machine.

```
(:action run-service-rvc
 :parameters (?c - user ?vm - vmachine ?pm - pmachine)
 :precondition (and (logged_cust ?c ?vm) (allocated ?vm ?pm)
                    (has_vm ?pm VMWare_vCenter)
                    (has_vm_version ?pm ver5))
 :effect (srv_running ?c rvc ?pm))

(:action make-admin-local
 :parameters (?c - user ?vm - vmachine ?pm - pmachine
              ?s - service ?d - jdomain)
 :precondition (and (logged_cust ?c ?vm) (allocated ?vm ?pm)
                    (srv_running ?c rvc ?pm)(in_dom ?pm ?d))
 :effect (and (authorised_admin ?c loginop ?pm) (admin_dom ?c ?d)
           (authorised_admin ?c copyvm ?pm) (admin ?c ?pm)))
```

Security breaches represent the goals of an adversary, such as unauthorised access to a customer's sensitive information or escalation of privileges on a physical or virtual machine. Security breaches are represented as `:goals` in the PDDL problem definition and represent the final state that must be achieved after the actions comprised in an attack scenario are executed. A possible security breach can be determined when there exists a malicious user who is authorised to use a VM (`authorised_cust`) that is not being assigned to him (`not (use_vm ...)`) and performs a copy of the VM image (`vm_copied`). This security breach is described by the following PDDL code snippet.

```
(:goal (exists (?vm - vmachine ?c - user)
               (and (authorised_cust ?c ?vm)
                    (not (use_vm ?c ?vm))
                    (vm_copied ?vm ?c))))
```

5.2. Planning

We use a planner to determine whether there exists a sequence of legitimate or malicious actions, given an initial state of the system, that would make it possible for an

[4] `http://www.cvedetails.com/cve/CVE-2014-3790/`

adversary to perpetrate a security breach. If such a sequence of actions exists, it forms an attack scenario. The planner accepts as input the PDDL domain definition described in the previous section and the problem definition. The problem definition represents concrete object instances, their initial states, and the objective to be achieved (i.e. security breach). The problem definition makes use of the objects types and predicates defined in the domain model of the cloud configuration. For example, in the problem definition below (`Prob1`) we leverage the domain definition partially presented in the previous section (`CloudDomain`) to represent objects instantiations, such as physical machines (`m1`, `m2` and `m3`), virtual machines (`v1`, `v2` and `v3`), users (`c1`, `c2`, `c3` and `admin1`) and administrative domains (`dom_a` and `dom_b`). The problem definition defines the initial state of the objects from which an attack scenario aimed to achieve a security breach should be perpetrated. In this example, each VM is hosted (`allocated`) on a different physical machine, each physical machine exists in an administrative domain and each customer is authorised to use a different VM. The problem definition also includes the security breach to be achieved (i.e. the crime goal described earlier).

```
(define (problem Prob1)
 (:domain CloudDomain)
 (:objects m1 m2 m3 - pmachine
           v1 v2 v3 - vMachine
           c1 c2 c3 admin1 - user
           dom_a dom_b - domain
           ..... )
 (:init (allocated v1 m1) (allocated v2 m2) (allocated v3 m3)
        (use_vm c1 v1) (use_vm c2 v2) (use_vm c3 v3)
        (in_dom m1 dom_a) (in_dom m2 dom_a) (in_dom m3 dom_b)
        ..... )
 (:goal (exists (?vm - vmachine ?c - user)
                (and (authorised_cust ?c ?vm)
                     (not (use_vm ?c ?vm))
                     (vm_copied ?vm ?c)))))
```

We model the domain and problem definitions by using PDDL 3.0 and use SGPlan5[5] to generate possible attack scenarios. The rest of the section describes examples of possible insider and outsider attack scenarios[6].

5.2.1. Insider Attacks

An insider attack originates from people within the organisation, such as employees, former employees, contractors or business associates, who have insider information concerning the organisation's security practices, data and computer systems. The domain and the problem definition reflect the cloud configuration presented in Figure 2. In this scenario the administrator (`admin1`) at the hosting company is authorised to login to the physical machines (`m1` and `m2`) of her authoritative domain (`dom_a`) and perform copies of the images of the VMs hosted on `m1`. Possible attack modules include

[5] `http://wah.cse.cuhk.edu.hk/wah/programs/SGPlan/`

[6] A complete description of the examples can be found at `http://lili-pasquale.lero.ie/attackscenarios.zip`

untrusted-search-path (presented in Section 4), which exploits a vulnerability present in VMWare vCenter SA 5 virtualisation software.

For example, a malicious adversary can be an administrator (user c) who aims to copy the image of a VM (e.g., v8 hosted on m2) containing sensitive information of a customer (predicate vm_copied), while she is logged to the physical machine hosting the VM (predicate logged_admin). This security breach can be expressed as follows.

```
(:goal (exists (?c - user) (and (logged_admin ?c m2) (vm_copied v8 ?c)))
```

To achieve the above goal and adversary can perform the following sequence of actions from the cloud configuration represented in Figure 2. These are the actions comprised in the attack scenario identified by the planner (Case 1.1).

```
0: (login-admin admin1 m2 dom_a)
1: (untrusted-search-path m2)
2: (mark-as-compromised m2)
3: (copy-vm v8 m2 dom_a admin1)
```

The administrator can login to m2, since she has the right to do so as m2 belongs to her administrative domain (dom_a). Subsequently, she can execute an untrusted-search-path to compromise m2 (i.e. gain root privileges), and copy the image of v8.

If the target virtual machine (e.g., v10) is hosted on a physical machine (m3) that does not belong to the authoritative domain of the administrator, the number of steps required to copy the image of v10 increases, as shown in the following attack scenario (Case 1.2).

```
0: (login-admin admin1 m2 dom_a)
1: (untrusted-search-path m2)
2: (mark-as-compromised m2)
3: (ip-connect m2 m3)
4: (tcp-connect m2 m3 port5053)
5: (run-service-ovtrcd admin1 m2 m3)
6: (hp-openview-remote-buffer-overflow-exploit m2 m3 admin1)
7: (mark-as-compromised m3)
8: (copy-vm v10 m3 dom_b admin1)
```

First, the administrator has to compromise m2 (steps 0-2), as explained in the previous attack scenario. Subsequently, she connects to m3 to run service ovtrcd on port 5053 and executes the HP openview remote buffer overflow attack module[7] (steps 3-6). This exploits a vulnerability present in Windows Xp ServicePack 2 (CVE-2007-4-349), which allows an adversary to assign herself the root privileges when the ovtrcd service is executed. Subsequently, m3 is marked as compromised and the admin can copy the image of v10 (steps 7-8).

[7] http://www.iss.net/security_center/reference/vuln/HP_OVTrace_Service_Overflow.htm

5.3. Outsider Attacks

Outsider attacks are perpetrated by individuals from outside the organisation who do not have information related to the organisation security practices, data and computer systems. Outsider attackers can be registered customers or external individuals. To generate the attack scenarios we refer to the cloud configuration specified in Figure 3. In particular, we specify the permissions of three customers: `c1` and `c3` use virtual machines `v1` and `v5`, respectively, hosted on `m1`, while `c2` uses `v7` hosted on `m2`. Based on this modified cloud configuration, a malicious customer (`c1` or `c3`) can aim to copy the image of a virtual machine (e.g., `v7`) allocated to another customer (e.g., `c2`). This security breach can be expressed as follows, where the goal of a malicious user *c* is to become an authorised customer of `v7` that is already in use by another customer.

```
(:goal ((exists (?c - user) (and (authorised_cust ?c V7)
                                 (not (use_vm ?c V7)))))
```

[Figure 3 about here.]

In this case the planner generates the following attack scenario (Case 2.1).

```
0: (login-cust c1 v1)
1: (run-service-rvc c1 v1 m1)
2: (make-admin-local c1 v1 m1)
3: (connect-over-network c1 m1 m2 port5053 rvc)
4: (allocate-vm v6 c1)
5: (login-cust c1 v6)
6: (get-service-als c1 v6 m2)
7: (connect-unowned-vm c1 v6 v7 m2)
```

For example, `c1` can perform the login to her virtual machine, since she has the right to do so and exploit the vulnerability [8] present in `m1` virtualisation software to run the `rvc` service. Using the this service `c1` can grant herself administrative privileges over `m1` (steps 0-2). By using such privileges she can establish a network connection with `m2` and allocate a new virtual machine (e.g., `v6`) on her behalf (steps 3-4). Subsequently, `c1` can login to the newly created virtual machine and exploit the vulnerability (`CVE-2014-7878`[9]) brought by the HP Helion Cloud Development Platform installed on `m2`. In particular, since all virtual machines created using the `als` seed node image have a universal security key by default, the customer can access the other VMs in the `als` cluster that share the same security key. In the attack scenario above `c1` exploits the security key generated for her VM (`v6`) to login to `v7`, which is allocated to another customer (steps 5-7).

[Figure 4 about here.]

[8] `http://www.cvedetails.com/cve/CVE-2014-3790/`
[9] `http://www.cvedetails.com/cve/CVE-2014-7878/`

In a second scenario, Case 2.2, a virtual machine related change takes place. In particular, customer c1 installs the Gecko Content Management System (CMS) v2 to host web content. The customer can manage the web content by accessing Gecko through a standard web browser, such as Google Chrome. The new cloud configuration is shown in Figure 4. Changes in the cloud configuration may determine changes in the attack modules available in order to reflect the vulnerabilities brought by the new software. In this case, for example, Gecko CMS v2 brings a cross-site request forgery (CSRF) vulnerability (CVE-2015-1424[10]), which allows remote adversaries to hijack the authentication of legitimate users. This can be achieved by including a link or script in a page that accesses a site to which the user is known (or is supposed) to have authenticated access. For example, an adversary can circumvent the customer by enforcing her to open an HTML image element that references an action on c1's virtual machine, such as a new_user request creating a new VM user. Due to the CMS vulnerability it is likely that c1's browser keeps her authentication information in a cookie. If the cookie has not yet expired, the attempt by c1's browser to load the image will submit the malicious request to v1 successfully without asking for c1's approval. As a result, the external adversary can login to v1 as a legitimate user.

For this case, we hypothesise that an unknown adversary (ext) aims to gain access to a specific virtual machine (v7 hosted on m2). The goal of the adversary in this case is the same as the one defined for Case 2.1. For this security breach, the planner generates the following attack scenario.

```
0: (login-cust c1 v1)
1: (connect-to-vm-browser c1 v1 port1559)
2: (make-user-vm-gecko c1 ext v1 m1)
3: (run-service-rvc ext v1 m1)
4: (make-admin-local ext v1 m1)
5: (connect-over-network ext m1 m2 port1559 rvc)
6: (allocate-vm v8 ext)
7: (login-cust ext v8)
8: (get-service-als ext v8 m2)
9: (connect-unowned-vm ext v8 v7 m2)
```

In this scenario, c1 logins to v1 and connects to the Gecko CMS through a web browser (steps 0-1). The external attacker ext can enforce the execution of a command from c1's browser that creates a new local user of v1 by exploiting the CSRF vulnerability brought by the CMS (step 2). Once ext has access to v1, she can exploit the rvc service to grant herself administrative privileges [11] (steps 3-4). Using the administrative privileges, she can subsequently connect to m2, as m2 runs VMWare ESXi, and allocate a new virtual machine (v8) (steps 5-6). The external adversary can finally exploit the HP Helion Cloud Development Platform vulnerability [12] to gain access to v7 hosted on m2 (steps 7-9).

In the last case (Case 2.3) we deal with an unknown adversary (ext) trying to gain access to the database content (d) associated with a specific VM (e.g., v7). The

[10] http://www.cvedetails.com/cve/CVE-2015-1424/
[11] http://www.cvedetails.com/cve/CVE-2014-3790/
[12] http://www.cvedetails.com/cve/CVE-2014-7878/

cloud environment is the same as the one presented in Figure 4. This is a relevant case as it resembles the security breach [2] that arised at JP Morgan Chase last year. This security breach can be expressed as follows, where the goal of an adversary (`ext`) is to become an authorised user of `v7` (predicate `authorised_cust`) that is already in use by another customer (predicate `not(use_vm(...))`) and copy the content of a database (predicate `db_copied`) stored on `v7` (predicate `stored`).

```
(:goal ((exists (?ext - user ?d - database)
                (and (authorised_cust ?ext v7)
                     (not (use_vm ?ext v7))
                     (stored ?d v7)
                     (db_copied ?ext v7 ?d)))))
```

The attack scenario generated by the planner slightly extends the one presented in the previous case adding additional actions related to the database.

```
 0: (login-cust c1 v1)
 1: (connect-to-vm-browser c1 v1 port1559)
 2: (make-user-vm-gecko c1 ext v1 m1)
 3: (run-service-rvc ext v1 m1)
 4: (make-admin-local ext v1 m1)
 5: (connect-over-network ext m1 m2 port5053 rvc)
 6: (allocate-vm v8 ext)
 7: (login-cust ext v8)
 8: (get-service-als ext v8 m2)
 9: (connect-unowned-vm ext v8 v7 m2)
10: (connect-db ext v7 db1)
11: (query-db ext v7 db1)
12: (copy-db ext v7 db1)
```

In particular, once `ext` has gained access to `v7` (steps 0-9), she can connect to the database `db1` using the same credentials she has used to access `v7` (step 10) in order to perform a query on the relevant tables and subsequently copy their content (steps 11-12). In this case an adversary is able to gain access to the database content since the cloud provider does not require two-factors authentication to access the resources (database) installed in the VMs.

6. Evidence Collection and Monitoring

The attack scenarios generated from the representation of the cloud configuration, the security breaches, and the attack modules are used to configure the evidence collection activities. We also define the monitoring activity to identify potential changes that may affect the cloud configuration and cause modifications in the attacks scenarios. The remainder of this section describes the evidence collection and monitoring activities.

6.1. Evidence Collection

The evidence collection activity aims to identify and preserve data indicating that the basic actions comprised in the attack scenarios are taking place. During a digital investigation, the evidence collected helps demonstrate if and how a security breach

took place. Moreover, it allows identifying the individuals (customers and providers) responsible for introducing the vulnerabilities exploited by the malicious actions comprised in the attack scenario.

We map each action in the domain definition to a specific log entry of the software affected by the action execution. As the format for log data varies based on the software generating the logs we consider a generic and standard format to represent log entries. Our format includes the following fields: date and time of action, action identifier, user identifier, user IP, additional parameters, action effects, and log source. Additional parameters cover those included in the action representation modelled in the domain definition. The log source allow identifying the software and the virtual and/or physical machine generating the log entry. However, not all fields identified in our generic format have to be mapped to the specific fields of a log entry. For example, the log entries related to a user's login request to a VM only include her IP address, while the log entries identifying the authorisation performed during the login can also include the user ID and her credentials.

[Figure 5 about here.]

An example of a log entry extracted from the log of a VMWare ESXi Client application recording user login authorisation to access a specific VM is presented in Figure 5. This log entry is associated with the `login-cust` action in the domain definition. This figure maps the log entry fields to our generic log format. Note that the parameter of the `login-cust` action include the user ID and the VM a user is trying to access. The former parameter is inferred directly from the log entry, while the VM is identified from the block of grouped log entries pertaining to that VM (e.g., `VMx`). Note that the format of the log entries varies depending on their source. For example the action ID (`[FFA2AD20 info 'Vimsvc.ha-eventmgr' opID=4A67C3DF-00000003]`) in the log entry defined in Figure 5, would be different if another virtualisation software client is adopted. Therefore a mapping between the actions present in the domain definition and the ID's identified in the log entries must be defined at the initial setup by the system administrator and updated based on the configuration changes in the cloud.

Log sources may differ depending on the cloud configuration. A broad categorisation of the sources of log data is provided as follows.

- Application Logs are generated by any software installed in the cloud environment, such as Content Management Software (Gecko CMS), Virtual Machine Managers (VMWare vCenter or ESXi) or Cloud development platforms (HP Helion).
- Web Server Logs include information (e.g., user IP addresses, request time/date, address of the VM being accessed) about accesses to the VMs.
- Database Logs contain information generated by database management systems (e.g., IBM DB2, Oracle DMS, MongoDB) about queries performed to existing databases, the content associated with a query and modifications of databases content.

- Network Logs are generated by both the web server and the operating system running on a physical machine. They record information about how network connections are established, the data traffic on the network and who requested the connection.
- System Logs are generated by the operating systems running on the physical machines belonging to the cloud deployment. For example, VMWare WorkStation record information about the use and performance of resources. These logs can be further categorised into:
  - Identity Manager Logs containing data pertaining authentication attempts.
  - Deployment Logs recording how various resources, such as databases, storage and development platforms, are deployed across the cloud, their virtual and physical addresses and how and by whom they can be accessed.
  - Security Logs containing data about login attempts, and granting of user privileges.

The attack scenario presented in Case 2.1 described in Section 5.3, deals with an authenticated customer gaining access to a VM allocated to another customer. An example of the log data collected for this attack scenario is presented in Table 1. For example, the log entry corresponding to the `login-cust` action is inferred from the web server, application and system logs. The web server logs provide the user IP and the date and time a user tries to login to her VM. The application logs provide information about which VM a user is attempting to connect to while the system logs include information regarding users' authentication and its corresponding outcome.

[Table 1 about here.]

6.2. Monitoring

Changes in the cloud configuration, i.e. virtual machine or physical machine changes described in Section 3, may require the regeneration of the attack scenarios and, consequently, modification of the evidence collection strategy. Therefore data identifying virtual and physical machine changes must be monitored automatically. Examples of relevant changes are those determining the allocation/deallocation of VMs on a physical machine. The data to be monitored, pertaining to the allocation/deallocation of VMs, would be generated by the virtualisation software installed on the physical machines. Other changes to be monitored are modifications of the software configuration of physical and virtual machines. In order to identify software configuration changes, the system logs, specifically deployment logs, generated by the operating system and virtualisation software deployed on different physical machines must be monitored. These changes also require the system administrators to identify the vulnerabilities introduced/removed by the software installed/uninstalled, and to update the attack modules included in the domain definition accordingly.

The evidence collection strategy must be updated in order to reflect the changes in the attack scenarios. If the new attack scenarios do not comprise actions that were included in the previous evidence collection strategy, the log entries associated with

the execution of such actions must be removed from the evidence collection strategy. In particular if the attack scenarios comprise new (legitimate or malicious) actions that were not identified in the previous set of attack scenarios, the log entries associated with those actions must be included in the evidence collection strategy. For example, consider the attack scenarios generated for Case 2.1 and Case 2.2 described in Section 5.3. The difference in the cloud configuration between cases 2.1 and 2.2 is in the installation of `Gecko CMS` in `v1`. The `Gecko CMS` software introduces a new vulnerability into the cloud as described in Section 5.3. The vulnerability introduces a new set of attack modules that can be executed by an adversary to gain access to a VM that she is not authorised to use. When the monitoring activity detects the software configuration changes, it regenerates the attack scenarios based on the new vulnerabilities introduced by the change and the evidence collection activity is adapted accordingly. Table 2 identifies the differences in the actions comprised in either attack scenario. Additional actions included in the attack scenario generated for Case 2.2 are `connect-to-vm-browser` and `make-user-vm-gecko`. These new actions require the logging of access attempts performed by legitimate customers and the authorisation of external users to access a VM. Data pertaining to these actions can be found in the Web Server logs, Identity Manager logs and Security Logs.

[Table 2 about here.]

7. Evaluation

We evaluated our approach from both a qualitative and a quantitative perspective. From a qualitative perspective we estimate the percentage of data that our approach avoids collecting, compared to the case in which all possible data are preserved. We also discuss the effectiveness of our approach in handling false positives and negatives. From a quantitative perspective, we measured the overhead of generating attack scenarios and performing evidence collection activities in a real cloud environment.

7.1. Qualitative Evaluation

Our approach for adaptive evidence collection reduces the quantity of log data to be collected and stored for future use during digital investigations of security breaches in the cloud. The log data comprises events generated from the software installed in the physical machines, network traffic, allocation/deallocation of virtual machines, and changes in their software configuration. Preserving all the log data increases the operational cost due to the requirement of external storage capabilities. It also increases the effort necessary to forensic examiners and automated tools to analyse the data in order to reconstruct the events leading to a security breach. As an example, let us consider a medium sized organisation [27] composed of around 250 users, 250 user end points, 5 offices, 2 subnets, 2 databases, and a central data centre. We can assume that each subnet has an IPS, a switch and gateway/router, and the whole organisation has 2 firewalls and a VPN. In such a scenario potential evidence includes events generated from each user end point, such as login/logout, files access/creation/modification/deletion, and network traffic. We estimated having 50 Events Per Second (EPS) during non peaks and 2500 EPS during peaks. If an organisation experiences peaks for 5% of the total

time, we will have an average of 215 EPS (125 EPS for non-peak and 90 for peak) and, consequently, around 278640000 events in 15 days. Assuming that each event occupies around 50KB, the size of the database storing events happening within 15 days should be at least around 15TB.

Our approach uses attack scenarios to focus data collection only on the events that might be relevant for investigating possible security breaches, i.e. those determining whether the steps of the attack scenarios took place. Quantifying the reduction in the evidence to be collected highly depends on the cloud configuration and on the actions comprising the attack scenarios. In this section, we compare the number of actions whose log entries that the attack scenarios prescribe to collect with all possible log entries that can be collected. These possible log entries correspond to the actions defined within the domain definition of the cloud environment as described in Section 4.

Table 3 shows a rough estimation in the reduction of collected evidence for all cases presented in Section 5. For example, the cloud configuration shown in Figure 2 includes 11 unique action types, dealing with the creation, allocation and deallocation of VMs, the authentication and authorisation of users, and the execution of services and malicious actions exploiting vulnerabilities. The attack scenario generated in Case 1.1 for this cloud configuration only includes 4 unique action types. Therefore, our approach only prescribes to preserve the log entries associated with the action types identified in the attack scenario as opposed to those identified in the domain definition, reducing the amount of log data preserved by 63%. As described earlier, when a new attack scenario is generated any additional action to be monitored is added to the evidence collection activity. Starting from Case 1.1, we include attack scenarios incrementally, adapt the evidence collection activity accordingly, and calculate the reduction in the amount of data collected. However, note that this an estimated theoretical measure that depends on the cloud configuration adopted. As described in Section 7.2, in real cloud environments this measure can vary depending on the number of actions that are executed at runtime, which determine the amount of log entries that are generated dynamically.

[Table 3 about here.]

Another advantage of our approach is that it adapts the data collection strategies as soon as changes in the cloud configuration or in the available attack modules are detected. These modifications determine the generation of new/different attack scenarios that drive the adaptation of the evidence collection strategies.

False positives and negatives can threaten the validity of our approach. False positives correspond to attack scenarios that do not lead to security breaches. This can happen when a vulnerability exploited by an attack module comprised in the attack scenario has been fixed. For example, if we consider the second insider attack described in Section 5.2.1, we can give rise to a false positive if Windows Xp ServicePack 2 installed on M3 is updated and its vulnerability (`CVE-2007-4-349`) is fixed. Additionally, false positives can also arise when specific security controls have been put in place by cloud security administrators to prevent generated attack scenarios. For example, to make it less likely or invalidate the insider attack scenarios described in Section 5.2.1, additional authentication can be required to perform a copy of a VM.

Although false positives are undesirable, they do not pose risks in terms of loss of data that might be relevant in future digital forensic investigations. Instead, they might cause the collection of irrelevant evidence. False positives can be avoided by re-generating the attack scenarios over an updated domain model reflecting the current cloud configuration, where the legitimate and malicious actions reflect the information coming from the security controls and the vulnerability databases, respectively.

False negatives can pose higher security risks since they represent the missed identification of potential attack scenarios, which can cause the loss of evidence that might be relevant in future digital forensic investigations. False negatives might arise from an incomplete domain definition that does not include all cloud configuration components (e.g., installed software, allocated VMs and storage), it might neglect legitimate actions that a user is allowed to perform, or it might not cover all possible attack modules exploiting known vulnerabilities. In such cases, false negatives can be avoided by constantly updating the representation of the cloud configuration and the attack modules as soon as changes in the real cloud platform take place or when vulnerability databases are updated with new vulnerabilities that can be brought by the software installed in the cloud platform. In other cases, false negatives might be caused by vulnerabilities that are not known to the software vendor and might lead to unexpected zero-day attacks. To address false negatives a possible solution would be to preserve all the logs associated with the software components of the cloud configuration. However, this solution might be inefficient as it can have increased time and storage overheads. An alternative solution would be to use honeypots [28], which are decoy servers used as a trap to detect and analyse (new) malware, the vulnerabilities they exploit and their possible sources. Malware identified with honeypots can be used to update the domain definition with new attack modules, triggering an update in the possible attack scenarios and in the evidence collection activities. However, we recognise that malware analysis cannot always be fully automated. Further investigation on the use of honeypots to update attack modules will be addressed in future work.

7.2. Quantitative Evaluation

We assessed the efficiency of the planner (SGPlan5) in generating attack scenarios for cloud configurations of increasing complexity, and the absolute and relative overhead in terms of storage capacity and time necessary to perform evidence collection activities in a real cloud infrastructure.

7.2.1. Planning Efficiency

We measured the efficiency of SGPlan5 for generating attack scenarios for cloud configurations of increasing size (problem expansion) and attack scenarios comprising an increasing number of actions (attack expansion). Our experiments were conducted on an Ubuntu (64bit) virtual machine using 6GB RAM and hosted on a Mac OS 10.10.2, with 2.6GHz Intel Core I7 processor and 16GB RAM.

Problem Expansion. For the first experiment we consider a cloud configuration where the size of the problem definition is expanded at each iteration, while the number of steps of the generated attack scenarios remains constant (8 steps). We expand the size of the problem definition by starting from the cloud configuration defined in Section 5.1, hereafter referred to as a machine cluster. In particular, our machine cluster

consists of 10 virtual machines, hosted on 3 physical machines, where each physical machine in the cluster can establish a network connection to each other. Each cluster is connected to another one in sequence, i.e. a machine in the cluster can establish a network connection with a machine belonging to a subsequent cluster. In this way all machines in a cluster are reachable by traversing a specific number of machines in the cloud configuration. We start from 5 machine clusters and progressively add 5 machine clusters to the cloud configuration. The crime objective is to gain credentials to login to `m2` belonging to the first cluster included in the cloud configuration. The time taken by the planner to generate the attack scenarios for cloud configurations having an increasing number of entities (e.g., VMs, physical machines, and networks) is presented in Table 4. From our results, we can deduce that the time increases exponentially depending on the size of the problem definition.

[Table 4 about here.]

Attack Expansion. In this case we adopt and maintain a cloud configuration comprising 30 machine clusters. The crime goal at each iteration is to gain login credentials to access a physical machine belonging to a farther machine cluster. In particular, at each iteration the crime goal changes in order to enforce the generation of attack scenarios that should traverse five more clusters. The time taken by SGPlan5 to generate the attack scenarios comprising an increasing number of actions is presented in Table 5. From this table we can deduce that the complexity of the attack scenario does have an impact on the efficiency of the planner. However this is marginal compared to the impact determined by the cloud configuration size shown in Table 4.

[Table 5 about here.]

7.2.2. Overhead of Evidence Collection Activities

To assess the overhead of evidence collection activities we developed a more complete example leveraging databases and storage resources. Our example was deployed on the Google Cloud Platform[13]; this choice allows repeatability of results as Google Cloud Platform provides consistent CPU, memory and disk performance.

The cloud configuration of our example is very similar to that proposed to explain the attack scenarios generation in Section 5. It comprises physical machines `m1` and `m2`, which were hosted in North America, and physical machine `m3`, which was located in Europe. Each physical machine can accept network connections from the others and can host some VMs. In particular, `m1` hosted `v1`, `v3`, `v5` and `v6`, `m2` hosted `v4`, `v8` and `v9`, and `m3` hosted `v2` and `v7`. As the software configuration of each physical machine is not released publicly by Google Cloud Platform, we assumed that the service provider has no control over it. Each VM type was `f1-micro`, which allocates a single CPU on the Intel Sandy Bridge platform with a memory of 0.6GB. The operating system of each VM was Ubuntu 14.04 LTS. Virtual machines `v1` to `v4` served as backend servers controlled by the cloud service provider while `v5` to `v9`

[13] `http://cloud.google.com`

were designated as allocatable frontends, which could be outsourced to customers to run their own applications.

[Figure 6 about here.]

The backend VMs ran the MongoDB database software which provided database access to the applications running on the frontend VMs. This example focuses on the vulnerabilities introduced by MongoDB, specifically CVE-2013-4650[14] and CVE-2014-2917[15]. The vulnerability described in CVE-2013-4650 allows an authenticated user to obtain internal system privileges by leveraging username "__system". This would allow a user to escalate her privileges and gain access to data in the database associated with other VMs not allocated to her. The vulnerability described in CVE-2014-2917 allows a user with the appropriate privileges to copy the credentials granted to other users of the database. This was caused by MongoDB disclosing the users' credentials in its log files, which are accessible by users having root privileges. In this example we consider two users c1 and c2. The frontend VMs v5, v6 and v7 are allocated to c1, while v8 and v9 are allocated to c2. The database db1 is hosted on v4. The security breach we considered in this example was c2 being able to copy the credentials of c1. The attack scenario generated for this security breach is the following:

```
0: (login-cust c2 v9)
1: (request_db c2 v9)
2: (allocate_db c2 v9 v4 db1)
3: (authorise_to_db c2 v9 db1)
4: (mask_username c2 v9 db1 _system)
5: (escalate_privilege c2 v9 db1 high)
6: (login_db c1 db1)
7: (get_service_cmdline c2 v9 db1)
8: (copy_credentials c2 c1 db1)
```

In this scenario, c2 performs the login to v9 and requests access to the database (steps 0-1). Then, database db1 is allocated to v9 used by c2 (step 2), and MongoDB authorises an application installed by c2 on v9 to access the database (step 3). Subsequently, c2 exploits a vulnerability of MongoDB to mask her username and escalate her privileges (steps 4-5). Once c2 has escalated her privileges, she can gain command line access and copy the credentials of another user, e.g., c1 (steps 7-8). This is due to the fact that anytime a user accesses the database, her credentials are written in the MongoDB log file (steps 6). Once c2 has copied c1's credentials, she can execute the same operations c1 is granted to perform.

The operations identified in the attack scenario above were mapped to relevant log files, as shown in Table 6. Column *Action ID* identifies the action performed at each step of the attack scenario, while columns *Log Type* and *Log Name* refer respectively to the type and name of the log file from which evidence that the corresponding action is taking place should be collected. We also distinguish whether the log file is located

[14] http://www.cvedetails.com/cve/CVE-2013-4650/
[15] https://jira.mongodb.org/browse/SERVER-13644/

on the backend or frontend VMs; this allows us to separate responsibility for evidence collection between the service provider and the customer, respectively. The log entries relevant to the attack scenario were extracted and stored in a separate file.

For this example, log types can refer to: a) *Cloud System Logs*, containing operations related to the deployment of databases and VMs on physical machines, b) *VM System Logs* registering the operations performed within the VM by a customer such as installation of applications, accesses to a database, copies of data, and c) *Application Logs*, generated by the applications installed on a VM and containing user access history, operations performed and resources used. The log files from which evidence was extracted are the following:

- *auth.log* contains log entries pertaining to the authorisation of customers logging to VMs. Each log entry includes the following fields: time, customer's IP address, and SSH key used to perform the login.
- *activity_log.json* records the system level operations performed by the administrator to handle customers' requests and resource allocation.
- *mongodb.log* includes the operations performed during the usage of the MongoDB application.
- *syslog.log* is the system log generated by the operating system running on a VM.

[Table 6 about here.]

As the frontend VMs of our scenario ran a web-based application accessing the MongoDB database, we simulated the customers' utilisation of such applications by reproducing the behaviour (read/write operations) of the Yahoo! Cloud Serving Benchmark (YCSB)[16], which is used to compare the performance of NoSQL database management systems, such as MongoDB. In particular we considered the following cases:

- Case 1 considers the actions necessary to set up the cloud configuration shown in Figure 6 and those performed by customer `c2` to perpetrate the security breach illustrated previously.
- Case 2 considers the actions of Case 1 and assumes that the interaction of the customers with the frontend services generated 10000 operations requiring access to MongoDB.
- Case 3 considers the actions of Case 1 and assumes that the interaction of the customers with the frontend services generated 50000 operations requiring access to MongoDB.
- Case 4 considers the actions of Case 1 and assumes that the interaction of the customers with the frontend services generated 100000 operations requiring access to MongoDB.

[16]https://github.com/brianfrankcooper/YCSB

Table 7 shows the operations generated for each case and the absolute and relative storage overheads determined by evidence collection activities. The relative storage overhead obtained by preserving only the log entries associated with the steps of the attack scenarios was about 30 KB for each of the four cases. As it is possible to note the reduction in the storage overhead increases as the number of operations that are not related to the attack scenario increases. In particular, the reduction in the storage overhead was $\sim 96\%$ in Case 1 and reached $\sim 99.9\%$ in Case 4.

[Table 7 about here.]

As the time overhead determined by the attack scenarios generation has already been discussed in Section 7.2.1, we now assess the time overhead determined by the log extraction operations performed during evidence collection. In particular, we compare the time taken to extract the logs necessary to detect the attack scenario ($\tilde{3}$0KB) with that necessary to extract an arbitrary number of logs proportional to the number of customers' operations performed in Cases 2-4. Table 8 presents the amount of logs extracted and the corresponding time overhead. From our results we can conclude that the time overhead introduced by the evidence collection activities is minimal and linearly increases with the amount of logs to be extracted.

[Table 8 about here.]

8. Conclusion

In this paper we proposed the use of attack scenarios to configure evidence collection activities at the cloud service provider premises. In particular, we focused these activities on the preservation of the data necessary to detect the attack scenarios that can be perpetrated within the current cloud deployment. Moreover, we adapted evidence collection activities depending on changes in the cloud configuration or in the vulnerabilities that are brought by the software installed in the physical and virtual machines present in the cloud configuration. The method proposed in this paper can collect forensic evidence for those attacks exploiting known vulnerabilities and does not consider zero-day attacks exploiting unknown vulnerabilities. It is out of the scope of this paper to preserve evidence for security breaches determined by unknown vulnerabilities and this problem will be addressed in future work.

Our results demonstrate that using attack scenarios allows reducing the data collected in the cloud by focusing on those security breaches that are likely, while saving space and time necessary to store and process such data. Furthermore, planning techniques for generating the attack scenarios exhibit negligible times even for realistic data centres including 90 physical machines and 300 VMs. To generate attack scenarios for bigger cloud deployments in future work we will also investigate techniques to partition the domain and problem definition representing the cloud configuration. Although our approach has been applied to attack scenarios targeting cloud customers and providers of a IaaS cloud deployment, its benefits also apply to other cloud deployments, such as PaaS (Platform as a Service) and SaaS (Software as a Service).

One of the limitations of our approach is the potential loss of relevant evidence in case some of the possible attack scenarios are not generated. To address this issue,

we will investigate the use of *mutation testing* [29, 30, 31], which has been used to evaluate the quality of generated software test cases. Mutation testing aims to apply small modifications to a software program in order to check whether the test cases can identify the introduced fault. We can reuse mutation testing to apply small changes to the cloud configuration in order to identify whether some additional attack scenario can be realised. As described in Section 7.1, another solution to identify a more complete set of attack scenarios is to analyse malware collected through honeypots. In particular, we will analyse bot malware, which is self-propagating malware that infects a host and connects back to a central server forming a network of compromised devices. The identification of attack scenarios taking into account this kind of malware can be particularly useful for considering additional kinds of attack scenarios in which the cloud is used as a vehicle to perpetrate an attack. Finally we will map the model of the cloud configuration to existing cloud security reference architectures (e.g., [24]); this will make our approach more systematic as it would allow us to discover a more complete set of security breaches and attack scenarios arising from the interaction of the stakeholders with the cloud system.

Finally, we recognise that a further trigger for adapting changes in evidence collection is jurisdictional changes. These changes refer to the modification of the privacy and security regulations within the jurisdiction for which the cloud resources belong. A new regulation might allow for the collection of additional data or might further restrict what data can be collected. In either case, the evidence collection activity must adapt. How to adapt evidence collection activities depending on jurisdictional changes will be explored in future work.

Acknowledgment

This work was partially supported by the Science Foundation Ireland grants 10/CE/-I1855 and 13/RC/2094, and the ERC Advanced Grant no. 291652 (ASAP).

References

[1] M. S. Blumenthal, Hide and seek in the cloud, IEEE Security & Privacy 8 (2) (2010) 57–58.

[2] J. Cook, Tech More: JP Morgan Hacking JP Morgan Got Hacked Because It Forgot To Enable Two-Factor Authentication On A Server, `http://tinyurl.com/p5226yd` (2014).

[3] R. Rowlingson, A Ten Step Process for Forensic Readiness, International Journal of Digital Evidence 2 (3) (2004) 1–28.

[4] C. Sarraute, Automated attack planning, CoRR abs/1307.7808.

[5] E. Gutesman, A. Waissbein, The Impact of Predicting Attacker Tools in Security Risk Assessments, in: Proc. of the 6th Workshop on Cyber Security and Information Intelligence Research, 2010, pp. 75:1–75:4.

[6] L. Krautsevich, F. Martinelli, A. Yautsiukhin, Towards Modelling Adaptive Attacker's Behaviour, in: Foundations and Practice of Security, Springer, 2013, pp. 357–364.

[7] C. Sarraute, O. Buffet, J. Hoffmann, POMDPs Make Better Hackers: Accounting for Uncertainty in Penetration Testing, in: Proc. of the 26th AAAI Conference on Artificial Intelligence, 2012.

[8] E. LeMay, M. D. Ford, K. Keefe, W. H. Sanders, C. Muehrcke, Model-based Security Metrics Using Adversary View Security Evaluation (advise), in: Proc. of the 8th International Conference on Quantitative Evaluation of Systems, 2011, pp. 191–200.

[9] L. Pasquale, Y. Yu, M. Salehie, L. Cavallaro, T. T. Tun, B. Nuseibeh, Requirements-Driven Adaptive Digital Forensics, in: Proc. of the 21st Int. Requirements Engineering Conf., 2013, pp. 340–341.

[10] L. Pasquale, Y. Yu, L. Cavallaro, M. Salehie, T. T. Tun, B. Nuseibeh, Engineering Adaptive Digital Investigations using Forensics Requirements, CoRR abs/1402.0997.

[11] J. Dykstra, A. T. Sherman, Understanding Issues in Cloud Forensics: Two Hypothetical Case Studies, in: Proceedings of the Conference on Digital Forensics, Security and Law, 2011, pp. 45–54.

[12] S. D. Wolthusen, Overcast: Forensic Discovery in Cloud Environments, in: Proc. of the 5th International Conference on IT Security Incident Management and IT Forensics, 2009, pp. 3–9.

[13] M. Taylor, J. Haggerty, D. Gresty, D. Lamb, Forensic Investigation of Cloud Computing Systems, Network Security 2011 (3) (2011) 4–10.

[14] K. Ruan, J. Carthy, T. Kechadi, M. Crosbie, Cloud Forensics: An Overview, Proc. of the 7th IFIP Int. Conf. on Digital Forensics.

[15] G. Grispos, T. Storer, W. B. Glisson, Calm Before the Storm: The Challenges of Cloud Computing in Digital Forensics, Vol. 4 Issue 2, IGI Global, 2012.

[16] D. Reilly, C. Wren, T. Berry, Cloud Computing: Forensic Challenges for Law Enforcement, in: Proc. of the 5th International Conference for Internet Technology and Secured Transactions, 2010, pp. 1–7.

[17] D. Birk, C. Wegener, Technical Issues of Forensic Investigations in Cloud Computing Environments, in: Proc. of the 6th International Workshop on Systematic Approaches to Digital Forensic Engineering, 2011, pp. 1–10.

[18] J. Dykstra, A. T. Sherman, Acquiring Forensic Evidence from Infrastructure-as-a-Service Cloud Computing, Digital Investigation 9 (2012) S90–S98.

[19] Guidance Software, EnCase Forensics – Computer Forensics Data Collection for Digital Evidence Examiners, http://www.guidancesoftare.com/encase-forensics.htm.

[20] AccessData, FTK – AccessData Digital Forensics Software, http://www.accessdata.com/products/digital-forensics/ftk.

[21] J. Dykstra, A. T. Sherman, Design and Implementation of FROST: Digital Forensic Tools for the OpenStack Cloud Computing Platform, Digital Investigation 10, Supplement (0) (2013) S87 – S95.

[22] C. Shields, O. Frieder, M. Maloof, A System for the Proactive, Continuous, and Efficient Collection of Digital Forensic Evidence, in: Digital Investigations, Vol. 8, 2011, pp. 3–13.

[23] S. Pearson, R. Watson, Digital Triage Forensics: Processing the Digital Crime Scene, Syngress, 2010.

[24] E. B. Fernandez, R. Monge, K. Hashizume, Building a Security Reference Architecture for Cloud Systems, Requirements Engineering (2015) 1–25.

[25] J. C. Pelaez, E. B. Fernández, M. M. Larrondo-Petrie, Misuse Patterns in VoIP, Security and Communication Networks 2 (6) (2009) 635–653.

[26] W. R. Claycomb, A. Nicoll, Insider Threats to Cloud Computing: Directions for New Research Challenges, in: Proc. of the 36th Conf. of Computer Software and Applications Conference, 2012, pp. 387–394.

[27] M. Butler, Benchmarking Security Information Event Management (SIEM), http://tinyurl.com/q2vdd5s (2009).

[28] K. G. Anagnostakis, S. Sidiroglou, P. Akritidis, K. Xinidis, E. Markatos, A. D. Keromytis, Detecting Targeted Attacks Using Shadow Honeypots, in: Proc. of the 14th Conf. on USENIX Security Symposium, 2005, pp. 9–9.

[29] E. Martin, T. Xie, A Fault Model and Mutation Testing of Access Control Policies, in: Proc. of the 16th International Conference on World Wide Web, 2007, pp. 667–676.

[30] Y. Maezawa, H. Washizaki, S. Honiden, Extracting Interaction-Based Stateful Behavior in Rich Internet Applications, in: Proc. of the 16th European Conference on Software Maintenance and Reengineering, 2012, pp. 423–428.

[31] Y. L. Traon, T. Mouelhi, B. Baudry, Testing Security Policies: Going Beyond Functional Testing, in: Proc. of the 18th International Symposium on Software Reliability, 2007, pp. 93–102.

List of Figures

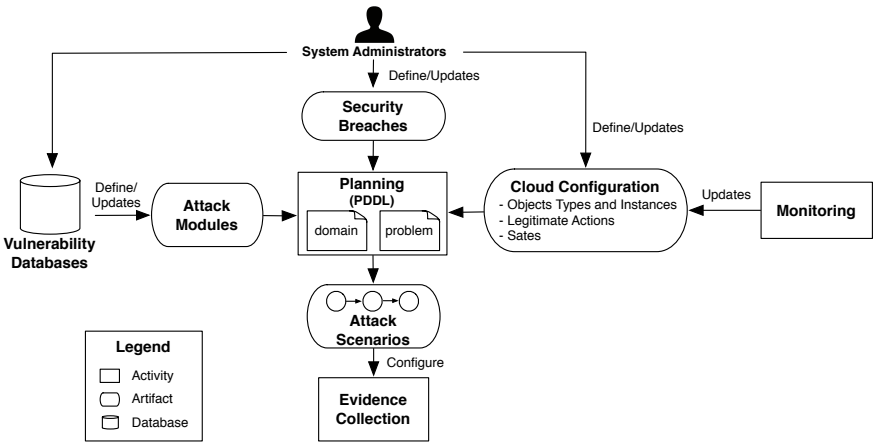

Figure 1: Overview of our approach for adaptive evidence collection.

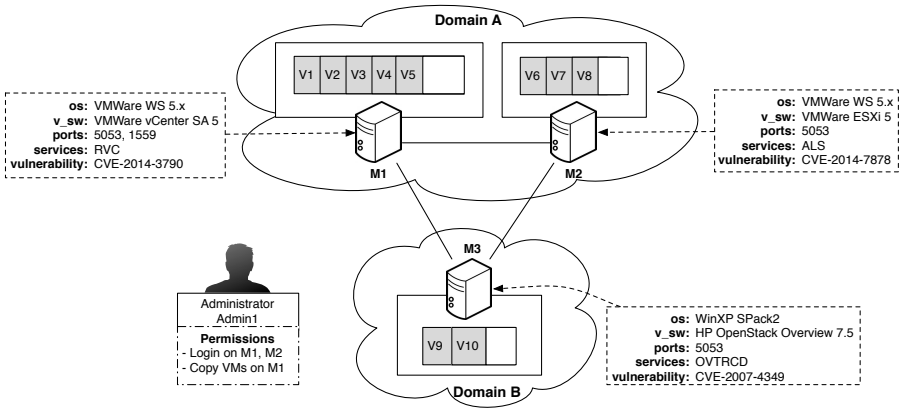

Figure 2: Initial Cloud Configuration.

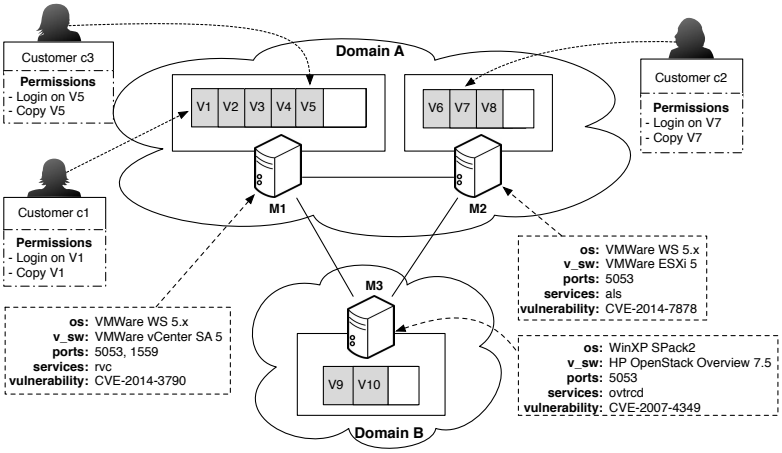

Figure 3: Cloud Configuration - Known Outsider Attack.

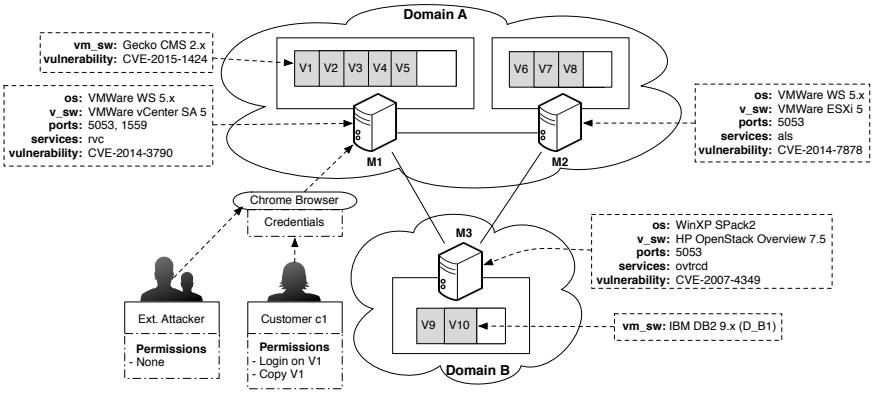

Figure 4: Cloud Environment - Unknown Outsider Attack.

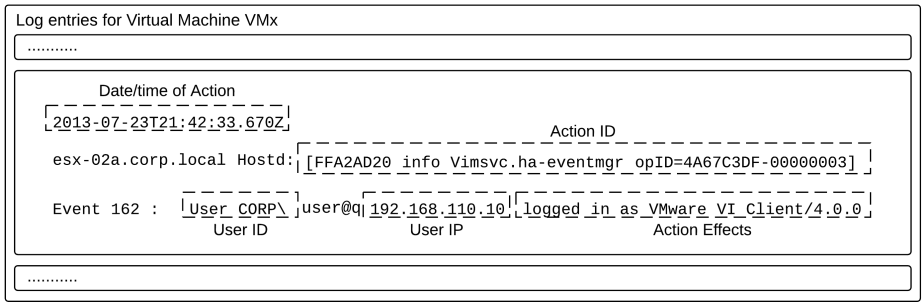

Figure 5: VMWare ESXi VI Client Log entry: User login authorisation to access a VM.

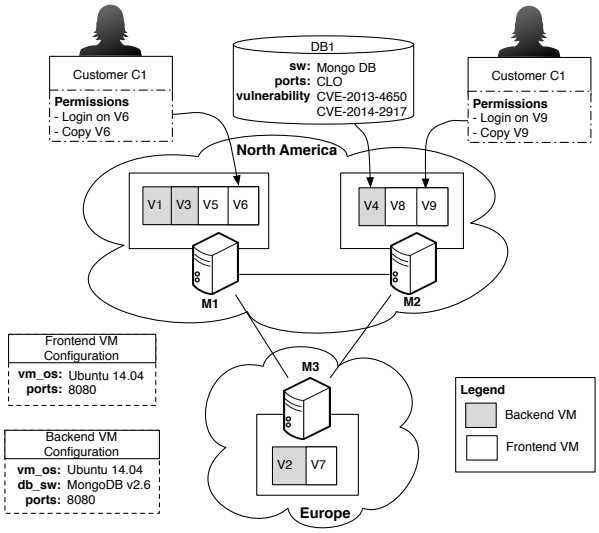

Figure 6: Evaluation Cloud Configuration.

List of Tables

Action #	User IP	User ID	Date/Time of Action	Action ID	Action Parameters	Action Effects	Source
0.	209.121.62.135	C1	[15/Mar/2015 17:37:23 +0:00]	login-cust	[C1 V5]	"logged_cust C1 V5"	Web Server Logs, Application Logs, System Logs
1.		C1	[15/Mar/2015 17:37:33 +0:00]	run-service-rvc	[C1 V5 M1]	"has_service C1 rvc M1"	System Logs, Application Logs
2.		C1	[15/Mar/2015 17:37:43 +0:00]	make-admin-local	[C1 V5 M1 ALS DA]	"authorized_admin C1 loginop M1" "admin-dom C1 DA" "authorized_admin C1 copyvm M1" "admin C1 M1"	System Logs
3.	209.121.62.135	C1	[15/Mar/2015 17:37:53 +0:00]	connect-over-network	[C1 M1 M2 port5053 RVC]	"has_credentials C1 M2"	Network Logs, Web Server Logs
4.	209.121.62.135	C1	[15/Mar/2015 17:38:03 +0:00]	allocate-vm	[V6 C1]	"use-vm C1 V6" "authorized_cust C1 V6"	Application Logs
5.	209.121.62.135	C1	[15/Mar/2015 17:38:13 +0:00]	login-cust	[C1 V6]	"logged_cust C1 V6"	Web Server Logs, System Logs
6.		C1	[15/Mar/2015 17:38:23 +0:00]	get-service-als	[C1 V6 M2]	"has_service C1 als M1"	System Logs, Application Logs
7.		C1	[15/Mar/2015 17:38:33 +0:00]	connect-unowned-vm	[C1 V6 V7 M2]	"authorized_cust C1 V7"	System Logs

Table 1: Log Data Generated in Case 2.1

Case 2.1	Case 2.2	Evidence Collection Change
allocate-vm	allocate-vm	
login-cust	login-cust	
	connect-to-vm-browser	Access to VMs performed by authorised customers through a web browser.
	make-user-vm-gekco	Creation of new users of the VM by Gecko CMS.
run-service-rvc	run-service-rvc	
make-admin-local	make-admin-local	
connect-over-network	connect-over-network	
get-service-als	get-service-als	
connect-unowned-vm	connect-unowned-vm	

Table 2: Changes in the evidence collection strategy when moving from Case 2.1 to Case 2.2.

Case No.	# Cloud Configuration Actions	# Monitored Actions	Reduction Percentage
1.1	11	4	64%
1.1, 1.2	15	8	47%
1.1, 1.2, 2.1	19	14	26%
1.1, 1.2, 2.1, 2.2	23	16	30%
1.1, 1.2, 2.1, 2.2, 2.3	30	19	37%

Table 3: Reduction in Data Collection.

# Clusters	# VMs	# Physical machines	# Networks	Time (sec)
5	50	15	9	0.10
10	100	30	19	0.87
15	150	45	29	3.70
20	200	60	39	10.13
25	250	75	49	24.55
30	300	90	59	46.79

Table 4: Evaluation of SGPlan5: Domain Expansion.

# Clusters Traversed	# Actions	Time (sec)
5	44	46.79
10	89	49.32
15	134	56.30
20	179	68.74
25	224	87.50
30	269	116.13

Table 5: Evaluation of SGPlan5: Attack Expansion.

Action ID	Log Type	Log Name
login_cust	VM System Log	auth.log (Frontend VM)
request_db	Cloud System Logs	activity_log.JSON (Cloud)
allocate_db	Cloud System Logs	activity_log.JSON (Cloud)
autherise_to_db	VM System Log	mongodb.log (Backend VM)
mask_username	VM System Log	syslog.log (Backend VM)
escalate_privilege	Application Log	mongodb.log (Backend VM)
login_db	Application Log	mongodb.log (Backend VM)
write_db	Application Log	mongodb.log (Backend VM)
get_service_cmdline	VM System Log	syslog.log (Backend VM)
copy_credentials	VM System Log	syslog.log (Backend VM)

Table 6: Mapping the steps of the attack scenario to the relevant logs files.

Case	# of Actions	Relative Overhead (KB)	Absolute Overhead (KB)
1	Setup	30	810
2	Setup + 10000	30	7770
3	Setup + 50000	30	35620
4	Setup + 100000	30	70640

Table 7: Storage overhead.

Case	Logs Extracted (KB)	Real Overhead (sec)
2	30	0.54
	577	0.57
3	30	1.89
	2970	1.99
4	30	2.95
	6042	3.28

Table 8: Time Overhead.