Journal of Information Processing Vol.23 No.4 392–401 (July 2015)

[DOI: 10.2197/ipsjjip.23.392]

Invited Paper

A Classification of Intrusion Detection Systems in the Cloud

Marwa Elsayed[1,a) Mohammad Zulkernine[1,b)

Received: December 3, 2014, Accepted: March 31, 2015

Abstract: Security is one of the most prominent challenges that hinder the acceleration of cloud adoption. Intrusion detection systems (IDSs) can be used to increase the security level of cloud environments. Therefore, the effectiveness of the IDS is a crucial issue for cloud security. However, the cloud presents new challenges and requirements, including scalability and adaptability, which effective IDSs need to address. Choosing the right deployment architecture significantly impacts the effectiveness of IDSs in the cloud. Additionally, robust IDSs need novel detection techniques to keep up with modern sophisticated attacks that target cloud environments. Hence, it is important to understand the advantages and limitations of different IDSs and how the deployment choice in cloud environments impacts the IDSs' effectiveness. This paper presents a novel classification scheme of the state-of-the-art of intrusion detection approaches in the cloud. This classification sheds light on the existing approaches with respect to the following aspects: deployment architecture and detection technique. We first classify the existing approaches based on their deployment architectures. Then, we present a comparative analysis of these approaches with respect to the detection techniques. We also provide detailed analysis of the strengths and weaknesses of existing approaches. The classification and analysis will help in the selection of the proper deployment architectures and detection techniques of IDSs in cloud environments.

Keywords: cloud security, intrusion detection

1. Introduction

Cloud computing represents a paradigm shift refactoring the IT landscape for delivering resources, systems and applications as services [1]. It promises impressive gains in delivering IT services such as the rapid elasticity, reliability, cost reduction, and quality of service improvement. These advantages are increasingly attracting governments, business organizations, and individuals to migrate their services and data into the cloud. The security risks and vulnerabilities leading to various attacks of diverse complexities and consequences are one of the prominent challenges of cloud adoption, as confirmed by various incident reports and research results [2], [3], [4]. Cloud computing environment with its distributed and open structure nature is rapidly gaining popularity, which makes it an attractive target for attackers to exploit vulnerabilities. Successful attacks can compromise the integrity, confidentiality, and availability of cloud data, software, and virtual assets.

An intrusion detection system (IDS) can offer additional security measures for the cloud environment. An IDS can monitor the cloud environment by investigating audit information about network traffic, application, software service, or virtual machine activities. It can also leverage the detection of malicious attempts, whether from external parties or legitimate users aiming at exploiting security vulnerabilities or violating security policies. An IDS can be applied across different layers of the cloud environment including the application, infrastructure, virtualization, and physical layers. In this sense, security monitoring and analysis via IDSs in the cloud layers is an effective way to increase consumers' trust by verifying the cloud security.

Employing an effective IDS in the cloud is a challenge from different aspects. One aspect is the complication of the security problem due to the cloud's deep stack of dependent layers. The functionality and security of a higher layer depend on its lower layers. This aspect is further augmented by the sophistication of modern attacks. Another aspect is the new requirements stemming from the unique characteristics of the cloud environment such as scalability and elasticity. These requirements pose additional challenges on the traditional IDSs in many ways. Hence, the development of robust cloud-oriented IDSs must identify and accommodate such unique cloud requirements. The last aspect is the deployment architecture selection as each choice has its own advantages and limitations with respect to the effectiveness of the IDS.

This paper proposes a classification of IDSs in the cloud. This classification sheds light on the current state-of-the-art of intrusion detection in the cloud. It also offers a comparative analysis to reason about developing trends in this research area as well as to characterize each approach with respect to the following aspects: deployment architecture and detection technique.

Several contributions can be identified in this paper: (i) it discusses the necessity of a classification based on the deployment architectures; (ii) it proposes a classification scheme of IDSs in

1 School of Computing, Queen's University, Kingston, ON, K7L 2N8 Canada
a) marwa@cs.queensu.ca
b) mzulker@cs.queensu.ca

the cloud; (iii) it presents the advantages and limitations of each deployment architecture; (iv) it proposes a comprehensive review along with a comparative analysis of the available detection techniques.

This paper is organized as follows: Section 2 presents related background and provides a general overview of the cloud computing paradigm as well as an anatomy of the cloud stack in depth. Then, it sheds some light on the cloud security issues in general, with more emphasis on the application, infrastructure, and virtualization layers which are considered as prime targets by attackers. Section 3 describes our proposed classification scheme of IDSs in the cloud based on their deployment architectures. Section 4 discusses the details of each category with respect to the detection techniques and provides a comparative analysis of the existing approaches. Finally, the paper outlines the open research issues along with concluding remarks.

2. Cloud Computing

The idea of cloud computing revolves from the NIST definition [5]: *"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction"*.

The cloud computing model has four deployment models [1]: Private cloud, a cloud platform dedicated for specific organization; Public cloud, a cloud platform available to public users to use the cloud infrastructure; Hybrid cloud, a private cloud that can be extended to use resources in public clouds; and Community cloud, a cloud platform that supports a community of several organizations having shared concerns where the cloud can be private or public, may be managed by these organizations or a third party, and may exist on premise or off premise.

2.1 Anatomy of Cloud Stack

As depicted in **Fig. 1**, the cloud stack is logically organized into five layers [6]. Each layer covers one or more cloud services. The cloud stack layers are as follows:

Application Layer: This layer provides software as a service (SaaS). It targets the end users who traditionally access the software services or applications provided by this layer through the internet. Cloud applications can be composed as a service from other services, using the concepts of service-oriented architecture (SOA). Google Apps [7] including Google Docs and Google Spreadsheets are examples of SaaS.

Software Environment Layer: This layer provides platform as a service (PaaS). It targets cloud app developers who implement their applications and deploy them on the cloud. It supplies APIs, platforms such as JVM and .NET, and integrated development environments (IDEs) such as Eclipse and Microsoft Visual Studio. This layer accelerates the development, deployment, and management of cloud applications without installing programming tools and platforms on the developers' local machines. Google's App Engine [8] is an example of PaaS.

Infrastructure Layer: This layer provides three distinct types

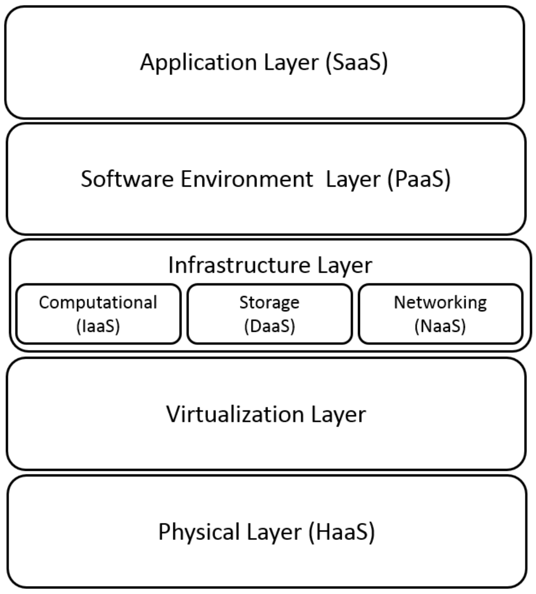

Fig. 1 The cloud stack.

of resources: computational, storage, and networking. These resources are delivered by cloud providers as internet-based services through virtualization technologies.

i) Computational resources: They offer infrastructure as a service (IaaS). They are provided to cloud users in the form of virtual machines (VMs). Amazon EC2 [9] is an example of IaaS.

ii) Storage resources: They offer data-storage as a service (DaaS). They are provided to the cloud users to store their data at remote disks and access them anytime from any place. Amazon's S3 [10] is an example of commercial cloud DaaS systems.

iii) Networking resources: They present the concept of network as a service (NaaS). Open VSwitch [11] is an example of NaaS.

Virtualization Layer: This layer represents the basic software for managing the physical hardware resources in the physical layer and the virtualized resources in the infrastructure layer. The major software foundation behind virtualization in the cloud is the Virtual Machine Monitor (VMM). The VMM facilitates the creation and management of VMs, each with separate operating systems (OSs) and applications. It manages the backend operation of these VMs by allocating the necessary resources such as CPU, memory, and storage. KVM [12] and Xen [13] are examples of the VMM.

Physical Layer: This layer is the backbone of the cloud including actual physical hardware resources and switches. In this regard, users of this layer are typically big enterprises with high IT requirements. They sublease hardware as a service (HaaS), where a HaaS provider operates, manages, and upgrades the hardware on behalf of its consumers for the duration of the sublease.

2.2 Security Issues in the Cloud

Each layer in the cloud stack has different implementations and different security requirements. This results in layer-specific or cross-layer vulnerabilities which complicate the development of a standard IDS model.

Cloud computing inherits most of the core technologies used in the Web and Internet, virtualization, and other foundation technologies. The integration of these technologies in cloud computing systems makes the cloud more vulnerable to security risks [14], [15]. Attackers can exploit novel attack venues as well as existing ones that are already associated with the use of these core technologies. Additionally, the complexity of the cloud that stems from the increased number of involved parties, devices, and applications often leads to an increase in the number of security holes. The cloud layers share three major security issues that hinder the acceleration of cloud services adoption as follows [16], [17], [18], [19], [20], [21]: loss of control, lack of isolation, and lack of regulation enforcement.

a) **Loss of control:** Data, applications, software services, and other assets in the cloud are typically hosted and maintained in a virtualized environment by third parties.

b) **Lack of isolation:** This is very critical due to the multi-tenancy characteristic of the cloud. Virtualization, which is the motor behind multi-tenancy in cloud layers, may suffer from vulnerabilities that can allow an attacker or an insider user to gain access to sensitive assets belonging to co-located tenants.

c) **Lack of regulation enforcement:** Cloud consumers may not be able to enforce regulations that govern the flow and storage of their information, or even verify if the provider complies with their security requirements.

The aforementioned issues are emerged due to wide attack vectors within or across the different layers of the cloud. The cloud application layer is a prime target by attackers due to the existence of various vulnerabilities. In a recent study, 96% of tested cloud applications have one or more serious security vulnerabilities [22]. Cloud-hosted application development relies mostly on existing web and internet technologies. Hence, most of the security problems related to web-enabled applications remain relevant when the same applications migrate to a cloud environment [23]. Vulnerabilities such as injection, cross site scripting, and information leakage are still prevalent in cloud applications as they account for more than 55% of the reported security flaws [22].

The cloud infrastructure and virtualization layers are also prime targets by attackers due to the existence of operational and configuration vulnerabilities [24], [25], [26], [27]. According to the NIST's bug report [28], the virtualization layer is considered a dangerous attack surface as a misconfigured VMM could result in a single point of compromise for the security of all hosted components. Attackers can focus their efforts on breaching vulnerabilities in the infrastructure layer (VMs) and then escape from one VM to another VM or to the virtualization layer (VMM). They may also directly exploit vulnerabilities in the virtualization layer in order to gain full control over the underlying physical machine on which the VMs are running. Virtualization vulnerabilities account for only a small subset of all vulnerabilities. However, it still represents a growing virtualization security concern especially in the cloud by creating a new attack surface against the VMM which can lead to various attacks such as cross-VM side channel, denial-of-service (DoS), malware, and rootkit attacks.

3. Classification of IDSs in the Cloud

This section discusses our proposed classification of the existing intrusion detection approaches in the cloud. This classification answers the following questions: (i) What is the chosen deployment architecture in each approach? (ii) What are the advantages and limitations of each deployment option? The classification scheme is based on the deployment architectures of these approaches as shown in **Fig. 2**.

The deployment architecture selection of an IDS in the cloud is crucial to its effectiveness. It affects two properties: visibility and robustness. Generally speaking, an IDS can be considered effective if: 1) it has a good visibility of the internal state of the monitored system; 2) it has a high robustness against attacks; 3) it can avoid any evasion attempts[*1]. To achieve these properties, conflicting requirements need to be resolved, which lead to a trade-off among the properties. In other words, an IDS needs relevant information from the monitored system to analyze in order to detect attacks effectively. The IDS can maintain a better visibility of the internal state of the monitored system if it is deployed to reside with the monitored system. As a consequence, better robustness against evasion is also achieved. However, this comes at the cost of weaker isolation from attacks which can disable the IDS itself or tamper with the collected information.

There exists a lack of classification or taxonomy of existing intrusion detection approaches in the cloud to understand their common characteristics and limitations. Patel et al. [29] propose a taxonomy of IDSs. This taxonomy classifies different aspects: response type, alarm management, detection method, data collection, audit-data source, time of detection, structure of the IDS, and technology layout. Modi et al. [30] address surveying the IDSs in the cloud with respect to their types and detection techniques. Therefore, we propose a comprehensive classification scheme based on the deployment architectures. This classification helps more in identifying how an architecture affects an IDS with respect to the visibility and robustness properties as well as the detection techniques. The place of deployment can be discussed from two perspectives: horizontal and vertical.

3.1 Horizontal Perspective

For horizontal placement, there are three choices to select from as follows:

Host-based IDS: It resides on the monitored host and performs intrusion detection by analyzing host-bound audit information about an operating system, applications, or users. It has a good visibility of the internal state of the monitored host.

Network-based IDS: It performs intrusion detection from outside the monitored host by analyzing network-bound audit information about the network traffic of the monitored host. It has a poor visibility of the internal state of the monitored host.

Hybrid IDS: It inherits the properties of the above mentioned

[*1] Evasion means that the IDS fails to recognize the attack. The attack succeeds to hide its malicious activity or the IDS lacks the ability to identify the attack. Attacking an IDS involves disabling the IDS itself or tampering with the collected information to prevent the IDS from detecting the malicious activity.

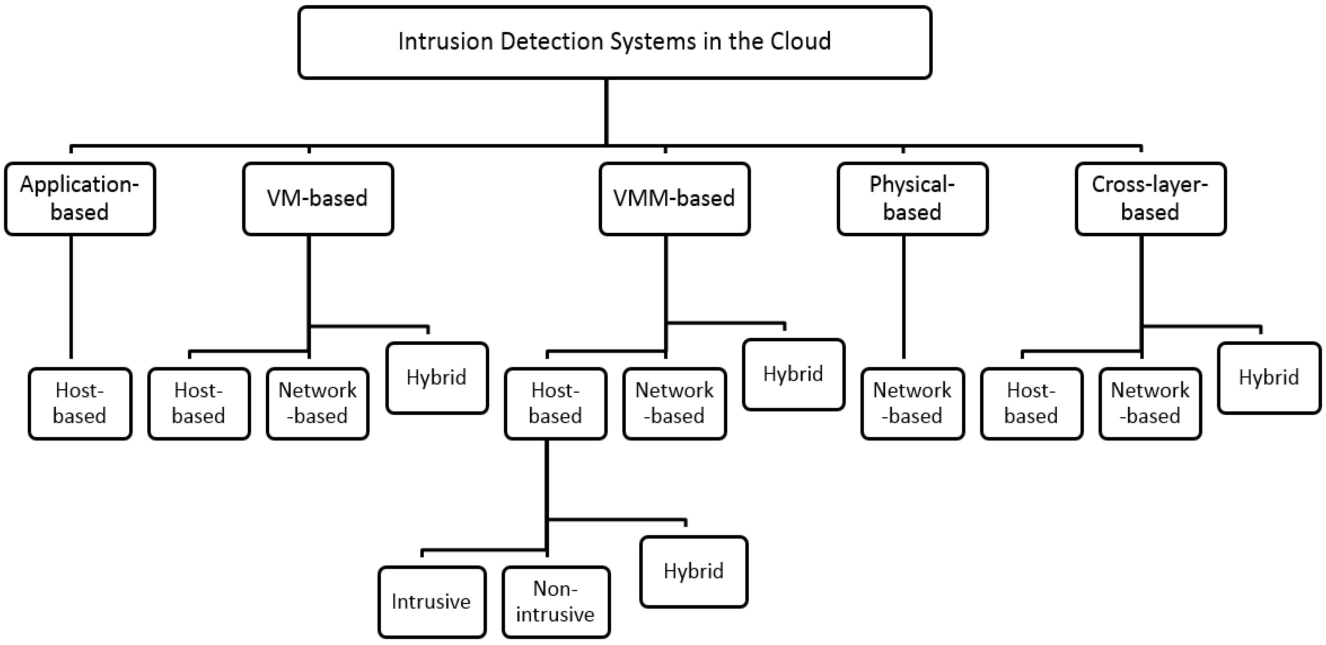

Fig. 2 Proposed classification of intrusion detection approaches in the cloud.

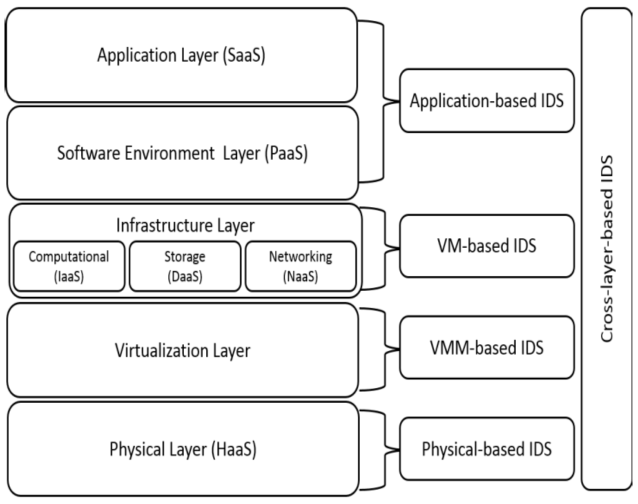

Fig. 3 Mapping between the deployment architectures and cloud stack layers.

types of IDSs.

3.2 Vertical Perspective

With respect to the vertical placement of the IDS, each deployment choice has its own advantages and disadvantages that reflect the trade-off between the visibility and robustness properties. **Figure 3** shows the mapping between the categories of IDSs from vertical perspective and the cloud stack layers. From this perspective, we classify IDSs in the cloud into five categories based on the deployment architectures as follows:

Application-based IDS: It monitors the events and activities occurring within a cloud application. It can detect attacks exploiting vulnerabilities or violating security requirements based on the information visible to the monitored application. It inherits the same properties of the host-based IDS. The good visibility of the internal state of the monitored application elevates its robustness against evasion attempts. However, this comes at the cost of weaker isolation between the IDS and attacks. Some application-based IDSs such as CloudFilter [31], RunTest [32], ROSIA [33], AdapTest [34], IntTest [35], CloudFence [36], ECloudFence [37], CDF [38] and VAM-aaS [42] are deployed in the application layer and offered as SaaS. Others including TMCA [39] and DEFCON [40] are deployed in the software environment layer and offered as PaaS.

VM-based IDS: It monitors the VMs based on the high-level information available from the perspective of the VM. It may inherit the same properties of a host-based, a network-based, or a hybrid IDS as clarified later in Section 4.2. VM-based IDSs are deployed in the infrastructure layer. IDSaaS [55], CFU [61], and CIDS [62] are examples of this type of IDSs.

VMM-based IDS: It allows monitoring the VMs managed by the VMM based on low-level information available from the perspective of the VMM. It can perform a host-based, a network-based, or a hybrid intrusion detection. This deployment architecture has a better visibility and higher robustness as it leverages the unique properties of the VMM as explained in details in Section 4.3.1. VMM-based IDSs (VMM-ID [43], SVMM [44], Cloudsec [45], IEH [46], VMFence [47], Maitland [48], NIDPS [49], MSVM [50], SnortFlow [52]) are deployed in the virtualization layer.

Physical-based IDS: It enables monitoring the physical network traffic to detect potential attacks. It inherits the same properties of a network-based IDS. This type of IDS may not be able to detect any attacks to a virtual network that runs entirely within the VMM. Physical-based IDSs are deployed in the physical layer. NIDS-CCE [56] is an example of this type of IDSs.

Cross-layer-based IDS: It spans multiple layers by combining different choices from the aforementioned IDSs. SilverLine [57] and mOSAIC-ID [58] are examples of this type of IDSs.

Further details of the proposed categories of IDSs along with a comparative analysis are presented in the next section.

4. Classification Details

Following the classification scheme presented in Section 3, we further study and discuss the existing approaches in terms of the applied detection techniques. Then, we provide a comparative analysis by detailing the common strengths and weaknesses of these approaches within the same category. Irrespective of the deployment architectures, an IDS can use an anomaly-based, a signature-based, or a hybrid detection technique. An anomaly-based detection technique compares the normal behavior of users, hosts, network connections, or applications against observed events in order to identify significant deviations or anomalies [64], [65]. A signature-based detection approach compares signatures that identify the patterns corresponding to known attacks or unauthorized behaviors against observed events in order to identify potential attacks or vulnerabilities. A hybrid detection combines anomaly-based and signature-based techniques. This section answers which detection technique is used in each proposed IDS and which attacks can the existing approaches detect?

4.1 Application-based IDSs

4.1.1 Existing Approaches

Most of the application-based intrusion detection approaches [31], [32], [33], [34], [35], [36], [37], [38], [39], [40] leverage the benefits of information flow control. Information flow control is a classical subject of computer security research which deals with restricting the flow of information between the objects manipulated by software applications [41]. There are two trends to incorporate information flow control with intrusion detection to monitor and attest information flow security. The first trend utilizes intrusion detection models based on the attestation of information flow security enforcements [31], [32], [33], [34], [35]. These enforcements can be specified through security policies that are permissible in a given application to achieve the integrity and confidentiality of data. The second trend implements intrusion detection models by leveraging information flow analysis techniques [36], [37], [38], [39], [40]. In what follows, we further discuss existing approaches with respect to these two trends.

Intrusion detection models [31], [32], [33], [34], [35] that leverage the attestation of information flow security enforcements can detect attacks violating the security policies. This is achieved by inspecting information flow through three phases. The first phase is defining security policies that enforce and guarantee the accepted flow of information. The second phase is monitoring the information flow to see if it conforms to the security policies. The final phase is raising an alarm when a policy violation is detected.

CloudFilter [31] detects attacks that violate the confidentiality of data-flow to prevent data leakage to unauthorized users or services. However, it may be defeated by disabling and evading its components by either a malicious insider or a malicious cloud provider. RunTest [32] and its evolutions ROSIA [33], Adaptest [34], and InTest [35] are proposed to be transparent solutions to increase their robustness in the sense that the attestation cannot be detected by attackers. These approaches detect and prevent attacks that violate the integrity of data-flow processing. They mainly rely on constructing and analyzing a data-flow integrity graph to reason about anomalous patterns as an indication of malicious services which may perform colluding attacks in a federated cloud application.

Intrusion detection approaches [36], [37], [38], [39], [40] that leverage information flow analysis techniques can detect attacks causing anomalous patterns of information flow without causing violations to security policies. Information flow analysis can be implemented in static, dynamic, or hybrid analysis techniques. Taint analysis is one of the information flow analysis techniques that aims at restricting how tainted data flows inside an application. Any taint analysis technique identifies the operations that can be affected by tainted data. It also imposes constraints over the data to be sanitized before reaching sensitive sinks. Taint analysis looks for misuses of tainted data by tracking it through the application in an attempt to detect and prevent attacks. Tracking of information flow in the cloud has two primary applications focusing on integrity and confidentiality properties of information.

CloudFence [36] and its enhanced design ECloudFence [37] use dynamic taint analysis to detect and prevent attacks that violate the confidentiality of data-flow in order to protect against data leakage and unauthorized data access. It is a transparent approach which increases robustness against attacks. However, it is still prone to be evaded by malwares as dynamic taint analysis techniques lack the ability to reason about non-explicit data flows which can occur via implicit assignments or control dependencies. Similarly, TMCA [39] extends the cloud computing platform of Google App Engine (GAE) for Python with dynamic taint analysis capabilities to detect injection and cross-site scripting (XSS) vulnerabilities. Although TMCA is transparent and isolated from attacks, it is still prone to evasion as it does not define a complete list of sensitive sinks in GAE. Specifically, it declares only the web server as a sensitive sink without considering other sensitive sinks in GAE. Accordingly, this does not close all the opening security holes that may exist in the cloud applications.

Alternatively, CDF [38] relies mainly on static data-flow analysis to detect vulnerabilities that can lead to data leakage and violation of data confidentiality. Although static analysis techniques have the ability to reason about all data assignments that take place on executed as well as unexecuted program branches, CDF has a low level of robustness as a malicious service provider can cheat and send source code which is totally different from its real implementation to a certified third party.

DEFCON [40] extends JAVA runtime environment (JDK) by combining static and dynamic techniques to leverage their advantages and minimize their limitations. It detects code vulnerabilities that can be exploited by attacks to violate the integrity and confidentiality of data in cloud applications. DEFCON is transparent which increases its robustness against attacks. It takes into consideration the analysis of data-flows through implicit assignments. However, it can still be evaded by attacks executing arbitrary code that induce non-explicit data flows through control dependencies. It has not been tested in cloud settings.

VAM-aaS [42] uses a different technique as it develops a static vulnerability analysis tool that uses vulnerability signatures to detect possible matches in the source code of a target system. Whenever a vulnerability is reported, mitigation actions specifying the security controls are applied to block the discovered vulnerability. It has a wide coverage of the top ten vulnerabilities specified by the OWASP (Open Web Application Security Project) [23]. However, it lacks the ability to detect zero-day or variants of known vulnerabilities that are beyond its signature database.

4.1.2 Comparative Analysis

The existing application-based intrusion detection approaches monitor and verify the security requirements of cloud consumers. This helps elevate the consumers' trust and confidence in cloud services and eliminate their fears from loss of control [31], [32], [33], [34], [35], [36], [37], [42] and lack of isolation [36], [37], [38], [39], [40]. In this sense, security monitoring and analysis of cloud applications via intrusion detection systems are effective to detect and prevent attacks violating the security requirements. This is specifically important to protect the integrity and confidentiality of information from either malicious or vulnerable cloud applications.

The majority of the application-based approaches implement anomaly-based intrusion detection techniques [31], [32], [33], [34], [35], [36], [37], [38], [39], [40]. The use of information flow control in these approaches aims at enriching the capability of detecting attacks that induce insecure information flow against sensitive assets.

Anomaly-based approaches [31], [32], [33], [34], [35] that leverage the attestation of information-flow security enforcements have the ability to detect attacks that violate the security policies. However, this trend has two shortcomings. First, the specified information flow policies may suffer from incorrectness or incompleteness which can negatively affect the effectiveness of this trend. Second, this trend lacks the ability to detect attacks that do not cause illegal information flow or violate security policies such as DoS attacks.

Alternatively, anomaly-based approaches [36], [37], [38], [39], [40] that leverage information-flow analysis techniques have the ability to detect attacks that may not cause illegal information flows. These approaches focus on detecting specific types of attacks that exploit improper input validation vulnerabilities namely SQL injection (SQLI) and cross-site scripting (XSS). The primary reason is that these approaches restrict information-flow analysis to dynamic data taint analysis which involves only the analysis of data dependencies but not control dependencies. Data taint analysis is very effective to detect the named attacks and vulnerabilities. Whereas, data flows alone are not sufficient to characterize the full range of attacks that can be revealed by intrusion detection systems. Also, dynamic taint analysis can reason about explicit data flows only. This opens doors for malware attacks that induce non-explicit flows. In general, dynamic taint analysis is precise and simple in contrast to static analysis but at the cost of high performance penalty. On the other hand, static analysis incurs no runtime overhead, augments the detection of security vulnerabilities during development rather than after production, and can reason about explicit and non-explicit flows. Whereas, it has limited applicability in case of applications where data flow requirements depend on runtime information and may produce false alarms.

A signature-based approach [42] uses a vulnerability-based technique. In general, the vulnerability-based signature trend outperforms attack-based signature trend. This is due to three main properties: broader coverage of threat detection, smaller size of signature database, and better detection of known vulnerabilities. However, the vulnerability-based signature trend still lacks the ability to detect unknown vulnerabilities that may be exploited by attackers.

4.2 VM-based IDSs

A VM-based IDS can perform a host-based, a network-based, or a hybrid detection similar to a VMM-based IDS. However, the latter has more advanced properties as detailed later in Section 4.3.1. Hence, we briefly outline the VM-based approaches as follows.

Host-based IDSs

A VM-based IDS with a host-based merit allows for monitoring the activity of the operating system and applications running on the VM and analyzing the information which is visible to the VM about them. It has a good visibility of the internal states of the corresponding VM. However, it is not isolated from the VM and as a result it can be exposed to attacks that can modify the monitor agent or subvert the acquired information. CFU [61] is an example of this type of IDSs.

Network-based IDSs

A VM-based IDS with a network-based merit is capable of monitoring the internal network traffic of the VM. Such an IDS has a poor visibility of the internal state of the system making it prone to evasion but it is isolated from attack manipulations targeting the corresponding VM. IDSaaS [55] is an example of this type of IDSs.

Hybrid IDSs

A VM-based IDS with a hybrid merit combines the features of both host-based and network-based IDSs. CIDS [62] is an example of this type of IDSs.

4.3 VMM-based IDSs

4.3.1 Existing Approaches

Host-based IDSs

A VMM-based IDS that has the merit of a host-based IDS [43], [44], [45], [46], [48] can capture the dynamic updates of the OSs and applications running on the monitored VMs. It has three main features leveraged from the VMM [67]: isolation ensures that the IDS cannot be exposed to attacks; introspection enables the IDS to be immune from any evasion attempts; and interposition enables the IDS to reason about certain VMs' processes executing privileged instructions.

Hence, a VMM-based IDS enjoys a rich view of the target systems (VMs) combined with a high robustness against attacks and evasion attempts. However, it faces a semantic gap problem [68]. This problem happens because the IDSs are deployed in the VMM where only low-level hardware information is provided about the guest VMs such as CPU register values and memory content. However, VMM-based IDSs need high-level semantic information from the OS-level such as the executed system calls and active processes. To tackle this semantic gap problem, there are three different deployment architectures of VMM-based IDSs as discussed in the following paragraphs:

Intrusive. These IDSs have monitoring components deployed inside the guest VMs in order to directly access rich high-level information from the OS-level about the monitored systems. This deployment architecture is a straightforward approach to close the semantic gap problem. This deployment option provides access to a large amount of rich OS-level information enabling efficient intrusion detection. However, it has the potential to expose the monitoring agents to malicious users or attackers of guest VMs. We are not aware of any intrusive approach proposed for the cloud context. VMM-Honey [59] and Lares [60] are some examples of intrusive approaches proposed for non-cloud virtualized environments.

Non-intrusive. These IDSs [43], [44], [45], [48], [53] use virtual machine introspection (VMI) techniques in order to obtain low-level information about the hosted VMs without using monitoring components inside the VMs. To tackle the semantic gap problem, some approaches [45] construct high-level information from the introspected low-level information relying on a prior OS-context knowledge. Cloudsec [45] utilizes VMI techniques in anomaly-based detection of memory-based rootkit attacks. The high-level data structures are examined for hidden processes to detect rootkits. However, Cloudsec is still prone to evasion through rootkit attacks attaching to processes which are not hidden. Alternatively, some other approaches [43], [44], [48], [53] rely only on the inspected low-level information. Maitland [48] utilizes VMI techniques in signature-based detection of malwares. The content of memory pages containing executable code is compared against known signatures of binary executables. Similarly, HCIDS [53] utilizes VMI techniques in signature-based detection of DoS attacks. VMM-ID [43] and SVMM [44] apply anomaly-based detection of malware attacks. They employ data mining and machine learning techniques to transform the inspected low-level data into useful information characterizing the normal execution behavior of the processes running on the monitored VMs. Both of them [43], [44] are robust as they take into consideration malwares that may be attached not only to hidden processes, but also to processes which are not hidden.

Both intrusive and non-intrusive (hybrid). These IDSs [46] employ both intrusive and non-intrusive components. They compare two types of high-level information: extracted and reconstructed. The extracted information is directly obtained by the intrusive components deployed in the monitored VMs. Non-intrusive components, deployed in the VMM, reconstruct the high-level information from the introspected low-level information. IEH [46] follows this deployment architecture. It applies anomaly-based detection based on any mismatch between these two types of information. IEH can detect rootkits which cause anomalous links with processes, files, or network connections.

Network-based IDSs

A VMM-based IDS that has a network-based merit [47], [49] leverages the VMI techniques. These techniques provide the capability of analyzing communications in the virtual network between VMs controlled by the VMM, and between VMM and VMs. VMFence [47] applies signature-based detection of network-based attacks relying on SNORT [54]. It may be evaded by unknown or variants of known attacks whose signatures the IDS is not aware of. NIDPS [49] combines signature-based and anomaly-based techniques to detect known as well as unknown network-based attacks.

Hybrid IDSs

A hybrid VMM-based IDS combines host-based and network-based merits. MSVM [50] and SnortFlow [52] combine anomaly-based and signature-based techniques to detect malware as well as network attacks. MSVM is based on the authors' previous tool called NICE [51] which is responsible to create an attack graph analytical model. MSVM inspects the network packets of each VM relying on SNORT. Whenever an attack is detected, the monitored VM is deeply inspected to detect any host-based intrusion. The host-based components are deployed following a hybrid approach that combines intrusive and non-intrusive components. Any information mismatch is detected to identify the hidden malicious processes running on the corresponding VM. At this point, MSVM selects an optimal countermeasure based on the attack graph model to reconfigure the virtual network and mitigate the attack. Rootkits attaching to unhidden processes can still evade MSVM.

4.3.2 Comparative Analysis

The VMM-based approaches offer a unique capability to inspect the access interactions between software running on the VMs and the underlying physical hardware. This capability facilitates the elimination of the fears from lack of isolation and loss of control. Additionally, they are able to monitor the security for all the components in the virtualized cloud environment.

The existing VMM-based approaches employ various types of intrusion detection techniques. Some approaches [43], [44], [45], [46] use anomaly-based techniques, some other approaches [47], [48], [53] use signature-based techniques, and others [49], [50], [52] are hybrid combining both techniques.

The majority of the approaches that perform host-based intrusion detection apply anomaly-based techniques [43], [44], [45], [46] except Maitland [48] and HCIDS [53]. In general, the creation of benign profiles for each monitored VM running under the control of the VMM is a challenging task. The main reason is that these profiles are heterogeneous and should be adaptable to accurately represent the dynamic changes in the normal behavior of the monitored VMs.

Maitland [48] and HCIDS [53] are host-based IDSs that apply signature-based intrusion detection techniques. This limits their detection ability to only the known attacks stored in their databases. In general, applying signature-based techniques to perform host-based intrusion detection in the cloud can suffer from the following challenges:

- The creation of a database of attack signatures for all operating systems and applications hosted on top of the VMM is challenging in terms of complexity and size of the signature database.
- The adaptation of this signature database is not an easy task

as dynamic changes occur during the migration of VMs between physical hosts.

VMM-based approaches leverage the VMI technique where some of them [43], [44], [45], [48], [53] choose to follow a non-intrusive deployment architecture. Others [46], [50], [52] combine intrusive and non-intrusive components. Generally, incorporating VMI techniques provides VMM-based IDSs with many benefits that elevate their effectiveness in terms of visibility and robustness.

4.4 Physical-based IDS

As there is no difference in physical-based IDSs deployed in a cloud or a non-cloud environment, we omit the discussion on physical-based IDSs in this paper. NIDS-CCE [56] is an example of this architecture. It employs SNORT to perform signature-based detection of network-based attacks, specifically flooding DoS attacks.

4.5 Cross-layer-based IDS

This type of IDSs spans multiple layers by combining different choices from the above mentioned IDSs. It can be deployed at different architectural layers in the cloud. It can have the merit of a host-based, a network-based, or a hybrid IDS. There are limited approaches that follow this deployment option [57], [58].

Host-based IDSs

SilverLine [57] combines VM-based and VMM-based deployment architectures. It applies anomaly-based detection relying on the concept of information flow control. SilverLine detects and prevents data leaks that result from compromise, misconfiguration, or cross-VM side channel attacks from co-resident cloud tenants. It ensures that data from one tenant is not propagated to untrusted services belonging to other tenants, or to unauthorized locations outside the specified network.

Hybrid IDSs

mOSAIC-ID [58] combines application-based, VMM-based, VM-based, and physical-based components. This approach aims at detecting different attacks such as SQLI, XSS, and DoS. It collects information from different architectural layers to monitor both the cloud resources and software components in federated clouds [66]. After that, complex event correlation and semantic analysis are performed to detect attacks. This approach can be robust against evasion as it collects information from different architectural levels. Conversely, its application-based and VM-based components can still be prone to attacks directed to the monitored application or the hosted VM.

5. Conclusion

The promise of the cloud cannot be fulfilled until cloud consumers attain more confidence in the security of the cloud environment. In this regard, intrusion detection can monitor and verify the security requirements of the consumers in order to increase the security level of the cloud environment. This is important to elevate the trust of the consumers in some critical domains, such as health and banking, to accelerate their process of adopting cloud technologies.

Throughout this paper, we explore intrusion detection in cloud computing based on the proposed classification scheme. We classify the existing IDSs with respect to their deployment architectures. For each IDS class, we compare and contrast the applied detection techniques. This study provides a guideline to consider the effect of the deployment architectures and detection techniques on the IDSs in the cloud. The classification and analysis help not only in choosing an appropriate existing IDS but also in evaluating any new IDS.

We observe that the research effort for intrusion detection solutions to address the security issues in the cloud is still inadequate. Further research effort is required to consider the utilization of application-based models in detecting other types of attacks. A deeper investigation is also required to gain the maximum benefits from deploying IDSs in the VMM. A VMM-based IDS can have distinctive features of full visibility and high robustness to perform a host-based, a network-based, or a hybrid intrusion detection. Cross-layer-based approaches can be explored more to offer an integrated solution that secures all the layers throughout the cloud stack as a whole.

This work focuses mainly on exploring intrusion detection with respect to two aspects: deployment architecture and detection technique. Exploring different aspects of IDSs such as the response methods taken by an IDS to prevent the detected attacks is left for future work. We plan to study how the response methods affect the effectiveness of the IDS. We also agree with the initiative of creating representative evaluation benchmarks for experimenting IDSs in real cloud environments [63].

Acknowledgments This research is partially supported by the Natural Sciences & Engineering Research Council of Canada (NSERC). The first author thanks the Schlumberger Foundation for supporting her Ph.D. study in Canada. The authors would also like to thank Dr. Komminist Weldemariam and Dr. Khalid Elgazzar for their comments on this work.

References

[1] Furht, B. and Escalante, A.: Chapter 1: Cloud Computing Fundamentals, *Handbook of Cloud Computing*, Springer (2010).

[2] Archer, J., Boehme, A., Cullinane, D., Kurtz, P., Puhlmann, N. and Reavis, J.: Top Threats to Cloud Computing V 1.0, Cloud Security Alliance (2010).

[3] Takabi, H., Joshi, J.B.D. and Ahn, G.: Security and Privacy Challenges in Cloud Computing Environments, *Proc. IEEE Security & Privacy*, Vol.8, No.6, pp.24–31 (2010).

[4] Cloud Security Alliance: Cloud Computing Vulnerability Incidents: A Statistical Overview (2013), available from ⟨https://cloudsecurityalliance.org/download/cloud-computing-vulnerability-incidents-a-statistical-overview/⟩ (accessed 2014-06).

[5] Mell, P. and Grance, T.: *The NIST Definition of Cloud Computing*, National Institute of Standards and Technology (2011).

[6] Ahson, S.A. and Ilyas, M.: Chapter 1: Understanding the Cloud Computing Landscape, *Cloud Computing and Software Services: Theory and Techniques*, Auerbach Publications (2011).

[7] Google Apps, available from ⟨http://www.google.com/apps/business/index.html⟩.

[8] Google App Engine, available from ⟨https://developers.google.com/appengine/?csw=1⟩.

[9] Amazon Elastic Compute Cloud (EC2), available from ⟨http://aws.amazon.com/ec2/⟩.

[10] Amazon Simple Storage Service (S3), available from ⟨http://aws.amazon.com/s3/⟩.

[11] Open VSwitch, available from ⟨http://openvswitch.org/⟩.

[12] Kernal-based Virtual Machine (KVM), available from ⟨http://www.linux-kvm.org/page/Main_Page⟩.

[13] Xen, available from ⟨http://www.xenproject.org/developers/teams/

hypervisor.html⟩.
[14] Grobauer, B., Walloschek, T. and Stocker, E.: Understanding Cloud Computing Vulnerabilities, *Proc. IEEE Security & Privacy*, Vol.9, No.2, pp.50–57 (2011).
[15] Xiao, Z. and Xiao, Y.: Security and Privacy in Cloud Computing, *Proc. IEEE Communications Surveys & Tutorials*, Vol.15, No.2, pp.843–859 (2013).
[16] Almorsy, M., Grundy, J. and Müller, I.: An analysis of the cloud computing security problem, *Proc. 2010 Asia Pacific Cloud Workshop*, Australia (2010).
[17] Subashini, S. and Kavitha, V.: A survey on security issues in service delivery models of cloud computing, *Proc. International Journal of Network and Computer Applications*, Vol.34, No.1, pp.1–11 (2011).
[18] Zissis, D. and Lekkas, D.: Addressing cloud computing security issues, *Proc. International Journal of Future Generation Computer Systems*, Vol.28, No.3, pp.583–592 (2012).
[19] Rong, C., Nguyen, S.T. and Jaatun, M.G.: Beyond lightning: A survey on security challenges in cloud computing, *Proc. International Journal of Computers & Electrical Engineering*, Vol.39, No.1, pp.47–54 (2013).
[20] Modi, C., Patel, D., Borisaniya, B., Patel, A. and Rajarajan, M.: A survey on security issues and solutions at different layers of cloud computing, *Proc. Journal of Supercomputing*, Vol.63, No.2, pp.561–592 (2013).
[21] Fernandes, D., Soares L., Gomes, J., Freire, M. and Inácio, P.: Security issues in cloud environments: A survey, *Proc. International Journal of Information Security*, Vol.13, No.2, pp.113–170 (2014).
[22] CENZIC. Cloud Applications Vulnerability Trends Reports (2014), available from ⟨http://www.cenzic.com/downloads/Cenzic_Vulnerability_Report_2014.pdf⟩ (accessed 2005-02).
[23] OWASP: The Ten Most Critical Web Application Security Vulnerabilities (2013).
[24] Vollmar, W., Harris, T., Long, L. and Green, R.: Hypervisor Security in Cloud Computing Systems, *Proc. ACM Computer Survey* (2014).
[25] Perez-Botero, D., Szefer, J. and Lee, R.: Characterizing Hypervisor Vulnerabilities in Cloud Computing Servers, *Proc. 2013 international Workshop on Security in Cloud Computing*, pp.3–10 (2013).
[26] Szefer, J., Keller, E., Lee, R.B. and Rexford, J.: Eliminating the hypervisor attack surface for a more secure cloud, *Proc. 18th ACM Conference on Computer and Communications Security*, pp.401–412, ACM (2011).
[27] Scarfone, K., Souppaya, M. and Hoffman, P.: NIST guide to security for full virtualization technologies (2011).
[28] National Vulnerability Database, CVE and CCE Statistics Query Page, available from ⟨http://web.nvd.nist.gov/view/vuln/statistics⟩
[29] Patel, A., Taghavi, M., Bakhtiyari, K. and JúNior, J.: Taxonomy and proposed architecture of intrusion detection and prevention systems for cloud computing, *Proc. 4th International Conference on Cyberspace Safety and Security*, pp.441–458, Springer Berlin Heidelberg (2012).
[30] Modi, C., Patel, D., Borisaniya, B., Patel, H., Patel, A. and Rajarajan, M.: A survey of intrusion detection techniques in cloud, *Proc. International Journal of Network and Computer Applications*, Vol.36, No.1, pp.42–57 (2012).
[31] Papagiannis, L. and Pietzuch, P.: CloudFilter: practical control of sensitive data propagation to the cloud, *Proc. 2012 ACM Workshop on Cloud Computing Security Workshop*, pp.97–102, ACM (2012).
[32] Du, J., Wei, W., Gu, X. and Yu, T.: Runtest: Assuring integrity of dataflow processing in cloud computing infrastructures, *Proc. 5th ACM Symposium on Information, Computer and Communications Security*, pp.293–304, ACM (2010).
[33] Du, J., Gu, X. and Yu, T.: On verifying stateful dataflow processing services in large-scale cloud systems, *Proc. 17th ACM Conference on Computer and Communication Security* (*CCS*), pp.672–674, ACM (2010).
[34] Du, J., Shah, N. and Gu, X.: Adaptive data-driven service integrity attestation for multi-tenant cloud systems, *Proc. 2011 IEEE 19th International Workshop on Quality of Service* (*IWQoS*), pp.1–9 (2011).
[35] Du, J. and Dean, D.: Scalable Distributed Service Integrity Attestation for Software-as-a-Service Clouds, *Proc. IEEE Trans. Parallel and Distributed Systems*, No.99, pp.1–11 (2013).
[36] Pappas, V., Kemerlis, V., Zavou, A., Polychronakis, M. and Keromytis, A.: CloudFence: Enabling Users to Audit the Use of their Cloud-Resident Data, Technical Report CUCS-002-12, Columbia University (2012).
[37] Pappas, V., Kemerlis, V., Zavou, A., Polychronakis, M. and Keromytis, A.: CloudFence: Data Flow Tracking as a Cloud Service, *Research in Attacks, Attacks, and Defenses*, Lecture Notes in Computer Science, RAID, Vol.8145, pp.411–431, Springer (2013).
[38] Weißbach, M. and Zimmermann, W.: Controlling Data-Flow in the Cloud, *Proc. CLOUD COMPUTING 2012: The 3rd International Conference on Cloud Computing*, GRIDs, and Virtualization, pp.24–29 (2012).
[39] Bello, L. and Russo, A.: Towards a taint mode for cloud computing web applications, *Proc. 7th Workshop on Programming Languages and Analysis for Security*, p.7, ACM (2012).
[40] Migliavacca, M., Papagiannis, I., Eyers, D., Shand, B., Bacon, J. and Pietzuch, P.: DEFCON: high-performance event processing with information security, *Proc. USENIX Annual Technical Conference* (2010).
[41] Denning, D.E. and Denning, P.J.: Certification of programs for secure information flow, *Proc. Comm. ACM*, Vol.20, No.7, pp.504–513 (1977).
[42] Almorsy, M., Grundy, J. and Ibrahim, A.: VAM-aaS: Online cloud services security vulnerability analysis and mitigation-as-a-service, *Proc. 2012 International Conference on Web Information Systems Engineering* (*WISE*), pp.411–425, Springer Berlin Heidelberg (2012).
[43] Azmandian, F., Moffie, M., Alshawabkeh, M., Dy, J., Aslam, J. and Kaeli, D.: Virtual machine monitor-based lightweight intrusion detection, *Proc. ACM SIGOPS Operating Systems Review*, Vol.45, No.2, pp.38–53 (2011).
[44] Azmandian, F., Kaeli, D., Dy, J., Aslam, J. and Schaa, D.: Securing cloud storage systems through a virtual machine monitor, *Proc. 1st International Workshop on Secure and Resilient Architectures and Systems*, pp.19–24, ACM (2012).
[45] Ibrahim, A., Hamlyn-Harris, J., Grundy, J. and Almorsy, M.: Cloudsec: a security monitoring appliance for virtual machines in the IaaS cloud model, *Proc. 2011 5th International Conference on Network and System Security* (*NSS*), pp.113–120, IEEE (2011).
[46] Xie, X. and Wang, W.: Rootkit detection on virtual machines through deep information extraction at hypervisor-level, *Proc. IEEE Conference on Communications and Network Security* (*CNS*), pp.498–503, IEEE (2013).
[47] Jin, H., Xiang, G., Zou, D., Wu, S., Zhao, F., Li, M. and Zheng, W.: A VMM-based intrusion prevention system in cloud computing environment, *Proc. Journal of Supercomputing*, Vol.66, No.3, pp.1133–1151 (2013).
[48] Benninger, C., Neville, S., Yazir, Y., Matthews, C. and Coady, Y.: Maitland: Lighter-weight vm introspection to support cyber-security in the cloud, *Proc. 2012 IEEE 5th International Conference on Cloud Computing* (*CLOUD*), pp.471–478, IEEE (2012).
[49] Gupta, S., Kumar, P. and Abraham, A.: A Profile Based Network Intrusion Detection and Prevention System for Securing Cloud Environment, *Proc. International Journal of Distributed Sensor Networks* (2013).
[50] Chung, C., Cui, J., Khatkar, P. and Huang, D.: Non-intrusive process-based monitoring system to mitigate and prevent VM vulnerability explorations, *Proc. 9th International Conference on Collaborative Computing: Networking, Applications and Worksharing* (*Collaboratecom*), pp.21–30, IEEE (2013).
[51] Chung, C., Khatkar, P., Xing, T., Lee, J. and Huang, D.: NICE: Network intrusion detection and countermeasure selection in virtual network systems, *Proc. IEEE Trans. Dependable and Secure Computing*, Vol.10, No.4, pp.198–211 (2013).
[52] Xing, T., Huang, D., Xu, L., Chung, C. and Khatkar, P.: Snortflow: A openflow-based intrusion prevention system in cloud environment, *Proc. Research and Educational Experiment Workshop* (*GREE*), pp.89–92, IEEE (2013).
[53] Nikolai, J. and Wang, Y.: Hypervisor-based cloud intrusion detection system, *Proc. 2014 International Conference on Computing, Networking and Communications* (*ICNC*), pp.989–993, IEEE (2014).
[54] Martin, R.: Snort – Light Weight Intrusion Detection for Networks, available from ⟨http://www.snort.org⟩.
[55] Alharkan, T. and Martin, P.: IDSaaS: Intrusion detection system as a service in public clouds, *Proc. 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing* (*ccgrid*), pp.686–687 (2012).
[56] Mazzariello, C., Bifulco, R. and Canonico, R.: Integrating a network IDS into an open source cloud computing environment, *Proc. 2010 6th International Conference on Information Assurance and Security* (*IAS*), pp.265–270, IEEE (2010).
[57] Mundada, Y., Ramachandran, A. and Feamster, N.: SilverLine: Data and network isolation for cloud services, *Proc. HotCloud* (2011).
[58] Ficco, M., Tasquier, L. and Aversa, R.: Intrusion Detection in Cloud Computing, *Proc. IEEE 8th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing* (*3PGCIC*), pp.276–283, IEEE (2013).
[59] Asrigo, K., Litty, L. and Lie, D.: Using VMM-based sensors to monitor honeypots, *Proc. 2nd International Conference on Virtual Execution Environments*, *VEE '06*, pp.13–23, ACM (2006).
[60] Payne, B.D., Carbone, M., Sharif, M. and Lee, W.: Lares: An Architecture for Secure Active Monitoring Using Virtualization, *Proc. IEEE*

Symposium on Security and Privacy, pp.233–247 (2008).
[61] Bakshi, A. and Yogesh, B.: Securing Cloud from DDoS Attacks using Intrusion Detection System in Virtual Machine, *Proc. 2nd International Conference on Communication Software and Networks, ICCSN'10*, pp.260–264, IEEE (2010).
[62] Kholidy, H. and Baiardi, F.: CIDS: A Framework for Intrusion Detection in Cloud Systems, *Proc. 2012 9th International Conference on Information Technology: New Generations* (*ITNG*), IEEE (2012).
[63] Milenkoski, A. and Kounev, S.: Towards benchmarking intrusion detection systems for virtualized cloud environments, *Proc. International Conference for Internet Technology And Secured Transactions*, pp.562–563, IEEE (2012).
[64] Scarfone, K. and Mell, P.: *Guide to Intrusion Detection and Prevention Systems* (*IDPS*) (Draft), NIST Special Publication 800-94 (2012).
[65] Endorf, C., Schultz, G. and Mellander, J.: *Intrusion Detection & Prevention*, 1st edition, McGraw-Hill Osborne Media (2003).
[66] mOSAIC Project – Open source API and platform for multiple clouds (2010), available from ⟨http://www.mosaic-cloud.eu⟩
[67] Garfinkel, T. and Rosenblum, M.: A virtual machine introspection based architecture for intrusion detection, *Proc. Network and Distributed System Security Symposium* (*NDSS*), pp.191–206 (2003).
[68] Chen, P. and Noble, B.: When virtual is better than real [operating system relocation to virtual machines], *Proc. 8th Workshop on Hot Topics in Operating Systems*, pp.133–138, IEEE (2001).

Marwa Elsayed is a Ph.D. Candidate and a Research Assistant in the School of Computing, Queen's University, Canada, where she is a member of the Queen's Reliable Software Technology (QRST) research group. Ms. Marwa Elsayed received her B.Sc. and M.Sc. degrees in Computer Science from the School of Computer and Information Sciences, Ain Shams University, Cairo, Egypt. Her current research interests include software security, cloud computing. Further information about her research and publications can be found at http://research.cs.queensu.ca/˜marwa/.

Mohammad Zulkernine is a Canada Research Chair in Software Dependability and an Associate Professor at the School of Computing, Queen's University, Canada. He leads the Queen's Reliable Software Technology (QRST) research group. His current research projects focus on software reliability and security that are sponsored by a number of provincial and federal research funding agencies and industry. Dr. Zulkernine was one of the program co-chairs of SSIRI 11, COMPSAC 12, and HASE 14. He is a senior member of the IEEE and the ACM, and a licensed professional engineer in the province of Ontario, Canada. Dr. Zulkernine received his Ph.D. from the University of Waterloo, Canada. More information about his research and publications can be found at http://research.cs.queensu.ca/˜mzulker/.