2020-10

Digital forensics cloud log unification: Implementing CADF in Apache CloudStack

Dalezios, N

10.1016/j.jisa.2020.102555
Journal of Information Security and Applications
Elsevier BV

Journal of Information Security and Applications 54 (2020) 102555

Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa

Digital forensics cloud log unification: Implementing CADF in Apache CloudStack

Nikolaos Dalezios, Stavros Shiaeles*, Nicholas Kolokotronis, Bogdan Ghita

School of Computing, Faculty of Technology, University of Portsmouth, Buckingham Building, Portsmouth, Hampshire, PO1 3HF, United Kingdom

ARTICLE INFO

Keywords:
Cloud computing
Computer crime
Forensics
Cloud Auditing Data Federation
CADF
CloudStack

ABSTRACT

Cloud computing is an important step in our era, delivering many advantages in business and our daily life. However, as every new technology, various challenges are brought into light with one of them being the misuse of Cloud computing environments for criminal activities. As such, Cloud service providers have to establish adequate forensic capabilities in order to support forensics investigations in the event of illegal activities in the cloud. In order to help forensics investigations, this paper deals with log format unification in cloud platforms using Distributed Management Task Force's (DMTF) Cloud Auditing Data Federation (CADF) standard. CADF event logging is utilised in the widely used OpenStack, and we have modified the Apache CloudStack platform to become forensically sound. Furthermore, we investigated the existing CloudStack platform along with the proposed CADF event model implemented, with regards to the principles of the Association of Chief Police Officers (ACPO) on handling digital evidence. The results are provided in this paper as well as an automated parsing tool/CADF event consumer, named C.Lo.D, which is freely available and can be downloaded from Github.

1. Introduction

THE uprising needs to reduce service costs, the integration of services between non-portable and portable devices and the ever-increasing demands on storage and computing power have led companies to cloud computing systems. The volume of the processing procedure is transferred to the cloud, making it imperative to define processes of forensic analysis but also to solve the open challenges/limitations of forensics in this new environment. As Dykstra [1] noted, crime heads to where people, data and money are gathered.

In the era of Big Data, Virtualization and Cloud Computing, the volume of information stored are disproportionately large concerning the speed of transfer. The process of acquisition during the investigation for digital evidence is the most decisive for its course since it will provide the investigator with the subject on which he/she will perform the forensic analysis. Limitations set by the volume of data are time, cost, and validity.

Cybercrime and Internet cannot be seen in isolation from one another. More specifically, the Internet, depending on how it is used by malicious users, divides cybercrime into 1) cyber-assisted, 2) cyber-enabled, 3) cyber-dependent.

Cybercriminals enjoy the same benefits of cloud usage with businesses. Increased computing power, increased capacity, energy and cost savings, anonymity and elasticity are just some of the features that malicious users can take advantage of, thus expanding their capabilities and goals. The creation and use of botnets, data-mining, crypto-mining and Command-n-Control (CnC) centres are now easier than ever. However, apart from the tools of criminals, the range of targets is also significantly increased. Once a customer decides to use the cloud services, he/she has to accept that his/her data are no longer available only for him/her or in a particular location. When the Internet of Things (IoT)/Internet of Anything (IoA) factor is added to the above equation, where each device can connect and communicate with a network – often using the cloud infrastructure – then, the targets (in number) and the complexity of attacks skyrocket [2].

A significant motivation for cloud usage in cybercrime is also the lack of tools and procedures of forensic analysis combined with open issues [3,4]. Many times, criminals do not even need anti-forensics, as long as they use any infrastructure outside the judicial and administrative jurisdiction of the authorities as a basis for any malicious energy.

A system's log files are a critical component for debugging and monitoring its operation and current status. During a forensic in-

* Corresponding author.
E-mail address: stavros.shiaeles@port.ac.uk (S. Shiaeles).

https://doi.org/10.1016/j.jisa.2020.102555
2214-2126/

vestigation, these files may contain important information related to the investigated incident. They are one of the key "witnesses" of what was happening to a system at a specific time.

It is generally accepted that the process of e-discovery, collection and analysis of digital evidence in cloud environments differs from that in-home/corporate computing systems as it is a cross-discipline of cloud computing and digital forensics [5]. The difference lies in the absence of tools, procedures but mainly in the case of the possible geographical dispersion of the system under investigation. Non-physical access to a system requires specific, technical and legal, conditions of remote access to be provided. At this point, questions are raised about ensuring the integrity of data.

In cloud environments where the amount of information of log files is enormous, useful information may not be easily located. Moreover, cloud-related issues such as fragmentation, geographical dispersion and different implementations make it difficult to identify useful information.

This paper investigates how the Cloud Auditing Data Federation (CADF) event model could be implemented in Apache CloudStack, whether the existing model is improved, and what test case scenarios could be considered.

The rest of this paper is structured as follows: Section II provides a literature review of recent studies on auditing and cloud forensics, Section III studies CloudStack's current event model and Section IV proposes DMTF's CADF event model. Section V presents the implementation, testing and validation of CADF for CloudStack. In Section VI discussion raises issues and suggestions regarding implementation and Section VII is examining CADF's compliance to ACPO principles. Section VIII presents a CADF log parsing tool. Finally, in Section IX, conclusions of this paper are presented, and in Section X, future directions are outlined.

2. Background

2.1. Related work

The issue of cloud log format unification was identified by Simou as part of the examination stage [6] while the first official and comprehensive attempt to collect all open issues in cloud forensics was held on behalf of NIST in 2014 [3]. A new study, in 2015, highlighted the reliance on the availability and type of log files on the Cloud service provider [7]. In particular, in Software as a Service (SaaS) and Platform as a Service (PaaS) models, it is the sole responsibility of the provider to make log files available without giving any assurance about the integrity of the deliverables and whether they can be accepted by the judicial authorities or not.

In 2011, while studying the Eucalyptus cloud platform, the interaction of its components was recorded in logs. This detected a Distributed Denial of Service (DDoS) attack that started through the cloud being examined [8]. The same year, Birk and Wegener were the first to suggest the use of a read-only API to enable cloud customer to download data from it and offer them to the investigators [9].

According to [10,11], due to the heterogeneous nature of cloud log files, at least 1) When the event is logged, 2) What is logged precisely, 3) Which is the format of logging, should be specified. Timestamp, Application, User, Session id, Severity, Reason and Categorization are defined as minimum components of logging following the "key-value" format.

In research regarding ACPO guidelines and Cloud Forensics [12], the impact and issues during a digital investigation on the cloud while trying to apply the four principles were presented. Methodology differs accordingly to the deployment model (private, public, hybrid, community). In a private cloud, data stores are accessible and in known locations – by authorized personnel. The organization's staff can also be familiar with the auditing procedure. On the other hand, public clouds are owned and managed by a Cloud Service Provider (CSP) in a way that clients have no clue about the underlying infrastructure.

Table 1
7W's of audit.

W	Description
What	What was the action and what was its result
When	When did the event take place
Who	Who triggered the event
From Where	From where was the event triggered
On What	What was the target of the action
Where	Who observed and reported the event
To Where	Where is the target located

As it has been already pointed out [13], in Infrastructure as a Service (IaaS) model, the client is responsible for the investigations – except for exceptional cases. This paper disagrees with this point. A CSP must actively assist and be part of the investigative process. CSP will be benefited in general more than the client if the investigation is successful and the case is closed. In case of an incident, it is likely that a vulnerability on the CSP's platform was exploited and sensitive data may have breached out. Often, the consequences of such cases can harm the corporate reputation and business activity of the CSP.

A different approach was attempted in 2015, suggesting the Open Cloud Forensics (OCF) model [11]. There are four entities in the OCF model - user, CSP, investigator and judicial authorities. Based on this, every access to the cloud produces Electronically Stored Information (ESI), a thing that is not acceptable to the legal authorities. In order to be accepted, the CSP must follow a specific procedure to render it valid to the judicial authorities. This process is continuous (Continuous Forensics Process). Although it does not explicitly refer to log files, the core idea includes them, as long as they belong to data that the CSP must validate.

Pătraşcu and Valeriu Patriciu presented in 2015 a forensics framework that can be integrated into existing cloud infrastructures as well as into new ones. After comparing two forms of representing the event data, "Management Metalanguage" by UnixWare[1] and Common Event Expression (CEE) Language,[2] they agreed on a combination of both. In this way, they leveraged the functionality of Management Language with the help of JSON data visualization [14]. Then, they divided the cloud architecture into Management and Virtualization. At the management level, they added a Cloud Forensics Module that assumes to log data from the interaction of the cloud components. At the level of virtualization, they suggested using Cloud Forensics Interface to collect data from the virtual machines internal. This suggestion - at least in terms of management - reminds strongly of the logic of a telemetry component. Their idea was applied only to KVM hypervisors. Data that can be collected are well above the log files since it is possible to download virtual disks and memory dumps.

In 2015, DMTF defined the seven essential questions that need to be answered in order to fully describe an event, known as the 7W's of audit [15] (see Table 1).

Kumar Raju and Geethakumari studied in 2016, the possibility of correlating information contained in log files from the perspective of the forensic investigator on the OpenStack platform [16]. A normalization process was applied to log files in order to resolve the problem of common and single format temporarily. Correlation of information was made on log files either from homogeneous artefacts or heterogeneous.

[1] http://uw714doc.sco.com/en/UDI_spec/m_mgmt.html.
[2] http://cee.mitre.org/language/1.0-beta1/cls.html.

Table 2
Logging formats.

Format	Type	Proposed by	Year	Status
CBE	Proprietary	IBM, Cisco	2003	Dead
CIM	Open	DMTF	2005	Alive
CEF	Proprietary	ArcSight	2006	Alive
CEE	Open	MITRE	2007	Dead
OLF	Proprietary	elQNetworks	2007	Dead
WELF	Proprietary	WebTrends	2008	Alive
LEEF	Proprietary	Q1 Labs	2013	Alive
CADF	Open	DMTF	2015	Alive

Sekhar and Murali again raised the issue of trust in the CSP by applying a Secure Logging Services system. Based on the homomorphic encryption of log file records, they protect them against any possible alteration [17]. This can come either from the CSP or from an attacker who has gained access to the CSP and the cloud internals.

The same year, a study proposed a safe logging approach based on data collection and display by combining three algorithms [18]:

1. SystemInit – performed on CSP and produced the necessary components for encryption.
2. KeyGen - performed by the user and produces a type of digital signature.
3. SecLogging - It combines the results of the above two algorithms and encrypts log blocks instead of log entries.

In 2017, the need to establish a single mechanism for collecting log files was identified since existing solutions either do not cover the different cloud models (IaaS, PaaS, SaaS) or do not overcome the obstacles of locating, reviewing and correlating information in them [19]. The most comprehensive solution was the CADF approach, but two issues emerged. The first issue was related to the absence of use cases and tests to demonstrate whether data collected on the basis of CADF are enough to conduct an investigation or not. The second one was the existence of too much data in the log files, a fact that makes research difficult.

The problem of integrating log files has re-emerged in the case study of 3 different storage service providers (Cloud Storage Service Provider), that is Amazon Web Services Simple Storage Service (AWSS3), Google Cloud Platform Storage and Microsoft Azure Storage [20]. The solution suggested is the use of a single format/structure in the log files. This occurs as a result of the following three steps: 1) Collection of log files from different providers, 2) Check for duplicates in the log files, 3) Conversion and normalization to the desired format. The logic followed refers to the one developed by CADF standard.

In 2018, a paper was presented on the detailed examination of cloud logging mechanisms to facilitate forensic investigation [21]. At the same time, CADF standard was implemented for CloudStack system. Penetration tests were then performed to verify the proper operation of this application, but only a Replay Attack type was able to be captured. The reason for this is that the implementation was only obtaining the API calls instead of examining each event explicitly.

The need for event logging is imperative for all sorts of projects, be it simple scripts of a few tens of lines of code or large-scale projects. However, the most important issue is *What* will be logged, *How* it will be presented, in *Which* format, *Where* and in *How* it will be stored. Various log file formats have been suggested and used by organizations and companies from time to time. Gagliardi Rocco, in his article on a corporate blog, lists some of the most crucial Logging formats [22].

It is indicative that from 2003 and onwards CIM and CADF stand out among the open logging standards (see Table 2).

Both are DMTF's work, but there are no other similarities. CIM is a standard for describing software and hardware features in a single way for different manufacturers. Different manufacturer products can be described using common fields such as device name, serial number, model etc. [23]. CADF is thoroughly studied at a later stage of this paper.

Until now (2019), the most widely accepted logging standard is syslog, as described in RFC 5424. It is implemented with minor variations from various operating systems. Although it was developed in 1983 as part of the "sendmail" project,[3] it was officially established as a standard in March 2009 [24]. The majority of tasks performed by modern operating systems and their applications, use syslog format in log files. The main fields of the syslog message are the facility code, severity level, process ID, timestamp, hostname and IP address of the resource logging the event.

2.2. Cloud auditing and forensics

Cloud Auditing is a continuous process designed to measure and provides CSPs with data on the performance and compliance with the safety requirements of their services. These are metrics and statistics that enable CSPs to monitor and continually improve their service and safety level. Cloud Auditing is usually conducted by a third entity if there is no dedicated team or department within the organization. Auditor undertakes to complete the process in stages, depending on the entity's strategy, the provider's checking over the safety of communications, system management and upgrades, data management, risk management and incident management and response. However, it is important to note the fact that many times in favor of the provider's Management, auditing focuses on the performance of system services rather than on the accurate presentation of what exactly is happening in systems at a technical level. Auditing should include management, technical staff and infrastructures [25]. It needs to be extended at all levels (from Hardware, Host OS and Virtualization Software, etc.) as well as in all development models (private, community, public and hybrid). Customers of cloud providers take access to auditing and monitoring data for granted. Customers will trust providers with their corporate and personal data and therefore, they require specific safeguards. Cloud Auditing is about audit procedures that can be expanded vertically in the organizational chart of an organization as well as across its entire infrastructure. However, it is imperative in particular for each department concerned to use a common language and toolbox. Actually, as Konoor points out [26], any operation in the infrastructure must be performed in such a way that it provides relevant information for future analysis and review.

Underlying cloud architecture is not friendly to forensics. Detection and analysis of any event require the use of quick tools but primarily of tools able to limit the scope of the investigation. Cloud forensics tools have to face multiple challenges [3]. When it comes to log files, in particular, there is a difficulty in terms of the criteria according to which it will be determined which logs are useful and which can be used. Furthermore, there is no specific standard for *WHAT* will be logged by the CSP. Regardless of the service model (IaaS, PaaS, SaaS) data that are valuable to the investigator of a cloud incident are not on the customer's side. On customer's side only temporary files, cookies and session information files can be found. These do not provide enough information to associate an activity with more than 3 out of 7 Ws. We see that analysis on the CSP's side is the only way. Taking the trust to the provider's entity for granted, a tool and therefore, the investigator himself, should be able to ask the provider for specific data based on criteria. Re-

[3] http://www.sendmail.org/.

ceiving them requires a secure communication channel, and their processing requires respecting the privacy of other users. The answer to the issues above is called Web API and API Endpoints. The provision of these interfaces is now available by default on every cloud platform. In a forensic investigation, it is necessary to minimize CSP's involvement. In practice, it depends on the cloud deployment model, the owner of the hardware infrastructure and the type of cloud platforms (proprietary or open-source). However, the use of open standards, such as the CADF, may minimize the CSP's possibilities to interfere in the log files.

Naaz and Ahmad studied the possibilities and limitations of FROST and UFED Cloud Analyzer tools [27], whereas a study of researchers was published the same year describing the experiences and difficulties encountered in developing three cloud forensics tools for the SaaS model [28]. These tools are kumodd, kumodocs and kumofs.

Simou pointed that due to the lack of cloud forensics tools, investigators use existing software solutions [4]. They either examine acquired files with them, considering as reliable by default, or they try to perform a remote analysis. However, the tools that enable remote analysis have not been tested and certified regarding the appropriateness and preservation of data integrity and thus do not provide any assurance that they will be accepted by the judicial authorities.

Finally, there are also custom cloud-based solutions applied by organizations for inside use. Netflix is one such case [29]. These solutions are proprietary and are not available to the general public, or they are commercial products, or at best they fall into Free and Open Source (FOSS) category.

The NIST Institute applies a process for the assessment of forensics tools. This assessment includes a series of tests called Computer Forensic Tool Testing (CFTT) and produces results in the form of reports (CFTT Reports). This creates a list of NIST-approved tools[4] for disk imaging, file carving, email parsing, etc. Approval of tools plays a significant role in the acceptance of the tool's results by the judicial authorities. Investigators clearly prefer the use of tools approved by NIST. However, the search in cloud forensics tools list illuminates the scarcity in this field. Only six tools are related to cloud forensics:

1. Belkasoft Evidence Center
2. Elcomsoft Cloud Explorer
3. Elcomsoft Phone Breaker
4. Internet Evidence Finder
5. Magnet AXIOM
6. UFED Cloud Analyzer

3. Current model

At a structural level, CloudStack consists of a number of subprojects. CloudStack events are divided into three categories, 1) Action events, 2) Usage events, 3) Alerts. They are also divided into 1) Standard events, 2) Long running job events. Finally, events are categorized as 1) Synchronous events, 2) Asynchronous/Scheduled events. Inside the *API* subproject, *352* different Action events are defined.

Event logging in CloudStack is performed on the *cloud-server* package, which is the core of the management server. The following operations take place (see Fig. 1):

1. The API call is redirected as a web request.
2. Checks are performed on the request.
3. API call's key parameters are stored.
4. The request is forwarded to the corresponding entity.

[4] https://toolcatalog.nist.gov/.

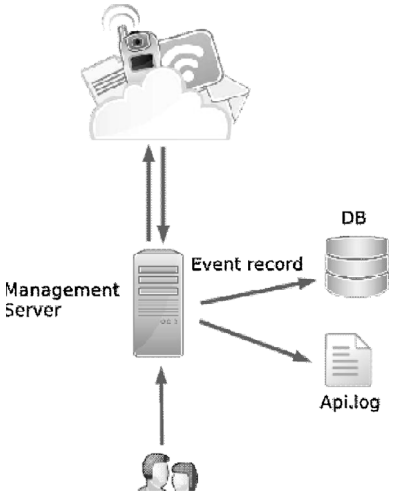

Fig. 1. CloudStack current event logging model.

Table 3
7W's and EventVO.

Question	CloudStack EventVO fields
What	Description
When	createDate
Who	uuid, userId
FromWhere	no information available
OnWhat	Description
Where	no information available
ToWhere	no information available

5. Entity executes the action and creates a response with the results.
6. Event is logged

The built-in class that is responsible for describing the events in CloudStack is *EventVO*. Fields named type, state, description, createDate and userId provide essential information that can be used in forensic investigation. Field "type" contains a string representing the resource that relates to action along with the response. Field "description" is a string and information merger, different for each event, so it does not have a specific structure in order to be parsed by automated tools.

According to Table 3, it is clear that no information is stored in the event concerning the geographical location, e-mail address, exact resource's physical and logical address, platform and system information that triggered the event.

4. Proposed model

DMTF's CADF event model [30] is a model to represent events related to the cloud. In a cloud infrastructure, all the resources interact with each other and exchange data. This exchange must comply with specific rules and regulations. CADF explicitly defines the term event. CADF classifies events based on the type of event, regardless of the cloud component, cloud application or deployment model. Valid event types are 1) monitor, 2) activity, 3) control. Monitor events provide information about the status or attributes of a resource. Activity events are events initiated by a resource against another resource. Control events contain information about the application of a policy. Depending on the type of event, some fields differ, but the core components for each event are standard (see Table 4). CADF collects data (see Table 5) from various cloud layers without exposing though any information or technical details about the underlying infrastructure. Observer resource is constantly monitoring every resource, including itself.

Table 4
Required CADF components.

Component	Description
OBSERVER	The Resource that generates the event record based on its observation of the actual event
INITIATOR	The Resource that initiated the event's Action
ACTION	The operation or activity the Initiator attempted to or performed against Target
TARGET	The Resource against which Action was performed
OUTCOME	The result of the status of the Action against Target

Table 5
CADF event fields.

Field	Description
typeURI	Definition of the event model's version and name
Id	Event's id
eventType	Type of Event
eventTime	The timestamp of the event as noted by the OBSERVER
Action	The action performed against TARGET
Outcome	The outcome of ACTION
Initiator	The resource that performed the Action
Initiatorid	INITIATOR's id
Target	The resource against who action was performed
Targetid	TARGET 's id
Observer	Resource monitoring other resources and logging events
Observerid	OBSERVER's id
Measurements	Measurements and statistics regarding the Event
Reason	More information about the OUTCOME
Name	The descriptive name for the event
Severity	The severity of the Event – used only by OBSERVER
Duration	ACTION's duration
Tags	Tags for storing extra information
Attachments	Extra information for an event
Reporterchain	Information on the reporting chain of the event

Table 6
7W's and CADF.

W	CADF component
What	EventType, ACTION, OUTCOME
When	REPORTER
Who	INITIATOR
FromWhere	INITIATOR
On What	TARGET
Where	OBSERVER
To Where	TARGET

Resource entity is classified into taxonomies [30] (Storage, Compute, Network, Data, Service, System, Unknown). Top-level taxonomies are also divided into secondary categories. Action and Outcome entities are also classified into Taxonomies.

CADF's goal is to provide all the resource-related information to investigators and system auditors in order to assist in tracking certain activities. It is described as the Crime Scene Investigation (CSI) for Clouds [31]. In particular, the model is designed to have the ability to answer all the 7 W's of forensics [15,32] (see Table 6).

5. Implementation

Three top-level classes were created during the implementation of the CADF Event model, 1) Cadf, 2) Resource, 3) Taxonomy. The CADF mechanism is implemented in the subproject *cloud-server*, as part of the management server (see Fig. 2).

Cadf class provides properties and methods for describing a CADF event based on the CloudStack event(EventVO). Cadf class fields *typeURI, eventType, action, outcome, eventTime, measurement, reason, observer, initiator* and *target* are defined. Class structure is similar to CADF model (see Table 4) so that serialization can be performed on each object in JSON format and be logged in a file.

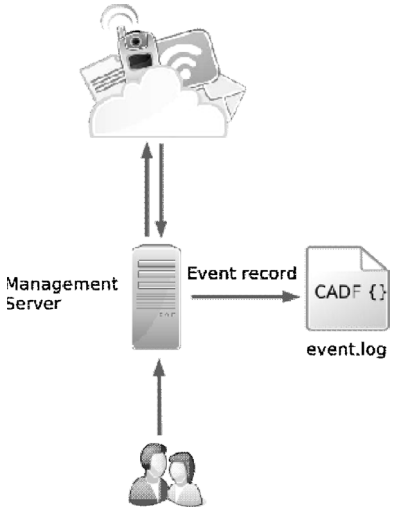

Fig. 2. CloudStack proposed event logging model.

Table 7
Target and action extraction from EventVO.type.

EventVO.type	Target Resource	Action
VM.CREATE	VM	CREATE
ROLE.PERMISSION.CREATE	ROLE.PERMISSION	CREATE

Key operations are applied to match CloudStack entities to CADF compliant model.

We tampered *ActionEventUtils* class, where generic CloudStack events are captured and performed the following operation in order to implement our mechanism without modifying the existing one:

1. When a generic CloudStack event occurs (*ActionEventUtils* class), a method is called (*createCadfRecord()*) with an EventVO object as argument. This method creates a Cadf object and stores it to a logger object
2. Cadf object is created by using as an argument in its constructor the generic CloudStack event object in order to extract all the information without making any changes to the actual event source code. Every object is a single event.
3. A check on the Cadf's mandatory fields is performed based on *eventType* field
4. Cadf object is converted into JSON representation with Google's Gson library (https://github.com/google/gson)
5. JSON representation of the object is logged

Inside Cadf's constructor, we collect all the necessary information and match all CloudStack and CADF entities accordingly:

1. Cadf class constructor's parameter is the generic CloudStack event object (EventVO). The class field *typeURI* is set in order to declare this event as a event.
2. CADF's Action and Target are extracted from EventVO.type property (see Table 7). A function (*setCADFAction()*) performs pattern matching against hashmaps to match the CloudStack

event action with a corresponding CADF Action. Cadf class field *action* is set. The hashmap table is implemented in Taxonomy class.
3. A function (*getCADFResourceName())* matches the extracted target property against a hashmap (Taxonomy class) with CADF standard resource names. *Target* Resource object is created (Resource class) based on this matching.
4. Cadf class field *eventType* is set based on the *action* class field. This is important, as for different types of events (*eventType* field) the mandatory fields of the record also differ.
5. A function (*mapEventStateToCADFTaxonomy()*) sets the Cadf class field's *outcome* value based on a matching between generic CloudStack's event *state* property (Taxonomy class)
6. Cadf class field *eventTime* is set on a UTC format
7. Cadf class field *Initiator* is created (Resource class)
8. Cadf class field *Observer* is created (Resource class)
9. Additional information related to the event is collected – currently the initiator host, user agent, userid, and account name are gathered, but these can be extended in a future work

CADF has been created to cover any kind of event for any platform. Some platforms use a specific resource for monitoring and another for management, but CloudStack does not offer this option. Furthermore, this implementation refers only to events that resulted from a user action. For this reason, there is a manual assignment to Initiator. Some additional information is being collected on the Management Server through a list of values added to CADF as custom fields, such as *initiator_userid* and *initiator_csAccountName* which particularly refer to CloudStack. Other fields are the Initiator's IP address, user-agent, platform etc.

Resource class implements properties and functionality of the resource. A resource is used to represent entities of a CADF event. Initiator, Observer and Target are Resource objects. In order to describe a Resource, there are mandatory properties such as typeURI, id, name, domain, credential, addresses, host, geolocation, attachments. Entities depending on the Resource, such as Host, are defined as subclasses.

Taxonomy class is auxiliary. It contains the enumerations of various fields of the CADF and Resource classes, such as *eventType, Action* and *Outcome*. It also contains the string constants about the taxonomy-related *Reason* fields as well the string constants for each type of resource supported by CADF. However, the most important operation of Taxonomy class is the creation of matching lists among entities of CloudStack and CADF. Matching CloudStack's resources and actions with corresponding CADF-compliant entities requires an understanding of each CloudStack event (352 total events). Events that could not be matched were defined as "*unknown*". In Taxonomy class 3 hashmap structures were created in order to match related entities

1. *CstoCadfResourceMapping* for Resource matching
2. *EventActionToTypeMapping* for Action matching
3. *EventResourcetoUuidMapping* for unique UUID assignment

Additionally, some property files were configured (*build/replace.properties* and *client/conf/log4j-cloud.xml.in*) so that our mechanism logs event records in a separate file.

Apache CloudStack 4.11 was forked from the original project to a branch named "*cadf_events*". This implementation is available for download and testing.[5]

5.1. Testing

Testing the functionality of the CADF model was performed in 3 ways:

1 CloudStack's web interface
2 CloudMonkey CLI[6]
3 DevCloud 4 testing environment and marvin scripts (deployDataCenter.py)

After committing this implementation, a pull request was placed in CloudStack's main repository with ID #3232. Pull request status.[7] A tool presented in Section 8 makes use of testing's output file containing all the events produced during the testing phase. The dataset is also available.[8]

5.2. Validation

All the changes made do not affect CloudStack's functionality. All the existing tests (mvn tests while constructed) were successfully passed along with the automated validation checks that are performed after a pull request. Some Cloudstack committers reviewed our proposal and commented suggestions for minor organizational issues which were addressed and fixed. Additional changes were made, and this pull request is now marked with GitHub labels "component:logging", "type:enhancement" and "type:improvement". In June 2019, a review was requested from the rest of the committers, which is still pending.

All the CADF event records that were logged during our tests were stored on a dataset following the JSON CADF format and is available for download[8] along with a dataset[9] which it contains CADF records from OpenStack' utilization for further research.

6. Discussion

6.1. Functionality

Upon completion of the implementation, the 7 Ws can be answered by CADF as opposed to the default CloudStack model. Comparison of Table 3 and demonstrates the superiority of the suggested model against the existing one. The functionality of the CADF model can be divided into *simple* and *advanced*. Simple is limited to filling only mandatory fields for each type of event, while advanced includes as many fields as possible. However, both in the OpenStack project and the Apache CloudStack, CADF was not the event mapping model for which the platform was designed. This leads to an arbitrary decision whether changes need to be made within each event to provide through source code the necessary information to fill in the CADF fields, or whether matching of fields and values should - by losing precision - be made at a central point.

The most appropriate decision is to combine the above approaches and to understand and change each event separately. However, the process has some drawbacks. It is time-consuming since it concerns 352 different events. It is complicated because each event according to its type contains various information and has a severe difficulty in terms of integration and merging with the original project, as for each event a change must be made to at least 1 CloudStack file.

6.2. Suggestions and improvements

In order for CloudStack's Action Events to comply with the CADF standard, it is necessary to match each one of them with the corresponding CADF event. CADF's Action Taxonomy includes 26 different actions. Matching is a significant problem because events

[5] https://github.com/ndalezios/cloudstack/tree/cadf_events.

[6] https://github.com/apache/cloudstack-cloudmonkey.
[7] https://github.com/apache/cloudstack/pull/3232.
[8] https://github.com/ndalezios/clod/blob/master/cs_events.
[9] https://github.com/ndalezios/clod/blob/master/raw_cadf_sample.

Table 8
Cases of event naming pattern issues.

CloudStack Event	Note
VM.DESTROY	Action "DESTROY" does not exist in CADF's Action Taxonomy
NETWORK.CREATE	Nothing to note – matches the suggested pattern
PROXY.DIAGNOSTICS	Action "DIAGNOSTICS" does not exist in CADF's Action Taxonomy. If it existed, it should be named "DIAGNOSE". The correct naming should be PROXY.DIAGNOSTICS.START
NET.RULEADD	Action RULEADD does not exist in CADF's Action Taxonomy and should be renamed to "RULE.CREATE".
CREATE_ RESOURCE_DETAILS	The action does not follow any naming pattern (use of "_") and therefore cannot be ported to CADF
CloudStack Event	Note
VM.DESTROY	Action "DESTROY" does not exist in CADF's Action Taxonomy

```
//TODO change value to VM.PASSWORD.RESET
public static final String EVENT_VM_RESETPASSWORD = "VM.RESETPASSWORD";
//TODO change value to VM.SSHKEY.RESET
public static final String EVENT_VM_RESETSSHKEY = "VM.RESETSSHKEY";
//TODO change value to ROUTER.DIAGNOSTICS.START
public static final String EVENT_ROUTER_DIAGNOSTICS = "ROUTER.DIAGNOSTICS";
```

Fig. 3. Excerpt from EventTypes.java with naming suggestions as comments.

Fig. 4. Event log line format.

in CloudStack are described verbally in the form of strings, by merging the target resource with the action. CloudStack's events do not follow a specific standard. Events are characterized by the *event_type* field in terms of resource and action. In contrast, CADF provides an *eventType* field whose values determine if it is a monitor, control or action event. Table 8 illustrates some indicative cases.

Following the aforementioned events cases, as defined in CloudStack, there is a need to use a single event type format exclusively for CloudStack, which could be generalized across other platforms. The suggested format is a *RESOURCE. (SUB-RESOURCE). (SUB-RESOURCE). ACTION.* Where, in uppercase Latin characters, RESOURCE and SUB-RESOURCES (if any) are nouns, and ACTION is a verb (*noun. (noun). (noun).verb*). By following this format, even if there is no mapping for the Resource or Action in the CADF's taxonomies, it is easy to create an events' map. Fig. 3 illustrates the comments inside source code with our naming suggestions

When a CloudStack user deletes one or more events from the database, apart from the deletion, no other action is taking place. There is no information on who removed what, when, from where etc. Events' removal should trigger an EVENTS.DELETE event. In fact, only upon the first installation of CloudStack, event storage should be empty. We also implemented this functionality as an improvement of the existing logging mechanism. Any action to delete event records must be considered as an event itself. Only in this way, a malicious deletion of a history of actions from CloudStack can be identified.

7. ACPO principles and CADF

As mentioned in [33], four principles related to the practices of handling and maintaining digital evidence have to be respected. Lallie and Pimlott [12] studied the application of the above principles to investigations in public clouds and highlighted a series of issues for each one of them. The CADF event model provides the opportunity to solve some of these issues.

Regarding the 1st principle, they detect a time gap between the incident and the beginning of the investigation, where there is no control over the data. However, the auditing and the proposed CADF event logging solution can solve this issue by enabling the investigator to collect log files from a time point A (before the incident being investigated) up to a time point B (at the time of the beginning of the investigation). In this way, although there is no data protection against alterations, there is a log entry of these alterations (if any).

Regarding the 2nd principle, since it is only about log files, there is read-only access permitted. Besides, "Acquisition" is a formal and defined process in common (not cloud) systems. Therefore, the required level of competency is not increased. In fact, the acquisition of log files can be carried out by the CSP itself – depending on the deployment model. However, if any change occurs in log files, this change should be considered as an event and should be logged. In particular, the process of deleting logged events does *not* constitute an event and is not registered in the CloudStack platform. We suggested and implemented creating an event record for this particular action.

According to the 3rd principle, provisioning of an audit trail for each action on the digital evidence should be provided. A third investigating entity should have the same results by performing the same actions. Due to the absence of a unified form of log files, the investigator often resorts to custom solutions depending on the platform investigated (OpenStack, CloudStack, AWS, MS Azure). The process of repeating actions to achieve the same result involves also developing this custom solution. The answer to this comes with the introduction of a specific logging template and the creation of standardized tools - either open source (where it is easier to check if they affect the integrity of evidence) or commercial solutions. A tool for all platforms can pass the necessary checks to be certified by NIST.

8. CADF consumers

The use of an event logging standard exempts developers from the need to study and configure the event analysis tools according to the platform that produced the log files. Therefore, they focus exclusively on the standard and the interpretation of its fields. One such attempt is the C.Lo.D tool (CADF Log Detective), available under Apache 2.0 license in a public repository at GitHub.[10] Clod's input is a log file containing log records in the form of CADF as JSON. Each line in the file usually includes information about the platform before and after the record (platform-specific information – see Fig. 4).

Clod parses input file and searches each line for the CADF event identifier. Once it locates it, it isolates it from where it was found, up to the point where the number of "{" and "}" symbols are equal. In this way, only CADF data are stored in a list. Then, CADF events are stored in a NoSQL database (see **Fig. 5**). In particular, a "document store" type of database manager is selected to ideally handle JSON documents. The open-source version of the MongoDB Community Server is used as the database manager. From this point

[10] https://github.com/ndalezios/clod.

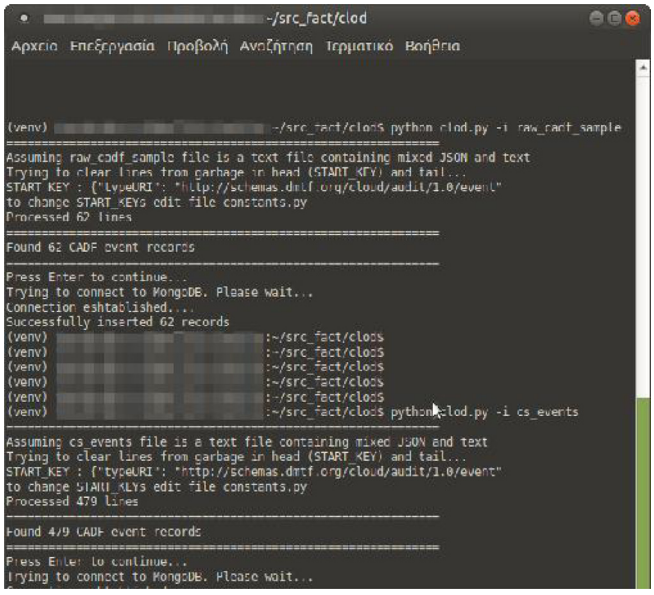

Fig. 5. Clod is processing two datasets. The file raw_cadf_sample is OpenStack's output, and cs_events is CloudStack's testing phase output.

onwards, an investigator is able to, with a MongoDB client or by source code, prepare and execute queries over stored CADF events.

9. Conclusions

This paper has proposed the CADF event model implementation for the Apache CloudStack project as the main logging format. CADF is a simple yet clear, expandable and robust model focused on the events and not the underlying technology of the cloud infrastructure. OpenStack, at its current version, is using CADF – OpenStack was CADF's targeted platform. Even OpenStack, though, does not fully implement CADF. This paper's implementation for CloudStack is not utilizing CADF's full capabilities. The reason for this is that in order for an event model to collect as much information possible, it needs to touch every single event that occurs on the platform. It has to be part of the initial design of the platform. It cannot be added later on as an enhancement, because it can cause major changes and side-effects to the entire project. Any platform or project that can be exploited, or attacked, or used by any manner by cybercriminals should take into consideration while being designed, to be as much forensically friendly as possible. The engineering team and the security team must work together and not only test a project for vulnerabilities, but also make sure that in case of an event, all the necessary information is collected, under a standard as CADF.

10. Future work

In future work, a set of cyber-crime scenarios on CloudStack should be executed and then perform forensic analysis by using just the information stored in the CADF logs. This will bring to the surface all the data that should be included in the event logging.

Forensics investigators, police officers, Law Enforcement Agencies and even judicial officers should examine if CADF and data extracted from CADF, can solve a digital crime case. Are CADF data enough to solve a case? At the same time, it should be examined if CADF data could be accepted by the judicial authorities.

Additionally, after having an implementation for OpenStack and CloudStack, one should port or develop CADF to OpenNebula, the third in ranking open source IaaS cloud platform.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.jisa.2020.102555.

CRediT authorship contribution statement

Nikolaos Dalezios: Methodology, Investigation, Software, Resources, Writing - original draft. **Stavros Shiaeles:** Supervision, Conceptualization, Methodology, Resources, Writing - review & editing. **Nicholas Kolokotronis:** Resources, Writing - review & editing. **Bogdan Ghita:** Writing - review & editing.

References

[1] Dykstra J. Seizing electronic evidence from cloud computing environments. In: Cybercrime and cloud forensics: applications for investigation processes. IGI Global; 2013. p. 156–85.
[2] Wall D. "Towards a conceptualisation of cloud (Cyber) crime," 2017, pp. 529–538, doi: 10.1007/978-3-319-58460-7_37.
[3] Iorga M., Simmon E. "DRAFT NISTIR 8006, NIST cloud computing forensic science challenges," National Institute of Standards and Technology, U.S. Department of Commerce, Jun. 2014.
[4] Simou S, Kalloniatis C, Gritzalis S, Mouratidis H. A survey on cloud forensics challenges and solutions. Security Commun Netw 2016;9(18):6285–314. doi:10.1002/sec.1688.
[5] Ruan K, Carthy J, Kechadi T, Crosbie M. Cloud forensics. In: Advances in digital forensics vii; 2011. p. 35–46. Berlin, Heidelberg. doi:10.1007/978-3-642-24212-0_3.
[6] Simou S, Kalloniatis C, Kavakli E, Gritzalis S. Cloud forensics: identifying the major issues and challenges. In: Advanced information systems engineering; 2014. p. 271–84. Cham. doi:10.1007/978-3-319-07881-6_19.
[7] Alqahtany S, Clarke N, Furnell S, Reich C. Cloud forensics: a review of challenges, solutions and open problems. In: 2015 international conference on cloud computing (ICCC); 2015. p. 1–9. doi:10.1109/CLOUDCOMP.2015.7149635.
[8] Zafarullah FA, Anwar Z. Digital forensics for eucalyptus. In: 2011 Frontiers of information technology; 2011. p. 110–16. doi:10.1109/FIT.2011.28.
[9] Birk D, Wegener C. Technical issues of forensic investigations in cloud computing environments. In: 2011 Sixth IEEE international workshop on systematic approaches to digital forensic engineering; 2011. p. 1–10. doi:10.1109/SADFE.2011.17.
[10] Marty R. Cloud application logging for forensics. In: Proceedings of the 2011 ACM symposium on applied computing; 2011. p. 178–84. New York, NY, USA. doi:10.1145/1982185.1982226.
[11] Zawoad S, Hasan R, Skjellum A. OCF: an open cloud forensics model for reliable digital forensics. In: 2015 IEEE 8th international conference on cloud computing; 2015. p. 437–44. doi:10.1109/CLOUD.2015.65.
[12] Lallie H, Pimlott L. Applying the ACPO Principles in Public Cloud Forensic Investigations. J Digital Foren Security Law Jan. 2012;7(1). doi:10.15394/jdfsl.2012.1113.
[13] Badger M.L., Grance T., Patt-Corner R., Voas J. "Cloud computing synopsis and recommendations," National Institute of Standards and Technology, Gaithersburg, MD, NIST SP 800-146, 2012.
[14] Pătraşcu A, Valeriu Patriciu V. Logging for cloud computing forensic systems. Int J Comput Commun Control Apr. 2015;10(2):222–9.
[15] Cloud Auditing Data Federation (CADF) Working Group, "Cloud Auditing Data Federation - (CADF-OpenStack) - a CADF representation for OpenStack." Distributed Management Task Force, Inc. (DMTF), 16-Apr-2015.
[16] Kumar Raju B, Geethakumari G. Event correlation in cloud: a forensic perspective. Computing Nov. 2016;98(11):1203–24. doi:10.1007/s00607-016-0500-2.
[17] Sekhar BC, Murali G. Access control for cloud forensics through secure logging services. In: 2017 International conference on energy, communication, data analytics and soft computing (ICECDS); 2017. p. 3527–32. doi:10.1109/ICECDS.2017.8390116.
[18] Chen Z, et al. Secure logging and public audit for operation behavior in cloud storage. In: 2017 IEEE international conference on computational science and engineering (CSE) and ieee international conference on embedded and ubiquitous computing (EUC), 1; 2017. p. 444–50. doi:10.1109/CSE-EUC.2017.85.
[19] Alobaidli H., Nasir Q., Iqbal A., Guimaraes M., "Challenges of cloud log forensics," 2017, pp. 227–230, doi: 10.1145/3077286.3077302.
[20] Sukmana MIH, Torkura KA, Cheng F, Meinel C, Graupner H. Unified logging system for monitoring multiple cloud storage providers in cloud storage broker. In: 2018 International conference on information networking (ICOIN); 2018. p. 44–9. doi:10.1109/ICOIN.2018.8343081.

[21] Alobaidli H., Nasir Q., Abutalib M., "CADF logging infrastruture for cloud computing digital forensics," 2018.
[22] Gagliardi R. Security log standard - still an open question. SCIP Labs 15-Mar-2018. Online]Available https://www.scip.ch/en/?labs.20180315 Accessed: 21-Oct-2018.
[23] Sturm R, Pollard C, Craig J. Application performance management (APM) in the digital enterprise: managing applications for cloud, mobile, iot and eBusiness. Morgan Kaufmann; 2017.
[24] Gerhards R. "The syslog protocol," RFC editor, RFC5424, Mar. 2009.
[25] Pătraşcu A, Patriciu VV. Logging framework for cloud computing forensic environments. In: 2014 10th international conference on communications (COMM); 2014. p. 1–4. doi:10.1109/ICComm.2014.6866662.
[26] Konoor DK. Auditing in cloud computing solutions with OpenStack. In: 2016 IEEE international conference on cloud computing in emerging markets (CCEM); 2016. p. 176. doi:10.1109/CCEM.2016.042.
[27] Naaz S, Ahmad F. Comparitive study of cloud forensics tools. Commun Appl Electron Jun. 2016;5(3):24–30. doi:10.5120/cae2016652258.
[28] Roussev V, Ahmed I, Barreto A, McCulley S, Shanmughan V. Cloud forensics–tool development studies & future outlook. Digital Investig Sep. 2016;18:79–95. doi:10.1016/j.diin.2016.05.001.
[29] Blog NT. Netflix SIRT releases Diffy: a differencing engine for digital forensics in the cloud. Medium 17-Jul-2018 [Online] Available https://medium.com/netflix-techblog/netflix-sirt-releases-diffy-a-differencing-engine-for-digital-forensics-in-the-cloud-37b71abd2698 Accessed: 21-Dec-2018.
[30] Cloud Auditing Data Federation (CADF) Working Group, "Cloud Auditing Data Federation (CADF) - data format and interface definitions specification." Distributed Management Task Force, Inc. (DMTF), 19-Jun-2014.
[31] R. Basham, G. Chung, M. Rutkowski, and B. Topol, "An overview of cloud auditing support for OpenStack," presented at the OpenStack Summit, Atlanta, USA, 13-May-2014.
[32] Bangur S, Verma D. Adoption of Cloud Auditing Data Federation (CADF) standard by IBM spectrum virtualize. IBM Syst. Mar-2017.
[33] Williams J. "ACPO_Good_Practice_Guide_for_Digital_Evidence_v5.pdf." ACPO Crime Business Area.