UNIVERSITÀ DEGLI STUDI DI SALERNO

DISTRA - MIT

Dottorato Internazionale di Ricerca

Sistemi Informativi e Ingegneria del Software

XII Ciclo, Nuova Serie

Tesi di Dottorato in

Forensic Readiness Capability for Cloud Computing

Doctoral Dissertation of

Lucia De Marco

Ph.D. Coordinator	Advisors
Prof. Filomena Ferrucci	Prof. Filomena Ferrucci
	Prof. M-Tahar Kechadi

Anno Accademico 2014-2015

ACKNOWLEDGMENT

The University of Salerno where I got my education basis gave me the chance to participate to this extraordinary experience of an international Ph.D. At the end of such a programme I can say that I am deeply grateful to it that better educated and changed my person in many different ways.

In particular, I have to immensely thank my supervisor Professor Filomena Ferrucci from University of Salerno, for her continuous and invaluable support and advice. I had the privilege to learn from such an extraordinary academic, and the honour to be guided by such a big experienced woman, who always involved me despite the physical distance that this experience put between us.

I also have to thank my supervisor Professor Tahar Kechadi from University College Dublin, who adopted me in Ireland since the the first day I landed. He granted me the honour and privilege to study and work on an incredibly fascinating and interdisciplinary research area. He always professionally guided me throughout the time I spent in Dublin, and offered me the possibility to be totally integrated in a foreign context.

I wish to thank Dr. Michela Bertolotto from University College Dublin, for her support and constant presence throughout this long-lasting experience. She has been an irreplaceable lighthouse for some specific concerns and I am grateful to her too.

I wish to thank the Ph.D. student Pasquale Salza from University of Salerno, for his determining contribution to the final activities of my research work.

I also wish to thank Dr. Sameh Abdalla from University College Dublin, who I worked and spent very nice days with. It has been a pleasure to work with him.

I also wish to thank my colleagues and personnel from both University of Salerno and University College Dublin, for providing a professional, competent, and stimulating education environment.

I lovely thank my friends from Italy, Ireland, and other parts of the world, who have always been a great source of laughter, joy and support, very important to tackle every day of such a big experience.

A final but deeply special and lovely thanks to my father, mother, and sister, for believing in my dreams, for supporting me in the incredibly bad days and challenges that I faced, and for never denying a encouragement and a smile in the good days. The physical distance never been a problem when love is deeply true.

Contents

Abstract

Cloud computing services represent the actual computation delivery to the most of customer communities. Such services are regulated by a contract called Service Level Agreement (SLA), cosigned between customers and providers. During its validity time several contractual constraints have to be respected by the involved parties. Due to their popularity, cloud services are enormously used and unfortunately also abused, especially by cyber-criminals. A manner for guaranteeing and enhancing cloud service security is the provisioning of a forensic readiness capability to them. Such a capability is responsible to perform some activities aimed to prepare the services for a possible forensic investigation. Sometimes, the crimes are related to some contractual constraint violations without the parties are aware of. Thus, a dedicated forensic readiness capability interacting with cloud services and detecting the SLA violations by analysing some cloud log files can guarantee more control on such contracts. In this dissertation, a formal model aimed to represent a forensic readiness capability for the cloud that detects contractual violations is presented, together with a prototype system running on a specific case study.

Chapter 1

Introduction

1.1 Problem

During the last half century humans have been both actors and spectators of rapid changes in the whole technological world, especially for what concerns both the devices and the manner in which computation is accessed. In the modern IT era, the computation capability is delivered through some services, and at this time the most popular vehicle for services is the cloud computing architecture type.

Cloud computing [81] is a manner in which hardware and software resources are delivered through the Internet at a pay per use rate. Cloud services are designed to provide a computing environment that utilizes virtual resources that dynamically allocate the underlying physical ones. The result is to balance the load and to scale resources provisioning up and down in order to guarantee some services arriving at the needs of the end users.

Unfortunately, the ease of access to such resources is exploited by the criminals who design more sophisticated and targeted methods to hack any type of digital device, or to exploit existing computing platforms for illegal behaviour. A digital forensic investigation (DFI) is triggered in order to conduct and resolve a cyber crime [80]. Since a standard for DFI processes does not exist, several models have been proposed in the literature. Each of them represents an investigation process composed of phases and sub-phases, when necessary. Sometimes, a DFI is customised for specific computing architectures.

Information systems and computing capabilities delivered through the Internet in the form of services are regulated by a Service Level Agreement (SLA) contract co-signed by a generic Application Service Provider (ASP) and the end user(s) [50]. Also cloud services are regulated by SLAs, where all the constraints are detailed. The contracts are co-signed by the parties, and have legal validity in case of a court litigation.

Security and reliability of cloud applications are very important issues, broadly considered by both cloud customers and the scientific community. Therefore, a manner to render a service more protect in case of cyber crimes can be considered as a matter of urgency. The provisioning of a forensic readiness capability [123] to a cloud environment is considered as a possible solution to this issue, and broadly discussed in this dissertation.

1.2 Motivations

Cloud computing services have been being adopted in both private and business contexts. This escalation is a revolutionary event for forensics; thus, the branch of cloud forensics has been introduced by the necessity to deal with a computing infrastructure that cannot be investigated with the available forensic tools and procedures [112] . At the same time, practitioners are required to manage cloud evidence respecting the existing admissibility and reliability principles for digital evidence [1, 89].

The adaptation of existing forensic procedures to computing features is a constant and challenging task; moreover, the provisioning of a forensic readiness (FR) capability to computing infrastructures is sometimes complicated. Such a capability aims to prepare computing architectures for forensic investigations with the advantage of optimizing the whole process [111].

FR can be conceived as the provisioning of an information system communicating with such architectures, with the purpose of identifying, collecting, and storing critical data coming from them. This FR capability must be provided to cloud computing architectures because, due to their escalating popularity, they can be object of several attacks, and a way to conduct forensic investigations effectively, saving time, money and resources, must be designed.

The side effect of such capability is the enhancement of security aspects of a cloud infrastructure and the immediate availability of information derived by a constant monitoring of the platform itself. Such information can obviously be related to potential forensic evidence; also to some organisational internal aspect, such as data access control or customer behaviour or profiling, thus becoming a source of data to exploit for different business-related purposes.

1.3 Challenges and Objectives

Cloud solutions have such a strong influence on all the aspects of human life, from personal to business. In most cases, alongside progress, technology is a fertile ground for the increase of criminal activities. Unfortunately, this phenomenon is becoming more commonplace.

ICT progress is the engine of the Digital Forensic (DF) science, which deals with the investigation of information gathered from digital devices in order to resolve court cases. A constant issue in this discipline is the evolving nature of the investigation procedures, which have to follow the giant steps moved by technology in short time periods in order to be effective.

In this doctoral dissertation some challenges have been derived and analysed; they concern some technical, organisational, and legal aspects of cloud forensics generated by some specific features of cloud architectures. In forensic readiness some challenges of them have been considered and then addressed by the proposal, i.e, by the formal model and the prototype system representing a forensic readiness capability for cloud computing platforms.

1.4 Contribution

A DFR capability for the cloud is meant to observe and record changes in the underlying computing architecture for the aims described above. Such changes concern the operations happening in the cloud with respect to the SLA constraints related to potential crimes. The output of the capability include important investigative details about the recorded information and the detection of contractual clause violations. A means for implementing such a DFR capability in the cloud includes a representation of the information to monitor. The most effective representation is the adoption of formalisms.

One of the contributions of this doctoral dissertation on the topic of forensic readiness is represented by a definition for Forensic Readiness. Then, a reference architecture for the implementation of an FR system for the cloud is designed and illustrated, together with some constraints and advantages. A rigorous systematic literature review is conducted in this dissertation in order to determine when forensic readiness has been considered by the existing DFI processes proposed by the scientific community. The SLA constraints relevant for forensic readiness are mapped on the most crucial security threats and sources of attacks in the cloud.

In this dissertation, a classification about some contractual contents is proposed discussed; then an automation for SLAs classification is designed and validated. Natural language-based SLAs clauses, cloud logs, and several entities necessary to output a comparison between them, have been structured via formal specifications. The formal model utilizes tuple, set theory, and functions, to represent the necessary entities.

The formal model is validated via a prototype system implementing a case study, illustrated in the final chapters of this dissertation. This prototype has been utilised to design the architecture, and to implement the routine detecting some SLAs constraints violations. The system has been tested to detect a crime happening in the underlying architecture and specific test cases have been designed.

The automation for SLA classification correctly identifies the information to monitor; then they are given as input to the developed prototyped system. Another input to such a prototype is represented by cloud services logs: they are obtained in a controlled and simulated environment. The prototype routines are launched and the test cases results are obtained: they highlight that the contractual violations are detected in a real time manner, and also the potential crime alert is generated by using the illustrated mapping between the cloud log information and the security threats.

All the contributions of such a dissertation can be enlisted as follows:

- A definition for a forensic readiness capability;
- A reference architecture for a forensic readiness capability for the cloud;
- A systematic literature review about forensic readiness in digital forensic investigations;
- A mapping of SLA constraints onto cloud security threats;
- A classification of SLA forensic-related constraints derived by the previous mapping, together with its automation and assessment;

- A formal model about a cloud forensic readiness capability and constraint violations;
- A prototype system design validating the formal model about a cloud forensic readiness capability;
- A system test suite planning and execution on a case study for a cloud forensic readiness capability.

1.5 Thesis Organisation

The doctoral dissertation is structured as follows: Chapter 2 presents an overview of cloud computing; in Chapter 3 the Digital Forensic science and Digital Forensic Readiness are described; Cloud Forensic is illustrated in Chapter 4. A systematic literature review about forensic readiness in digital investigations is discussed in Chapter 5. Service level agreement contracts are introduced in Chapter 6 and their impact on cloud forensic readiness in Chapter 7. In Chapter 8 a classification for SLAs and its automation are proposed. The formal model for cloud forensic readiness is presented in Chapter 9, and the prototype system architecture for it is described in Chapter 10. A case study on Amazon S3 is discussed in Chapter 11. The simulation platform necessary to generate log files to feed the prototype is illustrated in Chapter 12, while the system testing in Chapter 13. Some conclusion and future work are presented in Chapter 14 that closes the dissertation itself.

Chapter 2

Cloud Computing

2.1 Introduction

This chapter is dedicated to provide an overview of cloud computing. An introduction about the history of the cloud begins the chapter. Some details about definition, features, service models, and deployment models are illustrated.

2.2 History

In the modern IT era computation capability is delivered through some *computing services*. This name became popular to the public when the Service Oriented Architecture Web Services (SOA-WS) spread in the 90s [7]. The concept of delivering IT and computation capabilities through a network dates back to the late 60s. In 1969 J.C.R. Licklider, who contributed to the development of the Advanced Research Projects Agency Network (ARPANET), promoted the concept of an *intergalactic computer network* [77]. Such a scientist had a vision and a hope that in the future every individual would have access to data and applications from anywhere. In 1961 the computer pioneer John McCarthy in 1961 predicted that *computation may someday be organised as a public utility* [51].

The most popular and accurate definition of network-delivered service is the one published by the World Wide Web consortium (W3C), which states that a web service is *a software application identified by a URI, whose interfaces and bindings are capable of being defined, described, and discovered as XML artefacts. A web service supports direct interactions with other software agents using XML-based messages exchanged via Internet-based protocols* [130].

The word *cloud* was firstly introduced in the context of telecommunications in the 1990s, as *telecom cloud* when some data communication providers started using Virtual Private Network (VPN) services. An important feature of such VPNs is their capability to maintain a level of bandwidth comparable to that of non virtual networks; the main advantage is to reduce costs, perform dynamic routing that allows resource balancing across the network, and to increase bandwidth efficiency, among others.

Cloud computing service architecture shares many of those features. Indeed, it is designed to provide a computing environment that utilizes virtual resources that dynamically allocate the underlying physical ones. The result is to balance the load, and to scale resource provisioning up and down in order to guarantee some services arriving at the needs of the end users.

Cloud services can be seen as evolution of SOA-WS services; the cloud is taking advantages from them to build the services architecture, as described in the next section.

2.3 Definition and Features

Cloud computing is a manner of delivering hardware and software resources through the Internet at a pay per use rate. There exists an effective and complete definition for cloud computing provided by the National Institute of STandards (NIST) [81] for cloud computing: *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. This cloud model is composed of five essential characteristics, three service models, and four deployment models.*

Such a definition is the most widely adopted because it includes all the features that make the cloud different from the previous computing delivery architectures. As mentioned in the definition, there are five characteristics concerning the physical infrastructure, and that represent the strengths of the cloud itself.

The first characteristic concerns the cloud services resources; they are automatically provisioned by the provider depending on the customer needs, without physical interactions with the provider.

Secondly, such services are available over the Internet, which can be accessed from any physical location and through almost any digital device. The only requirement to guarantee the access to services is the reliability of the internet connection, which is a customer concern.

The third characteristic affirms that the resources are shared among several customers that utilize the services they subscribed for. The manner in which such resources are pooled gains its advantage from a multi-tenant model, where the provisioning of both physical and virtual resources is dynamic, according to customers necessities and demands. Another advantage of such multi-tenant model is that the customer has no perception of the physical location of the resources, but (s)he can locate them at a higher level of abstraction, such as continent, country, or data centre.

Fourth, the resources are elastically provisioned and released. For most providers, this happens in an automatic manner, in order to satisfy the requests of the users. From the customers point of view, the provisioned resources appear to be unlimited and always available in the quantity they required.

Fifth and last in the order, the cloud resources are automatically controlled and optimised; such a monitoring capability is performed at a level of abstraction, which has to be appropriate to the type of service resources. The reason why such monitoring is performed is to attempt to guarantee some degree of transparency of the underlying infrastructure for both providers and customers.

Cloud computing architectures are delivered on three different service models; they are called Infrastructure, Platform, and Service as a Service, acronymed as IaaS, PaaS, and SaaS, respectively. The differences among them are in the type of services available on each model and the quantity of the resources under the control of customers and providers, as illustrated in Figure 2.1.

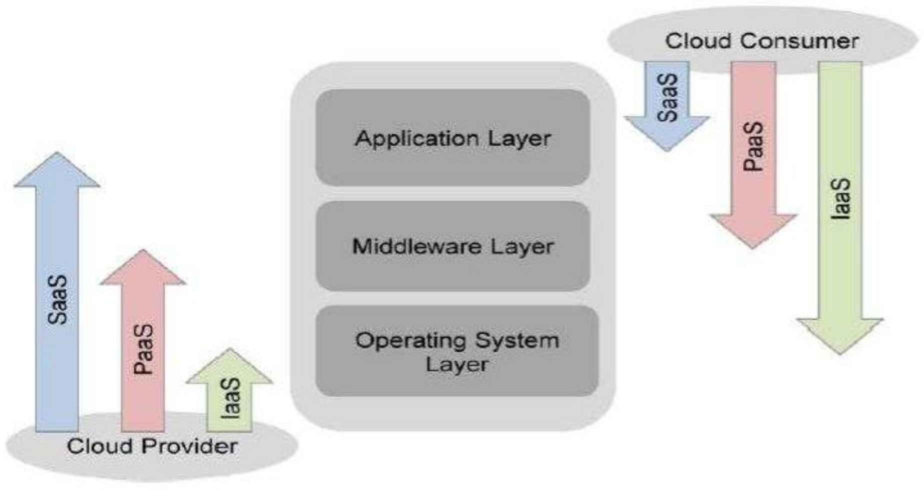


Figure 2.1: Scope of Controls between Provider and Consumer [79]

In SaaS the customers can use cloud services without managing or controlling the underlying physical infrastructure resources, with some exception related to some specific configuration settings.

Platform as a Service is where a customer can deploy his applications developed by using the technologies available in such PaaS. As well as in SaaS, at this level of the architecture the developer cannot have control of the underlying infrastructure; instead he can have control over his applications, including some configuration settings of the environment hosting them.

In an Infrastructure as a Service a customer can utilize some computing resources such as processing, storage, or networks, to cite just a few. The customer can use them to run arbitrary applications, comparable to if such resources were physically located in his office. The resource control owned by the customer in this case is about the applications deployed and running on such cloud service, plus a limited control of select networking components, such as host firewalls.

The whole cloud computing architecture infrastructure is build by using this three-layer services model, as illustrated in Figure 2.2.

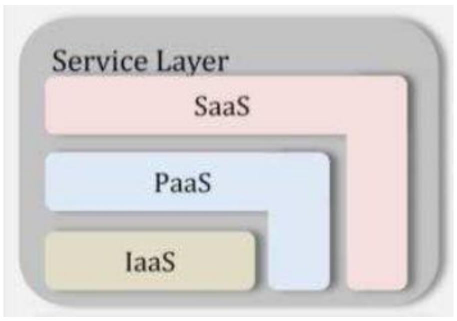


Figure 2.2: Cloud Service Orchestration [79]

The deployment models for cloud computing services are divided into four categories, namely public, private, community and hybrid. NIST provides details for each of them as follows [81].

In a public cloud computing architecture the infrastructure can be accessed and used by the general public, provided by a cloud provider which can be a business, academic, or government organisation, or some combination of them.

A private cloud infrastructure is provisioned for being used exclusively by a single organisation. Such infrastructure can be owned, managed, and operated by the same organisation that uses it, or a third party, or some combination of them.

In a community cloud model the infrastructure is dedicated to a specific community of consumers requirements. The customers belong to organisations that have shared concerns, e.g., security requirements, or policy. The resources can be owned, managed, and operated by one or more organisations in the community, a third party, or some combination of them.

Finally, in a hybrid cloud deployment model the infrastructure is a composition of two or more distinct cloud infrastructures among private, community, or public; the elements of the composition remain unique but combined by using some standard or proprietary technology enabling data and application portability in order to facilitate data transmission.

2.4 Summary

In this chapter an overview of cloud computing is provided. It includes an introduction about the history of the cloud, then the definition of such a computation delivery provided by NIST is described. Some details about definition, features, service models, and deployment models characterising cloud computing are illustrated.

In the following chapter the digital forensic science is described: an overview of such a discipline is provided, together with its definition and historical evolution. Also digital forensic readiness is introduced, together with a literature summary about forensic readiness proposals. Finally, a definition for forensic readiness is presented and discussed to close the chapter.

Chapter 3

Digital Forensics

3.1 Introduction

This chapter is dedicated to the digital forensic science. An overview of such a discipline is provided, together with its definition and historical evolution. Also digital forensic readiness is introduced, together with a literature summary about forensic readiness proposals. Finally, a definition for forensic readiness is presented and discussed to close the chapter.

3.2 Overview

During the last half century humans have been both actors and spectators of rapid changes in the whole ICT world, especially for what concerns both the devices and the manner in which computation capability is accessed. Technological progresses have been such a strong influence on all the aspects of human life, from personal to business. Indeed, analogical devices and computation are considered an ancient memory, due to the fact that everything is now digital, such as music, movies, pictures, watches, telephones, connections, to cite just a few [122]. The utilisation of the Internet is growing bigger and bigger, utilised for every type of activity, and accessed by any kind of device. At this time it is almost unthinkable being off-line.

In most cases, alongside progress, technology is a fertile ground for the increase of crimes. The ease of access to such resources is exploited by the criminals who design more sophisticated and targeted methods to hack any type of digital device, or to exploit existing computing platforms for illegal behaviour. Unfortunately, this phenomenon is becoming more commonplace, and a manner to fight it is the application of forensic science.

The forensic science concerns with applying science to law for resolving court cases. It utilizes techniques coming from several disciplines, as illustrated in Figure 3.1.

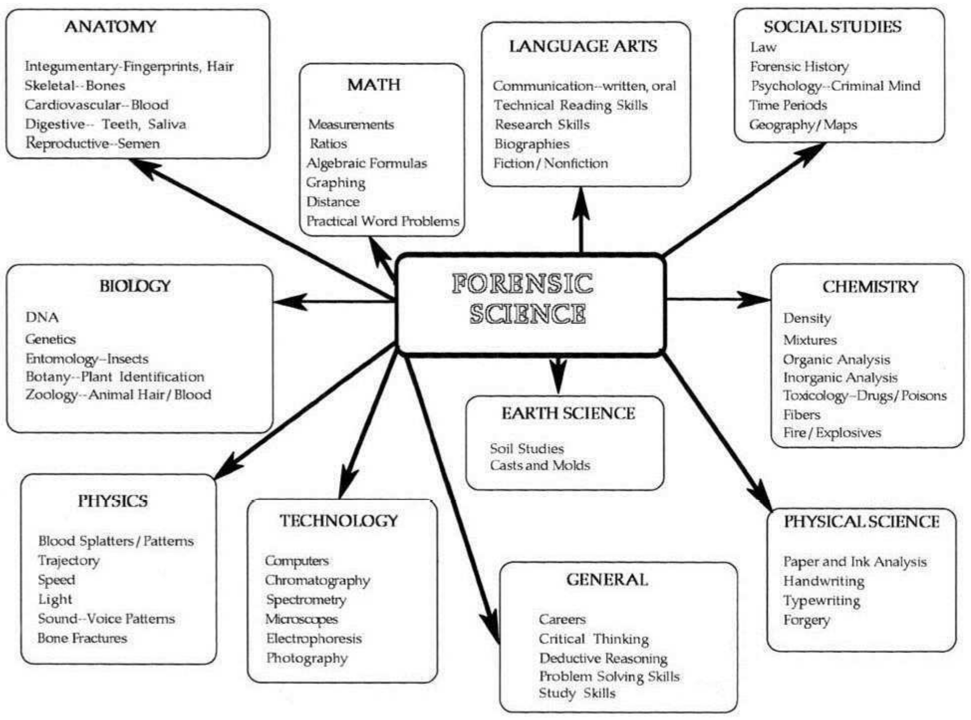


Figure 3.1: Forensic Science Map [104]

The origins of the forensic science lie several centuries B.C. [104], and along the millenniums, it followed the scientific and the technological progresses. In modern times, e.g., at the beginning of 2000s, the ICT progresses led to the modern Digital Forensic (DF) science, which deals with the investigation of legal evidence gathered from digital devices in order to resolve court cases. The place where the crime happened is called crime scene, and it is also the place where the investigation begins and where the evidence has to be collected. In DF the crime scene is digital, namely place where a digital crime happened, e.g., a device or a network in some cases.

3.3 DF Definition and Evolution

There exists no universally accepted definition for Digital Forensics, but the widest one was coined in 2001 during the first digital forensic research workshop [95]. From that definition, *digital forensics is the use of scientifically derived and proven methods toward the preservation, collection, validation, identification, analysis, interpretation, documentation and presentation of digital evidence derived from digital sources for the purpose of facilitating or furthering the reconstruction of events found to be criminal, or helping to anticipate unauthorised actions shown to be disruptive to planned operations.*

A constant issue in this discipline is the evolving nature of the investigation procedures; consequently, the forensic practitioners must deal with the ICT progresses. In single machines forensics the crime scene is well delimited and composed of single machines where the potential evidence can be located. Such devices are seized by law enforcement on the crime scene; then they remain under supervision and control of the practitioners until the case resolution.

The most common forensic procedures adopted by law enforcement for potential digital evidence are described by both UK [1] and USA [89] guidelines. For instance, the officers have to guarantee that the examined devices are collected following some specific procedures. Moreover the content of the machines might be not altered, so the investigation has to be conducted on copies; such copies have to be obtained following some guidelines.

With the diffusion and utilisation of computer networks by the general public, forensic practitioners assisted in the establishment of Network Forensics (NF) branch [95], which is the application of digital forensic procedures to computer networks. In order to be applicable, such procedures enlarged the crime scene. In addition to the computers, the devices can be included in network paths, e.g., routers, access points, switches, and server machines. ICT progress continues to pose increasingly difficult challenges to DF, which in most cases are addressed with the introduction of forensic tools and procedures customised for specific computing environments. Several tools and procedures have been established, as well as adaptations of forensic procedures to the features of computer science, e.g., for networks and mobile phones [10, 22, 24, 53, 103]. At the same time, a great deal of different investigations processes has been presented in the literature in order to satisfy some specific requirements; some of which are summarised in two literature reviews [5, 102].

More recently, cloud computing services are been being adopted predominantly in both private and business contexts. As described by Gartner [54] and depicted in Figure 3.2, the public cloud services spending is expected to record an annual growth rate of 17.7% from 2011 to 2016, equivalent to 210 billion dollars. This escalation together with its prediction can be considered as a revolutionary event for forensics, which generated the branch of cloud forensics, described later in Chapter 4.

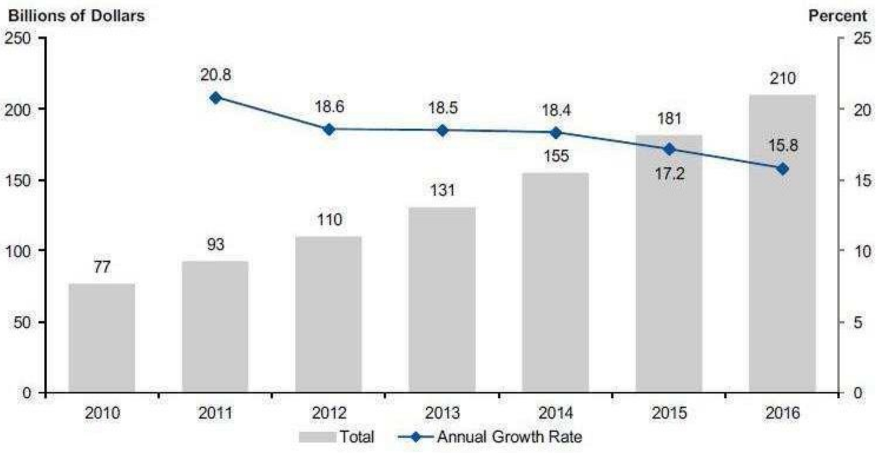


Figure 3.2: Public Cloud Service Market 2010 - 2016 [54]

3.4 Digital Forensic Readiness

In Digital Forensics some scientists proposed the idea of providing a computing infrastructure with a capability to make it ready and prepared for forensic investigations and procedures. Such a capability is called Digital Forensic Readiness (DFR), and it was introduced by Tan [123] in 2001. A definition provided by the same author affirms that DFR is *"the ability of an organisation to maximize its potential to collect digital evidence and minimizing the costs of an investigation"*.

In order to prepare computing architectures for forensics, a readiness capability must be imagined as the provisioning of an information system communicating with such architectures. The principal aim of a dedicated system is identifying, collecting, and storing critical data coming from the underlying computing infrastructure, which are the potential evidence.

A crime can happen or not; thus, the pure sense of such a capability is to render a digital context pro-active for something that can theoretically never take place. This consideration can lead the reader to doubt about the effectiveness of DFR. Specifically, if such a capability is dedicated to perform some activities whose output might never be utilised, what is the booster for spending some effort to design and implement a dedicated system is a legitimate doubt.

A positive side effect of DFR is provide an approach for addressing some issues of DF, and enhance some security and privacy issues of a computing environment. In 2004, few years after DFR was introduced, Rowlingson proposed a ten step process designed for organisations willing implement digital forensic readiness [111]. The process includes some key activities necessary for gathering potential digital evidence complying with the admissibility and reliability principles for court cases. It poses emphasis on the features that a forensic readiness system needs to be effective.

3.4.1 DFR Proposals in the Literature

An interesting approach for managing forensic readiness is discussed in [106] by Reddy et al. The examined issues deal with human, technical, and departmental management problems for implementing a DFRS in large organisations. The examination leads the authors to propose a solution composed of frameworks rather than ad-hoc systems. Such a novel architecture is provided for assisting the realisation of an optimal level for managing DFR. It is composed of detailed functional requirements determined by a literature survey, and it is also supported by an early proof-of concept prototype system to demonstrate that it is feasible.

Other work stressed the importance of a DFR capability in order to enhance the internal security of an organisation. In [59] the overlap between the DF and some Information Security (IS) best practices was examined by Grobler et al. The consideration made is that some DF aspects can be considered as IS best practices missing events prosecution procedures. These best practices excluded the requirements for the preservation of digital evidence necessary for investigations. The organisations adopting the actual best practices cannot prosecute events and information related to security controls. In the authors opinion, DFR is the solution for implementing the respect of the legal admissibility guidelines on evidence gathered during investigation procedures. A dedicated system can enrich the security strategies of an organisation; this is justified by the main feature of providing a way to prepare the existing computing infrastructure for incident handling by collecting potential digital evidence. Thus, DFR is a good candidate to become a component of the IS best practices, demonstrating that protecting valuable company information resources is critical.

In [46], Endicott et al. discussed several Network Forensics aspects. The authors analysed some situations when cyber-targets are powerless with respect to attackers and intruders exploiting and disrupting the networks. The authors affirmed that forensic readiness for network infrastructures can be a valuable solution in order to decrease the power of such attacks. Thus, a theoretical framework to implement network forensic readiness in enterprise contexts is proposed.

In [35] the availability of digital evidence is discussed by Danielsson et al. It must be collected in a proper and pro-active manner in order to render the investigations effective and successful. For this purpose, a DFR capability must be implemented into an organisation, and it must follow a structured approach. Its implementation includes several features regarding national and international legislation, together with their constraints and requirements about data collection and preservation, and user data privacy protection. Such an approach aims to pro-actively seek the sources of digital evidence, and to configure the existing computing infrastructure for collecting and preserving the potential evidence. The proposal takes into account relevant and established standards and best practices, nevertheless it considers some already existing organisational routines, such as some data collection operations performed for purposes different from forensics, which can record some events related to potential crimes. Finally, it provides guidance for reporting the incidents to law enforcement, including the content, the format, the criteria for the report itself and the manner in which the interaction between the law enforcement and the affected parties is regulated.

Again, the impact of DFR on a corporate context was analysed by Pangalos et al. [96], where some positive aspects were highlighted, e.g., the enhancement of the security strategy of an organisation, the reduction of security incidents, the availability of evidence, and the derived effectiveness of an investigation.

Mouton et al. discussed another proposal for implementing a forensic readiness capability concerns Wireless Sensors Networks [84]. A dedicated prototype was designed as an additional layer to the existing infrastructure in order to not modify the original architecture of an existing IEEE 802.15.4 network. The prototype is designed according to a list of requirements that have not been tested in real wireless sensor network scenarios. Thus, the requirements usability has been tested through a prototype implemented as an additional layer of the network architecture.

A DFR capability is considered crucial and necessary to be provided also to cloud computing architectures through a dedicated system [115] by Ruan et al. It is responsible to perform pro-active forensic investigations activities; it offers some positive side effects, such as increased security and control on data access.

In [126] a forensic readiness capability usage was discussed for Public Key Infrastructure (PKI) by Valjarevic et al. A PKI system is a set of hardware, software, people, policies, and procedures necessary to create, manage, store, distribute, and revoke digital certificates. These systems are used to implement information system security services such as authentication and confidentiality. The authors investigated a set of policies, guidelines, and procedures, together with a model for implementing a forensic readiness framework for such systems. Some requirements for either preserving or improving information security and at the same time not altering the existing business processes of such PKI systems is the analysed and addressed issue.

3.4.2 A Definition for DFR

In this section one of the contributions of this dissertation is presented. It concerns a definition for digital forensic readiness capability. The approach to design such a definition has considered the related literature discussed in Section 3.4.1; an initial version of [36] has been recovered and refined as it follows:.

A digital forensic readiness capability is shaped as an information system communicating with a computing architecture. The main aim is collecting and monitoring sensitive and critical information potentially related to digital crimes before they happen, leading to save time and money for the investigations. Data are closely related to the system artefacts and logging tools available at the moment. The collected data should be encrypted in order to guarantee more protection, and stored on a place accessible by selected subjects.

The reason to shape a capability with an information system is driven by the necessity to represent something abstract, as a capability is, with something else which can be seen, as an information system is [119]. Nevertheless, with the execution of a process some output can be generated; they are related to proactive forensic tasks aimed to prepare an infrastructure for possible investigations. Such output are the mere operations of the capability itself, namely they compute some input data collected from the monitored infrastructure. This information is composed of log files and additional system artefacts related to them, because they can hide facts related to digital crimes, or close to their happening. The relation of this data to crimes will be defined in the following.

The reason why data have to be encrypted and stored into an place different from where it is gathered relies in some forensic best practices described in [1, 89], which provide details about evidence admissibility principles, among other procedures.

Finally, as mentioned in [36] the provided definition for digital forensic readiness is considered general and adaptable to every computing infrastructure, hence valid both for the past and the future, as well as for the mentioned cloud infrastructures.

3.5 Summary

In this chapter an overview of digital forensic is provided. A broad description of such a discipline begins the chapter, together with its definition and historical evolution. Also digital forensic readiness is introduced and described, together with a literature summary about forensic readiness proposals. Finally, a definition for forensic readiness is presented and discussed to close the chapter.

In the following chapter another aspect of digital forensics is examined, namely the adaptation of forensic to cloud computing. An overview of Cloud forensics is provided, then the main challenges derived by the literature are illustrated. Forensic readiness in the cloud is then described, and a reference architecture for a cloud forensic readiness system is designed and explained.

Chapter 4

Cloud Forensics

4.1 Introduction

This chapter is dedicated to the adaptation of forensic to cloud computing. An overview of Cloud forensics is provided, together with its three dimension model: technical, organisational, and legal. The main challenges of each dimension are derived by the literature and illustrated. Forensic readiness in the cloud is then described, and a reference architecture for a cloud forensic readiness system is designed and explained.

4.2 Overview

Cloud Forensics (CF) [112] deals with the management of crimes committed both to hack cloud platforms and that use the cloud as means to commit crimes. Cloud architecture novelties lead the forensic practitioners to deal with a computing infrastructure that cannot be investigated with the available forensic tools and procedures. Nevertheless, the practitioners are required to manage cloud evidence respecting the admissibility and reliability principles for digital evidence [24].

Some cloud features have been used to build a cube model for cloud forensics [112], which is composed of technical, organisational, and legal dimensions, as depicted in Figure 4.1.

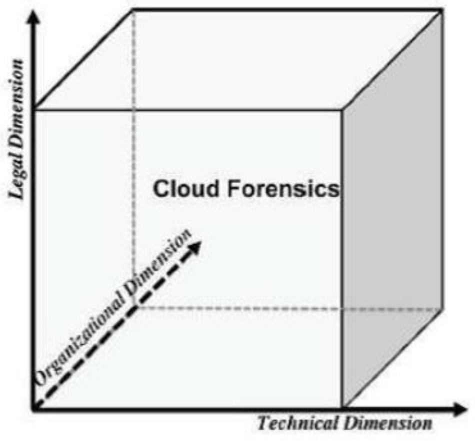


Figure 4.1: Cloud Forensics 3-Dimensions Model [112]

The technical dimension deals with tools and procedures for performing forensic investigations; the organisational one concerns with the manner of establishing a forensic capability, e.g., what are the roles and the responsibilities to assign into a cloud organisation; finally, the legal dimension covers issues about multi-jurisdiction, multi-tenancy, and Service Level Agreement (SLA) policies.

4.3 Cloud Forensic Challenges

Cloud architectures generate some challenges [15, 107, 112, 115, 131] structured according to the cube model depicted in Figure 4.1. A representation relating forensic challenges to the cloud features determining them is represented in Table 4.1.

4.3.1 Technical

Cloud services are elastic, meaning that they are provisioned and released responding to users scaling demands. The services run on an infrastructure composed of multiple machines located potentially in different geographical zones without precise routing information; the resources are virtualised by using some virtual machines (VMs) [81]. From a forensic perspective, these features determine a reduced access to data, because the providers intentionally hide data location to facilitate ubiquitous access and replicas. Furthermore, the physical control of the architecture components is lacking; it varies for the three services models, as depicted in Figure 2.1, becoming larger when a customer moves to the bottom of the architecture. Another issue in cloud architectures concerns the heterogeneity of the log files: because there is no standard for the formats, each provider can customize its own log type. Nevertheless, there is no timestamps synchronisation among several data centres and server machines under a single provider scope, as well as among different providers components.

4.3.2 Organisational

Conducting a forensic investigation in the cloud might involve data and services information belonging to both providers and customers. There might be also situations where the cloud providers out-source some services from third parties, thus the scope of an investigation becomes wider. Moreover, such out-sourced services can be based on a cloud architecture, hence all the issues related to the replication of data on multiple data centres located potentially under different physical jurisdictions escalates. The lack of legal expertise specific for these features determines that there is big uncertainty about the measures to undertake in case of cross-providers or third parties resources supplying.

4.3.3 Legal

Cloud physical resources are virtualised to be used by multiple consumers via a multi-tenant model; they are also dynamically assigned according to the demands. The principal issue is the trade-off between multi-tenancies and tenants data privacy, i.e., what is the correct trade-off to guarantee multi tenancy and at the same time preserve tenants data privacy. Another side effect of the on-demand elasticity is the spread of customers and providers data under different jurisdictions; in most cases also the SLAs do not include information about the manner for determining data ownership or what can be the jurisdiction to consider, namely whether the one related to the physical location of the customer, or to providers machines and which provider; in this case the contracts might be tailored to include proper constraints. Few proposals exist in the literature discussing and addressing this issue [47, 94, 129].

Table 4.1: Cloud Forensics Main Challenges [39]

CF Challenges	**Elasticity**	**Multiple Locations**	**VM**	**Broad Network Access**	**Third Party Service**	**Cross - Providers**	**SLA**
Reduced data access	X	X	X				
Lack of physical control	X	X	X				
Lack of standard	X	X	X				
Multiple log formats	X		X				
No timestamps synchronisation		X		X			
No routing information		X	X	X			
Lack of investigation expertise		X				X	
Inappropriate legal measures		X			X	X	X

CF Challenges	Elasticity	Multiple Locations	VM	Broad Network Access	Third Party Service	Cross - Providers	SLA
Multi - tenancy	X						X
Multiple jurisdiction		X					X

4.4 Cloud Forensic Readiness

A forensic readiness capability must be provided to cloud computing architectures because, due to their escalating popularity (see Figure 3.2) they can be object of several attacks; thus, a way to conduct forensic investigations effectively, e.g., saving time, money, and resources, must be designed.

A result of a recent survey [113] conducted by Ruan et al. can corroborate this needs; indeed, almost 90% of the interviewees familiar with digital forensics, stated that "a procedure and a set of tool-kits to pro-actively collect forensic-relevant data in the cloud is important".

Dykstra et al. in [40] analysed some existing forensic tools like EnCase in a cloud context; the result confirmed that the data collected by those tools are unreliable, because some cloud features require more effort for performing forensics than simply tailoring the existing tools and procedures. The reason for this need is due to the fact that new technical requirements must be managed for complying with the legal principles required for digital evidence.

The same authors proposed a proper remote forensic acquisition suite of tools for an open-source cloud environment [41]. This suite named FROST provides a forensic capability into the IaaS level of OpenStack, an open-source Cloud Computing platform. FROST performs data collection from provider machines and from the host operating system, and renders the data available to the users, because it is assumed that the customers are cooperative during the investigations. The data collected in FROST include virtual machines images, logs coming from the API requests, and the OpenStack firewall logs. This suite is considered by the authors as a way for enhancing forensic readiness into the cloud because it performs the necessary investigation preparation activities, such as data collection.

Also in [124] a manner for achieving digital forensic readiness in the cloud is described by Trenwith et al. It is composed of a remote and central logging facility for accelerating the acquisition of data; the model was also prototyped for Windows platforms.

4.5 Reference Architecture for Cloud Forensic Readiness System

One of the contribution of this doctoral dissertation is represented by a reference architecture for a cloud forensic readiness system (CFRS) [36]. The approach undertaken to conceive such an architecture was based on an examination of the most common cloud technical features, as described in Section 2.3, in order to understand the potential sources of digital evidence among artefacts and managing tools included in a cloud infrastructure.

Such a reference architecture provides a general approach and understanding about the necessary operations that a CFRS must perform at a high level of abstraction. Its main advantage resides in its design, which is not constrained by any specific and / or technical configuration; rather, it is flexible and customizable, and it can be considered as a template for most organisations and cloud service providers who will implement a forensic readiness capability.

The forensic readiness system is designed to communicate with an existing cloud infrastructure without altering the original components (see Fig. 4.2). It includes two main sub-systems; the former is called forensic database, dedicated to the collection of some cloud services information, namely the potential evidence. Such data are classified depending on the type; the possible types are monitored

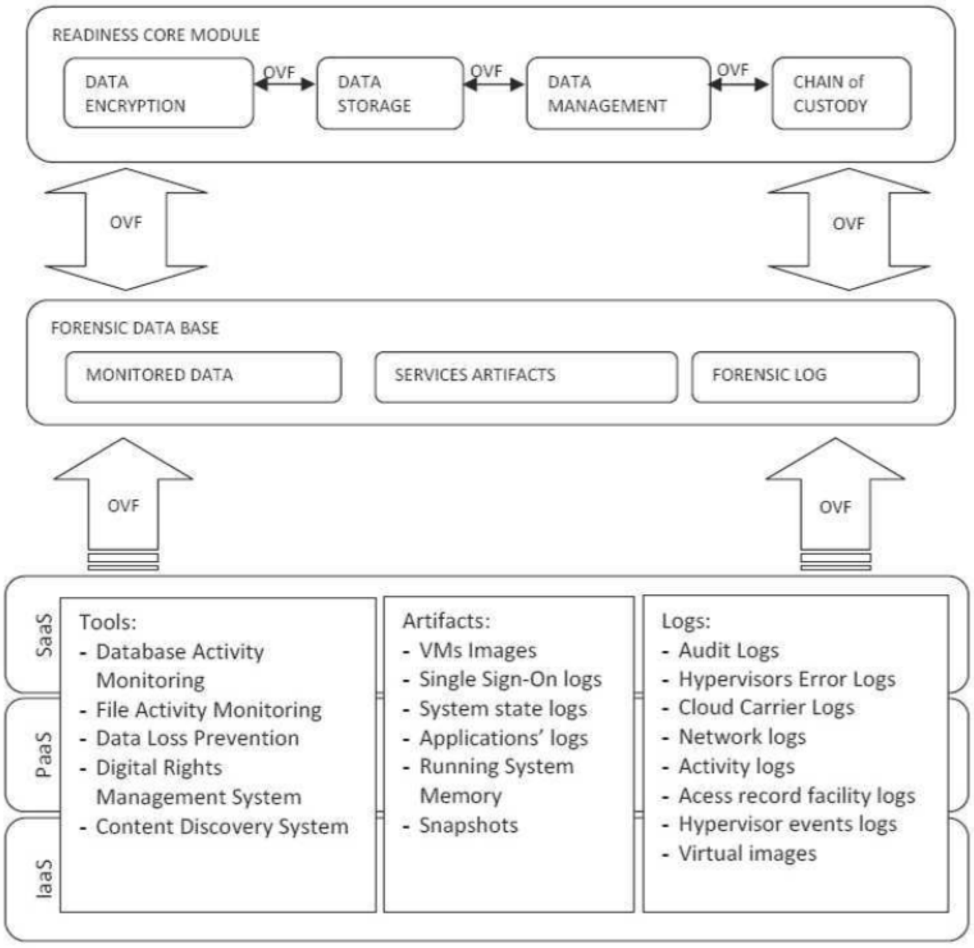


Figure 4.2: Reference Architecture for a Cloud Forensic Readiness System [36]

data, services artefacts, and forensic log; dedicated sub-systems are included in the architecture design.

The monitored data sub-system refers to information coming from cloud facilities dedicated to data monitoring and control [29] e.g., database and file activity monitoring, URL filtering, data loss prevention, digital rights management system, and content discovery system. The database and file activity monitoring tools are capable of recognizing whenever a huge amount of data is pushed into the cloud or replicated, thus indicating a data migration. The data loss prevention facility is used for monitoring data in motion; it also manages policies and rights. URL filtering controls the customers connections to the cloud services, thus it can be used during the reconstruction of a case time-line. The digital rights management system implements and monitors customers rights and restrictions on data, as stated by the SLAs and terms of use contractual clauses co-signed by providers and customers; the content discovery system includes tools and processes aimed to identify sensitive information in storage components of a cloud architecture, hence their output can allow to identify some data violations or misuses.

The forensics artefacts sub-system is dedicated to the storage of a significant quantity of artefacts gathered from the provider side, i.e., from the SaaS, the VM images and the Single Sign-On logs; from the PaaS, the system states and applications logs; and from the IaaS, the snapshots and the running system memory.

Cloud auditors logs and error logs coming from the virtual machines hypervisors are instead collected by the forensic log; both of them are relevant for incident response procedures and crime investigations. Also some information from the cloud carrier has to be considered in the forensic log module. A cloud carrier is an intermediate between customers and providers, responsible for providing connectivity and transport of the services to the customers through the network and other access devices [114]. Therefore, some information suitable for forensic investigations include network logs, activity logs, access record facility logs, hyper-visor event logs and virtual images.

The second main component of the cloud forensic readiness system is the readiness core module, which performs different activities on the gathered data, executed by dedicated sub-systems. The collected data are encrypted and stored by dedicated sub-components, i.e., data encryption and data storage, respectively. The data management sub-system performs forensic analysis and knowledge extraction with the purpose of reconstructing a correct and reliable event time-line about the recorded information. Finally, the chain of custody report necessary for cases resolution [89] is performed by the chain of custody sub-system.

A communication and data exchange channel is necessary between the cloud infrastructure and the forensic readiness system, and also among the several sub-systems. For this purpose, the Open Virtualisation Format (OVF) standard language [93] is considered suitable for the design and the distribution of the system. This standard language is capable of creating and distributing software applications to be executed on different VMs, independently from the hyper-visors and from the CPUs architectures. Moreover, it exploits the XML standard to establish the configuration and the installation parameters; it can be extended for future VM hyper-visors developments, thus considered extremely flexible and adaptable for future versions of a forensic readiness system. In the reference architecture depicted in Figure 4.2 the OVF communication channel between the cloud and the system can be used to convert some data formats into a specific target one, in order to render the necessary information readable and usable by the system itself.

4.5.1 Operations in a CFRS

The initial activity of a cloud forensic readiness system is data collection. The valuable forensic data are the ones represented in the bottom boxes of Figure 4.2. They include cloud services artefacts and output from some existing cloud monitoring tools [29].

Data are gathered from the cloud and manipulated outside the architecture. In order to accomplish the UK [1] and U.S. [89] guidelines concerning the preservation of the potential digital evidence, the collected data have to be copied and secured to avoid tampering. This is performed by dedicated data storage and encryption sub-systems (see Figure 4.3), where proper digital sign and data securing routines are implemented. This step is necessary for preserving the original copies

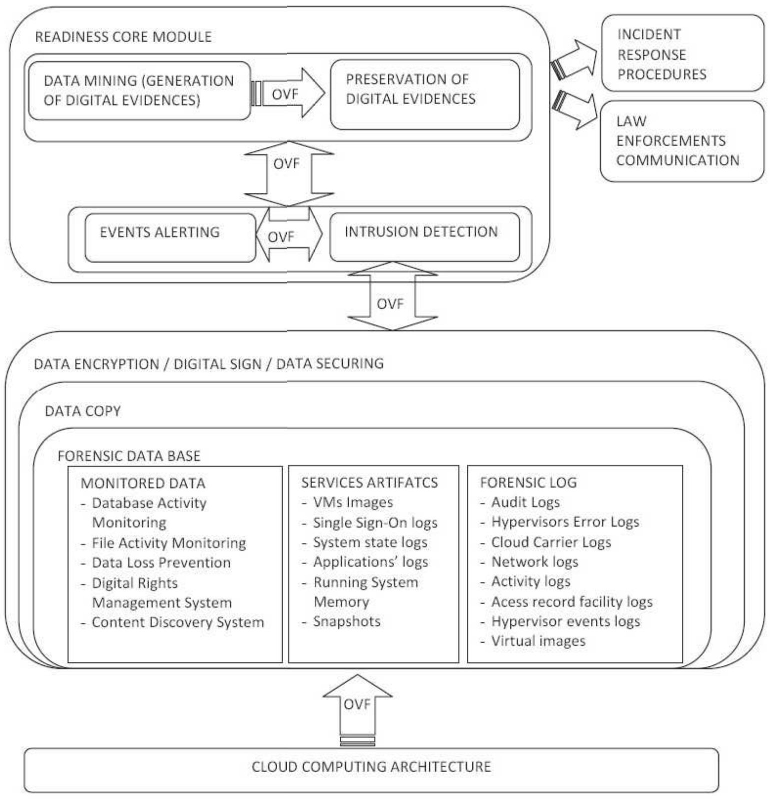


Figure 4.3: Reference Architecture for a Cloud Forensic Readiness System
A Different View [37]

when forensic activities are performed.

The whole system activities and modules are constantly running and collecting the most up-to-date data. All this information is fed to the intrusion detection sub-system, responsible for relating the available information, in order to detect suspicious behaviours. This sub-system has to consider the co-signed SLAs clauses [81] necessary for correctly detecting contractual violations. The intrusion detection system component communicates with the events alerting one, as it generates alarms as soon as suspicious behaviours and contractual violations are detected. Such alarms might be different depending on the type of events, but this is out of the scope of a forensic readiness capability.

The data mining module of Figure 4.3, which is the data management of Figure 4.2, is responsible for hidden knowledge extraction, necessary to generate the incident-related evidence, and to relate the data and the sequence of events happened and happening, leading to construct a correct and reliable time-line.

The evidence must be treated considering guidelines, best practices, and laws used in court admissibility for cases prosecutions. For this purpose, proper and dedicated policies and routines are implemented in the preservation of digital evidence module. Some information related to the them, e.g., location, treatment, date, time, time zone, or system component, have to be recorded, in order to maintain a reliable chain of custody necessary for prosecution purposes, which is performed by the Chain of Custody sub-system depicted in Figure 4.2.

The proposed cloud forensic readiness system has to be communicative with the possible competent bodies involved in the criminal cases management, which can mean transmitting the necessary information related to the detected case, such as a contractual violation. The competent bodies can be private or public incident responses; thus dedicated communication interfaces with their information systems can become necessary, as well as depicted in Figure 4.3, where the competent bodies are incident responses and law enforcement.

4.5.2 Forensic Readiness System Constraints

In order to obtain the most from the proposed FR system, some constraints must be verified. Initially, the cloud infrastructure to be furnished with such a capability must provide the necessary monitoring tools to gather the data from, considered common components to most cloud providers [29], therefore their presence should be verified. Summarizing, they are listed as:

- components dedicated to the monitoring of both databases and files, necessary for detecting data migrations;

- tools for filtering URLs, aimed to verify the connections made by different IP addresses;
- tools with the purpose of controlling policies and rights established by the co-signed contracts.

Big importance is assigned also to the potential evidence data sources. This encompasses several logs generated by appropriate logging facilities present in cloud architectures, as well as system images gathered through dedicated tools.

Another requirement concerns the capability of installing the necessary OVF communication channels, responsible for data transmission. From both an organisational and legal perspective, these communication channels should require authorisation for data exchange. In this manner, the involved cloud actors will be warned, and eventual privacy violation threats can be managed.

4.5.3 CFRS Advantages

The implementation and the usage of a forensic readiness system for cloud computing architectures are very important for multiple purposes. Implicitly, the first aim is rendering a cloud infrastructure ready for digital forensics by executing the operations described in Section 4.5.

One of the system side effects is the enhancement of cloud customers data privacy, together with major internal security; this can happen because a wider control and monitoring will be performed by the system itself for protecting critical and sensitive information [36]. Nevertheless, the reconstruction of cases time-line accompanied by a related chain of custody document [89] is the biggest contribution of such added value.

A cloud organisation or provider might realize that being prepared for managing crimes or incidents can be vital for both the reliability and the reputation of the offered services; indeed a pro-active gathering of digital evidence minimizes the impact of a forensic investigation on a cloud organisation routines and performances [111, 123].

The detection of Service Level Agreement clauses violations can be managed by a CFRS, as it will be discussed in this dissertation. Indeed, the sub-system dedicated to the events reconstruction, see Figure 4.2, can be capable of determining a source of attack, or the exact time when a customer data violation happened, relating them to some specific service levels described in an SLA.

The implementation of a CFRS can be helpful also to address some cloud challenges for forensics described in Section 4.3. A manner for aligning multiple log files formats and for synchronizing several machines timestamps can be implemented; such a solution can lead to a correct and reliable reconstruction of events time-lines, not necessarily related to the crimes, but also to some processes executions.

From a cloud forensic organisational perspective, the usage of such a system can be the means to assigning roles and responsibilities necessary for managing cloud incidents, such as investigator or incident handler; indeed, the people trained for the system can be responsible for some system modules and the manner in which data are managed in the cloud organisation.

Finally, from a legal point of view, a CFRS can highlight the main issues regarding the jurisdiction borders; it can become the instrument for alerting proper governmental institutions, helping to address a more general problem.

4.6 Summary

In this chapter an overview of Cloud forensics is provided, together with its three dimension model: technical, organisational, and legal. The main challenges of each dimension are derived by the literature and illustrated . Forensic readiness in the cloud is then described, and a reference architecture for a cloud forensic readiness system is designed and explained.

In the following chapter forensic readiness in digital investigation processes is examined by performing a Systematic Literature Review. The phases composing such an investigation are detailed in specific sections, and some conclusions are reported at the end of the chapter.

Chapter 5

Forensic Readiness in Investigation Processes

5.1 Introduction

This chapter is dedicated to an examination of forensic readiness in the context of digital investigations. They are initially defined and described, and then the presence of forensic readiness in the literature involving them is investigated. For this aim a Systematic Literature Review is performed, and the phases composing it are detailed in specific sections.

5.2 Digital Forensic Investigation

A Digital Forensic Investigation (DFI) is defined as a process of collecting, identifying, preserving, analysing, and presenting digital evidence in a manner that is legally acceptable [80]. The most known and used DFI definitions have been proposed by the National Institute of Standards (NIST) [66] and the U.S. National Institute of Justice (NIJ) [90], as depicted in Figure 5.1.

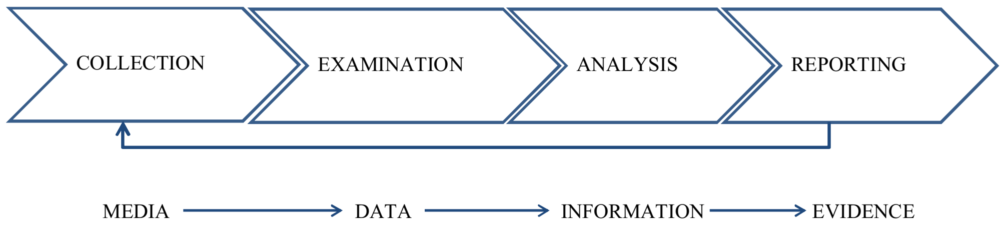


Figure 5.1: NIST DFI Process Definition [66]

Since a standard for DFI processes does not exist, several models were proposed in the literature during the last few years [5, 6, 12, 19, 71, 95, 102, 105, 108, 128]. Each of them represents a digital investigation process decomposed in phases and sub-phases when necessary. Sometimes, the DFI is customised for specific computing architectures [10,24,53,95,103]. In this chapter a rigorous systematic literature review process has been performed in order to determine when forensic readiness has been considered by the existing DFI processes proposed by the scientific community.

5.3 Systematic Literature Review

The Systematic Literature Review (SLR) method was introduced in Medicine by Archie Cochrane in 1972 from the necessity of identifying effective evaluations to guide research investments and health care provisioning [32]. The main aspect leading to the formulation of such a research method was the lack at that time of an organised and critical summary of medical trials.

The SLR process is composed of three different phases: planning, conducting, and reporting. In the planning phase, the first step is the formulation of some research questions; they will be answered at the end of the review process. Subsequently, a search string to feed some repositories must be composed, in order to gather initial results. Some inclusion and exclusion criteria are defined, together with a description of the quality criteria, applied in the following phases.

During the conducting phase all the operations involve the retrieved papers. Once the literature is gathered, the inclusion and exclusion criteria are applied in order to determine if the sought papers can be considered as primary studies. As a secondary step, the quality assessment is performed through the utilisation of the quality criteria, defined in the planning phase. From the selected studies, some data extraction, monitoring and synthesis are executed.

The final reporting phase is dedicated to the composition of the review report, containing the final results and the answers to the research questions.

This method is defined as systematic, due to some activities which have less biases than a simple literature review. Such activities are even performed by humans, i.e., researchers, once they are stabilised the repetition of the whole process will provide the same results. A search string is defined in a systematic manner, together with the research method, the inclusion and exclusion criteria, and the manner for assessing the sought literature collection.

One of the principal disadvantages of adopting an SLR is its duration, which is considered a very time-consuming process. On the other hand, there are many advantages in adopting it, which lead to its consideration for this study. For instance, and SLR provides a decomposition of big information in more accessible small pieces, together with a minimisation of the biases. Moreover, it facilitates use of the collected information by practitioners, decision makers, and researchers. Its main application reasons reside in its capability of locating previous studies in a certain area, and of finding effective research methodologies, and of verifying whether a topic was investigated before, thus becoming a more critical, comprehensive and systematic evaluation method for previous studies.

Some SLR process variations have been proposed for disciplines other than

Medicine, such as the SLR guidelines for Software Engineering [69]. They are effective for conducting a structured, repeatable, and scientific state of art or literature review about a specific research topic. Such guidelines have been widely applied in Software Engineering [70] and also in other computer science branches, like digital forensics. Indeed in [5] an SLR about some existing DFI models has been conducted. As result, the authors found a model suitable for their purpose; thereby they derived a novel model for their needs.

To the best of the knowledge, this is the unique SLR conducted in digital forensics, but other non-systematic literature review about the forensic process models have been published. For instance, in [102] fifteen published papers about DFIs have been collected in a non-systematic and non-exhaustive review, as affirmed by the authors.

5.4 Digital Forensic Process Models SLR

In this section the SLR guidelines for software engineering [69] are applied on a specific topic in digital forensics, namely forensic investigations, with the aim of providing an understanding the evolution of the published work, and the detection of a forensic readiness capability among the collected results.

5.4.1 Planning Phase

The planning phase is composed of the following sub-phases. The first step is the specification of the research questions (RQs). They will be answered at the end of the whole SLR process, i.e., in the report phase in Section 5.4.4. According to [69], the RQs must respect the Population, Intervention, Comparison, Outcome, Context (PICOC) guidelines:

- *Population:* models for the DFI;

- *Intervention:* how the DFI models are presented (e.g., involved actors; computing environment);
- *Comparison:* works that report only the definition of the DFI without modelling, or that focus only on a specific phase, or on other sub-branches of the digital forensic science;
- *Outcome:* different models for the DFI, involving different phases, or structured in several manners, or built for different computing environment;
- *Context:* academic context; the SLR is undertaken by a PhD student, without any external contribution.

The Research Questions are defined as it follows:

1. How many DFI models have been proposed in the literature?
2. Which types of actors are involved?
3. Do the models comprise taxonomy of incidents?
4. What type of computing environment is considered by the various models?
5. What are the common features among the models?

The investigated literature repositories are: ACM Digital Library, IEEE Xplore Digital Library, Citeseer, SpringerLink, and SCOPUS. In the sections "title", "abstract" and "keywords" of the mentioned repositories, the following search string has been used: *(computer OR digital) AND forensic AND process AND model.* No publication year range was applied to the search form: the investigated topic is very recent, but in order to corroborate it, that parameter was left open to ensure to include also previous works, if any.

The Inclusion / Exclusion criteria are formulated as it follows:

Inclusion: the studies proposing new models for the DFI process must be included; these models must be composed of at least four phases, because the definition of the DFI process includes four necessary phases (see Figure 5.1).

Exclusion: the studies not written in English must be excluded; also studies that do not present any model or present only the definition of the DFI process will be excluded. Moreover, the studies composed from one to three phases will be excluded; also the studies that model only a specific phase. Finally, all the studies focusing on specific sub-branches of the forensic science, e.g., digital image or audio forensics, will be not included in the report.

Each examined study is assessed by quality criteria are defined to evaluate the importance and the impact of each model in the academic world, in such a way that the included studies can be considered as a basis for developing innovative DFIs. The used quality criteria are:

1. What is the international impact of the model?
   (a) Is the model cited and used by other models in the result?
   (b) Is the model exploited in the business context, i.e., security agencies?
2. What is the venue and where the model was presented?
   (a) Was it published in International Journal?
   (b) Was it presented at an International Conference?

Once the proposed models are collected, they will be analysed to extract several characteristics, such as phases and sub-phases, the involved actors, the computing environment, the year of publication, and so on. Finally, the resulting literature review will be summarised to outline what are the commonalities among the DFIs in order to answer the aforementioned research questions.

5.4.2 Conduction Phase

The SLR was conducted during the last six months; it used the search string described in Section 5.4.1 to feed to the aforementioned digital repositories, and the number of resulting studies is shown in Table 5.1. The primary studies have been gathered by applying the Inclusion / Exclusion criteria to the abstract, introduction, and conclusions sections of the sought documents. A summary of this phase is provided in the last five rows of Table 5.1 with the total number of sought studies, the number of the duplicated results, and the number of included and excluded studies. In Table 5.2, a list of the 33 included studies is presented.

Table 5.1: SLR Results per Repository

ACM Digital Library	65
IEEE Xplore	122
Citeseer	79
Scopus	250
Springer Link	1
TOT Studies	**517**
TOT Duplicated Studies	236
Actual Examined Studies	281
Actual Excluded Studies	248
Actual Included Studies	**33**

Table 5.2: Details on the 33 results of the SLR

ID	**Title**	**Year**
01	A Hypothesis-Based Approach to Digital Forensic Investigations [21]	2006
02	Artificial Intelligence Based Model for Incident Response [61]	2011
03	Computer Forensics Guidance Model with Cases Study [92]	2011
04	How to Find Exculpatory and Inculpatory Evidence Using a Circular Digital Forensics Process Model [68]	2008

ID	Title	Year
05	Two-Dimensional Evidence Reliability Amplification Process Model for Digital Forensics [67]	2008
06	Network Forensic Frameworks: Survey and Research Challenges [101]	2010
07	Convergence of Digital and Traditional Forensic Disciplines: a First Exemplary Study for Digital Dactyloscopy [62]	2011
08	Harmonised Digital Forensic Investigation Process Model [127]	2012
09	VoIP Evidence Model: A New Forensic Method for Investigating VoIP Malicious Attacks [74]	2012
10	Digital Forensic Readiness in the Cloud [124]	2013
11	A System for Formal Digital Forensic Investigation Aware of Anti-Forensic Attacks [109]	2012
12	A Common Process Model for Incident Response and Computer Forensics [52]	2007
13	Computer Forensics Field Triage Process Model [110]	2006
14	The Federal Court, the Music Industry and the Universities: Lessons for Forensic Computing Specialists [17]	2003
15	MFP: The Mobile Forensics Platform [2]	2003
16	Systematic Digital Forensic Investigation Model [6]	2011
17	The Enhanced Digital Investigation Process [12]	2004
18	Common Phases of Computer Forensics Investigation Models [128]	2011
19	Digital Forensic Model Based On Malaysian Investigation Process [99]	2009
20	UML Modelling of Digital Forensic Process Models [71]	2008
21	An Event-Based Digital Forensic Investigation Framework [20]	2004
22	A New Approach of Digital Forensic Model for Digital Forensic Investigation [3]	2011
23	Identification of User Ownership in Digital Forensic using Data Mining Technique [73]	2012
24	Network Forensics for Cloud Computing [55]	2013
25	Applying a Stepwise Forensic Approach to Incident Response and Computer Usage Analysis [78]	2009
26	Taxonomy of Computer Forensics Methodologies and Procedures for Digital Evidence Seizure [117]	2006
27	Integrated Digital Forensic Process Model [72]	2013
28	An Integrated Data-Flow Based Model for Digital Investigation [27]	2009
29	Honing Digital Forensic Processes [25]	2013
30	Modelling the Forensics Process [4]	2012
31	The Modelling of a Digital Forensic Readiness Approach for Wireless Local Area Networks [91]	2012
32	A Multi-Component View of Digital Forensics [60]	2010
33	Analysis and Correlation [121]	2002

5.4.3 An additional Conduction Phase

An additional conduction phase has been performed with the necessity of covering some missing existing DFIs. The literature referenced by the resulting thirty-three papers has been examined, and the same inclusion, exclusion and quality criteria have been applied on it. It is worth noting that in this manner some additional DFIs have been gathered; they did not appear in the initial thirty-three publications probably because the adopted keywords did not match with them. The list of these additional papers is presented in Table 5.3.

Table 5.3: DFIs common to most of the 33 SLR results

ID	Title	Year	Cited By
A1	A Road Map for Digital Forensic Research Technical Report [95]	2001	01, 04, 05, 06, 08, 10, 16, 17, 18, 20, 21, 22, 23, 25, 27, 28, 30, A3
A2	Electronic Crime Scene Investigation A Guide for First Responders [90]	2001	01, 03, 08, 12, 16, 17, 20, 21, 28, 30, 31, A3
A3	An Examination of Digital Forensics Models [108]	2002	01, 05, 06, 08, 13, 16, 17, 18, 19, 20, 21, 22, 28, 30
A4	Digital Evidence and Computer Crime [23]	2004	01, 03, 06, 12, 13, 19, 20, 27, 28, 32

5.4.4 Reporting Phase

In this section, the SLR outcome is examined in order to provide answers to the research questions presented in Section 5.4.1.

Data Extraction

Table 5.4 shows the features extracted from the thirty-seven collected studies. Such features include the used modelling diagram, the names of the principal phases, the total number of phases and sub-phases, the involved actors, and the computing environment they were designed for, where included.

Table 5.4: Features of the 37 Forensic Process Models

ID	Diagram Type	Phases	# of phases; sub-phases	Actors	Computing Environment
01	finite state machine	observation; hypothesis formulation; prediction; testing and searching	4		

ID	Diagram Type	Phases	# of phases; sub-phases	Actors	Computing Environment
02	flow chart	gather information; database; searching and identification; sorting; analysis; result	6	Artificial Intelligence System	
03	flow chart	preparation; physical forensics and investigation; digital forensics; reporting and presentation; closure	5; 33		
04	circle diagram	initialisation; evidence collection; evidence examination & analysis; presentation; case termination	5; 15	Inspector ; Manager; Investigators Team	
05	"Umbrella" diagram	initialisation; evidence collection; evidence examination & analysis; presentation; case termination	5; 16	Inspector; Manager	
06	flow chart	preparation; detection; incident response; collection; preservation; examination; analysis; investigation; presentation	9		Computer Networks
07	flow chart	strategic preparation; physical acquisition; operational preparation; data gathering; data investigation; data analysis; documentation	7; 6		
08	flow chart	incident detection, first response, planning, preparation, incident scene documentation, potential evidence identification, potential evidence collection, potential evidence transportation, potential evidence storage, potential evidence analysis, presentation and conclusion	12; 7; 6 (parallel actions)	Judge, Jury, Accused, Lawyers, Prosecutors; Company Management Team, Shareholders and Involved Employees	
09		terminal state/available evidence; information gathering; evidence generation; print generated evidence	4; 5	Investigators	VOIP
10		logs identification; collection; compression; encryption; transportation; storage; examination	7		Cloud Computing
11	flow chart	preparation; collection & preservation; analysis of anti-investigation attacks; analysis of regular attacks; presentation & reporting	5; 3		Computer Networks

ID	Diagram Type	Phases	# of phases; sub-phases	Actors	Computing Environment
12	flow chart	pre incident preparation; pre analysis phase; analysis phase; post analysis phase	4; 11		
13	workflow	planning; triage; usage/user profiles; chronology/timeline; Internet activity; case specific evidence	6; 6		
14	flow chart	preparation; running; assessment; investigation; learning	5; 10		
15	circular flow chart	acquire; preserve; normalize; analyse; examine results	5		Remote Computer Networks
16	flow chart	preparation; securing the scene; survey & recognition; documentation; communication shielding; evidence collection; preservation; examination; analysis; presentation; result	11; 2		
17	flow chart	readiness; deployment; trace back; dynamite; review	5		
18	flow chart	pre-process; acquisition & preservation; analysis; presentation; post-process	5		
19	flow chart	planning; identification; reconnaissance; analysis; result; proof & defence; diffusion of information	7; 6		
20	UML activity diagram	preparation; investigation; presentation	3; 5	Investigator; Prosecution; Defence; Court	
21	flow chart	readiness; deployment; physical crime scene investigation; digital crime scene investigation; presentation	5; 16		
22	flow chart	preparation; interaction; reconstruction; presentation	4; 14	Investigator; Court	
23		evidence disk; forensic data extraction; disk image; data carving; attribute distribution; data transfiguration; data analysis; presentation	8; 1		

ID	Diagram Type	Phases	# of phases; sub-phases	Actors	Computing Environment
24		data collection; separation; aggregation; analysis; reporting	5		Computer Networks
25	flow chart	case identification; planning; usage pattern analysis; user-files analysis; reporting	5; 10		
26	flow chart	identification; acquisition of evidence; authentication of evidence; analysis; presentation	5; 3		
27		preparation, incident, incident response, physical investigation, digital forensic investigation, presentation	6; 36		
28		preparation; incident response; recording; collection; examination; analysis; presentation; preservation; feedback; handover	10; 13		
29		preparation; preservation/storage; extraction/survey; examination/analysis; reporting	5	Forensic Examiners, Digital Investigators, Managers	File Systems, Computer Networks
30	flow thing model	acquisition, processing, examination, analysis, presentation	5	Collector, Examiner, Analyst, and Presenter	
31	block diagram	monitoring, logging, preservation, analysis, reporting	5		Wireless LAN
32		incident response and confirmation; physical investigation; digital investigation; incident reconstruction; present findings; incident closure	6; 3		
33		collecting evidence; analysis of individual events; preliminary correlation; event normalizing; event deconfliction; second level correlation; timeline analysis; chain of evidence construction; corroboration	9		

ID	Diagram Type	Phases	# of phases; sub-phases	Actors	Computing Environment
A1	2-D schema	identification; preservation; collection; examination; analysis; presentation; decision	7; 41		
A2		collection; examination; analysis; report	4; 7	Prosecution; Defence; First Responder; Investigator	
A3	abstract	identification; preparation; approach strategy; preservation; collection; examination; analysis; presentation; returning evidence	9		
A4	abstract	preparation; survey; documentation; preservation; examination & analysis; reconstruction; reporting	7		Single Machines

Answering the Research Questions

In this section, the answers to the research questions are as provided:

RQ1: How many DFI models have been proposed in the literature?
AQ1: There are 37 different DFI models presented in the literature.

RQ2: Which types of actors are involved?
AQ2: From the 37 DFI models, only 11 include the involved actors; the common ones among them are identified as: Inspector/ Investigator/ Investigation Team; Manager; Prosecution; Defence; First responder; Court.

RQ3: Do the models comprise taxonomy of incidents?
AQ3: Unfortunately, there is no taxonomy and no handling strategy. In few cases some incidents scenarios are reported as examples.

RQ4: What type of computing environment is considered by the various models?
AQ4: Papers with ID 6, 11, 24, and 29 present a model for Computer Networks; the number 9 described a DFI for VOIP, number 10 was customised for Cloud Computing, number 15 was for Remote Computer Network, number 31 was de-

signed for Wireless LAN; finally, the paper with ID A4 presents a model for non-networked computers.

RQ5: What are the common features among the models?

AQ5: The discovered common features are explained in the following:

- Some models include the readiness phase, which is slightly different from the preparation phase: in this phase some operations for collecting data before the crimes happen are executed.
- In some models, the crime scenes to be investigated are differentiated in physical and digital, depending on the nature of the investigated devices. From these crimes, scenes the evidence and footprints are collected, and different procedures to preserve them are implemented, based on the nature of the devices.
- The design model most commonly used is the flow chart.

5.4.5 Discussion: Lessons Learned

The SLR is a structured and well formulated procedure. From its application to the DFI models some general guidelines can be summarised. First, the computer architecture is not a mandatory feature for designing a novel DFI, but it would be preferable for this to be expressed, in order to have a reliable basis for some specific computing environments, since it happens in the literature works that added this feature. Moreover, it would be more complete to also include the actors / stakeholders in order to assign the responsibilities and to schedule the investigations phases in the most appropriate way and without overlapping. The common features discussed in the previous section might be considered for future models since they are existing in all the sought studies. Finally, remembering that the principal aim was to investigate about the forensic readiness capability in the published literature, only two models include it among their phases. The manner in which it is conceived is very close to a preparation activity, present in thirteen process models, by including some preventive data collection operations from the target computing architectures.

5.5 Summary

An examination of forensic readiness in the context of digital investigations is the topic of this chapter. In order to investigate the presence of such a capability in the context of digital investigations a Systematic Literature Review is performed. Thus, the chapter is structured by following the phases of an SLR. Finally, the results of such an investigation are illustrated and reported.

The following chapter is dedicated to introduce Service Level Agreements. Such contracts are described in terms of life cycle, importance in IT services, and monitoring literature proposals.

Chapter 6

Service Level Agreement Contracts

6.1 Introduction

This chapter is dedicated to introduce Service Level Agreements. Such contracts are described in terms of principal contents; then their life cycle is illustrated; also their importance in IT services is presented, in order to clarify the impact in forensics. Some contractual monitoring literature proposals are illustrated in the chapter: they represent the basis for a forensic readiness capability aimed to monitor some contractual constraints. Finally, an examination of service metrics derived by SLAs for SOA-based and Cloud-based services is provided.

6.2 Service Level Agreements

Information systems and computing capabilities are delivered through the Internet in the form of services; they are regulated by a Service Level Agreement (SLA) contract [50] co-signed by a generic Application Service Provider (ASP) and the end user(s), as happens for instance in the cloud [81]. In such a type of contract several clauses are established; they concern the level of the services to guarantee, also known as quality of service (QoS) parameters, and the penalties to apply in case the requirements are not met during the SLA validity time, among others. The SLAs contracts are written in natural language, and may be personalised by idioms, so that different lingual versions may exist for each of them.

An SLAs validity begins when a customer is looking for a particular (set of) service(s) and it finishes when such a provisioning is terminated (see Figure 6.1). During this time period, both parties are responsible for respecting the clauses, due to the legal value of the document [13, 98]; therefore, a dedicated contract management facility should be part of the service provisioning because of the contractual importance and contents [30]. Some effort has been made in the literature to address this challenge, as discussed in the following sections. In particular, different metrics are exploited to measure specific contractual constraints concerning some non-functional requirements of the services, e.g., availability or performance indicators, are included in the contracts.

6.3 SLA Life Cycle

In a free trade context, the users have the freedom to choose which services they prefer for a specific need. Once such a choice is made, a user usually contacts a specific service provider, which will be responsible for delivering the service to the user after the instantiation of a particular Service Level Agreement contract. An SLA concerning the provisioning of IT services is defied as a *formal, negotiated, document in qualitative and quantitative terms detailing the services offered to a customer* [63].

To the best of our knowledge, an SLA follows the life cycle depicted in the UML [116] state chart diagram in Figure 6.1. An SLA is initially defined via a contractual template, which is customised by the provider depending on some users variation requests on the standard offer. Subsequently a negotiation phase happens, where solutions to the change requests are included, together with information about expenses, penalties, and reports. The SLA Co-Signed phase determines that both entities agree on the actual contractual contents, then the service provisioning can begin. The SLA has a validity time, that can be either explicitly expressed with start and end dates, or with an initial date together with a time interval, both included in the document. Such validity time begins after the contract is co-signed by both parties in the SLA Execution phase.

During its life cycle, an SLA can be subject to revisions to resolve some change requests instantiated by either parties. In case no solution is arrived at, the service provisioning has to be terminated and the SLA no longer has validity.

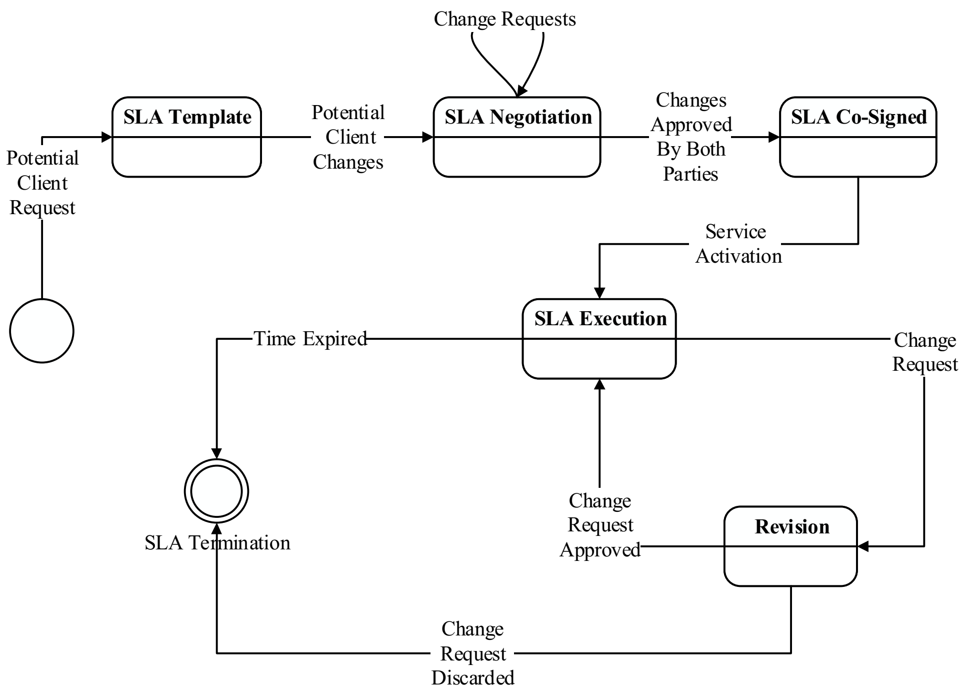


Figure 6.1: SLA Life Cycle [38]

6.4 SLA in IT Services

The IT service provisioning has been being subject to an escalating evolution in terms of deployment architectures, ranging from classical client-server deployment models to cloud computing infrastructures, including all the intermediate varieties (see Figure 6.2) [18].

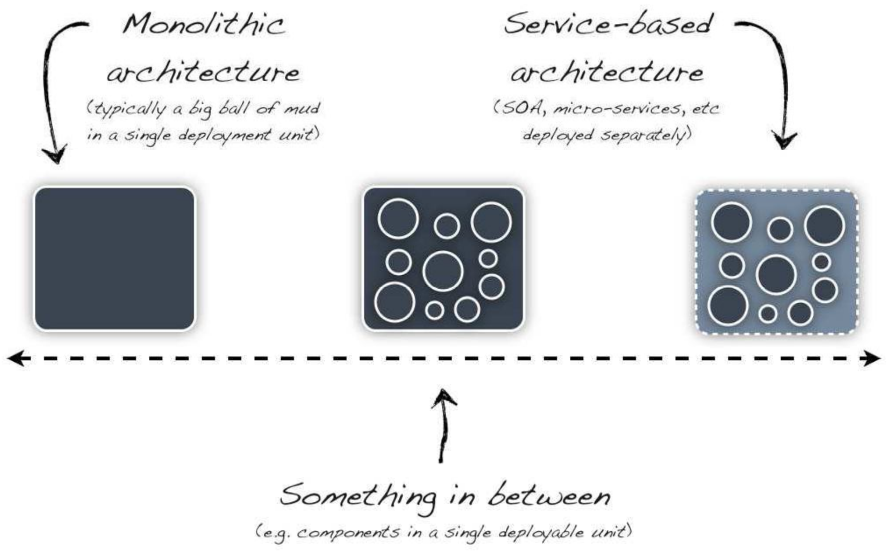

Monolithic architecture
(typically a big ball of mud in a single deployment unit)
Service-based architecture
(SOA, micro-services, etc deployed separately)
Something in between
(e.g. components in a single deployable unit)

Figure 6.2: IT Service Architecture Evolution [18]

The necessity of regulating a fair service out-sourcing via co-signed SLAs aims to give an explicit understanding to the end users about what a service is, where and when it is delivered, and how to use it; also duties and responsibilities of both parties and the possible interaction of a third party is outlined.

The main contents of an SLA concern a definition and description of the service, some rules and regulations about its delivery, some performance measures together with possible tolerance intervals about the levels of the services to guarantee, and the pricing and penalties measures in case such tolerances are not respected. For the sake of monitoring and user satisfaction purposes, it is good practice to provide service levels that can be audited, managed, and measured [76].

One of the biggest issue in automatic SLA monitoring is the lack of a standard for representing the contracts. Unfortunately, the WSLA framework developed by IBM [65] for both specifying and monitoring SLAs is customised for SOA web services. The main advantage of such a proposal is the elasticity of the framework and its capability to customize the underlying computing architecture parameters to measure. This is facilitated by the open access to the resources and to their performances calculators, which is not possible in the cloud.

In the literature, some effort has been made in order to outline some SLAs commonalties, as discussed in [13], where the anatomy of such contracts is provided. The authors of this paper affirm that an SLA is composed of:

- the provided service levels and the metrics used for guaranteeing them;
- the services time duration and their granularity;
- the billing structure;
- the policy about the level measurement;
- the reporting manners about service guarantee violations.

6.5 SLA Monitoring

Automating the management of a textual document is a big challenge, specially if this document is a contract with legal validity like SLAs stipulated for cloud services. One of the reasons to implement a dedicated automatic contractual management system can be the detection of contractual violations to be exploited in many circumstances, such as forensic readiness activities. Several approaches exist in the literature that exacerbate the automatic management of SLAs, and the different modalities to face this issue are discussed in this section.

Some work dedicated to manage the SLAs decompose the issue by monitoring the single constraints included in the contract. For instance, a framework called DESVI [43], proposed by Emeakaroha et al., is dedicated to monitor the performances of low level cloud resources in order to detect if the obtained measures respect the constraints extracted from SLAs with the goal of detecting QoS violations. This work has been used by Emeakaroha et al. as a background monitoring platform in [45] to demonstrate its efficiency in monitoring a single cloud data centre. Also Brandic et al. in [16] used DESVI as a low level resource value calculator, but the metrics applied on the resources have to output a value required to match with a specified threshold to prevent possible contractual violations. In [83] Morshedlou et al. proposed a pro-active resources allocation prototype for reducing the negative impact of SLAs' violations, and for improving the level of users' satisfaction. In [82] a prototype for an autonomic SLA enhancement is discussed by Maurer et al. It behaves as a resource parameter reconfiguration tool at virtual machine level of cloud infrastructures, with the main advantage of reducing SLA violations and of optimizing resource utilisation. Instead, in [44] Emeakaroha et al. proposed an SLA monitoring and violation detection architecture that plays at a cloud application provisioning level, where some metrics are exploited to monitor at runtime the resources consumption behaviour and performance of the application itself. In [26] Cedillo et al. presented an approach to monitor some cloud services non-functional SLA requirements. A middle-ware interacting with services and applications at runtime is designed; it analyses the information and provides some reports as soon as an SLA violation is identified. Last but not least, SALMonADA [85] is a platform proposed by Muller at. al that utilizes a structured language to represent the SLA, which is then automatically monitored to detect whether any violation occurs or not; this detection is performed by implementing a technique based on a constraints satisfaction problem.

6.6 Services Metrics in SLA

Automatic monitoring of textual SLAs has been investigated in many research works. In most cases the contributions deal with approaches designed for translating contractual constraints into a specific format; the purpose is to monitor constraints about resources metrics parameters, because they are the measurable entities described in a contract. Research efforts have been conducted in different computing contexts, and an overview is given in the following sections.

6.6.1 Service Oriented Architecture

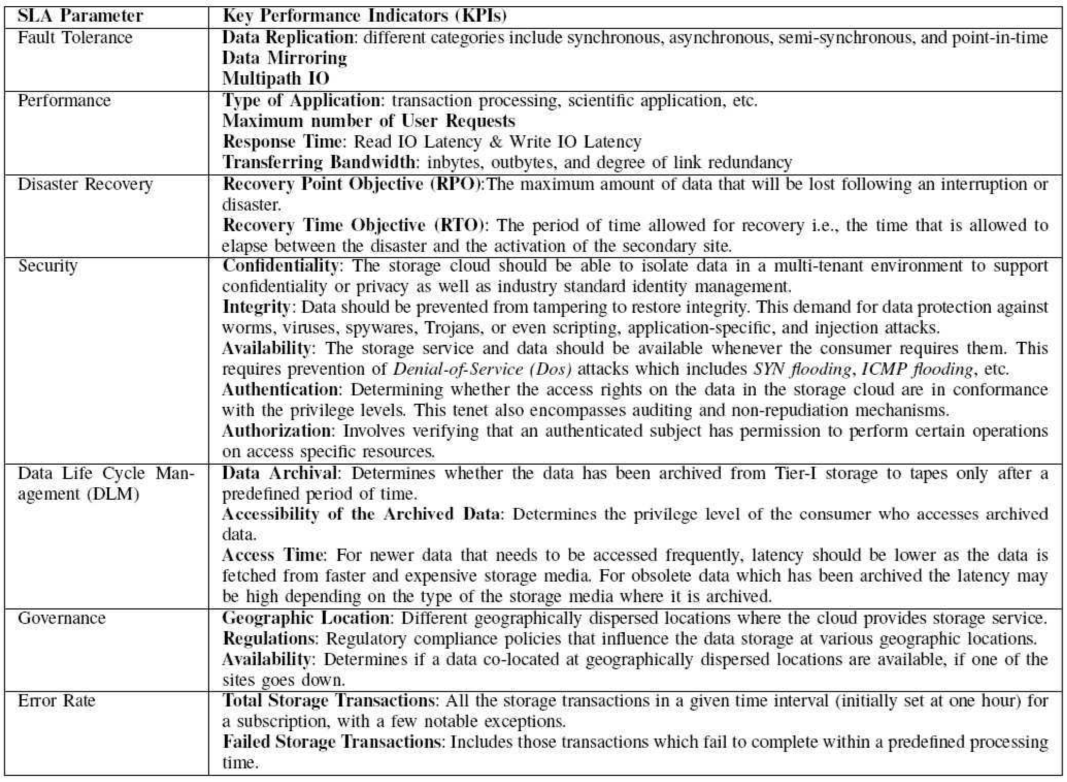
In [8] the improvement of a protocol describing the phases of an SLA life cycle is investigated by Amani et al. The proposed approach begins with an evaluation of the WS-Agreement protocol [11], which provides standard templates for SLAs management. Such a protocol is considered to be lacking of a proper evaluation phase, dedicated to detect service levels violations, thus it is furnished with a resources monitoring facility where some metrics are instantiated. The metrics are dedicated to the measure of operational performances, resources availability, percentage of product requests completed in time, and repair time commitments. The central focus of such a protocol improvement has the goal of determining a fair manner to calculate the penalties related to the resource levels violations constraints. To accomplish this final aim the authors utilize some defined metrics for measuring the unavailability time of the service, and the monthly charge to the customers, necessary in order to determine such penalties.

In [125] SLAs aggregation within business processes is investigated by Unger et al. The contracts are structured in concepts of parties, parameters, and service level objectives, formally modelled with tuples, logic predicates, boolean algebra, and normal forms. The aggregation in a business process happens once some specific requirements are arrived at, and consequently a target service level objective is generated. The aggregation process considers several pieces of information, such as the parameters responsible to express the used metrics necessary to calculate the final quality of service levels. The considered parameters refer to quality attributes such as availability, performance, security, integrity, and reliability of the services.

6.6.2 Cloud Computing

Some efforts in monitoring SLAs by considering the architectural resources metrics have been made in the cloud too. For instance, in [34] a protocol for negotiating SLAs among several actors is defined by Czajkowski et al.; the contractual contents are represented via mathematical tuples. The used information includes the definition of some metrics used to measure the service resources. The idea behind their approach is to parametrize the resources through a set of attributes, intended as particular properties of the resources, e.g., bandwidth, latency, or space. In this approach the time metrics are expressed in this format "Wed Apr 24 20:52:36 UTC 2002", while scalar metrics are expressed in x real-valued units u, e.g. 512 MB. Sometimes, the expected values of the resource properties are constrained in capacity ranges, which are partitioned in exclusive or inclusive upper or lower limits.

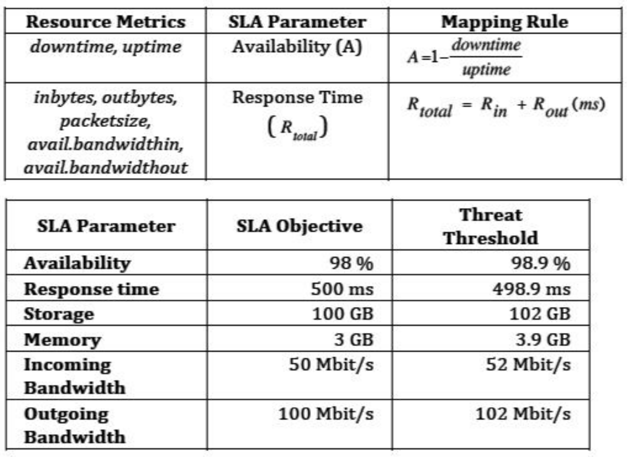
In [57] SLAs are monitored in the context of a Storage as a Service facility, discussed by Ghosh et al. The contractual clauses are decomposed in several services parameters, which are the atomic information of some defined service level objectives (SLO) describing the QoS levels to guarantee. Such SLOs are measured by some key performance indicators (KPI), essentially the metrics to use. The used parameters for this particular service are derived by a deep analysis of the major cloud storage providers, such as Amazon, Salesforce, Google, Microsoft, Rackspace, for instance. The parameters are fault tolerance, performance, disaster recovery, security, governance, data life cycle management, and error rate, which are measured with the metrics, i.e. KPIs depicted in Figure 6.3.



SLA Parameter	**Key Performance Indicators (KPIs)**
Fault Tolerance	**Data Replication**: different categories include synchronous, asynchronous, semi-synchronous, and point-in-time **Data Mirroring** **Multipath IO**
Performance	**Type of Application**: transaction processing, scientific application, etc. **Maximum number of User Requests** **Response Time**: Read IO Latency & Write IO Latency **Transferring Bandwidth**: inbytes, outbytes, and degree of link redundancy
Disaster Recovery	**Recovery Point Objective (RPO)**:The maximum amount of data that will be lost following an interruption or disaster. **Recovery Time Objective (RTO)**: The period of time allowed for recovery i.e., the time that is allowed to elapse between the disaster and the activation of the secondary site.
Security	**Confidentiality**: The storage cloud should be able to isolate data in a multi-tenant environment to support confidentiality or privacy as well as industry standard identity management. **Integrity**: Data should be prevented from tampering to restore integrity. This demand for data protection against worms, viruses, spywares, Trojans, or even scripting, application-specific, and injection attacks. **Availability**: The storage service and data should be available whenever the consumer requires them. This requires prevention of *Denial-of-Service (Dos)* attacks which includes *SYN flooding*, *ICMP flooding*, etc. **Authentication**: Determining whether the access rights on the data in the storage cloud are in conformance with the privilege levels. This tenet also encompasses auditing and non-repudiation mechanisms. **Authorization**: Involves verifying that an authenticated subject has permission to perform certain operations on access specific resources.
Data Life Cycle Management (DLM)	**Data Archival**: Determines whether the data has been archived from Tier-I storage to tapes only after a predefined period of time. **Accessibility of the Archived Data**: Determines the privilege level of the consumer who accesses archived data. **Access Time**: For newer data that needs to be accessed frequently, latency should be lower as the data is fetched from faster and expensive storage media. For obsolete data which has been archived the latency may be high depending on the type of the storage media where it is archived.
Governance	**Geographic Location**: Different geographically dispersed locations where the cloud provides storage service. **Regulations**: Regulatory compliance policies that influence the data storage at various geographic locations. **Availability**: Determines if a data co-located at geographically dispersed locations are available, if one of the sites goes down.
Error Rate	**Total Storage Transactions**: All the storage transactions in a given time interval (initially set at one hour) for a subscription, with a few notable exceptions. **Failed Storage Transactions**: Includes those transactions which fail to complete within a predefined processing time.

Figure 6.3: SLA Parameter and Metric Relation [57]

The framework DESVI [43], proposed by Emeakaroha et al., monitors low level cloud resource metrics to identify whether some parameter values respect the constraints extracted from some SLAs clauses, in order to detect QoS violations. A mapping between low level resources metrics to high level contractual constraints is implemented in a dedicated component called LoM2HiS [42] proposed by the same authors, moreover, such a component is responsible for checking whether or not such constraints are respected by the service providers. The resources parameters monitored in this study are illustrated on the bottom of Figure 6.4; instead the top section of the same figure provides an example of the mapping discussed in the DESVI proposal.



Resource Metrics	SLA Parameter	Mapping Rule
downtime, uptime	Availability (A)	$A = 1 - \frac{downtime}{uptime}$
inbytes, outbytes, packetsize, avail.bandwidthin, avail.bandwidthout	Response Time (R_{total})	$R_{total} = R_{in} + R_{out}$ (ms)

SLA Parameter	SLA Objective	Threat Threshold
Availability	98 %	98.9 %
Response time	500 ms	498.9 ms
Storage	100 GB	102 GB
Memory	3 GB	3.9 GB
Incoming Bandwidth	50 Mbit/s	52 Mbit/s
Outgoing Bandwidth	100 Mbit/s	102 Mbit/s

Figure 6.4: SLA Parameter and Metric Relation [43]

The DESVI framework has been utilised in [45] by Emeakaroha et al. for monitoring a single cloud data centre resources parameters, indeed different scenarios have been analysed in the proposed experiments. Also in another framework proposal designed to prevent possible contractual violations [16], Brandic et al. used DESVI as the underlying platform. In particular, based on some requirements described in the SLA, specific resources performances metrics are applied on some cloud low level resources. The expected values are tested to match with a specific threshold calculated by mining a data collection of past metrics calculations.

The *user satisfaction* point of view is considered by Morshedlou et al. as the resource parameter to be tested in [83]. For reducing the negative impact of SLAs' clause violations, the authors suggest a pro-active resources allocation approach.

The considered metrics are defined as *users willingness to pay for a service* and *user willingness to pay for certainty*. They are obtained by aggregating some resource parameter atomic measures.

Another SLA monitoring and violation detection approach for the cloud is proposed in [44] by Emeakaroha et al. It controls information at the application provisioning layer, by monitoring some resources allocation, scheduling and deployment facilities. The monitoring happens at runtime and it has to detect some application metrics in order to determine resources consumption and application performances. The information related to the metrics are collected by application or operating systems log files. The metrics used in this approach are all devoted to calculate service performances such as guaranteeing the application response time and throughput values to be respected by the constraints of the SLAs. Although they can vary depending on the application type, the parameters are CPU and memory utilisation, among others.

An approach to monitor some cloud services non-functional requirements is presented in [26] by Cedillo et al. The information to monitor is contained in SLAs; the discussed solution for this issue is shaped as a middle-ware interacting directly with services and applications at runtime. The middle-ware analyses the information and provides some reports as soon as an SLA violation is identified. The presented case study considers some parameters as services availability, which is measured through the metric robustness of service. The authors argue that the middle-ware is capable of measuring different services parameters by instantiating proper metrics, and the choice is strongly related to the SLA constraints to consider.

A European community report [48] analyses different research projects devoted to automate different aspects of the SLAs life cycle in the cloud. Some projects investigate the management of the service level parameter violations, and consider some specific metrics, although with different purposes and final objectives.

For instance, Cloud4SOA [28] aims to monitor the performances of business-critical distributed PaaS applications. The proposed solution compares the actual performances values with their expected values described in some SLAs constraints. The main challenge faces the variety of metrics among the providers and it is addressed by implementing specific APIs. Moreover, a set of unified metrics across the PaaS providers is selected to monitor the application execution and usage. Such metrics are at the application and infrastructure level, and include application / database response time, cloud response time, web container response time, application status, memory and CPU usages.

In the EC research report in [48] some other projects involve the measurement of some cloud services attributes, such as availability or response time, but unfortunately the metrics used are not reported.

The cloud dynamic resource allocation is the issue investigated in [82], where Maurer et al. propose a prototype for an autonomic SLA resource parameters reconfiguration tool at cloud virtual machines level. The main advantage is to reduce SLA violations by optimizing resources utilisation. The workload volatility is the used parameter leading to a better reconfiguration, and the rapidity of changes in the machines workload is the used metric. For the considered resources, e.g., CPU, storage, memory, or bandwidth, the utilisation is divided in three categories, namely over-provisioned, normal-provisioned and under-provisioned. They are used to calculate the target value of a resource provisioning that optimizes the global resources utilisation. The autonomic resources adaptation is then computed, depending on two different parameters: the first considers a cost function capable of determining the penalties to pay according to the number of SLA violations; the latter pays attention to the workload volatility defined as the intensity of changes in the measured workload for a specific resource.

6.7 Literature Description

Contractual management is a delicate discipline that involves different perspectives, such as legal validity, rights granting, and responsibilities guaranteeing. In computing services this topic becomes instead less complicated. Dealing with machines is a rational activity due to the logic and unambiguous nature of the computing components. Indeed, even the examined literature presented in Section 6.6 deals with different computing architectures, the parameters to monitor guaranteeing SLA constraints involve machine and network attributes that can be rationalised through specific metrics. All the considered metrics involve the quantification of some resources physical attributes, such as bandwidth, latency, space, fault tolerance, error rate, response time, memory, and CPU.

The considered parameters do not take into account information about time, which is instead included in the SLAs. Indeed, beyond the whole contract validity time, individual constraints may express different values for the same parameters in different time intervals; for instance a service availability can be 99% during the working days and 50% at the weekends. Consequently, the values measured with a specific metric have to match depending on temporal information for correctly detecting a service level violation.

In other cases, some atomic metrics are combined to generate others, such as service availability, workload volatility, robustness of a service, and willingness to pay for a service. Moreover, the metrics aggregation can provide measures for other quality attributes included in the SLAs, as for instance the ones concerning security aspects.

An escalation of metrics aggregation can be also a driver toward an additional research topic, namely the automatic management of the whole SLA life cycle. It can be obtained by aggregating studies that concern the automation of individual phases, where the gaps can be bridged by designing facilities capable of measuring specific parameters. For instance, it would be interesting to obtain a measure evaluating the necessity of a customer to revise the SLA during its validity time, by calculating some user-related attributes.

6.8 Summary

Service Level Agreements are introduced in this chapter. Such contracts are described in terms of principal contents; then their life cycle is illustrated; also their importance in IT services is presented, in order to clarify the impact in forensics. Some contractual monitoring literature proposals are illustrated in the chapter: they represent the basis for a forensic readiness capability aimed to monitor some contractual constraints. Finally, an examination of service metrics derived by SLAs for SOA-based and Cloud-based services is provided.

The following chapter is also focusing on SLAs; they are considered in the context of forensic readiness. An explanation of the contractual constraints is presented and then a selection of some of them necessary for forensics is provided. Finally, a mapping of the selected constraints on the principal security threats for the cloud is illustrated.

Chapter 7

SLA in Cloud Forensics Readiness

7.1 Introduction

This chapter is focusing on SLAs in the context of forensic readiness. An explanation of the contractual constraints, named service level objectives, proposed by some EU guidelines is presented. Some of them are necessary for forensics, so their selection is broadly detailed in dedicated sections: their presence on a hundred publicly accessible SLAs is calculated. Then, a mapping of constraints on the principal security threats for the cloud is illustrated. Finally, a selection of primary, secondary, and optional service level objectives for forensic readiness is derived.

7.2 Service Level Agreements Interaction

In the cloud the SLAs are co-signed between a provider and a customer who subscribes a service utilisation. Additional SLAs can exist in other circumstances, for example SLAs co-signed among different providers for hardware and software resources out-sourcing, or SLAs involving third parties. Most of the times, the customers are unaware of the complete data flow along different sub-providers; this is because the chain of sub-services necessary to accomplish an activity and the related SLAs are not disclosed to unconcerned parties. Figure 7.1 illustrates a possible customers and providers interaction governed by SLAs. Interactions and SLAs can expand depending on the chain of services out-sourcing.

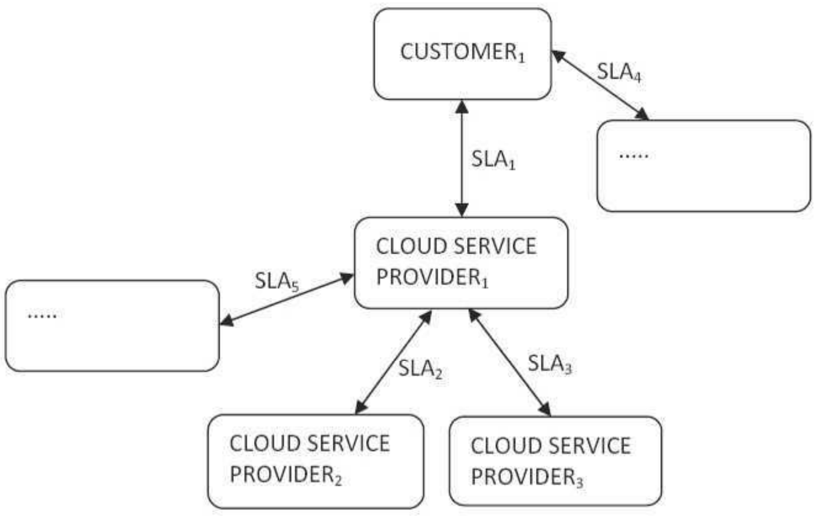


Figure 7.1: SLAs Interactions [37]

An SLA is composed of a set of clauses, which describe all the constraints, behaviours, and duties of the co-signer parties in order to guarantee the level of the predefined services. For instance, some clauses concern the metrics necessary for measuring the described services level attributes, such as latency or average transmission errors rate. In this chapter an analysis of the SLAs contents is discussed, together with a classification of some contractual contents in a cloud forensic readiness context.

7.3 Contractual Constraints

The structure of an SLA may differ from one cloud service provider to another. However, such a contract is composed of several sections. Among these sections, an SLA can be structured as a set of Service Level Objectives (SLO). In a European Union guideline document [49] the described SLOs are catalogued as follows:

- Performance;
- Security;
- Data management;
- Personal data management.

The focus of this section is to outline the SLOs necessary for digital forensic readiness. The approach undertaken is composed of the following steps. Initially, the presence of the SLOs mentioned in [49] has been verified in most public cloud service providers that have accessible SLAs. The annual list of the hundred most important cloud providers published by the top news source CRN has been utilised for this purpose [33]. After the selection has been made, as described in Section 7.3.1, the resulting SLOs are matched with the cloud threats discussed in a cloud security alliance report [31]. The mapping result is shown in Table 7.7.

7.3.1 SLO Selection

The annual list of the hundred most important cloud providers [33] is composed of providers in every cloud service model. The model categorisation utilised in this document includes IaaS, PaaS, SaaS, Storage, and Security; moreover, there are twenty providers for each model. After an initial screening of the document, some included providers are still open projects, therefore excluded from the SLO selection study due to not providing the necessary information. The whole set of analysed providers is composed of seventy-six elements; unfortunately only half of the selected elements provides a public SLA, mostly as providers of infrastructure model services.

All the analysed SLOs are described in the following tables, depending on what category they belong to: the description is composed of their name, together with a definition, and the percentage of their presence in the analysed thirty-eight SLAs. In Table 7.1 the attributes composing the performance SLOs are described; in Table 7.2 the ones for security SLOs; in Table 7.3 the attributes related to data management; and in Table 7.4 the attributes composing data protection SLOs.

Table 7.1: Performance SLO Presence Percentage

Name	**Definition**	**Presence**
Level of uptime	Time of the service availability, expressed as a percentage	84.2%
Percentage of successful requests	Number of requests processed by the service without errors over the total number of submitted requests, expressed as a percentage	0%
Percentage of timely service provisioning requests	Number of service provisioning requests completed within a defined time period over the total number of service provisioning requests, expressed as a percentage	0%
Average response time	Statistical mean over a set of cloud service response time observations for a particular form of request	0%
Maximum response time	Maximum response time target for a given particular form of request	0%
Number of simultaneous connections	Maximum number of separate connections to the cloud service at one time	0%
Number of simultaneous cloud service users	Target for the maximum number of separate cloud service customer users that can be using the cloud service at one time	0%
Maximum resource capacity	Maximum amount of a given resource available to an instance of the cloud service for a particular cloud service customer	2.7%
Service throughput	Minimum number of specified requests that can be processed by the cloud service in a stated time period	0%

Name	**Definition**	**Presence**
External connectivity	Service capability to connect to systems and services which are external to the cloud service	7.8%
Support hours	Time period a provider accepts general inquiries and requests from the customer	0%
Support responsiveness	Maximum time the provider acknowledges a customer about inquiries or requests	0%
Resolution time	Target resolution time for customer requests	39.4%
Data retrieval period	Time necessary to a customer to retrieve a copy of personal data from the service	0%
Data retention period	Time period a provider retains backup copies of the customer data during the termination process	7.8%
Residual data retention	Customer data retained after the end of the termination process	0%

Table 7.2: Security SLO Presence Percentage

Name	**Definition**	**Presence**
Level of redundancy	Level of redundancy of the service supply chain taking into account the percentage of the components or service that have fail-over mechanisms	0%
Service reliability	Ability of the service to perform its function correctly and without failure over some specified period	2.7%
User authentication and identity assurance level	Level of assurance of the mechanism used to authenticate a user accessing a resource	0%
Authentication	Availability of the authentication mechanisms supported by the CSP on its offered cloud services	0%
Mean time required to revoke user access	Arithmetic average of the times required to revoke users access to the cloud service on request over a specified period of time	0%
User access storage protection	Mechanisms used to protect cloud service user access credentials	0%
Third party authentication support	Third party authentication is supported by the service and with which technologies	0%
Cryptographic brute force resistance	Strength of a cryptographic protection applied to a resource based on its key length	0%
Key access control policy	Whether a cryptographic key is protected from access, when it is used to provide security to the service	0%
Cryptographic hardware module protection level	Level of protection afforded to cryptographic operations in the service through the use of cryptographic hardware modules	0%
Percentage of timely incident reports	Service incidents reported to the customer in a timely fashion. Percentage of the number of defined incidents reported within a predefined time after discovery, over the total number of defined incidents to the cloud service	0%

Name	Definition	Presence
Percentage of timely incident responses	Incidents assessed and acknowledged by the provider in a timely fashion. Percentage of the number of defined incidents assessed and acknowledged by the provider within a predefined time limit after discovery, over the total number of defined incidents to the cloud service	0%
Percentage of timely incident resolutions	Percentage of defined incidents against the service resolved within a predefined time limit after discovery	0%
Logging parameters	Parameters captured in the service log files	0%
Log access availability	Log file entries a customer has access to	0%
Logs retention period	Time period logs are available for analysis	0%
Certifications applicable	List of certifications held by the provider for a service, including body, expiration date and renewal period	0%
Percentage of timely vulnerability corrections	Number of vulnerability corrections performed by the provider, represented as a percentage of the number of vulnerability corrections performed over the total number of them	0%
Percentage of timely vulnerability reports	Number of vulnerability reports to the customer, represented as a percentage of the number of vulnerability reports over the total number of them	0%
Reports of vulnerability corrections	Mechanism to inform customers of vulnerability corrections applied to the systems together with its frequency	0%
Cloud service change reporting notifications	Type of change, mechanism and notification frequency to the customers	0%
Percentage of timely cloud service change notifications	Number of change notifications made within a specified period of time over the total number of change notifications, expressed as percentage	0%

Table 7.3: Data Management SLO Presence Percentage

Name	Definition	Presence
Cloud service customer data use by the provider	Policies for any intended use of customer data	0%
Cloud service derived data use	What is the derived data created by the provider from customer data, the intended uses and rights of the customer	0%
Data mirroring latency	Difference between the time data are placed on primary storage and the time same data are placed on mirrored storage	0%
Data backup method	List of methods used to backup cloud service customer data	13.8%
Data backup frequency	Time period between complete backups of customer data	0%
Backup retention time	Time period a given backup is available in data restoration	2.7%
Backup generations	Number of backup generations available in data restoration	0%

Name	Definition	Presence
Maximum data restoration time	Committed time to restore customer data from a backup	0%
Percentage of successful data restorations	Committed success rate for data restorations, expressed as percentage between the number of data restorations performed for the customer without errors over the total number of data restorations	0%
Data deletion type	Quality of deleted data	0%
Percentage of timely effective deletions	Percentage between the number of customer data deletion requests completed within a predefined time limit over the total number of deletion requests	0%
Percentage of tested storage retrievability	Amount of customer data retrievable after being deleted	0%
Data portability format	Electronic formats a customer data can be transferred to / accessed from the service	0%
Data portability interface	Mechanisms used to transfer customer data to and from the service	0%
Data transfer rate	Minimum rate customer data can be transferred to / from the service using the mechanisms stated in data interface	0%

Table 7.4: Data Management SLO Presence Percentage

Name	Definition	Presence
Applicable data protection codes of conduct, standards, certifications	Data protection codes of conduct, standards and certification mechanisms that the service complies with	7.8%
Processing purposes	Processing purposes acting as controllers	0%
Temporary data retention period	Maximum time period temporary data are retained after identified as unused	0%
Cloud service customer data retention period	Maximum time period customer data are retained before destruction, after acknowledgement of a delete request or contract termination	0%
Number of customer data law enforcement disclosures	Number of personal data disclosures to law enforcement authorities over a predefined period of time	0%
Number of personal data disclosure notifications	Number of personal data disclosures to law enforcement authorities actually notified to the customer over a predefined period of time	0%
List of tier subcontractors	Providers subcontractors involved in the processing of customers personal data	0%
Special categories of data	List of specific categories of personal data	0%
Personal data breach policy	Data breach policies adopted by the providers	0%

Name	Definition	Presence
Documentation	List of documents the provider makes available to demonstrate compliance to data protection requirements and obligations	0%
Data geolocation list	Geographical locations where customers data may be stored and processed by the provider	2.7%
Data geolocation selection	Whether customer can choose a given geographical location for the storage of his data	0%
Access request response time	Time period the provider shall communicate the information necessary to allow the customer to respond to access requests by data subjects	0%

From the percentage of presence described above in Tables 7.1 - 7.4, only ten SLOs out of the proposed sixty-six, i.e., 15.1%, are present in the public SLAs retrieved from [33]. Some others are instead present in other contractual documents, like terms of service (TOS) or general agreements. However, some of the absent features are useful to prevent some cloud threats.

7.3.2 SLO and Security Threats

According to the Cloud Security Alliance document [31], the main threats for cloud services security can be:

- Data breach / loss: a customer can lose control of their data; there can be several causes, such as multi-tenancy, provider vulnerabilities, network misuse;
- Hijacking: attackers can gain access to customers credentials to manipulate data, return false information, redirect the navigation to illegitimate sites. In addition, the power of a cloud provider can be leveraged to launch subsequent attacks. Cloud vulnerabilities permitting hijacking attacks include mash-up authorisation, the transitive nature of cloud, or authentication and authorisation vulnerabilities;
- Insecure APIs: attackers can be aware of the service architecture and design

details; the providers should select what to render publicly available through encryption, abstraction, or encapsulation mechanisms;

- DoS and DDoS: such attacks prevent users from access to their data or applications. Not much effort has been expended do defend services platforms from such threats;
- Malicious insiders: an attacker can have access to sensitive information. Even with some encryption techniques implemented, the system is still vulnerable to malicious insiders;
- Abuse of services: attackers use cloud platforms to address their attacks, and to host illegal materials. Nevertheless, providers allow quick and easy services subscriptions; this makes it harder to detect an offender identity;
- Lack of transparency: cloud organisations promise cost reduction together with operational and security improvement; several risks and issues can then arise, not disclosed to enterprises and organisations moving to cloud services;
- Shared technology: multi-tenancy architectures and shared resources represent key points of elastic scalability guaranteed by cloud organisations; unfortunately these features introduce some vulnerabilities. A compromised component shared in the architecture can represent a threat for the whole system.

In order to prevent some security threats via a forensic readiness capability, some cloud parameters need to be measured. The measurement constraints to be guaranteed are identified in the SLOs, but not in all of them; thus, according to their relation with the main cloud threats, they can be classified as Primary, Optional, and Unnecessary SLOs. In Table 7.5 the primary SLOs related to the cloud threats are shown, and also the rationale behind the proposed mapping. In the same manner, a mapping is proposed for secondary SLOs in Table 7.6.

Table 7.5: Primary SLOs - Security Threats Mapping

SLO	**Cloud Threat**	**Motivation**
Level of uptime	DoS or DDoS - Hijacking	If the cloud has been unavailablefor longer how much it is allowed, then there can be a DoS or DDoS attack caused by a hijacking threat
Percentage of successful requests	DoS or DDoS - Hijacking	If the percentage is lower than how much allowed, then the network can be under DoS or DDoS attack caused by a hijacking threat
Average response time	DoS or DDoS - Hijacking	If the response time is bigger than how much allowed, then the network can be under DoS or DDoS attack caused by a hijacking threat
Max response time	DoS or DDoS - Hijacking	If the response time is close to the maximum allowed, then the network can be under DoS or DDoS attack caused by a hijacking threat
Max number of simultaneous connections	DoS -Hijacking	A number of connections from a single client close to the maximum allowed can determine a DoS attack caused by a hijacking threat
Max number of simultaneous cloud service users	DDoS	A number of simultaneously connected users close to the maximum allowed can be a sign of an DDoS attack
Percentage of service provisioning requests in time	DoS or DDoS - Hijacking	If the percentage is lower than how much allowed in time, then the network can be under DoS or DDoS attack caused by a hijacking threat
External connectivity	DoS or DDoS - Hijacking	If the measures of jitter and latency are out-of-bounds, then there may be a DoS or DDoS attack caused by a hijacking threat
Backup method	Data breach	The availability of the backup method is useful to identify possible unhallowed access to data backup
Data transfer rate	DoS or DDoS - Hijacking	If the data transfer rate is lower than the allowed average, then the network can be under DoS or DDoS attack caused by a hijacking threat

Table 7.6: Secondary SLOs - Security Threats Mapping

SLO	**Cloud Threat**	**Motivation**	**Optionality**
Max resources capacity	DoS or DDoS - Hijacking	If the resource capacity is close to the maximum, then the resources can be under DoS or DDoS attack caused by a hijacking threat	It is an optional feature because applicable only to network resources, e.g., storage resources capacity is not enough

SLO	**Cloud Threats**	**Motivation**	**Optionality**
Service throughput	DoS or DDoS - Hijacking	If the throughput is lower than the allowed average, then the network can be under DoS or DDoS attack caused by a hijacking threat	It is optional because related to the actual user usage of the network, e.g., if the network is not used, the value is affected negatively
Log access availability	All	It is useful to know what information can be read by an intruder on the network	It is optional because hardly made public by providers
Backup retention time	Data breach	When a backup is made available is useful to identify possible unhallowed access to data backup	The parameter is usually expressed by days or weeks, which is a long period to detect real-time violations
Applicable data protection, codes of conduct, standards, specifications	All	It is useful to know what are the measures to defend from intruders	This SLO is often generic and approximate: providers can deny to disclose their security measures

All the SLOs that are not considered in Tables 7.5 and 7.6 are consequently classified as unnecessary, as they are not measurable with monitoring tools, e.g., support hours, or are not necessary for forensic readiness purposes, e.g., cloud service change reporting notifications. In Table 7.7 a summary of the results discussed in this section is illustrated.

Table 7.7: SLOs Classification

Primary	**Optional**	**Unnecessary**
Level of uptime	Max resources capacity	All the others SLOs
Percentage of successful requests	Service throughput	
Average response time	Log access availability	
Max response time	Backup retention time	
Max number of simultaneous connections	Applicable data protection, codes of conduct, standards, specifications	
Max number of simultaneous cloud service users		
Percentage of service provisioning requests in time		

External connectivity		
Backup method		
Data transfer rate		

7.4 Final Remarks

Evidence to be collected during a forensic investigation has to respect some court admissibility guidelines [1, 89]. In some cases, such guidelines can be in contrast with some SLA constraints expressing jurisdictional principles. For instance, let us assume that a SaaS cloud service provider *X* is responding to the European Jurisdiction. It can out-source additional services from a storage cloud service provider *Y* responding to the Asian or Middle East laws. The SLA regulating the relationships between *X* and *Y* includes some clauses that do not allow the collection of evidence, such as network logs or database transaction logs, and that regulate data access depending on other jurisdictions. Let us also assume that a customer of *X* accessing to the service from the U.S. is victim of a data breach crime, and law enforcement has to conduct an investigation. Very likely, depending on both the SLAs regulating the relationships between the customer and *X*, and *X* and *Y*, respectively, such an investigation cannot be finalised due to the presence of the clauses denying access to the potential evidence. The hypothetical investigation can collect evidence-related data belonging to the communications between the customer and the service provider *X*; moreover the logs from both customer and provider sides can be utilised, as well as the performance indicators, and the values of the used metrics to evaluate the resources parameters. However, once the investigation has to deal with the infrastructure of the service provider *Y*, depending on the expressed constraints the access to the necessary information can be denied to law enforcement.

7.5 Summary

SLAs in the context of forensic readiness is the topic of this chapter. An explanation of the contractual constraints, named service level objectives, proposed by some EU guidelines is presented. Some of them are necessary for forensics, so their selection is broadly detailed in dedicated sections: their presence on a hundred publicly accessible SLAs is calculated. Then, a mapping of constraints on the principal security threats for the cloud is illustrated. Finally, a selection of primary, secondary, and optional service level objectives for forensic readiness is derived.

The following chapter is dedicated to a classification of SLAs for forensic readiness. A possible classification in three classes of some contractual contents is provided and motivated. Then, an automation of such a classification is proposed: it is performed by a customisation of a text engineering tool. At the end of the chapter an assessment of such an automation is performed utilising a dataset of thirty-six public SLAs.

Chapter 8

Automatic SLA Classifier

8.1 Introduction

This chapter is dedicated to a classification proposal of SLAs, dedicated to detect the sections of the contracts to monitor by a forensic readiness capability. Such a classification is composed of three classes, representing a definition of a parameter to monitor, a value of such a parameter, and all the remaining sentences to discard. Then, an automation of such a classification is proposed: it is performed by a customisation of an already existing text engineering tool. At the end of the chapter an assessment of such an automation is performed utilising a dataset of thirty-six public SLAs.

8.2 SLOs Structure

The SLA contracts are written in natural language and legal jargon because they have validity in case of a court litigation. The considered contents are related to the forensic readiness context, and they are identified as Service Level Objectives (SLOs) as discussed in Chapters 6 and 7. Every SLA is different from provider to provider, so the necessity for an automation of SLOs recognizer can be considered as a matter of urgency.

It is reasonable to affirm that an SLA is a set of SLOs, because each of them describes a single parameter without overlapping with the remaining ones. An SLO is composed of a set of sentences, usually more than one. The SLO describes a constraint about a parameter of a service level included in an SLA, together with the value and a unit measure and / or a percentage value of such a parameter. An SLO in some cases can describe the metrics used by the service provider to calculate the value of the parameter it indicates. Generally, the description is textual, expressed in natural language; in some other cases, beside the textual description, a mathematical formula is textually described. The time interval to consider is also an important feature. Every SLA has a validity time period expressed either explicitly with start and end dates, or with a start date plus a time period, such as a *billing month* or a *solar year*. The SLOs included in the SLA are not necessarily constrained during the same time interval; indeed they can have validity during a different time period, which can even end after the SLA validity time, e.g., the *backup retention time* SLO can finish after the SLA termination; certainly, no SLO can begin before an SLA starting time.

8.3 SLO Violation

An SLO is legally violated when the actual value of the parameter is calculated by the formula included in the constraint description, and it does not respect the foreseen value included in the description too. In the cases the metric / formula to calculate the value of the parameter is not expressed, the actual value can be calculate by using some documented metrics, in such a way the comparison can be performed. An SLO violation has in general a billing consequence for the provider; indeed, it can determine that the customer receives some extra money on the account because the service level was violated in the provider side, so out of the customer control. Moreover, from the forensic perspective, one or more SLO violations can be symptoms of a cyber attack in the service infrastructure. The mapping proposed in Section 7.3.2 is an example if such relationships.

8.4 SLO Classification

The sentences composing the SLOs can be classified by following the proposed approach [86]:

1. Definition
2. Value
3. Not Definition

The assumption made in this approach is that every sentence of an SLO can fall only in one class; a sentence is a sequence of words enclosed in two full stops, where the initial one is discarded. The sentences composing an SLO can represent either a value together with a unit measure or percentage for the constrained parameter, or the metric description. So, the classifier has to recognise whether a sentence of an SLO is a definition, a numeric value, or not a definition; in the latter case the sentence is discarded by forensic readiness. A subsequent step of the classification is the identification of mathematical and textual formulas from the detected definition; in this way the class *Definition* can be split in two:

- Mathematical Formula
- Textual Formula

The mathematical formula represents a manner in which the detected description can be easily translated into a mathematical formula, namely some strategic words are recognised, such as *adding, divided by, rate, ratio, ...*.

The textual formula instead describes a definition that outlines a textual description for the metric necessary to the computation of the value of a parameter included in the SLO, which cannot be represented with a mathematical formula.

It is necessary to mention that the difference between mathematical and textual formulas is very narrow; namely a sentence can belong to either one just depending on some few words due that the description is anyway textual, but in both cases they represent a description for a formula. This reason motivates also the presence of one class and two subclasses, instead of proposing four classes in total.

8.5 Automatic Classifier for SLAs

In order to design and develop an automatic SLA classifier, some Natural Language Processing (NLP) techniques have to be utilised. Such techniques have the aim to elaborate the document containing an SLA and to obtain the information about the SLOs necessary to forensic readiness to feed a dedicated service monitoring tool. A means of addressing this challenge in SLACFR is represented by the usage of Algorithm 1 ([39]). The algorithm correctly identifies what words to either consider or discard. The computation recognizes a set of words representing the information according to the formal model described in Chapter 9, to populate specific sets, such as resources, attributes and unit measures, which contain elements to be considered without being combined with anyone else. Conversely, the set of indicators *I* includes information derived from the other three domains. The output is represented by the population of the sets of resources, attributes, unit measures, and indicators.

Algorithm 1 Text to Formal Rules

```
1: procedure T2FR($SLA$)                                        ▷ Textual SLA
2:     Information Extraction techniques set up parameters;
3:     while Textual SLA not ended do
4:         $current \leftarrow InformationExtractiontechniquesoutput$;
5:         if $currentisrecognised$ then
6:             $PopulateR||A||U$;
7:         else
8:             $discardcurrent$;
9:         end if
10:    end while
11:    $I \leftarrow combinationofR, A, U$;
12:    return $R, A, U, I$;                              ▷ The sets are populated
13: end procedure
```

The principal open-source Information Extraction (IE) [58] tools are, Apache OpenNLP, OpenCalais, DBpedia, and GATE. The principal tasks of OpenNLP are tokenisation, sentence segmentation, part-of-speech tagging, named entity extraction, chunking, parsing, and coreference resolution. All these tasks are present also in OpenCalais and GATE with a more usable graphical interface. OpenCalais can annotate documents with rich semantic meta-data, including entities, events, and facts. However, the output of the tool is text enriched by annotations, which are not user-customizable.

GATE [56] performs all the tasks described for the other tools and has the advantage of being customizable; indeed, it allows to create customised types of annotations using a JAPE transducer personalisation; also the annotation that the tool recognizes can be customised. Due to its features GATE is used for the implementation of the automatic classifier for SLAs [86]. The most used component from GATE for this automation is the Information Extraction (IE) one. The input to the system is a dataset of SLA documents in MS Word or Adobe formats. The output of the classifier is composed of the same documents annotated. The annotations are added to some sentences of the documents and they correspond to the identified classes described before. Subsequently, the tool extracts the representation of the metrics for the SLO parameters. The implementation of the classifier works following two sequential steps:

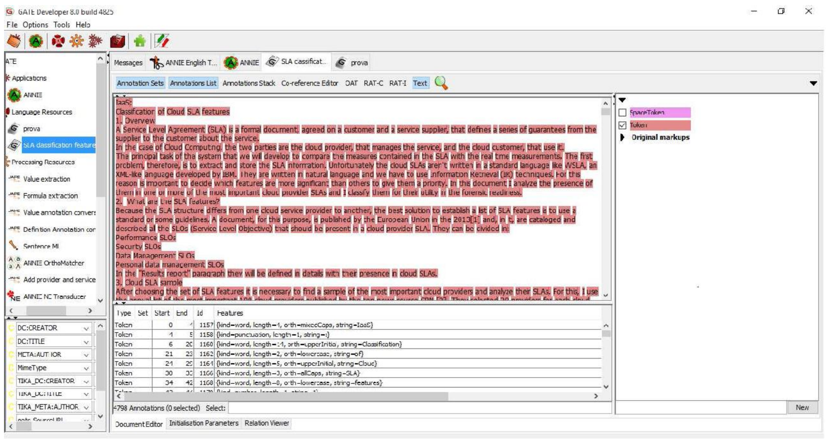
- Step 1: classification of the sentences of the document according to the three classes described before;
- Step 2: identification of the mathematical and the textual formulas included in the *Definition* class from the previous step.

8.5.1 Step 1

The ANNIE (A Nearly-New Information Extraction system) plug-in is the principal and most used component of the software GATE. It takes an input document that is annotated as output of the process. Step 1 is composed of a pipe of activities, where each depends on the output of the previous one and gives the input for the subsequent. Almost each activity of this pipe is responsible to implement and execute a specific information extraction technique. In this section the whole pipe of phases composing Step 1 is described.

- Document Reset: this phase is responsible to delete all the annotations already included in the document; it cleans the file so that it is ready for the whole annotation process.
- Tokenisation: the tokeniser component in GATE implements a word segmentation technique. It divides the document in tokens, such as numbers,

punctuations, and so on. The tokeniser implementation in GATE is based on the primitives Java types of the Class Character; such types are combined via regular expressions and an initial annotation of *token* is assigned to them. Subsequently, each token is recognised as a *kind*, such as word, number, symbol, or punctuation, orth, length, and string.



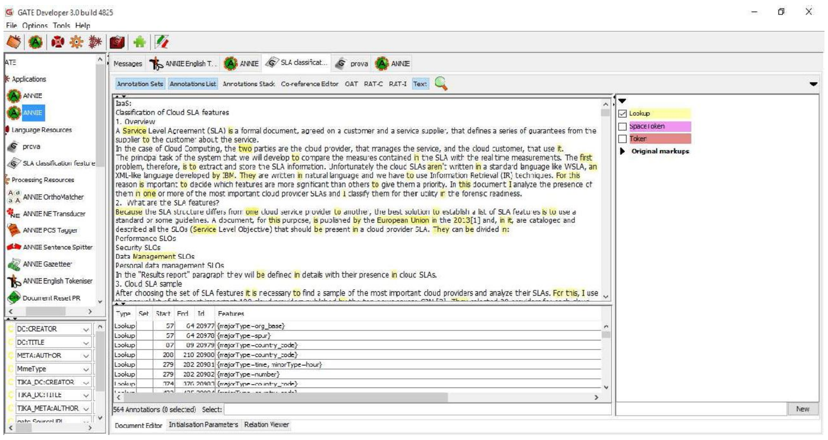
IaaS:
Classification of Cloud SLA features
1. Overview
A Service Level Agreement (SLA) is a formal document, agreed on a customer and a service supplier, that defines a series of guarantees from the supplier to the customer about the service.
In the case of Cloud Computing, the two parties are the cloud provider, that manages the service, and the cloud customer, that use it.
The principal task of the system that we will develop to compare the measures contained in the SLA with the real time measurements. The first problem, therefore, is to extract and store the SLA information. Unfortunately the cloud SLAs aren't written in a standard language like WSLA, an XML-like language developed by IBM. They are written in natural language and we have to use Information Retrieval (IR) techniques. For this reason is important to decide which features are more significant than others to give them a priority. In this document I analyze the presence of them in one or more of the most important cloud provider SLAs and I classify them for their utility in the forensic readiness.
2. What are the SLA features?
Because the SLA structure differs from one cloud service provider to another, the best solution to establish a list of SLA features is to use a standard or some guidelines. A document, for this purpose, is published by the European Union in the 2013[1] and, in it, are cataloged and described all the SLOs (Service Level Objective) that should be present in a cloud provider SLA. They can be divided in:
Performance SLOs
Security SLOs
Data Management SLOs
Personal data management SLOs
In the "Results report" paragraph they will be defined in details with their presence in cloud SLAs.
3. Cloud SLA sample
After choosing the set of SLA features it is necessary to find a sample of the most important cloud providers and analyze their SLAs. For this, I use

Type	Set	Start	End	Id	Features
Token		0	4	1157	{kind=word, length=4, orth=mixedCaps, string=IaaS}
Token		4	5	1158	{kind=punctuation, length=1, string=:}
Token		6	20	1160	{kind=word, length=14, orth=upperInitial, string=Classification}
Token		21	23	1162	{kind=word, length=2, orth=lowercase, string=of}
Token		24	29	1164	{kind=word, length=5, orth=upperInitial, string=Cloud}
Token		30	33	1166	{kind=word, length=3, orth=allCaps, string=SLA}
Token		34	42	1168	{kind=word, length=8, orth=lowercase, string=features}

4798 Annotations (0 selected) Select:

Figure 8.1: Tokenisation Example

- Gazetteer: this component is responsible to perform a Named Entity Recognition (NER) technique. It identifies the names of the entities based on some lists. Such lists are simple text file with an entry for each line. Each list represents a set of names depending on a domain, such as cities, week days, organisations, and so on. An index exists to give access to such lists. As default the gazetteer creates a special annotation named *Lookup* for each entry found in the text. GATE gives the possibility to create a gazetteer with personalised lists.



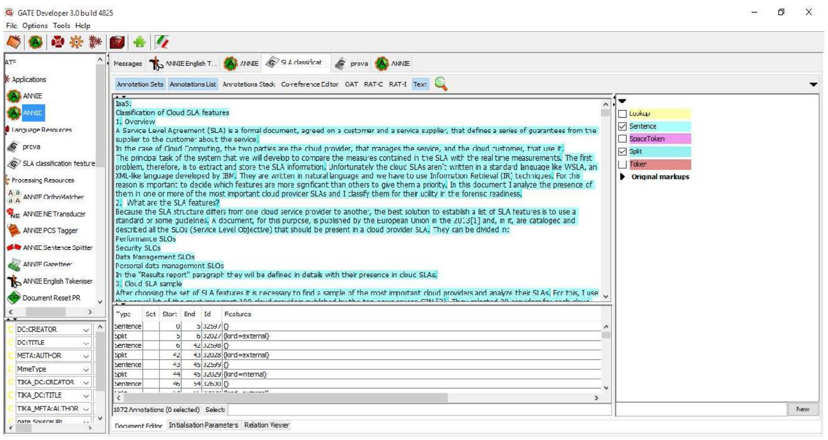
IaaS:
Classification of Cloud SLA features
1. Overview
A Service Level Agreement (SLA) is a formal document, agreed on a customer and a service supplier, that defines a series of guarantees from the supplier to the customer about the service.
In the case of Cloud Computing, the two parties are the cloud provider, that manages the service, and the cloud customer, that use it.
The principal task of the system that we will develop to compare the measures contained in the SLA with the real time measurements. The first problem, therefore, is to extract and store the SLA information. Unfortunately the cloud SLAs aren't written in a standard language like WSLA, an XML-like language developed by IBM. They are written in natural language and we have to use Information Retrieval (IR) techniques. For this reason is important to decide which features are more significant than others to give them a priority. In this document I analyze the presence of them in one or more of the most important cloud provider SLAs and I classify them for their utility in the forensic readiness.
2. What are the SLA features?
Because the SLA structure differs from one cloud service provider to another, the best solution to establish a list of SLA features is to use a standard or some guidelines. A document, for this purpose, is published by the European Union in the 2013[1] and, in it, are catalogued and described all the SLOs (Service Level Objective) that should be present in a cloud provider SLA. They can be divided in:
Performance SLOs
Security SLOs
Data Management SLOs
Personal data management SLOs
In the "Results report" paragraph they will be defined in details with their presence in cloud SLAs.
3. Cloud SLA sample
After choosing the set of SLA features it is necessary to find a sample of the most important cloud providers and analyze their SLAs. For this, I use

Type	Set	Start	End	Id	Features
Lookup		57	64	20977	{majorType=org_base}
Lookup		57	64	20978	{majorType=spur}
Lookup		87	89	20979	{majorType=country_code}
Lookup		208	210	20980	{majorType=country_code}
Lookup		279	282	20981	{majorType=time, minorType=hour}
Lookup		279	282	20982	{majorType=number}
Lookup		374	376	20983	{majorType=country_code}

564 Annotations (0 selected) Select:

Figure 8.2: Gazetteer Example

- Sentence Splitter: this component implements a Sentence Breaking technique. The Sentence Splitter divides the text in sentences and it utilises a Gazetteer composed of a list of abbreviations that help to distinguish the acronyms from the full stops ending the sentences. Every sentence is annotated with a *Sentence* type.



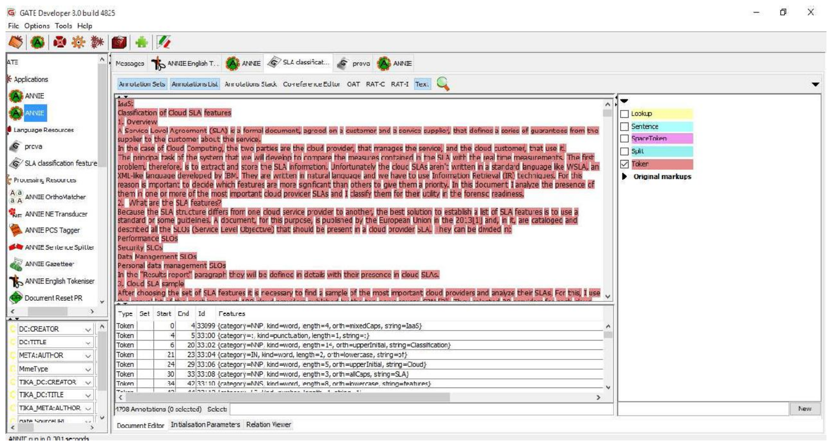
IaaS:
Classification of Cloud SLA features
1. Overview
A Service Level Agreement (SLA) is a formal document, agreed on a customer and a service supplier, that defines a series of guarantees from the supplier to the customer about the service.
In the case of Cloud Computing, the two parties are the cloud provider, that manages the service, and the cloud customer, that use it.
The principal task of the system that we will develop to compare the measures contained in the SLA with the real time measurements. The first problem, therefore, is to extract and store the SLA information. Unfortunately the cloud SLAs aren't written in a standard language like WSLA, an XML-like language developed by IBM. They are written in natural language and we have to use Information Retrieval (IR) techniques. For this reason is important to decide which features are more significant than others to give them a priority. In this document I analyze the presence of them in one or more of the most important cloud provider SLAs and I classify them for their utility in the forensic readiness.
2. What are the SLA features?
Because the SLA structure differs from one cloud service provider to another, the best solution to establish a list of SLA features is to use a standard or some guidelines. A document, for this purpose, is published by the European Union in the 2013[1] and, in it, are catalogued and described all the SLOs (Service Level Objective) that should be present in a cloud provider SLA. They can be divided in:
Performance SLOs
Security SLOs
Data Management SLOs
Personal data management SLOs
In the "Results report" paragraph they will be defined in details with their presence in cloud SLAs.
3. Cloud SLA sample
After choosing the set of SLA features it is necessary to find a sample of the most important cloud providers and analyze their SLAs. For this, I use

Type	Set	Start	End	Id	Features
Sentence		0	5	32597	{}
Split		5	6	32027	{kind=external}
Sentence		6	42	32598	{}
Split		42	43	32028	{kind=external}
Sentence		43	45	32599	{}
Split		44	45	32029	{kind=internal}
Sentence		46	54	32600	{}

1072 Annotations (0 selected) Select:

Figure 8.3: Sentence Splitter Example

- Part-of-Speech Tagger: the tagger component of GATE implements the part-of-speech tagging technique. Such a component adds to the previously obtained tokens the correct category, which has to be intended as a part-of-speech, namely a verb, a noun, a pronuon, etc. The used lexicon is derived by a training made on a big dataset of articles of the Wall Street Journal.



IaaS:
Classification of Cloud SLA features
1. Overview
A Service Level Agreement (SLA) is a formal document, agreed on a customer and a service supplier, that defines a series of guarantees from the supplier to the customer about the service.
In the case of Cloud Computing, the two parties are the cloud provider, that manages the service, and the cloud customer, that use it.
The principal task of the system that we will develop to compare the measures contained in the SLA with the real time measurements. The first problem, therefore, is to extract and store the SLA information. Unfortunately the cloud SLAs aren't written in a standard language like WSLA, an XML-like language developed by IBM. They are written in natural language and we have to use Information Retrieval (IR) techniques. For this reason is important to decide which features are more significant than others to give them a priority. In this document I analyze the presence of them in one or more of the most important cloud provider SLAs and I classify them for their utility in the forensic readiness.
2. What are the SLA features?
Because the SLA structure differs from one cloud service provider to another, the best solution to establish a list of SLA features is to use a standard or some guidelines. A document, for this purpose, is published by the European Union in the 2013[1] and, in it, are cataloged and described all the SLOs (Service Level Objective) that should be present in a cloud provider SLA. They can be divided in:
Performance SLOs
Security SLOs
Data Management SLOs
Personal data management SLOs
In the "Results report" paragraph they will be defined in details with their presence in cloud SLAs.
3. Cloud SLA sample
After choosing the set of SLA features it is necessary to find a sample of the most important cloud providers and analyze their SLAs. For this, I use

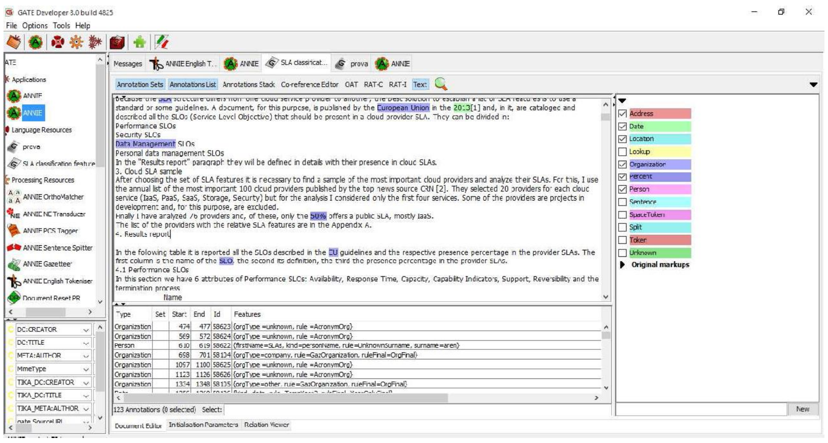
Type	Set	Start	End	Id	Features
Token		0	4	33099	{category=NNP, kind=word, length=4, orth=mixedCaps, string=IaaS}
Token		4	5	33100	{category=:, kind=punctuation, length=1, string=:}
Token		6	20	33102	{category=NNP, kind=word, length=14, orth=upperInitial, string=Classification}
Token		21	23	33104	{category=IN, kind=word, length=2, orth=lowercase, string=of}
Token		24	29	33106	{category=NNP, kind=word, length=5, orth=upperInitial, string=Cloud}
Token		30	33	33108	{category=NNP, kind=word, length=3, orth=allCaps, string=SLA}
Token		34	42	33110	{category=NNS, kind=word, length=8, orth=lowercase, string=features}

1798 Annotations (0 selected) Select:

ANNIE run in 0.201 seconds

Figure 8.4: Part-of-Speech Tagging Example

- Semantic Tagger: in the ANNIE component such a tagger is based on the JAPE language, which uses a mixture of Java code and regular expressions. This phase is responsible to produce some specific annotations enriched with additional feature, such as for instance the annotation *Person* with feature *gender* and values *male, female*; the annotation *Location* with feature *locType* and values *region, airport, city, country, county, province, other*. Some values of the features are derived by the lists used by the Gazetteer, but they can also be customised depending on the specific needs. The implemented technique is the Named Entity Recognition (NER).



standard or some guidelines. A document, for this purpose, is published by the European Union in the 2013[1] and, in it, are catalogued and described all the SLOs (Service Level Objective) that should be present in a cloud provider SLA. They can be divided in:
Performance SLOs
Security SLOs
Data Management SLOs
Personal data management SLOs
In the "Results report" paragraph they will be defined in details with their presence in cloud SLAs.
3. Cloud SLA sample
After choosing the set of SLA features it is necessary to find a sample of the most important cloud providers and analyze their SLAs. For this, I use the annual list of the most important 100 cloud providers published by the top news source CRN [2]. They selected 20 providers for each cloud service (IaaS, PaaS, SaaS, Storage, Security) but for the analysis I considered only the first four services. Some of the providers are projects in development and, for this purpose, are excluded.
Finally I have analyzed 76 providers and, of these, only the 50% offers a public SLA, mostly IaaS.
The list of the providers with the relative SLA features are in the Appendix A.
4. Results report

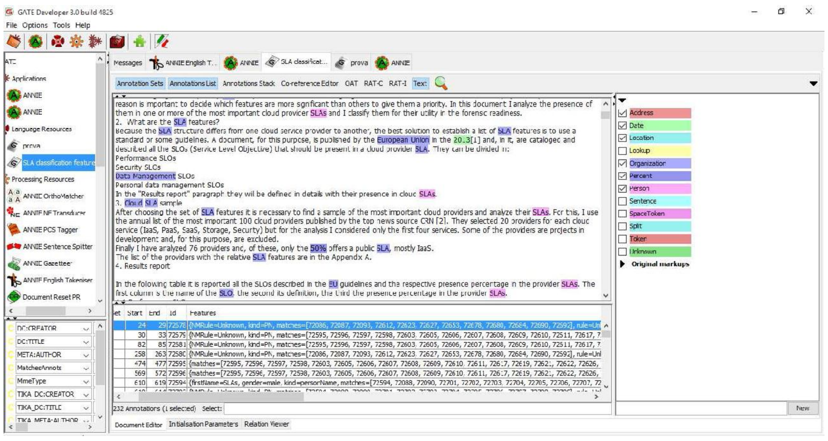
In the following table it is reported all the SLOs described in the EU guidelines and the respective presence percentage in the provider SLAs. The first column is the name of the SLO, the second its definition, the third the presence percentage in the provider SLAs.
4.1 Performance SLOs
In this section we have 6 attributes of Performance SLOs: Availability, Response Time, Capacity, Capability Indicators, Support, Reversibility and the termination process
Name

Type	Set	Start	End	Id	Features
Organization		474	477	58623	{orgType =unknown, rule =AcronymOrg}
Organization		569	572	58624	{orgType =unknown, rule =AcronymOrg}
Person		610	619	58622	{firstName=SLAs, kind=personName, rule=UnknownSurname, surname=aren}
Organization		698	701	58104	{orgType=company, rule=GazOrganization, ruleFinal=OrgFinal}
Organization		1097	1100	58625	{orgType =unknown, rule =AcronymOrg}
Organization		1123	1126	58626	{orgType =unknown, rule =AcronymOrg}
Organization		1334	1348	58105	{orgType=other, rule=GazOrganization, ruleFinal=OrgFinal}

123 Annotations (0 selected) Select:

Figure 8.5: Semantic Tagging Example

- Orthographic Co-reference (Ortho Matcher) e Pronominal Co-reference: this component implements a co-reference resolution technique. The Ortho Matcher module allows the adding of relationships to the entities identified by the Semantic Tagger. Such a module assigns a type to the annotations that have not been tagged by the Semantic Tagger. The Pronominal Co-reference module performs a check on the pronouns included in the document and their context in order to obtain the flow of references to the same entity the pronoun is referring to.



reason is important to decide which features are more significant than others to give them a priority. In this document I analyze the presence of them in one or more of the most important cloud provider SLAs and I classify them for their utility in the forensic readiness.
2. What are the SLA features?
Because the SLA structure differs from one cloud service provider to another, the best solution to establish a list of SLA features is to use a standard or some guidelines. A document, for this purpose, is published by the European Union in the 2013[1] and, in it, are catalogued and described all the SLOs (Service Level Objective) that should be present in a cloud provider SLA. They can be divided in:
Performance SLOs
Security SLOs
Data Management SLOs
Personal data management SLOs
In the "Results report" paragraph they will be defined in details with their presence in cloud SLAs.
3. Cloud SLA sample
After choosing the set of SLA features it is necessary to find a sample of the most important cloud providers and analyze their SLAs. For this, I use the annual list of the most important 100 cloud providers published by the top news source CRN [2]. They selected 20 providers for each cloud service (IaaS, PaaS, SaaS, Storage, Security) but for the analysis I considered only the first four services. Some of the providers are projects in development and, for this purpose, are excluded.
Finally I have analyzed 76 providers and, of these, only the 50% offers a public SLA, mostly IaaS.
The list of the providers with the relative SLA features are in the Appendix A.
4. Results report

In the following table it is reported all the SLOs described in the EU guidelines and the respective presence percentage in the provider SLAs. The first column is the name of the SLO, the second its definition, the third the presence percentage in the provider SLAs.

Set	Start	End	Id	Features
	24	29	72578	{NMRule=Unknown, kind=PN, matches=[72086, 72087, 72093, 72612, 72623, 72627, 72653, 72678, 72680, 72684, 72690, 72592], rule=Un
	30	33	72579	{NMRule=Unknown, kind=PN, matches=[72595, 72596, 72597, 72598, 72603, 72605, 72606, 72607, 72608, 72609, 72610, 72611, 72617, 7
	82	85	72581	{NMRule=Unknown, kind=PN, matches=[72595, 72596, 72597, 72598, 72603, 72605, 72606, 72607, 72608, 72609, 72610, 72611, 72617, 7
	258	263	72580	{NMRule=Unknown, kind=PN, matches=[72086, 72087, 72093, 72612, 72623, 72627, 72653, 72678, 72680, 72684, 72690, 72592], rule=Unl
	474	477	72595	{matches=[72595, 72596, 72597, 72598, 72603, 72605, 72606, 72607, 72608, 72609, 72610, 72611, 72617, 72619, 72621, 72622, 72626,
	569	572	72596	{matches=[72595, 72596, 72597, 72598, 72603, 72605, 72606, 72607, 72608, 72609, 72610, 72611, 72617, 72619, 72621, 72622, 72626,
	610	619	72594	{firstName=SLAs, gender=male, kind=personName, matches=[72594, 72088, 72090, 72701, 72702, 72703, 72704, 72705, 72706, 72707, 72

232 Annotations (1 selected) Select:

Figure 8.6: Co-referencing Example

After the preprocessing pipe of activities, the machine learning plug-in included in GATE is executed. It is responsible to execute a sentence classification process, aimed to classify the sentences of the document depending on the three classes described in Section 8.4. In the SLA Classification context several features are added to a customisation of such machine learning component in GATE. The input to this component is the set of sentences obtained by the Sentence Splitter. Then it has to recognise the *Class* which is an annotation added to the semantic tagger; the feature of such annotation is *type* and its values are *definition, value, not_definition*.

8.5.2 Step 2

This step is dedicated to the extraction of mathematical formulas and parameter values from the sentences classified as *Definition* and *Value*. The input to this step is the output of Step 1, namely the classified sentences of the document given as input. Then, the sentences with *feature* = *definition* OR *value* are elaborated by a dedicated transducer JAPE file, respectively. The output from this step is an additional feature to the already existing annotations.

The sentences *definition* are analysed token by token; the current token is matched with a set of mathematical keywords, e.d., adding, subtracting, divided by, rate, and so on. At the end of the process two features are extracted: the formula which is the definition of the parameter expressed mathematically; and the period, which is the time interval during which the constraint has to be valid.

From the sentences classified as *value* some other features are extracted: the numeric value of the parameter, its unit measure, a condition that determines how to compare the actual value and can be one among $\{=, <>, \leq, <, \geq, >\}$, kind that is the name of the parameter, and the value time that is the time period during which the constraint has to be valid.

8.6 Assessment

The automatic SLA classifier is assessed in order to validate the behaviour of the proposed GATE customisation so some experiments are run in order to obtain the necessary information [86]. The experiment utilises thirty-six SLAs where twenty-seven are from some Cloud providers and the remaining nine are from some SOA-based Web Services. Two different assessment have been performed, each concerns a step of the proposed automation, namely Step 1 and Step 2 described above.

8.6.1 Step 1

The SLAs documents have been used to feed the sentence splitter component of the software, and then all the identified sentences have been manually annotated with the correct class. The number of sentences among all the thirty-six SLAs classified as *definition* is 71, while the *value* sentences are 39; the total number of identified sentences is 2016, so the sentences classified as *not definition* and discarded by the assessment is 1906. A leave-one-out cross classification method has been performed with five classification algorithms, which are Support Vector Machine (SVM), Perceptron Algorithm with Uneven Margins (PAUM), Naive Bayes, K-nearest neighbour (KNN), and C4.5 decision tree algorithm. The dataset is divided in training set and test set; the training set is composed of thirty-five SLAs documents and the test set of one; the classification algorithm runs thirty-six times and the training and test sets change for every run, so in turn a different SLA document is used as test set.

At the end of the leave-one-out for each of the five algorithms some measures are calculated, namely precision, recall, and F-measure. Precision is intended as the ratio between the number of real positive results, namely the sentences correctly classified, and the sum of real positives and false positives. Recall is calculated as the ratio between the number of real positive results and the sum of real positives and false negatives, namely the sentences classified in another class but actually belonging to it. The F-measure is the harmonic average of precision and recall, calculated as

$$\frac{2*realpositives}{2*realpositives+falsepositives+falsenegatives}$$

Some parameters are set for three of the five classification algorithms; the two left unaltered are Naive Bayes and C4.5. The parameters set for SVM are cost and value. Cost (c) is the soft margin allowed for the errors; it will be in the range between $10^{-3} and 10^{3}$ in multiples of 10. Value (τ) represents the irregular margins of the classifier; when τ is set to 1 it represents the standard execution of SVM; when the training set items have a small number of positive examples and a big number of negative ones τ is set to a value lower than 1 in order to have a better F-measure; in this experiment the training set items have a small number of positive examples than negatives, so τ varies between 0 and 1 with some jumps of 0.25.

In PAUM the number of negative and positive margins can vary; they are set to the values in the following sets, respectively $\{-1.5, -1, -0.5, 0, 0.1, 0.5, 1.0\}$ and $\{-1, -0-5, 0, 0.1, 0.5, 1, 2, 5, 10, 50\}$.

In the K-nearest neighbours algorithm the parameter representing the number of neighbours can vary; the default value is set to 1, but the more such a number increases, the more the classification noise is reduced but it lows the precision; in this experiment it varies between 1 and 5.

Results

The SVM algorithm on the classes *Definition* and *Value* performs an increase of the values for precision, recall and F-measure coincident with the increase of the c parameter; they become more stable with c $>$ 1. The final result is a value for precision, recall, and F- measure lower than 60%; this is caused by some outfit that have parameter *definitions* not matching with the most common pattern, which play an influence on the classification. The *value* sentences are not so many, usually one per document, so the values for precision, and F- measure is lower than 50%, and the recall is lower than 60%.

The PAUM algorithm on the class *Definition* performs irregularly when the parameters n and p increase. The best results happen with n = -0.5 and p = 0, with precision and recall $>$ 50% and F-measure a bit lower than 50%. On the class *Value* PAUM performs better with n = 0.5 and p = 1, with values for precision, recall and F-measure lower than 40%.

K-NN on the classes *Definition* and *Value* performs worse than SVM and PAUM; it classifies worse and worse when the parameter k increases. The best result on the class *Definition* is obtained with k set to 1, but such a result is worse than SVM and PAUM due to an F-parameter value around 40%, a precision a bit lower than 50%, and a recall lower than 40% and lower than the F-measure. On the class *Value* KNN performs the best results with k = 1 and becoming stable on $\leq k \leq 4$. The best results output an F-measure, precision and recall values around 30%.

The classification algorithm that behaves better for the classes *definition* and *value* is SVM with the cost c parameter set to 1. It outputs a precision and a recall values bigger than 55% on *definition* and a bit lower than 50% for *value*. The classification algorithms without parameters, namely C4.5 and Naive-Bayes have bad behaviours, with a precision and recall = 0 , indeed they fail to classify the sentences for both *Definition* and *Value* classes

8.6.2 Step 2

This assessment phase is dedicated to the effectiveness of the transducer components customised in GATE and their capability to correctly recognise the mathematical formulas. The utilised dataset is identical to the one of the previous assessment phase, namely thirty-six SLAs documents. The training set is composed of the whole dataset items; every sentence of the documents is manually annotated with the *definition*, *value* and *not definition* classes. The mathematical formulas of the thirty-six documents are 25 up to the 71 *definitions* annotated. The assessment utilises some distance formulas to calculate the errors, which are the Levenshtein / Hamming distance metric, the Jaccard similarity metric, and the similarity cosine.

The Levenshtein distance metric represents the minimum number of elementary changes necessary to transform string A to string B, where $A \neq B$. An elementary change can be the cancellation, replacement, or insertion of a single character in A.

Both thie following analyse the sentences word by word, instead of using the character as distance unit measure. The Jaccard similarity is mainly utilised to compare similarity and diversity of sample sets. In this case the sample is a string, and the single elements are the single words composing it. The value is obtained as a ratio between the difference of the sizes of union and intersection sets of words by the size of the union set.

The similarity cosine is an heuristic technique that measures the similarity of two numeric vectors by calculating their cosine. In textual contexts, the vectors are composed of the frequency of the words considered in the strings to calculate the similarity on. The word frequency is the number of times such a word recurs in a text, so the k^{th} element of the vector represents the frequency of word k, 0 otherwise. The values of the vector elements vary in a range for 0 to +1, where +1 indicated that the both texts include exactly the same words, but not necessarily in the same order; 0 indicates that both texts have no word in common.

Also a manual semantic analysis is performed on couples of strings, in order to check if they have a different meaning in presence of a similar text.

Results

The Levenshtein distance metric performs a check character by character; the average result is a similarity value of 91% The main difference between the oracle and the actual results relies in the presence of a blank space character nearby the parenthesis symbol, which does not alter the strings. In some cases the similarity value is lower, between 70% and 80% and this is because some words are not present in the oracle, or some numbers are textually-represented, not by digits.

The Jaccard similarity metric gives an average result of 57%. The reason is due to the checks performed on single words, where a word is a sequence of characters enclosed by blank spaces. In the previous metric the presence of blank spaces was already detected, and such a presence consequently contributes to lower the average result of the Jaccard metric because a difference by a final or ending character is interpreted as the presence of two totally different strings, e.g. *month)* and *month)*.

The similarity cosine gives an average result of 71%; it also verifies the similarity of two strings at words level, but considering the word frequency instead of the set of characters composing it.

The results of the manual semantic analysis indicate that the implementation output does not differ much from the oracle. Analysing the details of the results, the main differences with the oracle concern again some blank space characters, or the representation of numbers with strings instead of digits, which do not represent a semantic difference between the two strings. The manual analysis of the results can affirm that the actual output is semantically identical to the oracle.

8.7 Summary

A classification proposal of SLAs dedicated to detect the sections of the contracts to monitor by a forensic readiness capability is the topic of this chapter. Such a classification is composed of three classes, representing a definition of a parameter to monitor, a value of such a parameter, and all the remaining sentences to discard. Then, an automation of such a classification is proposed: it is performed by a customisation of an already existing text engineering tool. At the end of the chapter an assessment of such an automation is performed utilising a dataset of thirty-six public SLAs, used to run five classification algorithms.

The following chapter is dedicated to the proposal of the formal model representing the forensic readiness capability for cloud computing. Such a model utilises formalisms as set theory, tuples, and functions in order to represent and relate the concept abstractions involved in it, e.g., SLA and cloud logs.

Chapter 9

Formal Model

9.1 Introduction

This chapter is dedicated to introduce the formal model representing the forensic readiness capability for cloud computing. Such a model utilises formalisms as set theory, tuples, and functions in order to represent and relate the concept abstractions involved in it, e.g., SLA and cloud logs. Moreover, some constraints among these concepts are represented in the form of theorems, and some specific definitions are designed.

The chapter begins with an overview of some related work concerning contractual monitoring formal representations. Then, an abstraction of the input and output of FR is designed in order to clarify its information flow. Finally, all the equations and definitions are illustrated.

9.2 Related Work

A forensic readiness system for the cloud is meant to observe and record information from the underlying computing architecture to render it forensically prepared. Such information concerns operations happening in the cloud to be related to some SLA constraints. The capability output include some important investigative details about the recorded information and the detection of contractual clauses violations.

Contractual monitoring is a topic actively investigated in the recent past in different contexts. Moreover, researchers provide customised manners to structure and represent the SLAs contents. The most effective representation is the adoption of formalisms. Then, natural language-based SLAs clauses have been structured via formal specification methods.

For instance, in [34] Czajkowski et al. focus on the design of a protocol for negotiating SLAs among several actors. Different types of SLAs are defined, and some formalisms are utilised, such as tuples for describing an SLA. Also some definitions concerning the metrics to use for services are provided.

In [120] Skene et al. formalise the SLAs by using set theory for defining the concepts of actions, actors, events, parties, actions requirements. The purpose is to determine the possible SLAs degree of monitorability in the context of services provisioning through the Internet.

In [97] a framework called Contract Log for monitoring SLAs is presented by Paschke et al., which uses several formalisms. The SLAs have been categorised depending on the purpose they have been written for. Their contractual contents are formalised with different kinds of rules, such as derivation rules, reaction rules, integrity rules and deontic rules; all of them are included in a homogeneous syntax and knowledge base. Finally, the conceptual framework has been evaluated by a tool running some specific test suites.

In [125] the concepts of parties, SLA parameters, and service level objectives are used by Unger et al. for formalizing SLAs in order to provide a manner for aggregating more SLAs in a single business process. In this formal model several formalisms are utilised, such as tuples, logic predicates, boolean algebra, and normal forms.

Instead in [57] the contracts are decomposed by Ghosh et al. in concepts of services parameters, service level objectives, and key performance indicators; all these entities are formalised via tuples. The SLAs concern a storage as a service facility where a design model for a dedicated monitoring system is provided.

In [75] formal specifications are used by Ishakian et al. for representing and transforming SLAs in order to address the issue of verifying efficient workload colocation of real-time applications. The approach allows transforming the SLAs whenever they do not meet the workload efficiency requirements into an equivalent SLA that respects the same QoS. The used formalism is the tuple. The proposal includes a reasoning tool used by the transformation rules process that comprehends inference rules based on a database of concepts, propositions, and syntactic idioms.

9.3 SLACFR

A Cloud Forensic Readiness for SLA management (SLACFR) formal model is aimed to provide a theoretical approach to structure the management of the SLA contracts for cloud computing services in the context of a forensic readiness capability. Its principal purpose is to record some information about the cloud behaviour with respect to SLAs. This information is structured as a set of comparisons between an attribute of a cloud entity and a (set of) constraint(s) on that attribute at a specific time.

The capability recognizes suspicious information in real-time: they represent a violation of a contractual constraint, such that some pre-investigative activities are executed. The input of forensic readiness is composed of both information about some cloud attributes and some SLA constraints, all represented with formal rules. The execution begins on the availability of the contract(s) to monitor. The text is properly parsed via information extraction techniques [58] and transformed into a set of formal rules.

The used approach to build this formal model follows a bottom up strategy: the contents of the SLAs discussed in Chapter 7 are decomposed and structured to represent a constraint on a cloud entity. The cloud information is gathered from service logs; they represent some resources information, and are used to compute the actual value of a specific entity. For information coming from the cloud logs, a bottom up approach is followed: the contents of the logs are decomposed and structured to represent individual cloud entities and the operations changing their values.

The formal model utilizes mathematical formalisms such as tuple, set theory, and functions [14], to represent both SLAs, cloud logs, and SLA violation detection [37–39]. The abstraction of the computation of SLA violations detection is depicted in Figure 9.1.

9.4 Cloud Computing Formal Representation

Let CA be the set of cloud architectures, it is defined by equation 9.1:

$$CA = \{ca_1, ca_2, ca_3, \dots, ca_j\}, j \in \mathbb{N} \tag{9.1}$$

A cloud architecture ca is composed of at least two data centres, then

$$ca = \{D^{ca} \subseteq D : |D^{ca}| \geq 2\}$$

where D is the set of data centres, represented by equation 9.2

$$D = \{d_1, d_2, d_3, \dots, d_j\}, j \in \mathbb{N} \tag{9.2}$$

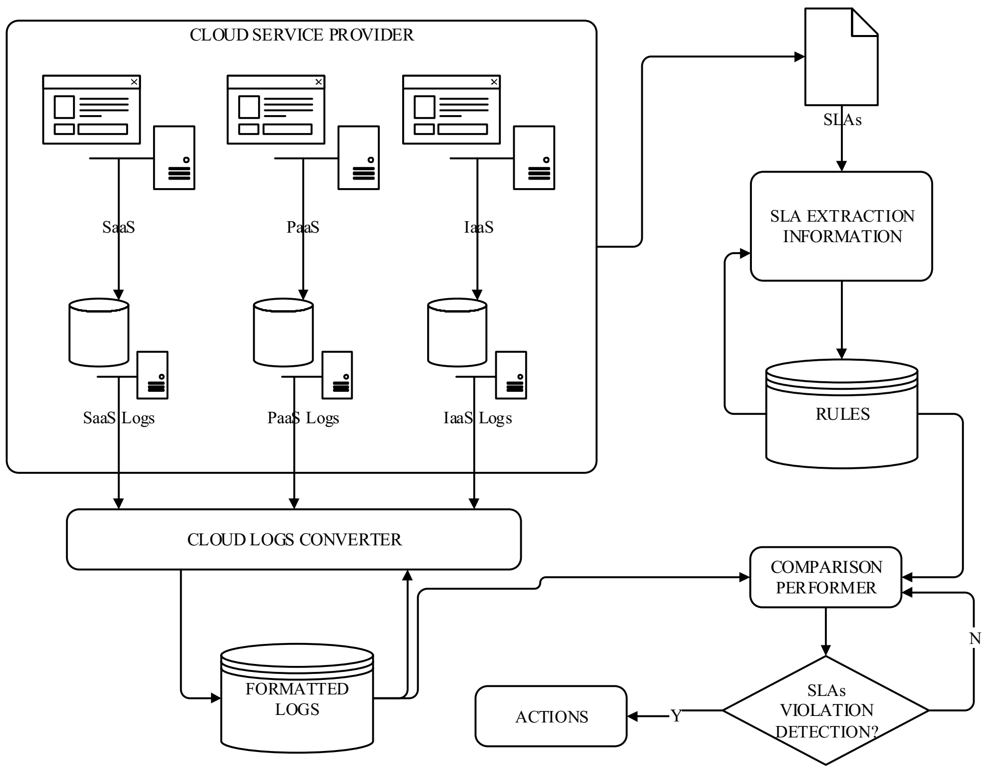


Figure 9.1: SLACFR Capability [38]

A data centre d is an entity containing one or more machines, which can be both physical and virtual. The virtual machines are lying above a physical machine, and managed by a virtual machine monitor. In a Cloud architecture ca more data centres are connected each other, such that they can compose a network. Let d be a data centre belonging to the set of data centres D^{ca}, it is described by the tuple in equation 9.3

$$\begin{gathered} d = \langle P^d, V^d, vmm, N^d \rangle \qquad (9.3) \\ P^d \subseteq P \\ V^d \subseteq V \\ vmm \in VMM \\ N^d \subseteq CN \end{gathered}$$

where P is the set of physical machines, V the set of virtual machines,VMM is the set of virtual machine monitors, and N^d is the set of nodes that d is connected with. N^d is subset of the set of all the connections CN. Each element of the set N^d describes the connected data centres.

$$\begin{gathered} CN = \{n_1, n_2, n_3, \ldots, n_j\}, j \in \mathbb{N} \\ n = \langle d_1, d_2 \rangle \end{gathered} \tag{9.4}$$

Theorem 1. In a cloud computing architecture c there exist at least two data centres such that $|CN| \neq 0$.

Proof. By definition $|D^{ca}| > 1$, then at least $D^{ca} = \{d_1, d_2\}$,
where $d_1 = \langle P^{d_1}, V^{d_1}, vmm_{d_1}, N^{d_1} \rangle$ and $d_2 = \langle P^{d_2}, V^{d_2}, vmm_{d_2}, N^{d_2} \rangle$
$N^{d_1} = \{n_1 | n_1 = \langle d_1, d_2 \rangle\}$
$N^{d_2} = \{n_2 | n_2 = \langle d_2, d_1 \rangle\}$
$N^{d_1} \subseteq CN$ and $N^{d_2} \subseteq CN \Rightarrow CN = N^{d_1} \cup N^{d_2} \Rightarrow CN = \{n_1, n_2\} \Rightarrow |CN| \neq 0$. □

Both physical and virtual machines can be composed as a set of resources R, either software or hardware, as described by equation 9.5.

$$\begin{gathered} P = \{R^p | R^p \subseteq R\} \\ V = \{R^v | R^v \subseteq R\} \\ R = \{r_1, r_2, r_3, \ldots, r_j\}, j \in \mathbb{N} \end{gathered} \tag{9.5}$$

A resource r is described by a set of attributes A, e.g., filename, date of creation, size. Let a be an attribute, it belongs to the set of attributes A as described by equation 9.6.

$$A = \{a_1, a_2, a_3, \ldots, a_j\}, j \in \mathbb{N} \tag{9.6}$$

$$r = \{A^r | A^r \subseteq A\} \tag{9.7}$$

$$|A^r| \neq 0 \forall r \in R$$

Theorem 2. The set of attributes A cannot be empty.

Proof. Every resource $r \in R$ is described by a set of attributes A^r which is subset of A.

$$A = \bigcup A^r_{\forall r \in R} \Rightarrow |A| \leq \bigcup |A^r|$$

$$\overset{bydef.}{\Rightarrow} |A^r| \neq 0 \forall r \in R$$

$$\Rightarrow |A| \neq 0$$

□

During the execution of a cloud service, the value of an attribute of a resource is subject to change via an operation o. An operation o is an element of the set of operations O. Each operation is described by a mathematical tuple composed of a sender s that is the executor of the operation, a result $value(a^r)$ that describes the value assigned to attribute a of resource r, an operation resource r, an attribute a, and an operation time t_o. The operation value is a mathematical function that assigns a value to an attribute of a resource. The assigned value can be either a numeric or textual. A sender s is an entity performing operations in the Cloud, either a human or a system process, e.g., an Internet session or an application process. Let s be a sender, it belongs to the set of senders S.

$$S = \{s_1, s_2, s_3, \ldots, s_j\}, j \in \mathbb{N} \tag{9.8}$$

$$O = \{o_1, o_2, o_3, \ldots, o_j\}, j \in \mathbb{N} \tag{9.9}$$

$$o = \langle s, a^r, value(a^r), u, t_o \rangle \tag{9.10}$$

$$u \in U$$

$$value : a^r \to value(a^r) = k \in \mathbb{Z} \cup TEXT$$

$$TEXT = \{k | k \notin \mathbb{Z}\}$$

The information about the flow of operations are stored in a cloud log *cl*, which is an element of the set of cloud logs *CL*. Every log is composed of a set of operations O^{cl} concerning an attribute *a* of a resource *r*; the set O^{cl} is included in the set of all the operations *O*.

$$CL = \{cl_1, cl_2, cl_3, \ldots, cl_j\}, j \in \mathbb{N} \tag{9.11}$$

$$cl = O^{cl}$$

$$O^{cl} \subseteq O$$

9.5 SLA Formal Representation

An sla *l* is an element of the set of slas *L*. An sla is described by a mathematical tuple composed of a set of service levels *SL*, the validity starting time t_{start} and the validity ending time t_{end}.

$$L = \{l_1, l_2, l_3, \ldots, l_j\}, j \in \mathbb{N} \tag{9.12}$$

$$l = \langle SL, t_{start}, t_{end} \rangle$$

$$t_{end} \geqslant t_{start}$$

A service level *sl* is an element of the set of service levels *SL*. Each *sl* has a validity time interval, determined by a starting and an ending time; during this time, some indicators related to a service level attribute for a specific service resource need to be verified, hence an *sl* is described by the following tuple, composed of the set of indicators *I*, the attribute *a* of the resource *r*, and the starting and finishing times, t_s and t_f respectively.

$$SL = \{sl_1, sl_2, sl_3, \dots, sl_j\}, j \in \mathbb{N} \tag{9.13}$$
$$sl = \langle I, a^r, t_s, t_f \rangle$$
$$t_f \geqslant t_s$$

An indicator *i* is an element of the set of the indicators *I*. An indicator is described by a mathematical tuple composed of a conditioned value $c\ k\ u$ of the attribute $a^{r^{sl}}$ for the resource r. A condition c belongs to the set of conditions C; in case any condition is not expressed in the contractual text, the value of *c* will be set as $=$. A value *k* is related to the attribute $a^{r^{sl}}$ of the resource *r*; *u* is the optional unit measure used to express the value *k*, belonging to the set of unit measures U. The conditioned value $c\ k\ u$ has to be verified through the application of the metric *m* belonging to the set of metrics M. The metrics can be either atomic or composed, namely the value is obtained by combining more atomic metrics.

$$I = \{i_1, i_2, \dots, i_j\}, j \in \mathbb{N} \tag{9.14}$$
$$i = \langle cku, m \rangle$$

$$c \in C$$
$$C = \{\leq; \geq; <; >; <>; =\} \tag{9.15}$$

$$u \in U$$
$$U = \{u_1, u_2, \dots, u_j\}, j \in \mathbb{N}$$

$$m \in M$$

$$M = \{m_1, m_2, \dots, m_j\}, j \in \mathbb{N}$$

9.6 Forensic Readiness

A cloud forensic readiness capability f is an element of the set of forensic readiness capabilities F. A capability is described by a mathematical tuple composed of a set of comparisons Q and a set of SLAs L, where L is described in the previous section, while Q is defined in the following to illustrated how a comparison is intended.

$$\begin{gathered} F = \{f_1, f_2, f_3, \dots, f_j\}, j \in \mathbb{N} \\ f = \langle Q, L \rangle \\ Q \neq \varnothing \\ L \neq \varnothing \end{gathered} \tag{9.16}$$

A forensic readiness capability is responsible to perform a comparison between an indicator of an attribute for a resource expressed in a service level $i \in I^{sl}$ and a set of operations of a specific cloud log O^{cl} where the sequence of values of this attribute is contained. The comparison is evaluated according to the metric $m^{i^{sl}}$ used to calculate the value. Formally, these entities are correlated in the following tuple. A comparison q belongs to the set of comparisons Q. A comparison tuple is composed of a cloud log cl, an indicator that has to be verified $i \in I^{sl}$, and the comparison time t_q.

$$Q = \{q_1, q_2, q_3, \dots, q_j\}, j \in \mathbb{N} \tag{9.17}$$
$$q = \langle cl, i^{sl}, t_q \rangle$$
$$t_q \geq t_{o^{cl}}$$
$$\forall o \in O^{cl}, \forall i \in I^{sl}, (a^r)^{O^{cl}} = (a^r)^{I^{sl}}$$

Definition 1. SLA Violation
Given a comparison $q \in Q$ and a service level $sl \in SL$, the comparison is considered an SLA violation if the values of the set of operations O^{cl} composing the cloud log cl on the attribute a of the resource r at the time t are different from the conditioned value cku of the related indicator i^{sl} about the service level sl, on the same attribute a of the same resource r. The validity has to be determined during the correct time interval, namely the time of the operations have to be included in the time interval $t_f(i) - t_s(i)$; the value $value(a^r)$ is obtained by the application of the metric m^i.

$$m^i(a^r) = value(a^r) \neq (cku)^i \tag{9.18}$$
$$t_s(i) \leq t_{value(a^r)} \leq t_f(i) \tag{9.19}$$

9.7 Summary

The formal model representing the forensic readiness capability for cloud computing is the main topic of this chapter. Such a model utilises formalisms as set theory, tuples, and functions in order to represent and relate the concept abstractions involved in it, e.g., SLA and cloud logs. Moreover, some constraints among these concepts are represented in the form of theorems, and some specific definitions are designed.

The chapter begins with an overview of some related work concerning contractual monitoring formal representations. Then, an abstraction of the input and output of FR is designed in order to clarify its information flow. Finally, all the equations and definitions are illustrated.

The following chapter is dedicated to the description of the prototype system; its architecture is designed, together with some design goals. The databases structure is described and represented with some UML diagrams. Also the interaction with the text extraction tool is presented, with details about the manner in which the information is gathered and recorded. Then, the core module routines are illustrated, together with the interactions with the utilised existing libraries.

Chapter 10

Prototype System

10.1 Introduction

This chapter is dedicated to the description of the prototype system; its architecture is designed, together with some design goals. The databases structure is described and represented with some UML diagrams. Also the interaction with the text extraction tool is presented, with details about the manner in which the information is gathered and recorded. Then, the core module routines are illustrated, together with the interactions with the utilised existing libraries.

10.2 Architecture

In this chapter the prototype for cloud forensic readiness for SLA violation detection system is discussed [39]. The objective of the prototype is to perform comparisons between SLA constraints and real-time cloud services resources attributes measures. The prototype implements the comparison performer module depicted in Figure 9.1, which obtains the input from external tools it is interfaced with [87]. A general prototype system architecture is depicted in Figure 10.1, whose components are detailed in the sub-system decomposition diagram in Figure 10.2; finally, the hardware / software mapping is depicted in the diagram in Figure 10.3. Both Log Indexing and Extraction and SLA Extraction components are dedicated to the structuring of both cloud logs and SLAs information respectively. SLA documents are manipulated with Information Extraction techniques automated with the software GATE as described in Chapter 8, and the indicators described in equation 9.14 in Section 9.5 are generated. Pseudo real cloud log information is used to structure the cloud log entities of equation 9.11 described in Section 9.4. Such information is stored in the Storage component, and the Matching component is responsible for performing the comparisons between the two inputs, and generating information structured as described by equation 9.17. The Values Matcher module is responsible for computing the values of some resources attributes gathered from the Storage module. The computation is performed via specific metrics, obtained by information from real SLAs and stored in the Storage module in the format described in equation 9.5. The measures are obtained by the Formula Evaluator component, and compared with the related SLA constraints, storing them persistently. The Storage module includes all the functions used to manage the databases.

10.3 Design Goals

In this section, some design goals of the prototype system are discussed [87].

- Performance
  - Response time: the system compares the constraints contained in the SLA with the log information in real-time and according to the deadlines foreseen by the SLAs constraints (e.g., Monthly Uptime Percentage is verified every month);

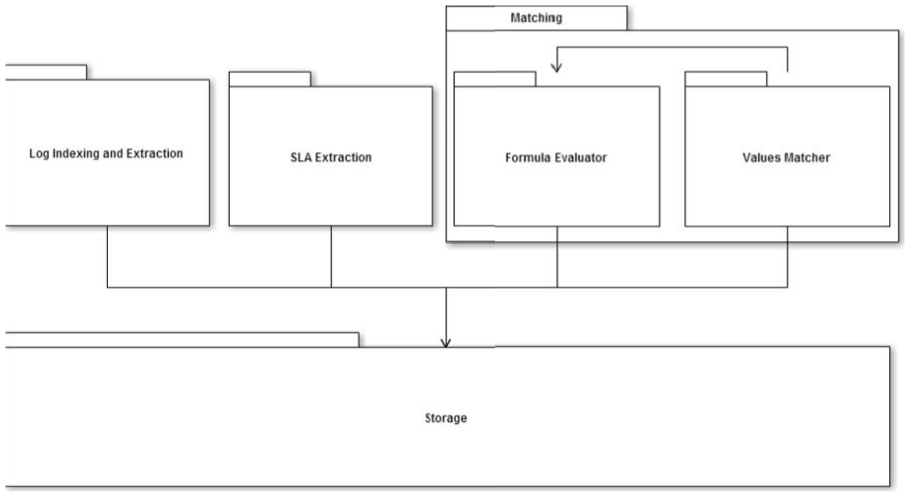


Figure 10.1: SLACFR System Architecture

- Memory: the system does not require a big amount of RAM capacity, because most of the generated data are stored in databases.

- Reliability
  - Robustness: the system manages possible user input errors, like configuration errors (e.g. a non-existent file path into configuration file);
  - Effectiveness: the system is effective during the start - stop time period determined by the user; it performs without errors in order to recognize all possible SLA violations;
  - Fault tolerance: errors can be generated by hardware and software faults, such as flaws of the two DBMS (MySQL and MongoDB), and hardware problems like lack of persistent storage memory.

- Maintenance
  - Portability: the system can run on every operating system and platform;

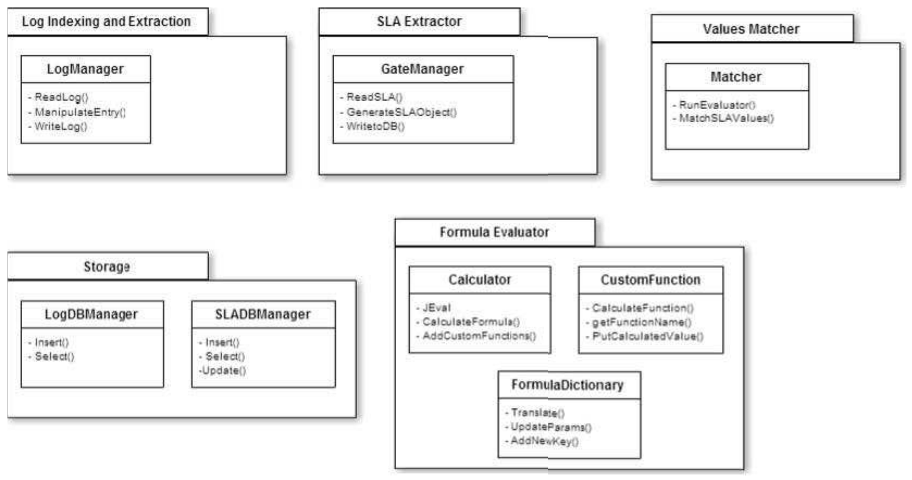


Figure 10.2: SLACFR Sub-Systems Decomposition

  - Legibility: the code is furnished with simple but exhaustive comments, facilitating the impact analysis in case of change requests during maintenance activities.

- End user
  - Usability: the system intuitive, furnished with a user manual and proper on-line help.

- Persistent data management
  - DBMS: both a relational and non-relational DBMS are utilised, MySQL and MongoDB, respectively. MySQL stores data extracted from the SLAs, because such data are composed of entities in relation, i.e, SLOs with values and definitions. MongoDB is used for cloud logs and for the output of the Values Matcher system components. All this data have no relations, and can be stored in simple query-able collections.

- Access control and security

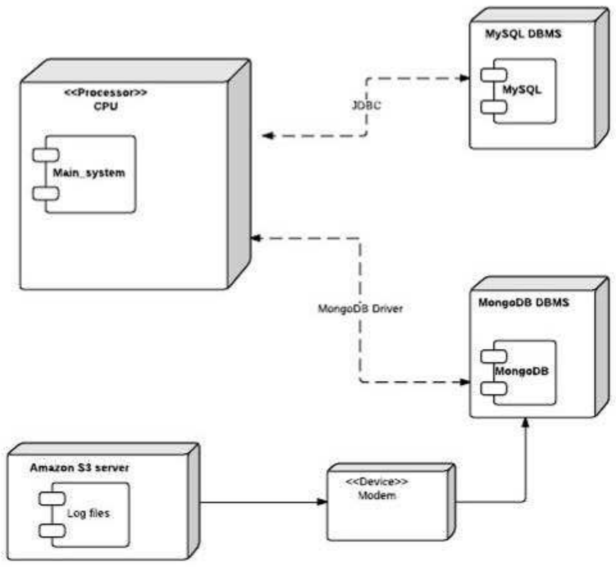


Figure 10.3: SLACFR HW / SW Mapping

- Access matrix: there is only a single user in every instance of the system, and the interaction is limited to start and stop the system, and to receive notification about contract violations detections. An access matrix is not necessary;
- Data protection: it is guaranteed via by MySQL and MongoDB credentials;

• Boundary conditions

- Start and stop: the single system user has to set up a series of parameters in a configuration file, and then he launches the executable file. Such a forensic readiness system theoretically should never stop, but if the user prefers, he can interrupt the system, and the system stores all the information calculated before being stopped;
- Exception management: possible runtime exceptions are managed by

the system with user-friendly error messages graphical interfaces explaining plain errors to the user.

10.4 Databases

10.4.1 SLA Database

The SLA database model is illustrated in this section [87]. In Figure 10.4 is depicted the UML Entity-Relation model, and a logic model follows. Every SLA is formed by *n* SLOs and *n* Definitions. Every SLO can have more than one value in a single SLA, e.g, multiple services with different definition for uptime; and for every SLOValue there is a different definition. The logic model of the SLA database is composed of four tables, and their primary keys are in italic:

SLA (*ID*, provider, service)

SLO (*name, id_SLA*)

Definition (*ID_Sla, name*, form, period, unit, valuetime, service);

SLOValue (*name, id_SLA*, valuetime, value, operator, unit, value_service, definition_name)

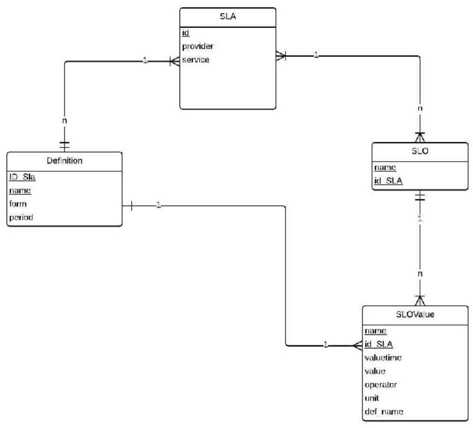


Figure 10.4: SLACFR ER Diagram

10.4.2 Log Database

The log database is managed by a non-relational DBMS, specifically MongoDB. This choice is driven by the following considerations. The number of entries can be huge and query on a big file could be really difficult for the system purposes. Moreover, in order to calculate real values to match with contractual values, a large number of queries has to be launched, sometimes in a short time period. MongoDB is one of the most popular non relational databases, and it provides some useful features, such as flexible data model: data can have any structure and it can be dynamically modified. Indeed, this feature fits perfectly with the system necessities, because the logs can have different structures and information depending on the log sources. Again, MongoDB is highly scalable, from a single server to thousands of nodes. It can deploy in the cloud and across multiple data centres. MongoDB provides a query language that allows various query types; the availability of drivers for any programming language facilitates the queries in a really easy way. Database initialisation and population happens via dedicated Java modules of the prototype system. The module retrieves the log file entries and stores them directly into a MongoDB collection.

10.4.3 Comparison Database

The information generated by the system are structured in an Entity - Relation database, called Comparison Database. It is composed of three tables, representing the calculated data, the detected contractual violations, and the detected attacks. The first table stores information about the calculated SLO, such as the name, the obtained value, and the time the computation happens. The second one is about the contractual violations detected by the system; the information is composed of the name of the violated SLO, the difference between the real value and the value contained in the SLO, the SLO unit of measure, and the time the violation is detected. The attack table stores data about the identified security attack, calculated on the basis of the detected violations; the table stores both name and time of the identified attack.

Calculated Data(Type, Value, Time)

Detected Violations(SLO, Difference, Unit, Time)

Attack(Name, Time)

10.5 Interaction with GATE

GATE tool is interfaced with the SLACFR prototype via GATE APIs. After the classification and the information extraction has been performed by GATE, the output is transformed into objects, and the features into class variables. There exist five Java classes executing this data exchange: the class Gate_main loads GATE and the ANNIE plug-in; then it launches the sub-applications in the pipeline, including the personalised transducers. Then, two objects are instantiated, one of class DefinitionSet, composed of a set of Definition objects, and one of class SLA, composed of a set of object SLO and the previous DefinitionSet. The variables of the class Definition are the corresponding annotation features. All the objects are instantiated and stored in the SLA relational DB, by a dedicated Java module that communicates with the storage component. In this way GATE will be called only when the cloud SLA is modified. During a normal execution, main_system obtains SLA information directly from the database, using the provided interface classes. The UML Class Diagram of the *gate* system package, interfacing with the GATE tool, is depicted in Figure 10.5.

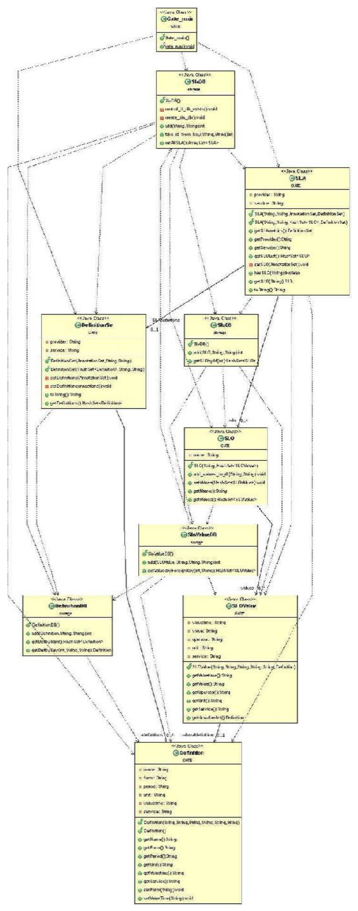


Figure 10.5: UML Class Diagram for the SLA Extraction System Module

10.6 Matching Component

In this section the matching engine that performs the contractual violation detections is described. Such an engine is composed of a set of modules performing: data aggregation and indexing, comparison, and eventual alert. An UML Class Diagram is depicted in Figure 10.6. The engine obtains input from the SLAs and log databases, then it calculates the actual SLOs values and compares them to the values contained in the SLAs object. The system starts initializing the MongoDB log database and recovering all SLA objects stored into the SLA database. The matching operation is launched for each SLA.

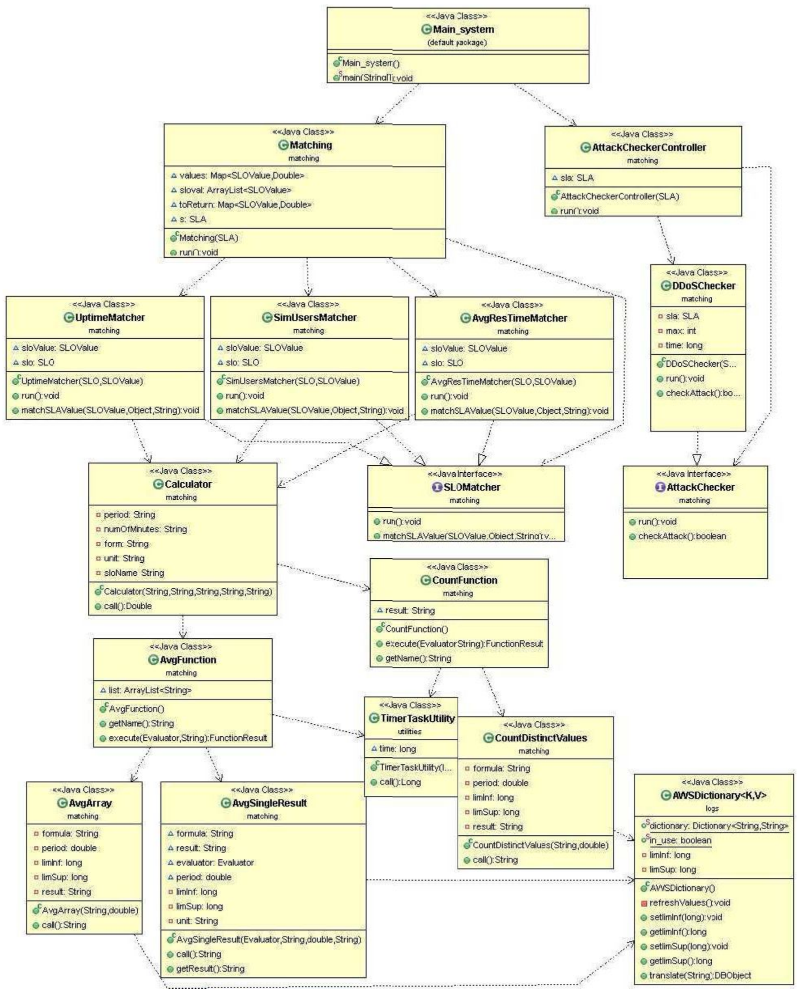


Figure 10.6: UML Class Diagram for the Matching System Module

In order to calculate the actual values, some SLO formulas have to be utilised. The formulas are stored in the *form* field of the *definition* table. Every definition is related to its SLO with a foreign key, i.e., definition_name in the SLO table. A mathematical formula is needed by the matcher in order to compute the SLO numeric value. A formula is one or more mathematical operations applied to one or more variables and / or numerical constants. The system translates a variable in a query, then it retrieves its numerical value querying some logs information.

An external library called JEval [64] performs the formula evaluation. It gives a mathematical formula as input to an evaluator that parses and calculates the corresponding value. In order to translate each atomic variable of the query, a dictionary is implemented; it maps relations between such atomic variables and a query .

Every SLO has its own time window based on the evaluation period, as specified into the SLA. The parameters of the queries are the upper and the lower bounds of the time value. Time value is expressed in milliseconds, as a Java Calendar Time. This parametrisation is necessary to have the right result set based on time windows.

To use the dictionary to retrieve the right query, it is necessary to invoke methods setLimInf(Long time) e setLimSup(Long time) to update the upper and lower bounds. Then, the method translate(String key) returns the query matching with the given atomic variable, namely the dictionary key.

The following line of code is an example of the population of a dictionary entry, e.g., the Error Code parameter used to calculate the MUP for Amazon S3.

dictionary.put("error "InternalError" or "ServiceUnavailable" ", " $and : [$or: [ErrorCode : "InternalError ", ErrorCode : "ServiceUnavailable "], $and : [Time: $gte : "+limInf+", Time:$lte : "+limSup+"]] ");

The dictionary is implemented as a static Java class, then it needs an access policy from each running thread launched by the system. This policy has been realised using a lock on the static class: before using the dictionary, every thread checks a variable containing this lock, and then the access can be granted or denied.

This mapping is very difficult to be performed automatically, because very sophisticated textual manipulation techniques are necessary. In this prototype it is performed manually, and updated as soon as additional information is included. Nevertheless, most of the entries can be valid for every service, such as HTTP messages name and code. In Figure 10.7 some flow charts about the calculation of the actual values are depicted. The diagram on the left hand side represents a generic formula evaluation; the middle diagram depicts the calculation of an average formula; and the right hand side diagram describes a counting formula.

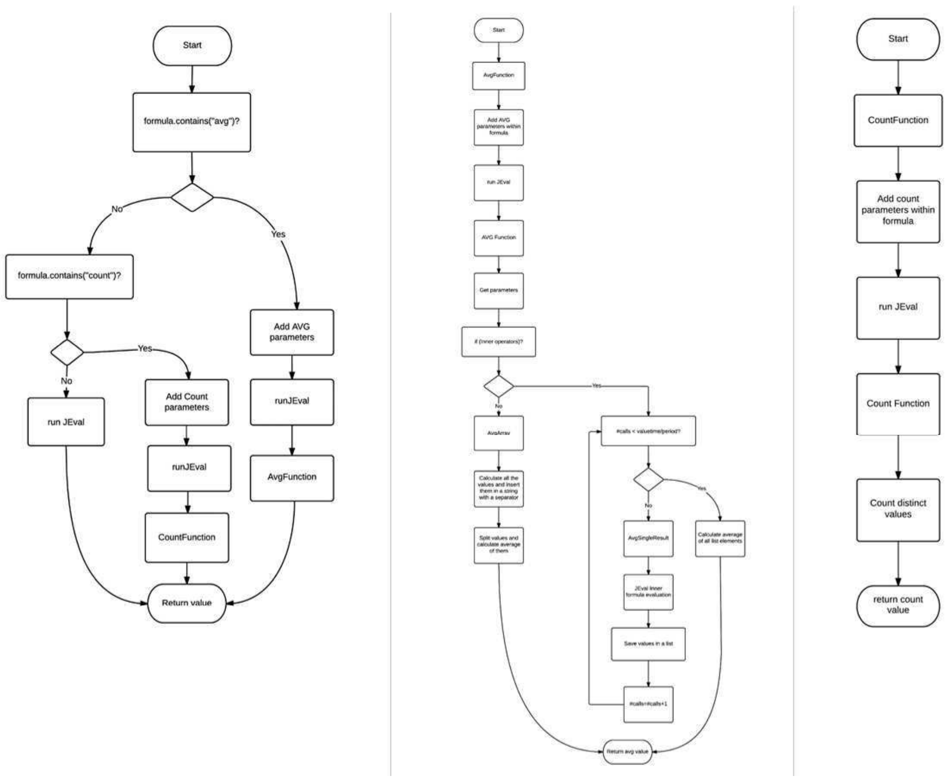


Figure 10.7: Flow Chart Diagrams for Resources Attributes Values Calculation

10.7 Summary

The description of the prototype system is the topic of this chapter; its architecture is designed, together with some design goals. The databases structure is described and represented with some UML diagrams. Also the interaction with the text extraction tool is presented, with details about the manner in which the information is gathered and recorded. Then, the core module routines are illustrated, together with the interactions with the utilised existing libraries.

The following chapter is dedicated to the description of a case study utilised in this dissertation in order to execute the prototype system. Some motivations and a description of some assumptions open the chapter, then a scenario is presented. Subsequently, the SLA and the logs of the chosen cloud platform are detailed, and the customisation of the proposed cloud forensic readiness formal model on it is illustrated.

Chapter 11

Case Study

11.1 Introduction

This chapter is dedicated to the description of a case study utilised in this dissertation in order to execute the prototype system. Some motivations and a description of some assumptions open the chapter, then a scenario is presented. Subsequently, the SLA and the logs of the chosen cloud platform are detailed, and the customisation of the proposed cloud forensic readiness formal model on it is illustrated.

11.2 Overview

It is well documented that the amount of information accessible by cloud service customers is large in IaaS level, medium in PaaS, and small in SaaS (see Figure 2.1). Thus, the choice of an IaaS cloud service suits the requirements of this section, because it grants the access to a bigger amount of information than in the other two service models.

The choice has been on Amazon Simple Storage Service (S3) [9], which is an Infrastructure as a Service (IaaS) provided by AWS. It owns a public SLA, including several constraints. They are composed of definitions, values, and formulas of the parameters necessary to be calculated by the proposed prototype system. Nevertheless, also a detailed description of the service access log files is provided by Amazon S3. This has been the leader to such a choice, namely the availability to be aware of the structure of the data utilised to calculate the actual value of the SLA parameter to monitor, in order to perform the comparisons.

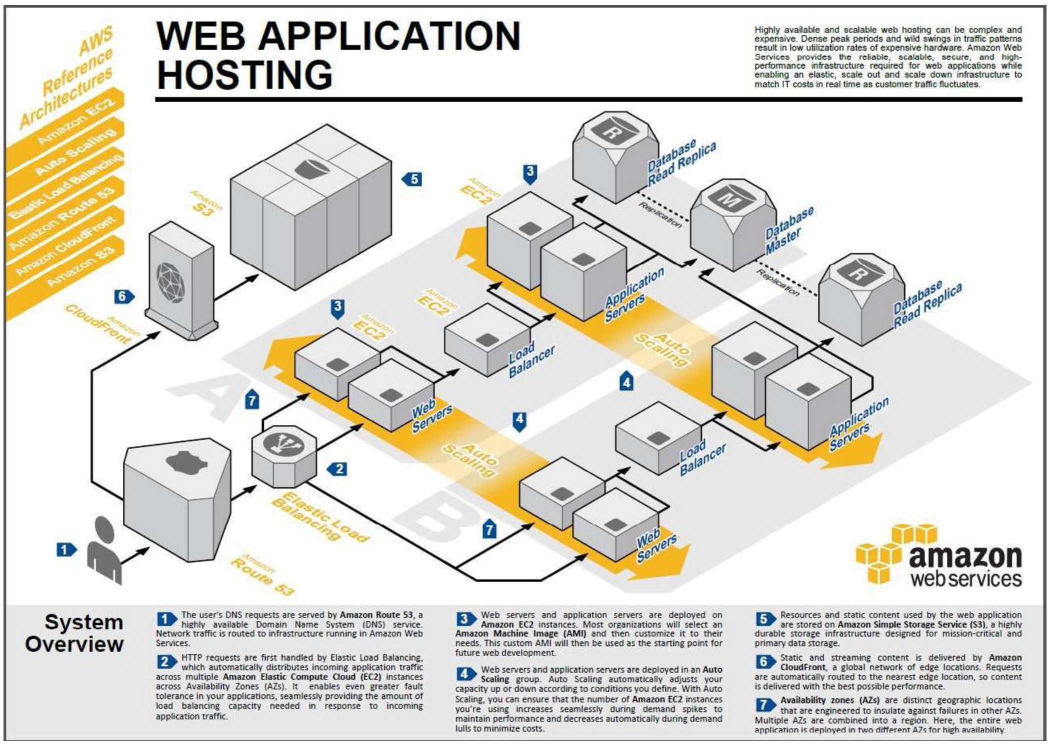
It is worth to mention that some other options have been considered but then discarded, because of several reasons: the lack of a public SLA but the presence of a description of the log file structure; the presence of both elements, but the difficulty to apply the formula necessary to calculate the actual values of the parameters because some specific and strongly technical details hidden to customers were required; the presence of SLAs but the lack of a description useful to interpret the log files.

The following section explains the scenario of a typical situation were the proposed formal model and prototype system play a significant role.

11.3 Scenario

Bob is an administrator of a wiki website; he needs the contents of his website to be available to the public 24/7; moreover he would like that the platform hosting the mentioned wiki website is scalable, flexible, and accessible from every country. To full-fill these requirements, Bob decides to host his website on a cloud platform, but he is aware of the possibility of data security issues. Thus, Bob would like the hosting platform to be capable of generating alerts when a security attack takes place, to facilitate some forensic activities to verify the platform behaviour is in line with the contract he signed. Assuming that some cloud services are provisioned with a forensic readiness capability, Bob will chose one of them.

The Amazon Web Services (AWS) platform is chosen by Bob because it suits his needs. It provides an external system performing forensic readiness activities. Bob selects a standard utilisation of AWS regulated by a public SLA. Bob uses his Elastic Compute Cloud (EC2) instances to run his wiki website. The data are stored using the Relational Data Base Service, included in the AWS pool. The static website contents, like HTML and CSS pages, images, video, are located on the Simple Storage Service (S3). Figure 11.1 depicts the AWS hosting logical architecture used by Bob to store all his contents



Highly available and scalable web hosting can be complex and expensive. Dense peak periods and wild swings in traffic patterns result in low utilization rates of expensive hardware. Amazon Web Services provides the reliable, scalable, secure, and high-performance infrastructure required for web applications while enabling an elastic, scale out and scale down infrastructure to match IT costs in real time as customer traffic fluctuates.

1 The user's DNS requests are served by **Amazon Route 53**, a highly available Domain Name System (DNS) service. Network traffic is routed to infrastructure running in Amazon Web Services.

2 HTTP requests are first handled by Elastic Load Balancing, which automatically distributes incoming application traffic across multiple **Amazon Elastic Compute Cloud (EC2)** instances across Availability Zones (AZs). It enables even greater fault tolerance in your applications, seamlessly providing the amount of load balancing capacity needed in response to incoming application traffic.

3 Web servers and application servers are deployed on **Amazon EC2** instances. Most organizations will select an **Amazon Machine Image (AMI)** and then customize it to their needs. This custom AMI will then be used as the starting point for future web development.

4 Web servers and application servers are deployed in an **Auto Scaling** group. Auto Scaling automatically adjusts your capacity up or down according to conditions you define. With Auto Scaling, you can ensure that the number of **Amazon EC2** instances you're using increases seamlessly during demand spikes to maintain performance and decreases automatically during demand lulls to minimize costs.

5 Resources and static content used by the web application are stored on **Amazon Simple Storage Service (S3)**, a highly durable storage infrastructure designed for mission-critical and primary data storage.

6 Static and streaming content is delivered by **Amazon CloudFront**, a global network of edge locations. Requests are automatically routed to the nearest edge location, so content is delivered with the best possible performance.

7 **Availability zones (AZs)** are distinct geographic locations that are engineered to insulate against failures in other AZs. Multiple AZs are combined into a region. Here, the entire web application is deployed in two different AZs for high availability

Figure 11.1: Amazon Web Services Reference Architecture [9]

11.4 Amazon SLA and Logs

Amazon S3 is the chosen cloud computing service that is furnished with an external forensic readiness capability. It provides a public SLA and a detailed description of the service access log file structure. Among all the legal constraints of the SLA, the service level uptime is considered in this case study because it is the most present SLO in the public SLAs, as analysed in Chapter 7.3.1 with a percentage of presence of 84.2%, and classified as the first primary SLO (see Table 7.7).

The service level uptime in Amazon S3 is expressed as *Monthly Uptime Percentage*. This parameter becomes a quality attribute of the Amazon S3 cloud service. *Amazon Web Services will use commercially reasonable efforts to make Amazon S3 available with a Monthly Uptime Percentage [...] of at least 99.9% during any monthly billing cycle.*

In the SLA also the metric to calculate the attribute value is included: *Monthly Uptime Percentage is calculated by subtracting from 100% the average of the Error Rates from each five minute period in the monthly billing cycle.* Again, Amazon carefully defines the aforementioned Error Rates as *the total number of internal server errors returned by Amazon S3 as error status InternalError or ServiceUnavailable divided by the total number of requests during that five minute period* [9].

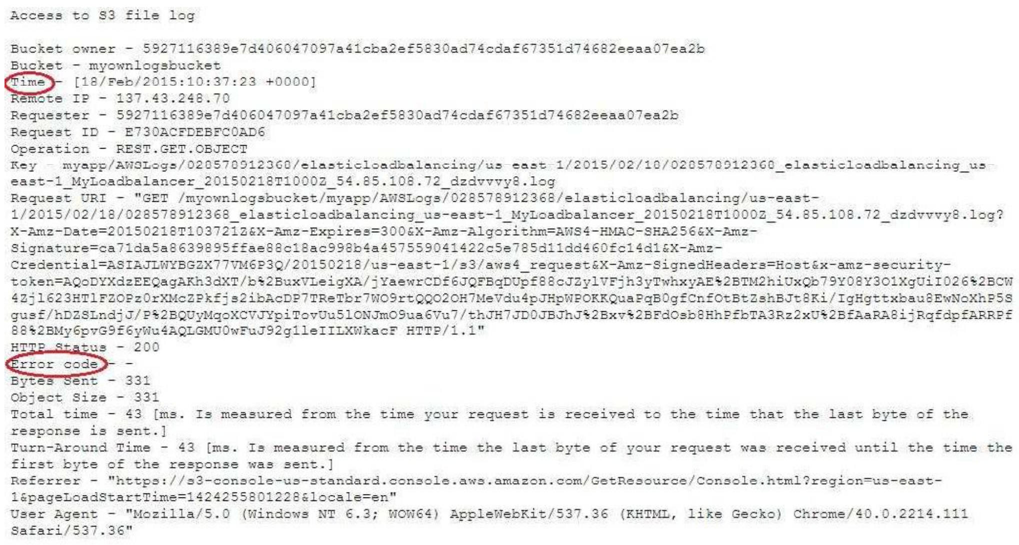
In order to guarantee the respect of the monthly uptime percentage service level, an SLACFR capability has to consider the information about server responses to the HTTP requests made to S3 every five minutes. Such data are available in the server access log files available in S3, which collects some information every request made to the service. All the fields in an S3 log entry are space-delimited; an example of such a log is shown in the following line.

79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be mybucket [06/Feb/

2014:00:00:38 +0000] 189.48.46.51 - 3E57427F33A59F07 WEBSITE.GET.OBJECT /photos/2014/08/puppy.jpg "GET /mybucket/photos/2014/08/puppy.jpg?x-foo=bar" 200 - 14200 14200 15 15 "http://www.puppypicturest.com/" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/36.0.1985.143 Safari/537.36" -

As mentioned in the AWS S3 documentation page, any field can be set to "-" to indicate that information is unknown or unavailable, or that this field is not applicable to the request.

An example of the structure of such logs is provided in Figure 11.2. The information in circles is necessary to calculate the Error Rates, i.e., the time and the Error Code.



```
Access to S3 file log

Bucket owner - 5927116389e7d406047097a41cba2ef5830ad74cdaf67351d74682eeaa07ea2b
Bucket - myownlogsbucket
Time - [18/Feb/2015:10:37:23 +0000]
Remote IP - 137.43.248.70
Requester - 5927116389e7d406047097a41cba2ef5830ad74cdaf67351d74682eeaa07ea2b
Request ID - E730ACFDEBFC0AD6
Operation - REST.GET.OBJECT
Key - myapp/AWSLogs/028578912368/elasticloadbalancing/us-east-1/2015/02/18/028578912368_elasticloadbalancing_us-
east-1_MyLoadbalancer_20150218T1000Z_54.85.108.72_dzdvvvy8.log
Request URI - "GET /myownlogsbucket/myapp/AWSLogs/028578912368/elasticloadbalancing/us-east-
1/2015/02/18/028578912368_elasticloadbalancing_us-east-1_MyLoadbalancer_20150218T1000Z_54.85.108.72_dzdvvvy8.log?
X-Amz-Date=20150218T103721Z&X-Amz-Expires=300&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Signature=ca71da5a8639895ffae88c18ac998b4a457559041422c5e785d11dd460fc14d1&X-Amz-
Credential=ASIAJLWYBGZX77VM6P3Q/20150218/us-east-1/s3/aws4_request&X-Amz-SignedHeaders=Host&x-amz-security-
token=AQoDYXdzEEQagAKh3dXT/b%2BuxVLeigXA/jYaewrCDf6JQFBqDUpf88cJZylVFjh3yTwhxyAE%2BTM2hiUxQb79Y08Y3O1XgUiI026%2BCW
4Zjl623HTlFZOPz0rXMcZPkfjs2ibAcDP7TReTbr7WO9rtQQO2OH7MeVdu4pJHpWPOKKQuaPqB0gfCnfOtBtZshBJt8Ki/IgHgttxbau8EwNoXhP5S
gusf/hDZSLndjJ/P%2BQUyMqoXCVJYpiTovUu5lONJmO9ua6Vu7/thJH7JD0JBJhJ%2Bxv%2BFdOsb8HhPfbTA3Rz2xU%2BfAaRA8ijRqfdpfARRPf
88%2BMy6pvG9f6yWu4AQLGMU0wFuJ92g1leIILXWkacF HTTP/1.1"
HTTP Status - 200
Error code - -
Bytes Sent - 331
Object Size - 331
Total time - 43 [ms. Is measured from the time your request is received to the time that the last byte of the
response is sent.]
Turn-Around Time - 43 [ms. Is measured from the time the last byte of your request was received until the time the
first byte of the response was sent.]
Referrer - "https://s3-console-us-standard.console.aws.amazon.com/GetResource/Console.html?region=us-east-
1&pageLoadStartTime=1424255801228&locale=en"
User Agent - "Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/40.0.2214.111
Safari/537.36"
```

Figure 11.2: Amazon S3 Sever Access Log [38]

In a specific five minute time period, the capability has to collect the log files for all the requests made, i.e., from Time_0 to Time_{+5min}. Then, the Error Rate is calculated counting the number of Error Code equals to InternalError or ServiceUnavaiable, and divided by the total number of requests during that five minute time period.

Considering that every entry in the log has a related Error Code field, the total number of requests during that five minute period is obtained by the following formula:

$$5minRequests = \sum_{0}^{+5min} ErrorCode$$

The Error Rate of a five minute time period is obtained by the following formula:

$$5minErrorRate = \frac{\sum_{0}^{+5min} ErrorCode = InternalErrorORServiceUnavaiable}{5minRequests}$$

In order to have a monthly value, the computation has to be done during a billing month time period. Every solar day has $60/5 * 24 = 288$ five minute time periods; this value has to be multiplied for the number of days of a billing month. Assuming that a billing month is composed of thirty days, we will have $288 * 30 = 8640$ five minutes time periods. Finally, the monthly uptime percentage will be obtained by the following formula:

$$AverageErrorRate = \frac{\sum_{0}^{8640} 5minErrorRate}{8640}$$

$$MUP = 100\% - AverageErrorRate$$

11.5 Formal Model Example on Amazon S3

The Amazon S3 SLA has a validity time that begins when a customer has access to S3 until he decides to terminate. The solar year 2015 is the S3 SLA validity time.

$$L = \{S3\}$$

$$S3 = \langle S3SL, 01/01/2015, 31/12/2015 \rangle$$

The service level uptime is expressed as Monthly Uptime Percentage, MUP for short. *Amazon Web Services will use commercially reasonable efforts to make Amazon S3 available with a Monthly Uptime Percentage [...] of at least 99.9%*

during any monthly billing cycle. April 2015 is considered as the billing month range.

$$a = MonthlyUptimePercentage$$

$$r = S3server$$

$$S3SL = \{MUP\}$$

$$MUP = \langle I, MonthlyUptimePercentage^{S3server}, 01/04/2015, 30/04/2015\rangle$$

$$I = \{i_1\}$$

$$c = atleast$$

$$atleast = \geq$$

$$c = \geq$$

$$i = \langle \geq 99\%, MUP_m\rangle$$

The unit measure is not expressed, but the percentage symbol, so it is set to %.

Recalling the definition of this attribute: *Monthly Uptime Percentage is calculated by subtracting from 100% the average of the Error Rates from each five minute period in the monthly billing cycle.* Error Rate is *the total number of internal server errors returned by Amazon S3 as error status InternalError or ServiceUnavailable divided by the total number of requests during that five minute period* [9].

The metric for calculating MUP is a composed metric, because it needs the computation of the *AverageErrorRate* that depends on *5minErrorRate*, depending again on *5minRequests* .

$$M = \{MUP_m, AverageErrorRate_m, 5minErrorRate_m, 5minRequests_m\}$$

$$MUP_m = 100\% - AverageErrorRate_m$$

$$AverageErrorRate_m = \frac{\sum_0^{8640} 5minErrorRate_m}{8640}$$

$$5minRequests_m = \sum_{0}^{+5min} ErrorCode$$

$$5minErrorRate_m = \frac{\sum_0^{+5min} ErrorCode = InternalErrorORServiceUnavaiable}{5minRequests}$$

Every five minutes the SLACFR capability collects the S3 Server Access Log files generated by the execution of the service hosting the wiki website of Bob. The logs contain every HTTP request made to the service, and the response is stored. The information from those logs is mapped in the formal model in the following way.

$$R = \{S3server\}$$

$$A = \{MonthlyUptimePercentage\}$$

$$S3server = \{\{MonthlyUptimePercentage\}^{S3server}\}$$

During the execution of an S3 instance, the value of an attribute of a resource is subject to change via an operation o. Each operation is described by a mathematical tuple composed of a sender s that is the executor of this operation, a result $value(a^r)$ that describes the value assigned to attribute a of resource r, an operation resource r, an attribute a, and an operation time t_o. The sender is the Remote IP address field of the log in Figure 3.2.

$$S = \{137.43.248.70\}$$

The operation o is the operation field of the S3 Server Access Log file.

$$O = \{REST.GET.OBJECT\}$$

The components of the operation are described in the following tuple.

$$REST.GET.OBJECT = \langle 137.43.248.70, MonthlyUptimePercentage^{S3server}$$

$$,,,18/Feb/2015:10:37:23+0000\rangle$$

The *value* element of the tuple is empty, as well as the unit measure. More precisely, the *value* element is neither InternalError nor ServiceUnavailable, as depicted in Figure 11.2.

In order to build a log $cl \in CL$, the SLACFR capability translates many S3 Server Access Log files in a set of operations O^{cl}. From this mapping, the cloud log feeds the metric MUP^i_m in order to determine the elements of the set of comparisons Q described by equation 9.17 in Section 9.6.

11.6 Summary

The description of the case study utilised in this dissertation in order to execute the prototype system is the focus of this chapter. Some motivations and a description of some assumptions open the chapter, then a scenario is presented. Subsequently, the SLA and the logs of the chosen cloud platform are detailed, and the customisation of the proposed cloud forensic readiness formal model on it is illustrated.

In the following chapter the simulation platform utilised to generate some Amazon S3-like log files is described. Such files are then used to feed the prototype system in order to run some test cases aimed to detect the possible contractual violations. So, the technical requirements of the platform are described, together with the simulated attack routine, and the generation of the log file dataset. Such dataset is then converted into a format that can be matched by the forensic capability, so such converter routine is illustrated.

Chapter 12

Log Files Generation

12.1 Introduction

In this chapter the simulation platform utilised to generate some Amazon S3-like log files is described. The log files needed for the system testing have to be compliant with the case study described in Chapter 11 in order to be correctly matchable with the related SLA. The necessity to obtain simulated logs is derived by the unavailability of such information: log files are private because they represent all the flow of operations happening during the execution of a Cloud service, Amazon S3 in this case, for a specific customer.

Such files are then used to feed the prototype system in order to run some test cases aimed to detect the possible contractual violations. So, the technical requirements of the platform are described, together with the simulated attack routine, and the generation of the log file dataset. Such dataset is then converted into a format that can be matched by the forensic capability, so such converter routine is illustrated.

12.2 Simulation Platform

Amazon S3 is the Simple Storage Service of Amazon [9]. In this cloud service the files are organised in structures called *bucket*. They are are similar to machine file system folders, but they have no capacity limit. Amazon S3 allows to use buckets for hosting static websites, composed of HTML, CSS, and JS pages, and to memorise the access logs concerning such resources. Such logs are used to test the prototype system for detecting SLA violations in a cloud forensic readiness context. In a theoretical usage of the cloud forensic readiness system the log files should be retrieved directly from the service provider. In this case study instead, the generated logs are stored locally in a file and imported by the prototype tool into a not-relational data base, as described in Chapter 10.

The simulation platform is a replacement of the real Amazon S3 service that cannot be used because it is a private and a charged service on which a cyber attack can be extremely hard to be executed. An Amazon S3-similar platform is needed to automatically execute multiple requests from different machines that simulate a normal usage of the original service. The choice fell on a small Web application named LittleS3 [100] that is free and open-source and that provides the same functionalities of the original service. LittleS3 [100] is used to generate Amazon S3-compliant log files. It runs on a Tomcat Web server, and it receives HTTP requests in order to GET, PUT or DELETE an object from a specified bucket. As well as Amazon S3, LittleS3 allows to manage the service objects and buckets through a Web interface. The service files are stored on its own Web server, which makes very easier to manage requests and resources, and to monitor the service status and the host machine availability. Also the simulated log file are stored on Tomcat, as server access log files, where they are collected from.

The diagram in Figure 12.1 represents the different components of the simulation platforms and their interactions with the SLACFR prototype system described in Chapter 10 and the automatic SLA classifier described in Chapter 8.

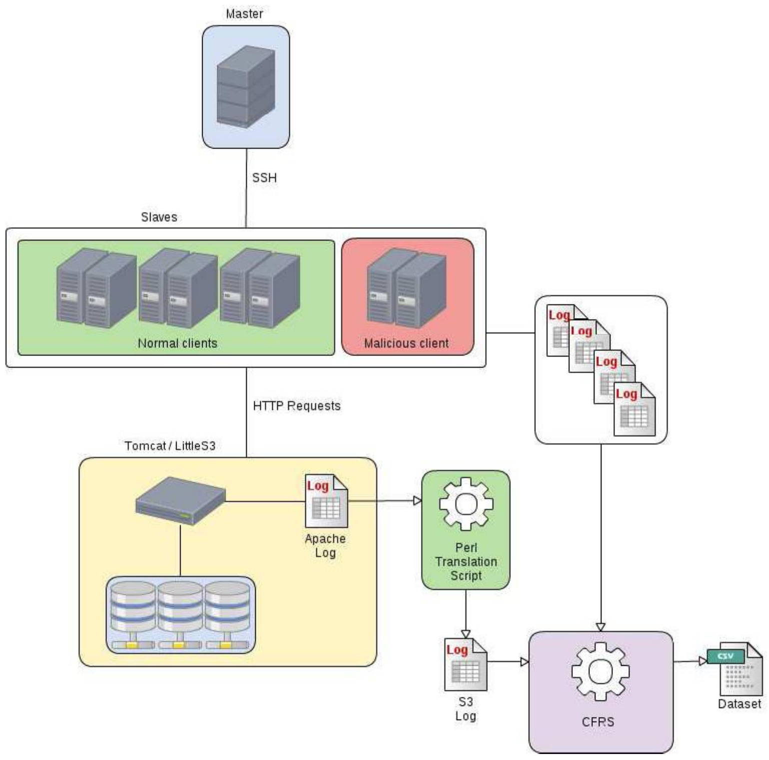


Figure 12.1: Simulation Platform

- Tomcat & LittleS3: the Amazon S3 service is executed on a machine; it waits for requests to be executed, and it records every access on the Tomcat access log;

- Simple DoS Attack [118]: it is an open source project utilised to perform automatic requests from some service clients; such a tool is executed on multiple machines with a different parameter set-up depending on the nature of the request, i.e., attack or not-attack;

- Perl Script: it is a conversion script written in Perl that utilises regular expressions to convert the Tomcat access log files format to the Amazon S3 one;

- CFRS: it is the system prototype described in Chapter 10.

The master machine utilises the SSH protocol to order the slave machines to execute the Simple DoS Attack command. The malicious client executes the *evil* version of such command that will perform multiple and frequent requests to the server. It will request always the same file with the aim to increase the response time and simulate a DoS attack. The attack will be performed along five minutes, and then the client will perform standard requests. Such simulation will be repeated many times, increasing in every time the number of five minute time interval duration of the *evil* command, namely the first time it will last five minutes, the second time ten, and so on. Both slave machines and Little S3 server produce log files that will be used to compose the log files dataset. The slave machines log files are used to identify the five minutes time intervals when an attack was executed; the Tomcat log files are converted to the Amazon S3 format and then utilised to perform the computation determining the actual values of the SLA parameters to monitor.

12.2.1 Technical Requirements

The machines technical requirements are described in this section, together with the manner in which the Simple DoS Attack command is invocated [87].

Several virtual machines are set-up in the simulation environment; they are dedicated to the execution of the normal clients, the malicious client, the master node, and the LittleS3 server. There is one server machine (IP: 192.168.60.17) where Little S3 is installed on; one malicious client machine (IP: 192.168.60.6); one master machine (IP: 192.168.60.2; Public IP: 193.205.186.57) dedicated to the scheduling of the Simple DoS Attack command; ten normal client machines (IP range : 192.168.60.[18-28]). The technical features are listed in Table 12.1.

Table 12.1: Machine Technical Requirements

	Normal Client	**Malicious Client**	**Master**	**Little S3 Server**
CPU	1 core	2 core	2 core	2 core
RAM	512 MB	2 GB	2 GB	2 GB
DISK	20 GB	20 GB	20 GB	20 GB

12.2.2 Simple DoS Attack

SimpleDos.jar is the Java program used to automate the experiment process. A copy of this program is installed on each client / master machine.

When the Simple DoS Attack command runs, ten files with different sizes will be created using the following command: *this.executeCommand(”dd if=/dev/urandom of=”+name+i+” bs=”+i*10*1024+” count=1”);*

The .jar file is launched with different options, depending on the role of the client machine:

java -jar SimpleDoS.jar http://[server_address] littleS3-2.3.0/[bucket_name] [mode] [mode] can have 3 different values:

- 0 for Normal client: it executes an HTTP request every 5 seconds with probability of 40% on a randomly chosen file from the set of created files, e.g., an image or a text file. The request type will be a GET with 70% of probability and a PUT / DELETE with 30% of probability;
- 1 for Malicious client: every hour the malicious routine is launched; it starts a pool of 10 threads, each one sends a PUT request with the file with the biggest size chosen from the set;
- 2 for Health checker: it acts as a normal client that checks the response time of the service; as soon as a time-out is detected a command to restart the service is executed.

12.3 Log Files Dataset

A big enough dataset of log files has to be generated for the prototype system testing. In order to obtain the log files correctly, the logging module of Tomcat in LittleS3 has to be configured. The log files format is edited in the file *server.xml* in this manner [87]:

pattern="%h %l %u %t &quot;%r&quot; %s %b &quot;%Refereri&quot; &quot;%User-Agenti&quot; %D %q"

%D and *%q.* are added at the end of the file, where *%D* allows to obtain information about the time used to process the requests, expressed in milliseconds, and *%q* the query string. This information will be used to generate logs containing the same information of the Amazon S3 access logs.

The log file entries are generated as soon as the server receives HTTP requests and provides HTTP responses. The requests sent by the client machines can either represent a standard behaviour or attempt to violate the service by sending multiple and frequent requests, aimed to perform a DoS or DDoS cyber attack.

Moreover, a data preparation and transformation phase follows the collection one, because the Apache logs format does not match with the Amazon S3 one. A script written in Perl, which is well suited to the use of regular expressions, automates the format transformation of Tomcat-format files to Amazon S3-format files, without altering the contents.

In Table 12.2 the mapping between Tomcat log structure and Amazon S3 log structure is reported.

Some fields in the Apache column are marked with - when there is no value to be used to convert to the Amazon S3 format. Indeed information in the S3 logs result to be more accurate than in the Apache log format. The fields that have no match have been generated using the method described in Table 12.3.

The Apache access log file is a text file composed of several entries separated

Table 12.2: Tomcat and Amazon S3 Log File Structure Mapping

Apache Example Entry	**Apache Access Log Field**	**Amazon S3 Access Log Field**
-	-	Bucket Owner
-	-	Bucket
[10/Oct/2000:13:55:36 -0700]	Time	Time
127.0.0.1	Remote host	Remote IP
-	-	Requester
-	-	Request ID
”GET /apache_pb.gif HTTP/1.0”	Request line	Operation
”GET /apache_pb.gif HTTP/1.0”	Request line	Key
”GET /apache_pb.gif HTTP/1.0”	Request line	Request-URI
200	Status code	HTTP status
-	-	Error Code
2326	Size of object returned	Bytes Sent
2326	Size of object returned	Object Size
-	-	Total Time
7	Time to process the request	Turn-Around Time
”http://www.example.com/start.html”	Referrer	Referrer
”Mozilla/4.08 [en] (Win98; I ;Nav)”	User-Agent	User-Agent
	-	Version Id

by the new line character, and every field is separated by a blank space. A regular expression is used to obtain for every entry each field; then they are converted and stored to the Amazon S3-line log file.

```
(/^([\w\.:-]+)\s+([\w\.:-]+)\s+([\w\.-]+)\s+\[(\d+)\/(\w+)\/(\d+):(\d+):
(\d+):(\d+)\s?([\w:\+-]+)]\s+”(\w+)\s+(\S+)\s+HTTP\/1\.\d”\s+(\d+)\s+
([\d-]+)((\s+”([^”]+)”\s+”)?([^”]+)”)?\s(\d+)\s\?atk=(\d+)$/)
```

The Perl script can be executed on Ubuntu system launching the following command on the terminal: `$ perl parser.pl Apache_access_log_file > S3_log_file` where parser.pl is the Perl script, Apache_access_log_file is the Apache log file and S3_log_file is the output file.

These commands will create a new file named S3_log_file containing the Amazon S3-simulated access log file. A the end of this process, such log entries are stored on MongoDB. An example of the generated logs follows.

```
“_id” : ObjectId(“550b410fe4b0fd7341b84bed”), “BucketOwner” :“5927116389e7d406047
097a41cba2ef5830ad74cdaf67351d74682eeaa07ea2b”, “BucketName” : “mybucketwebsite”, “Day”
: “15”, “Month” : “02”, “Year” : “2015”, “Hours” : “03”, “Minutes” : “11”, “Seconds” :
“09”, “Time” : NumberLong(“1423969869000”), “GMT” : “+0100”, “RemoteIP” : “201.92.110.117”,
“Requester” : “-”, “RequestID” : “01KAIWBCH8X4XL90”, “Operation” : “WEBSITE.GET.OBJECT”,
“Key” : “wp-includes/js/jquery/ui/accordion.css”, “Request” : “GET”, ”URI” : “/wp-includes/js/jquery/
ui/accordion.css?ver=1.11.2”, “HTTPVer” : “HTTP/1.1”, “HTTPStatus” : “200”, ”ErrorCode”
: “-”, “ByteSent” : “8508”, “ObjectSize” : “8508”, “TotalTime” : “104”, “TurnAroundTime”
: “104”, “Referrer” : “http://www.laclessidracicciano.it/”, “UserAgent” : “Mozilla/5.0 (Win-
dows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/32.0.1700.107 Sa-
fari/537.36”, “VersionID” : ” ”
```

The metrics described in the Amazon S3 SLAs used to calculate the service parameters constraint the calculation time interval on five minutes [9]. Some values are calculated on the generated log files on such five minute time intervals to facilitate the final comparison between actual and constrained values:

- AVG (object size) : average size of the requested objects, in bytes;
- Success rate: number of HTTP requests with response code 200 divided by the total number of requests;
- Error rate: number of HTTP requests with response code 500 divided by the total number of requests;
- PUT rate: number of PUT type requests divided by the total number of requests;

- GET rate: number of GET type requests divided by the total number of requests;
- AVG (response time): average server response times;
- Sim User: number of users simultaneously connected to the service; this is calculated considering the number of different IP addresses making a server request during the last five minutes;
- Uptime: percentage value of the uptime parameter, calculated with the metric specified in the Amazon S3 SLA;
- Uptime Violation: boolean value indicating whether an actual uptime level is violating the value specified in the Amazon S3 SLA;
- Sim User Violation: boolean value indicating whether an actual Sim User value is violating the value specified in the Amazon S3 SLA;
- AVG (response time) Violation: :boolean value indicating whether an actual average response time value is violating the value specified in the Amazon S3 SLA;
- Attack: boolean value used indicating whether an service attack is detected.

12.4 Summary

The simulation platform utilised to generate some Amazon S3-like log files is described in this chapter. Such files are then used to feed the prototype system in order to run some test cases aimed at detecting the possible contractual violations. So, the technical requirements of the platform are described, together with the simulated attack routine, and the generation of the log file dataset. Such dataset is then converted into a format that can be matched by the forensic capability, so such converter routine is illustrated.

In the following chapter the usage of such simulated log files is described, and the prototype system testing is illustrated. The testing phase is planned, executed, and reported in dedicated sections. The test suite is composed of seven test cases aimed to detect different contractual violations.

Table 12.3: Amazon S3 Log File Missing Field Generation Method

Amazon S3 Access Log Missing Field	**Generation Method**
Bucket Owner	Static field the bucket owner is a constant value corresponding to an alphanumeric string.
Bucket	Static field the bucket name is a constant value corresponding to the name of bucket. A constant bucket name ID is used to fill this field. All the log entries will have the same Bucket ID.
Requester	Static field for WEBSITE.GET.OBJECT operations on public objects from a not-authenticated user, the requester ID is a - character, like in this simulation.
Request ID	A unique alphanumeric string. The Perl script generates a unique alphanumeric string for each request.
Error Code	Depending on the HTTP response, this field reports the error name. The Perl script generates the name of the error according to both HTTP response number and the error code provided in the Amazon S3 documentation.
Total Time	Its value depends on the size of the requested object. This value is calculated as: [Turn-Around Time]+[Network Write Speed]. The average of the Network Write Speed has been calculated by doing GET requests on objects (text and images) of different sizes (1MB, 2MB and 5MB) on different day hours. The Perl script uses a download speed that ranges randomly from 30MB/s to 70MB/s.
Version Id	This field in not used for Website services. For each log entry a - constant is generated for this field.

Chapter 13

System Testing

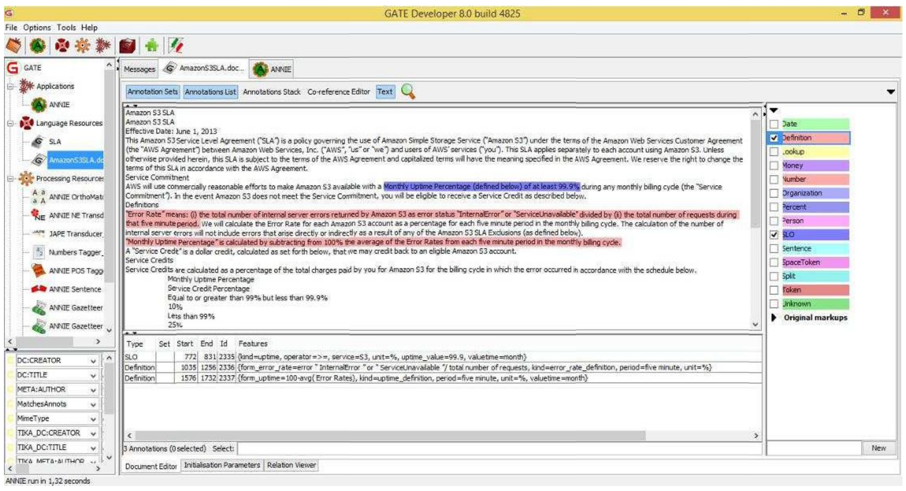
13.1 Introduction

In this chapter a system testing based on the case study described in Chapter 11 is performed. A cloud service user Bob performs his daily security controls using SLACFR. The system detects anomalies about SLA parameters, such as Monthly Uptime Percentage, Average Response Time, and Number of Simultaneous Connections, on Amazon S3. At the same time, the system raises a warning message to Bob as soon as such anomalies are detected; possible security attacks can be also identified, and this information has to be sent to Bob too. Such a test suite is composed of seven test cases aimed to detect different contractual violations, and are planned, executed, and reported in dedicated sections.

13.1.1 Planning

In order to catch the anomalies, the system extracts the SLOs values from the SLA document with GATE. Figure 13.1 depicts a phase of the execution of GATE with the customised JAPE and Transducer files.

According to the document, one of the extracted SLO is the Monthly Uptime



Amazon S3 SLA
Amazon S3 SLA
Effective Date: June 1, 2013
This Amazon S3 Service Level Agreement ("SLA") is a policy governing the use of Amazon Simple Storage Service ("Amazon S3") under the terms of the Amazon Web Services Customer Agreement (the "AWS Agreement") between Amazon Web Services, Inc. ("AWS", "us" or "we") and users of AWS' services ("you"). This SLA applies separately to each account using Amazon S3. Unless otherwise provided herein, this SLA is subject to the terms of the AWS Agreement and capitalized terms will have the meaning specified in the AWS Agreement. We reserve the right to change the terms of this SLA in accordance with the AWS Agreement.
Service Commitment
AWS will use commercially reasonable efforts to make Amazon S3 available with a Monthly Uptime Percentage (defined below) of at least 99.9% during any monthly billing cycle (the "Service Commitment"). In the event Amazon S3 does not meet the Service Commitment, you will be eligible to receive a Service Credit as described below.
Definitions
"Error Rate" means: (i) the total number of internal server errors returned by Amazon S3 as error status "InternalError" or "ServiceUnavailable" divided by (ii) the total number of requests during that five minute period. We will calculate the Error Rate for each Amazon S3 account as a percentage for each five minute period in the monthly billing cycle. The calculation of the number of internal server errors will not include errors that arise directly or indirectly as a result of any of the Amazon S3 SLA Exclusions (as defined below).
"Monthly Uptime Percentage" is calculated by subtracting from 100% the average of the Error Rates from each five minute period in the monthly billing cycle.
A "Service Credit" is a dollar credit, calculated as set forth below, that we may credit back to an eligible Amazon S3 account.
Service Credits
Service Credits are calculated as a percentage of the total charges paid by you for Amazon S3 for the billing cycle in which the error occurred in accordance with the schedule below.
Monthly Uptime Percentage
Service Credit Percentage
Equal to or greater than 99% but less than 99.9%
10%
Less than 99%
25%

Type	Set	Start	End	Id	Features
SLO		772	831	2335	{kind=uptime, operator=>=, service=S3, unit=%, uptime_value=99.9, valuetime=month}
Definition		1035	1256	2336	{form_error_rate=error " InternalError " or " ServiceUnavailable "/ total number of requests, kind=error_rate_definition, period=five minute, unit=%}
Definition		1576	1732	2337	{form_uptime=100-avg(Error Rates), kind=uptime_definition, period=five minute, unit=%, valuetime=month}

3 Annotations (0 selected) Select:

Figure 13.1: GATE Execution

Percentage (MUP), calculated by *subtracting from 100% the average of the Error Rates from each five minute period in the monthly billing cycle.* Error Rates is *the total number of internal server errors returned by Amazon S3 as error status InternalError or ServiceUnavailable divided by the total number of requests during that five minute period.* The system has to calculate the MUP value day by day, using the access logs provided by S3. Thus, the system gets all the HTTP requests registered in every five minutes intervals, and using the following formulas, it calculates the real value of MUP, as described in Chapter **??**.

$$MUP = 100\% - AverageErrorRate$$

$$AverageErrorRate = \frac{\sum_0^{8640} 5minErrorRate}{8640}$$

$$5minRequests = \sum_0^{+5min} ErrorCode$$

$$5minErrorRate = \frac{\sum_0^{+5min} ErrorCode = InternalErrorORServiceUnavaiable}{5minRequests}$$

The system has to recognize a suspect MUP difference equal to 0.1% on the value contained in the SLA document. According to the parameter definition, a decrease of monthly uptime percentage corresponds to an unusual increase of internal server errors. This problem could be caused by an attack on Amazon S3 server (DDoS, DoS) (see Table 7.5). Every 24 hours the MUP value is stored in the Calculated Data DB, and then compared with the values of previous days. At the end of the billing month, the final calculated value is compared with the SLO value.

The other SLOs to test are the Average Response Time (ART), and Number of Simultaneous Connections (NSC). Unlike MUP, ART is not extracted by the public Amazon S3 SLA. Its definition is obtained by the European guidelines definition [49]; the value to be respected is obtained by a benchmark [88] where the average response time has to be of 500 milliseconds every five minutes. According to the EU guidelines, ART is the average of all the requests response times from each five minute time period in a billing month cycle. Response time is only the Turn-Around Time without transmission time, because related to the file size. The used formula for ART is:

$$ART = \frac{\sum_0^{+5min} ResponseTimes}{numberofrequests}$$

Also the NSC parameter is not extracted by GATE. It is defined using the European guidelines definition [49]. NSC is the number of single customers using the service in a minute time period. Its value can be negotiated depending on application needs. In this scenario, Bob owns a wiki website, and the number of connections cannot be extremely high. Then, such a number can be assumed to be set as 500 connections in a single minute. The time period is set very short to allow to catch an attack, such as DDoS. The used formula for NSC is a count of different IP addresses performing a request in a minute time period:

$$NSC = \sum_0^{+1min} IPAddressperformingrequests$$

In order to execute the comparisons, some logs are necessary. Some simulated Amazon S3 logs are generated as described in Chapter 12, and utilised for this system testing.

13.1.2 Execution

The system starts recovering all the SLAs objects stored in the SLA database. A matching operation is launched for each SLO. Three SLOs violation recognition have been implemented: Monthly Uptime Percentage, Average Response Time, and Number of Simultaneous Connections. In order to catch an MUP violation during a time period shorter than a billing month, some additional metrics have been added to the test code, namely to calculate the uptime percentage per hour (HUP) and per day (DUP).

$$HUP = 100\% - AverageErrorRate_h$$

$$AverageErrorRate_h = \frac{\sum_0^{12} 5minErrorRate}{12}$$

$$5minRequests = \sum_0^{+5min} ErrorCode$$

$$5minErrorRate = \frac{\sum_0^{+5min} ErrorCode = InternalErrorORServiceUnavaiable}{5minRequests}$$

$$DUP = 100\% - AverageErrorRate_d$$

$$AverageErrorRate_d = \frac{\sum_0^{288} 5minErrorRate}{288}$$

$$DUP = \frac{\sum_0^{24} HUP}{24}$$

$$MUP = \frac{\sum_0^{30} DUP}{30}$$

The system calculates HUP, and after 24 hours it can compute the average of all the previous values. After thirty days the average of all the DUP can calculate the MUP value. For this purpose, two circular arrays to store such data have been used. Every calculated uptime value obtained is stored in a MongoDB collection. The task calculating HUP, is a Java thread *Calculator* launched every hour: it evaluates the necessary formula stored in the database using JEval. The JEval function *Avg* calculates information average for every five minutes time period; it launches a *Callable* thread that searches the atomic elements into the related formula; then, using the dictionary, it executes the corresponding MongoDB queries. The calculated values are obtained, and the formula is evaluated by JEval. Its value is returned to the *Avg* function that performs the average of all values computing the Average Error Rate. Then a simple subtraction is calculated. The UML sequence diagram in Figure 13.2 depicts the interactions of the system components.

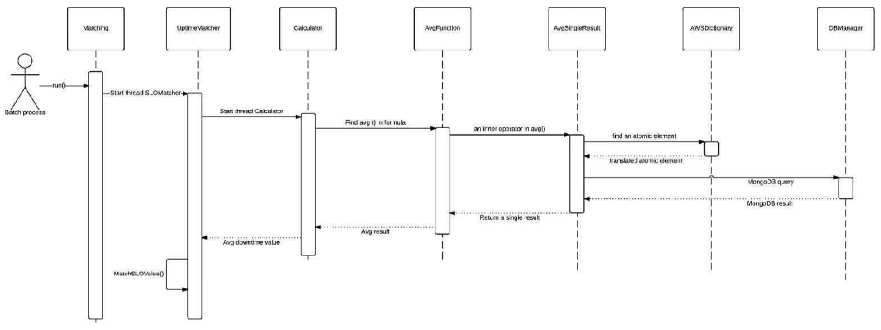


Figure 13.2: MUP Calculation UML Sequence Diagram

The test execution to verify the ART SLO is similar to the previous one. A thread *Calculator* is launched every five minutes evaluating the formula using JEval. The *Avg* function calculates the average of all response times, i.e., field Turn-Around Time of the logs, using a *Callable* thread that extracts all the requests every five minute time periods. The average result is returned to Calculator for the matching between the real value and the SLO value. The UML sequence diagrams in Figure 13.3 and 13.4 depict the interactions of the system components without and with errors, respectively

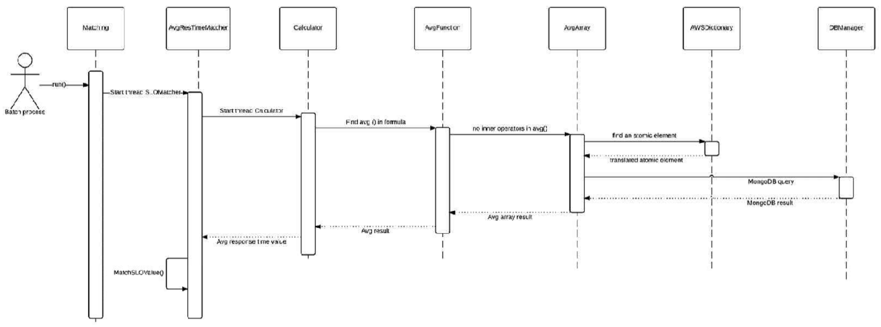


Figure 13.3: ART Calculation UML Sequence Diagram - No Errors

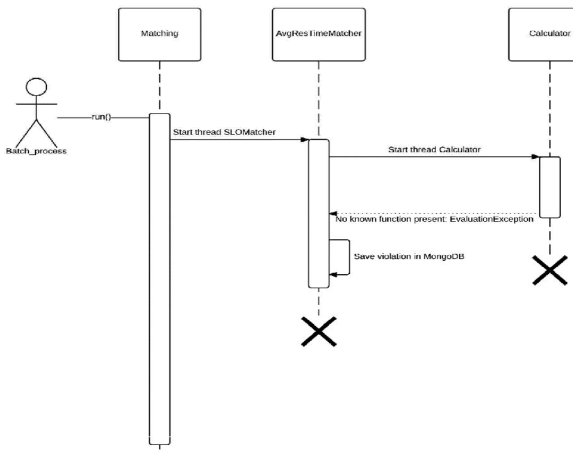


Figure 13.4: ART Calculation UML Sequence Diagram - Errors

For the NSC SLO a thread *Calculator* is launched every five minutes evaluating the formula using JEval. A new *Count* function uses a *Callable* thread that counts during every minute the distinct recorded IP Addresses, i.e., field Remote IP in MongoDB. The total is returned to *Count*, and the count result to Calculator for the matching between the real value and the SLO value. The UML sequence diagram in Figure 13.5 depicts the interactions of the system components.

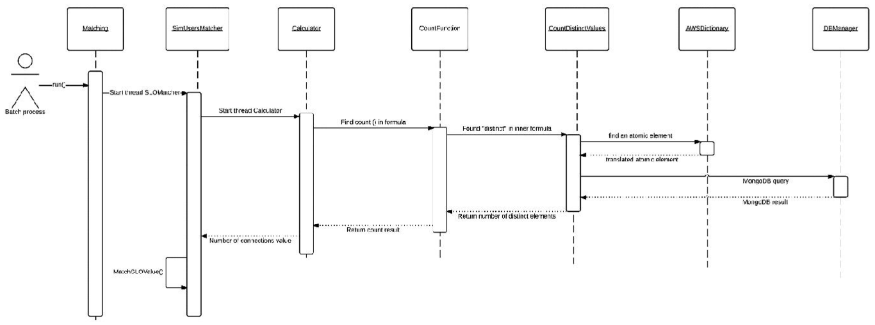
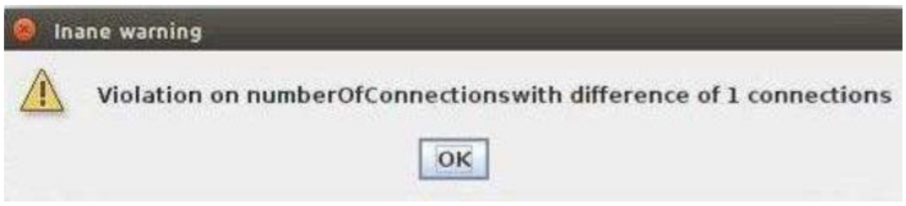


Figure 13.5: NSC Calculation UML Sequence Diagram

After the actual values for the three SLOs have been calculated, the system sends them to the matching component: it extracts values and operators, such as $\leq$ or =, from the SLO value object, and then it builds the matching formula. This is evaluated by JEval, that returns the boolean result of the match. If the result is false, the difference between the actual and the contractual SLOs values is calculated, then it is stored in MongoDB and notified to the user via a pop-up similar to the one in Figure 13.6.


Inane warning

Violation on numberOfConnectionswith difference of 1 connections

Figure 13.6: SLACFR Violation Alert

In addition to the match for single SLOs, the system uses the information stored in the violation database, recorded as soon as a JEval matching result is negative, to detect a possible attack, composed of multiple SLOs violations (see Table 7.5). In this case, anomalies in the three implemented SLOs can be caused by a DDoS attack. This functionality is implemented using another thread execution; it is launched and then paused during a time interval equal to the maximum time period of the three chosen SLOs; then it wakes up and seeks HUP, NSC, and ART violations. The UML sequence diagram in Figure 13.7 depicts the interactions of the system components. If all the violations are gathered from the database, then a notification is sent to the user via pop-up similar to the one in Figure 13.8.

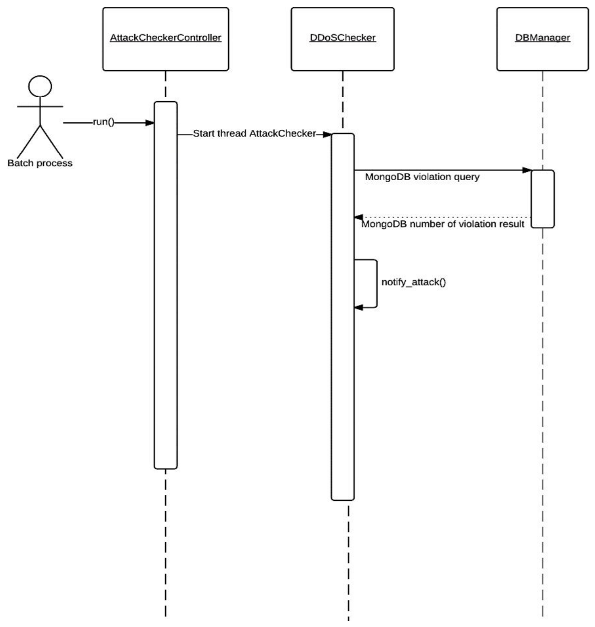
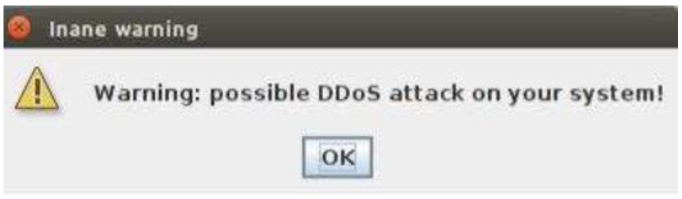


Figure 13.7: DDoS Calculation UML Sequence Diagram


Inane warning

Warning: possible DDoS attack on your system!

Figure 13.8: SLACFR Attack Alert

13.2 Results

In this section, some tests executions results are reported. The input of the tests is represented by S3-like log files and S3 public SLA; in particular, the three SLOs of MUP, ART, and NSC extracted by GATE are considered. Because the log recorded a system execution during a single hour, the MUP parameter cannot be calculated; instead, it is scaled down to the hourly uptime percentage, described by the formula HUP in the previous section, whose constraint remains unaltered, i.e., $\geq 99\%$. The constraint for ART is $\leq 500 milliseconds$, while the NSC value has to be ≤ 10.

The used logs about one hour service execution have eleven entries; each entry has a different IP address. The *turn around time* and *error code* fields of the eleven rows is changed in order to detect the violations in the following test cases.

An example of a log entry is described by the following code:

5927116389e7d406047097a41cba2ef5830ad74cdaf67351d74682eeaa07ea2b mybucketwebsite [15/02/2015:03:07:53 +0100] 66.249.78.20 - UGBXDJ9QZGSG2EX5 WEBSITE.GET.OBJECT fanarea.html "GET /fanarea.html HTTP/1.1" 500 InternalError 0 0 9999 99999 "-" "Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)" -

The followed approach is to identify all the possible combinations of the three SLOs violations. The total number of test cases should be eight because we have three SLOs that can be violated or not at one time, i.e., $2^3 = 8$. Excluding a test case where no violation is detected, we obtain $8 - 1 = 7$ test cases. The first three test cases detect a violation of a single SLO each; then, from the fourth to the sixth a violation for two SLOs, and finally a violation for all the three SLOs, with a related DDoS attack information. In the following list, the output of the seven test cases is described.

- Case1: HUP violation detection

Some log entries *Error Code* field is set to InternalError. The system detects a violation for HUP SLO, then one entry is stored in the violation database.

"_id" : ObjectId("55366fcbe4b0ad64725155d5"), "SLO" : "hourlyUptimePercentage", "Difference" : -1.6, "Unit" : "%", "Time" : NumberLong("1423973273000")

- Case2: ART violation detection

  The *Turn-Around Time* field is 999 millisecond for some log entries. The system detects a violation for ART SLO, then one entry is stored in the violation database.

  Violations database:

  "_id" : ObjectId("553665c6e4b050717035da71"), "SLO" : "averageResponseTime", "Difference" : 20013.32, "Unit" : "ms", "Time" : NumberLong("1423969973000")

- Case3: NSC violation detection

  Eleven log entries have eleven different IP addresses. The system detects a violation for NSC SLO, then one entry is stored in the violation database.

  Violations database:

  "_id" : ObjectId("553664d6e4b050717035da6c"), "SLO" : "numberOfConnections", "Difference" : 1, "Unit" : "num", "Time" : NumberLong("1423969733000")

- Case4: HUP and ART violations detection

  The *Turn-Around Time* field is 999 millisecond for some log entries, also some and *Error Code* fields are set to InternalError. The system detects a violation for both HUP and ART SLOs, then two entries are stored in the violation database.

  Violations database:

“_id” : ObjectId(“55366fcbe4b0ad64725155d5”), “SLO” : “hourlyUptimePercentage”, “Difference” : -1.6, “Unit” : “%”, “Time” : NumberLong(”1423973273000”)

“_id” : ObjectId(“553665c6e4b050717035da71”), “SLO” : “averageResponseTime”, “Difference” : 20013.32, “Unit” : “ms”, “Time” : NumberLong(”1423969973000”)

- Case5: HUP and NSC violations detection

  The *Error Code* field is set to InternalError for some log entries, which have recorded eleven different IP addresses. The system detects a violation for both HUP and NSC SLOs, then two entries are stored in the violation database.

  Violations database:

  “_id” : ObjectId(”55366fcbe4b0ad64725155d5”), “SLO” : “hourlyUptimePercentage”, “Difference” : -1.6, “Unit” : “%”, “Time” : NumberLong(”1423973273000”)

  “_id” : ObjectId(”553664d6e4b050717035da6c”), “SLO” : “numberOfConnections”, “Difference” : 1, “Unit” : “num”, “Time” : NumberLong(”1423969733000”)

- Case6: ART and NSC violations detection

  The *Turn-Around Time* field is 999 millisecond for some log entries, which have recorded eleven different IP addresses. The system detects a violation for both ART and NSC SLOs, then two entries are stored in the violation database.

  Violations database:

  “_id” : ObjectId(”553665c6e4b050717035da71”), “SLO” : “averageResponseTime”, “Difference” : 20013.32, “Unit” : “ms”, “Time” : NumberLong(”1423969973000”)

  “_id” : ObjectId(”553664d6e4b050717035da6c”), “SLO” : “numberOfConnections”, “Difference” : 1, “Unit” : “num”, “Time” : NumberLong(”1423969733000”)

- Case7: HUP, ART, NSC violation detections; DDoS attack generation

  Eleven log entries have eleven different IP addresses. The *Turn-Around Time* field is 999 millisecond for some of them. Some log entries *Error Code* field is set to InternalError. The system detects a violation for HUP, ART, and NSC SLOs, then three entries are stored in the violation database, and one entry in the attack database when a DDoS is recognised.

  Violation database:

  "_id" : ObjectId("55366fcbe4b0ad64725155d5"), "SLO" : "hourlyUptimePercentage", "Difference" : -1.6, "Unit" : "%", "Time" : NumberLong("1423973273000")

  "_id" : ObjectId("553665c6e4b050717035da71"), "SLO" : "averageResponseTime", "Difference" : 20013.32, "Unit" : "ms", "Time" : NumberLong("1423969973000")

  "_id" : ObjectId("553664d6e4b050717035da6c"), "SLO" : "numberOfConnections", "Difference" : 1, "Unit" : "num", "Time" : NumberLong("1423969733000")

  Attacks database:

  "_id" : ObjectId("55378610e4b0245276203cad"), "Attack" : "DDoS", "Time" : NumberLong("1423973273000")

13.3 Summary

A system testing for the cloud forensic readiness prototype is described in this chapter. Such a test suite is composed of seven test cases aimed to detect different contractual violations. The first three test cases detect a violation of a single SLA parameter each; then, from the fourth to the sixth a violation for two parameters, and finally a violation for all the three with a related DDoS attack information.

At the same time, the system raises a warning message to the user as soon as such anomalies are detected; the identified security attacks raise an additional warning message, i.e., the seventh test case, and the user is informed as well.

The following chapter is dedicated to the conclusion of the whole dissertation. A summary is provided, together with some limitations of the research work and some possible directions for the future.

Chapter 14

Conclusion and Future Work

The adaptation of existing forensic procedures to computing novelties is a constant and challenging task; moreover, the provisioning of a forensic readiness capability to computing infrastructure is becoming more and more complicated. FR is conceived as the provisioning of an information system communicating with an underlying computing architecture with the purpose of identifying, collecting, and storing critical data coming from them, representing potential evidence. This FR capability must be provided to cloud computing architectures because, due to their escalating popularity, they can be object of several attacks. Thus a way to conduct forensic investigations effectively, saving time, money and resources, must be designed. A DFR capability for the cloud is meant to observe and record changes concerning the operations happening in the cloud with respect to the SLA constraints related to potential crimes. The capability output include important investigative details about the recorded information and the detection of contractual clause violations. A contribution of this doctoral dissertation to the topic of forensic readiness is discussed, and a definition for DFR is provided. Moreover, a reference architecture for the implementation of an FR system for the cloud is designed and illustrated, together with some constraints and advantages.

A means for implementing such a DFR capability in the cloud includes a representation of the information to monitor. The most effective representation is the adoption of formalisms. Then, in this doctoral dissertation natural language-based SLAs clauses, cloud logs, and several entities necessary to output a comparison between them, have been structured via formal specifications. The formal model utilizes tuple, set theory, and functions, to represent the necessary entities. The formal model is validated via a prototype system implementing a case study, illustrated in the final chapters of this dissertation. This prototype has been utilised to design the architecture, and to implement the routines detecting some SLAs constraints violations. Moreover, by using the mapping between the cloud log information and the security threats identified in the central chapters, the system has been used to detect a possible crime happening in the underlying architecture.

In the future, several efforts can be made to expand this research topic. The proposed formal model can be enriched with information considered necessary in a forensic readiness capability. For instance, some law principle formal representations can be added to the existing ones, paying attention to do not alter the relations among them. Indeed, case studies involving conflictual SLAs can be the drivers for this extension, as illustrated in Section 7.4.

From the prototype system illustrated in the last chapters, several conclusions can be drawn. First of all, the manner in which information is extracted from SLAs needs a lot of research efforts in text engineering in order to automate the whole process. Indeed, in this simulation, the managed information has been very limited, using only three parameters; and some system configuration files have been manually generated because some information extraction techniques specific for contracts have not been implemented.

The main limit of the whole research work is the focus on a single case study, even it has been very time-consuming. In the future, the possibility to expand the set of case studies can lead to some additional research directions.

For example, the comparison module can be useful to address some of the cloud forensic challenges described in Table 4.1, namely the multiple log formats and the lack of timestamps synchronisation, because the system successfully relates cloud information coming from log files by using a dictionary. Unfortunately, the dictionary is provider-related, because some SLA parameters can be calculated differently from one provider to another. With the availability of multiple dictionaries a common mapping between SLOs and log files components can be drawn by composing several provider-related items.

In addition, log information used for forensic readiness can vary among providers. Nevertheless, if several dictionaries are generated and merged, then a common format for log files an be designed, and maybe also a standard can be proposed to the community.

Moreover, the SLACFR formal model and its prototype actually increase some security aspects of a cloud provider. Though the system has been tested only on an IaaS service, which grants more access to information than PaaS and SaaS, the warning messages allow the users to be aware of what is happening in the cloud in real-time. In this way, every type of action can be undertaken.

Finally, a very important extension of this capability can be driven by the availability of historical data, namely cloud log files. Such a big dataset can consent some reasoning, knowledge extraction, and prediction information useful to foresee some SLA violations detection. Also the design of a cyber-attack prediction metric can be a possible application of such a data availability.

Bibliography

[1] ACPO - Association of Chief Police Officers. Good Practice Guide for Computer Based Electronic Evidence. 2007 [on-line] http://www.acpo.police.uk/asp/policies/Data/ACPO%20Guidelines%20v18.pdf

[2] Adelstein, F. MFP: The Mobile Forensics Platform. *International Journal of Digital Evidence,* 2003, vo. 2, no. 1.

[3] Ademu, I.O., Imafidon, C.O., Preston, D.S. A New Approach of Digital Forensic Model for Digital Forensic Investigation. *International Journal of Advanced Computer Science and Applications,* 2011, vol. 2, no.12, pp. 175-178.

[4] Al-Fedaghi, S., Al-Babtain, B. Modelling the Forensics Process. *International Journal of Security and Its Applications,* 2012, vol. 6, no. 4, pp. 97-108.

[5] Alharbi, S., Weber-Jahnke, J., Traore, I. The Proactive and Reactive Digital Forensics Investigation Process: A Systematic Literature Review. *Communications in Computer and Information Science*, 2011, vol. 200, 87-100.

[6] Agarwal, A., Gupta, M., Gupta, S., Gupta, S.C. Systematic Digital Forensic Investigation Model. *International Journal of Computer Science and Security,* 2011, vol. 5, issue 1, pp. 118-131.

[7] Alonso, G., Casati, F., Kuno, H., Machiraju, V. *Web services*. Springer Berlin Heidelberg. 2004, pp. 123 - 149.

[8] Amani, N., Hajipour, P., Seyedmostafaei, F. An Appropriate Violation Detection Scenario for Service Level Agreements Based on WS-Agreement Protocol. *Journal of Convergence Information Technology,* 2010, vol. 5, no. 1, pp. 40-47.

[9] Amazon Web Service, [on-line] http://aws.amazon.com/, accessed on 22/02/2015.

[10] Ambhire, V. R., Meshram, B. B. Digital Forensic Tools. *IOSR Journal of Engineering,* 2012, vol. 2, issue 3, pp. 392-398.

[11] Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S., Xu., M. Web Services Agreement Specification (WSAgreement). *GWD-R (Proposed Recommendation), Open Grid Forum,* 2007.

[12] Baryamureeba, V., Tushabe, F. The Enhanced Digital Investigation Process Model.' *Proc. DFRWS Workshop,* 2004, pp. 1-9.

[13] Baset, S.A. Cloud SLAs: present and future. *ACM SIGOPS Operating Systems Review,* 2012, vol. 46, no. 2, pp. 57-66.

[14] Ben-Ari, M. *Mathematical Logic for Computer Science*, Springer, first edition 1993.

[15] Birk, D., Wegener, C. Technical Issues of Forensic Investigations in Cloud Computing Environments. *IEEE 6th International Workshop on Systematic Approaches to DF Engineering*, 2011, pp. 110.

[16] Brandic, I., Emeakaroha, V.C., Maurer, M., Dustdar, S., Acs, S., Kertesz, A., Kecskemeti, G. LAYSI: A Layered Approach for SLA-Violation Propagation in Self-Manageable Cloud Infrastructures. *COMPSACW Workshop,* 2010, pp.365 - 370.

[17] Broucek, V., Frings, S., Turner, P. The Federal Court, the Music Industry and the Universities: Lessons for Forensic Computing Specialists. *Proc. Australian Computer, Network and Information Forensics Conference,* 2003, pp. 1-8.

[18] Brown, S. *Software Architecture for Developers,* Coding the Architecture, 2013.

[19] Carrier, B.D., Spafford, E.H. Getting Physical with the Investigative Process. *International Journal of Digital Evidence*, 2003, vol. 2, issue 2, pp. 1-20.

[20] Carrier, B.D., Spafford, E.H. An Event-Based Digital Forensic Investigation Framework. *Proc. DFRWS Workshop,* 2004, pp. 11-13.

[21] Carrier, B.D. *A hypothesis-based approach to digital forensic investigations,* Ph.D. Dissertation, Purdue University. ProQuest, 2006.

[22] Casey, E. *Handbook of Computer Crime Investigation*. New York: Academic Press, 2002.

[23] Casey, E. *Digital Evidence and Computer Crime,* 2nd edition, Academic Press, Elsevier Science, 2004.

[24] Casey, E. *Digital Evidence and Computer Crime*, 3rd edition. Academic Press, Elsevier Science, 2011.

[25] Casey, E., Katz, G., Lewthwaite, J. Honing Digital Forensic Processes. *Digital Investigation,* 2013, vol. 10, no. 2, pp. 138-147.

[26] Cedillo, P., Gonzalez-Huerta, J., Abrahao, S., Insfran, E. Towards Monitoring Cloud Services Using Models@run.time. *International Workshop on Models at run.time,* 2014, in press.

[27] Cheng, C.P., Shaw, R.S., Liang, T.C., Fu, T.Y. An Integrated Data-Flow Based Model for Digital Investigation. *Proc. EB Conference,* 2009, pp. 507-515.

[28] Cloud4SOA Project, [on-line] http://www.cloud4soa.eu/

[29] Cloud Security Alliance. *Security Guidance for Critical Areas of Focus in Cloud Computing v 3.0*, 2011[on-line] https://cloudsecurityalliance.org/guidance/csaguide.v3.0.pdf

[30] Cloud Security Alliance. *Mapping the Forensic Standard ISO IEC 27037 to Cloud Computing*, 2013 [on-line] https://cloudsecurityalliance.org/download/ mapping-the-forensic-standard-isoiec-27037 -to-cloud-computing/

[31] Cloud Security Alliance. *Top Threats Working Group. The notorious nine: cloud computing top threats in 2013*. 2013 [on-line] https://downloads.cloudsecurityalliance.org/initiatives/top_threats/ The_Notorious_Nine_Cloud_Computing_Top_Threats_in_2013.pdf

[32] Cochrane, A. Cochrane Collaboration. *Cochrane Reviewers Handbook,* 2003, Version 4.2.1.

[33] CRN - The Channel Company. *The 100 Coolest Cloud Computing Vendors Of 2015*. 2015. [on-line] http://www.crn.com/news/cloud/300075525/the-100-coolest-cloud-computing-vendors-of-2015.htm

[34] Czajkowski, K., Foster, I., Kesselman, C., Sander, V., Tuecke, S. SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems, *Job scheduling strategies for parallel processing,* Springer Berlin Heidelberg, 2002, pp. 153-183.

[35] Danielsson, J., Tjostheim, I. The Need for a Structured Approach to Digital Forensic Readiness. *Digital forensic readiness and e-commerce (IADIS)*, 2004, pp. 417 - 421.

[36] De Marco, L., Kechadi, M-T., Ferrucci, F. Cloud Forensic Readiness: Foundations, *Proc. of the* 5^{th} *International Conference on DF & Cyber Crime (ICDF2C),* 2013, Springer International Publishing, LNICST series, vol. 132, pp. 237-244.

[37] De Marco, L., Abdalla, S., Ferrucci, F., Kechadi, M-T. Formalization of SLAs for Cloud Forensic Readiness, *Proceedings of the* 2^{nd} *International Conference on Cloud Security Management (ICCSM)*, 2014, Academic Conferences and Publishing International Limited, Reading, UK, Dr. Barbara Endicott-Popovsky University of Washington, Seattle, USA Edition, pp. 42 - 50.

[38] De Marco, L., Ferrucci, F., Kechadi, M-T. SLAFM: A Service Level Agreement Formal Model for Cloud Computing. *Proceedings of the* 5^{th} *International Conference on Cloud Computing and Services Science (CLOSER)*, Scitepress 2015, pp. 521-528.

[39] De Marco, L., Ferrucci, F., Kechadi, M-T. A Cloud Forensic Readiness Model for Service Level Agreements Management. *Proceedings of the* 14th *European Conference on Cyber Warfare and Security (ECCWS)* 2015, Academic Conferences Limited, pp. 346-354.

[40] Dykstra, J., Sherman, A.T. Acquiring Forensic Evidence from Infrastructure-as-a-Service Cloud Computing: Exploring and Evaluating Tools, Trust, and Techniques, *Proc. of the* 12th *Annual DF Research Conference, DFRWS,* Digital Investigation, 2012, vol. 9, pp. 9098.

[41] Dykstra, J., Sherman, A.T. Design and Implementation of FROST: Digital Forensic Tools for the OpenStack Cloud Computing Platform, *preprint submitted to the* 13th *Annual DFRWS Conference*, 2013.

[42] Emeakaroha, V. C., Brandic, I., Maurer, M., Dustdar, S. Low level metrics to high level SLAs-LoM2HiS framework: Bridging the gap between monitored metrics and SLA parameters in cloud environments, *High Performance Computing and Simulation (HPCS),* 2010 International Conference on, IEEE, pp. 48-54.

[43] Emeakaroha, V.C., Calheiros, R.N., Netto, M.A., Brandic, I., De Rose, C.A. DeSVi: An Architecture for Detecting SLA Violations in Cloud Computing Infrastructures, *ICST Cloud Comp Conference, 2010.*

[44] Emeakaroha, V.C., Ferreto, T.C., Netto, M.A.S., Brandic, I., De Rose, C.A.F. CASViD: Application Level Monitoring for SLA Violation Detection in Clouds, *IEEE COMPSAC Conference,* 2012, pp. 499 - 508.

[45] Emeakaroha, V.C., Netto, M.A., Calheiros, R.N., Brandic, I., Buyya, R., De Rose, C.A. Towards Autonomic Detection of SLA Violations in Cloud

Infrastructures, *Future Generation Computer Systems,* 2012, vol. 28, issue 7, pp. 1017-1029.

[46] Endicott-Povsky, B., Frinckle, D.A. A theoretical framework for organizational network forensic readiness. *J. Comput.* 2007, vol 2, issue 3, pp. 111.

[47] European Network and Information Security Agency (ENISA). *Cloud Computing: Benefits, risks and recommendations for information security* 2009 [on-line] http://www.enisa.europa.eu/activities/risk-management/files/deliverables/cloud-computing-risk-assessment

[48] European Commission, *Cloud Computing Service Level Agreements - Exploitation of Research Results*, Directorate General Communications Networks, Content and Technology Unit E2 Software and Services, Cloud, 2013 [on-line], Editor: Dimosthenis Kyriazis, http://ec.europa.eu/information_society/newsroo m/cf/dae/document.cfm?doc_id=2496

[49] European Commission - DG CONNECT Cloud Select Industry Group, Cloud Service Level Agreement Standardisation Guidelines, 2014, [on-line] https://ec.europa.eu/digital-agenda/en/news/cloud-service-level-agreement-standardisation-guidelines, C-SIG-SLA.

[50] Ford, G. *Service level agreements,* New Review of Academic Librarianship, 1996, vol. 2, issue 1, pp. 49-58.

[51] Foster, I., Zhao, Y., Raicu, I., Lu, S. Cloud computing and grid computing 360-degree compared. *Grid Computing Environments Workshop, GCE*, 2008, IEEE, pp. 1-10.

[52] Freiling, F.C., Schwittay, B. A Common Process Model for Incident Response and Computer Forensics. *IMF,* 2007, vol. 7, pp. 19-40.

[53] Garfinkel, S.L. Digital forensics research: The next 10 years. *Digital Investigation,* 2010, vol. 7, Supplement, pp. 64-73.

[54] Gartner, *Forecast Overview: Public Cloud Services, Worldwide, 2011-2016*, 2013 [on-line] http://www.forbes.com/sites/louiscolumbus/2013/02/19/gartner-predicts-infrastructure-services-will-accelerate-cloud-computing-growth/

[55] Gebhardt, T., Reiser, H.P. Network Forensics for Cloud Computing. *Distributed Applications and Interoperable Systems,* 2013, pp. 29-42.

[56] General Architecture for Text Engineering - GATE - *University of Sheffield*, [on-line] https://gate.ac.uk/

[57] Ghosh, N., Ghosh, S.K. An Approach to Identify and Monitor SLA Parameters for Storage-as-a-Service Cloud Delivery Model. *GC Wkshps,* 2012, pp. 724-729.

[58] Grishman, R. Information extraction: Techniques and challenges. *Information extraction a multidisciplinary approach to an emerging information technology*. Springer Berlin Heidelberg, 1997, pp. 10-27.

[59] Grobler, T., Louwrens, B. Digital forensic readiness as a component of information security best practice. *Proc. of New Approaches for Security, Privacy and Trust in Complex Environments, IFIP TC- 11,* 22nd *International Information Security Conference*, 2007, vol. 232, pp. 13-24.

[60] Grobler, C.P., Louwrens, C.P., von Solms, S.H. A Multi-Component View of Digital Forensics. *Proc. ARES Conference,* 2010, pp. 647-652.

[61] Hasan, R., Raghav, A., Mahmood, S., Hasan, M.A. Artificial Intelligence Based Model for Incident Response. *Proc. ICIII Conference,* 2011, vol. 3, pp. 91-93.

[62] Hildebrandt, M., Kiltz, S., Grossmann, I., Vielhauer, C. Convergence of Digital and Traditional Forensic Disciplines: a First Exemplary Study for Digital Dactyloscopy. *Proc. ACM MM& Sec Workshop,* 2011, pp. 1-8.

[63] Information Technology Infrastructure Library (ITIL), [on-line] http://www.itil-officialsite.com

[64] JEVAL, [on-line] jeval.sourceforge.net

[65] Keller, A., Ludwig, H. The WSLA framework: Specifying and monitoring service level agreements for web services. *Journal of Network and Systems Management*, 2003, vol. 11, issue 1, pp. 57-81.

[66] Kent, K., Chevalier, S., Grance, T., Dang, H. Guide to Integrating Forensic Techniques into Incident Response. *Special Publication 800-86, National Institute of Standards and Technology (NIST)*, Gaithersburg, Maryland, 2006.

[67] Khatir, M., Hejazi, S.M., Sneiders, E. Two-Dimensional Evidence Reliability Amplification Process Model for Digital Forensics. *Proc. WDFIA Workshop* 2008, pp. 21-29.

[68] Khatir, M., Hejazi, S.M. How to Find Exculpatory and Inculpatory Evidence Using a Circular Digital Forensics Process Model. *Int. J. Electron. Secur. Digit. Forensic,* 2009, vol. 2, no. 1, pp. 68-76.

[69] Kitchenham, B., Charters, S. Guidelines for performing Systematic Literature Reviews in Software Engineering. *EBSE Technical Report*, 2007, vol. 1, pp. 1-57.

[70] Kitchenham, B., Pretorius, R., Budgen, D., Brereton, O.P., Turner, M., Niazi, M., Linkman, S. Systematic Literature Reviews in Software Engineering A Tertiary Study. *Information and Software Technology,* 2010, vol. 52, issue 8, pp. 792-805.

[71] Kohn, M.D., Eloff, J.H.P., Olivier, M.S. UML Modelling of Digital Forensic Process Models. *Proc. ISSA Conference,* 2008, pp. 1-13.

[72] Kohn, M.D., Eloff, M.M., Eloff, J.H.P. Integrated Digital Forensic Process Model. *Computers & Security,* 2013, vol. 38, pp. 103-115.

[73] Kumar, K., Sofat, S., Aggarwal, N., Jain, S.K. Identification of User Ownership in Digital Forensic using Data Mining Technique. *International Journal of Computer Applications,* 2012, vol. 50, no. 4, pp. 1-5.

[74] Ibrahim, M., Abdullah, M.T., Dehghantanha, A. A VoIP Evidence Model: A New Forensic Method for Investigating VoIP Malicious Attacks. *Proc. CyberSec Conference,* 2012, pp. 201-206.

[75] Ishakian, V., Lapets, A., Bestavros, A., Kfoury, A. Formal Verification of SLA Transformations, *IEEE World Congress on Services,* 2011, pp. 540-547.

[76] Larson, K. D. The role of service level agreements in IT service delivery. *Information Management & Computer Security,* 1998, vol. 6, issue 3, pp. 128-132.

[77] Lee, J., McCarthy,J., Licklider, J.The beginnings at MIT. *IEEE Annals of the History of Computing*, 1992, vol 14, issue 1, pp. 18-30.

[78] Lim, K.S., Lee, S.B., Lee, S. Applying a Stepwise Forensic Approach to Incident Response and Computer Usage Analysis. *Proc. CSA Conference,* 2009, pp. 1-6.

[79] Liu, F., Tong, J., Mao, J., Bohn, R., Messina, J., Badger, L., Leaf, D. NIST Cloud Computing Reference Architecture. *NIST Special Publication 500-292*. 2011.

[80] McKemmish, R.*What is forensic computing? Trends and issues in crime and criminal justice*, Canberra, Australian Institute of Criminology, 1999.

[81] Mell, P., Grance, T. *Final Version of NIST Cloud Computing Definition,* 2011 [on-line], http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[82] Maurer, M., Brandic, I., Sakellariou, R. Self-Adaptive and Resource-Efficient SLA Enactment for Cloud Computing Infrastructures. *IEEE CLOUD Conference,* 2012, pp. 368 - 375.

[83] Morshedlou, H., Meybodi, M.R. Decreasing Impact of SLA Violations: A Proactive Resource Allocation Approach for Cloud Computing Environments. *IEEE Transactions on Cloud Computing,* 20014, vol.2, no.2, pp. 156-167.

[84] Mouton, F., Venter, H.S. A prototype for achieving digital forensic readiness on wireless sensor networks. *AFRICON*, 2011, pp. 16.

[85] Muller, C., Oriol, M., Franch, X., Marco, J., Resinas, M., Ruiz-Corts, A., Rodriguez, M. Comprehensive explanation of SLA violations at runtime.

IEEE Transactions on Services Computing, 2014, vol. 7, issue 2, pp. 168-183.

[86] Napoli, G. Sviluppo e sperimentazione di un tool per lestrazione automatica di vincoli contrattuali da Service Level Agreement. University of Salerno, master thesis in computer science, 2015.

[87] Napolitano, S. A Log based tool for monitoring service level agreements violations in cloud environments. University of Salerno, master thesis in computer science, 2015.

[88] Nasumi Benchmark [on-line] http://cache.nasuni.com/Resources/Nasuni_Cloud_Storage_Benchmark_Report.pdf

[89] National Institute of Justice. *Electronic Crime Scene Investigation Guide: A Guide for First Responders*, 2008, [on-line] http://www.nij.gov/publications/pages/publication-detail.aspx?ncjnumber=219941

[90] National Institute of Justice. *Electronic Crime Scene Investigation: A Guide for First Responders.* 2011 [on-line] http://www.ncjrs.org/pdles1/nij/187736.pdf

[91] Ngobeni, S., Venter, H., Burke, I. The Modelling of a Digital Forensic Readiness Approach for Wireless Local Area Networks. *Journal of Universal Computer Science,* 2012, vol. 18, no. 12, pp. 1721-1740.

[92] Noureldin, S.H., Hashem, S., Abdalla, S. Computer Forensics Guidance Model with Cases Study. *Proc. MINES,* 2011, pp. 564-571.

[93] Open Virtualization Format. *OVF Standard.* [on-line] http://www.dmtf.org/standards/ovf

[94] Orton, I., Aaron, A., Endicott-Popovsky, B. Legal Process and Requirements for Cloud Forensic Investigations. *Cybercrime and Cloud Forensics: Applications for Investigation Processes*, ed. Keyun Ruan, IGI Global, Forthcoming, 2012.

[95] Palmer, G. A Road Map for Digital Forensic Research. *Report from the First Digital Forensic Research Workshop (DFRWS),* 2001.

[96] Pangalos, G., Ilioudis, C., Pagkalos, I. The importance of corporate forensic readiness in the information security framework. In: 19^{th} *IEEE International Workshop on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE)*, 2010, pp. 1216.

[97] Paschke, A., Bichler, M. Knowledge Representation Concepts for Automated SLA Management, *Decision Support Systems,* 2008, vol. 46, issue 1, pp. 187-205.

[98] Patel, P., Ranabahu, A.H., Sheth, A.P. *Service Level Agreement in Cloud Computing*, 2009 [on-line], http://corescholar.libraries.wright.edu/knoesis/78

[99] Perumal, S. Digital Forensic Model Based On Malaysian Investigation Process. *International Journal of Computer Science and Network Security,* 2009, vol. 9, no.8, pp. 38-44.

[100] Peterson, J. LittleS3 - Github - https://github.com/igorhvr/littles3

[101] Pilli, E.S., Joshi, R.C., Niyogi, R. Network Forensic Frameworks: Survey and Research Challenges. *Digital Investigation,* 2010, vol. 7, no. 1, pp. 14-27.

[102] Pollitt, M. An Ad Hoc Review of Digital Forensic Models. *Systematic Approaches to Digital Forensic Engineering, Second International Workshop on,* 2007, pp.43-54.

[103] Pollitt, M. A History of Digital Forensics. *IFIP Int. Conf. Digital Forensics*, 2010, pp. 3-15.

[104] Rankin, S. *Forensics Science Central.* 2005 [on-line] http://forensicsciencecentral.co.uk/history.shtml

[105] Rasmi, M., Jantan, A., Al-Mimi, H. A new Approach for Resolving Cyber Crime in Network Forensics Based on Generic Process Model. *Proc. ICIT Conference,* 2013, pp. 1-12.

[106] Reddy, K., Venter, H.S. The architecture of a digital forensic readiness management system. *Computers & Security,* 2013, vol. 32, pp. 73-89.

[107] Reilly, D., Wren, C., Berry, T. Cloud Computing: Pros and Cons for Computer Forensics Investigations. *International Journal of Multimedia and Image Processing (IJMIP)*, 2011, vol. 1, pp. 26-34.

[108] Reith, M., Carr, C., Gunsch, G. An Examination of Digital Forensic Models. *International Journal of Digital Evidence,* 2002, vol.1, issue 3, pp. 1-12.

[109] Rekhis, S., Boudriga, N. A System for Formal Digital Forensic Investigation Aware of Anti-Forensic Attacks. *IEEE Transactions on Information Forensics and Security,* 2012, vol. 7, no. 2, pp. 635-650.

[110] Rogers, M., Goldman, J., Mislan, R., Wedge, T., Debrota, S. Computer Forensics Field Triage Process Model. *Journal of Digital Forensics, Security and Law,* 2006, vol. 1, no. 2, pp. 19-37.

[111] Rowlingson, R., A ten step process for forensic readiness. *International Journal of Digital Evidence*, 2004, vol. 2, issue 3, pp. 1 28.

[112] Ruan, K., Carthy, J., Kechadi, T., Crosbie, M. Cloud forensics: an overview. *Advances in Digital Forensics VII,* 2011.

[113] Ruan, K., Baggili, I., Carthy, J., Kechadi, T. Survey on cloud forensics and critical criteria for cloud forensic capability: a preliminary analysis. *Proc. of the* 6^{th} *Annual Conference on Digital Forensics, Security and Law*, 2011.

[114] Ruan, K., Carthy, J. Cloud Computing Reference Architecture and its Forensic Implications: A Preliminary Analysis. *Proc. of the* 4^{th} *International Conference on Digital Forensics & Cyber Crime (ICDF2C),* 2012, vol. 114, pp. 1-21.

[115] Ruan, K., Carthy, J., Kechadi, T., Baggili, I. Cloud forensics definitions and critical criteria for cloud forensic capability: An overview of survey results. *Digital Investigation*, 2013, vol. 10, issue 1, pp.-34-43.

[116] Rumbaugh, J., Jacobson, I., Booch, G. *Unified Modeling Language Reference Manual, The*. Pearson Higher Education, 2004.

[117] Sansurooah, K.Taxonomy of Computer Forensics Methodologies and Procedures for Digital Evidence Seizure *Proc. ADF Conference,* 2006, pp. 1-13.

[118] Simple DoS Attack, https://github.com/jtf323/SimpleDOSAttack

[119] Sommerville, I., Sawyer, P. Requirements engineering: a good practice guide. *John Wiley and Sons, Inc.*, 1997.

[120] Skene, J., Skene, A., Crampton, J., Emmerich, W. The Monitorability of Service-Level Agreements for Application-Service Provision, *Proc. International Workshop on Software and Performance,* 2007, pp. 3-14.

[121] Stephenson, P. Analysis and Correlation. *Computer Fraud and Security,* 2002, vol. 12, pp. 16-18.

[122] Sood, A., Tellis, G.J. Technological Evolution and Radical Innovation. *Journal of Marketing* , 2005, vol. 69, no. 3, pp. 152-168.

[123] Tan, J., Forensic Readiness, Technical report, @*Stake Organization*, 2001 [on-line] http://isis.poly.edu/kulesh/forensics/forensic_readiness.pdf

[124] Trenwith, P. M., Venter, H.S. Digital forensic readiness in the cloud. *Proc. of Information Security for South Africa*, 2013, pp.1-5.

[125] Unger, T., Leymann, F., Mauchart, S., Scheibler, T. Aggregation of Service Level Agreements in the Context of Business Processes. *Proc. ICEDOC Conference*, 2008, pp. 43-52.

[126] Valjarevic, A., Venter, H.S. Towards a Digital Forensic Readiness Framework for Public Key Infrastructure systems. *Proc. of Information Security South Africa (ISSA),* 2011, pp.1-10.

[127] Valjarevic, A., Venter, H.S. Harmonised Digital Forensic Investigation Process Model. *Information Security for South Africa (ISSA),* 2012, pp. 1-10.

[128] Yusoff, Y., Ismail, R., Hassan, Z. Common Phases of Computer Forensics Investigation Models. *International Journal of Computer Science & Information Technology,* 2011, vol. 3, issue 3, pp. 17-31.

[129] Ward, B.T., Sipior, J.C. The Internet Jurisdiction Risk of Cloud Computing. *Information Systems Management,* 2010, vol. 27, issue 4, pp. 334-339.

[130] World Wide Web Consortium Home Page, [on-line] http://www.w3.org

[131] Zargari, S., Benford, D. Cloud Forensics: Concepts, Issues, and Challenges. 3rd *International Conference on Emerging Intelligent Data and Web Technologies (EIDWT),* IEEE, 2012, pp. 236-243.