NTNU – Trondheim
Norwegian University of
Science and Technology

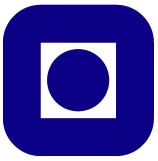
Automated Security Compliance Tool for the Cloud

Kazi Wali Ullah

Master in Security and Mobile Computing
Submission date: June 2012
Supervisor: Danilo Gligoroski, ITEM
Co-supervisor: Tuomas Aura, Aalto University

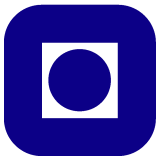
Norwegian University of Science and Technology
Department of Telematics

NORWEGIAN UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY, MATHEMATICS AND ELECTRICAL ENGINEERING



PROBLEM DESCRIPTION

Student's Name:	Kazi Wali Ullah
Project Title:	Automated Security Compliance Tool for the Cloud
Project Description:	Security, especially security compliance, is a major concern for large scale adoption of clouds in the enterprise environment. For business reason, enterprises require certain levels of security compliance from cloud providers. Thus, we need an automated compliance tool to express the service level of various cloud providers. The tool will enable the cloud providers to share their security compliance information using a common framework. In turn, the common framework allows consumers to shop intelligently among various cloud providers based on their security needs. The security compliance can be based on a standard compliance policy (e..g., HIPAA) or it can be more granular level (e.g., multi-factor authentication = true, firewall = true, VPN from project to users = true). The scope of this thesis is to integrate a compliance mechanism (e.g., CloudAudit frameworks) to a cloud platform (e.g., OpenStack). The expected output is to expose a set of APIs for OpenStack cloud with which consumers can verify the security requirements.
Department:	Department of Telematics
Responsible Professors:	Tuomas Aura, Aalto University, Finland Danilo Gligoroski, NTNU, Norway
Submission Date:	30-06-2012



Norwegian University of Science & Technology

Degree Programme in Security and Mobile Computing

ABSTRACT OF
MASTER'S THESIS

Author:	Kazi Wali Ullah		
Title:	Automated Security Compliance Tool for the Cloud		
Date:	June 30, 2012	**Pages:**	viii + 67
Professorship:	Data Communication Software	**Code:**	T-110
Supervisors:	Professor Tuomas Aura Professor Danilo Gligoroski		
Instructors:	Jukka Ylitalo D.Sc. (Tech.) Abu Shohel Ahmed M.Sc. (Tech.)		

Security, especially security compliance, is a major concern that is slowing down the large scale adoption of cloud computing in the enterprise environment. Business requirements, governmental regulations and trust are among the reasons why the enterprises require certain levels of security compliance from cloud providers. So far, this security compliance or auditing information has been generated by security specialists manually. This process involves manual data collection and assessment which is slow and incurs a high cost. Thus, there is a need for an automated compliance tool to verify and express the compliance level of various cloud providers. Such a tool can reduce the human intervention and eventually reduce the cost and time by verifying the compliance automatically. Also, the tool will enable the cloud providers to share their security compliance information using a common framework. In turn, the common framework allows clients to compare various cloud providers based on their security needs. Having these goals in mind, we have developed an architecture to build an automated security compliance tool for a cloud computing platform. We have also outlined four possible approaches to achieve this automation. These possible four approaches refer to four design patterns to collect data from the cloud system and these are: API, vulnerability scanning, log analysis and manual entry. Finally, we have implemented a proof-of-concept prototype of this automated security compliance tool using the proposed architecture. This prototype implementation is integrated with OpenStack cloud platform, and the results are exposed to the users of the cloud following the CloudAudit API structure defined by Cloud Security Alliance.

Keywords:	Cloud Security Compliance, Cloud Audit, Cloud Control Matrix (CCM), OpenStack, OpenVAS, OSAPI
Language:	English

Acknowledgements

It is a pleasure to thank those who made this thesis possible. I would like to specially thank my home supervisor, Prof. Tuomas Aura, for his time and valuable suggestions during the whole thesis work. At the same time, I am grateful to my host supervisor, Prof. Danilo Gligoroski, for his timely help and suggetions regarding my thesis.

I owe my gratitude to my instructors, Jukka Ylitalo and Abu Shohel Ahmed, from NomadicLab Ericsson Research Finland, for their constant support and guidance during the thesis work. Both of their knowledge and experience in this field have helped me greatly to overcome many challenges during my thesis work.

I would also like to thank NoadicLab, Ericsson Research Finland, for providing me this excellent working environment.

Last, but not the least, I want to express my gratitude to my wife, Fariha, who keep inspiring me during the thesis work.

Espoo, June 30, 2012

Kazi Wali Ullah

Abbreviations and Acronyms

AMQP	Advanced Message Queuing Protocol
API	Application Programming Interface
CCM	Cloud Control Matrix
CLI	Command Line Interface
COBIT	Control Objectives for Information and related Technology
CSA	Cloud Security Alliance
CSP	Cloud Service Provider
EC2API	Elastic Compute Cloud Application Programming Interface
DSS	Data Security Standard
GRC	Governance, Risk Management and Compliance
GUI	Graphical User Interface
HIPAA	Health Insurance Portability and Accountability Act
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
IaaS	Infrastructure as a Service
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
NASL	Nessus Attack Script Language
NIST	National Institute of Standards and Technology
NTP	Network Time Protocol
NVT	Network Vulnerability Tests
OAP	OpenVAS Administration Protocol
OMP	OpenVAS Management Protocol
OTP	OpenVAS Transfer Protocol
OpenVAS	Open Vulnerability Assessment System
OSAPI	OpenStack Application Programming Interface
PaaS	Platform as a Service

PCI	Payment Card Industry
REST	Representational State Transfer
SaaS	Software as a Service
SLA	Service Level Agreement
UTC	Co-ordinated Universal Time
VM	Virtual Machine
WSGI	Web Service Gateway Interface
XML	Extensible Markup Language

Contents

List of Figures

Chapter 1

Introduction

The number of cloud service providers (CSP) is increasing fast as the widely used pay-per-use business model has attracted millions of customers over the world. This increasing number of cloud vendors facilitates the potential customers to get more variety options to meet the requirements of their product. At the same time, the customer has to compare and evaluate many different cloud vendors to select the most suitable vendor for their products. Since, most of these cloud vendors today are using proprietary solutions to deliver cloud based services, it is difficult to compare different cloud vendors under common evaluation criteria.

The problem arises due to several important facts residing in the cloud computing arena. One of the most important issues is that the cloud vendors are not applying open standards available for cloud computing to build the cloud services. The most important reasons behind this are that the cloud computing standardization work is still ongoing and that there exist many proprietary solutions today that have been launched even before the standards were developed. Another important issue is the cost associated with implementing cloud services according to standards. Cloud vendors will not willingly incur that cost unless there is a significant demand from the large portion of the customer base to implement the standards. Again, there is lack of motivation for some vendors to be open and compliant with standards rather than using their closed proprietary solutions.

If all the cloud vendors are put to use open standards, then the sole problem would be to analyze the features provided by a cloud vendor against those standards. In reality, the use of proprietary, non-standard solutions has made the auditing procedure very challenging and complex. The same problem

is at hand when it comes to analyzing the security measures of a cloud vendor for compliance with standards. While the security issues associated with cloud computing have been under continuous study to make the cloud computing more and more secure, there is no mechanism available today to compare the security features provided by the different cloud vendors against the standards. In addition, there is no mechanism as of today to verify, in real time, the security features implemented by a cloud vendor. And this is identified as one of the topmost demand by the users in the Martin Kuppinger *Top Trends 2012-2013 Report* [51]. In this report, it is also mentioned that since the Cloud is beyond the immediate control of IT, there will still be a lack of tools and standards in the areas of authorization and auditing. Also, the lack of auditability in real time or near-real time is one of the major obstacles for large scale adoption of cloud computing [55].

Therefore, this thesis explores the possibility to build an automated security compliance tool for cloud computing that will allow a cloud user to verify the security measures against the standards in an on demand basis. However, the security area for cloud computing is itself a broad topic and there are several different cloud platforms with different properties to work with. Therefore, in this thesis work, we aim to develop a proof-of-concept automated security compliance tool focusing on one cloud platform.

1.1 Research Problem

By far, the most important concern with cloud computing is the various security issues. Although all the major cloud vendors present today provide many security measures for their clients, it is impossible for a client to verify or compare the security measures provided by different cloud service providers under a common security evaluation platform. Being aware of this issue, Cloud Security Alliance (CSA) [6] has developed some guidelines and frameworks to facilitate a common open, extensible and secure interface through which a cloud provider is able to provide security assurance to its customers. CloudAudit [8] is one such framework developed by CSA.

While the CloudAudit deals with providing a common interface for the auditing process for a cloud service provider, Cloud Control Matrix (CCM) [5] is developed to guide cloud vendors and its customers to assess the risks related to a particular cloud service provider. The CCM provides a detailed guideline incorporating industry accepted standards such as ISO 27001/27002 [20, 21], HIPAA [34], ISACA COBIT [10], PCI DSS [25], NIST security standards

[24]. Therefore, if all the cloud vendors present today implement their systems by the regulations summarized in the CCM and provide a common interface, using the CloudAudit framework, for a client to verify the security measures, then a client can confidently verify, assess and compare the risks from different cloud vendors.

Although the framework and guidelines from CCM and CloudAudit are already there to facilitate the security-compliance-related information flow inside a cloud vendor, yet how to automatically generate this compliance-related information is still under research. Even if a cloud vendor advertises that it has implemented all the security measures according to the standards, there is no way to verify such a claim on demand by a user. To overcome this situation, we aim to automate the risk assessment process in a cloud vendor. In order to automate the risk assessment process, we plan to use various techniques that will allow us to generate the required information on demand and without human intervention.

In this thesis, we aim to build an intelligent engine that acquires the necessary information on demand from a target cloud system. The acquired information is tested by the engine for compliance with CloudAudit, CCM and other standards. Finally, the results will be passed on to the user. To accomplish this task, we have chosen the open source cloud computing platform OpenStack [26], which supports the Infrastructure as a Service (IaaS) cloud service model.

Before we can jump into the actual development, we need to find answers to a few questions. The first question that we need to answer is which parts of the auditing can be automated. As it is not possible to automate everything, we need to answer this question and focus on a small part that we are going to automate in this project. The second challenging question is how to integrate the automated auditing process in the targeted cloud infrastructure (OpenStack). And the final question is what approaches are there that can be utilized for the automated security compliance analysis based on the CSA requirements.

1.2 Goals of the Thesis

This thesis project is primarily focused on cloud security compliance, with the following goals:

1. Analyze the CloudAudit and Cloud Control Matrix (CCM) for cloud

security specifics that is going to be used in order to build the framework. Here, the cloud security specifics refer to the cloud security controls, interface definitions, relation with different standards, etc.

2. Analyze different approaches to automating the compliance check process in light of the OpenStack cloud computing platform.
3. Design an architecture for building an automated security compliance tool for the cloud incorporating CloudAudit, CCM and other necessary 3rd party tools in the OpenStack cloud computing platform.
4. Build a proof-of-concept prototype of the automated security compliance tool for cloud.

1.3 Contributions

The main contribution of this thesis is that we have developed a novel architecture for building an automated security compliance tool for the cloud. Although our architecture is developed focusing on a specific cloud platform, OpenStack, it can be used in other cloud platforms as well with only minor modifications. Furthermore, we have also implemented a proof-of-concept solution using this architecture integrated with OpenStack cloud platform. While developing the architecture, we have found four possible ways of collecting data from the cloud to build the automatic security compliance check. Two of these four possible ways are novel and we have implemented these two ways as a proof of concept.

Besides the main contribution stated above, another important contribution can be pointed out here. During the thesis work, we have reached the conclusion that many of the compliance checks can not be done automatically. Many of the security controls in the standards need to be checked manually or can be automated only partially. Considering this issue, we have developed our architecture in such a way that the administrator of a cloud vendor can insert manual entries in the automated security compliance tool for these controls. These manual entries, however, cannot be verified by the tool and hence create a trust issue between the client and the vendor.

1.4 Structure of the Thesis

The rest of the thesis is organized as follows: Chapter 2 introduces the necessary background information that is required to understand the technologies involved in this project. Chapter 3 gives an overview of the design decisions that we have made to implement the solution. Chapter 4 gives detailed information about the implementation of our solution. In chapter 5, we evaluate our work against various evaluation criteria. Chapter 6 discusses some analytical perspectives on the developed solution. And finally in Chapter 7, we conclude our report along with some suggestions for future work.

Chapter 2

Background

This chapter introduces the necessary background information that is essential for understanding the rest of this thesis project. Since our thesis project is about security compliance, we start this chapter by describing briefly what security compliance is. Later, we move on to cloud computing, related security terms, frameworks and standards. Finally, we introduce an open source vulnerability scanner named OpenVAS that we have used in our prototype implementation of an automated security compliance tool for cloud.

2.1 Security Compliance

In order to understand the security compliance, we have to distinguish it from security itself. While security refers to a mechanism that have to be used in order for a system to be in a safe state from prospective threats, security compliance refers to a state of compliance with a given set of security requirements. Therefore, while security itself is used to protect a system from threats, security compliance has nothing to do with this protection. Rather, security compliance ensures that the security measures taken to protect the system are compliant with the necessary requirements. In general, the audit and compliance refers to the process that an organization implements to achieve the followings [53]:

- Identifying the set of requirements that the organization must abide with.
- Acting accordingly so that the requirements are met.

- Monitoring the systems that the processes are followed consistently.

To focus more on the security side of the compliance procedure, Klaus Julisch from IBM Research has defined the security compliance as follows [49]:

> *"Security compliance, in IT systems, is the state of conformance with externally imposed functional security requirements and of providing evidence (assurance) thereof."*

Now we can summarize the security compliance as to comply, for a system, with external security requirements. This external security requirements can be the government issued regulations, industry accepted best practices or any internal company policies. However, these days security compliance generally indicates the compliance with industry accepted security standards such as NIST, ISO 270001/27002, HIPAA, PCI, etc. This is the compliance that we have targeted to achieve in this thesis project. Although there is a human behavioural side of the security compliance whether an employee wants to comply with the policy or not [59], we, in this project, focus only in the technical part of the security compliance.

2.1.1 Why Security Compliance?

Cloud computing can be seen as a new term for an old trend. This viewpoint arises from the fact that cloud computing is generally used to deliver the same old products such as email service or web service using a different mechanism. It is important to realize that we already have well defined protocols and standards for these sort of services for many years. Therefore, the question arises why it is important to have the security compliance for cloud infrastructure while it is providing the same set of services. Nathaniel Borenstein and James Blake from Mimecast [14] have answered this question by saying that this compliance is important to gain the trust of the nervous users [41]. This is understandable as the companies willing to move towards a cloud service provider to deliver their product, looses the control over the underlying system and do not know the inner workings of the cloud systems. Hence, the clients opt for a cloud vendor to be compliant with standards that they can trust.

While compliance helps drive security, it does not equal actual security. Nonetheless, if a system is compliant with a well-established security standard, it can survive the most common security threats. The *2012 Data*

Breach Investigation Report [63] presented by Verizon [64] outlines the fact that non-compliance is one of the main reasons for data breaches in the Payment Card Industry. In this report, it was stated that 96% of the companies that suffered the breach have not achieved compliance with the PCI DSS. Only the remaining 4% of companies were still under attack despite having achieved the compliance with PCI DSS. This is a clear indication of how much difference can it make to have the security compliance.

There are several other important reasons for security compliance in general. The first important and essential use of security compliance is the auditing procedure. It is because of the fact that what is being audited and enforced is compliance, not security. The second important aspect of compliance is that despite extensive research [52], it is difficult to measure the security, in general, for a system. However, measuring for compliance is feasible and there are matrices published for this purpose [48, 65]. The third importance of compliance, specially the security compliance, is that it plays a significant role in ensuring governance and service level agreements (SLA) between the cloud vendor and the client as indicated in [43]. Finally, in today's world, security compliance or auditing plays a significant role for a security tool to be successful in business. If there is a new security tool that pops out of some research lab which is not recognized or used by any security auditor, there may be no value or business for that tool.

2.1.2 Challenges in Automating Security Compliance Check

Security compliance check refers to verifying a system against some security standards to determine whether the system complies with the standard or not. So far, there have been manual auditing procedures for this purpose. This manual auditing process involves data collection and decision making by security experts and generally costs a lot of money and time. In contrast, using automated security compliance check procedures, human intervention can be reduced to a great extent which can be very time and cost efficient. However, we have identified several challenges that need to be overcome in order to build an automated security compliance tool for a system. These challenges are listed in the following:

- The first challenge in automating the security compliance is to formalize the set of external requirements with which the system has to comply. The requirements determination is difficult due to having a large num-

ber of standards and the fact that not all standards are suitable for all types of systems. At a more granular level, even every security control of a standard may not be appropriate for all systems.

- Unfortunately, the standards that we have at our disposal today, are very abstract with no or minimal guidance for implementation. This property of the standards have made the automation process extremely difficult, as for the implementation of some controls, heuristic values need to be chosen to verify the compliance status.
- Third challenge is to determine what data or information needs to be extracted from the system to verify the security controls.
- Fourth challenge is to determine a feasible way to extract these data. Some of the information required for verification can be obtained externally while there are some information that can only be extracted internally by the system itself.
- To extract these data, the system may need to be modified which can be challenging for an already deployed and functional system.
- Data must be delivered in a secure way to the authorized compliance tool so that it does not fall into the hands of an attacker.
- Finally, providing assurance for the compliance status determined by the automated tool is also a big challenge. Since there may be some heuristic values to determine the compliance status, client needs to be assured about the decision or needs to be given more information about the compliance check procedure.

Another barrier for automating the security compliance is that many of the controls stated in the standards require manual intervention which can not be automated. For example, the physical security (personnel physical entry or exit to the facility, hardware security, etc.) related controls cannot be verified using the automated security compliance tool.

All the above mentioned challenges are generic in nature and apply to automating the security compliance check for any system. There are even more challenges in the cloud computing platforms to achieve the same security compliance check. Based on the *State of Enterprise Security Report 2010* [61] by Symantec Corporation [60], the most problematic areas from the security perspective are (most problematic on the top):

- Infrastructure as a Service
- Platform as a Service

- Server Virtualization
- Endpoint Virtualization
- Software as a Service

All of the above areas are inherent to cloud computing making the automation of security compliance check much more challenging in the cloud computing arena.

2.2 Cloud Computing

Cloud computing is a relatively new paradigm for delivering computing resources as a service to heterogeneous end users. End users typically use a web-based interface to access the cloud computing service. There may be also desktop-based or mobile-based interfaces to access the cloud services. Users use these interfaces from their machine to access the service while the business logic or the software and data remain in the cloud system. The most common definition used by the researchers to define cloud computing is the definition provided by NIST which is quoted in the following [54]:

> *"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."*

The characteristics that have made cloud computing exalted are: on-demand self-service, broad network access, resource pooling, rapid elasticity and measured service [54]. All of these characteristics are of utmost importance for a company that does not want to have the hassle of maintaining its own computing infrastructure. Therefore, the company can only concentrate on its products while offloading the setup of the computing infrastructure, maintenance, security, etc. to the CSP by paying a fee.

2.2.1 Service Models

From the user perspective of a cloud system, the most important consideration is the service model as it defines the boundary about how much control

the user will be given for the service. There are three different types of service models available in cloud computing which are as follows [54]:

Software as a Service (SaaS): SaaS model allows the consumer of the cloud only to access the application running on a cloud infrastructure. The consumer will have no control over the cloud infrastructure or over the application itself rather than setting some application configuration related parameters. In this model, the client usually do not have any information regarding the underlying components of the system over which the application is running. This makes easier for the consumers as they do not have to maintain the underlying operating system, network, storage, etc. by themselves. All of these resources are maintained by the cloud service provider. There are endless examples of business using SaaS model today, some of the best knowns are Google Docs [16], Salesforce Customer Relationship Management [11], etc.

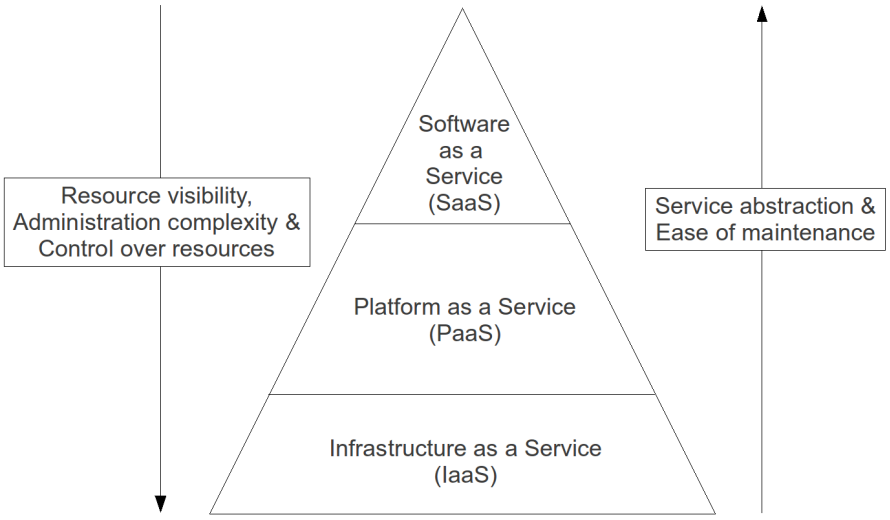


Figure 2.1: Cloud Service Models

Platform as a Service (PaaS): PaaS model allows a consumer to deploy his/her own application into the cloud infrastructure. However, the application created by the consumer must comply with the programming languages, libraries, services and tools provided by the cloud service provider. In this model, the consumer does not have any control on the underlying components on which the application is running, but has complete control over the

application that is owned and deployed by him. Examples of PaaS service provider are Google App Engine [15], Microsoft's Windows Azure [36], etc.

Infrastructure as a Service (IaaS): IaaS model provides the bare computing resources to a consumer upon which a consumer can run any pieces of software at his will. This choice of software includes operating systems and any other application that the consumer requires. In this model, the consumer does not manage the underlying cloud infrastructure. However, the consumer can mange its storage requirements, operating systems and applications and possibly be able to choose some network configurations. Some of the best known IaaS service providers are Amazon Web Services (AWS) [2], Rackspace [4], etc.

2.2.2 Deployment Models

Based on the specific business, operational, and technical requirements, each company chooses a deployment model for a cloud computing solution . There are four different deployment models available today and these are:

Private Cloud: In this deployment model, only a single organization owns or uses the cloud infrastructure. However, the management and maintenance can be administered by the organization itself or by any other third party company.

Community Cloud: This deployment facilitates the use of cloud infrastructure by multiple companies sharing common interests. The shared interests or concerns may include business area, security requirements, business policy, compliance issues etc.

Public Cloud: This deployment scenario occurs when the cloud provider allows the cloud infrastructure to be used by general public. The usage of the cloud service may be free or for a fee depending on the cloud service provider's policy.

Hybrid Cloud: Hybrid cloud deployment occurs when the cloud infrastructure is a combination of two or more above mentioned models. This

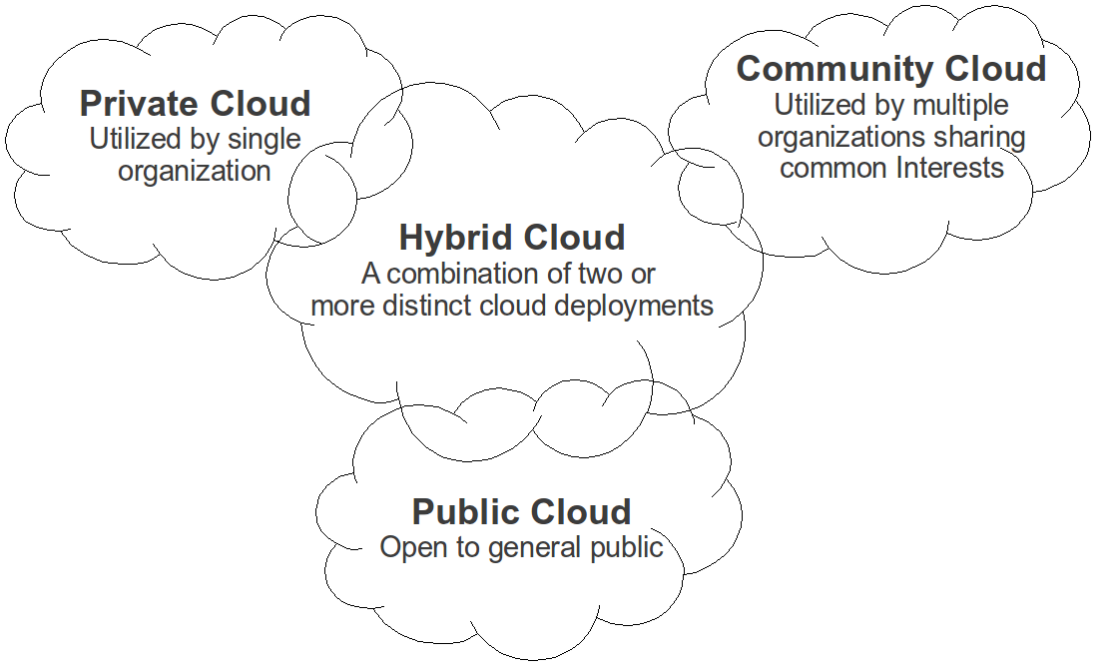


Figure 2.2: Cloud Deployment Models

scenario is useful when an organization has a private cloud to support its regular business, however requires more computing resources due to traffic burst for a short amount of time. During the traffic burst, the organization can balance the load by utilizing a community or public cloud. In this way, the organization can make sure that it never goes overloaded and all the services are delivered in time. However, utilizing a hybrid cloud deployment model requires strong co-ordination of data and control logics between two or more different clouds.

It is this deployment model which defines the level of control that an organization may have over the cloud infrastructure. In the private cloud model, the organization owns the cloud and have the total control over the cloud infrastructure. The reverse is true for the public clouds where an user of the cloud have minimal control over the cloud infrastructure.

2.3 OpenStack: The Selected Cloud Computing Platform

OpenStack [26] is the open source cloud computing platform that has drawn significant amount of attraction from the cloud community in just couple of years. OpenStack is relatively new in the cloud arena and was launched in the year 2010 when the U.S. National Aeronautics and Space Administration (NASA) and Rackspace founded this OpenStack organization for the development of open-source cloud computing platform. Although the OpenStack community initially started with only two companies, NASA and Rackspace, currently (June 18, 2012) the OpenStack community has 180 companies as participating members including companies like Dell, AMD, Intel, Cisco, HP, Ericsson to name a few. The mission of OpenStack includes:

- To produce an open-source Cloud Computing platform that will meet the needs of public and private clouds regardless of size.
- Simple to implement.
- Massively scalable.
- Feature rich.

The initial version of OpenStack was released in October, 2010, and was named Austin. After that OpenStack has released 4 more versions named Bexar, Cactus, Diablo and Essex while Essex is the latest, released in May, 2012. In our thesis project, we have worked with the release Diablo, and from here on, all the discussion about OpenStack will be done in light of the release Diablo.

2.3.1 OpenStack Projects

Being a collaborative software project, OpenStack runs many projects to support its mission. There are currently three core projects running in OpenStack and these are OpenStack Compute, OpenStack Object Storage and OpenStack Image Service. Along with these three core projects, two new projects named OpenStack Identity and OpenStack Dashboard are being incubated with the release Diablo. All of these five OpenStack projects are described briefly in the following paragraphs.

OpenStack Compute: The codename for this project is Nova which was initially developed at NASA. Nova is the cloud computing fabric controller, the main part of IaaS cloud system, designed to provision and manage large networks of virtual machines, creating a redundant and scalable cloud computing platform. Nova provides all the necessary pieces of software, control panels and APIs that are required to create, run and maintain a cloud system. The design guidelines that influenced the development of Nova are - component based architecture, high availability, fault tolerance, recoverable, open standards, API compatibility.

OpenStack Object Storage: The codename for this project is Swift which is also the name given at Rackspace where it was developed initially. Swift allows the creation of object storage using clusters of standardized servers that can store and access data in the range of petabytes. One thing to understand about Swift is that it is not a filesystem or a real time data storage system, rather a long term data storage system intended to store, retrieve, leverage, modify a more permanent type of data such as virtual machine images, photo storage, email storage, etc. Swift provides software logic for data distribution and replication across the servers and therefore cheap commodity hardware can be used to build this massive storage service in stead of using dedicated expensive hardware. Popular use cases of Swift include service providers who wants to provide IaaS based storage service, document storage, archiving and backend of Microsoft SharePoint.

OpenStack Image Service: This project allows OpenStack to provide discovery, registry and delivery services for virtual machine images. As with the earlier projects, this project also has a codename and the name is Glance. Using Glance, a user can query for the stored images that are stored in various back end storages. Glance provides RESTful API services in order to facilitate the querying for the stored images.

OpenStack Identity: This is one of the two new projects included in the Diablo release with the codename Keystone. Keystone implements OpenStack's identity APIs and provides identity, token, catalog and policy services to the OpenStack projects. All the APIs are RESTful and uses SSL over HTTP (HTTPS) to protect the secret credentials.

OpenStack Dashboard: This project provides the baseline user interface for managing OpenStack services. This project is more commonly known with its codename Horizon. Horizon provides an web based user interface for OpenStack and is developed using Django [12] framework. Horizon was initially developed with support for the Nova project only. Later on, it started to support other OpenStack projects such as Swift and Glance. Although Horizon only provides basic user interface for OpenStack services, it is extensible and manageable by an administrator by adding new functionalities into the project.

Among these five above mentioned OpenStack projects, only Nova and Horizon are important from the perspective of the thesis. We have carried out the security compliance check against the Nova system and for that reason we had to add new functionality in the Nova project. Also we have used a modified Horizon project to display our security compliance related information in to the OpenStack dashboard. Since Nova is the most important OpenStack project from this thesis's perspective, we will take a deeper look at Nova in the next section.

2.3.2 OpenStack Nova Architecture

OpenStack is an Infrastructure as a Service (IaaS) cloud computing platform and Nova, also known as OpenStack Compute, is the core piece of software that facilitates the IaaS cloud computing platform for OpenStack. Nova is similar in scope to Amazon EC2 [1] and Rackspace cloud servers [7]. Figure 2.3 depicts the architecture for OpenStack Nova and the components are -

Cloud Controller is at the heart of Nova which interacts with other modules in different ways such as Hyper Text Transfer Protocol (HTTP), Advanced Message Queuing Protocol (AMQP) and local methods. It also manages the global state of the system.

Object Store provides the storage related facilities. OpenStack Swift is an example of object storage service that can be integrated with Nova.

Auth Manager provides the authentication and access related functionality. Initially the *nova.auth.manager* class handled the authentication related

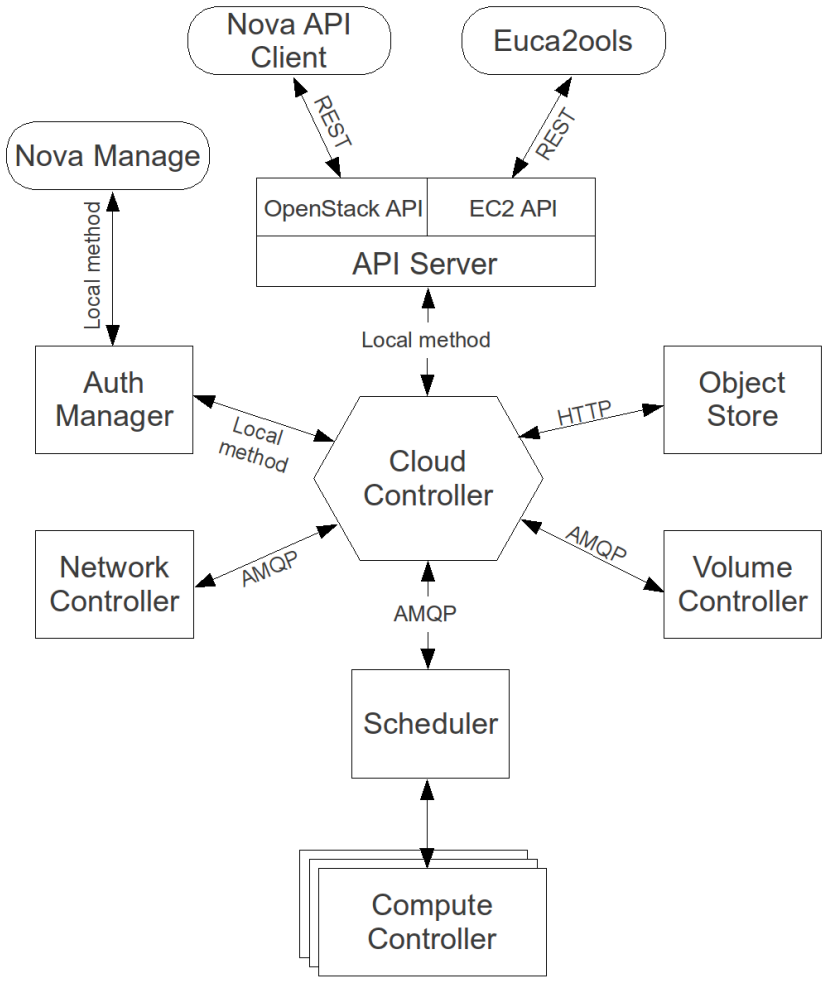


Figure 2.3: OpenStack Nova Architecture

functionality which is now deprecated. OpenStack Keystone project has now taken place in order to provide the authentication functionality.

Volume Controller controls the storage related functionality such as creation, deletion, attaching, detaching, etc. for a compute node. In this way, a compute node can have a persistent storage medium as the storage attached to a compute node is non-persistent and loses all the data when the compute storage is detached or the instance is terminated.

Network Controller configures the network for the host machine in OpenStack. It manages the IP addressing, VPN configurations, creating security

groups etc. for compute nodes.

Scheduler maps the API calls to an appropriate OpenStack component. It also runs an algorithm to select the resource from an available pool of resources for a task.

Compute Controller is responsible for compute resources where the instances are deployed.

API Server is the interface for the outside world to interact and manage the OpenStack cloud infrastructure. It provides RESTful APIs to the outside world and it communicates to other modules of Nova through the cloud controller.

2.4 Information Security Standards

Despite having installed security mechanisms, we often hear about information security breaches. One reason behind this is that the security mechanisms are not applied correctly. To address this issue, information security standards and various regulations have been introduced. These standards and regulations provide a guideline to ensure that an adequate level of security is in place, resources are used properly and the best known security practices are adopted.

These standards and regulations are often initiated by industry bodies, organizations and governments. Some organizations, having same business, can create a standard for that business which everybody has to follow to ensure interoperability, to prevent vendor lock-in, to permit open middleware, etc. One example of such standard is the Payment Card Industry (PCI) Data Security Standard (DSS) [25]. This standard was developed by a number of major credit card companies including American Express, MasterCard worldwide and VISA international.

Some industries are often regulated by the government, and the government introduces acts and regulations for this purpose. One such example is the Health Information Privacy and Accountability Act (HIPAA) [34]. This act was developed by the US government to not only protect the security of

the health-related data but also to improve the portability and continuity of health insurance and for many other purposes.

Besides the above two types of standards and regulations, there are some standards developed which are generic in nature and may be applicable to any information system. Some of the best known information security standards in this aspect are ISO 27001 [20], ISO 27002 [21] and NIST [24]. Since, we have used ISO 27002 in our project for the compliance, we will give a brief description of the ISO 27002 standard in the following section.

2.4.1 ISO 27002

ISO 27002 was developed by International Organization for Standardization (ISO) and the latest version is ISO 27002:v2005 with the title - *"Information technology - Security techniques - Code of practice for information security management"*. This is an internationally accepted standard that has been followed by tens or hundreds of thousands of organizations worldwide as a norm for good practices in the information security arena.

ISO 27002:v2005 is not a true information security standard; it is more like an advisory document. This document provides a structured way to address the information security risks. The best practices in the information security arena are presented in this standard in 11 security domains:

1. Security policy
2. Organizing information security
3. Asset management
4. Human resources security
5. Physical and environmental security
6. Communications and operations management
7. Access control
8. Information systems acquisition, development and maintenance
9. Information security incident management
10. Business continuity management
11. Compliance

These 11 security domains are covered with 39 main security categories and there are hundreds of security controls specified for these main security categories. Each main security categories composed of two components -

1. A control objective that states what is to be achieved by this security category.
2. One or more controls to achieve the control objective.

Each control is composed of three components. These are -

Control statement: Defines what this specific security control does to satisfy the control objective.

Implementation guidance: Provides some guidance to implement this security control.

Other information: Some additional information is provided that may be important when implementing the control.

Example controls from ISO 27002 can be found in section 4.3.1 and section 4.3.2.

2.5 Cloud Control Matrix

The Cloud Control Matrix (CCM) [5] is a control framework with the set of baseline security controls for cloud computing to provide a guideline for the cloud users to assess the risk associated with a cloud service provider. The set of baseline security controls are defined by Cloud Security Alliance (CSA) [6] with the goal of promoting the use of best practices in the cloud computing arena to facilitate the security assurance. The goal of the CSA CCM is to bridge the gap between cloud security and industry accepted security standards and thus to provide a standard for the cloud security area.

The CSA CCM provides a detailed understanding of security principles and concepts that are aligned with the CSA's security guidance [32] in 13 security domains. These 13 domains are denoted as the critical focus areas in cloud computing by CSA:

1. Cloud Computing Architectural Framework
2. Governance and Enterprise Risk Management

3. Legal and Electronic Discovery
4. Compliance and Audit
5. Information Lifecycle Management
6. Portability and Interoperability
7. Traditional Security, Business Continuity, and Disaster Recovery
8. Data Center Operations
9. Incident Response, Notification, and Remediation
10. Application Security
11. Encryption and Key Management
12. Identity and Access Management
13. Virtualization

While this CSA security guidance provides a guideline for security assurance in 13 security domains in cloud computing, the CSA CCM provides a customized relationship with other industry accepted standards present today. These industry accepted standards include - ISO 27001/27002, HIPAA, COBIT, PCI and NIST standard. This mapping between cloud security controls and the industry accepted standards' controls is the foundational core of the CCM.

For the security compliance check in cloud computing platform against a industry accepted standard, CCM provides the starting point by giving the list of security controls from that standard that must be aligned with cloud security. Therefore, by analyzing the CCM, one readily knows which controls to check from the standard for the cloud system to be compliant with that standard.

2.6 CSA CloudAudit Framework

CSA CloudAudit [8, 42] framework is a standardized way to facilitate information regarding the performance and security of a cloud service provider to an authorized client. This standardized way of getting information about performance and security enables a customer to analyze various performance and security status and thus enables comparing the services between different cloud service providers. Also, in this way it is easier for the cloud vendors to

maintain and deliver the information to multiple customers. Using CloudAudit framework, cloud vendors can provide the information once and update it periodically or when any change in the system occurs.

The development code name for CloudAudit was A6 - Automated Audit, Assertion, Assessment and Assurance API. The specification provides, according to the Internet Engineering Task Force (IETF) CloudAudit 1.0 draft document [42], *"a common interface, naming convention, set of processes and technologies utilizing the HTTP protocol to enable cloud service providers to automate the collection and assertion of operational, security, audit, assessment, and assurance information"*. The project was initiated by Christofer Hoff, director of cloud and virtualization systems at Cisco Systems Inc. [3], and later it became an official project of the Cloud Security Alliance (CSA) during October, 2010.

There are three core aspects of CloudAudit framework. First core part of CloudAudit is that it is HTTP based. An authorized user can receive the service only using HTTP. This also means that the service can be received from anywhere via Internet. The second core part of CloudAudit is that it defines two types of namespaces for providing the service. These two namespaces are the *glossary namespace* and *service namespace*. The *glossary namespace* provides definitions and sometimes additional documentation for a service, while the *service namespace* provides assertions about the local or remote services. The response from *service namespace* must be a valid HTML in a human readable form.

The final core part of CloudAudit is the compliance packs. These compliance packs define the namespaces for the controls defined in CCM and for the corresponding control from a particular standard. Currently there are five compliance packs available. These are CloudAudit-COBIT_Compliance Pack, CloudAudit-HIPAA_Compliance Pack, CloudAudit-ISO 27002_Compliance Pack, CloudAudit-NIST800-53_Compliance Pack and CloudAudit-PCI_Compliance Pack. These compliance packs are made available to the public as Microsoft Office Excel files. A sample structure of CloudAudit-ISO27002_Compliance Pack is shown in Table 2.1.

2.7 OpenVAS

The Open Vulnerability Assessment System (OpenVAS) [27] is a comprehensive and powerful vulnerability scanning and management solution. It is a

Control Area	**Control ID**	**Control Specification**	**ISO 27002-2005 Compliance Number**	**Control Name Description**	**CSA**	**CloudAudit/A6**
Compliance - Audit Planning	CO-01	Audit plans, activities and operational action items focusing on data duplication, access, and data boundary limitations shall be designed to minimize the risk of business process disruption. Audit activities must be planned and agreed upon in advance by stakeholders.	ISO/IEC 27002-2005 15.3.1	Information systems audit controls	/.well-known/cloudaudit/org/cloudsecurityalliance/guidance/CO-01/	/.well-known/cloudaudit/org/iso/27002-2005/15-3-1/
Information Security - Diagnostic / Configuration Ports Access	IS-30	User access to diagnostic and configuration ports shall be restricted to authorized individuals and applications.	ISO/IEC 27002-2005 11.4.4	Remote diagnostic and configuration port protection	/.well-known/cloudaudit/org/cloudsecurityalliance/guidance/IS-30/	/.well-known/cloudaudit/org/iso/27002-2005/11-4-4/
Security Architecture - Clock Synchronization	SA-12	An external accurate, externally agreed upon, time source shall be used to synchronize the system clocks of all relevant information processing systems within the organization or explicitly defined security domain to facilitate tracing and reconstitution of activity timelines. Note: specific legal jurisdictions and orbital storage and relay platforms (US GPS & EU Galileo Satellite Network) may mandate a reference clock that differs in synchronization with the organizations domicile time reference, in this event the jurisdiction or platform is treated as an explicitly defined security domain.	ISO/IEC 27002-2005 10.10.6	Clock synchronization	/.well-known/cloudaudit/org/cloudsecurityalliance/guidance/SA-12/	/.well-known/cloudaudit/org/iso/27002-2005/10-10-6/
Security Architecture - Audit Logging / Intrusion Detection	SA-14	Audit logs recording privileged user access activities, authorized and unauthorized access attempts, system exceptions, and information security events shall be retained, complying with applicable policies and regulations. Audit logs shall be reviewed at least daily and file integrity (host) and network intrusion detection (IDS) tools implemented to help facilitate timely detection, investigation by root cause analysis and response to incidents. Physical and logical user access to audit logs shall be restricted to authorized personnel.	ISO/IEC 27002-2005 10.10.1 ISO/IEC 27002-2005 10.10.2 ISO/IEC 27002-2005 10.10.3 ISO/IEC 27002-2005 10.10.4 ISO/IEC 27002-2005 10.10.5 ISO/IEC 27002-2005 15.2.2	Audit logging, Monitoring system use, Protection of log information, Administrator and operator logs, Fault logging and Technical compliance checking	/.well-known/cloudaudit/org/cloudsecurityalliance/guidance/SA-14/	/.well-known/cloudaudit/org/iso/27002-2005/10-10-1/ /.well-known/cloudaudit/org/iso/27002-2005/10-10-2/ /.well-known/cloudaudit/org/iso/27002-2005/10-10-3/ /.well-known/cloudaudit/org/iso/27002-2005/10-10-4/ /.well-known/cloudaudit/org/iso/27002-2005/10-10-5/ /.well-known/cloudaudit/org/iso/27002-2005/15-2-2/

Table 2.1: CloudAudit-ISO27002 Compliance Pack Structure

framework of several tools and services distributed as free software. OpenVAS started its journey as a fork from another open source network vulnerability scanning tool named Nessus [23]. Nessus is now an proprietary solution and for the free development of the solution, OpenVAS is distributed under GNU General Public Licence. The original company still contributes for the development of OpenVAS.

OpenVAS is used to scan a target machine or many target machines at the same time to find vulnerabilities. These vulnerabilities are mostly network security related, however, not limited to only network security. By using OpenVAS it is possible to test local system security, policy checks, patch management, etc. for a system. After the scanning is complete, OpenVAS produces a report with the vulnerabilities found for a target system associated with the risk level. Risk is defined in three levels: High, Medium and Low. This gives an indication of which vulnerability is the most threatening at this moment. Using this report, a system or network administrator can make the necessary changes to the system to make it more secure by alleviating the vulnerabilities.

The latest released version of OpenVAS is 5.0 and it was released in May, 2012. Therefore, we have used the earlier version of OpenVAS (version 4.0.0 released in March, 2011) in building our automated security compliance tool. The architecture for OpenVAS 4 is given in Figure 2.4. OpenVAS 4 structure is divided into three layers. The first layer is the Client layer, which basically constitutes the graphical or command line interface that is used to access and use the services of OpenVAS. The second layer is the Service layer and consists the scanning daemon along with the managerial and administrative daemons for OpenVAS. The final layer is the Data layer and consist a pool of tens of thousands of Network Vulnerability Tests (NVTs) and a database to hold the configurations and results. In the following paragraphs, each of the components depicted in Figure 2.4 is described briefly.

OpenVAS CLI: The command line interface (CLI), called the OpenVAS CLI, is used to access, manage and use OpenVAS system using only commands. This client is a module consisting a tool called *omp*, which is developed and maintained by Greenbone Networks [17]. OpenVAS CLI allows the automation of batch processes for OpenVAS. This CLI is particularly important for us as we have used this interface to integrate OpenVAS in our automated security compliance tool.

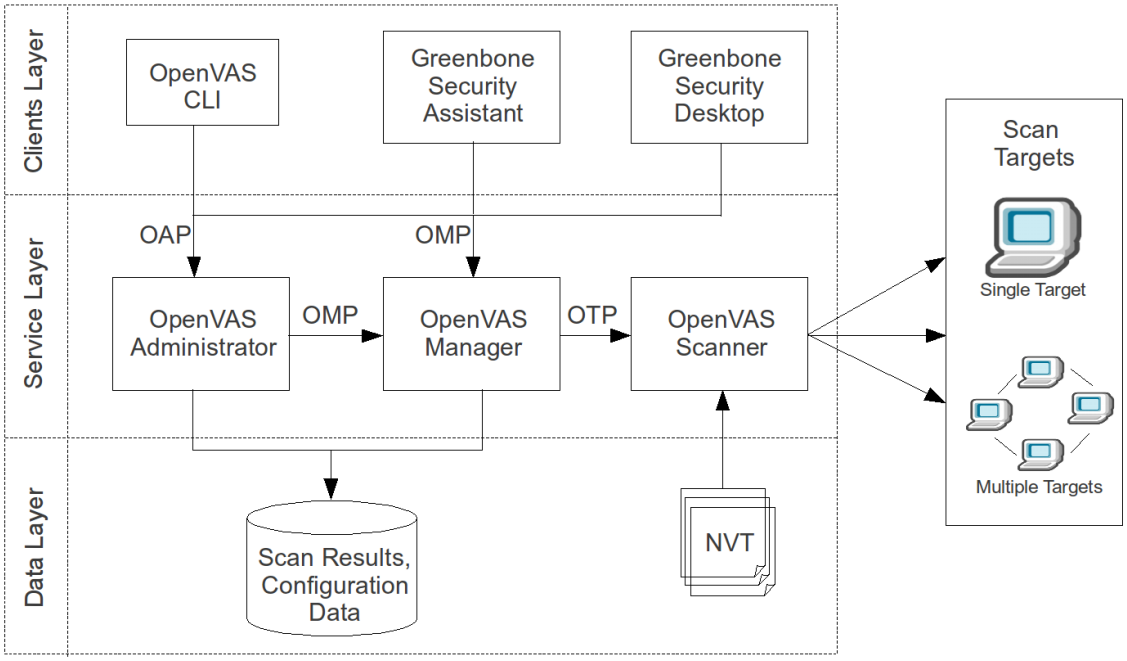


Figure 2.4: System Architecture for OpenVAS 4.0.0

Greenbone Security Assistant: Greenbone Security Assistant is the web based interface to access, manage and use the services of OpenVAS. This interface is again developed by Greenbone and has gained the most popularity among the user community.

Greenbone Security Desktop: This is the desktop base interface for OpenVAS developed using Qt framework. This desktop based control center for OpenVAS is also developed and maintained by Greenbone.

OpenVAS Manager: At the core of OpenVAS system lies the OpenVAS manager. All the intelligence required to build a full-fledged vulnerability management solution rather than just building a simple vulnerability scanner are put into the OpenVAS manager. This module controls the OpenVAS Scanner and the database. In order to control the scanner, it uses OpenVAS Transfer Protocol (OTP). And for other modules that want to communicate with the OpenVAS manager, it offers OpenVAS Management Protocol (OMP) which is an XML based stateless protocol. All the clients communicates to the OpenVAS system using this OMP protocol.

OpenVAS Administrator: This module offers a command line tool or a full fledged service for user management and NVT feed management for the OpenVAS system. It offers the OpenVAS Administration Protocol (OAP) to carry out the management related tasks. Users only with the *admin* role can use this OAP functionality.

OpenVAS Scanner: OpenVAS scanner is the core module that does the actual vulnerability tests. In order to do so, it executes the NVTs and returns the results. This module offers the OpenVAS Transfer Protocol (OTP) for carrying out the scanning-related task and using this protocol. The OpenVAS Manager controls this module.

Database: This is a SQLite [33] based database for OpenVAS system to hold all the configuration related data. It also holds all the scan results that have been done so far.

NVTs: Network Vulnerability Tests or NVTs are the scripts that enable the OpenVAS to detect the vulnerabilities. All of these NVTs are written in the Nessus Attack Script Language (NASL) [22]. Since every now and then new type of vulnerabilities are exposed, the OpenVAS has to cope with these new threats by being able to detect them. In order to do so, OpenVAS introduces new NVTs or modifies old ones in its pool of NVTs so that they can detect the new vulnerabilities. Therefore, the NVT pool is updated almost daily basis and there are currently around twenty five thousands of NTVs in the OpenVAS system.

For our automated security compliance tool, the most important parts of OpenVAS are the CLI and the NVTs. By using the "omp" tool for CLI we can access and use the OpenVAS service and therefore, can integrate OpenVAS in our solution. NVTs are essential for our solution as it is the NVTs that detect the vulnerabilities and generate the report for them. Therefore, if someone is looking for a new type of vulnerability or new type of report, then a new NVT must be created for that purpose.

2.8 Related Research in this Field

Despite the importance of automated auditing for security compliance in the cloud computing arena, we discovered that little research have been done in

this field. Nonetheless, quite a lot of research has been done in the related field of auditing in the cloud computing arena. For example, Prafullchandra et al. [57] and Rasheed [58] has identified the challenges for auditing against the PCI DSS standard in the cloud environment. Also, a theoretical architecture to achieve PCI compliance has been presented [57].

Soren Bleikertz et al. [39][40] has shown an approach to assess the network security configuration in the IaaS cloud platforms. This approach uses reachability and attack graphs in order to determine network related vulnerabilities in the system. Their approach is automated in the sense that the configuration data retrieval and the decision making is done without human intervention. The scope of their work is limited to only network security assessment.

Other related research has been published in the area of cloud privacy and risk assessment. Tancock et al. [62] have demonstrated a Privacy Impact Assessment Tool that can be used in order to assess the privacy law compliance. Furthermore, Kaliski et al. [50] has proposed to deliver assessment-as-a-service in the cloud computing environment since the environment is rather different than regular computing environment.

A particular related area of interest is cloud monitoring. Hasselmeyer and D'Heureuse [47] have identified the domain that separates the cloud computing environment from a regular computing environment from the perspective of monitoring. We believe that the results of such research are also valuable to achieve automated security compliance in the cloud environment as it also provides a guideline for where to look for the information. Chaves et al. [45] has indicated several challenges for cloud monitoring, some of which are also in line with our work. The reason we mention cloud monitoring here is that, since the cloud monitoring is already gathering data about the cloud system, one approach for automating security compliance tool could be to use these data. Therefore, cloud monitoring can be used as a part of the automated security compliance tool, or information from an existing cloud monitoring system can be reused to build an automated security compliance tool.

Chapter 3

System Architecture and Design Patterns

In this chapter we present the design of the developed solution in detail. Different components are put together to build the architecture. These include OpenStack, CloudAudit, Cloud Control Matrix, OpenVAS and ISO 27002. We start by presenting a high level description of our approach that we have taken to build this automated security compliance tool. Later we describe the system architecture in detail and finally we describe different approaches for collecting the data that are essential for this tool.

3.1 High Level Architecture

Figure 3.1 depicts the high level overview of the solution architecture. The first step to verify the compliance status of a system against a control is to collect the necessary data for that purpose. Therefore, as shown in Figure 3.1, we start by collecting data from the cloud in an engine which we call the data collection engine. Then we feed this data to the verification engine, which takes the decision whether the system is compliant or not. Finally, the verification engine exposes this compliance status to the client using the CloudAudit API.

From the Figure 3.1 we identify three core parts of the automated security compliance tool. These are:

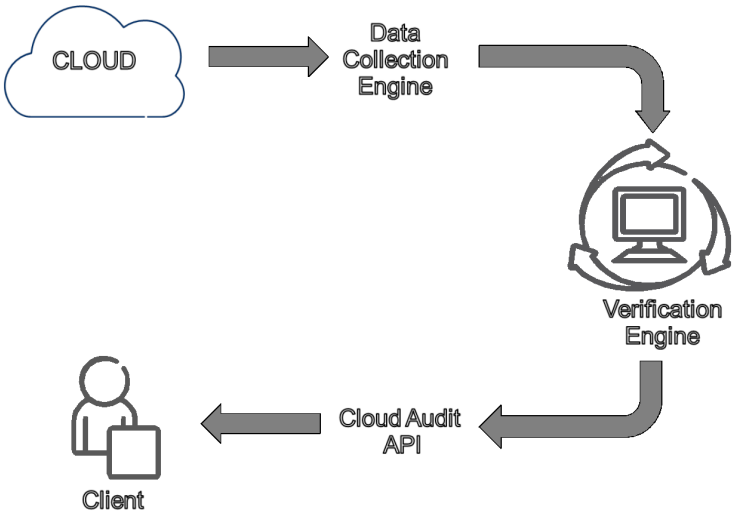


Figure 3.1: High Level Architecture for an Automated Security Compliance Tool

Data Collection Engine: The first essential part to build an automated security compliance tool is to collect data effectively and efficiently. By being effective we mean being able to collect all the necessary pieces of data and by being efficient we mean using resources efficiently while fetching the data. There can be many possible ways to collect data from a cloud and some of these are described in section 3.3.

Verification Engine: This is the core part of the automated security compliance tool. All the necessary intelligence for decision making and the knowledge about the controls from the standards lies in this verification engine. Using this intelligence and based on the data received from the data collection engine, it makes the decision whether the cloud system is compliant or not.

CloudAudit API: This is the user interface to the client. There can be many different ways to display or visualize this compliance related information. However, we decided to use the CloudAudit API to display this information as the CloudAudit framework is the IETF draft, designed solely for this purpose by CSA.

In the next section, we will look at the complete system architecture that we have developed for the OpenStack cloud computing platform.

3.2 System Architecture

Figure 3.2 depicts the system architecture that we have developed to build an automated security compliance tool targeting the OpenStack cloud platform. The main components of the system are listed in the following -

1. Users: Admin or client of the OpenStack cloud system.
2. Dashboard: OpenStack DJango web based dashboard for information display.
3. Keystone: OpenStack identity service for authentication and authorization.
4. Cloud Audit Framework: This includes cloud audit API server and the evidence engine. The API server receives the request from the user and calls the appropriate method in the evidence engine to handle the request. When this API server receives the response from the evidence engine, it forwards that response to the user. The CloudAudit evidence engine combines the functionality of the data collection engine and the verification engine as depicted in Figure 3.1.
5. CloudAudit and CCM: Defines the CloudAudit API format and the mapping of the cloud security controls to the industry accepted standards.
6. Nova API server: The OpenStack cloud computing fabric controller providing the API services.
7. OpenVAS: A vulnerability assessment tool used to assess the vulnerabilities of the OpenStack system.

Having defining the components of the system, we can now take a look at the system depicted in Figure 3.2. The user first logs in into the OpenStack dashboard using their credentials. At this stage, Keystone is used as an identity service. In the dashboard, the user is given a list of available compliance checks that are allowed to be verified by them. Once the user selects one of these compliance checks to be verified, the control flow is transferred to the CloudAudit API server. This eventually calls the method residing in the CloudAudit evidence engine that is responsible for the particular compliance check. At this point, the evidence engine method uses one of four different approaches to gather the necessary information. Depending on the chosen way, control may be transferred to the Nova API server or to a third party vulnerability assessment tool or maybe the evidence engine can only search

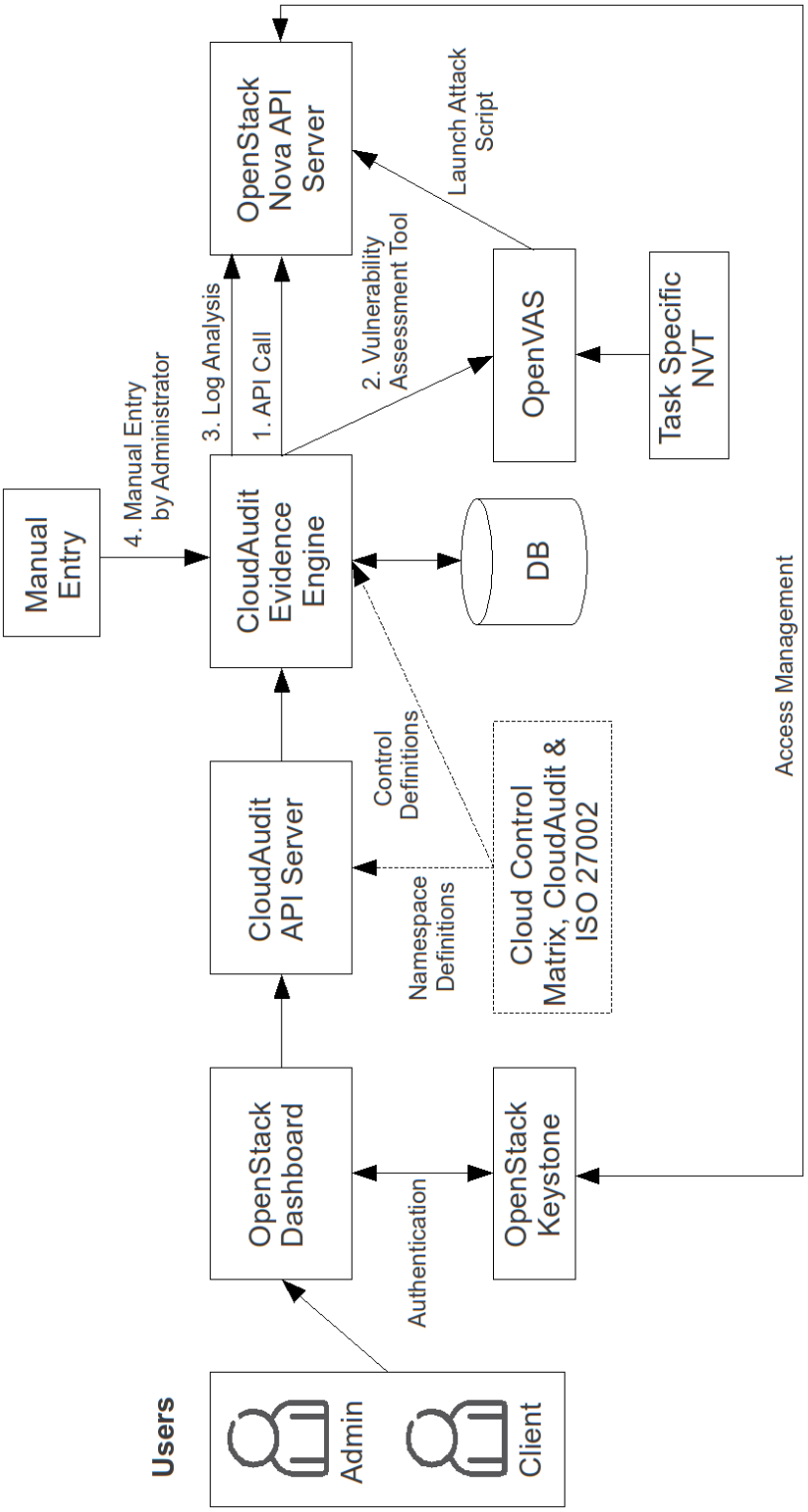


Figure 3.2: System Architecture for Automated Security Compliance Tool in OpenStack Cloud Platform

its database for manual entries. After information retrieval, the evidence engine method processes that information for the particular compliance check and sends back the results to the CloudAudit API server, which in turn returns the results to the user.

One thing to be stated here is that the dashboard can directly call the Nova API, which it normally does. However, in our compliance check architecture we have omitted that since we are not calling the Nova APIs directly; rather we are calling the newly exposed Nova APIs through the CloudAudit framework.

The CloudAudit evidence engine in Figure 3.2 corresponds to both the data collection engine and the verification engine shown in Figure 3.1. Therefore, all the necessary logic and intelligence required for data collection and verification is combined into the CloudAudit evidence engine. For the user interface, the CloudAudit API does the same in both the Figures 3.1 and 3.2.

3.3 Design Patterns

While developing the architecture, we have found four possible ways to collect data from the cloud to carry out the automatic security compliance check. Two of these four possible ways are novel and we have implemented these two ways as a proof-of-concept. The four approaches are described in the following subsections.

3.3.1 Application Programming Interface (API)

The easiest way to obtain any information from a system is that the system itself provides this information. This is our first approach to obtain necessary information from a cloud system when it is required by the CloudAudit evidence engine to verify against the security standards. In order to do so, we have used the API mechanism in the cloud system. We propose a new set of API in the cloud platform whose sole purpose will be providing necessary information that can be used by the CloudAudit framework.

For OpenStack we have exposed a new REST API in the OSAPI pool. When the CloudAudit evidence engine calls this API, the Nova API server executes the code and returns the necessary information to the callee. CloudAudit

framework uses this information to make the decision whether the cloud system is compliant with the specific control from the standard or not.

One important thing to notice here that the Nova APIs are access restricted. Therefore, in order for the CloudAudit evidence engine to call a Nova API, it must have the authorization information. However, since a user is already logged in to the OpenStack dashboard using his credentials, these user information can be used to call these APIs, which is the way the dashboard normally works.

The benefit of this approach is that any sort of information can be easily and reliably exposed to the automated security compliance tool as the information gathering is done in the system itself. However, there are two issues with this approach. The first issue is that the cloud system needs to be modified in order to expose the information through an API. The second issue is that the automated security compliance tool has to know which API to call to get this information. Therefore, tight coupling between the CloudAudit framework and the OpenStack Nova is required for this approach to work successfully.

3.3.2 Vulnerability Assessment Tool

The second approach that we have introduced for the automated security compliance check is the use of a vulnerability assessment tool. This vulnerability assessment tool gathers information about the system by running some scripts and generates the reports. This report is then analyzed by the CloudAudit framework and the verdict is then returned to the user.

For this approach, we have selected an open source vulnerability assessment tool called OpenVAS. OpenVAS can be used for many different kinds of vulnerability assessment such as local security checks, firewall checks, port scanning, backdoors and so on. OpenVAS is flexible enough in the sense that it can be used from simple tasks such as checking for open ports in a system to much harder tasks like verifying the company policies.

The main benefit of this approach is that the automated security compliance tool can work independently without any integration with the cloud system. It can collect the data externally without any modifications required to the cloud system. In some cases it can create an SSH tunnel to login into the cloud system, however, which still does not require any system modifications. The great benefits of this approach come with the drawback that it is much harder to collect the information externally. Also, it may take longer time

to run the scanning and then generating the report which may be crucial in an on-demand system.

3.3.3 Log Analysis

The third approach to gather the necessary information is by analyzing the logs of a system. In order to do this, the CloudAudit evidence engine has to login to the system by using SSH credentials. After that, it can analyze the log files of interest and can extract the required information on compliance. One important thing here is that the CloudAudit evidence engine must be given the SSH credentials to log in to the target system if this method is used.

In this thesis project, we do not use this approach as Piston Cloud Computing [56] has already developed a few automated control checks for OpenStack cloud platform using this approach. However, we depict the architecture of this approach in Figure 3.3.

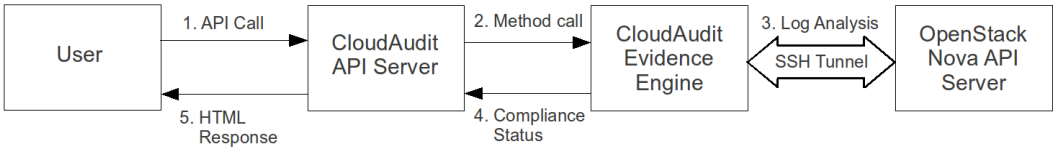


Figure 3.3: Control Flow for Automated Security Compliance Tool Using Log Analysis Mechanism in OpenStack Cloud Platform

The benefit of this approach is again that no system modification is required for the cloud infrastructure. The demerit of this approach is that the log files in different cloud systems are different and the way of representing the log information may be different. Therefore, the automated security compliance tool that uses this approach must be tightly coupled with the cloud system in order to be able to collect the information successfully.

3.3.4 Manual Entry

The final approach to retrieve compliance related information is by using manual entries provided by the cloud administrator. This can be done by

providing a user interface in the CloudAudit framework where an administrator will be given options to input necessary data that is required by the CloudAudit framework for compliance checks. These data will be kept inside a database in the CloudAudit framework and will be assessed against the standards by the framework when it is demanded by a user.

The reason behind introducing this approach is that there is some information required to verify some of the security controls that cannot be retrieved automatically from the system. For example, the client may want to verify that the cloud vendor is backing up the data as per the company policy. To verify this control the automated security compliance tool first has to know the policy for data backup and then it can verify from the system whether the data has been backed up or not as per company policy. The policy can be written into the automated security compliance tool by using this approach. There are other ways to provide this manual entry by a cloud vendor such as by XML files, text files and html files. However, in that case the the policy files will be cloud vendor specific and the automated security compliance tool needs to be integrated tightly with the cloud system to understand these files.

The main benefit of this approach is that any type of information, which might not be possible to generate automatically otherwise, can be provided to the CloudAudit evidence engine. Again this benefit comes with a drawback - as the cloud administrator provides the input manually, the trust issue remains between the client and the cloud vendor.

We have implemented the API and the vulnerability scanning data collection mechanisms in our prototype. Table 3.1 summarizes the advantages and disadvantages of each approach presented here.

Mechanism	**Advantage**	**Disadvantage**
API	1. Easier to extract information from the system itself. 2. Access control for the information may become easier.	1. System modification required on the cloud system side as new APIs need to be exposed. 2. The automated security compliance tool need to be tightly coupled with the cloud system as it must know which API to call.
Vulnerability Scanning	1. Can extract the data externally and hence no system modification is required in the cloud side. 2. Can collect data from an outside attackers point of view	1. It is difficult to collect data externally. 2. Using a 3rd-party scanning tool may require more time to collect data.
Log Analysis	1. No system modification is required on the cloud side.	1. Log files are different in different systems. Hence, tight coupling between the automated security compliance tool and the cloud system is required.
Manual Entry	1. Any type of information can be provided using this mechanism.	1. An internal database need to be maintained in the automated security compliance tool for this mechanism to work. 2. Trust Issue.

Table 3.1: Comparisons among different data collection mechanisms

Chapter 4

Implementation

This chapter describes detail information about the implementation of the prototype based on the architecture depicted in chapter 3. It presents the methodology that we have used to develop our prototype, the system components that we have used and how each module is developed and adapted.

4.1 Methodology

We followed the Evolutionary Prototyping [44] software development methodology for our prototype development. Evolutionary prototyping is an incremental software development lifecycle model where new features can be added at any time and or the software can be modified in response to end-user/ customer feedback. The most significant feature of the evolutionary prototyping model is that the final system concept is developed as the development of the prototype is moving forward. The prototyping procedure starts by designing the most salient features first. Then the prototyping process moves on with the more detailed design as the development is going on and with the feedback from that development.

For our research work, in order to build a prototype automated security compliance tool, we valued this on-the-go concept-development-feature of the Evolutionary prototyping much, as it suited our work flow. Also, the incremental software development lifecycle feature has allowed us to gradually add new features in our prototype system.

4.2 Reused System Components

In order to avoid the reinvention of the wheel, we have used several open source components and software to build our solution. The list of these components are in the following -

1. The first in the list is the OpenStack cloud computing platform. We have modified the OpenStack's Nova [35] project in order to implement our solution.
2. The second open source software that we have used is the Open Vulnerability Assessment System (OpenVAS) to assess the vulnerability in the target cloud system.
3. Thirdly, we have used Piston Cloud Computing's [29] CloudAudit framework [56] as the framework on top of which we developed our solution.
4. Finally, we have used and modified the OpenStack Horizon [18] project to play the role of the user interface.

4.3 Implemented Security Controls

During our project we have analyzed several standards such as ISO 27001, ISO 27002, NIST, PCI DSS and Cobit. After comparing these with each other, we have selected the ISO 27002 v2005 to be the standard that we are going to use for our automated security compliance tool. The reason behind selecting this standard is that the controls are described in a more elaborate way compared to other standards. There are also some implementation guidelines provided for each security control in ISO 27002 v2005, which are missing in the other standards that we have analyzed. Taking these advantages into consideration, we have selected two controls from ISO 27002 v2005, which are also present in Cloud Control Matrix (CCM), to be implemented in our prototype solution. These two security controls are quoted from the ISO 27002 standard in the following subsections.

One thing to clarify here is that CCM does not directly give the ISO 27002 control number; rather it gives the ISO 27001 control number. However, ISO 27002 is based on ISO 27001 and we can find the same controls as in ISO 27001 and 27002. Although CCM does not give us the ISO 27002 control number for a cloud compliance control ID, the CloudAudit compliance

pack gives an ISO 27002 control number for a cloud compliance control ID. Therefore, we are able to map ISO 27002 against CCM control IDs.

4.3.1 Clock Synchronization

This control is defined in ISO 27002 [21] section 10.10.6. The details about this control are as follows:

> ***Control:*** *The clocks of all relevant information processing systems within an organization or security domain should be synchronized with an agreed accurate time source.*
> ***Implementation guidance:*** *Where a computer or communications device has the capability to operate a real-time clock, this clock should be set to an agreed standard, e.g. Coordinated Universal Time (UTC) or local standard time. As some clocks are known to drift with time, there should be a procedure that checks for and corrects any significant variation. The correct interpretation of the date/time format is important to ensure that the timestamp reflects the real date/time. Local specifics (e.g. daylight savings) should be taken into account.*
> ***Other information:*** *The correct setting of computer clocks is important to ensure the accuracy of audit logs, which may be required for investigations or as evidence in legal or disciplinary cases. Inaccurate audit logs may hinder such investigations and damage the credibility of such evidence. A clock linked to a radio time broadcast from a national atomic clock can be used as the master clock for logging systems. A network time protocol can be used to keep all of the servers in synchronisation with the master clock.*

The importance of this security control is already mentioned in the "Other information" paragraph, stated above. In addition to that, it is also a very important issue from the perspective of a cloud user. This is due to the fact that some algorithms may not work reliably if the cloud system is not synchronized with UTC. Also, if the client of the cloud is a financial company like banks, then it is very important for them to have a precise timestamp for all their transactions. Otherwise, there may be severe consequences for this sort of companies if they fail to maintain a precise timestamp.

This control is mapped to cloud security compliance in the CloudAudit ISO 27002 compliance pack with control ID SA-12. This control ID SA-12 is same

as the CCM SA-12 and, in fact, these are the same security controls.

To verify this security control, we have used the following procedure -

1. Obtain the system time from a target machine for which we want to validate this control.
2. Obtain the UTC time from a reliable NTP server at the same time instant.
3. Decide whether the target system is synchronized with the UTC time or not.

Here, it is important to obtain the times form the target system and the Network Time Protocol (NTP) server at the exact same moment of time. Otherwise, the decision may become erroneous. For example, even if the target system is synchronized with the UTC time but the measurements from the two systems are taken at different time moments, then the verdict will be that the system is not synchronized with the UTC time. It is also possible that the target system is off by few milliseconds from the UTC time and the measurements taken from the target machine and the NTP server are also a few milliseconds apart. In this case, the verdict may result in saying that the target system is synchronized with the UTC time although it is not.

4.3.2 Remote Administrative & Diagnostic Port Protection

This security control is defined in ISO 27002 [21] section 11.4.4. The details about this control are as follows:

> ***Control:*** *Physical and logical access to diagnostic and configuration ports should be controlled.*

> ***Implementation guidance:*** *Potential controls for the access to diagnostic and configuration ports include the use of a key lock and supporting procedures to control physical access to the port. An example for such a supporting procedure is to ensure that diagnostic and configuration ports are only accessible by arrangement between the manager of the computer service and the hardware/software support personnel requiring access. Ports, services, and similar facilities installed on a computer or network facility, which are not specifically required for business functionality, should be disabled or removed.*
>
> ***Other information:*** *Many computer systems, network systems, and communication systems are installed with a remote diagnostic or configuration facility for use by maintenance engineers. If unprotected, these diagnostic ports provide a means of unauthorized access.*

Implementation of this control makes sure that no unauthorized access is possible through the open ports of a system. There are a lot in this control and not all can be verified for compliance automatically since this control also has some physical or manual process included in it. However, the line saying that "all the unnecessary ports not required for business functionality, should be disabled or removed" can be verified automatically. This is actually what we have done in our implementation: check for unnecessary open ports in the system. It is always considered as a security risk if a system has unused open ports.

This control is mapped to cloud security compliance in CloudAudit ISO 27002 compliance pack with control ID IS-30. This control ID IS-30 is same as in the CCM IS-30 and, in fact, these are the same controls.

4.4 Exposing New OSAPI

Referring to chapter 3, our first approach of collecting information for compliance check is using APIs. The benefit of this approach is that the API is running under the control of the system from which the information needs to be extracted. Therefore, it is easier for the system to produce that information reliably as compared to an external system which wants to gather the same information.

We have chosen this approach to verify the ISO 27002 "Clock synchroniza-

tion" control. As mentioned in section 4.3.1, we need to obtain the system time for this control to be verified. We can obtain this information by exposing a new API from the OpenStack which retrieves and sends the system time. We have chosen this approach to retrieve system time as this API approach suites better for this sort of system information retrieval and, if we had to use other approaches mentioned in chapter 3, then the process of getting the system time reliably would have been a much more difficult task.

In order to expose a new API, we have modified the OpenStack Nova [35] compute cloud fabric controller in such a way that it supports a new API in its OSAPI pool. OpenStack Nova is implemented in modular architecture using Python [30] Paste Deployment [28]. Also, most of the APIs are RESTful [46] APIs. Therefore, maintaining the Nova API architecture, we also designed and implemented a RESTful API and exposed it through Nova OSAPI. The steps that we have taken to expose a new API are given in the followings:

1. The first step is to define a URL for the new API. Since we are interested in getting the current system time where the Nova server is running, we defined the URL to be as "http://server_name:8774/v1.1/x/current_time". In this URL, "server_name" refers to the name of the server that hosts the Nova API server. 8774 is the OpenStack port defined for the Nova APIs. V1.1 is the API version. "x" is the user id and finally "current_time" is the name that invokes the Nova API for getting the current system time.

2. To map this URL to a controller and an action, OpenStack uses the Routes [31] package from the Python implementation. After receiving a URL, control is transferred to the controller class for that URL and there the URL is mapped to an action method.

3. The action method does the actual work, which is to retrieve the system time. These action method also takes parameters that were extracted from the URL. Finally, the action method returns the result in a dictionary data structure.

4. The result dictionary is then serialized to XML or JSON by the Web Service Gateway Interface (WSGI) [38] Controller and returned to the client who invoked the API.

In our automated security compliance tool, this API is invoked by the CloudAudit engine by sending an HTTP request to the above mentioned URL. In response, the CloudAudit evidence engine receives the current system time

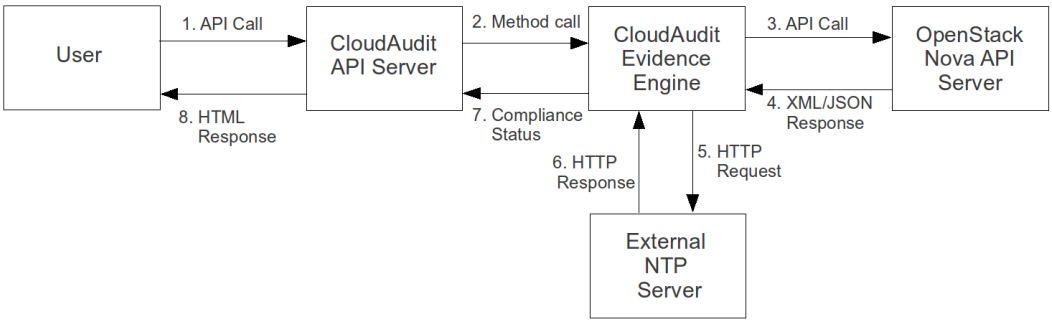


Figure 4.1: Control Flow for Automated Security Compliance Tool Using the API Mechanism

in XML or JSON format and processes it for further action. The overall architecture using this API approach is depicted in Figure 4.1.

4.5 Adding NVT in OpenVAS

The second approach to gather information is by using a vulnerability scanner. In our prototype automated security compliance tool we have used the open-source OpenVAS tool as a vulnerability scanner. Using this approach we can gather information about a target system externally.

We have used this approach to verify the ISO 27002 "Remote Administrative and Diagnostic Port Protection" control. As mentioned in section 4.3.2, we are only going to do a partial verification of this control by only scanning for open ports. This could have been done in many ways. The first three approach mentioned in chapter 3 can all be used for this purpose. Since the API approach and the Log analysis approach gather information from the system internally, we have decided that it is better to do this port scanning externally using the Vulnerability Scanner approach. This design decision was influenced by the fact that most of the attacks we can expect are from external systems since the control is about remote access.

In order to use the OpenVAS tool for our purpose, we had to add a new NVT into its scanning engine. The steps that we have taken to make it work in conjunction with our prototype solution are listed in the following -

1. We have written a new NVT for port scanning using NASL [22] scripting language. In the new NVT, we have put the logic for which ports should not be open for OpenStack. If the NVT finds any ports open

other than the operational ports required by OpenStack, it generates a report with the list of unnecessary open ports in the target system.

2. Rebuild the OpenVAS system to incorporate the newly added NVT.
3. The next step is to create a scan configuration for OpenVAS that includes only this NVT. This scan configuration tells the OpenVAS scanner which NVTs to run when the scanning starts.
4. Using the above scan configuration, we had to create a new task for OpenVAS. At this step, we also specify the target system against which the NVTs will run.
5. Finally, we had to write a shell script [37] to start, monitor and generate a report for the task we have created in OpenVAS.

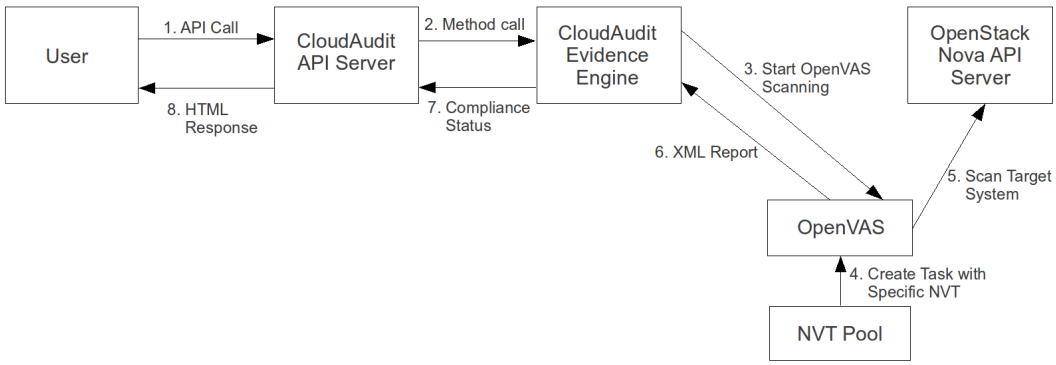


Figure 4.2: Control Flow for Automated Security Compliance Tool Using Vulnerability Scanning Mechanism

The shell script is launched by CloudAudit evidence engine when the "Remote Administrative & Diagnostic Port Protection" API is invoked by a user. When the task is finished, a report is generated in XML format which is then used by the CloudAudit evidence engine for further processing and to verify the control. The overall architecture using this OpenVAS vulnerability scanner approach is depicted in Figure 4.2.

4.6 Building Cloud Audit API/Evidence Engine

Our automated security compliance tool is built around the CloudAudit framework developed by the Piston Cloud Computing. The Piston Cloud Computing's CloudAudit framework only has support for four APIs which use log based analysis mechanism (Section 3.3.3) for compliance verification. This framework uses the NIST standard for the compliance check. We have enhanced the framework by adding our two new approaches: the API mechanism and the vulnerability scanner mechanism. Also, we have modified the code base control hierarchy to support the ISO 27002 standard.

The CloudAudit framework is implemented using Python Paste Deployment and all the APIs are RESTful. In order to build our solution using CloudAudit framework, we have taken following steps:

1. The first step is to register our APIs with the framework. This also defines where the control will be transferred upon receiving a request.
2. Next step is to define the controller class that receives and handles the request. For each control from the standard, one controller class class is required to handle the request. In order to handle the request, it eventually makes a call to a method in the CloudAudit evidence engine.
3. In the CloudAudit evidence engine, we have to define appropriate methods that gather the actual evidence for compliance, verify it against the control from the standard and finally return the result. For each different security control, there is one method in this evidence engine framework.
4. Upon receiving the result from the evidence engine, the controller class formats the result as HTML and returns the result to the CloudAudit API server from where it is sent back to the client.

In order to build the CloudAudit evidence engine, we had to take into account some implementation considerations for decision making. Also, to provide assurance of that decision to the user, we had to consider what data should be provided in the user interface. These issues are discussed in the following sub-sections.

4.6.1 Implementation Considerations

As mentioned in section 2.1.2, one of the biggest challenges in automating the security compliance checks is that there is no or minimal level of guidance to implement the controls from the standards. Therefore, to implement these controls we had to make some decisions on our own. We refer to these decisions as implementation considerations. For example, for the "Clock Synchronization" control, described in ISO 27002, there is no information about how much drift from the external NTP server can be allowed to consider the system to be still synchronized. However, this is important for us, as we have to make a decision based on the responses received form the cloud system and from the NTP server. Since we cannot retrieve the responses from the two systems at the exact same moment, we have to allow some drift in the time between the two systems. Therefore, for our implementation we have considered 1000 milliseconds of drift between the two systems to be still recognized as time synchronized.

4.6.2 Providing Assurance

Providing assurance of the decision made by the automated security compliance tool is another big challenge mentioned in section 2.1.2. As we are taking into account some implementation considerations on our own to implement the controls, a user of the tool needs to be given enough information about how the decisions have been made or how much accuracy is there. To facilitate this assurance for the client, we have designed our output in such a way that the client gets enough information about the implementation considerations along with the data that enabled the decision making.

The output from the CloudAudit evidence engine is generated as HTML and we have provided the following information in this HTML -

1. **Compliance Status:** Stating whether the system is compliant with this control or not.
2. **Compliance Details:** Additional information or data that enabled the evidence engine to make the decision.
3. **Control Description:** Provides the description of the control stating what this control is about.
4. **Implementation Considerations:** Provides information regarding the implementation considerations that have been taken in order to

implement the control.

5. **Verification Method:** This states which approach (API, vulnerability scanner, etc.) have been employed to collect data and how the collected data is used to implement the security control.

All of the above fields provide information to facilitate the assurance on the decision made by the automated tool. This information may also provide an indication of what needs to be corrected if the system is not compliant with the control. Some sample HTML responses can be found in appendix A.

4.7 OpenStack Dashboard Integration

The final part of the prototype implementation was to build a user interface for the auditing purpose. The OpenStack Horizon dahsboard project is already there serving as a graphical user interface (GUI) for OpenStack. Using this web-based user interface a user can create, launch and stop new virtual machine instances or monitor the status of the instances and so on. Since we are implementing the compliance checks for OpenStack, this dashboard is the most suitable place to display the information so that the user gets all the necessary information in a single place rather than going to several places for it.

The Horizon dashboard project is built using the Django [12] web framework. Django is a high level web framework that facilitates rapid development of web applications in a clean and modular way. When the user goes to the Horizon dashboard, they are asked to provide the credentials for logging in to the system. After being logged in, the user is redirected to a page where all the managerial links are given depending on the user's role. We have modified this managerial page to display links to the available compliance check APIs that we have developed. Thus, when the user clicks on one of these links, effectively the CloudAudit APIs gets invoked and the results are returned after the CloudAudit evidence engine has generated the compliance check result.

Chapter 5

Evaluation

This chapter discusses the post-implementation evaluation of the developed framework and the prototype. Since our aim was to develop a proof-of-concept prototype, our discussion here focuses mostly on the architectural side rather than evaluating the performance of the tool.

5.1 Security Perspective

Several security aspects are identified while evaluating our solution. This includes both the advantages and possible risks of the tool in terms of security. These are discussed in the following subsections.

5.1.1 The CIA Triad

One of the core models of computer security is the CIA triad. The three core components of the CIA triad are confidentiality, integrity and availability. Each of these components represent a fundamental security objective that helps identify problems and weaknesses in the information security. In the following paragraphs, we use this CIA model to evaluate our solution.

Confidentiality: Data or information confidentiality refers to the ability to hide information from those who are unauthorised to view it. As mentioned in section 2.1.2, one of the challenges is to extract and deliver the data securely to the automated security compliance tool. This is to make sure that no

unauthorized user or an outside attacker receives the confidential data. In order to achieve this data or information confidentiality, we have developed the framework in such a way that only authorized user of the cloud can access these data. As shown in Figure 3.2, a user has to login to the OpenStack dashboard using the user credentials. This will be verified by the OpenStack identity service Keystone. After successful authentication, an authorized user will be able to access the compliance-related information. In this way, only the authorized cloud users can access the confidential information.

This data confidentiality can be taken down into more granular level. Since the compliance related information is displayed in the OpenStack dashboard, it is possible to restrict the information display based on the user role. For example, while an administrator can have all the information regarding the compliance status, a general user may be allowed to access only partial information about the compliance.

Integrity: Integrity refers to the ability of protecting data from unauthorized modification. In other words, data should not be tampered in any unauthorized way once it has been submitted or requested by an authorized user. In our solution, the API mechanism is particularly susceptible to this security objective. As the API mechanism uses the HTTP to transfer data between the automated security compliance tool and the CSP, several attacks like man-in-the-middle (MITM) and phishing are possible to violate the integrity of the data. To ensure the integrity of the data, HTTPS (Secure HTTP) can be used between the automated security compliance tool and the CSP. Another possible option to protect the integrity of the data is to install the automated security compliance tool inside the CSPs' network. Therefore, all the communication will be using the CSPs' internal network, which is generally well protected from the outside world. In this case, no MITM or phishing attack is possible from outside the CSPs' network. However, it is still possible to launch these attacks from the internal network of the CSP.

Availability: Availability refers to the ability of delivering the information when it is needed by an authorized user. This security objective refers to the system being available despite power outage, network breakdown, hardware failures, etc. One particular type of attack, the denial-of-service (DoS), can harm the availability of our solution. This is due to that fact that all of the data collection mechanisms use the network to gather information and also the compliance status is exposed using the REST API. Therefore, any DoS or

Distributed DoS (DDoS) attack can cause disruption in getting the service. In order to prevent our automated security compliance tool from such a DoS or DDoS attack, the tool can be installed inside the CSP network and it can use the internal network to collect and send data to the CSP. The internal network of the CSP is well protected using the firewall and other security measures. Hence, the service of the automated security compliance tool will be available if there are no other failures like power outage, hardware failure or upgrade issues.

5.1.2 Possible Risks

Since all the decisions are made automatically based on the collected data, any flaws or incorrect data may cause the automated security compliance tool to produce erroneous results. We have identified two possible sources that can intentionally or unintentionally provide misleading information and thus produce faulty results. These two possible sources are as follows-

1. Cloud Vendor: This situation may occur when a cloud vendor intentionally provides incorrect data to the automated security compliance tool in order to get the results they wish. Or it could also be the case where the provided information was unintentionally incorrect. There are again two possible situations when an evil cloud vendor can take advantage of this security flaw. The first situation occurs when the API mechanism is used to collect data from the cloud. Since the cloud vendor is in charge of developing and exposing the compliance-related API, the vendor can decide how and what information to expose through this API. Hence, the vendor can provide incorrect information using this API mechanism.

The other situation occurs when the automated security compliance tool asks for the manual entry by the cloud administrator. In this way, the cloud vendor can again provide misleading information to the tool and thus succeed to produce faulty results by the automated security compliance tool.

The possible countermeasures to prevent incorrect information to be passed to the automated security compliance tool is to use a 3rd-party auditor. However, this requires human intervention and it is a manual process. Although it is possible for the cloud vendors to provide this sort of misleading or intentionally incorrect information, one strong motivation for them not to do this is the fear of loosing the trust of the clients. Any such scandal

will bring down the cloud vendor's reputation which will be harmful for its business.

2. Third Party Service Provider: The second possible source of incorrect information is the 3rd party services (if any) used by the automated security compliance tool. For example, to verify the "Clock Synchronization" control, we have used an external NTP server to retrieve the current time. In this case, we have trusted the external 3rd-party NTP server to provide the correct time information. However, if this external NTP server fails to provide the correct time, the automated security compliance tool may produce faulty results.

One possible countermeasure to protect the system from incorrect information provided by 3rd-party service providers is not to use any 3rd-party service at all. However, this may lead to building all of these 3rd party services in-house which can be very costly (if there exists no alternate way without taking this 3rd party service). The other possible option is to verify that the 3rd party services are working perfectly before any decision is made.

5.2 Latest Related Work

During the thesis, we have evaluated two of the related works that aim to provide security compliance information automatically. We briefly present these two works in the following sections and also outline the major differences with our work.

5.2.1 Cloud*e*Assurance

Cloud*e*Assurance [9] is a product of eFortresses company [13], which delivers products in the area of security and compliance. Cloud*e*Assurance is a platform where a cloud vendor can assess their security, governance, risks and compliance. The first version of this product was launched in April, 2012. The product works by presenting a web-based interface to a cloud vendor where the vendor has to provide a self assessment to a given sets of questions. These given sets of questions are holistically determined by the Cloud*e*Assurance based on the risk factors. After the cloud vendor provides the necessary self assessments to the Cloud*e*Assurance platform, it generates a score based on the given input and its internal holistically chosen formulas.

The higher the score is, the better the cloud vendor is in terms of security, governance, risks and compliance. This score is provisional and remains valid until 90 days. This score will move from being provisional to a validated state once this score is verified by a company on the HISPI managed Cloud Assurance Assessor Program (CAAP). The Holistic Information Security Practitioners Institute (HISPI) [19] is an institute that provides security training and certifications to the IT security professionals.

Based on the above introduction to Cloud*e*Assurance product, we can summarize the following differences with our prototype implementation-

- Data collection is not automated as the cloud vendor itself has to provide the self assessment manually to the system.
- The assurance score needs to be verified by a company on the HISPI managed Cloud Assurance Assessor Program (CAAP). This again involves human intervention and the verification is a manual process.
- The assurance score gives an overall idea of the cloud infrastructure indicating how good it is in terms of security, governance, risks and compliance. But it does not directly provide the information whether the cloud system is compliant with some specific control from the standards or not.

Although the data collection and verification is not automated in the Cloud*e*Assurance platform, the goal or focus area of this product is wider in scope than our thesis goals. While this product is focusing on security, governance, risks and compliance, we, on the other hand, focus specifically only on compliance.

5.2.2 Piston CloudAudit Framework

Piston Cloud Computing Inc. [29] provides a version of OpenStack for enterprise companies and was founded in the year of 2011. The core product of the company is the cloud operating system which is called the Piston Enterprise OS (pentOS) and is built on OpenStack. The pentOS includes the first implementation of the CloudAudit framework for OpenStack. The CloudAudit framework makes use of the Log Analysis approach that we have mentioned in section 3.3.3.

During our thesis we have evaluated the Piston CloudAudit framework available for open use at [56]. In this implementation, the security controls that are verified are from the NIST standard. All the implemented controls (there

are four in total) are fully automated as the data is collected automatically using log analysis and the decision is also made by the framework without any human intervention. Therefore, this work has much similarity with our work with one major difference: Piston CloudAudit framework only uses the log analysis mechanism for data collection, whereas we propose four different approaches for data collection including the log analysis. We also argue that not every necessary piece of information can be retrieved by the log analysis mechanism. Hence we introduce the other approaches as well.

Chapter 6

Discussion

Our goal in this thesis project is to explore the feasibility of building an automated security compliance tool for the cloud. This section discusses the key points of achieving this goal.

As no such tool existed when we started the project, we had to start from scratch. First, we had to identify how the system will work and what are the core components of such a tool. Using this first understanding about the tool, we built a high level architecture for it. Later, we developed a system-level architecture for the OpenStack cloud platform, using which we can actually implement an automated security compliance tool.

There were several challenges that we had to face during the implementation. The first challenge was to select the standard for which we will verify the compliance. Although most of the controls defined in one standard are defined also, sometimes a bit differently, in other standards, most of them do not provide any guidelines for the implementation. After studying several standards, we decided to use the ISO 27002 v2005 for implementing our proof-of-concept prototype. The advantage that we get from ISO 27002 over the other standards is that it provides some implementation guidelines for the controls defined in it. Also, ISO standards are the most widely explored and used by the enterprises as reported by Symantec in the *State of Security Report, 2010* [61].

While analyzing the standards, we observed that many of the controls defined in the standards require human input and for some of them it may not be possible to automate at all. For example, the physical security-related controls are extremely difficult to automate, if not impossible. Another observation in this respect is that some of the controls may only partially be

automated while the rest part of the control requires human intervention.

One of the goals of our thesis is to explore alternative approaches to automate the security compliance check. Here, the alternative approaches refer to the alternative ways of collecting data from the cloud and during this thesis we have found four possible approaches to accomplish that. One approach we consider particularly important is the use of a vulnerability scanner. One reason is that a vulnerability scanner is designed to assess computers, computer systems, networks or applications for weaknesses. To do this assessment, the vulnerability scanners collects data from the computers, which we intend to do here also. There are many types of vulnerability scanners as of today and they can do tasks like high-speed discovery, configuration auditing, asset profiling, sensitive data discovery, patch management integration and policy verification. A proprietary vulnerability scanning tool called Nessus [23] by Tenable Network Security is already providing support for the PCI DSS compliance check for LAMP (Linux, Apache, MySQL and PHP) server. All of these works are in line with our work to verify the compliance automatically. Therefore, using these vulnerability scanners we can reuse the existing knowledge that is already available and enhance this for the cloud environment where there are more challenges.

One of the major goals of this thesis is to develop a proof-of-concept prototype automated security compliance tool. During the implementation, we observed that the automated tool needs to be integrated with the target system very closely. It is mentioned in section 3.3 why this integration is important for the different approaches. The most important reason behind the close integration is that any kind of system-level modification may affect the data collection procedure of the automated security compliance tool, which in turn may affect the tool to work properly. From this, we can expect that the future automated security compliance tools will be closely coupled or integrated with the target system.

Chapter 7

Conclusion

The rapid evolution of cloud computing indicates that this will be the driving force for the the next generation of internet services. Although, as of today, there are mostly proprietary cloud platforms available, several open source cloud solutions are gaining attraction day by day. Despite this evolution, security, especially security compliance, remains as an issue prohibiting large scale adoption of cloud in the enterprise environment. And this is where an automated security compliance tool can help to build the trust between the cloud vendor and the user. At the same time, this tool can help reduce cost and time for the cloud vendor by automating the compliance check procedure. Hence, this project is important from both the theoretical point of view, by analyzing the alternative design approaches to build such a tool and from practical point of view, by implementing a prototype that shows how it can be done.

7.1 Summary of the Work

We started this project with an open problem and it was how to build an automated security compliance tool for the cloud. Initially, we started with a very high-level abstract architecture identifying what needs to be done in order to build such a tool. From this abstract architecture, we identified three core components of such a tool: (i) the data collection engine, (ii) the verification engine and (iii) the user interface to display the information. After identifying the core components, we explored the alternative ways to build the data collection engine as this is the first step towards building such an automated tool. During this part, we identified four different ways to

collect data from a cloud system. These are (i) an API, (ii) a vulnerability scanning tool, (iii) log analysis and (iv) manual entry. We also explored the benefits and demerits of all these four approaches during our work.

Later, we concentrated on the verification engine where all the intelligence of the tool lies. This verification engine takes the decision, based on the data received from the data collection engine, whether the system is compliant with some standard requirement or not. To build such an engine, we had to pay attention to some implementation considerations as there was a lack of information on implementing the controls in the standards.

The final part is to display the compliance information to the user. To do this, we have used the CloudAudit API definition standardized by CSA. The compliance information sent to the user not only contains the compliance status of the system, but also contains more detailed information that facilitated this decision making. This extra information is provided to the user to gain assurance about the decision by the automated tool.

Combining these three core components, we have built a framework to work with OpenStack cloud platform. This framework provides a system-level architecture to build an automated security compliance tool for the OpenStack cloud platform. Using this system architecture, we have implemented a proof-of-concept prototype for an automated security compliance tool. This prototype implementation is integrated with the OpenStack dashboard and Nova server from where a user of the OpenStack can automatically verify the compliance status of the implemented controls. Although our framework is primarily developed for the OpenStack cloud platform, the framework can be used for other cloud platforms as well with only minor modifications.

7.2 Future Work

The research during the thesis work opens up further possibilities to explore. As we have implemented two of the four approaches for data collection, the first objective would be to implement the other two approaches to make the tool more flexible.

The second objective is to evaluate the four approaches in terms of their performance metrics: required time, cpu, memory etc. Among these performance metrics, *time* will be one of the most important concerns for such an on demand information generating system. In order to measure the performance metrics, we can implement one security control using all four ap-

proaches and take the measurements. This will enable us to decide the most suitable approach for collecting data from the cloud.

During the thesis project, we have done research about how to automatically check the compliance status of the cloud platform, not of the virtual machines running on the platform. It will be a new area of research to verify the security compliance of the virtual machine instances running on the cloud with relevant standards.

Bibliography

[1] Amazon Elastic Compute Cloud (Amazon EC2). http://aws.amazon.com/ec2/ [accessed 09.05.2012].

[2] Amazon Web Services. http://aws.amazon.com/ [accessed 09.05.2012].

[3] Cisco Systems, Inc. http://www.cisco.com/ [accessed 30.04.2012].

[4] Cloud Computing, Managed Hosting, Dedicated Server Hosting by Rackspace. http://www.rackspace.com/cloud/cloud_hosting_products/servers/ [accessed 09.05.2012].

[5] Cloud Control Matrix (CCM) : Cloud Security Alliance. https://cloudsecurityalliance.org/research/ccm/ [accessed 18.01.2012].

[6] Cloud Security Alliance. https://cloudsecurityalliance.org/ [accessed 18.01.2012].

[7] Cloud Server and Virtual Server Hosting by Rackspace. http://www.rackspace.com/ [accessed 09.05.2012].

[8] CloudAudit. http://cloudaudit.org/CloudAudit/Home.html [accessed 24.04.2012].

[9] CloudeAssurance. https://www.cloudeassurance.com/ [accessed 24.04.2012].

[10] COBIT - IT Governance Framework - Information Assurance Control — ISACA. http://www.isaca.org/Knowledge-Center/cobit/Pages/Overview.aspx [accessed 19.01.2012].

[11] CRM - The Enterprise Cloud Computing Company - Salesforce.com. http://www.salesforce.com/ [accessed 09.05.2012].

[12] Django — The Web framework for perfectionists with deadlines. https://www.djangoproject.com/ [accessed 24.04.2012].

[13] eFortresses - Security & Compliance Solutions. http://www.efortresses.com [accessed 24.04.2012].

[14] Email Management, Compliance and Archiving Solutions for Business from Mimecast. http://www.mimecast.com/ [accessed 23.05.2012].

[15] Google App Engine. https://appengine.google.com [accessed 09.05.2012].

[16] Google Docs. http://www.google.com/google-d-s/documents/ [accessed 09.05.2012].

[17] Greenbone. http://www.greenbone.net/index.html [accessed 03.05.2012].

[18] Horizon: The OpenStack Dashboard Project - Horizon 2012.2 Documentation. http://horizon.openstack.org/ [accessed 18.01.2012].

[19] Information Security Training — IT Security Certification — HISPI. https://www.hispi.org/ [accessed 08.06.2012].

[20] ISO 27001, ISO27001 Information Security Standard. http://www.itgovernance.co.uk/iso27001.aspx [accessed 19.01.2012].

[21] ISO/IEC 27002 Code of Practice. http://www.iso27001security.com/html/27002.html [accessed 19.01.2012].

[22] NASL Guide. http://www.dn-systems.org/boss/doc/nasl_guide-20050103.pdf [accessed 27.04.2012].

[23] Nessus 5 - Tenable Network Security. http://www.tenable.com/products/nessus [accessed 03.05.2012].

[24] NIST.gov - Computer Security Division - Computer Security resource Center. http://csrc.nist.gov/publications/PubsDrafts.html [accessed 19.01.2012].

[25] Official Source of PCI DSS Data Security Standards. https://www.pcisecuritystandards.org/security_standards/index.php [accessed 18.01.2012].

[26] OpenStack >> Open Source Cloud Computing Software. http://openstack.org [accessed 18.01.2012].

[27] OpenVAS - About OpenVAS Software. `http://www.openvas.org/software.html` [accessed 18.01.2012].

[28] Paste Deployment - Paste Deploy v1.5.0 Documentation. `http://pythonpaste.org/deploy/` [accessed 24.04.2012].

[29] Piston Cloud Computing. `http://www.pistoncloud.com/` [accessed 24.04.2012].

[30] Python Programming Language - Official Website. `http://www.python.org/` [accessed 24.04.2012].

[31] Routes Documentation - Routes 1.13 documentatiion. `http://routes.readthedocs.org/en/latest/index.html` [accessed 24.04.2012].

[32] Security Guidance for Critical Areas of Focus in Cloud Computing V2.1. `https://cloudsecurityalliance.org/csaguide.pdf` [accessed 04.05.2012].

[33] SQLite Home Page. `http://www.sqlite.org/` [accessed 12.06.2012].

[34] Understanding Health Information Privacy. `http://www.hhs.gov/ocr/privacy/hipaa/understanding/index.html` [accessed 19.01.2012].

[35] Welcome to Nova's Documentation - 2012.1-dev documentation. `http://nova.openstack.org/` [accessed 18.01.2012].

[36] Windows Azure - Cloud Computing — Cloud Services — Cloud Application Development. `https://www.windowsazure.com` [accessed 09.05.2012].

[37] Writing Shell Scripts. `http://linuxcommand.org/writing_shell_scripts.php` [accessed 27.04.2012].

[38] WSGI - WSGI.org. `http://wsgi.readthedocs.org/en/latest/index.html` [accessed 24.04.2012].

[39] Bleikertz, S. Automated security analysis of infrastructure clouds. Master's thesis, Technical University of Denmark, Jun 2010. `http://nordsecmob.aalto.fi/en/programme/publications/nordsecmob_thesis_2010/` [Accessed 25.03.2012].

[40] Bleikertz, S., Schunter, M., Probst, C., Pendarakis, D., and Eriksson, K. Security audits of multi-tier virtual infrastructures in public infrastructure clouds. In *Proceedings of the 2010 ACM workshop on Cloud computing security workshop* (2010), ACM, pp. 93–102.

[41] Borenstein, N., and Blake, J. Cloud computing standards: Where's the beef? *Journal of Internet Computing, IEEE 15*, 3 (2011), 74–78.

[42] C. Hoff and S. Johnston and G. Reese and B. Sapiro. CloudAudit 1.0 IETF Draft. `http://tools.ietf.org/html/draft-hoff-cloudaudit-00` [accessed 24.04.2012].

[43] Cochran, M., and Witman, P. Governance and service level agreement issues in a cloud computing environment. *Journal of Information Technology Management 22*, 2 (2011), 41–55.

[44] Construx Software Builders Inc. Evolutionary Prototyping White Paper. `http://www.construx.com/File.ashx?cid=814` [accessed 24.04.2012].

[45] De Chaves, S., Uriarte, R., and Westphall, C. Toward an architecture for monitoring private clouds. *Communications Magazine Journal, IEEE 49*, 12 (2011), 130–137.

[46] Fielding, R. *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California, 2000.

[47] Hasselmeyer, P., and d'Heureuse, N. Towards holistic multi-tenant monitoring for virtual data centers. In *Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP* (2010), Ieee, pp. 350–356.

[48] Herrmann, D. *Complete guide to security and privacy metrics: measuring regulatory compliance, operational resilience and ROI*. CRC Press, 2007.

[49] Julisch, K. Security compliance: the next frontier in security research. In *Proceedings of the 2008 workshop on New security paradigms* (2009), ACM, pp. 71–74.

[50] Kaliski Jr, B., and Pauley, W. Toward risk assessment as a service in cloud environments. In *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing* (2010), USENIX Association, pp. 13–13.

[51] Kuppinger, M. KuppingerCole Trend Report - Top Trends 2012-2013, Apr 2012. `http://www.kuppingercole.com/report/trendreporttop2012200412` [accessed 12.06.2012].

[52] Littlewood, B., Brocklehurst, S., Fenton, N., Mellor, P., Page, S., Wright, D., Dobson, J., McDermid, J., and Gollmann, D. Towards operational measures of computer security. *Journal of Computer Security 2*, 2 (1993), 211–229.

[53] Mather, T., Kumaraswamy, S., and Latif, S. *Cloud security and privacy: an enterprise perspective on risks and compliance*. O'Reilly Media, Inc., 2009.

[54] Mell, P., and Grance, T. The NIST Definition of Cloud Computing. `http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf` [accessed 09.05.2012].

[55] Park, J., Spetka, E., Rasheed, H., Ratazzi, P., and Han, K. Near-real-time cloud auditing for rapid response. In *Advanced Information Networking and Applications Workshops (WAINA), 2012 26th International Conference on* (march 2012), IEEE, pp. 1252 –1257.

[56] Piston Cloud Computing. Piston CloudAudit Implementation on Github. `https://github.com/piston/openstack-cloudaudit` [accessed 24.04.2012].

[57] Prafullchandra, H., Owens, K., McAndrew, T., Chaubal, C., Ottenheimer, D., Tran, C., and han Yang. PCI-compliant cloud reference architecture. White Paper by HyTrust, Savvis, Coalfire Systems, VMWare and Cisco Systems. 2010.

[58] RASHEED, H. Auditing for standards compliance in the cloud: Challenges and directions. *The International Arab Journal of Information Technology 1*, 0 (2003).

[59] Siponen, M., Pahnila, S., and Mahmood, M. Compliance with information security policies: An empirical investigation. *Journal of Computer, IEEE 43*, 2 (2010), 64–71.

[60] Symantec Corporation. Endpoint, Cloud, Mobile and Virtual Security Solutions — Symantec. `http://www.symantec.com/index.jsp` [accessed 08.06.2012].

[61] Symantec Corporation. State of Enterprise Security Report 2010. `http://www.symantec.com/content/en/us/about/presskits/SES_report_Feb2010.pdf` [accessed 08.06.2012].

[62] Tancock, D., Pearson, S., and Charlesworth, A. A privacy impact assessment tool for cloud computing. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on* (2010), IEEE, pp. 667–676.

[63] Verizon. 2012 Data Breach Investigation Report. `http://www.verizonbusiness.com/resources/reports/rp_data-breach-investigations-report-2012_en_xg.pdf` [accessed 08.06.2012].

[64] Verizon. Security Services - Verizon Enterprise Solutions. `http://www.verizonbusiness.com/Products/security/` [accessed 08.06.2012].

[65] Voras, I., Mihaljevic, B., and Orlic, M. Criteria for evaluation of open source cloud computing solutions. In *Information Technology Interfaces (ITI), Proceedings of the ITI 2011 33rd International Conference on* (2011), IEEE, pp. 137–142.

Appendix A

HTML Response from CloudAudit

This section represents the output of the automated security compliance tool. When the user clics the links presented in the OpenStack dashboard for compliance check verification, the appropriate handler from the CloudAudit evidence engine collects data and verifies it against the standard. Then it makes the decision and prepares an HTML response that is delivered to the CloudAudit API server. This response is sent back to the client by this CloudAudit API server. Figure A.1 and Figure A.2 shows the sample output for the ISO 27002 controls "Clock Synchronization" and "Remote Administrative & Diagnostic Port Protection" respectively.

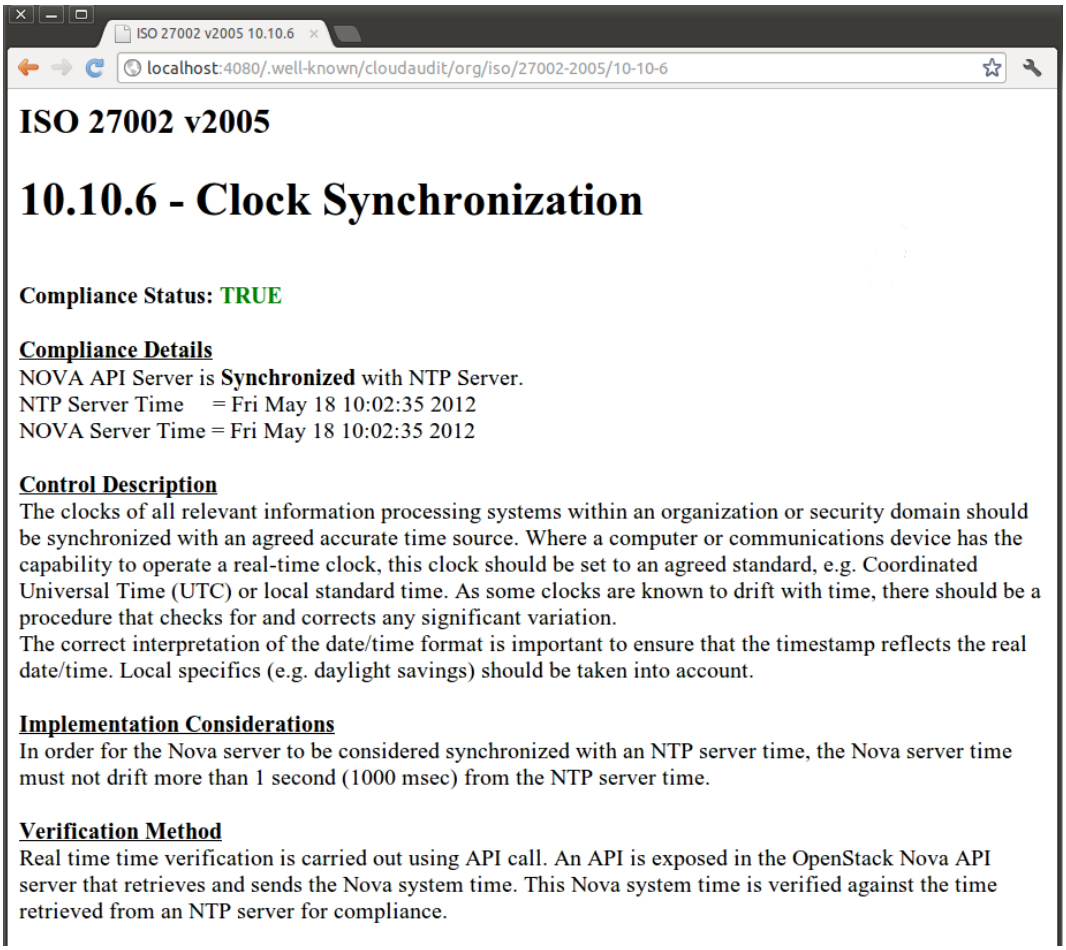


Figure A.1: Sample Output for the Security Control “Clock Synchronization” from the CloudAudit Evidence Engine

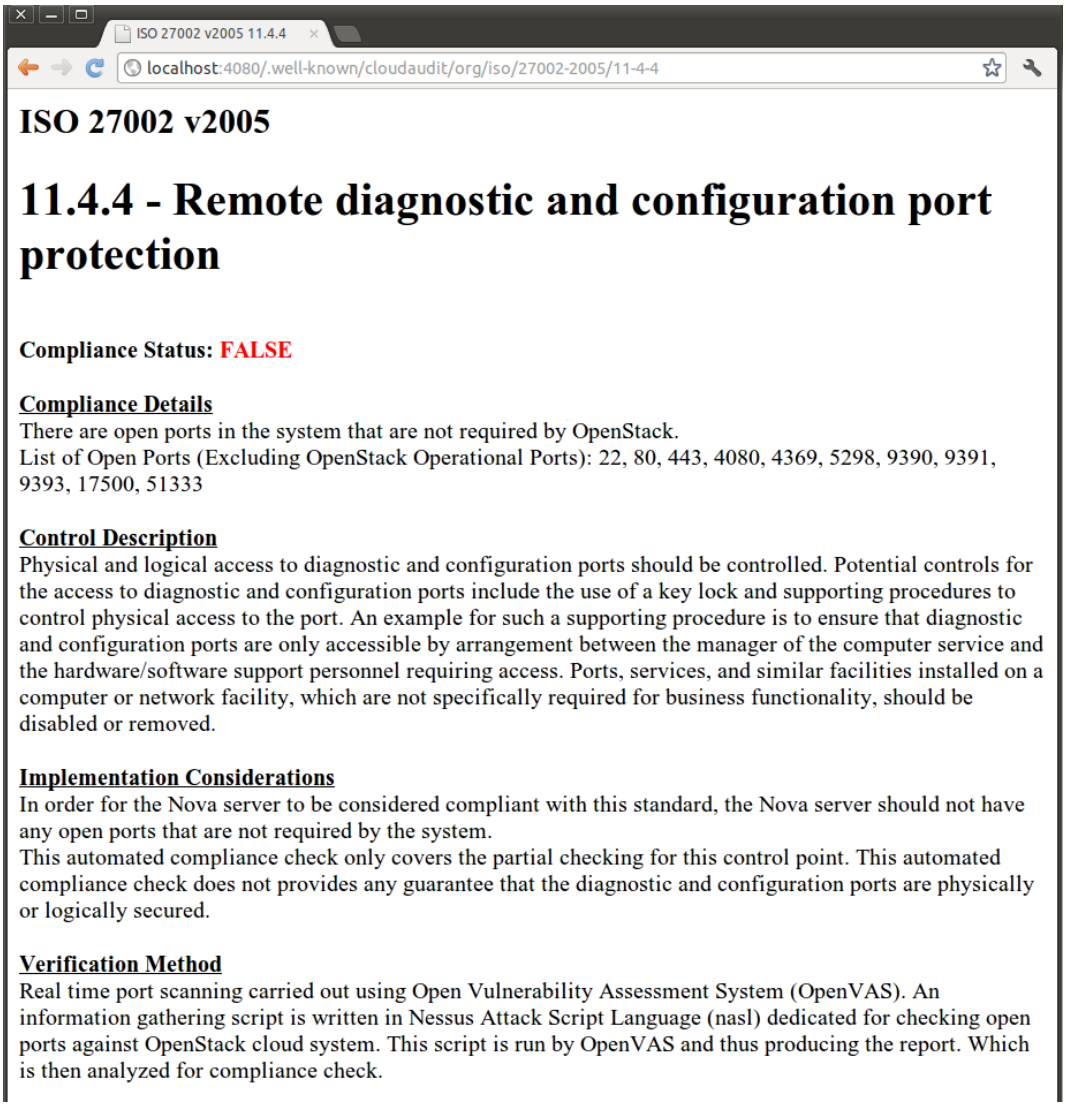


Figure A.2: Sample Output for the Security Control “Remote Administrative & Diagnostic Port Protection” from the CloudAudit Evidence Engine