Generating Cloud Monitors From Models To Secure Clouds

[1]Mr. Akshay Agrawal,[2]Mr. Ninad Sawant, [3]Mr. Vishal Shetty, [4]Mr. Hrushikesh Salve
[1]Asst.Professor,[2,3,4]UCoE Student,[1,2,3,4]Computer Engg. Dept. Shivajirao S. Jondhle College of Engineering & Technology, Asangaon, Maharshatra, India.
[1] akshay1661 @gmail.com,[2]ninad16sawant@gmail.com,[3]s.vishalshetty31@gmail.com,
[4]hrushikesh.salve@gmail.com

Abstract- Authorization is a very important security concern in cloud computing environment. It aims to control an access of the users to system. An oversized range of resources related to REST APIs typical in cloud makes an implementation of security requirements challenging and error-prone. To overcome the problem, security cloud monitors are used. It rely on model-driven approach to represent the functional and security requirements. Models are used to generate cloud monitors. The cloud monitors contain contracts that automatically verifies the implementation. It uses Django framework to implement cloud monitor and OpenStack to validate the implementation.

Keywords- Cloud Computing, REST APIs, Cloud Monitoring Framework.

I. INTRODUCTION

Cloud computing services offer REST APIs to their consumers. REST APIs, e.g., AWS, Windows Azure, OpenStack, define software interfaces allowing for the use of their resources in various ways. The REST architectural style exposes each piece of information with a URI[8][9], which results in a large number of URIs that can access the system. Data breach and loss of critical data are among the top cloud security threats. The large number of URIs further complicates the task of the security experts, who should ensure that each URI, providing access to their system, is safe-guarded to avoid data breaches or privilege escalation attacks. Since the source code of the Open Source clouds is often developed in a collaborative manner, it is a subject of frequent updates. The updates might introduce or remove a variety of features and hence, violate the security properties of the previous releases. It makes it rather unfeasible to manually check correctness of the APIs access control implementation and calls for enhanced monitoring mechanisms.

It uses UML (Unified Modelling Language) models with OCL(Object Constraint Language) to specify the behavioral interface with security constraints for the cloud implementation. The behavioral interface of the REST API[8] provides information regarding the methods that can be invoked on it and pre- and post-conditions of the methods. The pre- and post-conditions are usually given as the textual descriptions associated with the API methods. It relies on the *Design by Contract (DbC)* framework[10], which allow to define security and functional requirements as verifiable contracts. The methodology enables creating a (stateful) wrapper that emulates the usage scenarios and defines security-enriched behavioral contracts to monitor cloud.

II. AIM AND OBJECTIVE

a) Aim

The aim of the paper is to form a cloud monitor for authorization of information stored on cloud using REST options. It depends on design model created by application owners for creating a control monitor. It aims at regulation associate in nursing access of the users to system resources. An over-sized variety of resources related to REST APIs typical in cloud makes an implementation of security needs difficult and fallible.

b) Objective

The objective of the paper is to keep track of users (Employees) actions on system. Also monitoring traffic on system's network and to prevent system from cyber-attacks. Using cloud monitors for making a cloud system more secure.

III. LITERATURE SURVEY

Paper 1: Log-based cloud monitoring system for OpenStack:

Cloud systems are emerging to be fast computing environment, providing their users with resources and services over the Internet. OpenStack is a open source platform which helps in building private clouds for the Industries. Monitoring the activities in the cloud system is a task of utmost very importance and log files are often

 DOI : 10.35291/2454-9150.2020.0237

used for it. Logs management plays a very important role in various issues in the cloud network or helps in troubleshooting the issues. It's a system which monitor the logs of OpenStack components (i.e., Nova, Neutron, Cinder) in real-time and generate an alert for the information[2], debug, error, warning, and trace messages as soon as an issue recorded in the logs. It helps in easy readability to the administrators. Apart from the logic also helps to generate warning messages prior to the different resource quota like virtual CPUs, Floating IP's, etc. It develops the proposed system and demonstrate its feasibility and performances through simulation.

Paper 2: Industrial Cloud Monitoring System Based on Internet of Things:

Monitoring for organizations can get benefit from Cloud Computing and Internet of Things capabilities. The research proposes a new approach to monitor machines based on the low coast board (Omega2 Plus). The Omega2 Plus is an embedded system and its shape and size like a credit card work as a computer. It can transmit data to the cloud server directly. In any industrial factory there are multiple types of machines, the approach coupled machines logical outputs to Omega2 Plus GPIO and send data to web-based server to be monetarized. The Cloud Monitoring System (CMS) gives a real-time information about production data[3], current work, the employee ID working on a given machine, the production rate, and downtime. All machines monitored by supervisors or specialists sitting in a remote place virtually any part of the world. Additionally, the Cloud Monitoring System is simple and run by an untrained worker employed on the basic level.

Paper 3: A framework for monitoring and security authentication in cloud based on Eucalyptus:

Since the cloud computing was proposed, the research of cloud instantaneously swept the world and all kinds of cloud products have appeared in public eyes. But if the cloud platform security is not guaranteed, companies or individuals will not run and store their own data in the cloud. So how to put these cloud product promotions out and make interest to the company becomes the company's biggest problem, and the credible cloud's service has become an urgency to implement. The monitoring system collects data from cloud system real time, and makes the corresponding judgment according to the analysis results. At the same time, the system will report the data to the third party to record and modify the system rules timely. By these means, the cloud system can improve the stability of the service and make it more credible.

IV. EXISTING SYSTEM

In many companies, private clouds are considered to be an important element of data center transformations. Private clouds are dedicated cloud environments created for the internal use by a single organization. Therefore, designing and developing secure private cloud environments for such a large number of users constitutes a major engineering challenge. Usually, cloud computing services offer REST APIs (Representational State Transfer Application Programming Interface) to their consumers. The REST architectural style exposes each piece of information with a URI, which results in a large number of URIs that access system.

V. COMPARATIVE STUDY

Table 1: Comparative Study

Sr. no	Paper Name	Author/ publication	Technology	Advantage	Disadvantages
1)	Log-Based Cloud Monitoring System for OpenStack	Vaibhav Agrawal, Devanjal Kotia, Kamelia Moshirian, Mihui Kim*	Cloud Monitoring; Log-based; Real-time Monitoring; OpenStack; Convenient Managemet	OpenStack components in real-time and generate an alert for the information, debug, error, warning, It helps in easy readability to the administrators	Maintenance costs log types, and log sources can grow to overwhelming proportions—and the project often ends up killed. need highly competent staff to add, change, or repair it.
2)	Industrial Cloud Monitoring System based on Internet of Things	Saif Aldeen Saad Obayes Al -Kadhim, R. K. Al-Saedi	Cloud Monitoring, Machine Monitoring, Internet of Things, Omega 2+, Node.js.	The framework of the Cloud Monitoring Systems uses the cloud computing technology, which greatly improves the operating efficiency of the Internet of things.	The designing, developing, and maintaining and enabling the large technology to IoT system is quite complicated.
3)	A Framework for Monitoring and Security Authentication in Cloud based on Eucalyptus.	Zefeng Gao, Xiaoyong Li	Cloud Computing, Eucalyptus, KVM, XEN.	The monitoring system collects data from cloud system real time.	The implementation cost is high. Security is not guaranteed.

 DOI : 10.35291/2454-9150.2020.0237

VI. PROBLEM STATEMENT

It became tedious for organization provide cloud security for large number of users. Most of the approaches try to separate security concerns from core business logic, but few weave security aspects into primary models**.** In model driven approach, it is associated tools for monitoring security of clouds. The approach is not fully automatic because it requires the user intervention in filling in the missing lines of code in the generated code skeletons. It's done to keep the models readable and avoid cluttering them with too many low-level details.

VII. PROPOSED SYSTEM

The cloud monitoring framework that supports a semi-automated approach to monitoring a private cloud implementation with respect to its conformance to the functional requirements and API access control policy. The work uses UML (Unified Modeling Language) models with OCL (Object Constraint Language) to specify the behavioral interface with security constraints for the cloud implementation. The behavioral interface of the REST API provides an information regarding the methods that can be invoked on it and pre- and post-conditions of the methods. The pre- and post-conditions are usually given as the textual descriptions associated with the API methods. It relies on the Design by Contract (DBC) framework, which allows defining security and functional requirements as verifiable contracts. The methodology enables creating a (stateful) wrapper that emulates the usage scenarios and defines security-enriched behavioral contracts to monitor cloud. The proposed approach also facilitates the requirements traceability by ensuring the propagation of the security specifications into the code. It also allows the security experts to observe the coverage of the security requirements during the testing phase. The approach is implemented as a semi-automatic code generation tool .

VIII. ALGORITHM

THE PUSH-PULL ALGORITHM

Step 1: While(true)
Step 2: Set pull operation identifier is pulled<- false
Step 3: Waiting for the termination of the push_interval
Step 4: If is Pulled equals to true during Push_interval
Step 5: Push current update to master node
Step 6: else if (| current_value – previous_value | / (MAX - MIN)≥ UTD) Step 7: IsPushed<- false
Step 8: Push current update to master node Step 9: //end while

Keystone

Step 1: Receive credentials from client.
Step 2: Send token to client.
Step 3: Client send request + token to Cinder.

Step 4: if(token==valid) Goto step 5 else goto step 1.

Step 5: Client receives response.

IX. MATHEMATICAL MODEL

$$\text{change\_degree} = \frac{|Pi-Ci|}{MAX-MIN} \leq UTD$$

Where, change_degree describes the change between the current status of Producer i, which is defined as Pi and the last Status information that the consumer received, which is defined as Ci, MAX and MIN represents the maximal and minimal possible value of the status.

This hybrid model depends basically on the user requirements, which is defined as UTD1:

The P&P algorithm has three possible cases based on the value of UTD

1- Dominating push-based when UTD is relatively small,

2- Dominating pull-based when UTD is relatively large, and

3- None dominating when UTD is relatively moderate.

X. SYSTEM ARCHITECTURE

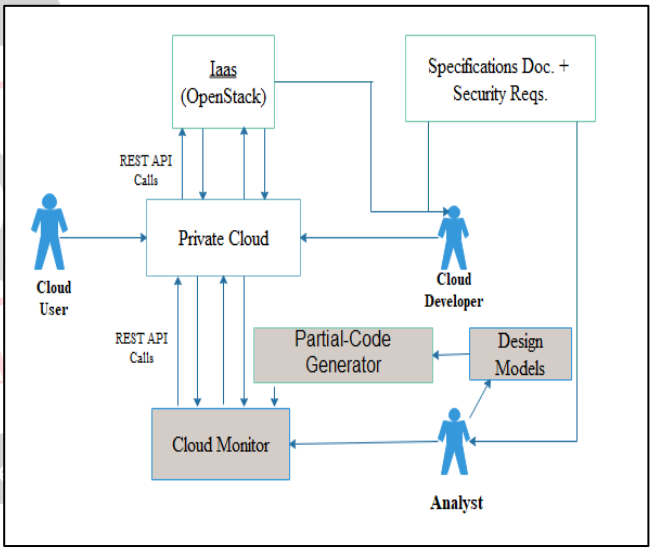

Fig.1: System Architecture

Description: Figure 1 presents the overall architecture of the Cloud Monitoring Framework. A cloud developer uses IaaS to develop a private cloud for her/his organization that would be used by different cloud users within the organization. In some cases, the private cloud may be implemented by a group of developers working collaboratively on different machines. The REST API provided by IaaS is used to develop the private cloud according to the specification document and required security policy. The cloud monitor is implemented on top of the private cloud.

The main original components of the work are highlighted as grey boxes in Figure 1. The security analyst develops the required design models based on the specification document and security policies. These models define the

 DOI : 10.35291/2454-9150.2020.0237

behavioral interface for the private cloud and specify its functional and security requirements. In addition, the design models define all the information required to build the stateful scenarios using REST as the underlying stateless architecture. The REST architectural style exposes each piece of information with a URI .which results in a large number of URIs that can access the system.

XI. ADVANTAGES

1) It relies on the model-driven approach to design APIs that exhibit REST interface features.

2) The proposed semi-automated approach aimed at helping the cloud developers and security experts to identify the security loop holes in the implementation by relying on modeling rather than manual code inspection or testing.

3) It helps to spot the errors that might be exploited in data breaches or privilege escalation attacks.

4) Openstack which is an open-source cloud computing platform provides free private cloud services.

5) The particular system is very easy to handle by it's users as well as admin.

XII. DESIGN DETAILS

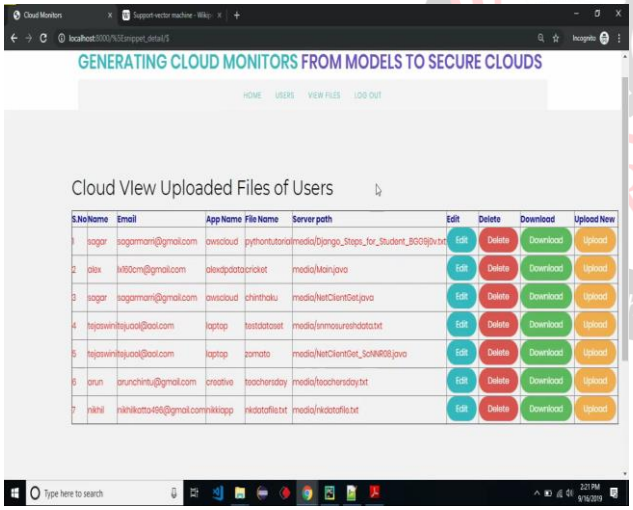

Fig.2: cloud view

XIII. CONCLUSION

Thus we have tried to implement paper "Irum Rauf, Elena Troubitsyna." ,"*Generating Cloud Monitors from Models to Secure Clouds* " . IEEE 2018. It rely on the model-driven approach to design APIs that exhibit REST interface features. The cloud monitors, generated from the models, enable an automated contract-based verification of correctness of functional and security requirements, which are implemented by a private cloud infrastructure. The proposed semi-automated approach aimed at helping the cloud developers and security experts to identify the security loopholes in the implementation by relying on modeling rather than manual code inspection or testing. It helps to spot the errors that might be exploited in data breaches or privilege escalation attacks.

REFERENCE

[1]Troubitsyna, E., & Rauf, I. (2018). Generating Cloud Monitors from Models to Secure Clouds. 2018 48th IEEE/IFIP International Conference on Dependable Systems and Networks.

[2] Agrawal, Vaibhav, et al. "Log-based cloud monitoring system for OpenStack." 2018 IEEE Fourth International Conference on Big Data Computing Service and Applications (BigDataService). IEEE, 2018.

[3] Al-Saedi, Ibtesam RK, and Saif Aldeen Saad Obayes Al-Kadhim. "Industrial Cloud Monitoring System Based on Internet of Things." 2018 Second World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4). IEEE, 2018.

[4] OpenStack Block Storage Cinder. https://wiki.openstack.org/wiki/ Cinder. Accessed: 26.03.2018.

[5] OpenStack Newton - Installation Guide. https://docs.openstack.org/ newton/install-guide-ubuntu/overview.html. Accessed: 20.11.2017.

[6] Gao, Zefeng, and Xiaoyong Li. "A framework for monitoring and security authentication in cloud based on Eucalyptus." 2015 IEEE International

Conference on Signal Processing, Communications and Computing (ICSPCC). IEEE, 2015.

[7] Mohamed Almorsy et al. Adaptable, model-driven security engineering for SaaS cloud-based applications. Automated Software Engineering,21(2):187–224, 2014.

[8] A. Holovaty and J. Kaplan-Moss. The Django Book. Online version ofthe Django Book, 2010. http://docs.djangoproject.com/en/1.2/.

[9] Adrian Holovaty and Jacob Kaplan-Moss. The definitive guide to Django: Web development done right. Apress, 2009.

[10] MM Alam et al. Model driven security for web services (mds4ws).In Multitopic Conference, 2004. Proceedings of INMIC 2004. 8th International, pages 498–505. IEEE, 2004Conference on Signal Processing, Communications and Computing (ICSPCC). IEEE, 2015.

 DOI : 10.35291/2454-9150.2020.0237