Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 4, April : 2024

GENERATING CLOUD MONITORS FROM MODELS TO SECURE CLOUDS

Mrs. Annepu Sruthi Patro[1],Karedla SeethaRam Sai Santhosh[2],Thomurothu Naveena[3], Keshav Allu[4], Gulla Bharat VenkatKumar[5]
[1]Assistant Professor, Department of Computer Science & Engineering, Raghu Engineering College, Vishakhapatnam, Andhra Pradesh
[2,3,4,5] Student of B-TECH, Raghu Institute of Technology, Vishakhapatnam, Andhra Pradesh
Email:- sruthi.annepu@raghuenggcollege.in, santhoshkaredla1438@gmail.com, naveenathomurothu@gmail.com , Keshavallu176@gmail.com, gullabharat@outlook.com

ABSTRACT

One major security risk in cloud computing settings is authorization. Its goal is to control user access to system resources. The execution of security standards is difficult and prone to errors because of the abundance of resources that are typically connected with REST APIs in the cloud. In this research, we suggest an implementation of security cloud monitor to mitigate this issue. To express the functional and security requirements, we use a model-driven approach. Cloud monitors are then created using the models. Contracts that are used to automatically verify implementation are contained in the cloud monitors. We implement cloud monitor using the Django web framework, and we validate our solution using OpenStack.

KEYWORDS:- security, security cloud monitor, OpenStack, REST APIs

1. INTRODUCTION

Private clouds are seen by many businesses as a crucial component of data center modernization. Private clouds are specific cloud environments built by a single company for internal usage. 72% of cloud customers utilize private clouds, while 67% use hybrid clouds, which include public and private cloud infrastructure. This information comes from the 2017 Cloud Survey. The businesses using private clouds range in size from 500 to over 2000 workers. It is therefore a significant engineering problem to design and construct safe private cloud environments for so many users.

REST APIs (REpresentational State Transfer Application Programming Interface) are typically provided by cloud computing services to their users. REST APIs define software interfaces that enable different ways to use their resources, such as AWS, Windows Azure, and OpenStack. Since every piece of information is exposed by the REST architectural style through a unique universal resource locator (URI), there are many URIs that can access the system. Among the biggest risks to cloud security are data breaches and the loss of important data. The duty of security professionals is made more difficult by the sheer amount of URIs. They need to make sure that every URI that grants access to their

system is protected in order to prevent data breaches and privilege escalation attacks.

The Open Source clouds' source code is updated frequently since it is frequently created collaboratively. The updates may break the security features of the earlier editions by adding or removing a number of features. It necessitates more sophisticated monitoring systems and makes it practically impossible to manually verify that the access control implementation of the API is accurate. In this research, we provide a system for cloud monitoring that facilitates a semi-automated way to monitoring compliance between an implementation of a private cloud and API access control policy and functional requirements. Our study specifies the behavioral interface with security constraints for the cloud implementation using UML (Unified Modeling Language) models with OCL (Object Constraint Language).

The REST API's behavioral interface gives information on the methods that can be called on it as well as the pre- and post-conditions associated with those methods. Pre- and post-conditions are typically provided as textual descriptions linked to the API calls in current usage. We utilize the Design by Contract (DbC) framework in our work since it enables us to specify functional and security needs as verifiable contracts. Our approach makes it possible to design a (stateful) wrapper that models usage scenarios and defines behavioral contracts with enhanced security for cloud monitoring. Moreover, by guaranteeing the propagation of the security specifications into the code, the suggested approach also makes requirements tracing easier.

This facilitates the observation of the testing phase's coverage of the security requirements by the security experts. Using OpenStack as a case study, the methodology is applied as a semi-automated code creation tool in the Python web framework Django. An open source cloud computing system called OpenStack offers Infrastructure as a Service, or IaaS. Promising findings from the validation using OpenStack encourage us to pursue the tool development detailed in this work. The structure of the paper is as follows: Our work is motivated by part II. An overview of our cloud monitoring framework is provided in Section III. We outline our design methodology for stateful REST service modeling in part IV. Section V provides a description of the contract generating technique.

2. LITERATURE SURVEY AND RELATED WORK

2.1)Model driven security for web services

AUTHORS: MM Alam et al.

Model driven architecture is an approach to increase the quality of complex software systems based on creating high level system models that represent systems at different abstract levels and automatically generating system architectures from the models. We show how this paradigm can be applied to what we call model driven security for Web services. In our approach, a designer builds an interface model for the Web services along with security requirements using the object constraint language (OCL) and role based access control (RBAC) and then generates from these specifications a complete configured security infrastructure in the form of Extended Access Control Markup Language (XACML) policy files. Our approach can be used to improve productivity during the development of secure Web services and quality of resulting systems.

2. 2) Run-time generation, transformation, and verification of access control models for self-protection

AUTHORS: Chen, Bihuan; Peng, Xin; Yu, Yijun; Nuseibeh, Bashar and Zhao, Wenyun (2014).

A self-adaptive system uses runtime models to adapt its ar-chitecture to the changing requirements and contexts. How-ever, there is no one-to-one mapping between the require-ments in the problem space and the architectural elements in the solution space. Instead, one refined requirement may crosscut multiple architectural elements, and its realization in volves complex behavioral or structural interactions manifested as architectural design decisions. In this paper we pro-pose to combine two kinds of self-adaptations: requirements-driven self-adaptation, which captures requirements as goal models to reason about the best plan within the problem space, and architecture-based self-adaptation, which cap-tures architectural design decisions as decision trees to search for the best design for the desired requirements within the contextualized solution space. Following these adaptations, component-based architecture models are reconfigured using incremental and generative model transformations. Com-pared with requirements-driven or architecture-based approaches, the case study using an online shopping bench-mark shows promise that our approach can further improve the effectiveness of adaptation (e.g. system throughput in this case study) and offer more adaptation flexibility

2.3)Towards development of secure systems using umlsec.

AUTHORS: Jan J¨urjens

We show how UML (the industry standard in object-oriented modelling) can be used to express security requirements during system development. Using the extension mechanisms provided by UML, we incorporate standard concepts from formal methods regarding multi-level secure systems and security protocols. These definitions evaluate diagrams of various kinds and indicate possible vulnerabilities.On the theoretical side, this work exemplifies use of the extension mechanisms of UML and of a (simplified) formal semantics for it. A more practical aim is to enable developers (that may not be security specialists) to make use of established knowledge on security engineering through the means of a widely used notation

2.4)Cloud computing the business perspective

AUTHORS: Sean Marston et al

The evolution of cloud computing over the past few years is potentially one of the major advances in the history of computing. However, if cloud computing is to achieve its potential, there needs to be a clear understanding of the various issues involved, both from the perspectives of the providers and the consumers of the technology. While a lot of research is currently taking place in the technology itself, there is an equally urgent need for understanding the business-related issues surrounding cloud computing. In this article, we identify the strengths, weaknesses, opportunities and threats for the cloud computing industry. We then identify the various issues that will affect the different stakeholders of cloud computing. We also issue a set of recommendations for the practitioners who will provide and manage this technology. For IS researchers, we outline the different areas of research that need attention so that we are in a position to advice the industry in the years to come. Finally, we outline some of the key issues facing governmental agencies who, due to the unique nature of the technology, will have to become intimately involved in the regulation of cloud computing.

2.5)An Extensive Systematic Review on Model-Driven Development of Secure Systems

AUTHORS: PhuHNguyenetal

Model-Driven Security (MDS) is as a specialised Model-Driven Engineering research area for supporting the development of secure systems. Over a decade of research on MDS has resulted in a large number of publications. Objective: To provide a detailed analysis of the state of the art in MDS, a systematic literature review (SLR) is essential. Method: We conducted an extensive SLR on MDS. Derived from our research questions, we designed a rigorous,

extensive search and selection process to identify a set of primary MDS studies that is as complete as possible. Our three-pronged search process consists of automatic searching, manual searching, and snowballing. After discovering and considering more than thousand relevant papers, we identified, strictly selected, and reviewed 108 MDS publications. Results: The results of our SLR show the overall status of the key artefacts of MDS, and the identified primary MDS studies. E.g. regarding security modelling artefact, we found that developing domain-specific languages plays a key role in many MDS approaches. The current limitations in each MDS artefact are pointed out and corresponding potential research directions are suggested. Moreover, we categorise the identified primary MDS studies into 5 principal MDS studies, and other emerging or less common MDS studies. Finally, some trend analyses of MDS research are given. Conclusion: Our results suggest the need for addressing multiple security concerns more systematically and simultaneously, for tool chains supporting the MDS development cycle, and for more empirical studies on the application of MDS methodologies. To the best of our knowledge, this SLR is the first in the field of Software Engineering that combines a snowballing strategy with database searching. This combination has delivered an extensive literature study on MDS.

.

3. Implementaion Study:-

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 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 safeguarded 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.

3.1 Proposed Methodology:-

We present a 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. Our 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.

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 4, April : 2024

In the current practice, the pre- and post-conditions are usually given as the textual descriptions associated with the API methods. In our work, we rely on the Design by Contract (DbC) framework, which allows us to define security and functional requirements as verifiable contracts.

➢ Our methodology enables creating a (stateful) wrapper that emulates the usage scenarios and defines security-enriched behavioural contracts to monitor cloud.

➢ The proposed approach also facilitates the requirements traceability by ensuring the propagation of the security specifications into the code. This 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 in Django a Python web framework.

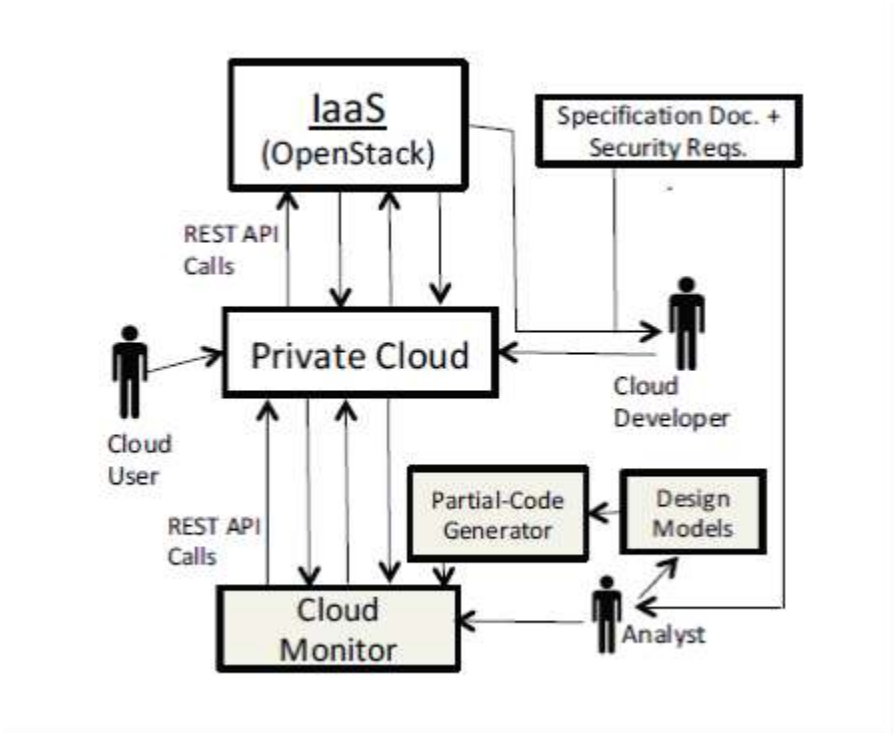

Fig 1:- System Architecture

4. METHODOLOGIES & Alogritham

4.1 MODULES

- 1. user
- 2.cloud
- 3. admin

· 4. Machine learning

1 User

It defines the access rights of the cloud users. A volume can be created, if the it has not exceeded its quota of the permitted volumes and a user Authorization is an important security concern in cloud computing environments. a POST request from the authorized user on the volumes resource would create a new volume. a DELETE request on the volume resource by an authorized user would delete the volume . if the user of the service is authorized to do so, and the volume is not attached to any instance .It aims at regulating an access of the users to system resources.

2 Cloud

The cloud monitors contain contracts used to automatically verify the implementation . 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, this private cloud may be implemented by a group of developers working collaboratively on different machines. We use Django web framework to implement cloud monitor and OpenStack to validate our implementation.

3 Admin

the cloud administrator using Keystone and users or usergroups are assigned the roles in these projects. It defines the access rights of the cloud users in the project. A volume can be created, if the project has not exceeded its quota of the permitted volumes and a user is authorized to create a volume in the project. Similarly, a volume can be deleted, if the user of the service is authorized to do so, and the volume is not attached to any instance, i.e., its status is not in-use.

4 Machine learning

Machine learning refers to the computer's acquisition of a kind of ability to make predictive judgments and make the best decisions by analyzing and learning a large number of existing data. The representation algorithms include deep learning, artificial neural network, decision tree, enhancement algorithm and so on. The key way for computers to acquire artificial intelligence is machine learning. Nowadays, machine learning plays an important role in various fields of artificial intelligence. Whether in aspects of internet search, biometric identification, auto driving, Mars robot, or in American presidential election, military decision assistants and so on, basically, as long as there is a need for data analysis, machine learning can be used to play a role.

4.2 ANN Algorithm:-

Using artificial neural networks (ANNs) for cloud monitoring can provide powerful insights into the performance, security, and optimization of cloud-based systems. Here's how it can be done:

1. **Data Collection**: The first step is to gather data from various sources within the cloud environment. This can include metrics such as CPU usage, memory usage, network traffic, disk I/O, application logs, and security logs.
2. **Feature Selection**: Once the data is collected, relevant features need to be selected for analysis. This involves identifying which metrics are most important for monitoring and decision-making.
3. **Data Preprocessing**: Raw data often needs preprocessing before it can be fed into a neural network. This may involve normalization, scaling, outlier detection, and handling missing values.
4. **Model Training**: After preprocessing, the data is used to train the neural network model. The architecture of the neural network can vary based on the specific requirements of the cloud monitoring task. For example, a recurrent neural network (RNN) or a convolutional neural network (CNN) may be suitable for time-series data or image data, respectively.

5. **Anomaly Detection**: One common use case for neural networks in cloud monitoring is anomaly detection. The trained neural network can learn the normal patterns of behavior in the cloud environment and flag any deviations as anomalies. This can help detect potential security threats or performance issues.
6. **Predictive Maintenance**: Neural networks can also be used for predictive maintenance in cloud environments. By analyzing historical data, the neural network can predict when hardware failures or performance degradation are likely to occur, allowing proactive measures to be taken to prevent downtime.
7. **Optimization**: Neural networks can help optimize resource allocation in the cloud by analyzing workload patterns and suggesting adjustments to resource provisioning based on predicted future demand.
8. **Visualization and Reporting**: The insights generated by the neural network can be visualized in dashboards or reports for easy interpretation by cloud administrators and other stakeholders.
9. **Continuous Learning**: Cloud environments are dynamic and constantly evolving, so it's important for the neural network model to be continuously updated and retrained with new data to adapt to changing conditions.

By leveraging artificial neural networks for cloud monitoring, organizations can improve the reliability, security, and efficiency of their cloud-based systems.

4.3 Decision Tree Algorithm:-

Decision tree algorithms can also be utilized for cloud monitoring tasks. Here's how you can apply decision trees in this context:

1. **Data Collection**: Similar to using neural networks, the first step is to gather relevant data from various sources within the cloud environment. This includes metrics such as CPU usage, memory usage, network traffic, disk I/O, application logs, and security logs.
2. **Data Preprocessing**: Once the data is collected, it needs to be preprocessed. This involves tasks such as data cleaning, handling missing values, and encoding categorical variables if any.
3. **Feature Selection**: Decision trees automatically select features based on their ability to split the data effectively. However, you may still need to analyze the importance of features and possibly perform feature engineering to improve the model's performance.
4. **Model Training**: Train the decision tree model using the preprocessed data. The decision tree algorithm recursively splits the data based on feature values to create a tree-like structure that represents decision rules.
5. **Anomaly Detection**: Decision trees can be used for anomaly detection by identifying data points that fall into rare or unexpected branches of the tree. Instances that follow an unusual path through the decision tree may be flagged as anomalies.
6. **Classification and Prediction**: Decision trees can classify cloud events or predict outcomes based on the input features. For example, they can classify whether a particular cloud instance is experiencing a performance issue or predict future resource usage based on historical data.
7. **Interpretability**: One of the advantages of decision trees is their interpretability. The decision rules represented by the tree structure are easy to understand and can provide insights into the factors affecting cloud performance or security.
8. **Ensemble Methods**: Decision tree algorithms can also be combined with ensemble methods such as Random Forest or Gradient Boosting to improve performance and robustness.

9. **Continuous Learning**: Like neural networks, decision tree models can also be updated with new data to adapt to changing conditions in the cloud environment.

Decision tree algorithms offer a transparent and interpretable approach to cloud monitoring, making them suitable for scenarios where understanding the reasoning behind decisions is important. They can effectively handle both classification and regression tasks and can be a valuable tool in a cloud monitoring toolkit.

5. RESULTS AND DISCUSSION SCREEN SHOTS

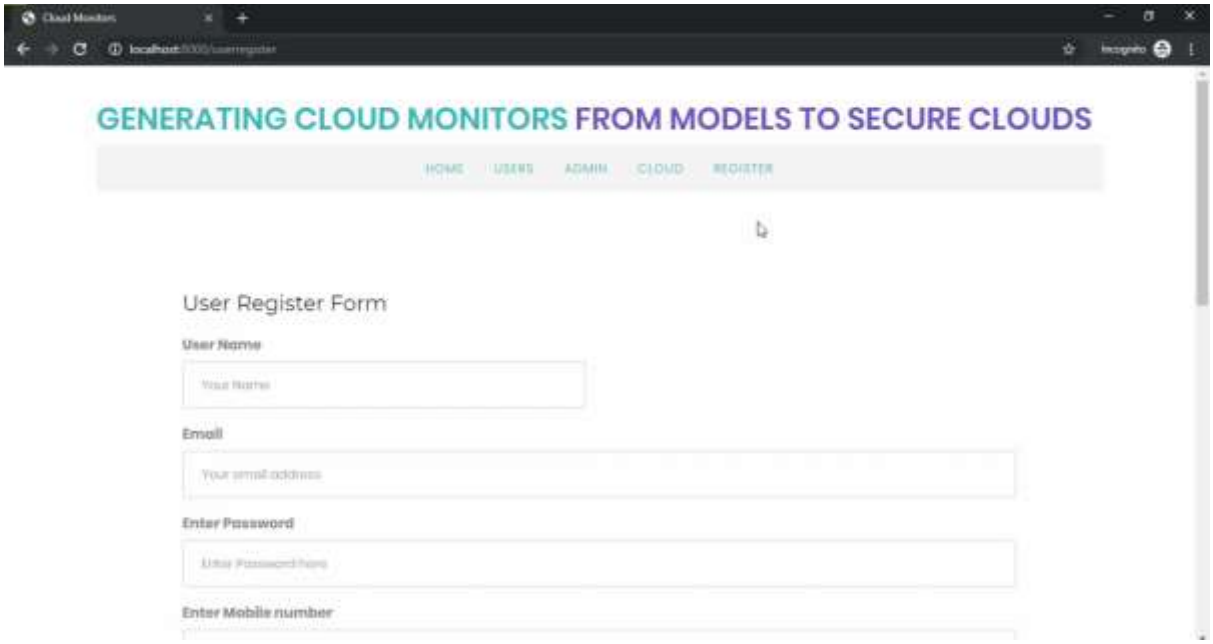

Fig 2:- User register

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 4, April : 2024

Fig 3:- Admin login

Fig 4:- Admin home page

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 4, April : 2024

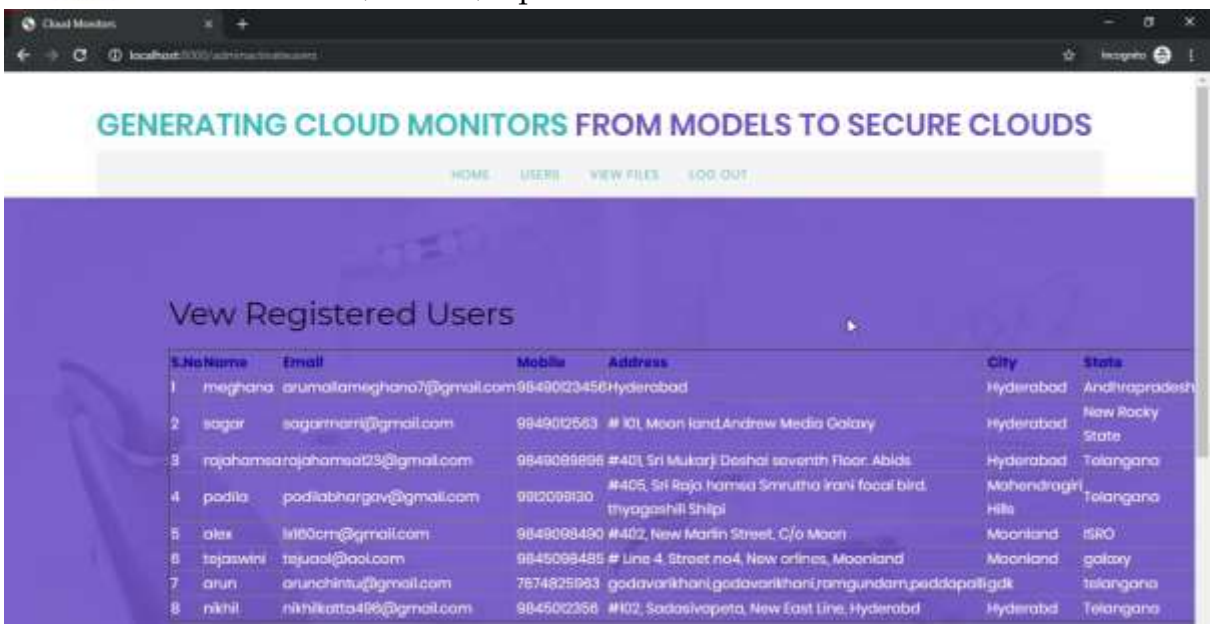

Fig 5:- Admin page where the user is approved for transcation

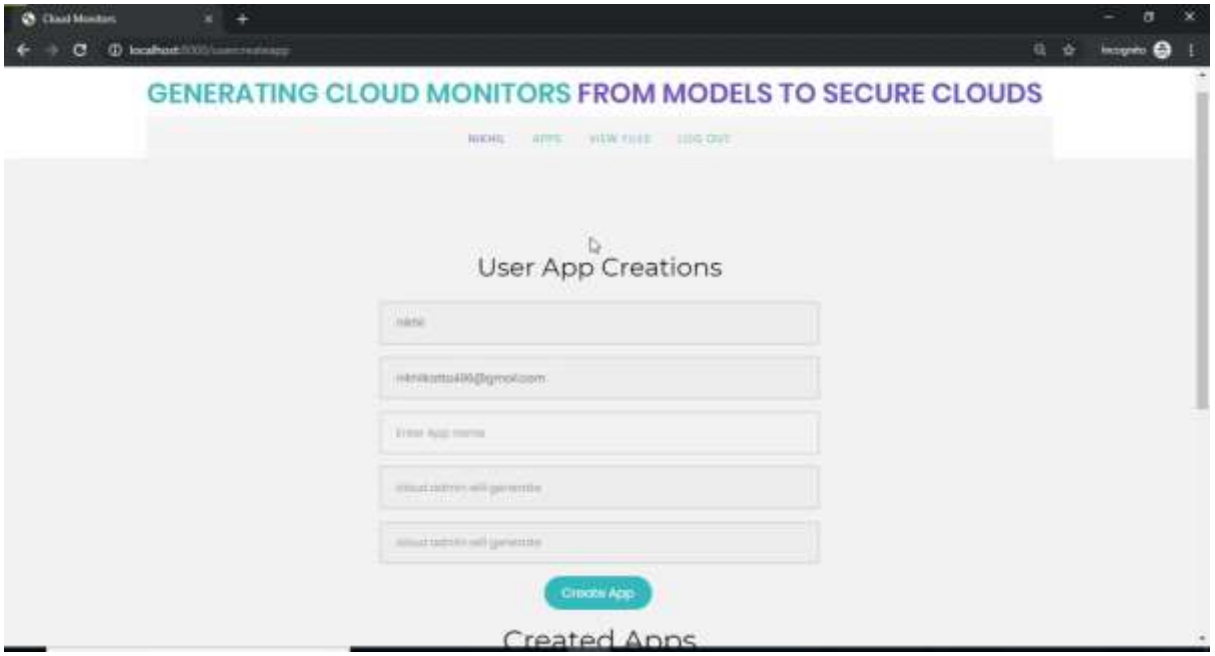

Fig 6:- User creates an service app

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 4, April : 2024

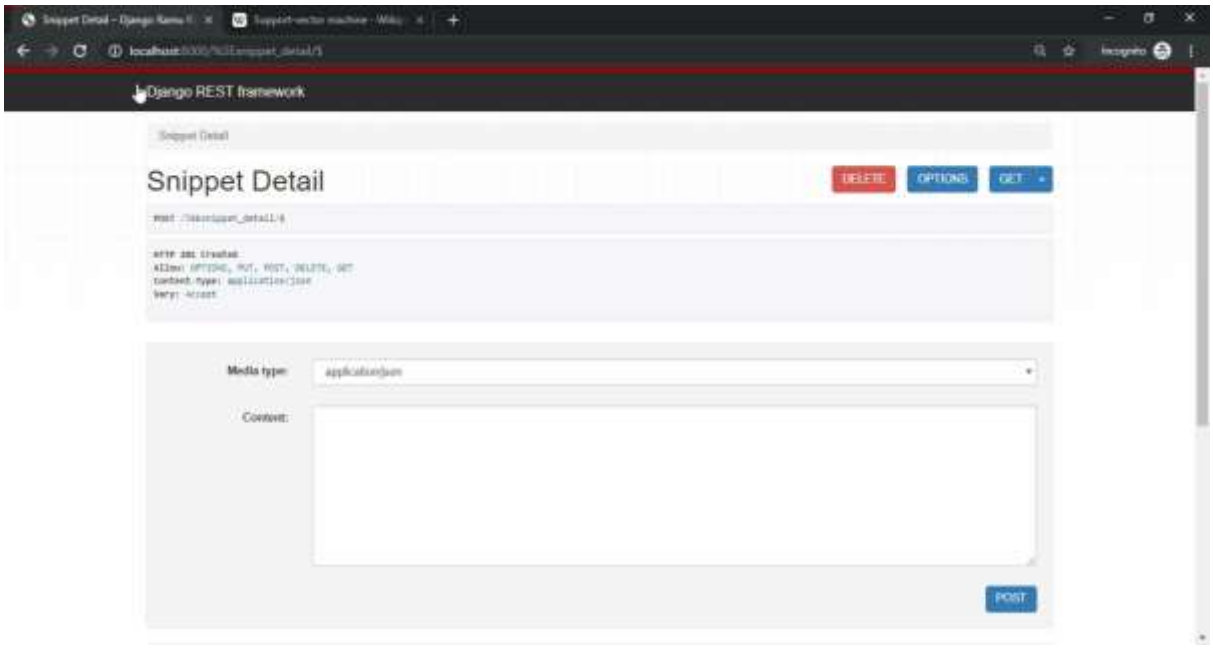

Fig 7:- Django rest service deployed on server

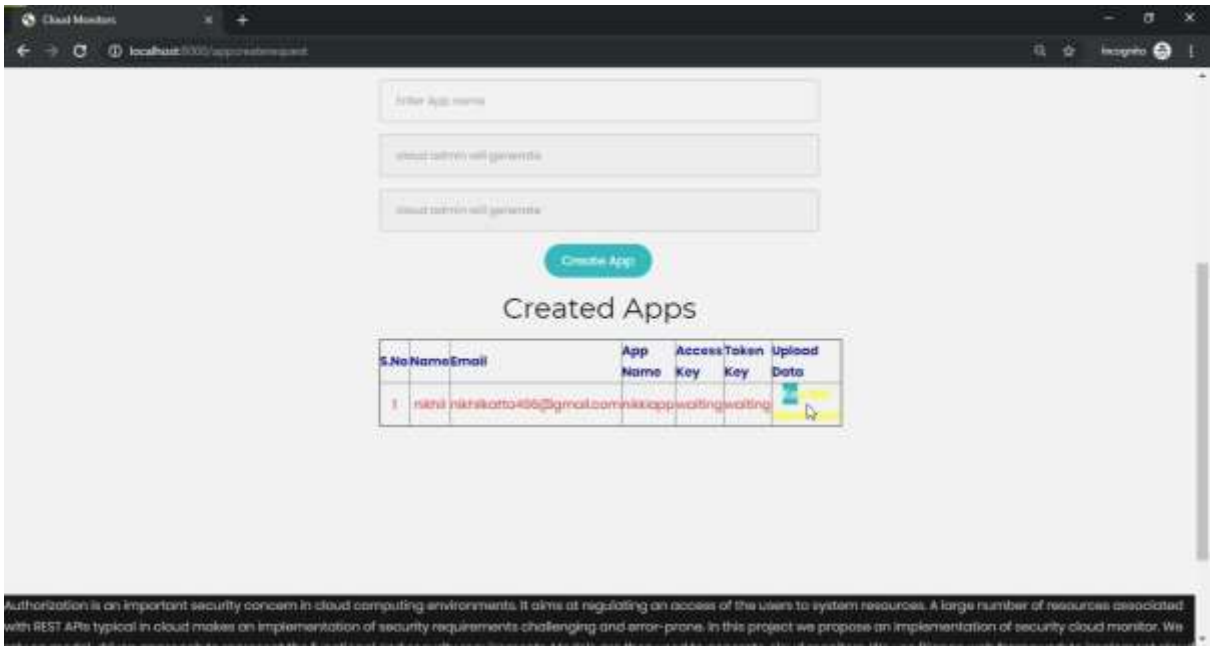

Fig 8 user check the security of the service app deployed on server

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 4, April : 2024

Fig 9:_ Cloud login form

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 4, April : 2024

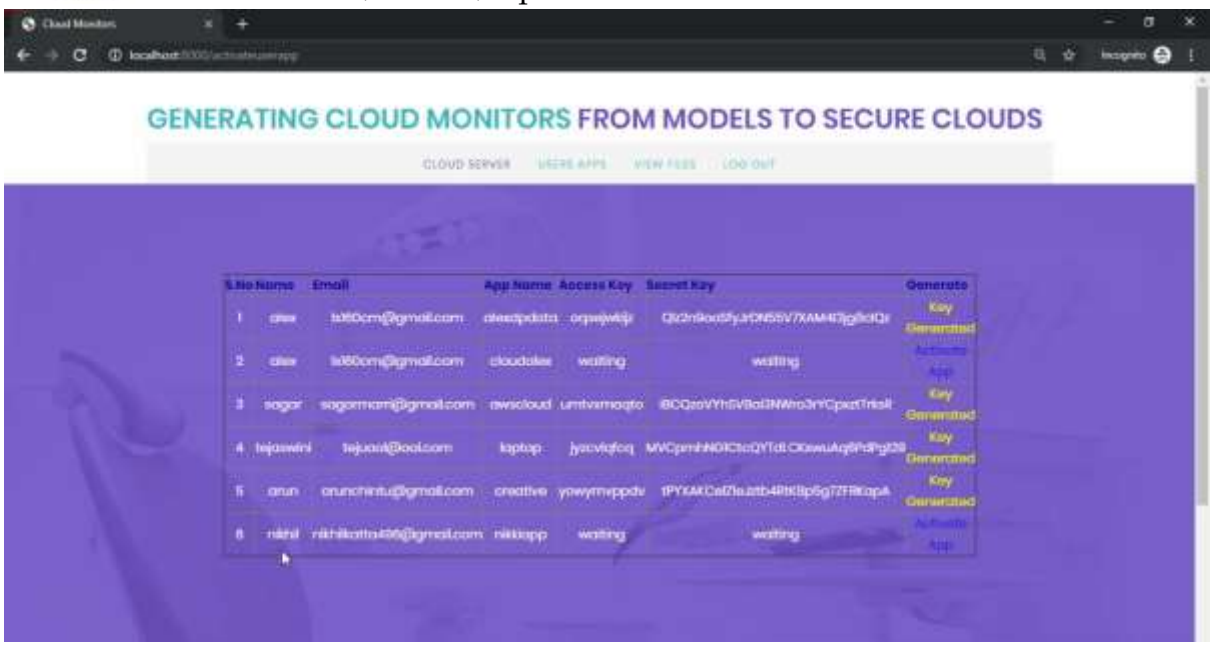

Fig 10;- Cloud approves the service app

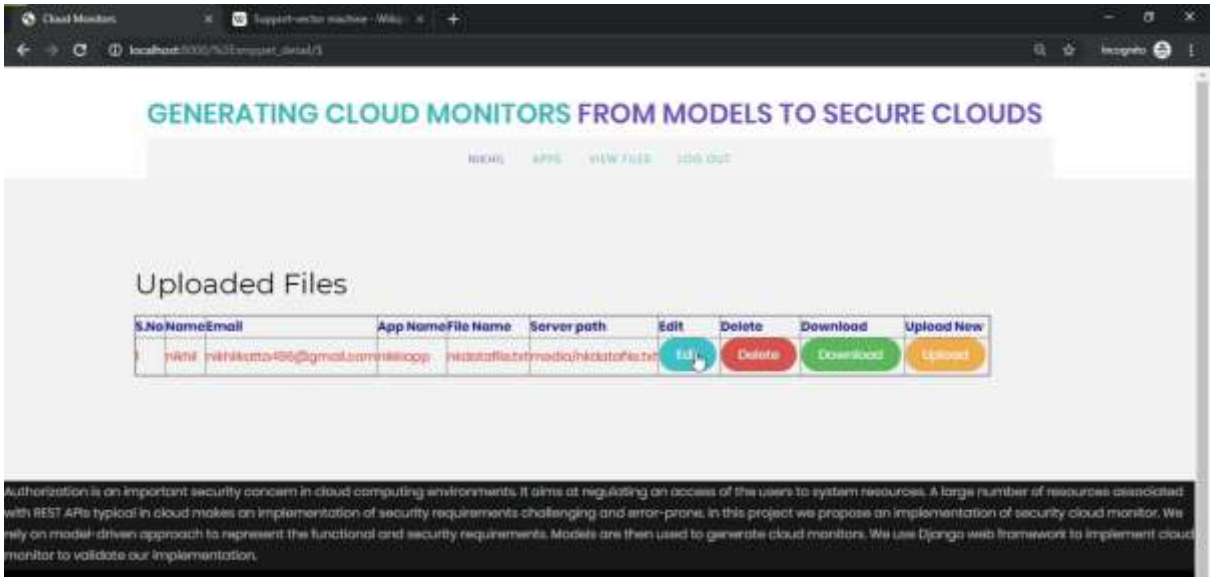

Fig 11:- User uploads file

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 53, Issue 4, April : 2024

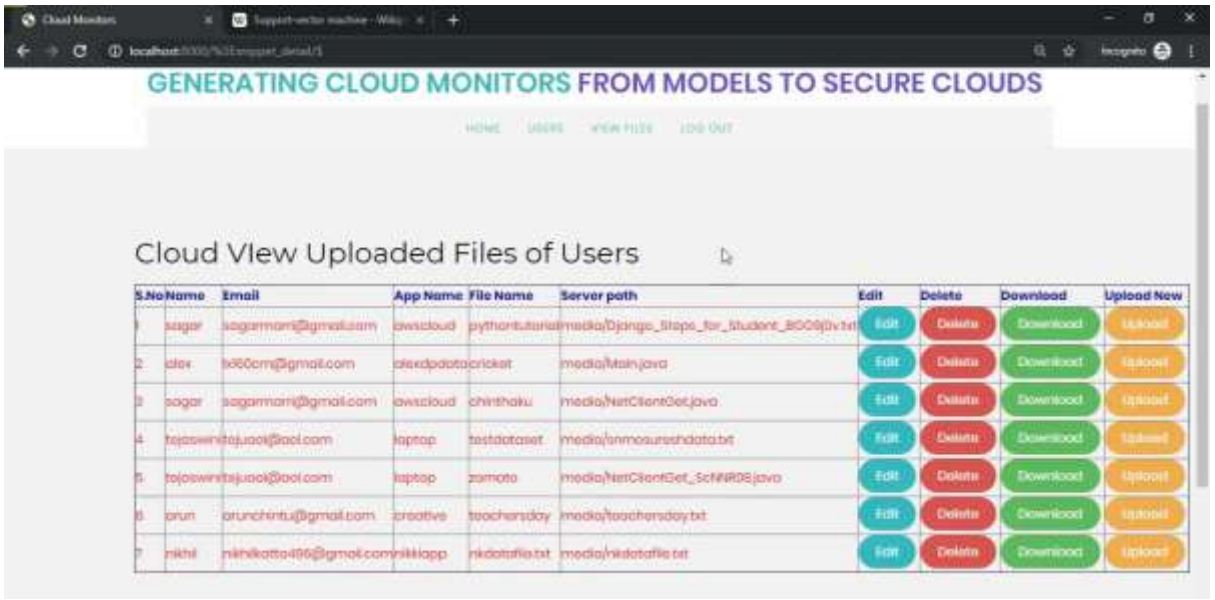

Fig 12:- cloud monitoring of the files uploaded

6 CONCLUSION AND FUTURE SCOPE

In this paper, we have presented an approach and associated tool for monitoring security in cloud. We have relied 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 modelling rather than manual code inspection or testing. It helps to spot the errors that might be exploited in data breaches or privilege escalation attacks. Since open source cloud frameworks usually undergo frequent changes, the automated nature of our approach allows the developers to relatively easily check whether functional and security requirements have been preserved in new releases.

7 REFERENCES

[1] Amazon Web Services. https://aws.amazon.com/. Accessed: 30.11.2017.

[2] Block Storage API V3 . https://developer.openstack.org/api-ref/ block-storage/v3/. retrieved: 126.2017.

[3] Cloud Computing Trends: 2017 State of the Cloud Survey. https://www. rightscale.com/blog/cloud-industry-insights/. Accessed: 30.11.2017.

[4] cURL. http://curl.haxx.se/. Accessed: 20.08.2013.

[5] Extensible markup language (xml). https://www.w3.org/XML/. Accessed: 27.03.2018.

[6] Keystone Security and Architecture Review. Online at https://www.openstack.org/summit/ openstack-summit-atlanta-2014/session-videos/presentation/keystonesecurity-and-architecture-review. retrieved: 06.2017.

[7] Nomagic MagicDraw. http://www.nomagic.com/products/magicdraw/. Accessed: 27.03.2018.

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

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

[10] urllib2 - extensible library for opening URLs. Python Documentation. Accessed: 18.10.2012.

[11] Windows Azure. https://azure.microsoft.com. Accessed: 30.11.2017. [

12] 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, 2004.

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

[14] Christopher Bailey et al. Run-time generation, transformation, and verification of access control models for self-protection. In Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, pages 135–144. ACM, 2014.

[15] Tim Berners-Lee et al. Hypertext transfer protocol–HTTP/1.0, 1996.

[16] Gaurav Bhatnagar and QMJ Wu. Chaos-based security solution for fingerprint data during communication and transmission. IEEE Transactions on Instrumentation and Measurement, 61(4):876–887, 2012. 10. Agarwal, P., et al. "Data-Driven Optimization of Spirulina Cultivation Using IoT and Genetic Algorithms." Bioprocess and Biosystems Engineering 44.11 (2021): 2151-2163. [7]Atallah, M. J., Blanton, M., & Fazio, N. (2005). Dynamic and efficient key management for access hierarchies. ACM Transactions on Information and System Security (TISSEC), 8(1), 1-39.

[8]Gartner. (2021). Magic Quadrant for Access Management.

[9]Jøsang, A., Keser, C., & Dimmock, N. (2007). The core of trust management. Journal of Trust Management, 1(1), 40-50.

[10]Yu, S., Wang, C., Ren, K., & Lou, W. (2007). Achieving secure, scalable, and fine-grained data access control in cloud computing. In IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications (pp. 46-54).