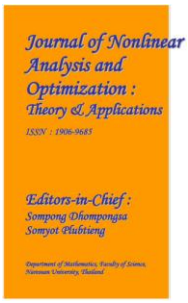
Journal of **N**onlinear **A**nalysis and **O**ptimization
Vol. 12, No. 2, (2021),
ISSN :**1906-9685**



SAFELY STORING DATA IN THE CLOUD AND CREATING CLOUDS FROM MODELS

VARUN MARAMRAJ,LAKKIREDDY PRIYANKA, Assistant Professor Department Of CSE
Sree Chaitanya College of Engineering, Karimnagar

ABSTRACT
One major security risk in cloud computing settings is authorization. Its goal is to control user access to system resources. The implementation of security standards is difficult and error-prone due to the abundance of resources connected with REST APIs, which are common in cloud computing environments.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.

I.INTRODUCTION
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 . According to the Cloud Survey 2017 ,private clouds are adopted by 72% of the cloud users, while the hybrid cloud adoption (both public and private) accounts for 67%. The companies, adopting private clouds, vary in size from 500 to more than 2000 employees. 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. 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, 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. It makes it rather unfeasible to manually check correctness of the APIs access control implementation and calls for enhanced monitoring mechanisms. In this paper, 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. 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. Moreover, 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 – and validated using OpenStack as a case study. OpenStack is an open source cloud computing framework providing IaaS (Infrastructure as a Service) . The validation using OpenStack has shown promising results and motivates us to continue the tool development described in this paper. The paper is organized as follows: section II motivates our work. Section III gives an overview of our cloud monitoring framework. In section IV, we present our design approach to modelling stateful REST services. The contract generation mechanism is described in section V. Section VI presents the tool architecture and our work with monitoring OpenStack. The related work and the conclusion are presented in sections VII and VIII correspondingly.

II.LITERATURE SURVEY

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) 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

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

4. Cloud computingthe 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.

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.

III.SYSTEM ANALYSIS

FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are,

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within

the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

IV.IMPLEMENTATION

MODULES DESCRIPTION

- User
- Cloud
- Admin
- Machine learning

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.

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.

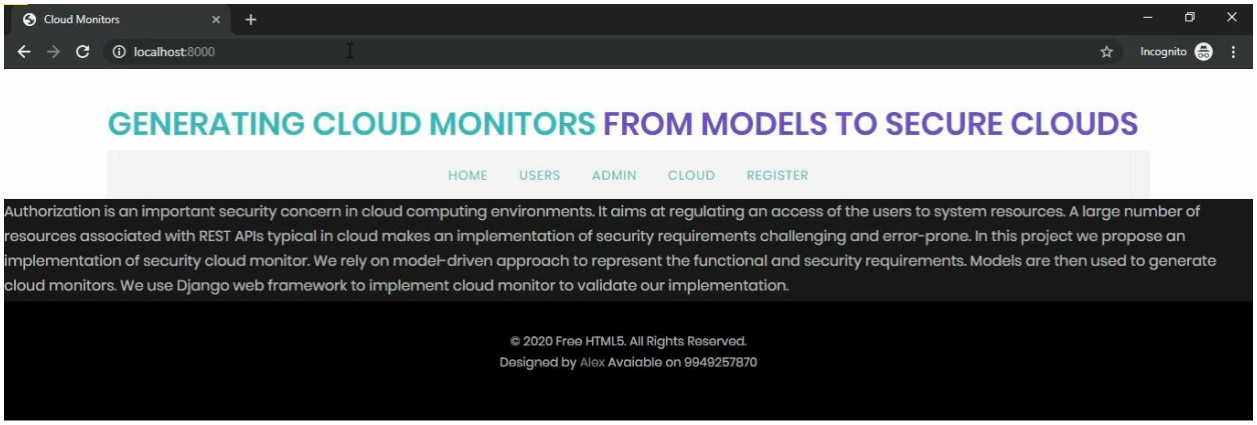
Admin

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.

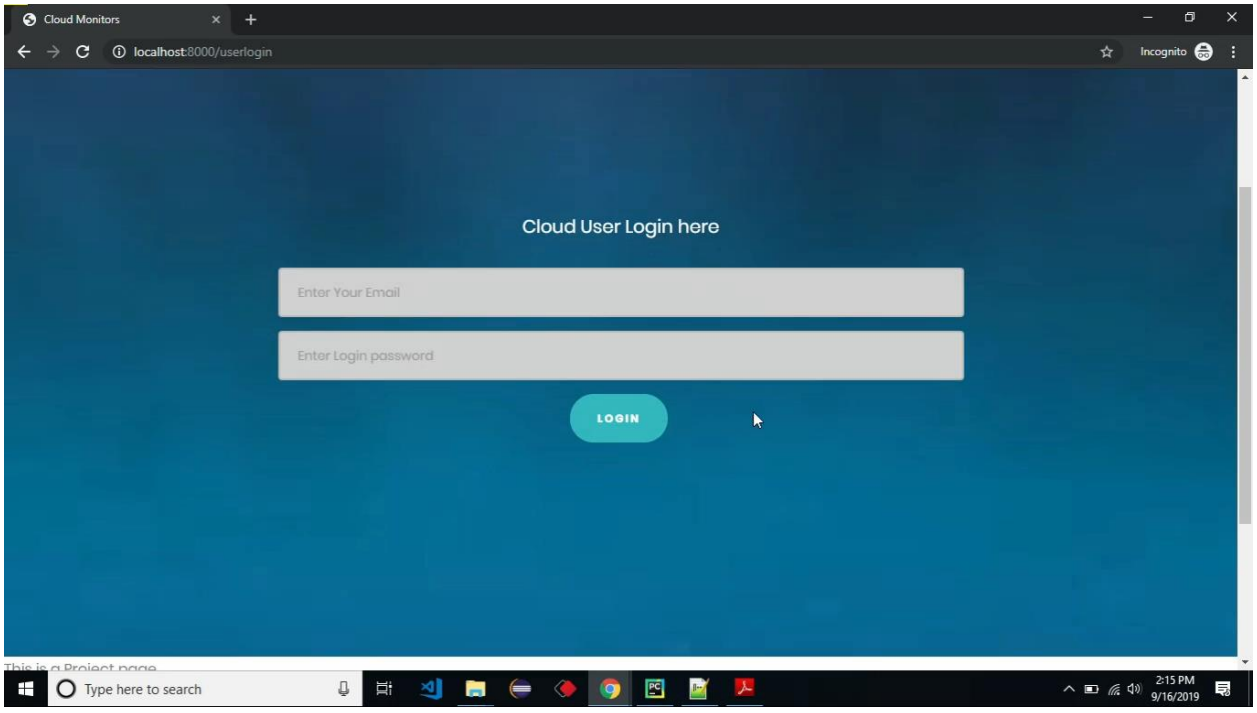
Machine learning

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.

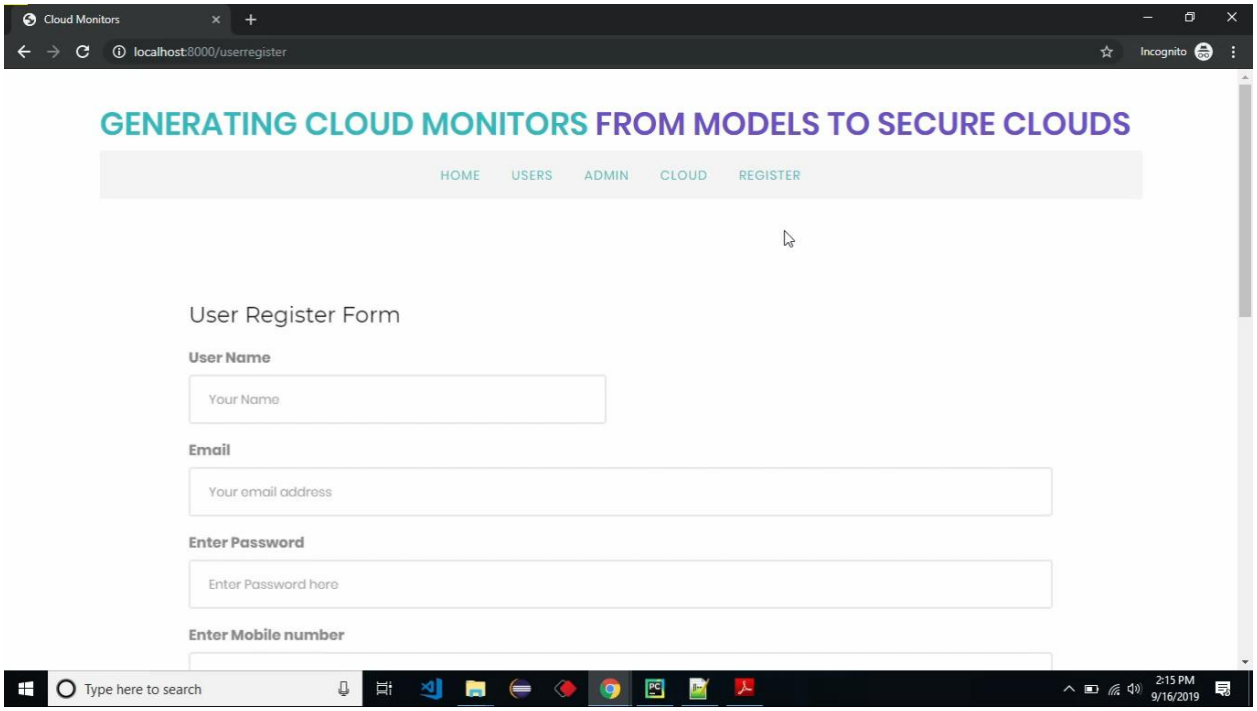
V.SCEEN SHOTS



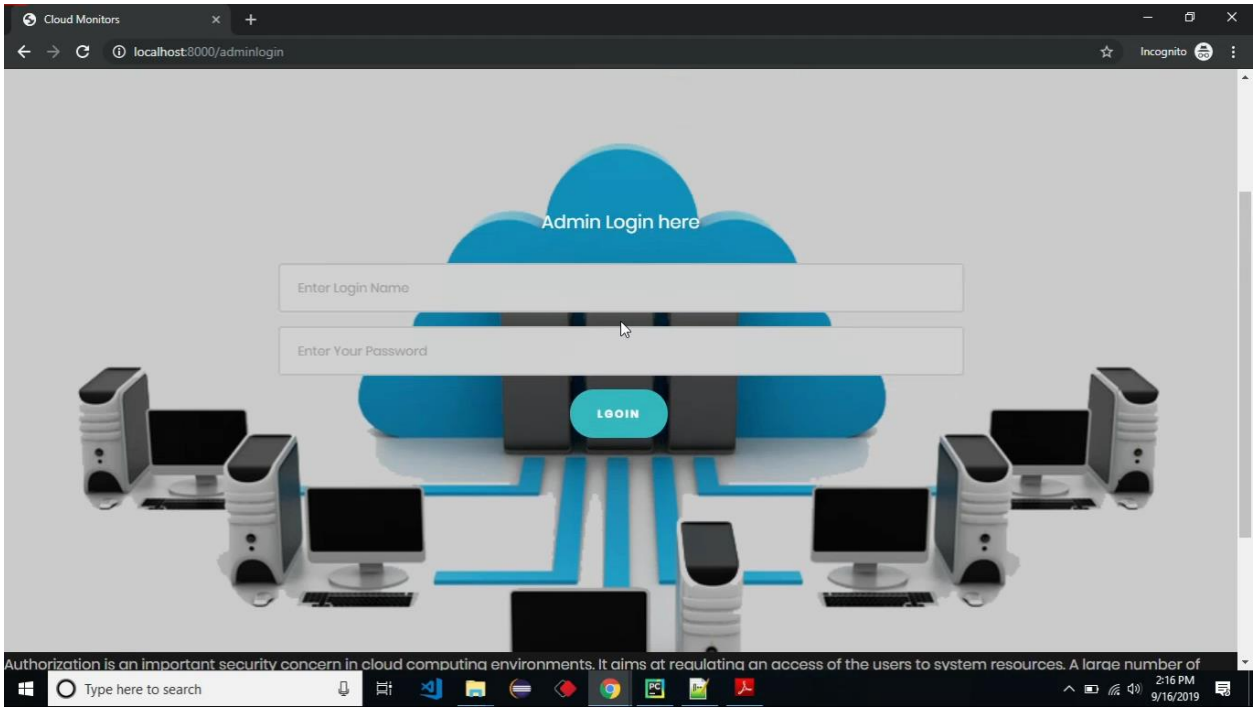
USER LOGIN

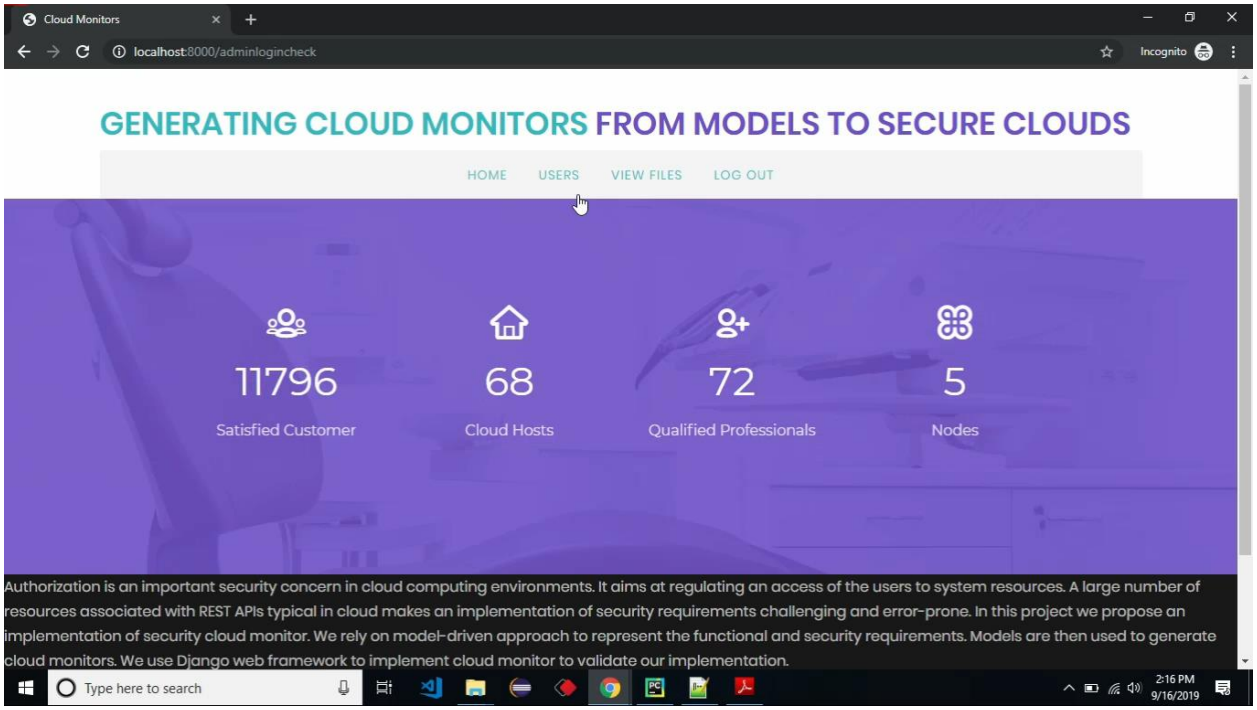


USER REGISTER

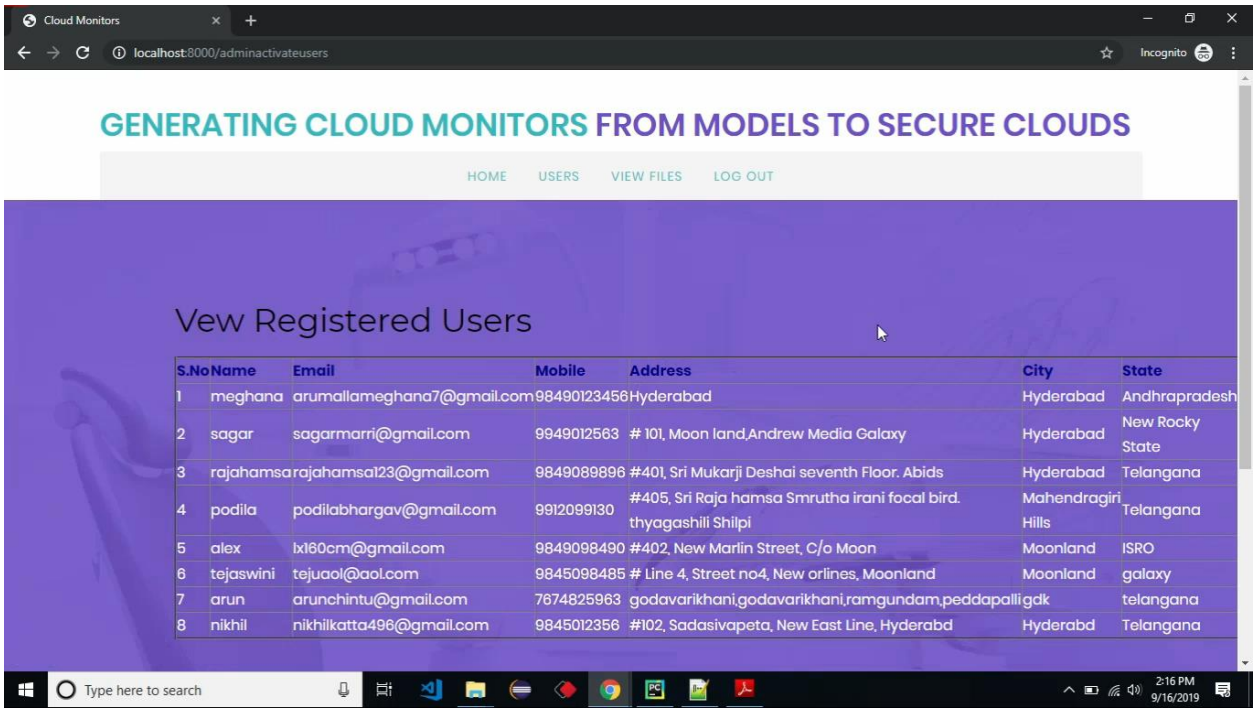


ADMIN LOGIN

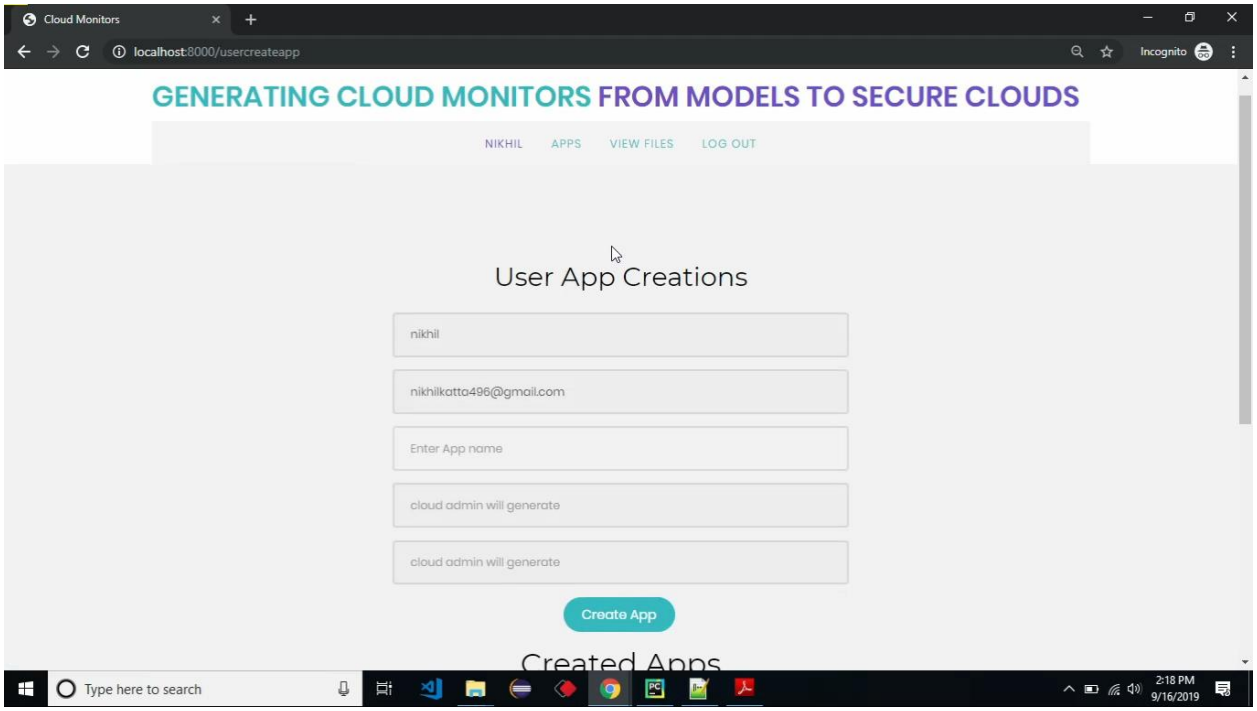




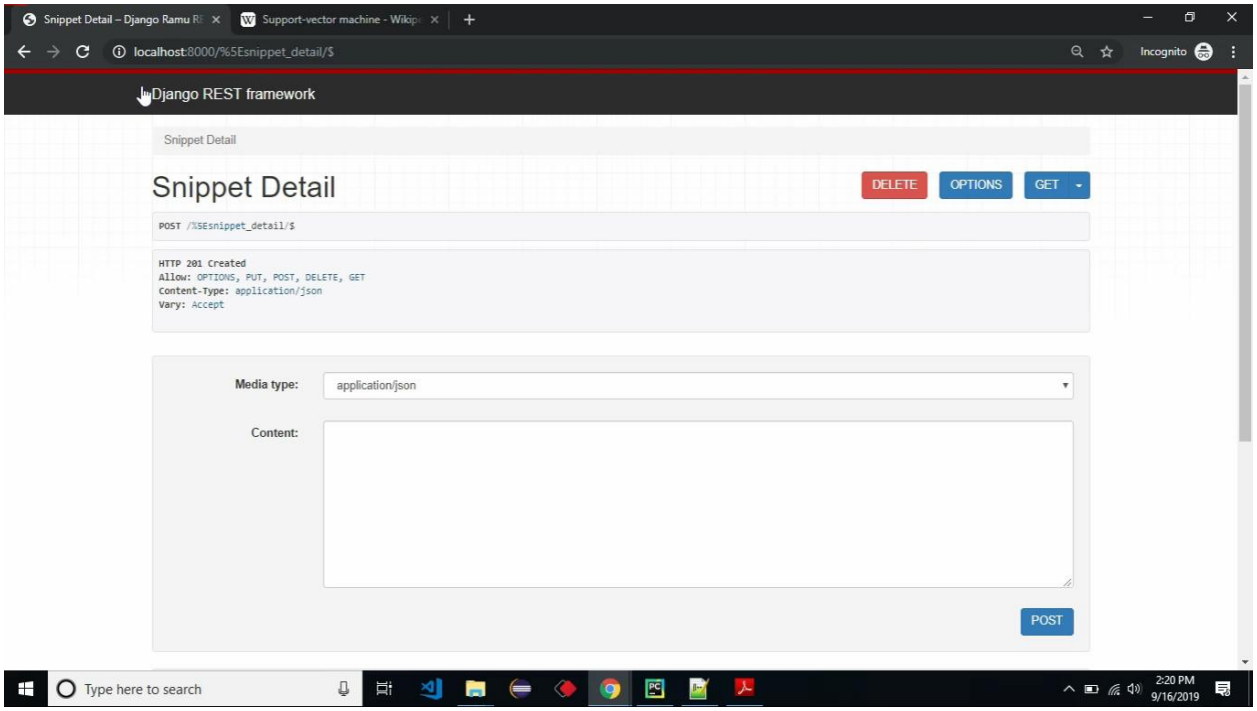
ADMIN APPROVE USER



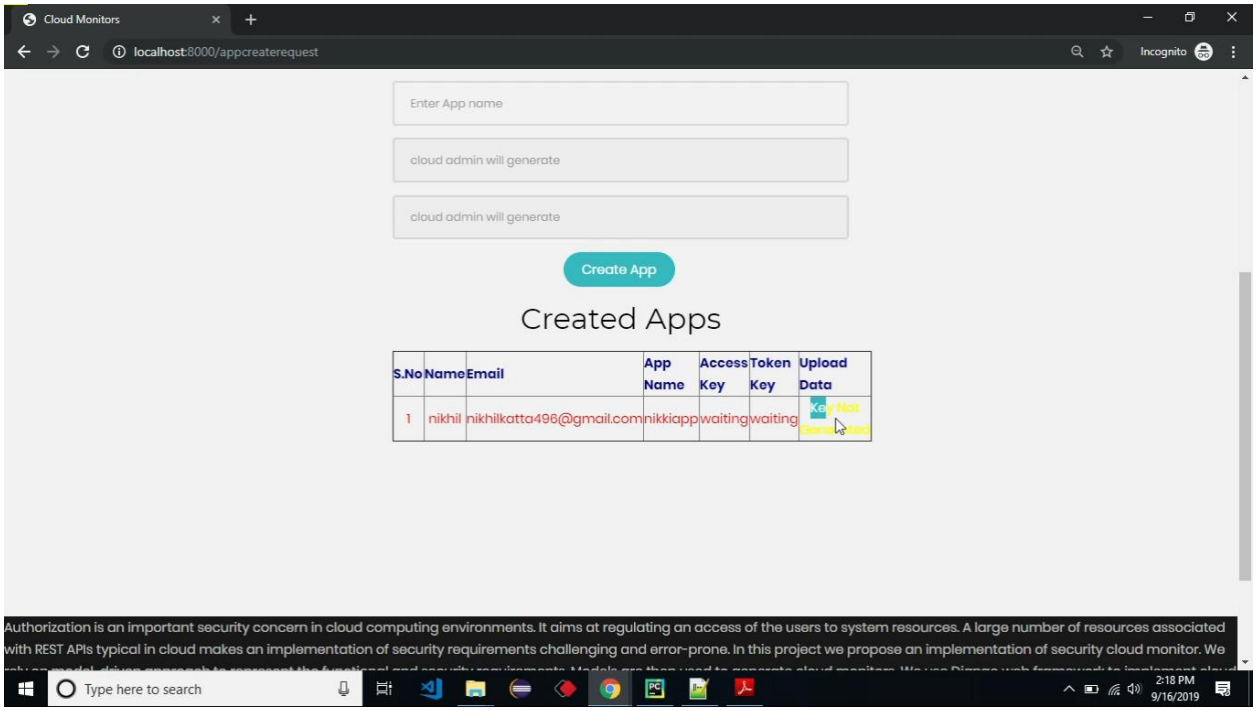
USER APP CREATION



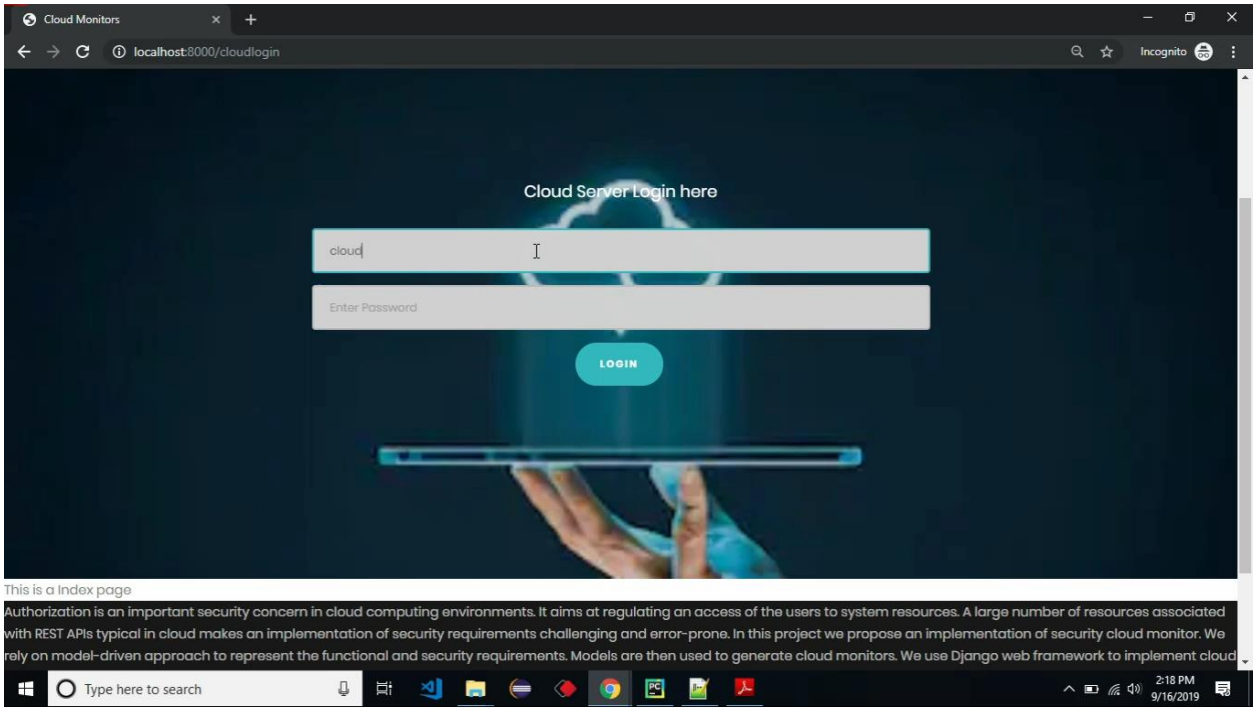
DJANGO REST



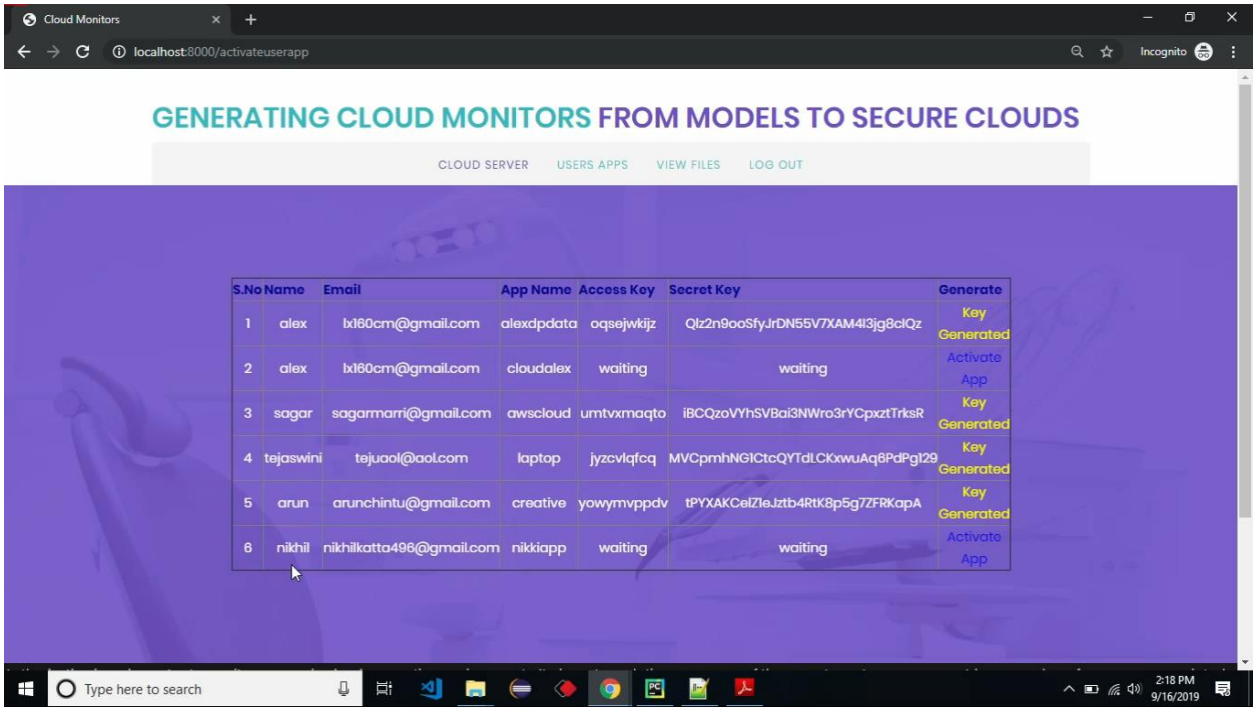
USER APP CHECK



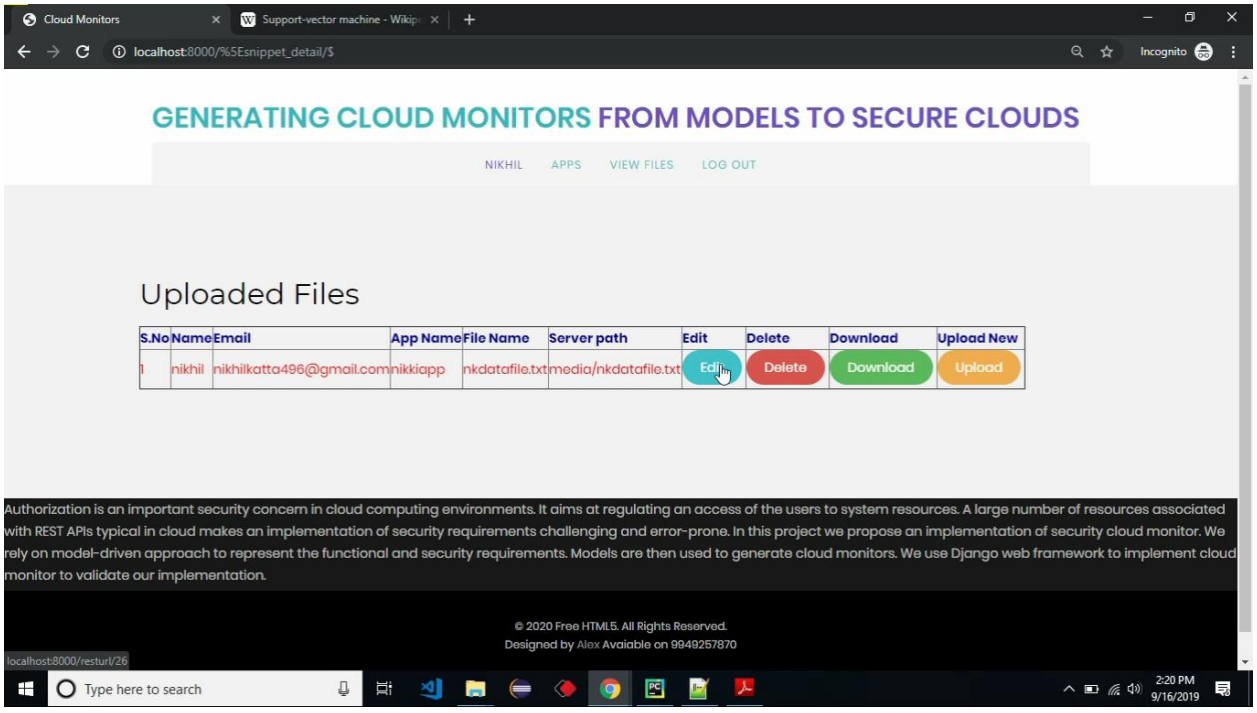
CLOUD LOGIN



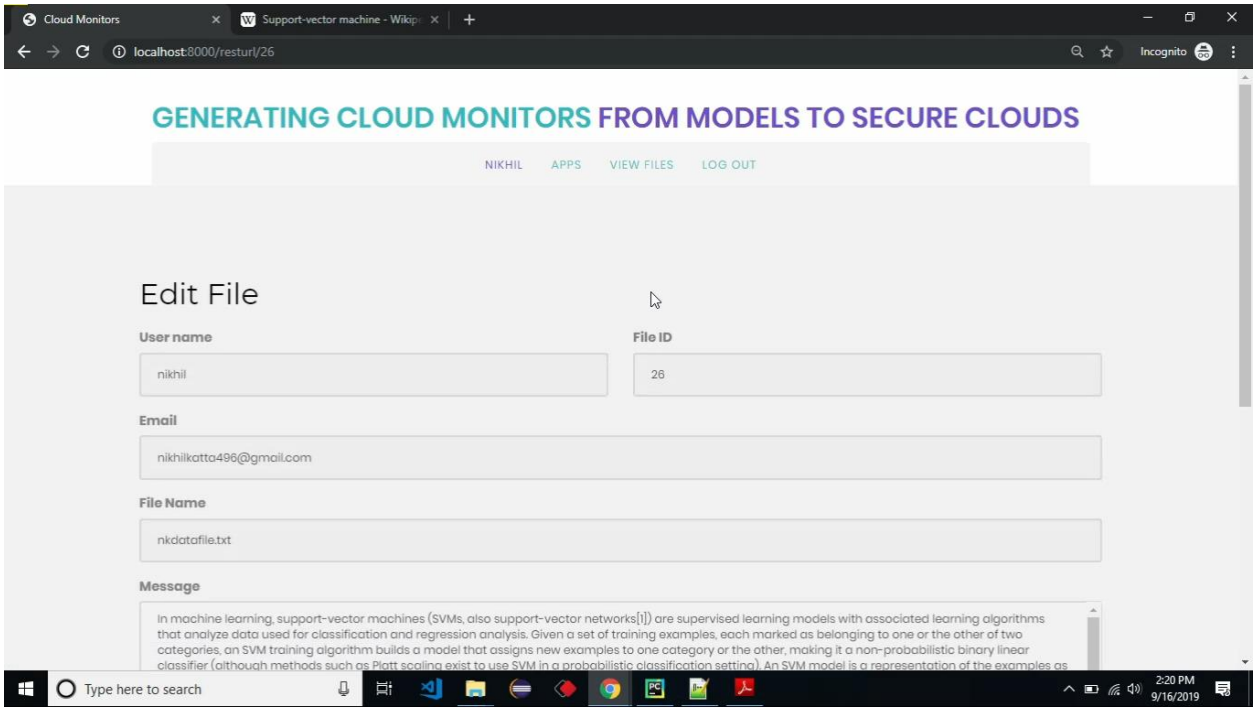
CLOUD APPROVE APP

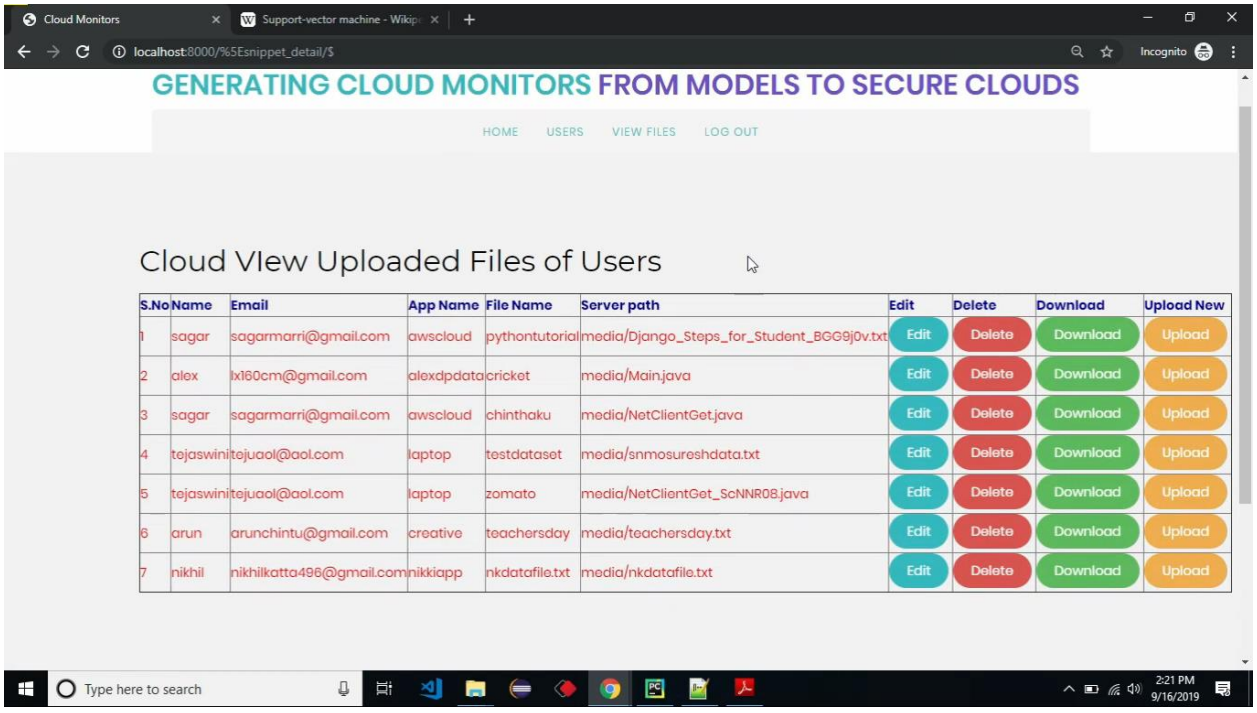


USER UPLOADED FILE



EDIT FILE





VI.CONCLUSIONS

We have described a method and related tool for cloud security monitoring in this paper. The model-driven approach has been our go-to method for creating APIs with REST interface characteristics. A private cloud infrastructure's automatic contract-based verification of the accuracy of functional and security requirements is made possible by the cloud monitors that are produced from the models. By using modeling instead of human code inspection or testing, the suggested semi-automated approach aims to assist cloud developers and security specialists in identifying security flaws in the implementation. Identifying mistakes that could be used as leverage in privilege escalation or data breach attempts is helpful. Since open source cloud frameworks are typically updated frequently, our automated method makes it quite easy for developers to verify that functional and security criteria are maintained in new releases.

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.
[17] David Ferraiolo et al. Role-based access control (rbac): Features and motivations. In Proceedings of 11th annual computer security application conference, pages 241–48, 1995.
[18] Django Software Foundation. Django Documentation. Online Documentation of Django 2.0, 2017. https://docs.djangoproject.com/en/2.0/.
[19] Michal Gordon and David Harel. Generating executable scenarios from natural language. In International Conference on Intelligent Text Processing and Computational Linguistics. Springer, 2009.
[20] Robert L Grossman. The case for cloud computing. IT professional, 11(2):23–27, 2009.
[21] A. Holovaty and J. Kaplan-Moss. The Django Book. Online version of The Django Book, 2010. http://docs.djangoproject.com/en/1.2/.
[22] Adrian Holovaty and Jacob Kaplan-Moss. The definitive guide to Django: Web development done right. Apress, 2009.
[23] Jan J¨urjens. Towards development of secure systems using umlsec. In International Conference on Fundamental Approaches to Software Engineering, pages 187–200. Springer, 2001.
[24] Nesrine Kaaniche et al. Security SLA based monitoring in clouds. In Edge Computing (EDGE), 2017 IEEE International Conference on, pages 90–97. IEEE, 2017.
[25] Ronald L Krutz and Russell Dean Vines. Cloud security: A comprehensive guide to secure cloud computing. Wiley Publishing, 2010.
[26] Marc Lohmann et al. A model-driven approach to discovery, testing and monitoring of web services. In Test and Analysis of Web Services, pages 173–204. Springer, 2007.
[27] Sean Marston et al. Cloud computingthe business perspective. Decision support systems, 51(1):176–189, 2011.
[28] Bertrand Meyer. Applying'design by contract'. Computer, 25(10), 1992. [29] Dejan Miloji˘ ci´ c et al. Opennebula: A cloud management tool. IEEE Internet Computing, 15(2):11–14, 2011.

[30] PhuHNguyenetal. Anextensivesystematicreviewonthemodel-driven development of secure systems. Information and Software Technology, 68:62–81, 2015.
[31] OMG. OCL, OMG Available Specification, Version 2.0, 2006.
[32] Alina Oprea et al. Mosaic: A platform for monitoring and security analytics in public clouds. In Cybersecurity Development (SecDev), IEEE, pages 69–70. IEEE, 2016.
[33] Ivan Porres and Irum Rauf. From nondeterministic uml protocol statemachines to class contracts. In Software Testing, Verification and Validation (ICST), 2010 Third International Conference on, pages 107– 116. IEEE, 2010.
[34] Irum Rauf and Ivan Porres. Beyond crud. In REST: From Research to Practice, pages 117–135. Springer, 2011.
[35] Irum Rauf and Ivan Porres. Designing Level3 behavioral REST web service interfaces. ACM Applied Computing Review, 11(3), 2011.
[36] Mehmet Sahinoglu. An input–output measurable design for security meter model to quantify and manage software security risk. IEEE Transactions on Instrumentation and Measurement, 57(6), 2008.
[37] Omar Sefraoui et al. Openstack: toward an open-source solution for cloud computing. International Journal of Computer Applications, 55(3), 2012.
[38] OMG UML. 2.4. 1 superstructure specification. Technical report, document formal/2011-08-06. Technical report, OMG, 2011.
[39] Jim Webber et al. REST in Practice: Hypermedia and Systems Architecture. O'Reilly Media, Inc., 2010.