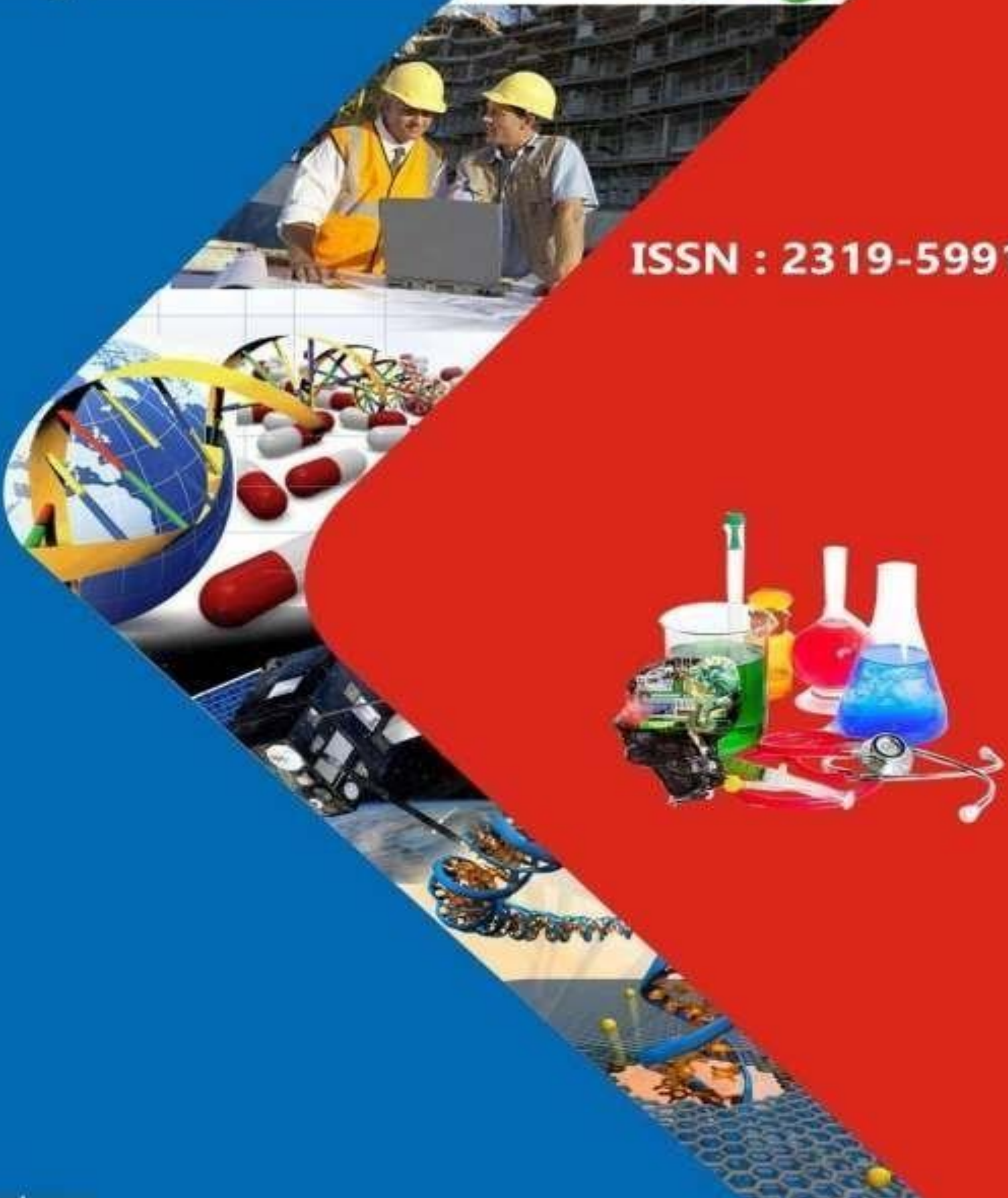


ISSN : 2319-5991

ISSN 2319-5991 www.ijerst.com

Vol. 11, Issue 4, 2018

MODEL-BASED CLOUD GENERATION: A NOVEL APPROACH FOR SECURING CLOUD STORAGE

[1]Chekka Sivanya,[2]Jellapuram Roja,[3]Mutyala Malini,[4]Kshirsagar Vaishnavi

[123]Assistant Professor,[4]Student

Department of CSE

Abdul Kalam Institute of Technological Sciences, Kothagudem, Telangana

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

ISSN 2319-5991 www.ijerst.com

Vol. 11, Issue 4, 2018

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

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

ISSN 2319-5991 www.ijerst.com

Vol. 11, Issue 4, 2018

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

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.

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

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

ISSN 2319-5991 www.ijerst.com

Vol. 11, Issue 4, 2018

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

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. 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

ISSN 2319-5991 www.ijerst.com

Vol. 11, Issue 4, 2018

piece of information with a URI, which results in a large number of URIs that can access the system.

DISADVANTAGES OF EXISTING 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.

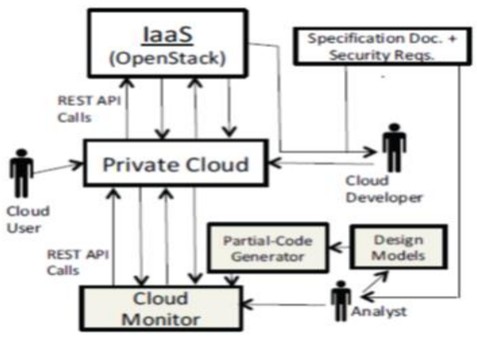
IV. PROPOSED SYSTEM:

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 functionalrequirements as verifiable contracts.

ADVANTAGES OF PROPOSED SYSTEM:

- 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.

V. SYSTEM ARCHITECTURE:



ISSN 2319-5991 www.ijerst.com

Vol. 11, Issue 4, 2018

VI.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.

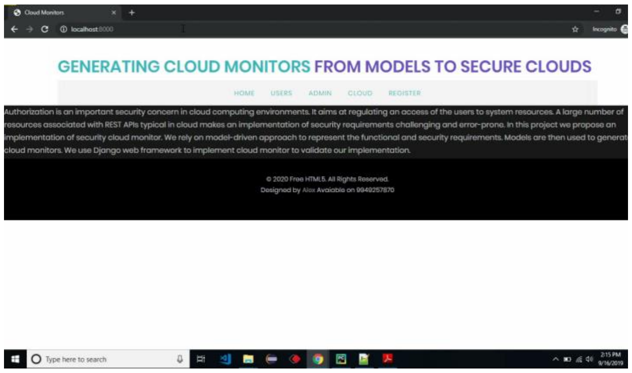
Machine learning

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.

ISSN 2319-5991 www.ijerst.com

Vol. 11, Issue 4, 2018

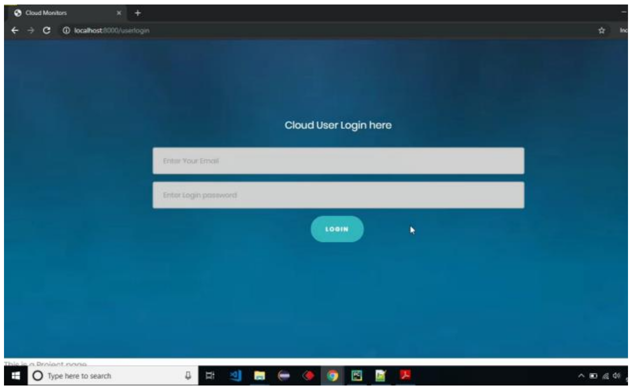
VII .SCEEN SHOTS


GENERATING CLOUD MONITORS FROM MODELS TO SECURE CLOUDS

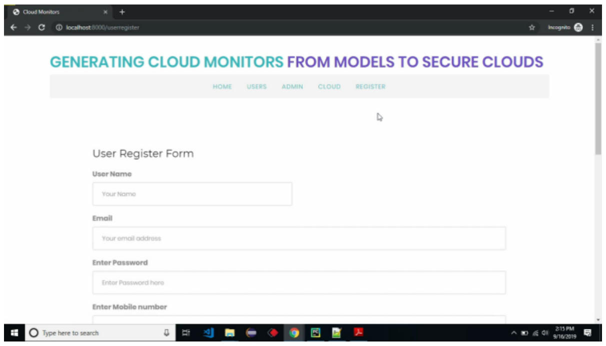
Authorization is an important security concern in cloud computing environments. It aims at regulating an access of the users to system resources. A large number of resources associated with REST APIs typical in cloud makes an implementation of security requirements challenging and error-prone. In this project we propose an implementation of security cloud monitor. We rely on model-driven approach to represent the functional and security requirements. Models are then used to generat cloud monitors. We use Django web framework to implement cloud monitor to validate our implementation.

© 2020 Free HTML5. All Rights Reserved.
Designed by Alex Avaiable on 9949257870

USER LOGIN

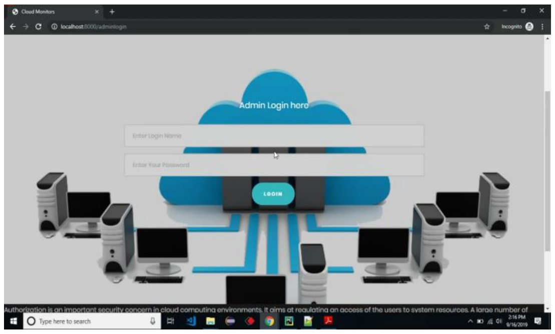


USER REGISTER

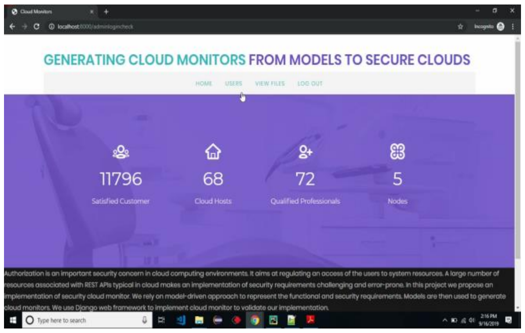

GENERATING CLOUD MONITORS FROM MODELS TO SECURE CLOUDS

User Register Form

ADMIN LOGIN

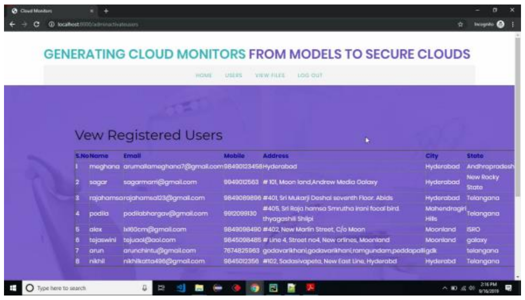


Authorization is an important security concern in cloud computing environments. It aims at regulating an access of the users to system resources. A large number of


GENERATING CLOUD MONITORS FROM MODELS TO SECURE CLOUDS

Authorization is an important security concern in cloud computing environments. It aims at regulating an access of the users to system resources. A large number of resources associated with REST APIs typical in cloud makes an implementation of security requirements challenging and error-prone. In this project we propose an implementation of security cloud monitor. We rely on model-driven approach to represent the functional and security requirements. Models are then used to generate cloud monitors. We use Django web framework to implement cloud monitor to validate our implementation.

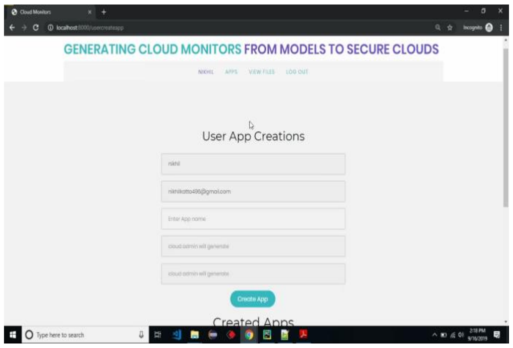
ADMIN APPROVE USER


GENERATING CLOUD MONITORS FROM MODELS TO SECURE CLOUDS

Vew Registered Users

S.No	Name	Email	Mobile	Address	City	State
1	meghana	arumallameghana7@gmail.com	98490123456	Hyderabad	Hyderabad	Andhrapradesh
2	sagar	sagarmani@gmail.com	9949012563	# 101, Moon land,Andrew Media Galaxy	Hyderabad	New Rocky State
3	rajahamsa	rajahamsa123@gmail.com	9849089896	#401, Sri Mukarji Deshai seventh Floor. Abids	Hyderabad	Telangana
4	podila	podilabhargav@gmail.com	9912099130	#405, Sri Raja hamsa Smrutha Irani focal bird. thyagashili Shilpi	Mahendragiri Hills	Telangana
5	alex	lxl60cm@gmail.com	9849098490	#402, New Marlin Street, C/o Moon	Moonland	ISRO
6	tejaswini	tejuaol@aol.com	9845098485	# Line 4, Street no4, New orlines, Moonland	Moonland	galaxy
7	arun	arunchintu@gmail.com	7674825963	godavarikhani,godavarikhani,ramgundam,peddapalli	gdk	telangana
8	nikhil	nikhilkatta496@gmail.com	9845012356	#102, Sadasivapeta, New East Line, Hyderabd	Hyderabd	Telangana

USER APP CREATION

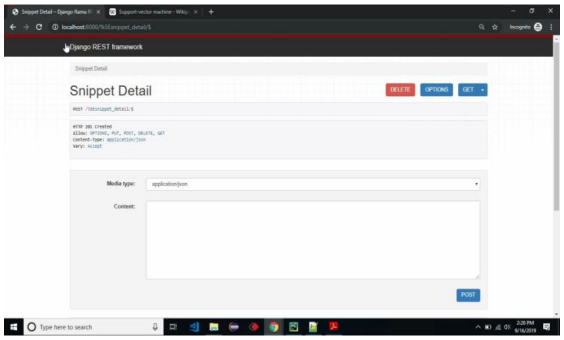

GENERATING CLOUD MONITORS FROM MODELS TO SECURE CLOUDS

User App Creations

DJANGO REST

ISSN 2319-5991 www.ijerst.com

Vol. 11, Issue 4, 2018

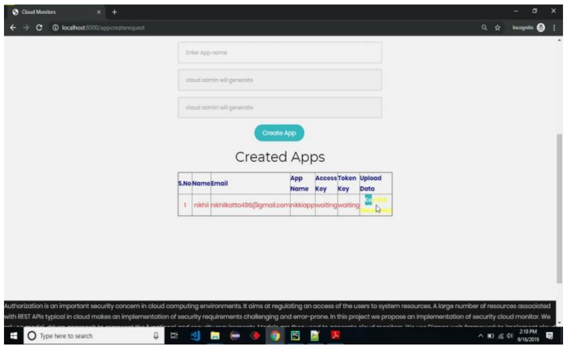


Snippet Detail

```
POST /%5Esnippet_detail/$

HTTP 201 Created
Allow: OPTIONS, PUT, POST, DELETE, GET
Content-Type: application/json
Vary: Accept
```

USER APP CHECK

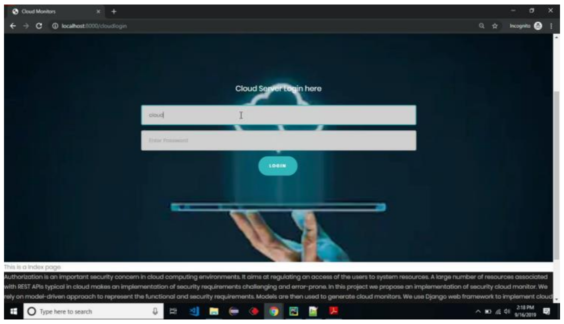

Cloud Monitors
localhost:8000/appcreaterequest
Enter App name
cloud admin will generate
cloud admin will generate
Create App
Type here to search

Created Apps

S.No	Name	Email	App Name	Access Key	Token Key	Upload Data
1	nikhil	nikhilkatta496@gmail.com	nikkiapp	waiting	waiting	[illegible]

Authorization is an important security concern in cloud computing environments. It aims at regulating an access of the users to system resources. A large number of resources associated with REST APIs typical in cloud makes an implementation of security requirements challenging and error-prone. In this project we propose an implementation of security cloud monitor. We

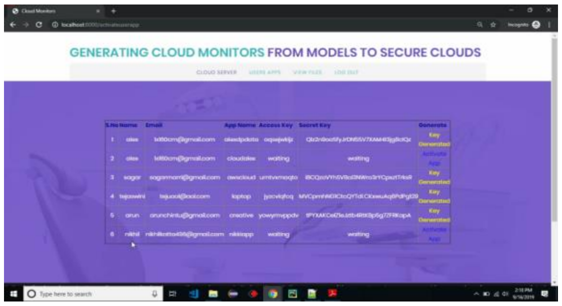
CLOUD LOGIN



This is a index page

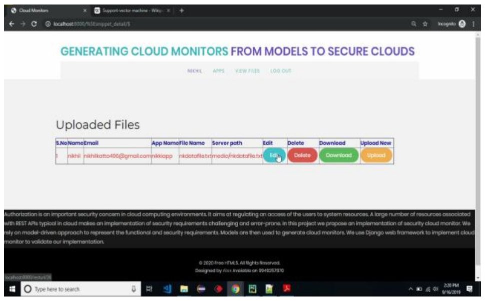
Authorization is an important security concern in cloud computing environments. It aims at regulating an access of the users to system resources. A large number of resources associated with REST APIs typical in cloud makes an implementation of security requirements challenging and error-prone. In this project we propose an implementation of security cloud monitor. We rely on model-driven approach to represent the functional and security requirements. Models are then used to generate cloud monitors. We use Django web framework to implement cloud

CLOUD APPROVE APP


Cloud Monitors
localhost:8000/activateuserapp
GENERATING CLOUD MONITORS FROM MODELS TO SECURE CLOUDS
CLOUD SERVER
USERS APPS
VIEW FILES
LOG OUT
Type here to search

S.No	Name	Email	App Name	Access Key	Secret Key	Generate
1	alex	lx180cm@gmail.com	alexdpdata	oqsejwkljz	[illegible]	Key Generated
2	alex	lx180cm@gmail.com	cloudalex	waiting	waiting	Activate App
3	sagar	sagarmani@gmail.com	awscloud	umtvemoqto	[illegible]	Key Generated
4	tejaswini	tejuaol@aol.com	laptop	jyacvlqfoq	[illegible]	Key Generated
5	arun	arunchintu@gmail.com	creative	yowymvppdv	[illegible]	Key Generated
6	nikhil	nikhilkatta496@gmail.com	nikkiapp	waiting	waiting	Activate App

USER UPLOADED FILE


Cloud Monitors
Support-vector machine - Wikip
localhost:8000/%5Esnippet_detail/$
GENERATING CLOUD MONITORS FROM MODELS TO SECURE CLOUDS
NIKHIL
APPS
VIEW FILES
LOG OUT
Type here to search

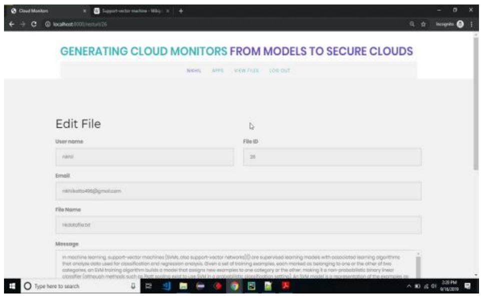
Uploaded Files

S.No	Name	Email	App Name	File Name	Server path	Edit	Delete	Download	Upload New
1	nikhil	nikhilkatta496@gmail.com	nikkiapp	nkdatafile.txt	media/nkdatafile.txt	Edit	Delete	Download	Upload

Authorization is an important security concern in cloud computing environments. It aims at regulating an access of the users to system resources. A large number of resources associated with REST APIs typical in cloud makes an implementation of security requirements challenging and error-prone. In this project we propose an implementation of security cloud monitor. We rely on model-driven approach to represent the functional and security requirements. Models are then used to generate cloud monitors. We use Django web framework to implement cloud monitor to validate our implementation.

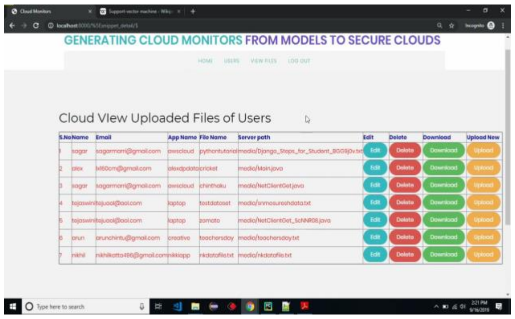
© 2020 Free HTML5. All Rights Reserved.
Designed by Alex Available on 9949257870

EDIT FILE


Cloud Monitors
Support-vector machine - Wikip
localhost:8000/restuit/26
GENERATING CLOUD MONITORS FROM MODELS TO SECURE CLOUDS
NIKHIL
APPS
VIEW FILES
LOG OUT
Type here to search

Edit File

User name: nikhil
File ID: 26
Email: nikhilkatta496@gmail.com
File Name: nkdatafile.txt
Message: In machine learning, support-vector machines (SVMs, also support-vector networks[1]) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). An SVM model is a representation of the examples as


Cloud Monitors
Support-vector machine - Wikip
localhost:8000/%5Esnippet_detail/$
GENERATING CLOUD MONITORS FROM MODELS TO SECURE CLOUDS
HOME
USERS
VIEW FILES
LOG OUT
Type here to search

Cloud VIew Uploaded Files of Users

S.No	Name	Email	App Name	File Name	Server path	Edit	Delete	Download	Upload New
1	sagar	sagarmani@gmail.com	awscloud	pythontutorial	media/Django_Steps_for_Student_BGG8j0v.txt	Edit	Delete	Download	Upload
2	alex	lx180cm@gmail.com	alexdpdata	cricket	media/Main.java	Edit	Delete	Download	Upload
3	sagar	sagarmani@gmail.com	awscloud	chinthaku	media/NetClientGet.java	Edit	Delete	Download	Upload
4	tejaswini	tejuaol@aol.com	laptop	testdataset	media/snmosureshdata.txt	Edit	Delete	Download	Upload
5	tejaswini	tejuaol@aol.com	laptop	zomato	media/NetClientGet_5cNNR08.java	Edit	Delete	Download	Upload
6	arun	arunchintu@gmail.com	creative	teachersday	media/teachersday.txt	Edit	Delete	Download	Upload
7	nikhil	nikhilkatta496@gmail.com	nikkiapp	nkdatafile.txt	media/nkdatafile.txt	Edit	Delete	Download	Upload

VIII.CONCLUSIONS

Throughout this article, we have described a method and related tool for monitoring the security of cloud computing environments. To create application programming interfaces (APIs) that display REST interface capabilities, we have relied on the model-driven method. An automatic contract-based verification of the validity of functional and

ISSN 2319-5991 www.ijerst.com

Vol. 11, Issue 4, 2018

security criteria is made possible by the cloud monitors, which are derived from the models. These monitors are implemented by a private cloud infrastructure. Rather than depending on human code inspection or testing, the semi-automated technique that was offered was intended to assist cloud developers and security specialists in identifying security flaws in the implementation. This was accomplished via the use of modeling. By doing so, faults that might potentially be exploited in data breaches or privilege escalation attacks can be identified and identified. As a result of the automated nature of our method, developers are able to verify with relative simplicity whether the functional and security criteria have been maintained in new versions. This is because open source cloud frameworks often go through rapid modifications.

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.

ISSN 2319-5991 www.ijerst.com

Vol. 11, Issue 4, 2018

[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.