## Contents

[Contents i](#_Toc482703964)

[List of Figures iii](#_Toc482703965)

[Abstract v](#_Toc482703966)

[Chapter 1 6](#_Toc482703967)

[Introduction 6](#_Toc482703968)

[1.2 Virtual Machine Introspection for security 7](#_Toc482703969)

[1.3 Support Vector Machine 8](#_Toc482703970)

[1.4 Summary 8](#_Toc482703971)

[Chapter 2 9](#_Toc482703972)

[Literature Review 9](#_Toc482703973)

[2.1 Intrusion detection system 9](#_Toc482703974)

[2.2 Novelties in cloud thread models 10](#_Toc482703975)

[2.3 Aim, Objectives and Methodology 14](#_Toc482703976)

[2.4 Summary 15](#_Toc482703977)

[Chapter 3 16](#_Toc482703978)

[Design 16](#_Toc482703979)

[3.1 Design Procedure 16](#_Toc482703980)

[3.2 System Architecture 16](#_Toc482703981)

[3.3 UML 18](#_Toc482703982)

[3.4 Summary 27](#_Toc482703983)

[Chapter 4 28](#_Toc482703984)

[Implementation 28](#_Toc482703985)

[4.1 Cloud Server 28](#_Toc482703986)

[4.2 Service Provider 33](#_Toc482703987)

[4.3 End User 34](#_Toc482703988)

[Chapter 5 38](#_Toc482703989)

**Testing …………………………………………………………………….**38

[5.1Introduction of Testing 38](#_Toc482703990)

[5.2 Software Testing. 38](#_Toc482703991)

[5.3 System Testing 41](#_Toc482703992)

[Chapter 6 43](#_Toc482703993)

[Results and Discussions 43](#_Toc482703994)

[Chapter 7 50](#_Toc482703995)

[Conclusion and Future Work 50](#_Toc482703996)

[7.1 Conclusion 50](#_Toc482703997)

[7.2 Future Work 50](#_Toc482703998)

[References 51](#_Toc482703999)

## List of Figures

[Figure 3.1 System Architecture 17](file:///C:\Users\Rakhi%20Chadaga\Desktop\print.docx#_Toc482708319)

[Figure 3.2 use case diagram of service provider 20](#_Toc482708320)

[Figure 3.3 use case for end user 21](#_Toc482708322)

[Figure 3.4 sequence diagram 22](file:///C:\Users\Rakhi%20Chadaga\Desktop\print.docx#_Toc482708323)

[Figure 3.5 class diagram 24](file:///C:\Users\Rakhi%20Chadaga\Desktop\print.docx#_Toc482708324)

[Figure 3.6 Activity diagram for cloud server 25](#_Toc482708325)

[Figure 3.7 Activity diagram for end user 26](#_Toc482708326)

[Table 5‑1: Test cases for the project 39](#_Toc482708327)

[Table 5‑2: Test cases for Integration testing 40](#_Toc482708328)

[Table 5‑3: Test cases for Input-Output 42](#_Toc482708329)

[Figure 6.1 Home page 43](#_Toc482708330)

[Figure 6.2 Cloud server login 44](#_Toc482708331)

[Figure 6.3 Registration. 44](#_Toc482708332)

[Figure 6.4 Service Provider Registration 45](#_Toc482708333)

[Figure 6.5 File upload 45](#_Toc482708335)

[Figure 6.6 File request 46](#_Toc482708336)

[Figure 6.7 Block user 46](#_Toc482708337)

[Figure 6.8 Send alert 47](#_Toc482708338)

[Figure 6.9 View alert 47](#_Toc482708339)

[Figure 6.10 Request secret key 48](#_Toc482708340)

[Figure 6.11 Download file 48](#_Toc482708341)

[Figure 6.12 Warning message 49](#_Toc482708342)

[Figure 6.13 Pie chart depicting the result 49](#_Toc482708343)

List of Tables

[Table 5‑1: Test cases for the project 39](#_Toc482707774)

[Table 5‑2: Test cases for Integration testing 40](#_Toc482707775)

[Table 5‑3: Test cases for Input-Output 42](#_Toc482707776)

## 

## Abstract

Cloud services are prominent within the private, public and commercial domains. Many of these services are expected to always have a critical nature, there foresecurity and resilience are increasingly important aspects. In order to remain resilient, a cloud needs to possess the ability to react not only to known threats, but also to new challenges that target cloud infrastructures.

However the detection capabilities of the current techniques that have been implemented in cloud infrastructure offers a range of beneficial properties such as, service transparency and elasticity that introduce a number of vulnerabilities which are the outcome of its underlying virtualized nature. In the proposed system a detection approach, comprising detection elements which form the cloud resilience architecture has been implemented. More specifically, the applicability of novelty detection under Support Vector Machine (SVM) is also exhibited.

The proposed novelty detector is a new detection technique which helps in the detection of the malicious file. Implementation of SVM improves the overall efficiency of the system.The novelty detector that is based on SVM algorithm can be adopted in systems that emphasizes on file upload and download. It also governs the same, aiming at securing the system. In order to empower the generic properties of the detection approach a secret key generation is implemented using RSA algorithm. The novelty detector offers reasonable security, usability and fits well with some practical applications for improving the detection and also reduces the probability of downloading the malicious document and provides an estimation of malicious files and non-malicious files.

# Chapter 1

## Introduction

Cloud computing is currently one the leading innovations. Most IT companies have their own cloud computing technologies. Though cloud is still a budding technology it possess a lot of risk to threats. In the nearest future, cloud will have to undergomany more security exploitation events around cloud computing providers and users, which will lead to many researches on cloud computing security in thenext decade. Hence, there has been a rapid change and growth in cloud computing security discipline, with ongoing efforts tocope with the requirements and capabilities regarding privacy and security issues.

Security in cloud computing focuses mainly on the attacks and hacking attempts related to cloud computing providers and systems. Somespecific security threats, vulnerabilitiesof services and service-oriented architectures require new taxonomies and classification criteria, so do attacks on cloud computing scenarios. Cloud computing is vulnerable to various attacks,and an initial attack taxonomy for these is based on the type of attack surfaces given. Cloud data centres are beginning to be used for variety of purposes –such as services across private, public and commercial domains. These need to be secure and resilient in the face of challenges. They include cyber-attacks as well as component failures and misconfigure. However, clouds have characteristics and intrinsic internal operational structures that can help the cloud to recover from its security threats.

This scheme can efficiently operate with signature-based approaches on an online basis in scenarios were decryption is feasible and cost-effective. Overall, the goal is to develop detection techniques that are specifically targeted at the cloud and integrate with the infrastructure itself in order to, not only detect, but also provide a good resilience technique .At the infrastructure level the elements that make up a cloud datacentre, i.e. cloud nodes, which are servers that run a hypervisor in order to host a number of Virtual Machines (VMs); and network infrastructure elements that provide the connectivity within the cloud and connectivity to external service users.

A cloud service is provided through one or more interconnected VMs that offer access to the outside world. Various techniques will be used in order to detect files, when a particular file is uploaded. The file extensions is one such way of indicating that the file contains malicious content, other way could be the hexadecimal that could probably be in a binary text file. These could be one way of detecting malicious files in cloud and thus by avoiding further hazards that can be caused by downloading these malicious file to the end-users.

Cloud networks have emerged as important components and they deal with many important operation which include intensive data processing and management tasks that provide the foundation for a number of services offered to the end-user. There have been extremely large investments from companies such as Google, Facebook, eBay, Microsoft and Yahoo! On their datacentres in order to support cloud services that aim to exploit all the capabilities offered by virtualized environments. Cloud environments can be attacked in number of ways and are prone to security threats that aim to exploit security loops holes. A great challenge that continues to exist in cloud networks is the in adequate detection of anomalous activities caused either by legitimate or malicious intent. In particular, the detection of malicious file is extremely critical since malicious content in the majority of cases is first point of initiation for larger attacks.

The security mechanisms aims to improve and adjust the performance of traditional Intrusion Detection Systems, under signature rule-based strategies which performs Deep Packet Inspection on network packets. It monitors system-related features on the Virtual Machine using Virtual Machine Introspection (VMI) methods in order to analyse and detect threats on the VM Operating System (OS). Outcomes derived by the expert research, aimed to develop a detection strategy that considers both system and network information as seen on each VM. Majority of the used rule based detection policies that depend on pre-defined signatures. Hence, the ability at adapting on new types of anomalies, which usually appear in cloud environments, is extremely limited. Some approaches went beyond the draw backs of signature-based techniques and made strategies that rely on statistical anomaly detection schemes. Signature-based techniques approaches mainly dealt with (up to a reasonably good level) the identification and detection of anomalies by considering only a single source of information which was either from the network or the system.

### 1.2 Virtual Machine Introspection for security

Virtual machine introspection (VMI) was first proposed in together with Livewire, a prototype IDS that uses VMI to monitor VMs. Xen Project is a hypervisor using a microkernel design, providing services that allow multiple computer operating systems to execute on the same computer hardware concurrently. It was developed by the Linux Foundation and is supported by Intel XenAccess is a monitoring library for guest OS running on top Xen that applies VMI and virtual disk monitoring capabilities to access the memory state and disk activity of a target OS. Further VMI-based approaches are virtual machine replay and detecting past intrusions. These approaches mandate that the system is clean when it starts being monitored, which our solution VMs does not require. Virtual machine monitors (VMMs) are important because different kinds of virtualization constitute a major facet of cloud computing.

### 1.3 Support Vector Machine

Support vector machines (SVMs, also support vector networks) are supervised Learning models with associated learning algorithms that analyse data that is used for classification and analysis, in machine learning. For 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. An SVM model is a representation where the input plotted as points in space,and are mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall. In case of performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

A support vector machine constructs a hyper plane or set of hyper planes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks. Intuitively, a good separation is achieved by the hyper plane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier.

### 1.4 Summary

In this chapter, we get to know the field of choosing this project and current trends in choosing it. It also give a brief insight about the project and what is to be achieved in this project.

# 

## Literature Review

The literature review chapter deals with published literature in scholarly journals, conference proceedings which are related to the topic of interest.

The File type identification and file type clustering may be difficult tasks that have an increasingly importance in the field of computer and network security. Classical methods of file type detection including considering file extensions and magic bytes can be easily spoofed. Content-based file type detection is a newer way that is taken into account recently. In this paper, a new content-based method for the purpose of file type detection and file type clustering is proposed that is based on the PCA and neural networks. The proposed method has a good accuracy and is fast enough.

True identification of a file format is a tedious task. There are catalogues containing several thousand of known file types, without having any global standard for the file types. File type detection methods can be categorized into three kinds: extension -based, magic bytes-based, and content-based methods, each of them has its own strengths and weaknesses, and none of them are comprehensive or fool proof enough to satisfy all the requirements.

The fastest and easiest method of file type detection is the extension-based method. Operating systems use such approach almost exclusively. All the file types, at least in the Windows-based systems, are generally accompanied by an extension. This approach can be applied to both binary and text files. While it does not need to open the files, it is by far the fastest way to classify the files. However, it has a great vulnerability while it can be easily spoofed by a simple file renaming.

### 2.1 Intrusion detection system

Cloud and Grid computing are the most vulnerable targets for intruder’s attacks due to their distributed environment. For such environments, Intrusion Detection System (IDS) can be used to enhance the security measures by a systematic examination of logs, configurations and network traffic. Traditional IDSs are not suitable for cloud environment as network based IDSs (NIDS) cannot detect encrypted node communication, also host based IDSs (HIDS) are not able to find the hidden attack trail. Kleber, schulter et al. Have proposed an IDS service at cloud middleware layer, which has an audit system designed to cover attacks that NIDS and HIDS cannot detect. The architecture of IDS service includes the node, service, event auditor and storage.

The detection capabilities currently provided by host-based antivirus software can be more efficiently and effectively provided as an in-cloud network service. Instead of running complex analysis software on every end host, we suggest that each end host runs a lightweight process to detect new files, send them to a network service for analysis, and then permit access or quarantine them based on a report returned by the network service. There is an increase in the dependence of cloud computing as consumers increasingly move to mobile platforms for their computing needs. Cloud technologies have become possible by tuberculation in order to share physical server resources.

### 2.2 Novelties in cloud thread models

AngelosMarnerides, Cyriac James investigates, the techniques for traffic analysis and anomaly detection are typically carried out independently in different parts of the network, either in the edge or in the core networks alone.Anomaly detection is an important problem that has been researched within diverse research areas and application domains, it refers to the problem of finding patterns in data that do not conform to expected beta. Many anomaly detection techniques have been specifically developed for certain application domains, while others are more generic. The existing techniques have been grouped into different categories based on the underlying approach adopted by each technique. For each category, key assumptions have been identified, which are used by the techniques to differentiate between normal and anomalous behaviour. At an abstract level, an anomaly is denied as a pattern that does not confirm to expected normal behaviour. A straightforward anomaly detection approach, there-fore, is to deny a region representing normal behaviour and declare any observation in the data which does not belong to this normal region as an anomaly[1]

Providing security in a distributed system requires more than user authentication with passwords or digital certificates and confidentiality in data transmission. Distributed model of cloud makes it vulnerable and prone to sophisticated distributed intrusion attacks like Distributed Denial of Service (DDOS) and Cross Site Scripting (XSS). To handle large scale network access traffic and administrative control of data and application in cloud, a new multi-threaded distributed cloud IDS model has been proposed. A multi-threaded cloud IDS model is proposed which can be administered by a third party monitoring service for a better optimized efficiency and transparency for the cloud user [2]

M. R. Watson, N. Shirazi, A. Mauthe investigates, a cloud security monitoring system, which automatically detects and makes response to anomalies that occur inside a cloud infrastructure. Observes specific changes of flow data with the intension to extract anomalies. In this work-in-progress paper we present one such taxonomy based on the motion of attack surfaces of the cloud computing scenario participants. High volume of data in cloud environment could be handled by a single node IDS through a multi-threaded approach. Cloud computing, enormous network access rate, relinquishing the control of data & applications to service provider and distributed attacks vulnerability, an efficient, reliable and information transparent IDS is required. A multi-threaded cloud IDS model is proposed which can be administered by a third party monitoring service for a better optimized efficiency and transparency for the cloud user [3]

Michael R. Watson, Noor-ul-Hassan Shirazi, and David Hutchison investigates, Intrusion detection (ID) as a type of security management system for computers and networks. An ID system gathers and analyses information from various areas within a computer or a network to identify possible security breaches, which include both intrusions (attacks from outside the organization) and misuse (attacks from within the organization. Traditional IDSs are not suitable for cloud environment as network based IDSs (NIDS) cannot detect encrypted node communication, also host based IDSs (HIDS) are not able to find the hidden attack trail. The architecture of IDS service includes the node, service, event auditor and storage. The node contains resources that are accessed through middleware which defines access-control policies. The storage holds behaviour based (comparison of recent user actions to usual behaviour) and knowledge-based (known trails of previous attacks) databases. The audited data is sent to IDS service core, which analyses the data and alarm to be an intrusion. The authors have tested their IDS prototype with the help of simulation and found its performance satisfactory for real-time implementation in a cloud environment. Although they have not discussed the security policies compliance check for cloud service provider and their reporting procedures to cloud users [4]

M. Garnaeva investigates, the implementation of IDS for monitoring and access control and also security in virtualisation. Mutual auditability can also significantly assist with incident response and recovery, since both the cloud provider and the cloud user could be either the source or the target of an attack. Auditability also enables the attribution of blame in search and seizure incidents, which can prove vital so that law enforcement agencies do not overreach in carrying out their duties. IDS implementation in cloud computing requires an efficient, scalable and virtualization based approach. In cloud computing, user data and application is hosted on cloud service provider’s remote servers and cloud user has a limited control over its data and resources. In such case, the administration of IDS in cloud becomes the responsibility of cloud provider. Although the administrator of cloud IDS should be the user and not the provider of cloud services [5]

H. Binsalleeh, L. Wang investigates, the intrusion detection message exchange format (IDMEF) standard has been used for communication between different IDS sensors. The authors have suggested the deployment of IDS sensors on separate cloud layers like application layer, system layer and platform layer. Alerts generated are sent to ,Event Gatherer program. Event gatherer receives and convert alert messages in IDMEF standard and stores in event data base repository with the help of Sender, Receiver and Handler plug-ins. The analysis component analyses complex attacks and presents it to user through IDS management system. The authors have proposed an effective cloud IDS management architecture, which could be monitored and administered by the cloud user. They have provided a central IDS management system based on different sensors using IDMEF standard for communication and monitored by cloud user [6]

The authorT. Brewster investigates, that many small malicious variants could pose the same financial problem for end users as one big nasty piece of malware. A subtle difficulty with understanding cloud computing threats arises from potentially inaccurate mental models of cloud computing as an always-available service. This viewpoint which arises from the general paradigm of drawing upon a commodity service with much the flavour of a utility can create a false sense of security, leading to inadequate security good practices, such as regular data backups across multiple cloud providers. As such, find that while cloud computing fails at the same rate as other types of systems, the impact of those failures manifest more [7]

A. K. Marnerides, P. Spachos, P. Chatzimisios investigates, the services and interface to control the deployed machines. Amazon Elastic Cloud Computing (EC2) is one the most known commercial and publicly available cloud computing service. In order to control the deployed machines, EC2 offers a SOAP interface for e.g. starting new instances of a machine, terminating an instances etc. In 2008 a weakness was found in this control service using a form of a Signature Wrapping Attackit was possible to modify an eavesdropped message despite of the digital signed operation. Thus, an attacker was able to execute arbitrary own machine commands on behalf of a legitimate cloud user .The attack incident can be reduced to two separate actions: attacking the cloud control interface (i.e. the cloud-touser attack surface) to get control of the cloud system, then attacking the service instances using the service-to-cloud attack surface [8]

Attacking the very same Amazon EC2 cloud system, the authors have illustrated the steps necessary to gain confidential information from running service instances. This attack scheme can be reduced to the following steps. First, manipulate the cloud-to-user surface for setting up a service instance on the same hardware with the victim’s service instance. Then, using that service instance, attack the cloud-to-service interface to gain hypervisor and hardware-related information (i.e. breaking privacy) or go further attacking the service to-cloud interface of the victim’s service instance in order to perform “service-to-service‘” attacks.

The author L. Kaufman investigates, the main idea of the Cloud Malware Injection attack is that an attacker uploads a manipulated copy of a victim’s service instance so that some service requests to the victim service are processed within that malicious instance. In order to achieve this, the attacker has to gain control over the victim’s data in the cloud system (e.g. using one of the attacks described above. As a side-effect, if the attacker uses a hijacked cloud service for attack message generation, he can trigger huge usage bills for cloud-provided services that the real user never ordered [9]

M. Christodorescu, R. Sailer, D. L. Schales investigates, cloud auditing is a difficult task to check compliance of all the security policies by the vendor. Cloud service provider has the control of sensitive user data and processes, so an automated or third party auditing mechanism for data integrity check and forensic analysis is needed. Privacy of data from third party auditor is another concern of cloud security. Combining the contemporary and historical viewpoints, we arrive at the position that many cloud computing security problems are not in fact new, but often will still require new solutions in terms of specific mechanisms. Existing contemporary works already explore many pertinent topics; we highlight here several areas that deserve more attention. One possible approach would be to formulate the security primitives around defending different stakeholders against different particular threat models[10]

### 2.3 Aim, Objectives and Methodology

The literature review clearly indicates the gap in the efficiency of the current algorithm to detect the malicious content. This motivated the proposed study wherein, the SVM algorithm will efficiently help in the detection of malicious file.

#### Aim

To detect the anomalies using a novelty detector approach that employs the one-class Support Vector Machine (SVM) algorithm and demonstrate the effectiveness of malicious file detection in cloud computing infrastructure.

#### Objectives

1. To analyse the security issues in cloud computing infrastructure
2. To design a new frame work based on SVM
3. To implement SVM within a Novelty detector.
4. To evaluate the effectiveness of Novelty detector.

#### Methodologies

Methodology for objective-1:

* Literature survey will be made to analyse the current securities based on malicious file detection.

Methodology for objective-2:

* A Frame work will be designed on cloud resilience architecture using SVM

Methodology for objective-3:

* Cloud resilience will be Implemented using virtualization and cloud technologies.

Methodology for objective-4:

* Efficiency of the proposed model can be evaluated using SVM.

### 2.4 Summary

Overall in this chapter, a depth survey is done on the existing system and the efficiency and techniques used of the existing system is reviewed which gave an insight towards the problems faced by the existing system, followed by aim objective methodology to create a new system has been explained in the end.

# 

## Design

Design is the technique which is used to do the system analysis, it would be necessary to identify the data that is required to be processed to produce the outputs. Design features can ensure reliability of the system and generate correct reports from the accurate data. It is also possible to determine whether the user can interact efficiently with the system.

The system registers the service provider and the end users, who can upload or download a required file. The cloud server checks for malicious files among the files that are uploaded by the service provider. An alert message is sent to the particular service provider if a

### 3.1Design Procedure

Design is the technique which is used to do the system analysis, it would be necessary to identify the data that is required to be processed to produce the outputs. Design features can ensure reliability of the system and generate correct reports from the accurate data. It is also possible to determine whether the user can interact efficiently with the system.

The process takes the data from the database and checks these data with the user inputted data. Once the process has been finished, the output or the report will be generated based on the process results.

### 3.2System Architecture

The system architecture clearly explains the entire system. The architecture consists of the following system entities.

* Cloud Server
* Service Provider
* End User

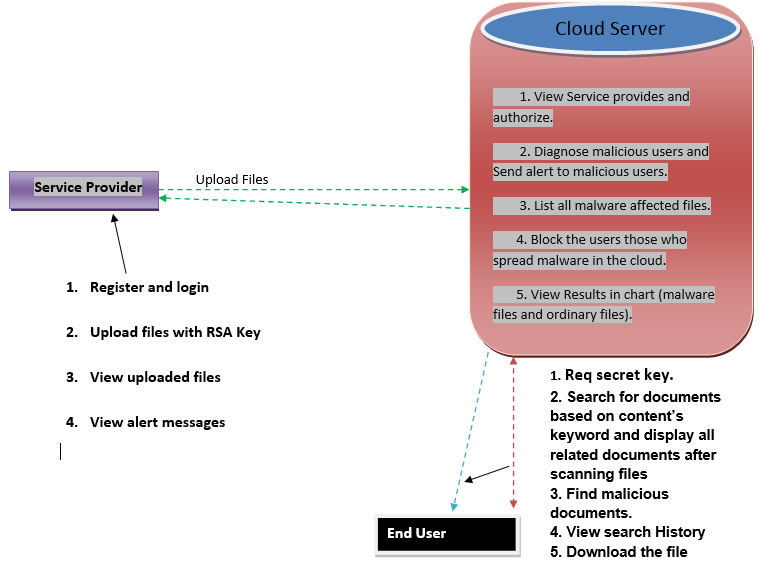


Figure 3.1 System Architecture

*Cloud server*

The cloud server authenticates the service provider and the user, based on the credentials provided. The Figure 3.1representsthe main functionality of the cloud server is to view all Files, View Service provides and authorize them,detect malicious users those who upload malicious file and find percentage of malicious files and send alert to service provider that uploads the file, list all the affected files. Cloud server also list end Users & Authorize, list all transactions block the users those who upload malicious file in the cloud.

*Service Provider*

Cloud servers registers the service provider based on the credentials provided, enabling the service provider to login to the system.The Figure 3.1represents the service provider whichplays main role in uploading the file by sending upload request to the cloud server. Cloud server authorises of the request and confirms the file upload the file is done. Service provider also documents the list of users and files that are uploaded by them, it also maintains a record of search history.Service provider receives an alert message from the cloud server if the uploaded file has malicious content.

*End users*

The end users are authenticated and authorized by the cloud server. The Figure 3.1represents the cloud server registration of the end users based on the credentials provided, enabling the end users to login to the system. End user has the following functionalities such as viewing the uploaded file, and downloading the required file. End user has the privilege of passing a query to make a search request. Prior to downloading the file a secret key with RSA encryption is sent to the end user by the cloud server. End user receives the secret key using which downloading of file is done, if the requested file has malicious content, the file cannot be viewed after downloading.

### 3.3 UML

UML stands for Unified Modelling Language. UML is a standardized general-purpose modelling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modelling Language is a standard language for specifying, Visualization, Constructing and documenting the artefacts of software system, as well as for business modelling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems. The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

#### 3.3.1 Use Case Diagram

A use case diagram in the Unified Modelling Language (UML) is a type of behavioural diagram defined by and created from a Use-case analysis. A use case diagram is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. It can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well.

Use case diagrams are usually referred to as behaviour diagrams used to describe a set of actions that some system or systems should or can perform in collaboration with one or more external users of the system (actors). Each use case should provide some observable and valuable result to the actors. Use case diagrams are in fact twofold - they are both behaviour diagrams, because they describe behaviour of the system, and they are also structure diagrams - as a special case of class diagrams where classifiers are restricted to be either actors or use cases related to each other with associations. The Figure 3.2represents the use case diagram of the system with the actors on the scene being the service provider, end users. Novelty detector has its many functionalities and provides various functionalities to each of its actor. The below diagram shows the interaction of various actors with the system.

*• Service provider*:

Figure 3.2 use case diagram of service provider is represented in the Figure 3.2 where the service provider plays main role in uploading the file by sending upload request to the cloud server. Cloud server authorises the service provider’s request to registerto the system. Service provider uploads files to the system and documents the list of users and files that are uploaded, it also maintains a record of search history.

The service provider is allowed to upload files only after the authentication and confirmation of the provider is validated by the cloud server. All the files that are uploaded are checked and verified, if any file contains malicious data, service provider receives an alert message from the cloud server.

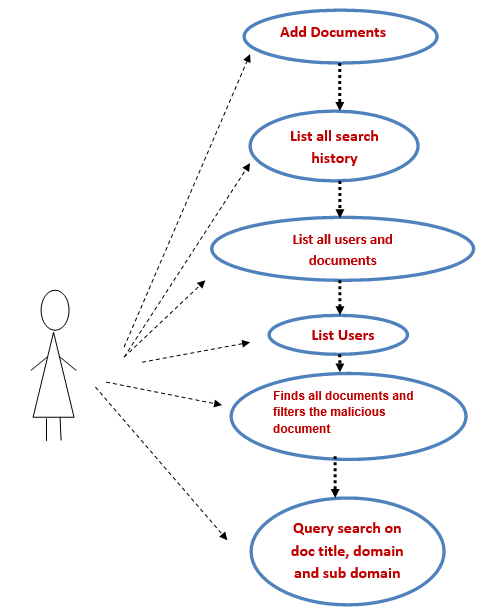


Figure 3.2 use case diagram of service provider

*•End user:*

In the Figure 3.3 use case for end user the end users are authenticated and authorized by the cloud server. Cloud servers registers the end users based on the credentials provided, enabling the end users to login to the system. End user has the following functionalities such as viewing the uploaded file, and downloading the required file. End user has the privilege of passing a query to make a search request. Prior to downloading the file a secret key with RSA encryption is sent to the end-user by the cloud server. End user receives the secret key using which downloading of file is done, if the requested file has malicious content, the file cannot be viewed after downloading.

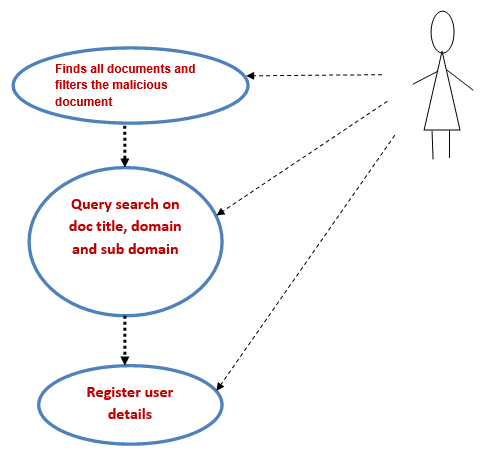


Figure 3.3 use case for end user

#### 3.3.2Sequence Diagram

A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and sequence of messages exchanged between graphical manner. The objects needed to carry out the functionality of the scenario.

Sequence diagrams shows, as parallel vertical lines (lifelines), different processor objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simpler runtime scenarios in a graphical manner.

The sequential process of login activity of the service provider and the end user is represented in the Figure 3.4. The authentication process of the cloud server. Sequential steps of uploading a file by the service provider and downloading the file by the end user.

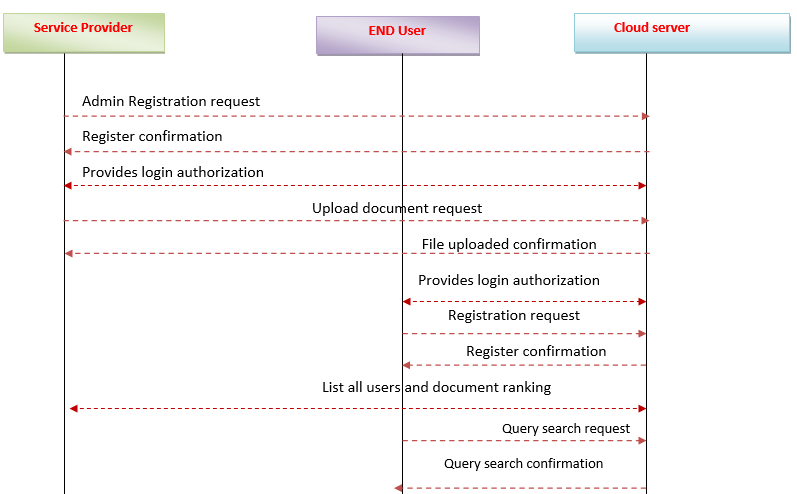
****

Figure 3.4 sequence diagram

#### 3.3.3 Class diagram

Class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes.

The class diagram is the main building block of object-oriented modelling. It is used both for general conceptual modelling of the systematics of the application, and for detailed modelling translating the models into programming code. The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.

The class diagram for the application frame work is as shown below:

* Cloud server
* Register
* End user

*Cloud server:*

The cloud server authenticates the service provider and the user based on the credentials provided as represented in Figure 3.5.The main functionality of the cloud server is to view all Files, View Service provides and authorize them, detect malicious users those who upload malicious file and find percentage of malicious files and to send alert to service provider that uploads the file, list all the affected files. Cloud server also list end Users & Authorize, list all transactions block the users those who upload malicious file in the cloud.

*Service Provider:*

The cloud server’s registers the service provider based on the credentials provided, enabling the service provider to login to the system which is represented in Figure 3.5Service provider plays main role in uploading the file by sending upload request to the cloud server. Cloud server authorises of the request and confirms the file upload the file is done. Service provider also documents the list of users and files that are uploaded by them, it also maintains a record of search history. Service provider receives an alert message from the cloud server if the uploaded file has malicious content.

*End users:*

The end users are authenticated and authorized by the cloud server. Cloud servers registers the end users based on the credentials provided, enabling the end users to login to the systemwhich is represented in Figure 3.5End user has the following functionalities such as viewing the uploaded file, and downloading the required file. End user has the privilege of passing a query to make a search request. Prior to downloading the file a secret key with RSA encryption is sent to the end user by the cloud server. End user receives the secret key using which downloading of file is done, if the requested file has malicious content, the file cannot be viewed after downloading.

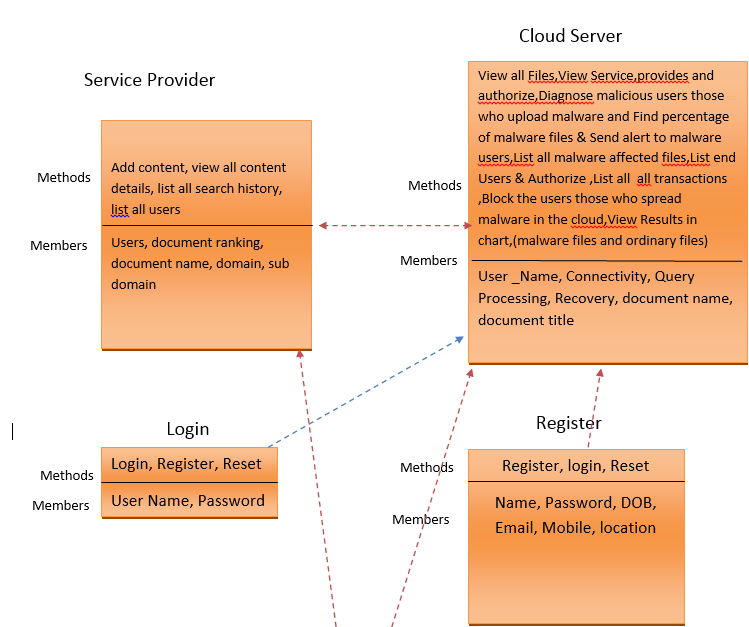
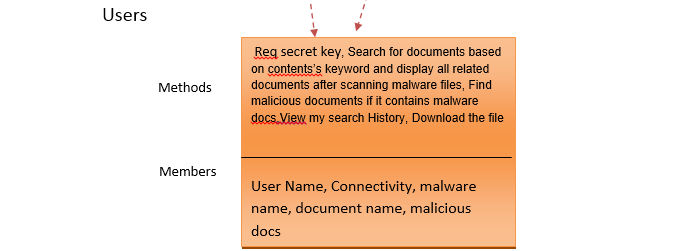
****

Figure 3.5 class diagram

#### Activity diagram

Activity diagrams are graphical representation of workflows of stepwise activities and actions with support for choice, iteration and concurrency.Activity diagram is another important diagram in UML to describe dynamic aspects of the system. Activity diagram is basically a flow chart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. Activity diagrams are intended to model both computational and organizational processes.

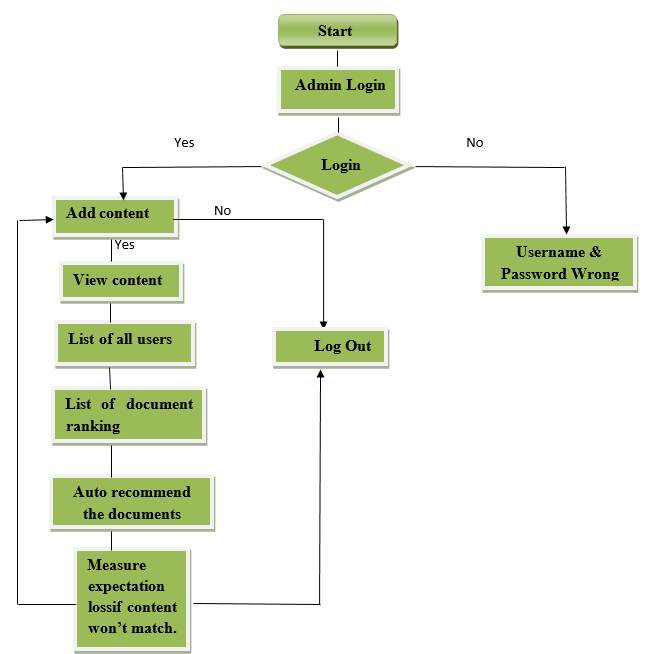


Figure 3.6 Activity diagram for cloud server

.

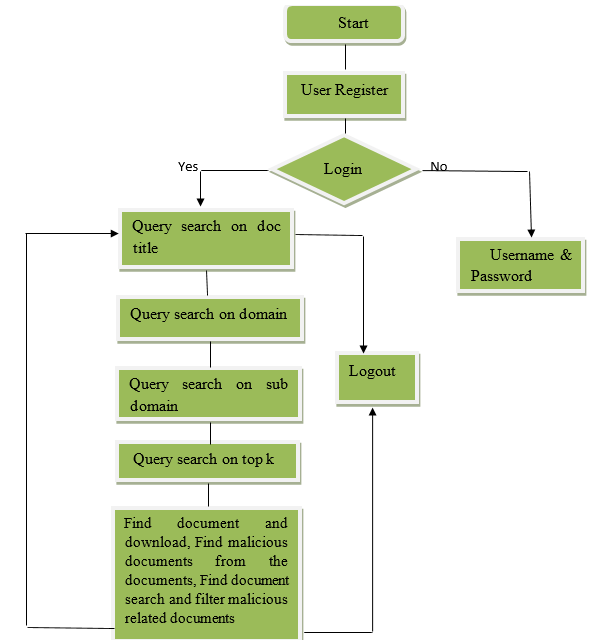
****

Figure 3.7 Activity diagram for end user

The Figure 3.6illustrates the flow of control in the cloud server module. In the cloud server module does the job of authentication. The cloud server module authenticates the user based on the credentials. If the credentials provided are correct, then cloud server enables the service provide to add content and various activities can be performed, else a message is displayed and login is terminated

The Figure 3.7shows flow of control in the end user module. In the end user module, the cloud server authenticates the user based on the credentials. If the credentials provided are correct, then cloud server enables the end user to view and download content and various activities can be performed, else a message is displayed and login is terminated.

### 3.4 Summary

In the design chapter the step by step explanation of the design is provided. The proposed designs that are explained in the design chapter by using UML diagrams, Class diagrams, Sequence diagrams and Activity diagrams which are considered to be efficient, and proved to be best suited.

# 

## Implementation

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus the implementation part can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective.

The implementation stage involves careful planning, investigation of the existing system and it’s constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

### 4.1 Cloud Server

* Admin Profile (Add, Edit).
* Account Details (Add, Provides authentication, Edit, Delete).
* When receives a login request from the Service Provider and End User, checks the profile and provides login authentication.
* Login details (Add, Save, Edit).
* Saves the login details of Service Provider and End User.
* Sends alert message.

**Description:**

Cloud Server provides authentication permission to the Service Provider and to the End User and saves the login details. When the Service Provider uploads a malicious file, the Cloud Server sends an alert message to the Service Provider. And when the End User tries to download that malicious file, the Cloud Server will prevent it from downloading the malicious file. The Cloud Server will detect malicious users those who upload malicious file and find percentage of malicious files and send alert to service provider that uploads the file, list all the affected files. Cloud server also list End Users & Authorize, list all transactions block the users those who upload malicious file in the cloud.

**Authentication by the cloud server:**

String name = request.getParameter("name");

String pass = request.getParameter("pass");

String type=request.getParameter("type");

String sql ="";

try

{

if(type.equalsIgnoreCase("cloud"))

{

sql = "SELECT \* FROM mdcci\_cloud where name='" + name + "' and pass='" + pass + "' ";

}

if(type.equalsIgnoreCase("provider"))

{

sql = "SELECT \* FROM mdcci\_provider p where name='" + name + "' and pass='" + pass + "' and authorize='Authorized' and p.name not in(select name from mdcci\_pauth)";

}

if(type.equalsIgnoreCase("user"))

{

sql = "SELECT \* FROM mdcci\_user p where name='" + name + "' and pass='" + pass + "' and authorize='Authorized' and p.name not in(select name from mdcci\_uauth)";

}

application.setAttribute("cname", name);

Statement stmt = connection.createStatement();

ResultSetrs = stmt.executeQuery(sql);

if (rs.next())

{

if(type.equalsIgnoreCase("provider"))

{

application.setAttribute("pname", name);

response.sendRedirect("P\_MainPage.html");

}

if(type.equalsIgnoreCase("user"))

{

application.setAttribute("ename", name);

response.sendRedirect("E\_MainPage.html");

}

if(type.equalsIgnoreCase("cloud"))

{

application.setAttribute("cname", name);

response.sendRedirect("C\_MainPage.html");

}

}

else

{

response.sendRedirect("wronglogin.html");

}

}

catch (Exception e)

{

out.print(e);

e.printStackTrace();

}

String name=request.getParameter("name");

try

{

int x=connection.createStatement().executeUpdate("insert into mdcci\_pauth values ('"+name+"','Un-Authorized')");

if(x>0)

{

response.sendRedirect("blkuser.jsp");

}

}

catch(Exception e)

{

e.printStackTrace();

}

try

{

ResultSetrs=connection.createStatement().executeQuery("select \* from mdcci\_provider p where p.name not in(select name from mdcci\_pauth)");

ResultSet rs1=connection.createStatement().executeQuery("select \* from mdcci\_provider p where p.name in(select name from mdcci\_pauth)");

while(rs.next())

{

//checks for the provider details

}

}

catch(Exception e)

{

e.printStackTrace();

}

**Unblocking:**

String name=request.getParameter("name");

try

{

int x=connection.createStatement().executeUpdate("delete from mdcci\_pauth where name='"+name+"'");

if(x>0)

{

response.sendRedirect("blkuser.jsp");

}

}

catch(Exception e)

{

e.printStackTrace();

}

try

{

String fname=request.getParameter("fname");

String name=request.getParameter("name");

SimpleDateFormatsdfDate = new SimpleDateFormat("dd/MM/yyyy");

SimpleDateFormatsdfTime = new SimpleDateFormat("HH:mm:ss");

Date now = new Date();

String strDate = sdfDate.format(now);

String strTime = sdfTime.format(now);

String dt = strDate + " " + strTime;

ResultSetrs=connection.createStatement().executeQuery("select \* from mdcci\_alert where fname='"+fname+"' and name='"+name+"'");

if(!rs.next())

{

connection.createStatement().executeUpdate("insert into mdcci\_alert values('"+fname+"','"+name+"','"+dt+"')");

//response.sendRedirect("csa.jsp");

}

}

catch(Exception e)

{

e.printStackTrace();

}

### 4.2 Service Provider

* Account Details (View, Update).
* View files.
* Upload files.

**Description:**

Service Provider sends login request to the cloud server. After it receives login authentication, the service provider can view the files in the cloud server and can also upload

files.Service provider also documents the list of users and files that are uploaded by them, it also maintains a record of search history. Service provider receives an alert message from the cloud server if the uploaded file has malicious content.

**Registration and authentication of the service provider**

String user = request.getParameter("user");

try{

int x=connection.createStatement().executeUpdate("update mdcci\_provider set authorize='Authorized' where name='"+user+"' ");

if(x > 0)

{

response.sendRedirect("vp.jsp");

}

}

catch(Exception e)

{

e.printStackTrace();

}

try

{

String s1 = "", s2 = "", s3 = "", s4 = "", s5 = "", s6 = "", s7 = "", s8, s9 = "", s10, s11, s12, s13;

inti = 0, j = 0, k;

Statement st21 = connection.createStatement();

ResultSetrs=st21.executeQuery("select \* from mdcci\_provider p where p.name not in(select name from mdcci\_pauth)");

while(rs.next())

{

i=rs.getInt(1);

//scans the details on name, email id, password, gender, address, mobile number, date of birth, pincode, location, profile picture

//prints the details

if(s11.equalsIgnoreCase("un-authorized"))

{

//prints the invalidity status

}

else

{

//prints the validity status

}

}

connection.close();

}

catch(Exception e)

{

out.println(e.getMessage());

}

### 4.3 End User

* Account details (View, Download).
* View flies.
* Download files.

**Description:**

End User sends the login request to the Cloud Server. When it receives the authentication permission, the End User can view and can also download the files. End user has the privilege of passing a query to make a search request. If the requested file has malicious content, the file cannot be viewed after downloading.

**Cloud Server and Service Provider**

Process involved in providing authentication to the Service Provider.

1. Service Provider send login request to the Cloud Server.
2. Cloud servers registers the service provider based on the credentials provided, enabling the service provider to login to the system.
3. Service provider uploads a file by sending upload request to the cloud server.
4. Cloud server authorises of the request and confirms the file, uploading the file is done.
5. Service provider also contains the list of users and files that are uploaded by them, it also maintains a record of search history.
6. Service provider receives an alert message from the cloud server if the uploaded file has malicious content.

**Registration and authentication of the end-user:**

String user = request.getParameter("user");

try

{

int x=connection.createStatement().executeUpdate("update mdcci\_user set authorize='Authorized' where name='"+user+"' ");

if(x > 0)

{

response.sendRedirect("vu.jsp");

}

}

try

{

String s1 = "", s2 = "", s3 = "", s4 = "", s5 = "", s6 = "", s7 = "", s8, s9 = "", s10, s11, s12, s13;

inti = 0, j = 0, k;

Statement st21 = connection.createStatement();

ResultSetrs=st21.executeQuery("select \* from mdcci\_user p where p.name not in(select name from mdcci\_uauth)");

while(rs.next())

{

i=rs.getInt(1);

//scans the details on name, email id, password, gender, address, mobile number, date of birth, pincode, location, profile picture

//prints the details

if(s11.equalsIgnoreCase("un-authorized"))

{

//prints the invalidity status

}

else

{

//prints the validity status

}

}

connection.close();

}

catch(Exception e)

{

out.println(e.getMessage());

}

**Cloud Server and End User**

Process involved in providing authentication to the End User.

1. End User sends a login request to the Cloud Server.
2. Cloud Server registers the end users based on the credentials provided, enabling the end users to login to the system.
3. End user has the following functionalities such as viewing the uploaded file, and downloading the required file.
4. End user has the privilege of passing a query to make a search request. Prior to downloading the file a secret key with RSA encryption is sent to the end user by the cloud server.
5. End user receives the secret key using which downloading of file is done.
6. If the requested file has malicious content, the file cannot be viewed after downloading.

try

{

ResultSetrs=connection.createStatement().executeQuery("select \* from mdcci\_provider p where p.name not in(select name from mdcci\_pauth)");

ResultSet rs1=connection.createStatement().executeQuery("select \* from mdcci\_provider p where p.name in(select name from mdcci\_pauth)");

while(rs.next())

{

//checks for the availability of the file

}

}

catch(Exception e)

{

e.printStackTrace();

}

# 

## Testing

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

### Introduction of Testing

Testing is a development procedure where programmers create tests as they develop software. The tests are simple short tests that test functionality of a particular unit or module of their code, such as a class or function. Software testing is any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results. Testing is more than just debugging. The purpose of testing can be quality assurance, verification or reliability estimation. Testing can be used as a generic metric as well.

### 5.2 Software Testing.

#### 5.2.1 Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is common for coding and unit testing to be conducted as two distinct phases.

Unit testing focuses verification effort on the smallest unit of Software design. Unit testing exercises specific paths in a module’s control structure to ensure complete coverage and maximum error detection. This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, is called Unit Testing.All error handling paths are also tested.

Table 5‑1: Test cases for the project

|  |  |  |
| --- | --- | --- |
| **Steps** | **Test Action** | **Results** |
| Step 1 | Enter the URL:  localhost:8080/Malware/Index.html | Index page loaded. |
| Step 2 | Click on Register | Register page loaded. |
| Step 3 | Create a new account | Account Created. |
| Step 4 | Click on login | Login page loaded. |
| Step 5 | Login to an existing account | Login Successful. |
| Step 6 | Click on view files | List of files loaded. |
| Step 7 | Input the query to search for files | Related files loaded based on keyword search. |
| Step 8 | Download the files | Download successful. |

In order to handle all the errors that occurred while running the application. The common errors we saw were reading a tuple with an attribute set to null and database connection getting lost.

For Testing, Top-Down design a decomposition process which focuses at the flow of control. The first step is to study the overall aspects of the tasks at hand and break it into a number of independent modules. The second step is to break one of these modules further into independent sub modules. One of the important features is that each level the details at lower levels are hidden. So unit testing was performed first and then system testing.

**Test objectives**

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

#### 5.2.2 Integrationtesting

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually tested, the combination of components is correct and consistent.

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.The main objective in this testing process is to take unit tested modules and builds a program structure that has been dictated by design.

**Steps to perform integration testing:**

Step 1: Create a Test Plan   
Step 2: Create Test Cases and Test Data   
Step 3: Once the modules have been integrated, execute the test cases   
Step 4: Fix the bugs if any and re-test the code   
Step 5: Repeat the test cycle until the components have been successfully integrated.

Table 5‑2: Test cases for Integration testing

|  |  |
| --- | --- |
| Name of the Test | Integration testing |
| Test plan | To check whether the system works properly when all the modules are integrated. |
| Test Data | Upload and download a file. |

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

#### User Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

User Acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wherever required. The system developed provides a friendly user interface that can easily be understood even by a person who is new to the system.

**Test Results:** All the test cases mentioned were successfully. No defects encountered.

#### Output Testing

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration. Hence the output format is considered in two ways, one is on screen and another in printed format.

#### Validation Checking

Validation tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Validation testing is centred on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input: identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Procedures: interfacing systems or procedures must be invoked.

### 5.3 System Testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

Table 5‑3: Test cases for Input-Output

|  |  |
| --- | --- |
| **Name of the Test** | **System Testing** |
| Item being tested | Over all functioning of the system with all functions properly linked. |
| Sample Input | A proper file must be uploaded. |
| Expected Output | The non-malicious files must be available for successful downloading. |
| Actual Output | Application reacts to user inputs in expected manner, allowing the user to download the needed non-malicious files. |
| Remarks | Successful |

#### 5.3.1White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

#### 5.3.2Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

# 

## Results and Discussions

This chapter deals with the various results that are obtained after the successful execution of the application. This not only tells the working of the project but also gives a detailed explanation of every step.



Figure 6.1 Home page

TheFigure 6.1, shows themain home page of the project which showcases various modules such cloud server,end-user,service provider, by choosing the options, one is directed to different modules and each module has its own functionality.

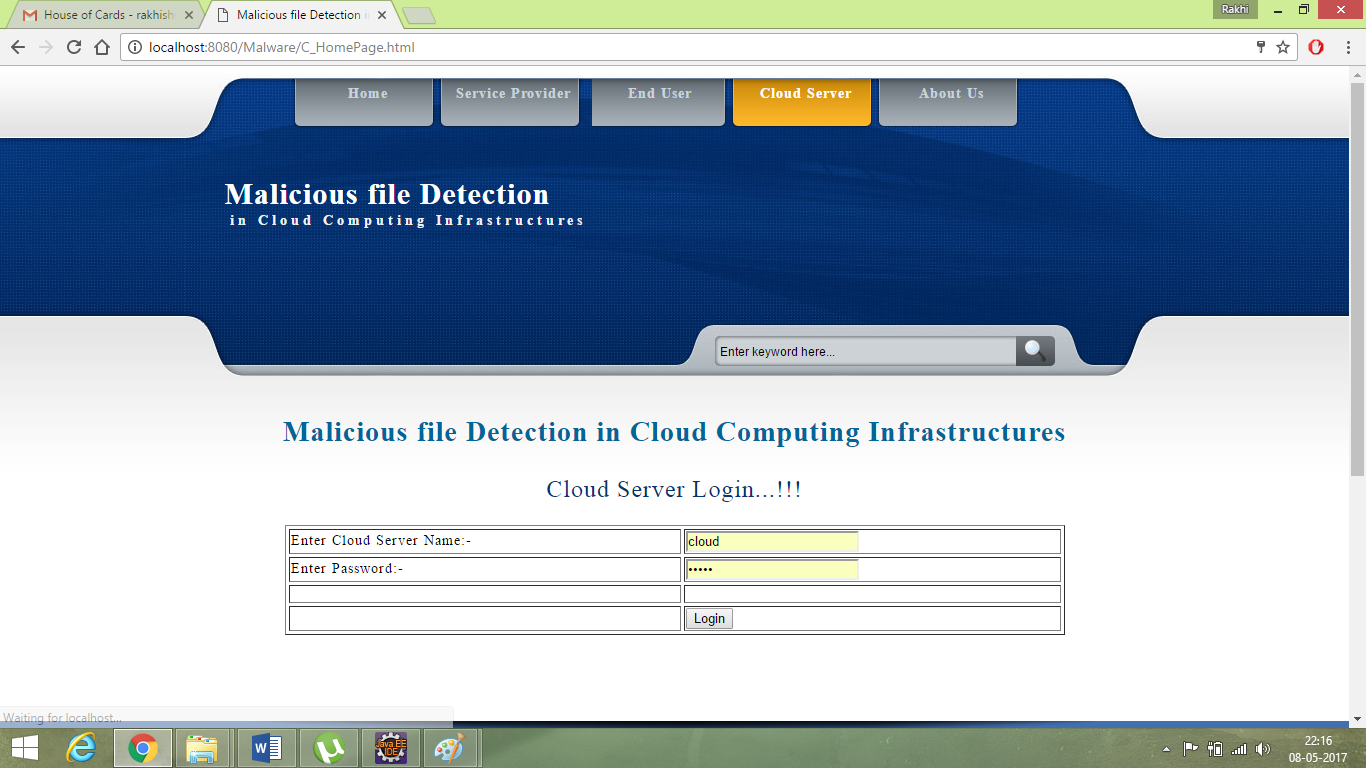


Figure 6.2 Cloud server login

TheFigure 6.2, shows the login page for the cloud server, which requests for the credentials needed. Once the credentials are accepted it logs in to the cloud server.

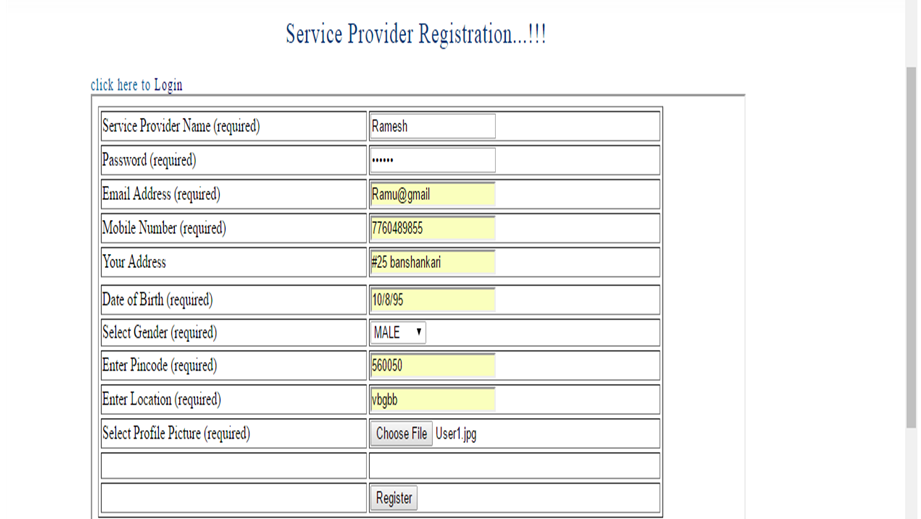


Figure 6.3Registration.

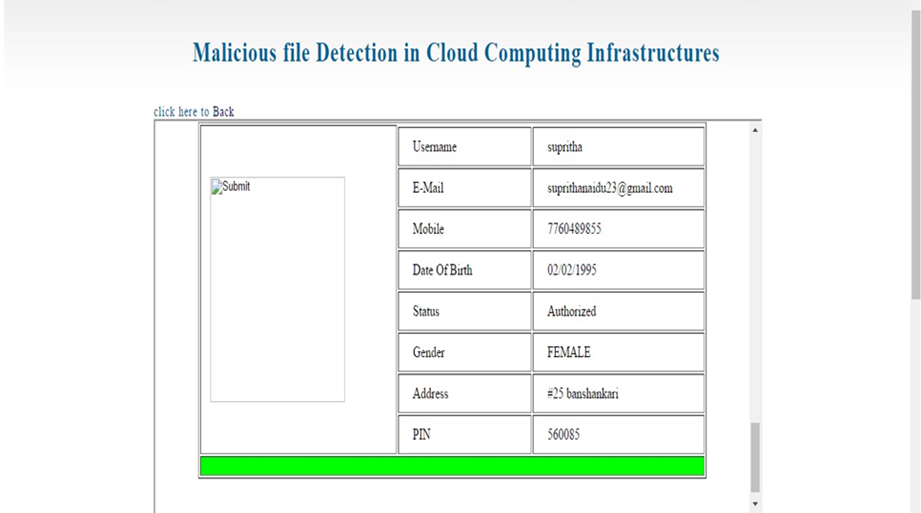
TheFigure 6.3, shows the registration form of the Service provider, where registration happens in order to perform various functionalities. Whichhas to be approved by the cloud.

Figure 6.4 Service Provider Registration

TheFigure 6.4, shows the page in the cloud server which can check the details of the registered service provider or end-user that must be authorized and does the authentication work. The cloud server has to click on the un-authorize button which will then authorize the registered person and the above result will be obtained.

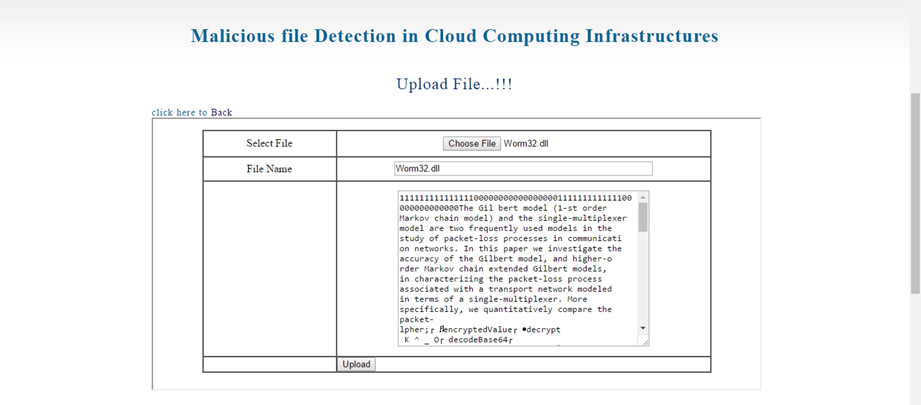


Figure 6.5 File upload

The Figure 6.5 is the screenshot of the file being uploaded by the service provider, which can later be downloaded by end-users.

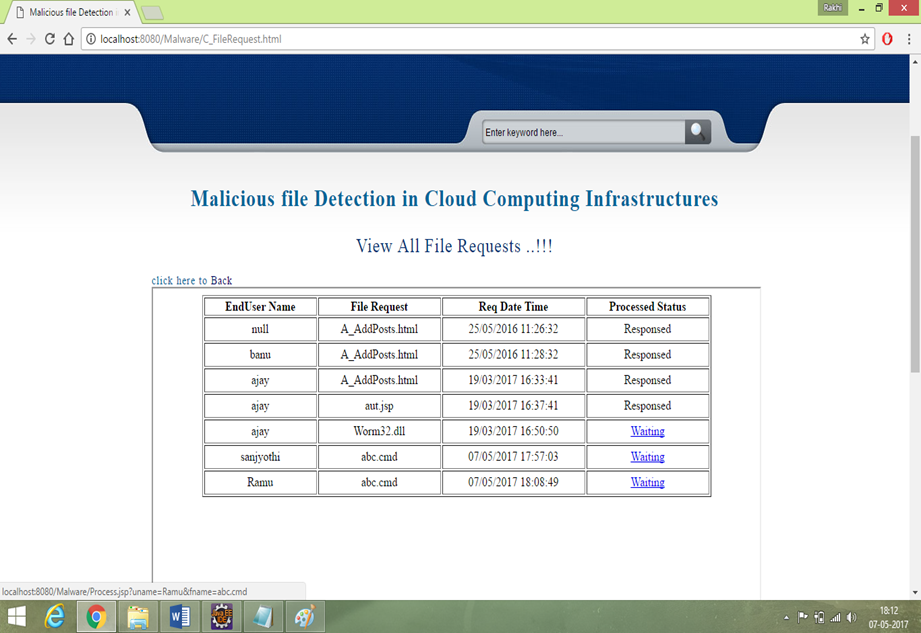


Figure 6.6 File request

The Figure 6.6 is the screenshot where the cloud server can view all the requests made by the service provider while uploading the file. The cloud server can respond to the request or he can keep the file request waiting.

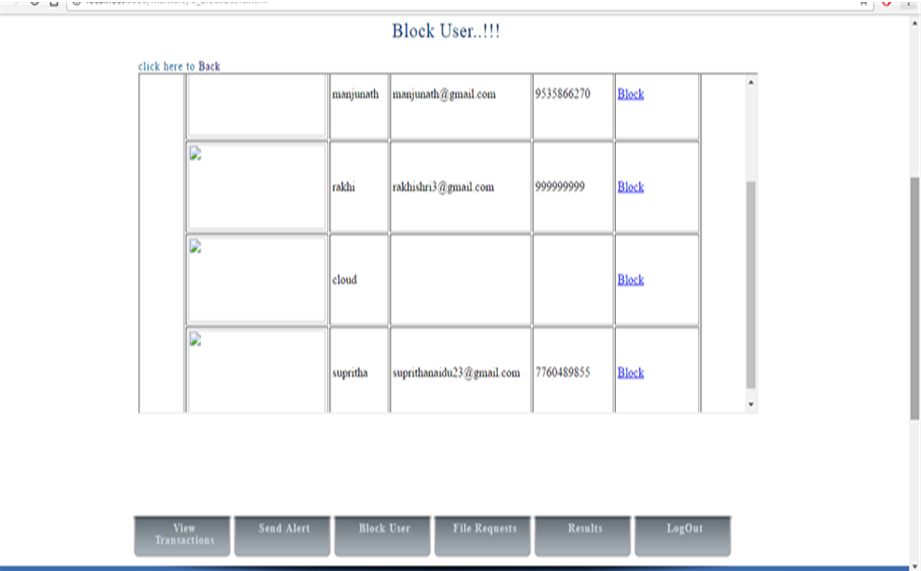


Figure 6.7 Block user

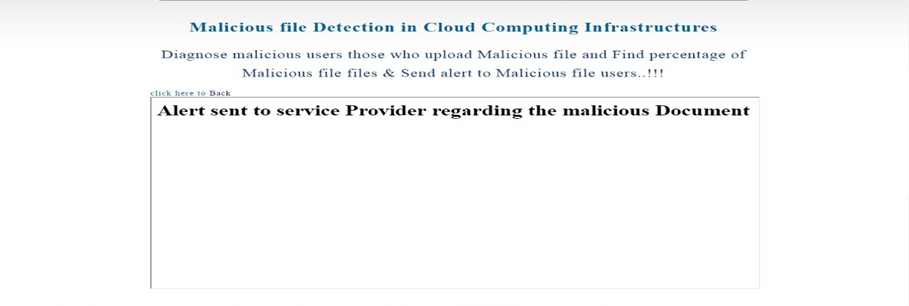
If the service provider uploads a malicious file, then the cloud server has the option of sending the alert or he can block the service provider for uploading the malicious file this is shown in Figure 6.7.

Figure 6.8 Send alert

The file uploaded by the service provider is a malicious file then the cloud server can send an alert messageFigure 6.8 to the cloud server instead of blocking the service provider.

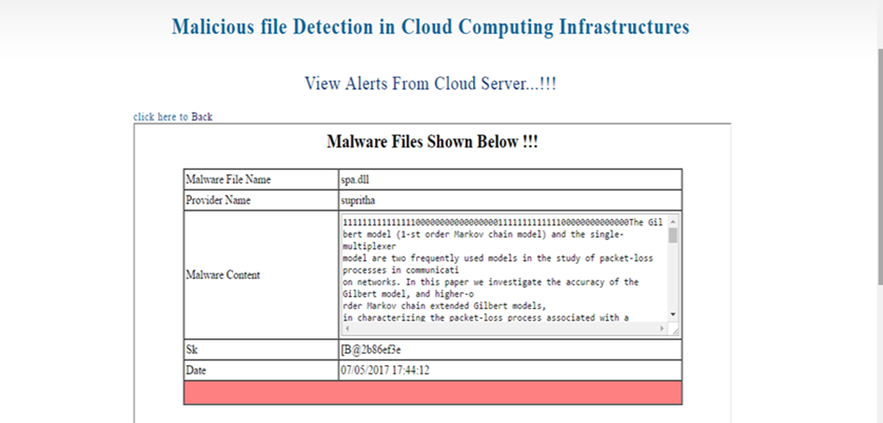


Figure 6.9 View alert

TheFigure 6.9, shows the alert message sent by the end-user can be viewed by the particular service provider that uploaded the file.

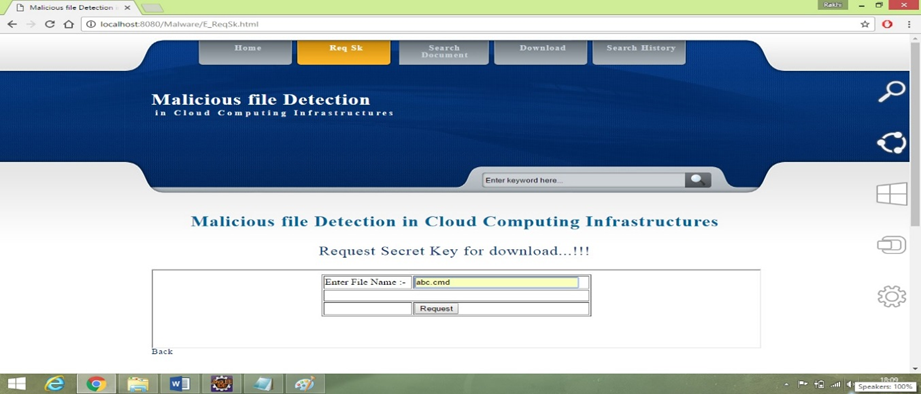


Figure 6.10 Request secret key

In order to download a particular file uploaded by the service provider, the end-user has to first request for the secret key as shown in Figure 6.10. It can be done by entering the file name and by requesting the secret key.

Figure 6.11 Download file

The cloud server sends the secret key of the requested file to the end user as shown in the Figure 6.11. Using this secret key, the end-user will be able to download the file.

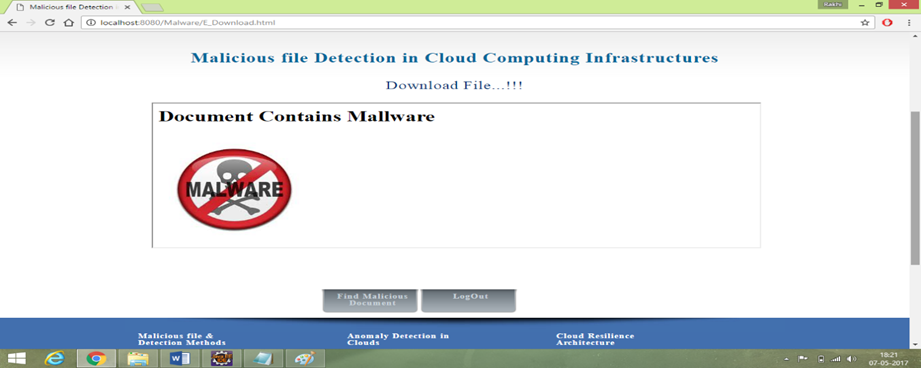


Figure 6.12 Warning message

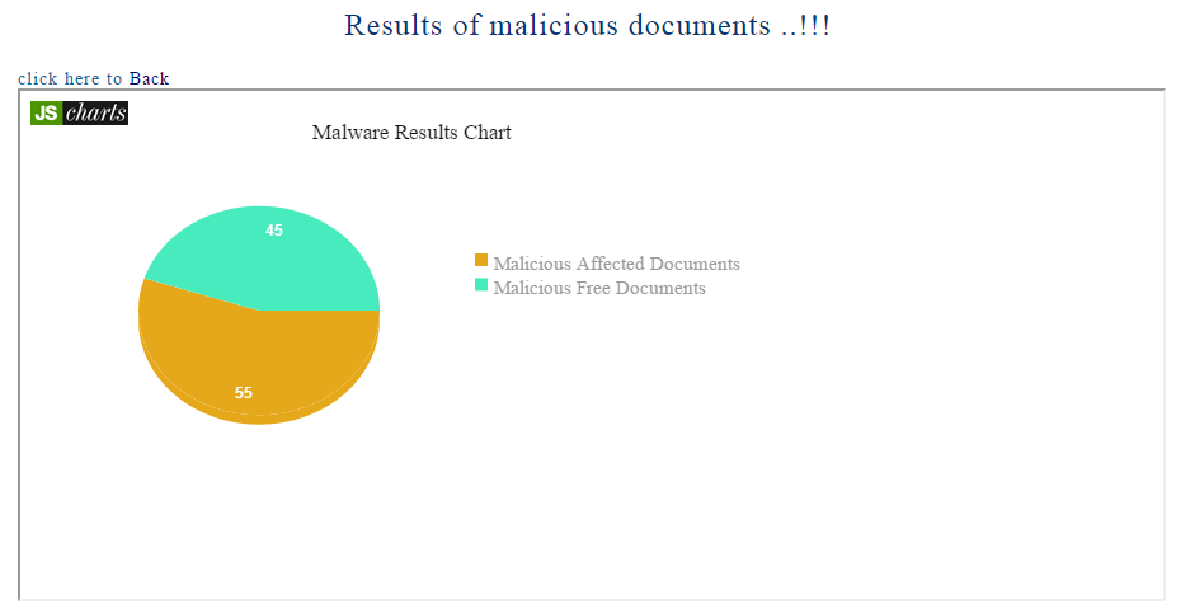
The Figure 6.12 shows the file requested by the end-user is a malicious file then the file cannot be downloaded, a message will popup,whichwill let the end-user know that the file contains malicious document.

Figure 6.13 Pie chart depicting the result

The chart in Figure 6.13 shows the percentage of files that are malicious and the file that are not affected.

# 

## Conclusion and Future Work

### 7.1 Conclusion

The proposed novelty detector is a new detection technique which helps in the detection of the malicious file. The architecture of SVM consists of vectors and graph, whichadopts a novelty method of detecting malicious content in a file. In order to empower the generic properties of our detection approach a secret key generation is implemented using RSA algorithm. The novelty detector offers reasonable security, usability and fits well with some practical applications for improving the detection. Novelty detector hence reduces the probability of downloading the malicious document and provides an estimation of malicious file and non-malicious file.

### 7.2Future Work

The novelty detector currently detects if the uploaded file is malicious or not, however in the future further enhancement can be made, such that it detects the malicious file and not let it be uploaded. The novelty detector can be made to detect wide variety of malicious file, overcoming the current limitation also, security of the system can be increased making the system more efficient.

## References

|  |  |
| --- | --- |
| [1] | c. A Marnedies, “Multi-level network resilience: Traffic analysis, anomaly detection and simulation,” *ICTAC,* vol. 2, no. Next Generation Wireless Networks and Applications, pp. 345-366, JUN E 2011. |
| [2] | P. S. J.P.Roherer J. P. G. Sterbenz, “Resilience and survivability in communication networks,” *Strategies, principles, and survey of disciplines,* vol. 54, no. 8, p. 1245, March 2014. |
| [3] | A. K. Marnerides, M. R. Watson, N. Malware analysis in cloud computing: Network and system characteristics.,” in *MalIEEE Globecom 2013,*, 2013. |
| [4] | M. R. Watson, “Towards a distributed, self-organizing approach to malware detection in cloud computing,” *FIP/IFISC IWSOS,* vol. 7, pp. 256-257, 2013. |
| [5] | W.-H. Chen, “Application of svm and ann for intrusion detection,” *Computers & Operations Research,* vol. 32, no. 10, pp. 2617-2634, 2015. |
| [6] | . Y. Tang, “vms modeling for highly imbalanced classification,” *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions,* vol. 39, no. 1, pp. 281-288, 2009. |
| [7] | P. Bahl, “Interference of multi-level dependencies in ACM SIGCOMM Computer Communication,” vol. 37, no. 4, p. 13, 2009. |
| [8] | W. H. Binsalleeh, “On the analysis of the zeus botnet crimeware toolkit, in Privacy Security and Trust (PST),,” Aug 2010. |
| [9] | . K. Nance, “Forensics examination of volatile system data using virtual introspection,” *SIGOPS Oper. Syst. Rev.,,* vol. 42, no. 3, p. 74–82, 2015. |
| [10] | M. Christodorescu, “Cloud security is not (just) virtualization security,” in *CCSW*, Newyork, 2009. |