8-2018

Cloudarmor: Supporting Reputation-Based Trust Management for Cloud Services

Unmesha Punyamurthula
St. Cloud State University, upunyamurthula@stcloudstate.edu

Cloudarmor: Supporting Reputation-Based Trust Management for Cloud Services

by

Unmesha Punyamurthula

A Starred Paper

Submitted to the Graduate Faculty of

St. Cloud State University

in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in Information Assurance

August, 2018

Abstract

Cloud services have become predominant in the current technological era. For the rich set of features provided by cloud services, consumers want to access the services while protecting their privacy. In this kind of environment, protection of cloud services will become a significant problem. So, research has started for a system, which lets the users access cloud services without losing the privacy of their data. Trust management and identity model makes sense in this case. The identity model maintains the authentication and authorization of the components involved in the system and trust-based model provides us with a dynamic way of identifying issues and attacks with the system and take appropriate actions. Further, a trust management-based system provides us with a new set of challenges such as reputation-based attacks, availability of components, and misleading trust feedbacks. Collusion attacks and Sybil attacks form a significant part of these challenges. This paper aims to solve the above problems in a trust management-based model by introducing a credibility model on top of a new trust management model, which addresses these use-cases, and also provides reliability and availability.

Table of Contents

List of Figures

Chapter I: Introduction

This paper mainly deals with the way consumers interact with cloud service providers. Security is one of the most significant concerns in a cloud environment based on the set of features they provide, some of them are as follows:

1. Highly dynamic and distributed in nature.
2. They are loosely coupled systems and most of the time do not need to maintain any state.
3. These services are highly competitive with the set of features they provide to consumers (so there is much selection for the consumers).

For the rich set of features provided by cloud services, consumers want to access the services while also protecting their privacy. In this kind of environment, cloud services protection will become a significant problem because of various attacks in the cloud networks. So, a system which lets the users access cloud services (without losing the privacy of data) with a trust and identity model is needed, so that communication and cloud services are protected.

Problem Statement

Based on the multiple features and services cloud servers are providing on a day-to-day basis and the increase in demand for such services in the recent few years, there has been cutthroat competition in the similarity of the services which multiple cloud providers are providing.

With most of the features and services from the competitors being alike, consumers have preferred the cloud service providers who are offering better security standards.

Some of the main concerns in this model are as follows:

1. Protecting consumers' privacy in such a distributed environment, without compromising on the set of services they can access.
2. Protecting cloud services from malicious users using security policies and comparing them based on the valid set of features they provide.
3. Providing the above mentioned by providing a trust management service and working on the availability of such services and networks.

So, regarding offering trust-based security for the consumers to access cloud services, the three significant sub-problems that need to be solved are:

1. Consumers' Privacy Protection
2. Cloud Services Protection
3. Trust Management Services Availability

Nature and Significance of the Problem

To further understand the impact of these problems in the current world, let us consider the use-cases from the cloud services and networks.

1. When consumers are accessing a cloud service, it is very critical that their privacy be protected. There can be multiple points of communication failure, which will give attackers confidential information about the consumer. Also, several kinds of attacks can be designed once the attackers can understand the type of services that consumers are accessing; the set of features they are interested in these services; the number of times consumers are spending accessing these services, and the type of access consumers are using. Once the attacker is aware of any of these details they will be

able to construct attacks such as a Sybil attack, which will restrict users' accessibility of services.

2. Service protection is something that should be included in the design of the service by the provider rather than an additional work that needs to be done after the completion of the service. Cloud services are attacked very often to compromise their availability or to leverage weak points which can be exploited for long-term system abuse. Also, once a service architecture and the loopholes in it are misused, there are attacks like collusion attacks that can be launched, which will severely compromise the availability of the service.

Cloud computing attempts to provide near infinite computing resources on-demand due to its high scalability. It eliminates the needs for consumers to plan far ahead for the hardware provisioning part of it. Many large companies, which operate in the cloud domain, are accelerating their pace in developing cloud computing systems and enhancing its services, thereby increasing their potential consumer base.

As cloud computing refers to not only software being delivered as service but also the virtual hardware that provides those services and based on the research presented, security and privacy complaints offered by the organizations that provide these are not sufficient and are becoming a bottleneck for the user to adapt to the cloud computing environments. Consequently, concerns about security issues, for example, accessibility, secrecy, information trustworthiness, control, and review ought to be considered.

On-demand cloud services are software applications that are running in the cloud computing infrastructure. Cloud computing allows providers to develop, deploy, and run applications that can quickly scale, perform better than average running speeds, and rarely fail, without any concerns about the infrastructure they are running on.

Cloud computing systems can achieve the following goals:

1. Availability: Availability refers to making the cloud computing systems (including applications and its infrastructure) usable anytime and anyplace irrespective of the consequences. Since the cloud computing systems are based on the web, the services should be accessible anywhere from the Internet (if possible only with a thin client). It should hold for all cloud computing services such as Software as a Service, Platform as a Service, Infrastructure as a Service and even Database as a Service. Since the services are available across the Internet and there is no specific time in which these services are available. They should be possible to any number of on-demand users at any time to make sure of the availability of the infrastructure and the software hosted on it. There can be two strategies that are used; they are hardening and redundancy.
2. Confidentiality: This refers to maintaining the users' privacy when they are using the services hosted in the cloud systems. Isolation of the cluster within the cloud domain or application of encryption algorithms (cryptographic algorithms) are the standard approaches extensively used by the cloud computing vendors.
3. Data integrity: Given the three main security challenges for cloud (multi-tenancy, divided responsibility, and dynamic environment) – one specific customer concern can be data protection, including access control, encryption, integrity and origin verification of the data. In cloud environments, the amount of data at rest, in transfer

> and use is considerably more substantial than in traditional networks (Ericsson White Paper, 2015).

Data protection requires support from the other security functional groups in the different phases of the data lifecycle management. Data protection takes care of secure data lifecycle management in the multi-tenant environment. It covers confidentiality, integrity and the availability of data at rest, in transfer and use (Ericsson, 2015)

Security monitoring and analytic functions play an essential role in data protection. Security-related events are collected from networks and nodes and correlated against indicators of compromise together with the system data. These functions provide both live security status (reactive response) and information on past security events, making it possible to respond to threats even before a new incident occurs (predictive response). On the other hand, when incidents occur, data provided by the security analytics shortens the time consumed in investigations and forensics (Ericsson, 2015).

Data in transit requires traffic separation, protection, filtering, and integrity protection that belong to the network and infrastructure protection controls. Infrastructure trustworthiness assurance and secure disengagement are based on equipment established trust and secure bootstrapping systems (Ericsson, 2015).

> Let us consider the need for a user to ensure the integrity of this data at all times. Data integrity in a cloud context is paramount in building enough trust for a user to put data in the hands of a cloud provider. Unfortunately, data integrity often does not receive as much attention as data encryption. (Ericsson, 2015, p. 7)

Integrity can guarantee the consistency of fundamental information, regardless of whether it is framework information or application information.

Most security-cognizant clients need affirmation of any of the more significant part of the accompanying:

- Time: When the information was created or ensured.
- Integrity: Assurance that information has not been transformed from its original frame.
- Origin check: The personality of the information maker, which can be particularly applicable amid legal sciences examinations, in encouraging crime scene investigation auditing, or for legitimate and authoritative reasons for existing is guaranteed.

These properties can be achieved using, for example, symmetric cryptography, Public Key Infrastructure (PKI), or Keyless Signing Infrastructure (KSI) (Friedman, Resnick, & Sami, 2007). These techniques can be used separately or can complement each other, and the selection of technology should be based on the use case and user requirements. PKI-based signing technology utilizes public-private vital pairs, while KSI technology utilizes data hashes and hash trees for generating and publishing a root hash for the data to be integrity protected (Ericsson, 2015). Integrity verification is done using signature tokens that enable verification of data relative to the previously published root.

Which technology to select for data integrity protection can be based, for instance, on the following aspects:

- The period of need for integrity protection (short-term versus long-term).

- The type of time information needed;
- The number of data items that need to be protected;
- The level to which the origin needs to be verifiable;
- Other organizational policies that, for example, mandate the use of specific cryptography.

Symmetric cryptography and KSI have benefits in the cloud or meaningful data contexts regarding efficiency. In KSI, integrity does not rely on a single key that could be breached: no key is needed to verify if data matches the root hash. Security management is also facilitated in KSI since there is no need for revocation. By combining data integrity and data encryption technologies with other security orchestration and management tools, unique use cases could be demonstrated including near real-time data integrity monitoring. Telecommunications cloud providers would be able to transparently demonstrate that cloud tenants own and control their data at all times (Ericsson, 2015)

Maintaining data integrity is a fundamental task when providing cloud services, if data gets modified by unauthorized entities, then it will lead to catastrophes for both the vendors and the customers in the cloud domain.

- Control: This means to monitor and regulate the use of services by the customers within the cloud domain.
- Audit: This is the in-depth monitoring of what is happening in the cloud system, related to both providing services and using services. Auditability refers to capturing evidence to provide information on the activities that occurred when a cloud service is running in the domain. This itself will be provided as a service so that the

underlying service providers can leverage this (much like a platform feature) and will give a sense of what happened in the systems and help administrators take appropriate actions based on the results.

Though there are many ways in which the problems above can be solved, one of the solutions is to have a trust-based management service, which has a feedback loop for maintaining the trust between the consumers and the providers so that that cloud services can be secured. Though a trust-based management service will solve the problems, if there is just a single service doing all of this for the entire cloud network, then it will be a single point of failure if attackers can exploit such a service with multiple attacks (for example a Sybil or collision attack).

The Objective of the Study

One of the primary objectives of this study is to provide methodologies and solutions to the problems as mentioned earlier. There are multiple interweaving solutions to these individual problems, but there are no better approaches to solve these problems together along with providing availability than the trust-based management approach. The primary objective, therefore, is to improvise on the solutions so that they can be combined and applied to a cloud architecture so that these security concerns can be resolved.

As stated earlier, this paper mainly concentrates on the three problems relating to accessing of cloud services, they are:

- Consumer Privacy Protection
- Cloud Services Protection
- Trust Management Services Availability

Since this paper solely focused on the issues stated, everything that is falling out of context regarding network or service is out of context for this paper and is assumed as a stable system.

Some of the things that are not covered by the working of the network, or answering theoretical questions are as follows:

1. There are no node level issues that are happening or all the nodes in the system are stable, and no malicious programs can arise from the stable systems.
2. Networks and services are not going down with the normal functioning of the systems, and networks and function are considered stable unless they are under attack.

Chapter II: Background and Review of Literature

Introduction

A typical solution to the security of the cloud network which provides services and contains consumers which access these services is a Trust Management Service (TMS) that manages the trust between various providers and services.

This kind of architecture contains various components such as:

- Cloud service provider layer
- Consumer layer
- Service advertisement through the Internet
- Service discovery through the Internet, and
- Interactions among all of these components

One illustration of such a system is shown in Figure 1.

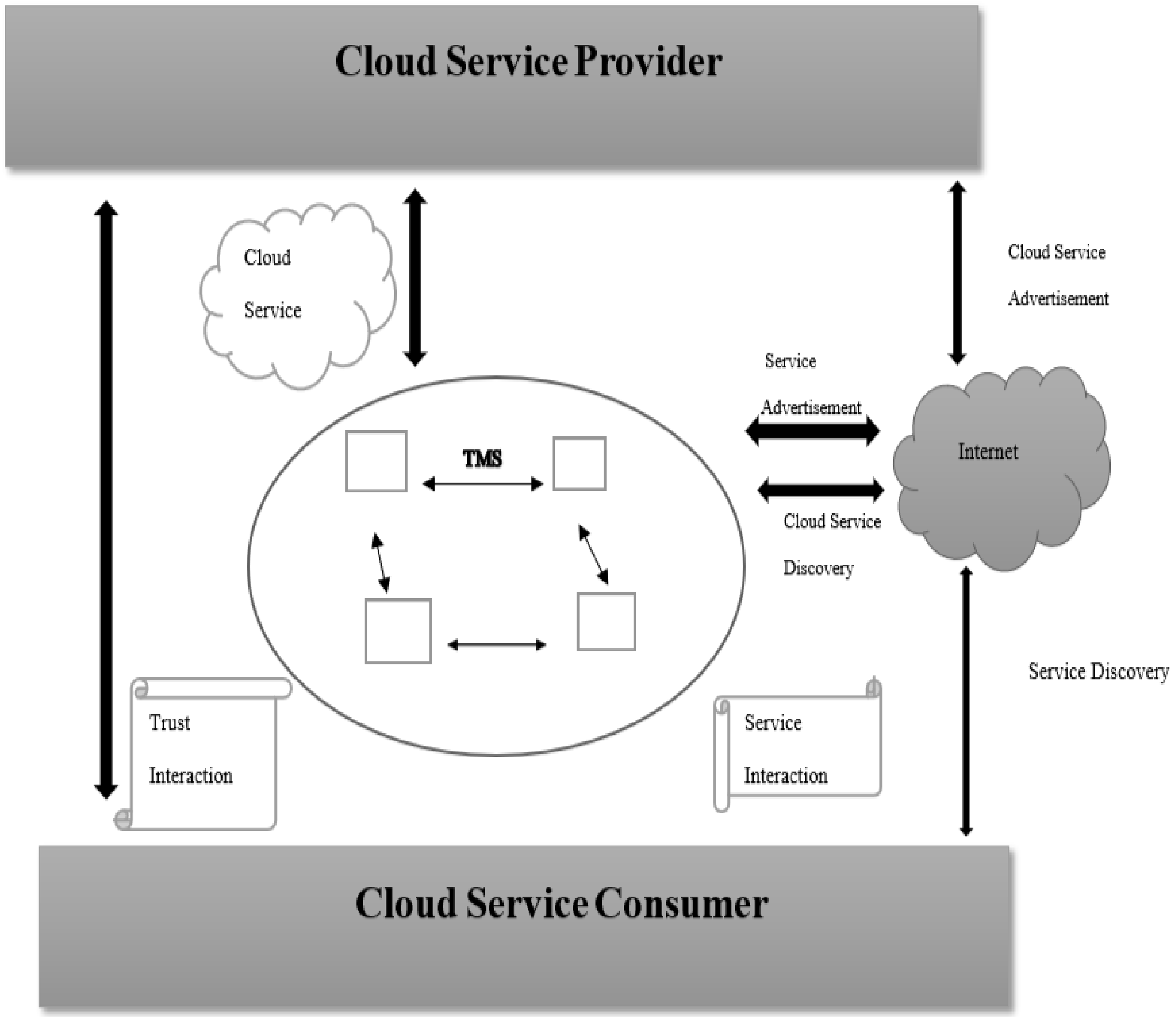


Figure 1. Cloud Service Consumer Architecture

Definitions of Terms

Cloud Service Provider. A Cloud Service Provider is an organization that offers some part of cloud computing ordinarily Infrastructure as a Service (IaaS), Software as a Service (SaaS) or Platform as a Service (PaaS) to different organizations or people. Cloud suppliers are now and then termed to as cloud specialist co-ops or CSPs.

Cloud Service Consumer. A Cloud service consumer is a short runtime part accepted by a software program when it gets to a cloud benefit gave by any Cloud Service Provider. Regular sorts of cloud benefit consumers can incorporate programming projects and services able to do remotely getting to cloud services with distributed service contracts, and workstations or PCs. Mobile devices running software capable of remotely accessing other IT resources positioned as cloud services or even thin clients running on any piece of hardware that can upload or download data can also be a cloud service consumer.

Cloud Service Interaction. A service interaction involves communication between the cloud service provider and consumer. A typical interaction is to use services provided by the cloud service provider. The consumer can be in different layers or even a hierarchical manner, where one consumer can get results from another consumer, and once the top of the tree is reached, the first consumer will interact with the service.

Cloud Service Advertisement. Advertising in very generic terms is a means of communication with the users of a product or service. Advertisements are messages paid for by the people who send them and are proposed to exhort or impact people who get them. Cloud Service Advertisement refers to advertising of cloud services as products to the consumers, so that consumers can discover the services.

Cloud Service Discovery. Cloud Service discovery refers to the process of discovering cloud services that are advertised by the providers of cloud services.

This is different from standard web service discovery as Web services use WSDL (Web Services Description Language) or USDL (Unified Service Description Language) to expose their interfaces and use UDDI (Universal Description, Discovery, and Integration) to publish their services to service registries for discovery, and there is no standard description for cloud services. For solving this, there is a cloud description ontology model that helps us in the cloud discovery process as described by (Karlof & Wagner, 2003).

Internet. The Internet, in some cases called basically “the Net” is an overall arrangement of PC organizes, a system of systems in which clients at any one PC can (in the event that they have authorization and the computer is connected to the network) get information from any other computer connected to “the Net.” Also, it can be used to talk directly to users at other computers, when all such interacting computers are within a network which is connected to the Internet (Course Hero, n.d.).

Trust Management Service. A Trust Management Service (TMS) provides the interface layer between cloud service consumers and cloud service providers for effective trust management. In particular, the trust management service spans across a distributed network with a sufficient number of nodes that expose the service interfaces, so that the consumer can give their feedback without worrying about a single endpoint to call and avoid a Single Point of Failure (SPOF). The providers can use or inquire about the trust results, and the providers can use this feedback and discuss on the path forward.

Also, as stated in Lai, Feldman, Stoica, and Chuang (2003), in distributed environments that have a reputation-based trust management framework, the service can enable features which allow one party to evaluate the trust of another party based on the feedback that another party has received during its previous interactions with some other parties. The reputation of another party can determine whether or not it meets a minimum trust threshold for future interactions. For example, many participants in online file sharing systems can decide whether or not to share files between various servers within the network or to trust a particular file server or not for downloading the files from it. Over the past few years, many reputation-based trust management systems have emerged for applications ranging from e-commerce to Web service selection.

Previous works on reputation systems have explored the performance, efficiency and the strength of various reputation scoring functions as well as algorithms for reputation management in decentralized environments. However, not much attention has been given to supporting the processing of feedback from multiple entities while also supporting the use of various reputation scoring capacities by various substances over similar feedback information. Such flexibility, variability and not fixing to a single scoring function or algorithm in choosing reputation scoring functions is thought to be desirable in an infrastructure-centric environment where many cloud service providers are hosting many services with different requirements for trust.

Because of an infrastructure hosting many different services by multiple Cloud Service Providers, each provider might have different reputation-based requirements, implementations and the metrics that the provider would like to collect. For example, consider an infrastructure is hosting many different services. Moreover, in turn, this can differ based on the clients, who are consuming the service. Each service provider may want to apply requirements, implementations, and metrics based on the service and client pair and these, in turn, need scalability. Given these aspects, the system can have infrastructure-centric environments, where distributed networks are not entirely treated as decentralized networks, rather reputation-based trust management is offered at the infrastructure level, which gets replicated across the network in all of the nodes.

A new Trust Management Framework is designed and implemented to take into account the factors of trust and reputation of cloud service consumers who were interacting with the cloud services. These trust factors are stored, analyzed and used in forming new interactions for those clients with the services. The trust management framework stores feedback on previous service interactions with clients and allow services are computing their own customized reputation scoring functions over the feedback collected.

Specifically, the following features are desired for the new version of Trust Management Service:

1. Support for multiple trusts and reputation evaluation methodologies: As discussed previously, different services might apply different trust evaluation methods, concerning the scoring functions with the same trust and feedback-based data. So, a trust management service that supports numerous notoriety scoring capacities over a similar trust-related input shared from different administrations is created. Trust Evaluation Caching: Caching is a concept where something which is retrieved from an external source is kept locally, to not query the external source repeatedly within the prescribed query time, or until the data on the external source is changed. It reduces communication overhead in the trust management framework and further reduces the network load in such a framework. Cache techniques like Bloom histograms, CacheMere, MemCache were developed and tested for trust evaluation results; these have shown to provide significant performance improvement over a theoretical Trust Management framework.
2. SDK (Software Development Kit) for application developers: The SDK toolkit will provide access to the underlying data stores in the Trust Management Framework where the reputation data is stored, and various analyzing techniques were

implemented. The trust management service frees application developers from writing their trust management software components.

3. Service implementation, deployment, and evaluation: The new trust management service has been implemented and deployed in both Local Area Networks (LANs) and Wide Area Network(WANs) and other distributed environments comprising several nodes with a realistic service-oriented architecture application scenario.

Trust Interaction. Trust interaction refers to interaction with the trust management service to either submit feedback or get feedback on particular entities within the network. In a typical interaction of the reputation-based TMS, a user either gives feedback regarding the factor of trust in a particular cloud service or requests the trust assessment of the service. Trust management is an approach to deal with survey and build up trusted connections. A few methodologies have been proposed for overseeing and getting to the trust score given alternate points of view.

Two different perspectives in the new Trust Management Framework, namely: (a) Service Provider Perspective (SPP) (b) Service Requester Perspective (SRP).

In the Service Provider Perspective, the service provider is the primary driver of the trust management system where the trust factor of the service consumer is assessed. On the other hand, in the Service Requester Perspective, the service requester assesses the trust factor of the service provider.

Trust Management Techniques. Different trust management techniques have been reported in the literature, which can be classified into four different categories: (a) policy, (b) recommendation, (c) reputation, and (d) prediction (Vyshnavi & Yadav, 2016).

To ease the discussion, clarification of these trust administration systems utilizing the service requester point of view (i.e., the cloud service consumers' viewpoint) is required. Similar systems can be connected to the next point of view (i.e., cloud service providers' viewpoint). Cloud service consumers and suppliers identify with lines speaking to trusted relations between them.

Figure 2 depicts the four trust management techniques. Cloud service consumers and providers relate to lines representing trusted relations between them.

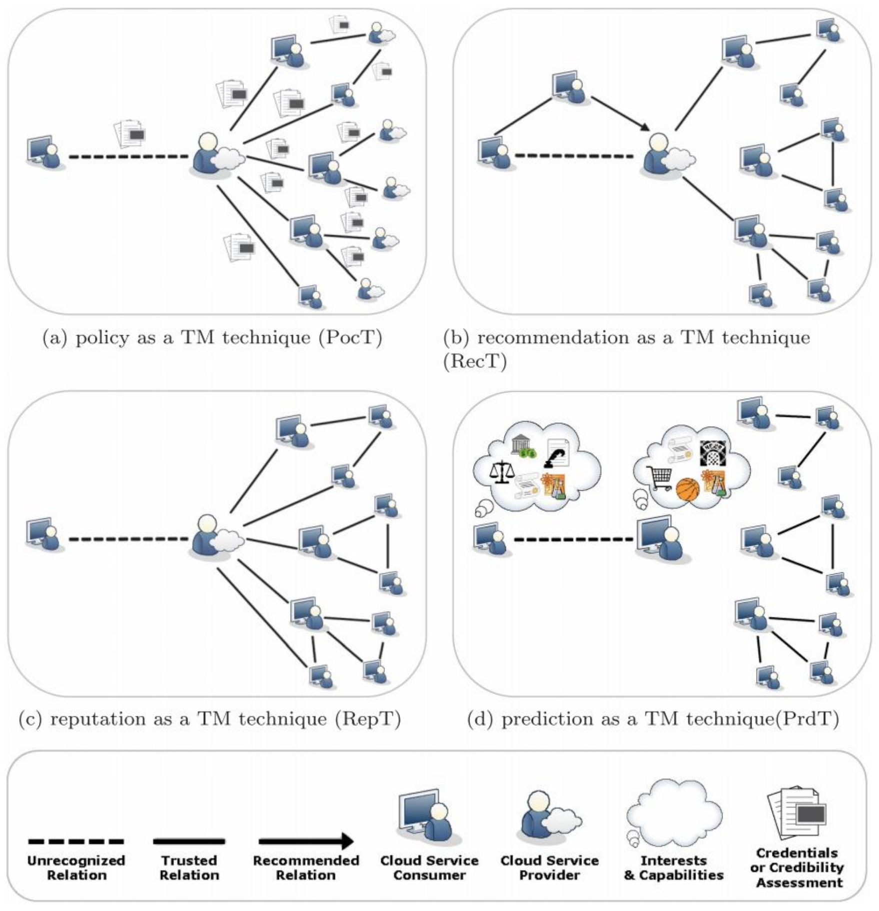

(a) policy as a TM technique (PocT)
(b) recommendation as a TM technique (RecT)
(c) reputation as a TM technique (RepT)
(d) prediction as a TM technique(PrdT)
Unrecognized Relation
Trusted Relation
Recommended Relation
Cloud Service Consumer
Cloud Service Provider
Interests & Capabilities
Credentials or Credibility Assessment

Figure 2. Trust Management Techniques (Noor, Sheng, Yao, Dustdar, & Ngu, 2015)

Policy as a trust management technique. One of the most straightforward ways in which a trust among the parties can be established in a cloud environment (Molka, Boukadi, & Ben-Abdallah, 2015), a peer-to-peer system or a grid is policy-based trust management. This technique has a set of policies, with a definite trust score configured for each policy and each policy controls the level of authorization. This type of configuration of Trust Score and Authorization Level is called a role. Each consumer who would like to access the cloud service has to assume this role to access the service.

The trust scores are set based on various approaches that are based on the past trust results, which are calculated when consumers are using the cloud service, or the set of permissions that users invoke for interactions. One of the approaches that can be used is the auditing which is captured during the interactions. Monitoring this audit data and setting definitive thresholds will define if a particular policy is met or not and the consumer can assume a what role. Another approach can include the amount of credibility given by the peers when they are interacting or the feedback mechanism which is collected from time to time based on the performance of nodes in the network.

The trust score threshold can be considered as a service plan (i.e., where the service level agreement is specified) and penalties can be assigned to the cloud service organization if there is a service level infringement in the provisioned cloud services. Violation by a cloud service provider is just an example; the same can apply to the consumer of the service, as well as the nodes in between. Further, this trust score threshold-based service agreement can apply to the trust management service node itself. The violations in the service agreement can be tracked with strict causes, thereby having a tracker on the number of violations and also be aware of the causes of each of those violations.

Also, as mentioned, real-time credibility can also be input in defining the trust score threshold. It can be measured by multiple factors such as the response to the security pings from the node, response time, total availability time of the node, and so on. Some of these are qualitative measurements, while others are quantitative as well. The feedback validity can be estimated utilizing a few factors, for example, the cloud service customers' experience including the nature of the input which differs from one person to another. Many researchers identify two features of credibility including trustworthiness and expertise (Balachandran & Sanyal, 2012). Regarding the approach based on the permissions of the nodes when interacting with other nodes, there can be a single sign-on (SSO) approach where the credentials disclosure and authentication happen once and afterward the cloud service consumer have an access approval for a few cloud services. Alternatively, the state machine approach can be used where the credentials disclosure and authentication happen once, and after that, the cloud service consumers have an access endorsement for a few cloud administrations. Credentials can be distributed based on various approaches such as sharing a secret private key or having a credential pair of a public key and private key. In each of these approaches, there are multiple protocols and methodologies present to distribute each key based approach. One such approach is based on certificates, where a central third party is present, and it distributes the certificates, and each entity that is under the interactions should regularly be in sync with the certificate provider so that their interactions with the service are uninterrupted.

Recommendation as a trust management technique. The other approach for trust management is to have a recommendation as a trust management technique (RecT) in the context of Internet dominated by service-oriented environments. The view of RecT comes from the social norms of everyday life, where a particular party knows the source of trusted feedback. This is much like everyday examples like a friend referring another friend to a restaurant that is better for their group, or referral to another brand which they use. Also, with recommendations, users can proudly display the recommendation given in their metadata information, which will allow the Trust Management Framework to make a better decision.

Recommendations can appear in different forms such as the direct recommendation or transitive recommendation. A direct recommendation happens when a cloud service consumer recommended a particular cloud service to their well-established and trusted relations or friends. A transitive proposal happens, then again, when a cloud service consumer believes a specific cloud benefit because of no less than one of their trusted relations trusts in the service. One interesting thing with transitive recommendations is that each transitive recommendation can be weighted based on the consumer it is coming from and also the path length from the consumer taking the decision to the consumer who recommended it directly. So, this becomes more like a graph problem with weighted edges.

Reputation as a trust management technique. Reputation as a trust management technique (RepT) is essential because the feedback of the various cloud service consumers can dramatically influence the reputation of a particular cloud service either positively or negatively. It is like direct customer reviews on the product or service, given directly by the end users.

However, one specific problem with this kind of reputation-based approach is that some reviews cannot be verified.

A simple example of this use case is checking the rating of movies online unless the viewers of the movie have been verified, it cannot be taken accurately as a right of reputation. Another example in which this has been resolved in a reputation-based system is browsing through e-commerce pages with customer comments on items, which they have bought (an example is a “verified purchase” on amazon.com) can have a direct or indirect influence on the trustworthiness of a particular entity.

Prediction as a trust management technique. Prediction as a Trust Management Technique (PrdT). Prediction as a trust management strategy is extremely valuable, mainly when there is no earlier data concerning the cloud services’ connections (e.g., past collaborations or history records). PrdT has been proposed in both the cloud environment and the service-oriented environment. The essential thought behind PrdT is that comparable disapproved entities, for example, cloud service consumers will probably trust each other.

Background Related to the Problem

Security has been and will remain one of the significant concerns of cloud services. Till all the security aspects of cloud computing have proper methodologies to deal with all of the security concerns, it will remain a significant bottleneck for the advancement regarding both development and adoption.

Data service outsourcing security. As companies and individuals produced and will continue to produce more and more data that has to be stored and later utilized to make the analysis. Rather than relying on the storage, they have on-site, they are driven to use cloud computing resources for storing and utilizing their data due to its on-demand service and also active regarding pricing and durability. However, once clients never again physically have their information, its privacy and integrity can be insecure.

For the previous concern, information encryption before outsourcing is the least complicated approach to ensure information protection and spontaneous battle access in the cloud and beyond. In any case, encryption additionally makes sending general information usage services, for example, a plaintext keyword search over printed information or question over databases a troublesome task. The small arrangement of downloading every one of the information and unscrambling it within is illogical, because of the enormous data transmission cost coming about because of cloud-scale frameworks. Besides, besides terminating local storage management, putting away information in the cloud fills no need unless individuals can undoubtedly seek and use that information.

This issue on the best way to look encoded information has as of late picked up considerably and prompted the improvement of available encryption procedures. At a higher degree, an available encryption conspire utilizes a prebuilt encrypted seek file that lets clients with suitable tokens safely look through the encrypted information employing watchwords without first decrypting it. However, thinking about the conceivably vast number of on-request information clients and the broad measure of outsourced information documents in the cloud, this issue is still especially tricky because gathering performance, system usability, and scalability requirements are challenging. In this specific situation, various intriguing yet difficult issues remain, including the similitude of searches over encoded information, and the protected positioned look over scrambled information, secure multi-keyword semantic search, secure range query, and even secure search over non-textual data such as graphical or numerical data.

Another urgent issue that emerges while outsourcing information to the cloud is ensuring information uprightness and long-haul storage correctness. Even though outsourcing data to the cloud is financially engaging for a long haul, largescale capacity, it does not promptly ensure data integrity and accessibility. This issue, if not appropriately tended to, can block the successful organization of a cloud architecture. Given that users no longer locally possess their data, they cannot utilize traditional cryptographic primitives to protect its integrity. Such natives more often than not require a nearby duplicate of the information for trustworthiness confirmation, which isn’t reasonable when storage is outsourced. Besides, the substantial measure of cloud information and the clients’ obliged computing abilities make data integrity inspecting in a cloud domain costly and imposing. Thus, empowering a brought together capacity inspecting design is vital for this developing cloud economy to wind up completely settled, and clients will require approaches to survey risks and pick up trust in the cloud.

From framework ease of use perspective, such a plan ought to cause extremely restricted examining overhead as far as calculation and transfer speed and besides join cloud information’s dynamic highlights and safeguard clients privacy when a specialized third-party auditor is introduced. Above capacity accuracy, other security issues emerge identified with distributed storage administrations. One critical security idea is verification of proprietorship. This system plans to keep the introduction of client information by employing side channels that outcome from cross-client de-duplication, which is broadly used to spare the space and data transmission CSPs require. Other testing security issues incorporate guaranteed information deletion and remote appraisal of adaptation to internal failure, that is the remote identification of hard-drive failure vulnerabilities in the cloud challenging security problems include assured data deletion and remote assessment of fault tolerance, that is the remote detection of hard-drive failure vulnerabilities in the cloud.

Computation outsourcing security. Another major administration empowered inside the cloud worldview is computation outsourcing. By outsourcing workloads to the cloud, clients' computational power is never again restricted by their asset compelled devices. Instead, they can enjoy the cloud is near unlimited computing resources in a pay-per-use manner without committing any substantial capital outlays.

However, current outsourcing practices operate in plaintext, that is it reveals both data and computation results to the commercial public cloud. It can raise significant security concerns, mainly when the outsourced computation workloads contain sensitive information, such as a business's financial records, proprietary research data, or even protected health information. Moreover, the cloud's operational points of interest are not sufficiently straightforward to clients. Therefore, different inspirations can make the cloud carry on unfaithfully and return inaccurate outcomes. These range from possible software bugs, hardware failures, or even outsider attacks to cloud servers deliberately being "lazy" to save computational costs. Consequently, we are in high need of secure calculation outsourcing systems to both ensure delicate workload data and guarantee that the calculation comes about came back from the cloud are right. This undertaking is troublesome, be that as it may, because of a few difficulties that the instrument configuration must meet all the while. Of, to begin with, such a system must be achievable as far as multifaceted computational nature. Something else, either the clients' cost can turn out to be restrictively vast, or the cloud will not have the capacity to finish the outsourced calculations in a sensible measure of time. Second, it must provide sound security guarantees without restricting system assumptions. Namely, it should strike a right balance between security guarantees and practical performance. Third, this system must empower generous computational reserve funds at the client side contrasted with the measure of exertion required to tackle an issue locally. Otherwise, users have no reason to outsource their computation to the cloud.

A current achievement in fully homomorphic encryption (FHE) has demonstrated the far-reaching consequences of secure calculation outsourcing to be suitable in theory. In any case, applying this general framework to conventional figuring assignments is still a long way from down to earth because of FHE activities' having a to a significant degree high many-sided quality, which can't yet be taken care of practically speaking n an alternate front, researchers are dealing with instruments for particular calculation outsourcing issues. On the other front, researchers are taking a shot at components for calculation outsourcing issues, for example, direct programming utilizing issue change, genomic calculation utilizing figuring portions, and viable. For example, linear programming utilizing issue change, genomic calculation utilizing specific calculation allotments, and productive confirmation of vast scale biometric calculations, all of which ought to provide much more practical effect than the more general solutions currently available.

Access control. In numerous application situations, for example, those in enterprises or organizations, clients' entrance to information is typically specific and profoundly separated. Various clients appreciate unique access benefits concerning the information. At the point when information is outsourced to the cloud, upholding secure, proficient, and dependable information access among countless users is crucial.

Customarily, to control the scattering of private information, clients set up a trusted server to store information locally free, and afterward control that server to check in the case of asking for clients have legitimate confirmation before giving them a chance to get to the information. From a security point of view, this access control engineering is not any more appropriate when information is outsourced to the cloud have proper certification before letting them access the data. From a security standpoint, this access control architecture is no longer applicable when data is outsourced to the cloud. Since information clients and cloud servers are not in the same put stock in the area, the server may never again be trusted entirely as an omniscient reference screen for characterizing and implementing access control policies and managing user details. In the event of either the server being compromised or a potential insider attack, users' private data might be exposed.

One conceivable way to deal with implement information access without depending on cloud servers could be to scramble information distinctly and reveal the relating decoding keys just to approved clients. This approach usually suffers from severe performance issues, however, and does not scale well mainly when a possibly vast number of on-request clients want fine-grained information to get to control. Researchers have been chipping away at how to understand a fine-grained get to control outline that ultimately uses the cloud's calculation asset abundance. Via this approach, data users would be able to delegate to the cloud most of the cumbersome securely. Employing this approach, information clients would have the capacity to safely delegate to the cloud the more significant part of the awkward client and information administration workloads. For example, dealing with visit client get to user and data management workloads, such as handling frequent user access privilege updates in large dynamic systems while still preserving the underlying data confidentiality against any unauthorized access.

Trustworthy service metering. Computing as an administration has seen remarkable development. The essential inspiration for this move is the guarantee of decreased working and capital costs, and the simplicity of powerfully conveying and scaling new administrations without keeping up a committed computing foundation. With expanded ubiquity and appropriation, be that as it may, new and unexpected difficulties have developed. A typical issue that cloud clients confront today is the failure to comprehend the cost impression of their outsourced calculation.

As stated in Xiong and Liu (2004), because customers have little permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To duplicate or to republish, to present on servers or on redistributing to records, requires earlier particular consent and an expense. No visibility into the infrastructure is allowed, and these charges may have no apparent direct connection to their application tasks. For example, when execution can be elastic in reaction to dynamic workloads, customers may face inexplicable charges in a pay-as-you-go billing model. This problem is further exacerbated in shared or “public” cloud infrastructures. Suppliers lessen capital and administration costs by multiplexing a few clients on a similar foundation utilizing equipment virtualization. While virtualization gives some level of confinement between clients, many mutual assets cannot be flawlessly disconnected. This can bring about unanticipated externalities that may expand an application’s asset impression. For instance, a getting into mischief application may cause reserve misses or system blockage and increment the calculation impression for different applications having the same physical stage.

In the meantime, even though suppliers do create pay with this plan of action, their productivity is not apparent. Many providers are reluctant to release certain revenue and expense estimates. Speculation that arises is that at least in part because providers are still struggling to figure out how to precisely monitor and bill their customers and their resource consumption. Fine-grained monitoring of virtualized computations is a tricky proposition, and it becomes harder when it should likewise be sufficiently substantial to put on a receipt. Subsequently, a few resources that are hard to monitor and attribute to client computations are not accounted. For instance, input and output time and internal network bandwidth are not metered, although these have a non-inconsequential effect on the supplier's working expenses and the execution of other applications. Similarly, sharing effects such as memory pressure due to contention among co-scheduled jobs cause costs that are both difficult to measure and challenging to attribute to their source causally. Therefore, suppliers either erroneously represent these expenses while passing the error on to their clients, or indeed bear the expenses, in this way expanding their particular working costs

As the dominant vacation period of distributed computing winds down, cloud suppliers and clients need to tweak their business techniques to remain practical. Even as early as 2008, 61% of IT executives and CIOs rated the “pay only for what you use” as a significant perceived benefit of the cloud model and more than 80% of respondents rated competitive pricing and offering performance assurances/SLAs as fundamental provider attributes (Sekar & Maniatis, 2011).

Regardless of this early affirmation that asset use and charging are top worries for IT administrators, these have gotten practically no consideration from the business or the research network. Consumers informal discussions with industry personnel and other researchers indicate that mainstream recognition on this subject is frequently very outrageous. One view trusts this is a “non-problem” in that the specific means as of now exist and it is just a short time before advertising powers settle it. The contrary view trusts this is a critical problem, but the systems do not have the technical means to do so. Given that public recognition is energized, but then there is little work in this specific circumstance, our objective in this position paper is to stimulate a dynamic talk in this subject.

Own position for distributed computing administrations to end up useful and practical, a precise system for undeniable asset bookkeeping is required. Such a system will profit both cloud clients and suppliers. It facilitates any worries that clients may have with suppliers’ estimating and execution ensures. It additionally furnishes clients with a reason for precisely looking at changed cloud suppliers. In the meantime, having such a structure empowers suppliers to steadfastly catch their working costs by charging for asset consumptions that they do not right now represent and keeping clients from endeavoring to amusement their charging methods. Moreover, facilitating the estimating execution concerns will empower more cloud appropriation and enhance productivity by expanding the general use of a supplier’s foundation (Sekar & Maniatis, 2011)

The objective is to give accurate resource accounting to cloud conditions, where the calculation is being rented from others. To formally help characterize the issue, a conceptual working model is demonstrated as follows. There are three logical participants: the customer C, the provider P, and the verifier V. At a high-level, C asks P to run the computation task T. Subsequently, P gives the C a consumption report R describing what resources it thinks C consumed. For example, a provider may report a time series of consumption vectors, whose elements correspond to CPU usage, memory bandwidth, memory size, I/O bandwidth, network bandwidth, and energy, aggregated over pre-determined time quanta, for the duration of the task. C takes this report R together with the task T and additional data (the roles of which will become clearer later in this section) to the verifier V and checks if R is a valid resource report for the T.

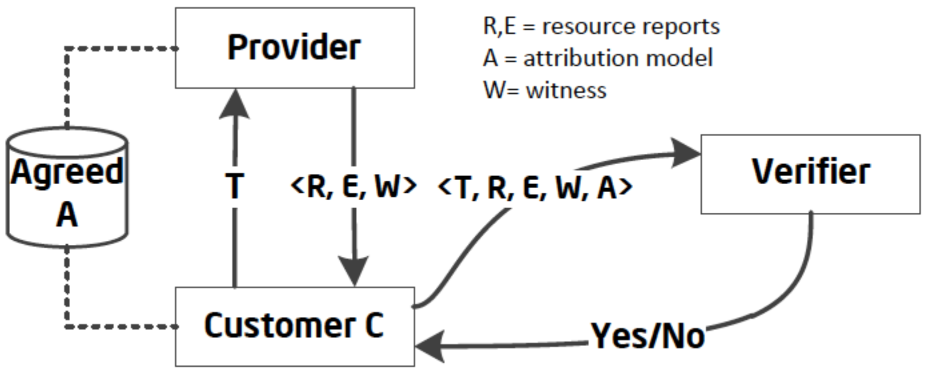


Figure 3. Conceptual Architecture (Sekar & Maniatis, 2011, p. 2)

Granularity. This determines the level of detail at which resources are tracked. On the coarse end, tracking might be done in units of core-hours for the duration of a task; on the subtle end, tracking might be done in units of cycles used per second-long period. In a sense, granularity defines the number and units of the components of a consumption vector. In the model, granularity as a definition or schema of what consumption reports look like is captured:

over what quantum of time resource vectors are provided to the customer, and what elements each resource vector contains. For example, EC2 charges for CPU instance hours, storage size and requests, Internet bandwidth usage, and specialized services such as load balancing. Similarly, Google's AppEngine bills users for CPU Time, in/out network bandwidth, storage, and email requests (Sekar & Maniatis, 2011).

Attribution. The attribution model determines how resource consumption is attributed to the owner of a task. This model represents the charging policy of a provider and covers issues such as whether a task owner should be charged for resources used for task migration, how task owners and the service provider bear the cost of tasks thrashing. To this end, optionally extend the consumption report R with a separate report of indirect, external consumption E, similar to R, but attributed to task T due to external factors such as contention. Together, R and E make up what the provider is going to invoice customer C for task T. In the model, the attribution model A as a function that computes the values of R and E gave a computation T under given conditions is represented. Intuitively, the attribution model can be regarded as a "golden simulator" of the leased architecture (including its management software). This allows a customer to simulate ahead of time how a computation will consume given some inputs and environment conditions (including assumptions about other cotenant customers' workloads) (Sekar & Maniatis, 2011).

Verifiability. Verifiability aims to give the customer assurances about two questions (a) did I consume what I was charged, and (b) should I have consumed what I was charged? The first question concerns itself with the veracity of the consumption vector. A provider should not be able to charge a customer with cycles it did not expand on her behalf. The second question concerns itself with the efficiency of the provider's infrastructure, concerning scheduling and provisioning. If the provider conservatively—or erroneously—used 1 GByte of main memory for a task that only required 1 MByte, it should arguably not be able to pass on the cost of its inefficiency to its customer. To aid in the verification process, the provider may optionally give C a witness W. Leave the specification of the W open depending on the type and level of verifiability desired by C. For example; the witness may merely be hardware attestations to guarantee the integrity of the reports. As discussed earlier, the customer has access to a logical verifier, an oracle that returns Yes/No answers given a consumption report R and E. The witness W, the task T, and the attribution model A. The simplest verifier might be one that uses W as a sanity check to identify gross tampering with R and E. A more involved verifier may use A together with the W to emulate the task T and compute local versions of R0 and E0. Then it can check if these emulated values are close to the actual R and E received. One concern is that having to run a verifier additionally reduces the appeal of outsourcing to the cloud. Note that the verifier need not be run by the customer (i.e., it could be a third-party software service) and that its computer cost will be significantly smaller than the original T (Sekar & Maniatis, 2011).

"Did I"–Verifiability

In this section, the first question focused will be: Did I consume what I was charged? Addressing this question requires infrastructure support to ensure that (a) The provider does not create spurious charges of cycles that were never consumed by any principal (i.e., some conservation of work), and (b) The provider correctly assigns the consumption of a resource to the principal responsible for using that resource. To address these sub-challenges, started with a possible clean-slate solution and then proceed to discuss how efficiently can it be approximated with existing technologies.

For the resources to obtain very fine-grained resource footprints of a broad spectrum, the reporting mechanism should ideally be implemented as a trusted hardware layer. It is because of two reasons. First, the hardware is in the best position to observe the physical resource usage of some resources (e.g., cache misses, I/O requests) that may be abstracted away from the software stack. Second, a hardware root-of-trust is more reliable than OS- or VMMlevel trust in a virtualized environment (Sekar & Maniatis, 2011).

Suppose a trusted hardware layer that provides the following primitive is present. For each time epoch, the hardware generates an attested report specifying the active atomic principals and the amount of each resource consumed by each such principal during this epoch. Specifically, a trusted monitoring component on the hardware generates for each timestep t an attested log entry of the form fU; hS1;:::; SNig, where U refers to an atomic principal associated with the hardware context and Sis are the various resources that need to be accounted. To provide the hardware layer with visibility into application-layer principals, extended ideas from prior work on resource containers on practically exposing higher-layer principals to lower-layers (Banga, Druschel, & Mogul, 1999; Sekar & Maniatis, 2011).

Having a trusted consumption monitor satisfies two essential requirements. First, because the reports are generated using in-hardware monitoring that is trusted to report actual consumption, a provider cannot convincingly charge customers for resources that it never consumed. Second, a provider cannot double-charge the same resources to multiple customers, since only a single customer is associated with every reported consumed resource, and the monitor reports only what was consumed (Sekar & Maniatis, 2011).

Practical approximations. While the above fresh start arrangement is adroitly total, there are three practical challenges: revealing bandwidth capacity for transmitting fine-grained per-age estimations, execution overhead to get such bore witness to estimations, and reliance on a trusted equipment crude that does not exist today. Next, potential answers to address each test is talked about (Sekar & Maniatis, 2011).

Revealing data transmission: Reporting the whole trusted asset utilization log depicted above might be restrictive. For instance, expecting reports of 10 32-bit asset highlights every second, a substantial supplier like Amazon (say 100K cases) would have no less than an overhead 10 32 100K 32 Mbps of outbound activity to send these reports (disregarding metadata and cryptography). To lessen this data transfer capacity cost, a gathered step that procedures the log sections before creating R are imagined. That is, given the authenticated reports over various time ages, a trusted aggregator produces a factual synopsis of the asset utilization vector crosswise over time (Sekar & Maniatis, 2011).

Different classes of resources may define domain-specific aggregation functions. For instance, we may need the entirety as an aggregator for CPU cycles, for memory we might need to find the maximum and the whole of the utilization, and for I/O resources we might need to count the total bandwidth-time footprint. Note that the aggregation can even occur inline, meaning that individual log entries require not be physically created or stored anywhere. Estimation overhead: Running fine-grained estimations on a for every guideline or per-CPU cycle granularity can present a non-unimportant execution penalty for the application forms being checked.

The following discusses three potential opportunities to reduce this performance impact. The first is offloading monitoring to a dedicated co-resident processor on a multi-core platform, similar to previous work on Log-Based Architectures. The main idea is to generate events (e.g., specific memory operations) that are efficiently routed to a secondary core, which then performs on-line analysis or packaging for off-line perusal, leaving the central core to continue performing its primary functionality (Chen et al., 2006).

The second is sampling. Instead of maintaining very fine-grained per-cycle or per-second counters, a subset or sample of the resource utilization can be used. Inspecting networks well with the possibility of aggregation to lessen announcing transmission capacity. If only interested in a coarse-grained or aggregate statistic, a suitable sampling strategy that can give tight bounds on the bias and variance of the estimate can be chosen. However, a sampling approach must be constrained to forestall adversarial activity such as cycle stealing where malicious VMs opportunistically swap themselves out before a sample is taken, to charge their activities to another VM's customer (Zhou, Goel, Desnoyers, & Sundaram, 2011).

Randomization is by all accounts a shoddy however viable answer for counter such malignancy. Advance along the range, previews of the asset utilization vector each time another main (e.g., a procedure) obtains or discharges an asset can be taken; Figure 4 shows this for the CPU, where process setting switches are the obtaining/discharge limits. For whatever length of time that the confided in the screen can ensure that no foremost other than the assigned one can utilize the asset between successive procure/discharge occasions, at that point, such estimations can be utilized to produce high-exactness reports with negligible overhead. Unwinding equipment reliance: Realizing such confided in equipment capacities includes typically long advancement cycles on the request of several months. A specific inquiry, at that point, the confided in revealing and total natives utilizing existing equipment and programming capacities can be approximated. The thoughts behind Log-based Architectures can help here.

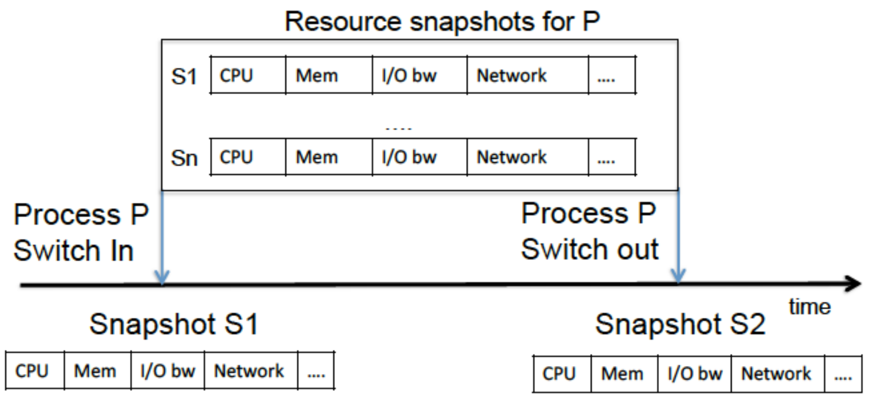


Figure 4. Taking Snapshots of the Resource Consumption Vector Before/After a Context Switch (Sekar & Maniatis, 2011).

Rather than performing a full per-principal attested measurement in an online fashion, a more straightforward, but dedicated hardware that records the instruction stream. Then, a post-processing step that reconstructs the sequence of actions and associates the resource consumptions to the active principals in each epoch can be included. The challenge here is to provide this post-mortem processing with sufficient context to do the attribution accurately. The capabilities, strictly speaking, can be implemented at the VMM layer. Unfortunately, modern VMMs are sufficiently complex and involve several millions of lines of code. Thus, adding the VMM to our trusted computing base may not be a feasible alternative. Fortunately, a minimal subset of the functionality that VMMs provide and this can be implemented as a small statically verifiable module is needed. Here the system can build on the promise of several recent efforts for building a small, but trusted, shim layer that performs a more restricted set of monitoring tasks. For example, recent work has shown that it is possible to implement a shim layer to intercept network packets and to isolate trusted functionality from a legacy OS. The challenge here is to evaluate whether the software-based solutions have enough visibility into monitoring low-level effects (e.g., cache stress) if they need to be reported as well (Mccune, Parno, Perrig, Reiter, & Isozaki, 2008).

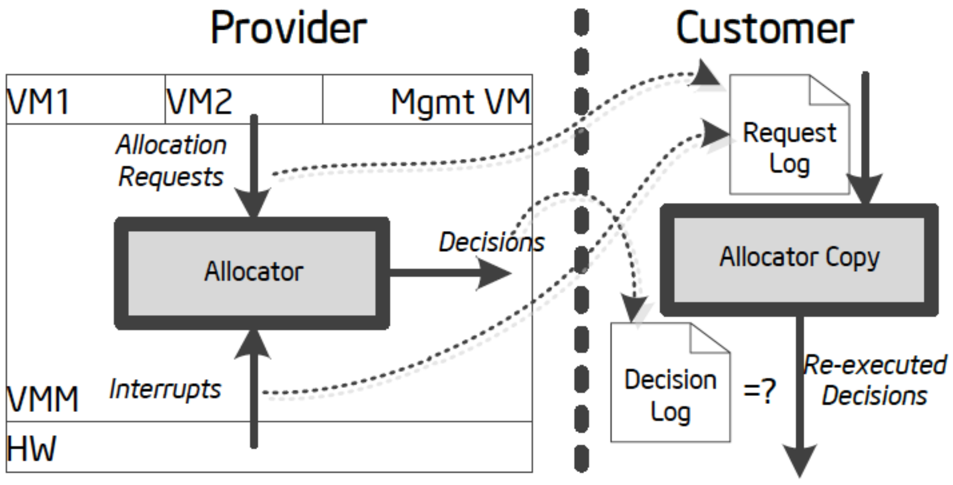


Figure 5. Prescriptive "Should I" Verifiability (Seka & Maniatis, 2011).

"Should I"—Verifiability

"Should I"-verifiability concerns itself not with whether specific usage attributable to a customer took place—the subject of the previous section–but with the justification for that usage. Given the workload at the infrastructure provider and the resource allocation policy agreed upon by the customer, should the customer's job have consumed the charged resources? Assume here that the customer and the provider have an agreed upon the specification of the "right" resource-allocation policy. The following two types of specifications are considered:

1. A prescriptive specification that describes in some executable pseudocode the resource allocation decisions that should be made, given current workloads; or
2. A quantitative specification that describes an upper bound for the resources the allocator should devote, as a function of the given task and other workloads.

Two kinds of studies “Should I”-verifiability below, first by considering ideal solutions and then relaxing those solutions to make them practical. In theory, a prescriptive specification can be verified via re-execution. Assuming that a self-contained component makes all resource allocation decisions—the allocator handling memory, CPU, I/O, and other resource allocation decisions—a verifier with a trustworthy log of all inputs and outputs to the allocator can re-execute it and compare allocation decisions to those logged, to ensure the remote allocator was run correctly. This is similar to the notion of remote verification of distributed computations, as in Peer Review (Haeberlen, Kouznetsov, & Druschel, 2007).

In Peer Review, a computation (the allocator in our case) is checked by a remote verifier by being re-executed according to a tamper-evident record of its inputs, comparing the outputs to a tamper-evident record of the outputs at the provider, and looking for deviations between the two streams of outputs. In practice, this idealized solution to prescriptive “should I”- verifiability is hampered by several challenges. To begin with, the raw log of contributions to the allocator is voluminous (perhaps different sections every second from each machine in the framework); gathering and imparting that log to the customer could require significant bandwidth and storage. Second, making that log tamper-evident requires computational resources for cryptographic or other “conditioning” of the log, possibly involving a TPM in the process. Third, the logic of a resource allocator (its code) or the policy may be proprietary to the provider, so sharing its specifics to enable re-execution may be unacceptable for privacy reasons. Quantitative “should I”- unquestionable status, in a perfect world, requires no data about the logic of an allocator, yet more data (than the prescriptive approach) about the condition of the entire framework. Given the (tamper-evident) time series of resource utilization and their allocations to different customer principals, the quantitative approach applies mathematical functions to the inputs to estimate an upper bound on utilization for the customer's task. If that upper bound is significantly lower than the charged resources, the customer suspects the provider for liberal scheduling. This concept appears related to SLA verification, with a twist. While SLA verification establishes that a service provided a minimum guaranteed level of resources to a customer, quantitative verifiability establishes that the service allocated to a customer (and subsequently charged for) no more resources than required by the submitted task. In practice, the challenges with quantitative verifiability are somewhat steeper regarding the volume and integrity-protection of raw data, since now the verifier requires not only decisions made by the allocator but also instantaneous measurements of state properties, such as utilization. Then again, undeniable quantitative nature does not require data about the allocator's code or arrangement, since the verifier is autonomous of the logic of the allocator and requires no re-execution or emulation. In what follows, approximations of these ideal verifiability approaches to address the similar challenges are studied (Haeberlen, Kouznetsov, & Druschel, 2007).

Practical approximations. First, consider prescriptive verifiability. The log-volume challenge is one of location. One way to address it is by moving the verifier (or the verifier’s trusted agent) closer to the source of the log, on the provider’s platform. For instance, if the provider’s platform includes an execution environment trusted by the verifier (e.g., a verifier-trusted minimal OS and application, booted on the provider’s platform using a hardware root of trust), then the verification could run there, obviating the need for transporting or storing large logs. This is complementary to doing local aggregation for “Did I”-verifiability. The second challenge that of the tamper-evidence of logs, remains difficult regardless of where the verifier operates. On the one hand, collecting the logs must be trustworthy and, on the other hand, the logs must maintain their integrity before analysis by the verifier. Similar to “Did I”-verifiability, an approach to mitigate this challenge might be to use a trusted hypervisor to collect and authenticate the log (e.g., by trapping on logged events so that the VM cannot evade monitoring). Before forwarding it to the verifier; e.g., via techniques like Secure In-VM Monitoring, combining this trusted log collector with a trusted execution environment for the verifier’s code would compound the benefits of addressing both log-related challenges. The final question is what the verifier code should be, whether it runs on the provider’s platform or at the customer (Sharif, Lee, Cui, & Lanzi, 2009).

Multitenancy security and privacy multitenancy. Multitenancy Security and Privacy Multitenancy is a fundamental trait of distributed computing. To improve asset use, CSPs frequently utilize equipment virtualization to conceal a figuring stage's physical qualities. This gives different clients a chance to run their particular application occurrences all the while on the same physical foundation without seeing each other’s information. Multitenancy expands utilization of the essential hardware resources and, with virtualization, facilitates the administration trouble for CSPs, taking into account proficient, and successful asset provisioning and re-designation without the requirement for any forthright equipment buy or setup.

Regardless of its advantages, this multi-occupant cloud condition likewise displays extreme security dangers and protection vulnerabilities to both the cloud foundation and cloud clients. Virtualized situations share comparable functionalities with existing working frameworks and applications in the physical condition, so programming bugs and recently recognized security vulnerabilities in these frameworks remain the essential threat to any virtualized multitenant environment. Thinking about the size of cloud frameworks, the potential danger from these security dangers can be considerably greater contrasted with that for a non-virtualized computing condition Besides, for resource management in the cloud; various virtualized application occasions must always be provisioned, designated, or even moved between numerous physical machines. Therefore, such powerful highlights in the multitenant condition additionally compound the issue’s many-sided quality and make accomplishing and keeping up steady security troublesome.

Multitenancy additionally opens entryways for potential protection spills. As specified beforehand, side-channel attacks display new dangers to cloud clients' data in the multitenant condition. In a current report, scientists utilized designing strategies to gather the virtualized asset distribution system from CSPs and effectively set their virtualized application example on the same physical machine as the target victim. They were then ready to separate the victim's private data through activity designs and opposite side-channel information. These outcomes demonstrate that even in an unequivocally segregated multitenant condition, this.

Multitenancy security and protection is one of the fundamental difficulties for people in general cloud, and discovering arrangements is significant if the cloud is to be broadly received.

In any case, little work exists today that tends to these issues as well as reliably and keeps up this dynamic registering condition's adaptability.

Security overhead and more. Although planning security into the cloud benefits clients and CSPs, it expands overhead for both. For clients specifically, such overheads could balance the cloud's monetarily engaging advantages and may strife with their explanations behind utilizing the cloud in any case. How to quantitatively explore the trade-offs between security overhead and cloud benefits is another exciting but essential problem. Any answer for this inquiry will enable clients to settle on better-educated choices previously moving to the cloud.

Since the beginning of cloud services, everybody concentrated on exploring the technology and creating new services and no serious concerns were given to security as such. It all started when there were a bunch of services on the cloud already, and security issues started creeping in. Since the system are already built, and security was never included in the requirements/design of the system, it generated much regression and fixing systems with security patches.

After many such failures, security engineers realized the importance of cloud security and started work on introducing multiple theories that would solve these problems. Moreover, since Consumer's privacy and Cloud service protection are one of the main concerns in cloud services, building a Trust management service appeared to be one of the core solutions concerning cloud security.

Literature Related to the Problem

Sybil attack. In this attack, malicious users exploit multiple identities (Ba & Pavlou, 2002; Douceur, 2002) to give numerous misleading feedbacks for a self-promoting or slandering attack. It is interesting to note that attackers can also use multiple identities to disguise their negative historical trust records (i.e., whitewashing attacks (Mogul, 2005).

Named after the case study of a woman with multiple personality disorder, a Sybil attack is a type of security threat when a node in a network claims multiple identities. Most systems, similar to a distributed system, depend on the suppositions of identity, where every PC speaks to one character. A Sybil attack happens when an insecure computer is hijacked to claim multiple identities. Problems arise when a reputation system (such as a file-sharing reputation on a torrent network) is tricked into thinking that an attacking computer has a disproportionally large influence. Similarly, an attacker with many identities can use them to act maliciously, by either stealing information or disrupting communication. It is essential to perceive a Sybil attack and note its threat keeping in mind the end goal to shield oneself from being an objective.

First described by Microsoft researcher John Douceur (2002), a Sybil attack relies on the fact that a network of computers cannot ensure that each unknown computing element is a distinct, physical computer. Some authorities have attempted to establish the identity of computers on a network (or nodes) by using certification software such as VeriSign, employing IP addresses to identify nodes, requiring passwords and usernames, and so forth. However, impersonation, both in the real and digital worlds, is commonplace. Friends may share passwords, communities may share website registrations, and some services provide a single IP address that is shared among users.

Sybil attacks have shown up in numerous situations, with broad ramifications for security, wellbeing, and trust. For example, an internet poll can be rigged using multiple IP addresses to submit a large number of votes. A few organizations have additionally utilized Sybil attacks to increase better evaluations on Google Page Rank.

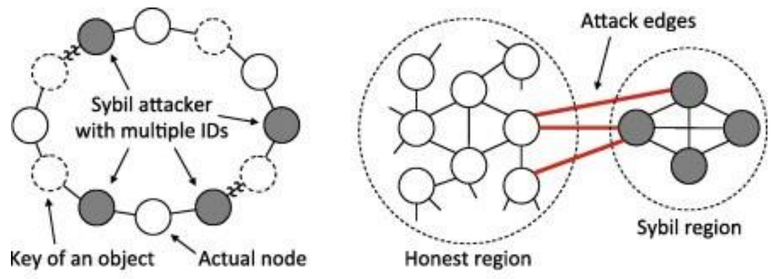


Figure 6. Demonstration of Sybil Attack (NS-2 Simulator, n.d.)

There are a couple of beyond any doubt fire approaches to shield a system from a Sybil attack, yet there is an extensive variety of writing committed to talking about choices for assurance and verification of competing identities.

Karloff and Wagner (2003) noted that the Sybil attack also poses a threat to routing mechanisms in sensor networks. To protect against the Sybil attack, An approval that every node is the main character displayed by the comparing physical node. There are two types of ways to validate identity.

- Direct Validation: The first type is direct validation, in which a node directly tests whether another node identity is valid.
- Indirect Validation: The second type is indirect validation, in which nodes that have already been verified are allowed to vouch for or refute other nodes.

Since the first analysis of the Sybil attack, several different approaches have been proposed to prevent or mitigate the attack: trusted certification, resource testing, radio resource testing, RSSI-based scheme, and crucial random pre-distribution.

Trusted certification. Trusted certification is by far the most frequently cited solution to defeating Sybil attacks (Rowaihy, Enck, McDaniel, & La Porta, 2007). It involves the presence of a trusted certifying authority (CA) that validates the one is to one correspondence between an entity on the network and its associated identity. This centralized CA thus eliminates the problem of establishing a trust relationship between two communicating nodes. Douceur (2002) has proven that trusted certification is the only approach that has the potential to eliminate Sybil attacks. However, trusted certification relies on a centralized authority that must ensure each entity is assigned precisely one identity, as indicated by possession of a certificate. In fact, Douceur offered no method of ensuring such uniqueness, and in practice, it must be performed by a manual or in-person process. It may be costly, or a create a performance bottleneck in large-scale systems. Moreover, to be effective, the certifying authority must ensure that lost or stolen identities are discovered and revoked. If the performance and security implications can be solved, then this approach can eliminate the Sybil attack (Kuma, 2016).

Resource testing. Resource testing is the most regularly executed answer for turning away Sybil attacks. The fundamental principle is that the quantum of computing resources of each entity on the network is limited. A verifier at that point checks whether every character has the same number of assets as the single physical gadget it is related with. Any inconsistency demonstrates the likelihood of a compromised node. Storage, computation, and communication were initially proposed as resources. In any case, for a framework, for example, a remote sensor network, an attacker may have capacity and calculation assets in substantial limits contrasted with resource starved sensor nodes. Alternatively, verification messages for verifying communication resources might flood the entire system itself. Hence, all three are inadequate choices for sensor network

Radio resource testing. Radio resource testing, proposed by Newsome et al. (cited in Sekar & Maniatis, 2011) is an expansion of the asset testing check strategy for wireless sensor networks. The fundamental assumptions of this approach are that any physical device has only one radio and that this radio is incapable of transmitting and receiving messages on more than one channel at any given time. As a concrete example, consider that a node wants to verify that none of its neighbors are Sybil identities. It can assign each of its n neighbors a different channel to broadcast some message. It can then choose a channel randomly on which to listen. If the neighbor that was assigned that channel is legitimate, it should hear the message. Resource tests have been suggested by many as a minimal defense against Sybil attacks where the goal is to reduce their risk substantially rather than to eliminate it (Sekar & Maniatis, 2011).

RSSI-based scheme. Demirbas and Song (cited in Sekar & Maniatis, 2011; Conner, Iyengar, Mikalsen, Rouvellou, & Nahrstedt, 2009) introduced a method for Sybil detection based on the Received Signal Strength Indicator (RSSI) of messages. The cooperation of one additional node (and hence one message communication) is required for the proper functioning of this protocol. Upon receiving a message, the receiver will associate the RSSI of the message with the sender ID included, and later when another message with the same RSSI but with different sender-id is received, the receiver would detect Sybil attack. A localization algorithm is used in this scheme. Sybil attacks can be detected with the completeness of 100% with few false positive alerts. Despite the way that RSSI is untrustworthy and that transmissions employing radio are non-isotropic, the utilization of proportions of RSSIs from numerous collectors tackles this issue.

Random key pre-distribution. This technique enables nodes to establish secure links to other nodes in wireless sensor networks. In crucial random pre-distribution, a set of keys are assigned at random to a node enabling it to discover or compute the standard keys that it shares with its neighboring nodes. The key ideas are the association of the identity with the key assigned to a node and the validation of the key. Validation involves ensuring that the network can validate the keys that identity might have. Thus, given a constrained arrangement of caught keys, there is little likelihood that a discretionarily created personality will work, for the keys related with an arbitrary identity are not likely to have a significant intersection with the compromised key set, making it hard for the fabricated identity to pass the critical validation.

Utilizing trusted gadgets resembles using trusted affirmation to shield against a Sybil attack. For this circumstance, identities are identified with specific hardware devices. Similar to a central authority creating certificates, there are a few ways to prevent an attacker from attaining multiple devices.

Collusion attack. In this attack, several vicious users collaborate to give numerous misleading feedbacks to increase the trust result of cloud services (i.e., a self-promoting attack (Douceur, 2002) or to decrease the trust result of cloud services (i.e., a slandering attack) (Friedman et al., 2007). This sort of malicious behavior can happen in a non-tricky manner where a specific malicious client gives various deceiving feedbacks to lead a self-promoting attack or a slandering attack.

Chapter III: Methodology

Introduction

As a part of proposing and evaluating a Trust Management for cloud services, a quantitative approach is used to showcase the results.

Methodologies that included feedback loop-based algorithms that will evaluate different trust-based models created within the network will be proposed. The simulation results will showcase the resilience of such a trust-based network against different types of attacks and compares itself with the existing models to show why it is a better approach for solving trust management for cloud services.

Design of the Study

In this paper, a design of a CloudArmor (CLOud consUmers creDibility Assessment and tRust manageMent of clOud seRvices) is introduced. It is a framework for reputation-based trust management in cloud environments.

In CloudArmor, trust is delivered as a service (TaaS) where trust is managed in a widely distributed network and includes feedback mechanism so that the trust factor can be re-evaluated from time to time. CloudArmor exploits techniques to identify credible feedbacks from wicked ones.

Some of the features that are a part CloudArmor are:

1. Zero-Knowledge Credibility Proof Protocol (ZKC2P). A ZKC2P that jam the customers' security, as well as empowers the TMS to demonstrate the believability of a specific purchaser's input is presented. An Identity Management Service (IdM) can help TMS in measuring the credibility of trust feedbacks without breaching

consumers' privacy is proposed. Anonymization techniques are exploited to protect users from privacy breaches in users' identity or interactions.

2. Credibility Model. The credibility of feedbacks plays a vital role in the trust management service's performance. Therefore, several metrics for the feedback collusion detection including the Feedback Density and Occasional Feedback Collusion. These metrics distinguish misleading feedbacks from malicious users. It additionally can distinguish essential and intermittent practices of collusion attacks (i.e., assailants who expect to control the trust comes about by giving numerous trust feedbacks to a particular cloud service in a long or short period). Besides, several metrics for the Sybil attacks detection including the Multi-Identity Recognition and Occasional Sybil Attacks are proposed. These measurements enable TMS to recognize deluding criticisms from Sybil attacks.
3. An Availability Model. High accessibility is a critical prerequisite to the trust management service. Thus, to spread several distributed nodes to manage feedbacks given by users in a decentralized way is proposed. Load adjusting techniques are misused to share the workload, accordingly continually keeping up the coveted accessibility level. The quantity of TMS nodes is resolved through an operational power metric. Replication methods are misused to limit the effect of inoperable TMS examples. The quantity of copies for every hub is resolved through a replication assurance metric that will be presented. This metric exploit particle shifting methods to absolutely foresee the accessibility of every hub. Each of these methodologies is aimed at solving each of the sub-problems mentioned in the problem description

section. Each one of these methodologies has their unique solutions, algorithms and explains how they solve the underlying problems in detail.

Data Collection

For experimentation, data will be reused from another experimental study and use it for our calculations. There is already an existing component for collecting cloud services and trust information. It is a cloud service crawler module that is based on an open source web crawler which has been extended to discover multiple services in the internet cloud.

Web crawler. Web crawlers are PC programs that breadth the web, 'perusing' all that they find. Web crawlers are generally called spiders, bots, and programmed indexers. These crawlers examine web pages to perceive what words they contain, and where those words are utilized. The crawler transforms its discoveries into a full record. The record is generally a noteworthy summary of words and the web pages that component them. Along these lines, when one approaches a web search tool for pages about hippos, the web index checks its file and gives you a rundown of pages that say hippos. Web crawlers look at the web routinely, so they, for the most part, have a leap forward record of the web.

To understand how this crawler functionality has been implemented, it is better to understand, functionality on how the crawlers usually work. The accompanying area will give a short outline of the same. Some of the crawler components include:

1. Seed List: Before a crawler sets out on its trip of crossing a list of locales, it is fundamental to build up an essential seed list of URLs that would, like this, comprise of different URLs on their goal pages, from where the web crawlers onto their coveted way can sling. It deals with what URLs and pages a crawler would wind up

extricating since it would experience every one of the pages that are associated with the seed URLs what's following, unique URLs that would all be transitively related and interlinked with these structural arrangements of URLs. The crawler would then take after its implicit crossing calculation through which it might creep the associated sets of pages and their hubs on a flat level or vertically. These days, the vast majority of the sites have a sitemap.xml document with a list of all URLs exhibit on the site, to help the web index bots find all pages on a visit.

2. Fetching the basic substance: Before beginning on its voyage, each web crawler has a type of DB to check against, which keeps up a list of all seed URLs. At that point, it needs to check if different URLs from a page should be crept, usually in light of the refresh recurrence of the site. Once the URLs to have crawled have been shortlisted, every one of the URLs that ought to be sought is pushed into the line that takes after a LIFO/FIFO configuration relying upon the crawler's calculation, and the URLs are evacuated as and when they get crept. In most such setups from specific points of view, the line comes as an advantageous mechanical assembly to improve the building of the whole system after that crawler goes and gets the page and spares it on a neighborhood machine. Honest to goodness bringing of the pages in layman's terms resembles going on a page and after that finishing a 'Right snap spare.' Bots achieve the similar functionalities in various ways. On the off chance that the site is more intuitive and have a great deal of AJAX cooperations, at that point, bots must be further developed or custom to get the data. In the wake of getting the data, it is then secured autonomously for extraction and arranging.

3. Discovery of new URLs and destinations: There are more than one trillion webpage pages shown on the web, with more site pages coming up each day. Hence, it is not achievable to store each one of these URLs in the line physically or even mechanically so far as that is concerned. Accordingly, each time a crawler is created, it is key to incorporate the Discover work in the web crawler code. Like this, one can set up a crawler to discover URLs without any other individual's information that ought to be crawled by bobbing from one seed URL to the partner URL. By and by, the way site pages are associated and interlinked; bots travel between various pages. Be that as it may, if there are free pages, which neither interface with various pages nor is associated from various pages, they are elusive. Along these lines, site administrators assume extra care to position them in sitemap.xml or incorporate the association for them somewhere on the site page with the objective that web look apparatuses can reach them successfully.
4. Intelligent slithering. Another important assignment to perform amid the Discovery of URLs is that to check regardless of whether these URLs have been crept beforehand in a similar slither. While discovering more URLs from each page that is crawled, it is outstandingly possible that the crawler may encounter URLs of site pages it has recently crawled previously, since the web is only an arrangement of interlinked website pages. Keeping in mind that the ultimate objective to avoid re-creeping these pages and shield the crawler from going into a circle, it is crucial to play out a deduplication check before crawling a page. If that page has just been slithered, one can push that page to the seed list seed once-over to empower disclosure of pages less

requesting. If not, well by then straightforwardly ahead and crawl the page joyfully to make an advance revelation of pages more available. If not, well at that point go ahead and slither the page joyfully. Crawlers today have developed a claim to fame set of complex inquiries that can help keep the excess stack on crawlers to crawl the pages various circumstances every day and besides the pages that get crawled various circumstances every day, therefore affecting their load time. Consequently, while creeping a page, another check is raced to see the last time stamp when that page was refreshed, per slither. In case a page is invigorated more regularly, by then it looks good to crawl that page every now and again to recognize and report the movements as need be. On the off chance that a page is invigorated less frequently, by then, it is rude to stack the page's server with reiterated crawl requests, and from this time forward it is only considerate to encourage the weight on the crawlers to creep the page superfluously.

5. Parsing. Once the pages have been gotten, the next errand is to get data from it. Web look apparatuses use changed computations and heuristics to find data from the substance exhibit on the site page. With the goal that when specific terms are looked, web crawlers appear relevant pages in light of the data they separated from these pages. As of late, there is a more prominent focus on SEMANTIC MARKUP and SEMANTIC RESULTS, wherein, web crawlers endeavor to deduce distinctive fields showed on the page. Distinctive markups proposed by schema.org is another first propel which helps webpage proprietors furthermore web crawlers. Regardless, its

focal point is that those little scale names help in parsing and getting data from the content. prompt.

6. Data Storage. When a user is composing a crawler, given the sheer volume, storage of data turns into a sufficiently major issue. Considering the measure of data that is crawled and required each day, standard SQL databases are not set up to manage that sort of volume on a normal start. Besides, this data does not have numerous social characteristics. It is the place for the Hadoop, and other NoSQL databases come into the photo. It can store and inquiry many data rapidly and complete the preparing of the same inside minutes. A few people likewise utilize frameworks, for example, Depth and S3 and other comparable administrations that could be utilized to store and offer data over various stages. Once in a while, people have used level records with sensible accomplishment. The open source rendition of the web crawler utilized for this hunt is one such web crawler that is constructed utilizing the highlights depicted previously. Crawlers will control the data that is being gathered in light of the number of companions, the space in which the crawler is set up and the time inside which the slithering can happen. Likewise, the data for the clients are gathered as verifiable records and reviews and are being put away in a database. This data incorporates the ID data for a client, many character data when he was getting to various administrations in the cloud, utilization designs (which can be replayed), and so on.

Tools and Techniques

- Database: Trust Feedback DB
- Registry: Trust Identity Registry

- Notification Mechanism

Hardware and Software Environment

The following are the Hardware and Software requirements needed for developing and running the prototype:

Hardware Requirements

• System	Pentium IV 2.4 GHz
• Hard Disk	40 GB
• RAM	512 MB

Software Requirements

• Operating system	Windows 10
• Coding Language	JAVA/J2EE
• IDE	NetBeans 7.4
• Database	MYSQL

Chapter IV: Implementation

Introduction

Cloud service consumers’ criticism is a decent source to evaluate the general dependability of cloud services. In this paper, novel techniques that help in detecting reputation-based attacks and allowing users to identify trustworthy cloud services effectively are presented. A credibility model that not just distinguishes deluding trust criticisms from collusion attacks yet, besides, identifies Sybil attacks regardless of these assaults happen in a long or short period (i.e., strategic or occasional attacks respectively) is introduced. An availability model that maintains the trust management service at the desired level is also developed.

System Design

Data Flow Diagram (DFD).

1. The DFD is also called a bubble chart. It is a basic graphical formalism that can be utilized to speak to a framework as far as information to the framework, different preparing did on this information, and this system generates the output data.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system. (Format is off here slide over to the left slightly)
3. DFD shows how the information moves through the system and how a series of transformations modify it. It is a graphical procedure that delineates data streams and the changes that are connected as information moves from input to output.

4. DFD is also known as a bubble chart; a DFD might be utilized to describe a framework at any level of reflection. DFD might be divided into levels that speak to expanding data streams and provide useful detail, see Figure 7 below as an example of a DFD.

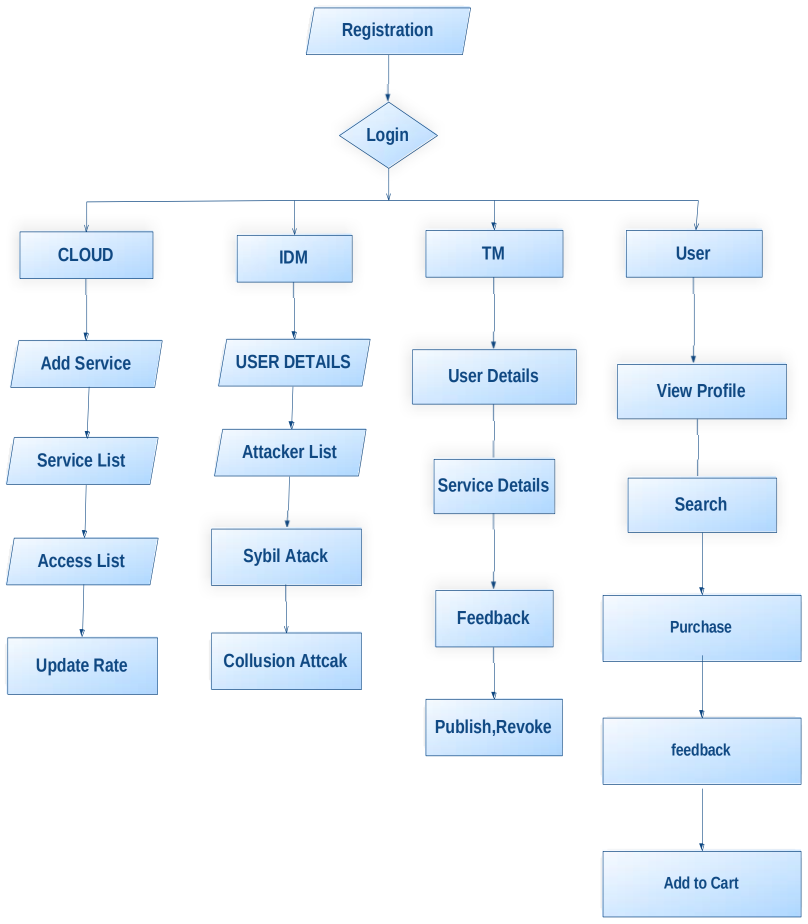


Figure 7. Data Flow Diagram

UML diagrams. UML stands for Unified Modeling Language. UML is an institutionalized broadly useful modeling language in the field of object-oriented programming. The standard is overseen and was made 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

present frame, UML is contained in two noteworthy parts: A Meta-model and documentation. Later on, some technique or process may likewise be added to or connected with, UML.
The Unified Modeling Language is a standard language for demonstrating, visualization, constructing and chronicling the relics of programming systems, and likewise for showing business processes and other non-programming structures. The UML is an essential part of developing object-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

Goals. The Primary objectives in the plan of the UML are as per the following:

1. Give clients a prepared to-utilize, an expressive visual demonstrating language with the goal that they can create and trade essential models. Give extendibility and specialization systems to broaden the first ideas.
2. Be free of specific programming dialects and improvement processes.
3. Give a formal premise to understanding the displaying language
4. Support the development of object-oriented devices showcased.
5. Integrate best practices.

Use case diagram. A utilization case chart in the Unified Modeling Language (UML) is a kind of behavioral outline characterized by and made from a use-case examination. Its motivation is to show a graphical outline of the usefulness gave by a framework concerning actors, their objectives (spoke to as use cases), and any conditions between those use cases. The primary inspiration driving a use case outline is to demonstrate what structure limits are performed for which on-screen character. Roles of the characters in the framework can be portrayed as in Figure 8.

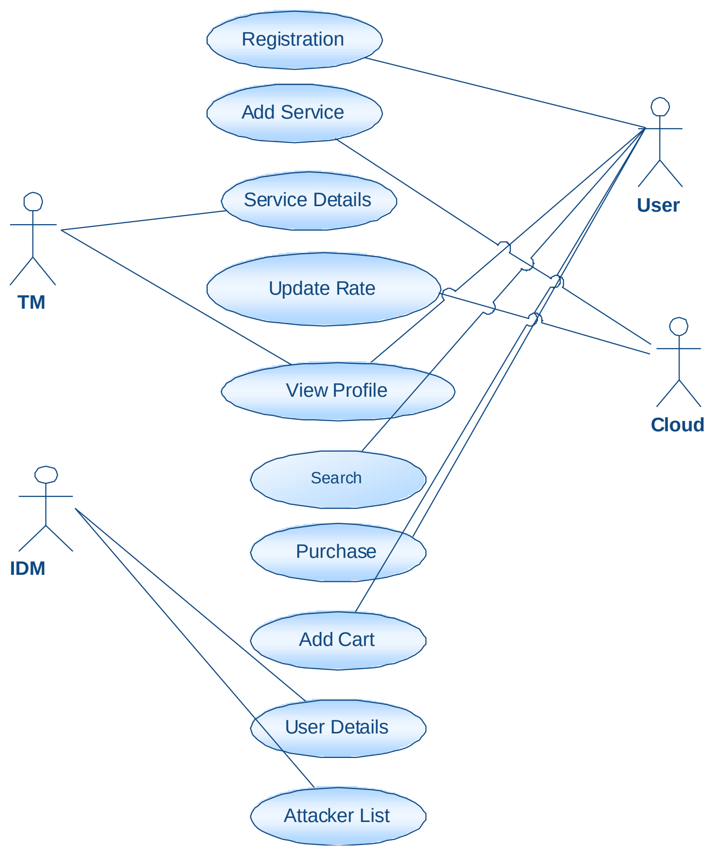


Figure 8. Use Case Diagram

Class diagram. In programming, designing a class diagram in the Unified Modeling Language (UML) is a sort of static structure outline that depicts the structure of a framework by demonstrating the framework’s classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains which information as seen in Figure 9.

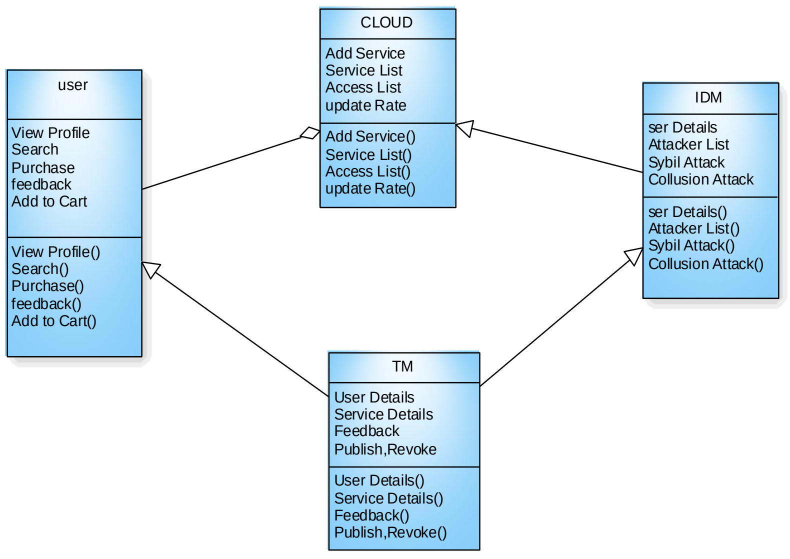


Figure 9. Class Diagram

Sequence diagram. An arrangement outline in Unified Modeling Language (UML) is a sort of connection graph that shows how shapes function with each other and in what game plan. It is a build of a Message Sequence Diagrams. Sequence diagrams are seen here in Figure 10, and they are called occasion charts, occasion situations, and timing outlines.

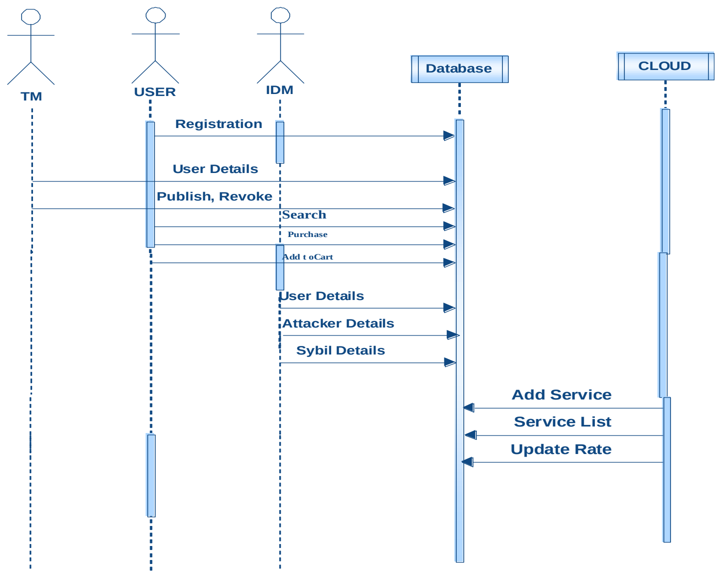


Figure 10. Sequence Diagram

Activity diagram. Activity Diagrams are graphical portrayals of work processes of stepwise exercises and activities with help for a decision, emphasis, and simultaneousness. In the Unified Modeling Language, movement charts can be utilized to portray the business and operational well-ordered work processes of parts in a framework. A movement graph demonstrates the general stream of control, as seen in Figure 11.

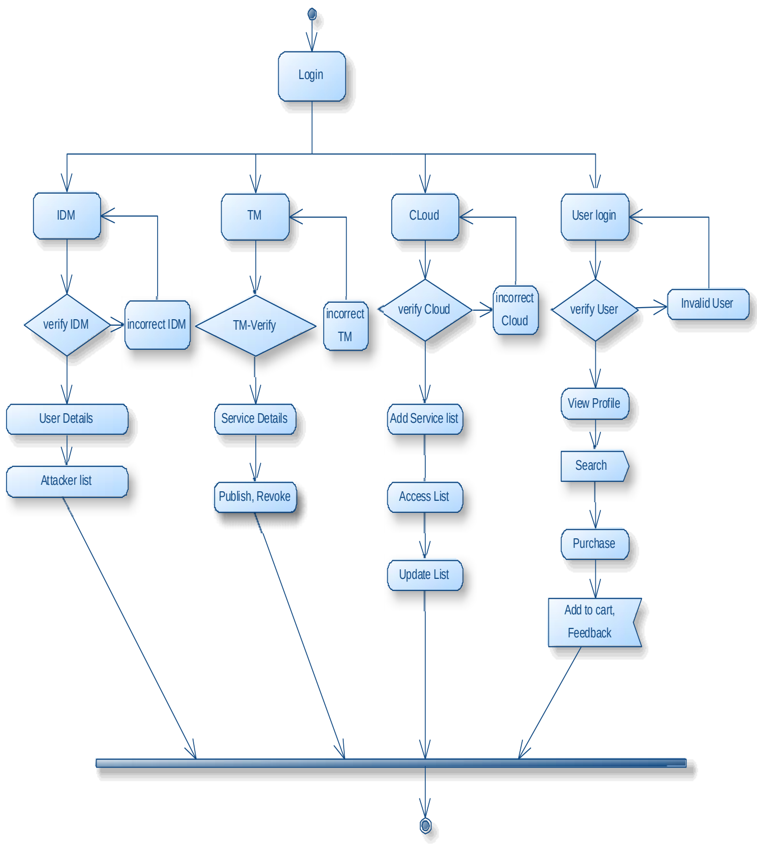


Figure 11. Activity Diagram

System Testing

The purpose of testing is to discover errors. Testing is the way toward endeavoring to find each possible blame or then again deficiency in a work thing. It gives a way to deal with check the convenience of parts, sub-congregations, gatherings and additionally a completed item. It is the way of practicing programming with the goal of guaranteeing that the programming framework lives up to its necessities and client desires and does not change unacceptable. There are different sorts of tests, and each test composed addresses a particular testing necessity.

Types of testing.

Unit testing. Unit testing includes the plan of experiments that approve that the interior program rationale is working legitimately and that program inputs create strong yields. All choice branches and interior code stream ought to be approved by management. It is the trying of individual programming units of the application, and it is done after the finish of an individual unit before combination with other units. It is a basic test that depends on learning of its development and is obtrusive. Unit tests perform first tests at the segment level and test a particular business process, application, and additionally framework arrangements. Unit tests ensure that each exceptional method for a business methodology performs precisely to the recorded particulars and contains unquestionably portrayed data sources and expected results.

Integration testing. Integration tests are intended to test coordinated programming parts to decide whether they keep running as one program. Testing is on occasion driven and is more worried about the necessary result of screens or fields. Incorporation tests exhibit that even though the segments were independently fulfilled, as appeared by effectively unit testing, the blend of segments is remedied and steady. Integration testing is mainly done for uncovering the issues that emerge from the mix of segments.

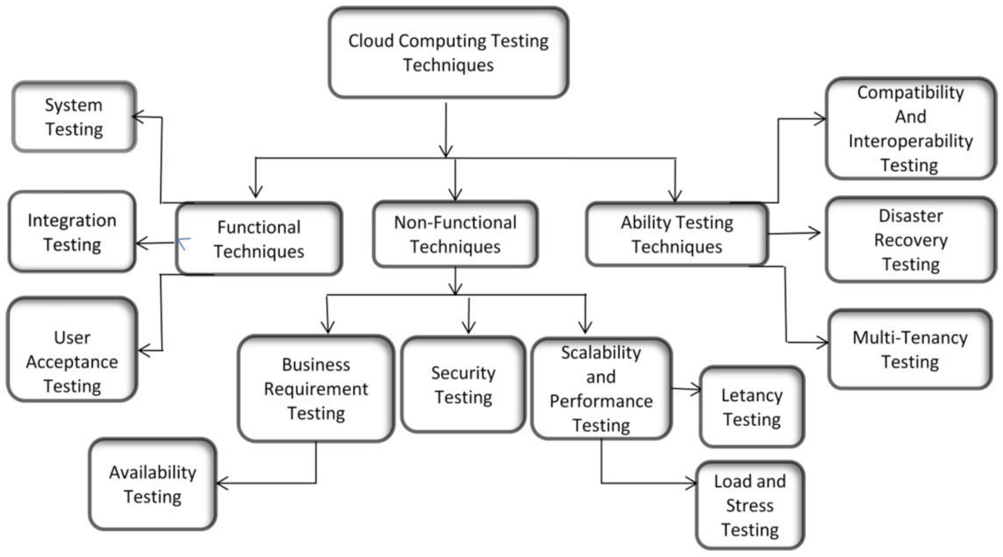

Cloud Computing Testing Techniques
Functional Techniques
Non-Functional Techniques
Ability Testing Techniques
System Testing
Integration Testing
User Acceptance Testing
Business Requirement Testing
Security Testing
Scalability and Performance Testing
Availability Testing
Letancy Testing
Load and Stress Testing
Compatibility And Interoperability Testing
Disaster Recovery Testing
Multi-Tenancy Testing

Figure 12. Cloud Testing Technique Types (Shrivastava, Gupta., & Tiwari, 2014).

Functional testing strategies. Functional testing is a procedure of value affirmation. It is performed for both remote and neighborhood applications. It is utilized to test every one of the highlights and elements of a system which incorporates programming and equipment. Functional testing includes conveying different assignments and contrasting the consequence of same undertakings and the standard yield. Functional Testing can be performed both physically and consequently with a human analyzer or programming program individually. Functional Testing more often than not depicts what the system does.

System testing. System testing will be testing performed on an utterly coordinated system to assess the system's consistency with its predefined requirements. It is performed concerning a Functional Requirement Specification(s) (FRS) or conceivably a System Requirement Specification (SRS) all in all structure. System testing tests the conduct, plan and the desires of the client.

Integration testing. Integration testing is the strategy in which every product module is tried as a gathering. It fits cloud registering systems with regards to a general business technique. Integration of cloud administrations ends up a major for business and wonder that grasps a cloud-based game plan that requires integration of data and interfaces in the cloud with the on-ask for the application. It is for the most part in charge of interfacing source and target systems, extricating information from source systems, intervening semantics and sentence structure of information and distributing the information to target systems.

User acceptance testing. This testing is done to check the presently gave cloud arrangement from the merchant. In this testing, business requirements are utilized to demonstrate that the Cloud arrangement that is conveyed addresses particular issues. This testing is done on both off-commence and at the start. Quick control and observing of test advance are permitted by on location testing.

Non-functional testing methods. This testing is improved the situation guaranteeing that a web application meets the predefined performance requirements. It is otherwise called a performance testing procedure. It is done against the non-functional requirements which mirror the nature of the item. It affects clients.

Business requirement testing. Associations ought to painstakingly, correctly look at their business requirements before moving their business to a cloud processing arrangement; this is on account of business requirements are the building obstructs for cloud registering arrangements. The business requirements can be accomplished by the surveys, workshops, and gatherings.

Cloud availability testing. These guarantees cloud administrations must be accessible consistently. There ought to be no downtime which could unfavorably influence the business of the customer.

Cloud security testing. It has turned out to be one of the essential parts of testing as security issues as expanding step by step in business. It is useful as it guarantees that business information is put away and transported securely. For recognizing access strategies to a system by utilizing a few apparatuses and procedures utilized by programmers can ensure the security of cloud arrangements.

In a cloud domain, arrange security is generally vital. A few security machines are in broad utilize, which ensures endeavors and server farms. These gadgets include the parts of interruption anticipation systems, firewalls, against infection, hostile to spam, and information misfortune avoidance. Security components are tried in three measurements: Accuracy, Effectiveness, and Performance.

Cloud scalability and performance testing. Cloud Scalability is that region of concern where the best possible measure of testing is required. Cloud Computing arrangements are constantly adaptable on demand. Cloud scalability and performance testing procedures assist us in measuring the cloud systems performance precisely and painstakingly (Kuma, 2016). Performance testing is in charge of discovering edges, bottlenecks, and constraints (Conner et al., 2009). Thus, performance testing is the endeavor to gauge response times and issues related to specific exercises while the structure is subjected to expanding load from different multi-customer errands. It chooses the point of confinement, responsiveness, dependability, throughput, and additionally the flexibility of a system under a given workload. It can survey age readiness, evaluate execution criteria, discover structure limits, think about execution characteristics of various systems, discover the wellspring of execution issues, bolster system tuning, and discover throughput levels.

Cloud load and stress testing. Load testing is used for making overpowering customer development and evaluating its response. It moreover tunes the execution of any application to meet particular rules. Then again, stress testing decides the ability of uses to keep up a specific level of adequacy. For any application, it is essential to work even under the best weight and care for security. Stress testing does this by making peak loads using test systems.

Latency testing. This testing includes estimating the latency (delay) between the activity and the reaction for any application in the wake of conveying it on the cloud.

Ability testing systems. Ability testing is done to guarantee that users get the fitting administrations from the cloud condition on demand. Under this classification, compatibility and interoperability testing, disaster recovery testing, multi-occupancy testing are performed.

Compatibility and interoperability testing. It is testing performed on the application to evaluate the application's similitude with the enrolling condition. A cloud application must be fit for working crosswise over different conditions and executed on different cloud stages. Thus, it is less demanding for the relocation of cloud applications starting with one foundation then onto the next. A compatibility test incorporates (an) equipment designs, (b) distinct stages, (c) PC peripherals, and (d) organize condition.

Disaster recovery testing. Disasters are an unavoidable conviction for any affiliation, yet while sure, fiascos are in like manner generally sporadic. The specialist organization of cloud favors that its cloud administrations must be accessible to users constantly. Disaster recovery time must be low after some disappointment happens. This testing is done to guarantee that the cloud administrations must be accessible to the user after some disappointment happens, with least or no information misfortune.

Multi-occupancy testing. Multi-tenure alludes to a guideline where a single example of the product keeps running on a server, serving different customer associations. It alludes to various association and customers utilizing on-demand advertising. The offering ought to be adaptable for every customer and ought to give information and additionally design level security to maintain a strategic distance from any entrance related issues.

Advantages of cloud testing. Cloud testing helps business in setting and keeping up the conditions, on-task for benefits, cuts down cost, resource pooling, end of capital utilization early, and more broad system get to, quick adaptability and decreasing in cycle diminishment time for various business offering in this powerful and quick going administrations and IT industry.

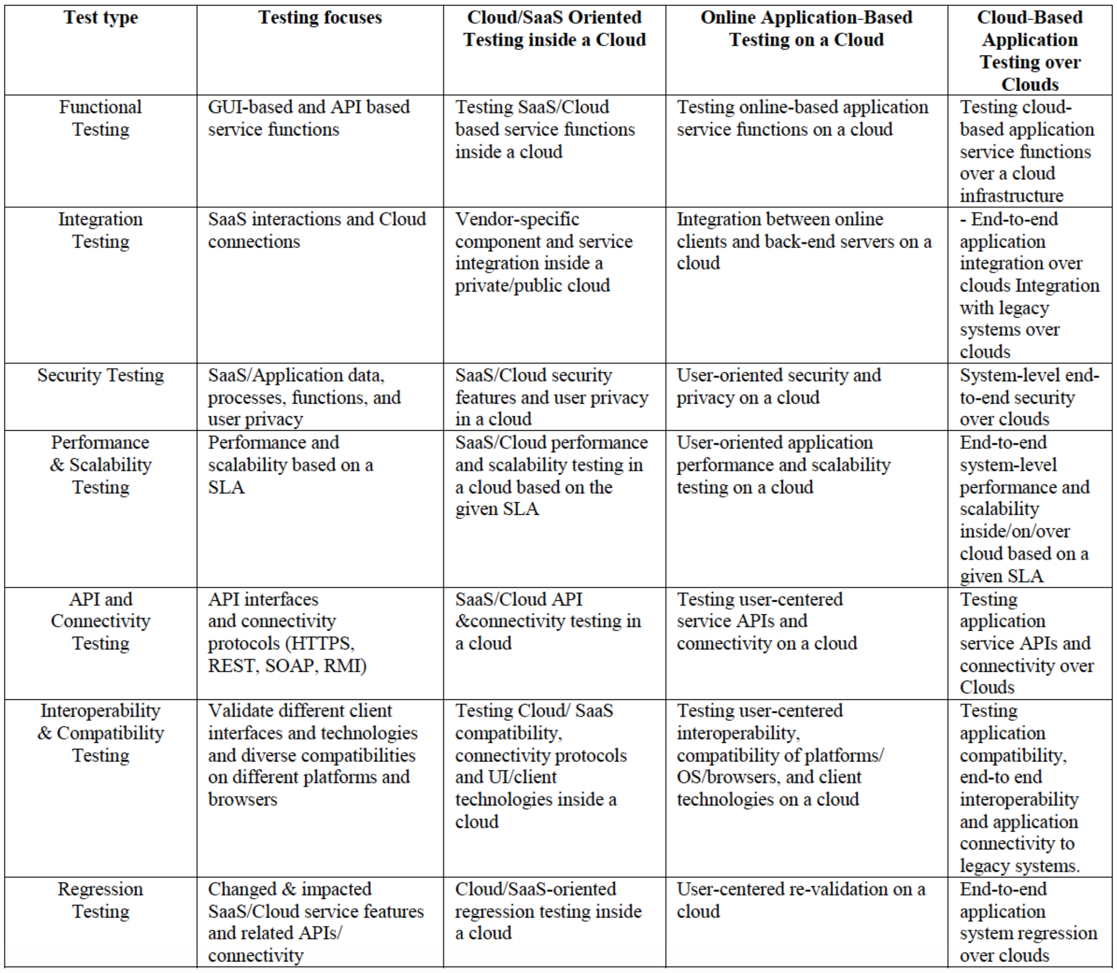
The following are a portion of the critical advantages concerning testing in the cloud:

1. Rapid provision of the test environment(s). Due to the dynamic nature of the real-world application regarding requirements and user in a short period, it is challenging for many companies to set up the infrastructure for testing use. With the help of the cloud, it is effortless for an organization to turn up the testing environments to fulfill project timelines. Test imitation of generation conditions can be reproduced by business/clients which help testing groups to approve the business situations and discover bugs more quickly. Reduced capital expense – cloud will take care of setting up for the testing infrastructure as when needed by the organization and decommission all the setup of servers once the testing is done which helps companies

to save their money. It helps many companies to get the work done at a lower price as compared to earlier as there is no cost associated with them in setting up that entire infrastructure in advance.

2. Fast customization of equipment assets. As companies are permitting cloud condition for their testing, it is simple for the association to reenact the generation situations to check the load, performance testing, verifying the scenarios in different environments with multiple browsers in different operating systems and the latest versions available in the market.
3. Support green computing and reducing carbon footprint.
4. Computing is the study of using computing resources efficiently. The global use of computing resources continues to grow dramatically due to the vast IT market and different industries. As the clear majority of the organizations have begun receiving cloud procedures which give the framework in light of interest, cloud arrangement empowers organizations to wind up more naturally agreeable. Practical use of resources, the cloud will take care of all business needs as and when required. By using the cloud, server efficiency and utilization have been drastically improved through the even distribution of workload. Testing in the cloud obtained the current distributed computing framework gave by the seller who helps in diminishing the cost of figuring, all things considered, with expanding testing effectiveness in the process.

Comparison among various cloud testing techniques and platforms. Following are the comparisons and various cloud testing platforms which are considered to be the key players in the cloud testing zone. These are the leading cloud testing providers as per the five categories: infrastructure, platform, security, storage, and software.



Test type	Testing focuses	Cloud/SaaS Oriented Testing inside a Cloud	Online Application-Based Testing on a Cloud	Cloud-Based Application Testing over Clouds
Functional Testing	GUI-based and API based service functions	Testing SaaS/Cloud based service functions inside a cloud	Testing online-based application service functions on a cloud	Testing cloud-based application service functions over a cloud infrastructure
Integration Testing	SaaS interactions and Cloud connections	Vendor-specific component and service integration inside a private/public cloud	Integration between online clients and back-end servers on a cloud	- End-to-end application integration over clouds Integration with legacy systems over clouds
Security Testing	SaaS/Application data, processes, functions, and user privacy	SaaS/Cloud security features and user privacy in a cloud	User-oriented security and privacy on a cloud	System-level end-to-end security over clouds
Performance & Scalability Testing	Performance and scalability based on a SLA	SaaS/Cloud performance and scalability testing in a cloud based on the given SLA	User-oriented application performance and scalability testing on a cloud	End-to-end system-level performance and scalability inside/on/over cloud based on a given SLA
API and Connectivity Testing	API interfaces and connectivity protocols (HTTPS, REST, SOAP, RMI)	SaaS/Cloud API &connectivity testing in a cloud	Testing user-centered service APIs and connectivity on a cloud	Testing application service APIs and connectivity over Clouds
Interoperability & Compatibility Testing	Validate different client interfaces and technologies and diverse compatibilities on different platforms and browsers	Testing Cloud/ SaaS compatibility, connectivity protocols and UI/client technologies inside a cloud	Testing user-centered interoperability, compatibility of platforms/ OS/browsers, and client technologies on a cloud	Testing application compatibility, end-to end interoperability and application connectivity to legacy systems.
Regression Testing	Changed & impacted SaaS/Cloud service features and related APIs/ connectivity	Cloud/SaaS-oriented regression testing inside a cloud	User-centered re-validation on a cloud	End-to-end application system regression over clouds

Figure 13. Comparison Of Different Testing Techniques (Shrivastva et al., 2014)

Functional tests give orderly demonstrations that capacities tried are accessible as determined by the business and functional necessities, framework documentation, and client manuals.

Functional testing is focused on the accompanying things:

- Valid Input   Identified classes of valid input must be acknowledged.

• Invalid Input	Distinguished classes of invalid data must be rejected.
• Functions	Recognized limits must be worked out.
• Output	Identified classes of use outputs must be worked out.
• Systems/Procedures	Interfacing systems or methodology must be conjured.

Association and readiness of functional tests are centered around prerequisites, fundamental functions, or individual experiments. Also, systematic coverage about identifies business process flows; data fields, predefined processes, and continuous processes must be considered for testing. Before functional testing is finished, extra tests are distinguished, and the compelling estimation of current tests is resolved.

White box testing. White box testing is testing in which the software tester knows the internal workings, structure, and dialect of the product, or possibly its motivation. It is utilized to test zones that can't become to form a black box level.

Black 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 different sorts of tests, must be composed of an authoritative source report, for example, determination or prerequisites record or a particular or requirements document. Blackbox tests, as most different sorts of tests, must be composed of a particular source record, for example, detail or prerequisites archive, or a determination or necessities report. It is testing in which the software under test is treated, as a black box and one cannot "see" into it. The test gives sources of info and reacts to yields without considering how the product functions.

Unit testing. Unit testing is generally accepted as a significant aspect of a joined code and unit test period of the software lifecycle, even though it is not phenomenal for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach. Field testing will be performed manually, and useful tests will be composed in detail.

Test objectives:

- All field entries must work correctly.
- Pages must be enacted from the recognized connection.
- The passage screen, messages, and reactions must not be deferred.

Highlights to be tested:

- Verify that the sections are of the right organization.
- No copy sections ought to be permitted.
- All connections should take the client to the right page.

Integration testing. Software integration testing is the incremental integration testing of at least two coordinated software parts on a single stage to deliver disappointments caused by interface defects.

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

Acceptance testing. User Acceptance Testing is a necessary period of any task and requires critical cooperation by the end client. It additionally guarantees that the framework meets the practical necessities.

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

Chapter V: Conclusion

Given the highly dynamic, distributed, and nontransparent nature of cloud services, managing and establishing trust between cloud service users and cloud services remains a significant challenge. Cloud service consumers' criticism is a decent source to evaluate the general dependability of cloud administrations. Be that as it may, noxious clients may work together to:

- The disadvantage of a cloud service by giving multiple misleading trust feedbacks (i.e., collision attacks), or
- Trap clients into trusting cloud benefits that are not reliable by making a few records and giving deceiving trust feedbacks (i.e., Sybil attacks).

In this paper, novel techniques that help in detecting reputation-based attacks and allowing users to identify trustworthy cloud services effectively are presented. Specifically, a credibility model demonstrates that not just distinguishes misdirecting trust criticisms from collusion attacks. Besides, identifies Sybil attacks regardless of these attacks happen in a lot of cloud vendors, service providers, and tenants must establish a shared view on cloud security to establish and drive trusted business in a short period (i.e., strategic or occasional attacks respectively) is introduced. An availability model that maintains the trust management service at the desired level is also introduced. A large number of consumers' trust feedback given on real-world cloud services (i.e., over 10,000 records) are collected to evaluate my proposed techniques. The experimental results demonstrate the applicability of the approach and show the capability of detecting such malicious behaviors.

Vendors of cloud, service provider and occupants must build up a standard view on cloud security keeping in mind the end goal to set up and drive trusted business. This shared view will enable cloud service providers to understand and fulfill the security needs of their customers. In various sectors, such as healthcare, automotive, manufacturing, public utilities or banking, these needs are also subject to applicable regulations.

Through the Trust Engine, this paper provides a conceptual basis to support different actors in their discussions about cloud security risks, threats, controls, management, and compliance, as well as other security requirements in a cloud system. Also, different design factors, best security design practices and the application of the right security technologies will need to be taken into account.

A right cloud service provider can combine a solid conceptual foundation and a shared understanding of customer needs with technical know-how in design and implementation. It will allow it to provide, in a cost-efficient way, more reliable operations, networks and components than many customers could achieve independently, and to fully realize the benefits of sharing resources through a cloud model.

There are a couple of directions for our future work. Distinctive trust management procedures, for example, reputation and proposal to expand the trust comes about precision can be intended to join for improving numerous factors. Performance optimization of the trust management benefit is another focal point of our future research work.

References

Ba, S., & Pavlou, P. (2002). Evidence of the effect of trust building technology in electronic markets: Price premiums and buyer behavior. *MIS Quarterly*, 26(3), 24j3-268.

Balachandran, N., & Sanyal S. (2012). A review of techniques to mitigate Sybil attacks. *International Journal of Advanced Networking Applications*, 1514-1518.

Banga, C., Druschel, P., & Mogul, J. C. (1999). Resource containers: A new facility for resource management in server systems. In *Proceedings of 3rd USENIX Symposium on Operating Systems Design and Implementation (ODSI)*, New Orleans, LA.

Chen, S., Falsafi, B. Gibbons, P. B., Kozuch, M., Mowry, T. C., Teodorescu, R., . . . Schlosser, S. W. (2006). Log-based architectures for general-purpose monitoring of deployed code. In *Proceedings of the 1st Workshop on Architectural and System Support for Improving Software Dependability* (pp. 63-65), San Jose, California.

Conner, W., Iyengar, A., Mikalsen, T., Rouvellou, I., & Nahrstedt, K. (2009). *Trust management framework for service-oriented environments*. A trust management framework for service-oriented environments. In *Proceedings of the 18th International Conference on World Wide Web*, Spain, 891-900.

Course Hero. (n.d.). *What is the internet*? Retrieved from https://www.coursehero.com/file/23710838/What-is-the-Internet/

Douceur, J. R. (2002). The Sybil attack. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS)*, Cambridge.

Ericsson White Paper. (2015). *Cloud security architecture*. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.670.4602&rep=rep1&type=pdf

Friedman, E., Resnick, P., & Sami. R. (2007). *Algorithmic game theory*. New York, NY: Cambridge University Press.

Haeberlen, A., Kouznetsov, P., & Druschel, P. (2007). Peer review: Practical accountability for distributed systems. In *SOSP 07 Proceedings of 21st ACM SIGOPS Symposium on Operating Systems Principles*, Stevenson, Washington.

Karlof, C., & Wagner, D. (2003). Secure routing in wireless sensor networks: Attacks and countermeasures. *Ad hoc Networks Journal, 1*(2-3), 293-315.

Kumar, V. (2016). Brief review on cloud computing. *International Journal of Computer Science and Mobile Computing, 5*(9), 1-5.

Lai, K., Feldman, M., Stoica, I., & Chuang, J. (2003). Incentives for cooperation in peer-to-peer networks. In *Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems,* University of Berkley, CA.

McCune, J. M., Parno, B. J., Perrig, A., Reiter, M. K., & Isozaki, H. (2008). Flicker: An execution infrastructure for TCB minimization. In *EuroSys, 2008,* Glasglow, Scotland.

Mogul, J. C. (2005). Operating systems should support business change. In *Proceedings of 10th Workshop on Hot Topics in Operating Systems HotOS X*, Santa Fe, NM, 43-48. Retrieved from https://pdfs.semanticscholar.org/a2b4/29090c4755dffa9b3f53f9c6132be9152f53.pdf

Molka, R., Boukadi, K., & Ben-Abdallah, H. (2015). Cloud description ontology for service discovery and selection. In *Software Technologies, 2015 10th International Joint Conference*.

Noor, T. H., Sheng, Q. Z., Yao, L., Dustdar, S., & Ngu, A. H. (2015). Cloudarmor: Supporting reputation-based trust management for cloud services. In *IEEE Transactions on Parallel and Distributed Systems, 0*(0).

Ns-2 Simulator. (n.d.). *What is Sybil attack*? Retrieved from http://ns2simulator.com/sybil-attack-in-ns2/

Rowaihy, H., Enck. W., McDaniel, P., & La Porta, T. (2007). Limiting Sybil attacks in structured p2- networks. In *INFOCOM, 26th IEEE International Conference on Computer Communication*, 2956.

Sekar, V., & Maniatis, P. (2011). Verifiable resource accounting for cloud computing services. In *Proceedings of the 3rd ACM Workshop on Cloud Computing Security*, Chicago, IL. 21-26.

Sharif, M. I., Lee, W., Cui, W., & Lanzi, A. (2009). Secure-VM monitoring using hardware virtualization. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, Chicago, IL.

Shrivastva, A., Gupta, S., & Tiwari, R. (2014). Cloud-based testing techniques. *International Journal of Computer Applications, 104*(5), 975-8887.

Vyshnavi, U., & Yadav. P. P. (2016). Trust management of cloud services using credibility assessment technique. *International Journal and Magazine of Engineering, Technology, Management and Research, 3*(10), 514-519. Retrieved from http://www.ijmetmr.com/oloctober2016/UVyshnavi-PPraveenYadav-71.pdf

Xiong, L., & Liu, L. (2004). Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. *IEEE Transactions on Knowledge and Data Engineering, 16*(7), 843-857.

Zhou, F., Goel, M., Desnoyers, P., & Sundaram, R. (2011). *Scheduler vulnerabilities and attacks in cloud computing*. Retrieved from https://archive.org/stream/arxiv-1103.0759/1103.0759#page/n0/mode/2up

Appendix A: Code Base

Some of the main parts of the code base in the system implementation are as below: -

Frontend Code Base

Home Page

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8" />
    <title>CloudArmor</title>
    <meta name="keywords" content="" />
    <meta name="description" content="" />
    <link href="styles.css" rel="stylesheet" type="text/css" media="screen" />
    <link href='http://fonts.googleapis.com/css?family=Merienda+One' rel='stylesheet' type='text/css' />
  </head>
  <body>
    <div id="header_bg">
      <div id="logo">
        <h1><a href="#" style="color: black">CloudArmor</a></h1>
        <center>
          <h1 style="font-size: 26px;line-height: 30px;font-family: 'Merienda One',
cursive;">Supporting Reputation-based
            Trust Management<br /> for Cloud Services</h1>
        </center>
      </div>
      <div id="prew_img">
        <ul class="round">
        <li><img src="images/header1.jpg" alt="" /></li>
        <li><img src="images/header2.jpg" alt="" /></li>
        <li><img src="images/header3.jpg" alt="" /></li>
        <li><img src="images/header4.jpg" alt="" /></li>
        <li><img src="images/header5.jpg" alt="" /></li>
        <li><img src="images/header6.jpg" alt="" /></li>
    </ul>
    <script type="text/javascript" src="lib/jquery.js"></script>
    <script type="text/javascript" src="lib/jquery.roundabout.js"></script>
    <script type="text/javascript">
        $(document).ready(function () {
            $('.round').roundabout();
        });

    </script>
</div>
<div id="menu">
    <ul>
        <li><a href="thome.jsp" class="active">Home</a></li>
        <li><a href="tuserdetails.jsp">User Details</a></li>
        <li><a href="tpdetails.jsp">Service_Details</a></li>
        <li><a href="tpfeedback.jsp">Feedback</a></li>
        <li><a href="index.html">Log out</a></li>
    </ul>
    <div class="clear"></div>
</div>
<div id="black_bg" style="height: 300px;color: white;background: url('images/tms.png')">
```

```
            </div>
        </div>
    </body>
</html>
```

Publishing Feedback to Backend DB

```
<%@page import="com.cloudarmor.kk.action.Dbconnection"%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.Connection"%>
<%
    Connection con = null;
    Statement st = null;
    Statement st1 = null;
    ResultSet rs = null;
    String[] data = request.getQueryString().split(",");
    System.out.println("data[0-]"+data[0]);
    String pitem = data[0].replace("%20"," ");
    System.out.println("data[0-1]"+pitem);

    try {
        con = Dbconnection.getConnection();
        st = con.createStatement();
        rs = st.executeQuery("select * from feed where status='No' AND feedback='" + pitem + "' AND
uid='" + data[1] + "'");
        if (rs.next()) {
```

```
                    st1 = con.createStatement();
                    int i = st1.executeUpdate("update feed set status='Yes' where status='No' AND feedback='" + 
data[0] + "'AND uid='" + data[1] + "'");
                    if (i != 0) {
                        response.sendRedirect("tpfeedback.jsp?pmsg=sucess");
                    }

                } else {
                    response.sendRedirect("tpfeedback.jsp?msgg=failed");
                }
            } catch (Exception ex) {
                System.out.println("Exception error in Publish"+ex.getMessage());
            }

        %>
```

Cart Details Page

```
        <%@page import="com.cloudarmor.kk.action.Dbconnection"%>
        <%@page import="java.sql.ResultSet"%>
        <%@page import="java.sql.Statement"%>
        <%@page import="java.sql.Connection"%>
        <%@page contentType="text/html" pageEncoding="UTF-8"%>
        <!DOCTYPE html>
        <html xmlns="http://www.w3.org/1999/xhtml">
            <head>
                <meta http-equiv="content-type" content="text/html; charset=utf-8" />
                <title>CloudArmor</title>
```

```
    <meta name="keywords" content="" />
    <meta name="description" content="" />
    <link href="styles.css" rel="stylesheet" type="text/css" media="screen" />
    <link href='http://fonts.googleapis.com/css?family=Merienda+One' rel='stylesheet' type='text/css' />
  </head>
  <style>
    #signup-form {
      width: 500px;
      margin: 0 auto;
      margin-top: 50px;
      margin-bottom: 50px;
      background: #fff;
      padding: 40px;
      border: 10px solid #f2f2f2;
      height: 200px;
      border-radius: 25px;
    }
  </style>
  <body>
    <div id="header_bg">
      <div id="logo">
        <h1><a href="#" style="color: black">CloudArmor</a></h1>
        <center>
          <h1 style="font-size: 26px;line-height: 30px;font-family: 'Merienda One',
cursive;">Supporting Reputation-based
            Trust Management<br /> for Cloud Services</h1>
        </center>
```

```
</div>
<div id="prew_img">
    <ul class="round">
        <li><img src="images/header1.jpg" alt="" /></li>
        <li><img src="images/header2.jpg" alt="" /></li>
        <li><img src="images/header3.jpg" alt="" /></li>
        <li><img src="images/header4.jpg" alt="" /></li>
        <li><img src="images/header5.jpg" alt="" /></li>
        <li><img src="images/header6.jpg" alt="" /></li>
    </ul>
    <script type="text/javascript" src="lib/jquery.js"></script>
    <script type="text/javascript" src="lib/jquery.roundabout.js"></script>
    <script type="text/javascript">
        $(document).ready(function () {
            $('.round').roundabout();
        });

    </script>

</div>
<div id="menu">
    <ul>
        <ul>
            <li><a href="uhome.jsp" class="active">Home</a></li>
            <li><a href="profile.jsp">User Profile</a></li>
            <li><a href="search.jsp">Search</a></li>
            <li><a href="cartdetails.jsp">Service List</a></li>
```

```
                <li><a href="index.html">Log out</a></li>
            </ul>
            <div class="clear"></div>
        </div>
        <div id="black_bg" style="height: 300px;background: transparent">
            <div id="signup-form" style="">
                <center> <h1 style="margin-top: 3px">Cloud Order Details</h1></center><br />
                <table style="margin-left: 100px;text-align: center" border="1">
                    <tr>
                        <th>Service Model</th>
                        <th>Service Name</th>
                        <th>Duration</th>
                        <th>Total Price</th>
                    </tr>
                    <tr>
                        <%
                            String uname = session.getAttribute("email").toString();
                            String pitem = null;
                            try {
                                Connection con = Dbconnection.getConnection();
                                Statement st = con.createStatement();
                                ResultSet rs1 = st.executeQuery("select * from cart where status='NO' AND
name='" + uname + "'");
                                while (rs1.next()) {
                                    pitem = rs1.getString("productitem");
                        %>
                        <td><%=rs1.getString("productname")%></td>
                    <td><%=rs1.getString("productitem")%></td>
                    <td><%=rs1.getString("quantity")%> Days</td>
                    <td><%=rs1.getString("total")%></td>
                  </tr>
                  <%
                      }
                    } catch (Exception e) {
                      System.out.println("Exception Error in cartdetails " + e.getMessage());
                    }
                  %>

                </table><br />
                <a href="paction.jsp?<%=pitem%>"><button class="button"  style="margin-left:
200px;width: 100px;height: 30px;background: orange;border: currentColor;border-radius: 10px;font-size:
15px;">Buy Cloud</button></a>
              </div>
            </div>
          </div>
        </body>
      </html>
```

Backend Code Base

Process Upload Request

```
        package com.cloudarmor.kk.action;


        import java.io.DataInputStream;
        import java.io.File;
```

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.UUID;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

public class Upload extends HttpServlet {

    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
response.setContentType("text/html;charset=UTF-8");
try (PrintWriter out = response.getWriter()) {
    /* TODO output your page here. You may use following sample code. */
    HttpSession session = request.getSession(true);
    String saveFile = "";
    String contentType = request.getContentType();
    if ((contentType != null) && (contentType.indexOf("multipart/form-data") >= 0)) {
        DataInputStream in = new DataInputStream(request.getInputStream());
        int formDataLength = request.getContentLength();
        byte dataBytes[] = new byte[formDataLength];
        int byteRead = 0;
        int totalBytesRead = 0;
        while (totalBytesRead < formDataLength) {
            byteRead = in.read(dataBytes, totalBytesRead, formDataLength);
            totalBytesRead += byteRead;
        }
        String file = new String(dataBytes);
        saveFile = file.substring(file.indexOf("filename=\"") + 10);
        saveFile = saveFile.substring(0, saveFile.indexOf("\n"));
        saveFile = saveFile.substring(saveFile.lastIndexOf("\\") + 1, saveFile.indexOf("\""));
        int lastIndex = contentType.lastIndexOf("=");
        String boundary = contentType.substring(lastIndex + 1, contentType.length());
        int pos;
        pos = file.indexOf("filename=\"");
        pos = file.indexOf("\n", pos) + 1;
        pos = file.indexOf("\n", pos) + 1;
pos = file.indexOf("\n", pos) + 1;
int boundaryLocation = file.indexOf(boundary, pos) - 4;
int startPos = ((file.substring(0, pos)).getBytes()).length;
int endPos = ((file.substring(0, boundaryLocation)).getBytes()).length;
File ff = new File(saveFile);
System.out.println("The File Location " + ff);
FileOutputStream fileOut = new FileOutputStream(ff);
fileOut.write(dataBytes, startPos, (endPos - startPos));
fileOut.flush();
fileOut.close();


/*Random Key Generation*/
String imgid = UUID.randomUUID().toString().substring(0, 3);
/**/
Connection = null;
ResultSet rs = null;
PreparedStatement psmnt = null;
FileInputStream fis;
try {
   connection = Dbconnection.getConnection();
   File f = new File(saveFile);
   psmnt = connection.prepareStatement("insert into product(imgid,image,iname)values(?,?,?)");
   fis = new FileInputStream(f);
   psmnt.setString(1, imgid);
   psmnt.setBinaryStream(2, (InputStream) fis, (int) (f.length()));
   psmnt.setString(3, saveFile);
   int s = psmnt.executeUpdate();
```

```
                if (s > 0) {
                    session.setAttribute(“imgid”, imgid);
                    session.setAttribute(“fname”, saveFile);
                    System.out.println(“Uploaded successfully !”);
                    response.sendRedirect(“additems.jsp”);
                } else {
                    System.out.println(“Error!”);
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}

// <editor-fold defaultstate=“collapsed” desc=“HttpServlet methods. Click on the + sign on the left to
edit the code.”>
/**
 * Handles the HTTP <code>GET</code> method.
 *
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error occurs
 * @throws IOException if an I/O error occurs
 */
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
```

```
        throws ServletException, IOException {
    processRequest(request, response);
}

/**
 * Handles the HTTP <code>POST</code> method.
 *
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error occurs
 * @throws IOException if an I/O error occurs
 */
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    processRequest(request, response);
}

/**
 * Returns a short description of the servlet.
 *
 * @return a String containing servlet description
 */
@Override
public String getServletInfo() {
    return “Short description”;
}// </editor-fold>
```

```
}
```

Email Sender

```
package com.cloudarmor.kk.action;


import com.sun.mail.smtp.SMTPTransport;
import java.util.Properties;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.URLName;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;


public class Mail {


  public static boolean sendMail(String msg, String userid, String to) {
    Properties props = new Properties();
    props.put("mail.smtp.host", "smtp.gmail.com");
    props.put("mail.smtp.socketFactory.port", "465");
    props.put("mail.smtp.socketFactory.class",
        "javax.net.ssl.SSLSocketFactory");
    props.put("mail.smtp.auth", "true");
    props.put("mail.smtp.port", "465");
```

```
// Assuming you are sending email from localhost
Session = Session.getDefaultInstance(props,
        new javax.mail.Authenticator() {
            protected PasswordAuthentication getPasswordAuthentication() {
                return new PasswordAuthentication("cloudmail107@gmail.com", "cloudmail123");
            }
        });

System.out.println("Message   " + msg);
try {
    Message = new MimeMessage(session);
    message.setFrom(new InternetAddress(userid));
    message.setRecipients(Message.RecipientType.TO,
            InternetAddress.parse(to));
    message.setSubject("Status Mail ");
    message.setText(msg);

    Transport.send(message);

    System.out.println("Done");
    return true;

} catch (MessagingException e) {
    System.out.println(e);
    e.printStackTrace();
    return false;
    // throw new RuntimeException(e);
```

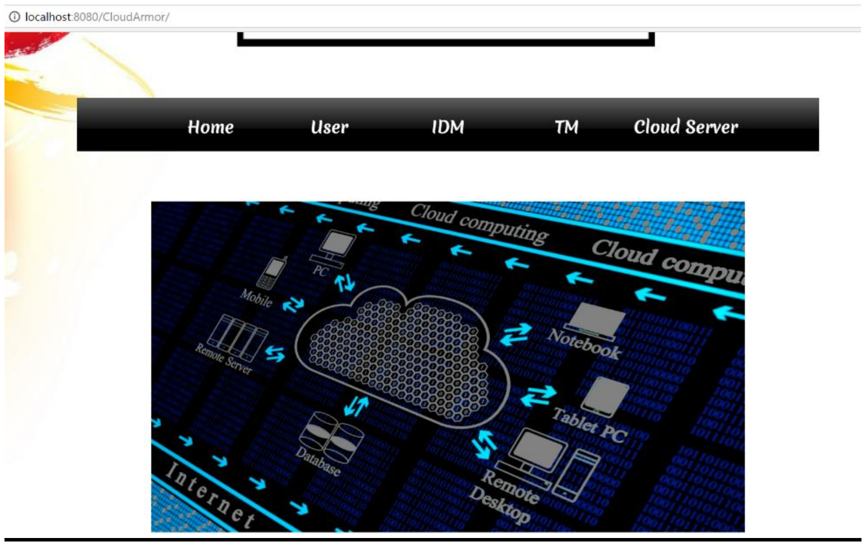
```
        }
    }
}
```

Maintaining DB Connection

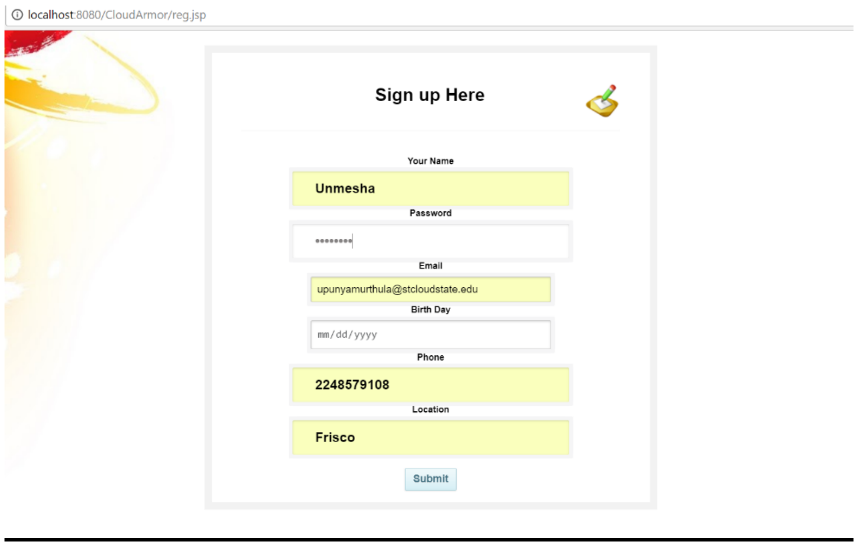
```
package com.cloudarmor.kk.action;


import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;


public class Dbconnection {
    public static Connection con;
        public static Connection getConnection() throws ClassNotFoundException, SQLException
        {
            Class.forName(“com.mysql.jdbc.Driver”);
            con = DriverManager.getConnection(“jdbc:mysql://localhost:3306/cloudarmor”,”root”,”root”);
            return con;
        }
}
```

Appendix B: Screenshots

Home Page



User Login Page - Registration



Sign up Here

Your Name

Unmesha

Password

••••••••

Email

upunyamurthula@stcloudstate.edu

Birth Day

mm/dd/yyyy

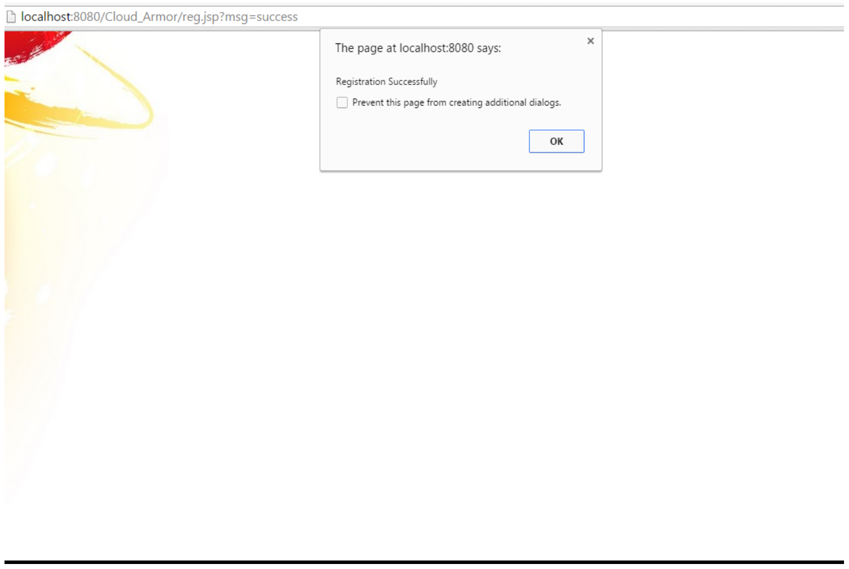
Phone

2248579108

Location

Frisco

Submit



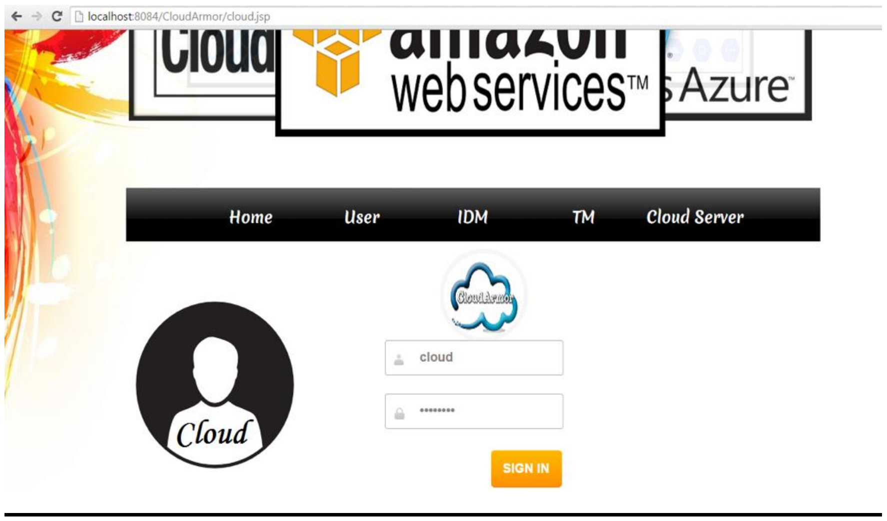
localhost:8080/Cloud_Armor/reg.jsp?msg=success

The page at localhost:8080 says:

Registration Successfully

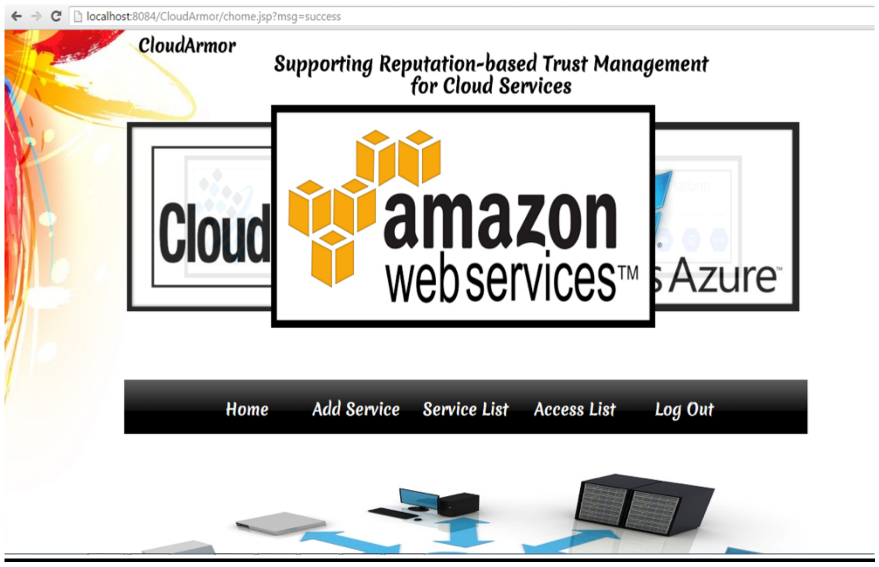
Prevent this page from creating additional dialogs.

Cloud Server Login Page

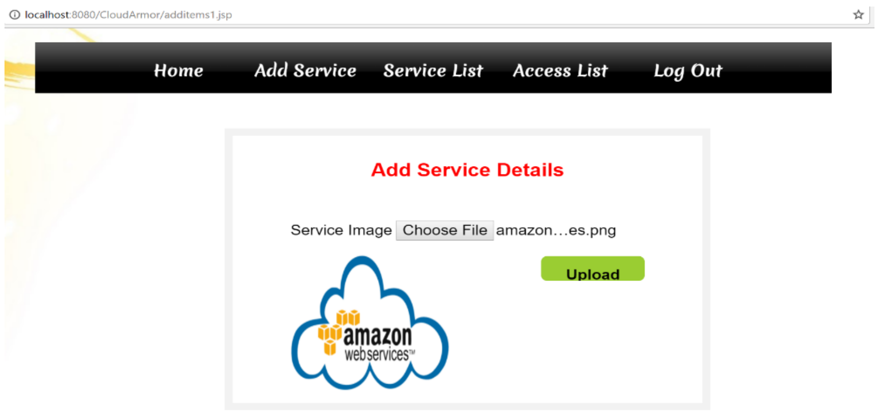


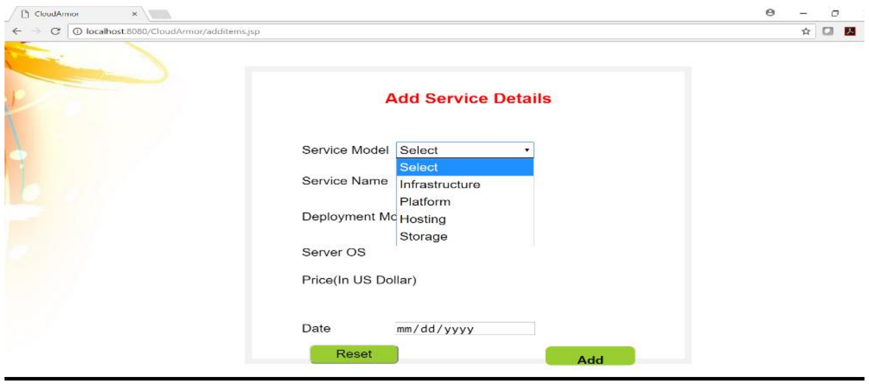
localhost:8084/CloudArmor/cloud.jsp

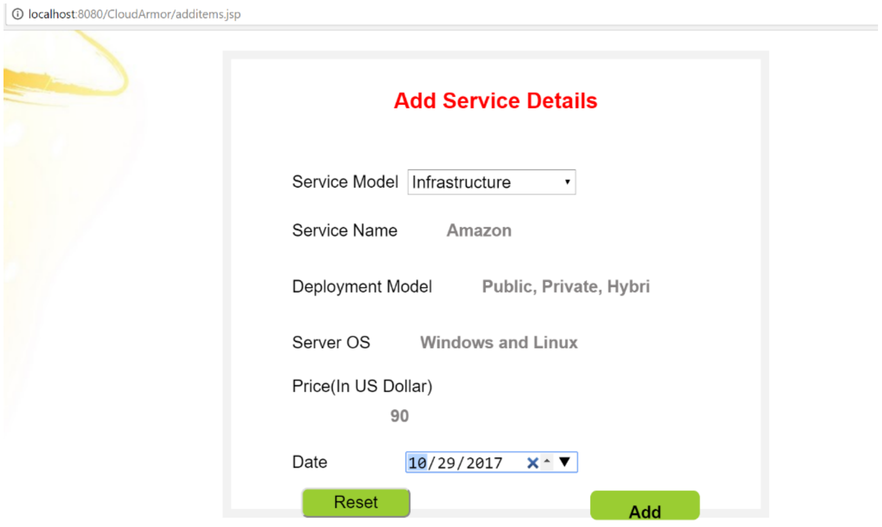
Cloud Server Home Page


localhost:8084/CloudArmor/chome.jsp?msg=success
CloudArmor
Supporting Reputation-based Trust Management for Cloud Services
Home
Add Service
Service List
Access List
Log Out

Add Service



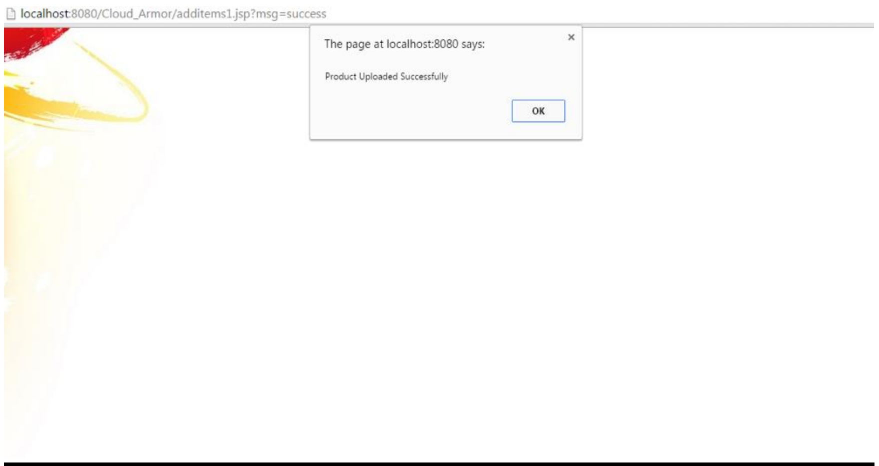




localhost:8080/CloudArmor/additems.jsp

Add Service Details

Service Model	Infrastructure
Service Name	Amazon
Deployment Model	Public, Private, Hybri
Server OS	Windows and Linux
Price(In US Dollar)	90
Date	10/29/2017

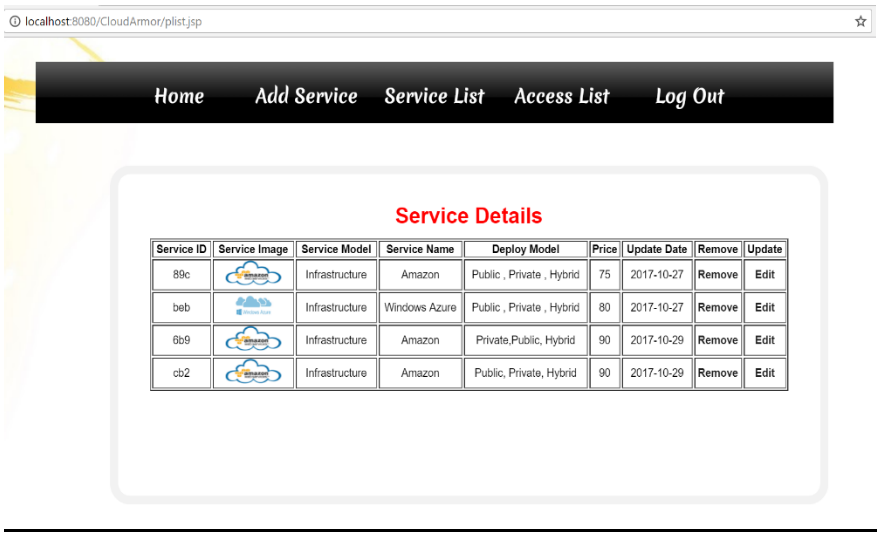


localhost:8080/Cloud_Armor/additems1.jsp?msg=success

The page at localhost:8080 says:

Product Uploaded Successfully

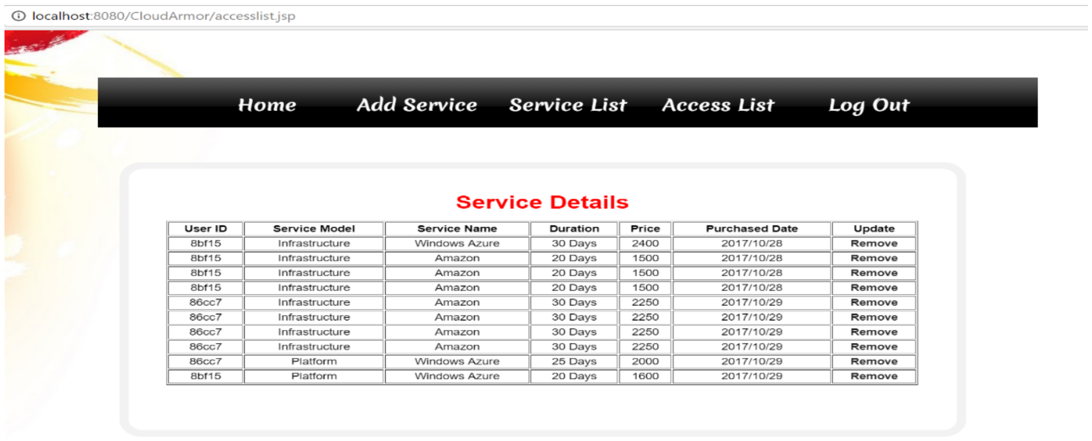
Service List



Service Details

Service ID	Service Image	Service Model	Service Name	Deploy Model	Price	Update Date	Remove	Update
89c		Infrastructure	Amazon	Public , Private , Hybrid	75	2017-10-27	Remove	Edit
beb		Infrastructure	Windows Azure	Public , Private , Hybrid	80	2017-10-27	Remove	Edit
6b9		Infrastructure	Amazon	Private,Public, Hybrid	90	2017-10-29	Remove	Edit
cb2		Infrastructure	Amazon	Public, Private, Hybrid	90	2017-10-29	Remove	Edit

Access

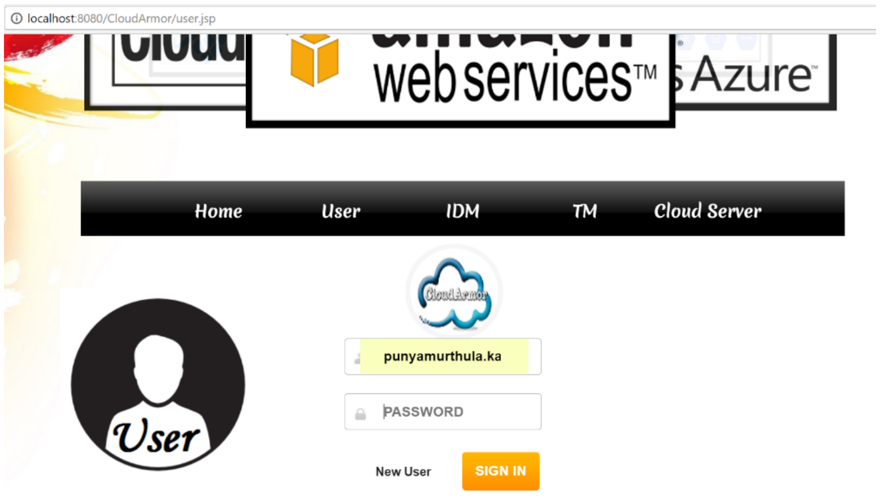


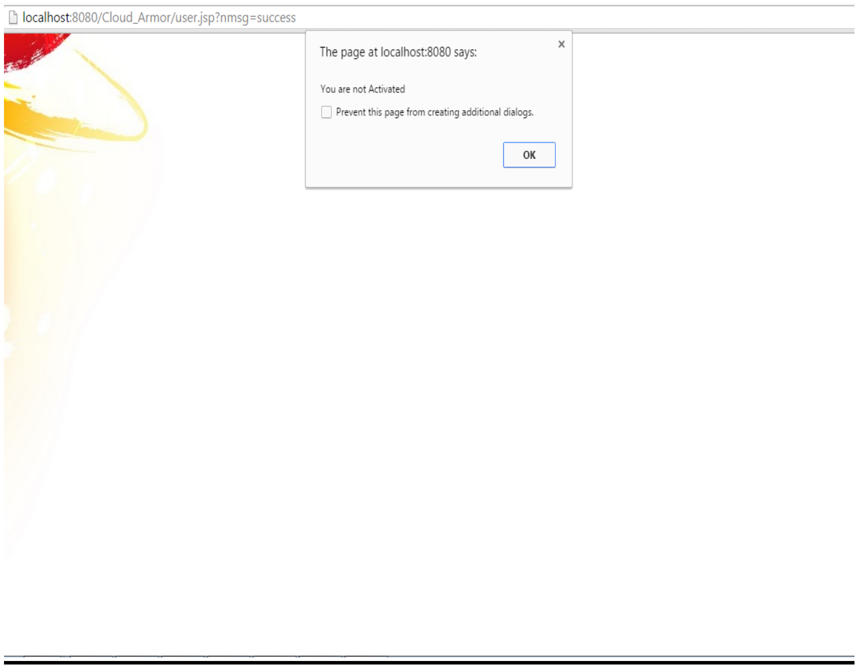
Service Details

User ID	Service Model	Service Name	Duration	Price	Purchased Date	Update
8bf15	Infrastructure	Windows Azure	30 Days	2400	2017/10/28	Remove
8bf15	Infrastructure	Amazon	20 Days	1500	2017/10/28	Remove
8bf15	Infrastructure	Amazon	20 Days	1500	2017/10/28	Remove
8bf15	Infrastructure	Amazon	20 Days	1500	2017/10/28	Remove
86cc7	Infrastructure	Amazon	30 Days	2250	2017/10/29	Remove
86cc7	Infrastructure	Amazon	30 Days	2250	2017/10/29	Remove
86cc7	Infrastructure	Amazon	30 Days	2250	2017/10/29	Remove
86cc7	Infrastructure	Amazon	30 Days	2250	2017/10/29	Remove
86cc7	Platform	Windows Azure	25 Days	2000	2017/10/29	Remove
8bf15	Platform	Windows Azure	20 Days	1600	2017/10/29	Remove

List

User Login




localhost:8080/Cloud_Armor/user.jsp?nmsg=success

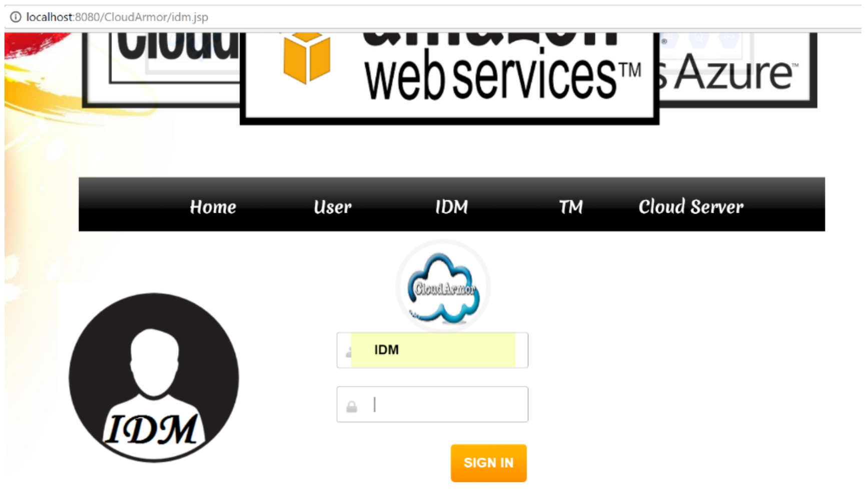
The page at localhost:8080 says:

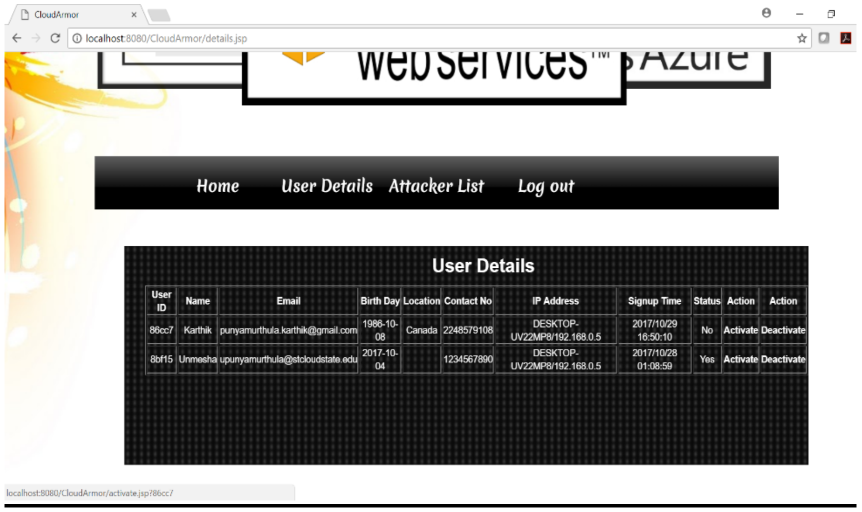
You are not Activated

Prevent this page from creating additional dialogs.

OK

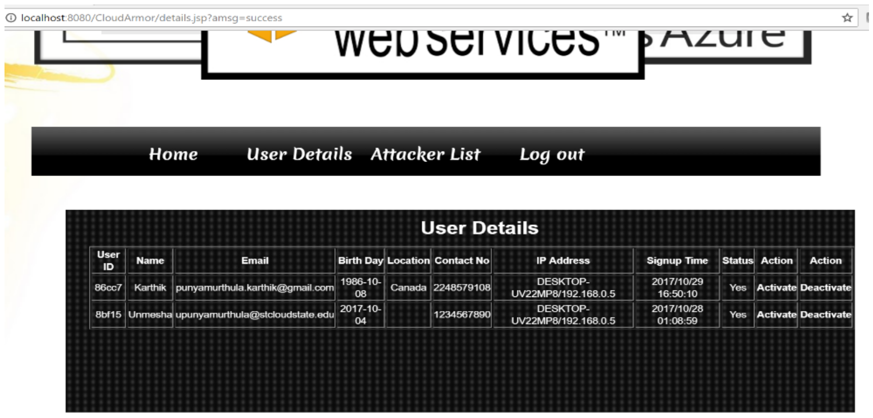
IDM




CloudArmor
localhost:8080/CloudArmor/details.jsp
web services
Azure
Home
User Details
Attacker List
Log out
localhost:8080/CloudArmor/activate.jsp?86cc7

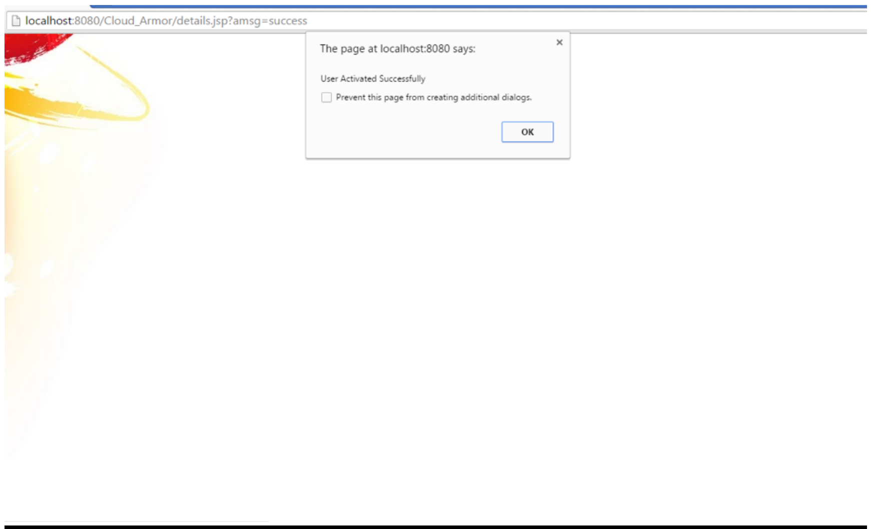
User Details

User ID	Name	Email	Birth Day	Location	Contact No	IP Address	Signup Time	Status	Action	Action
86cc7	Karthik	punyamurthula.karthik@gmail.com	1986-10-08	Canada	2248579108	DESKTOP-UV22MP8/192.168.0.5	2017/10/29 16:50:10	No	Activate	Deactivate
8bf15	Unmesha	upunyamurthula@stcloudstate.edu	2017-10-04		1234567890	DESKTOP-UV22MP8/192.168.0.5	2017/10/28 01:08:59	Yes	Activate	Deactivate



User Details

User ID	Name	Email	Birth Day	Location	Contact No	IP Address	Signup Time	Status	Action	Action
86cc7	Karthik	punyamurthula.karthik@gmail.com	1986-10-08	Canada	2248579108	DESKTOP-UV22MP8/192.168.0.5	2017/10/29 16:50:10	Yes	Activate	Deactivate
8bf15	Unmesha	upunyamurthula@stcloudstate.edu	2017-10-04		1234567890	DESKTOP-UV22MP8/192.168.0.5	2017/10/28 01:08:59	Yes	Activate	Deactivate

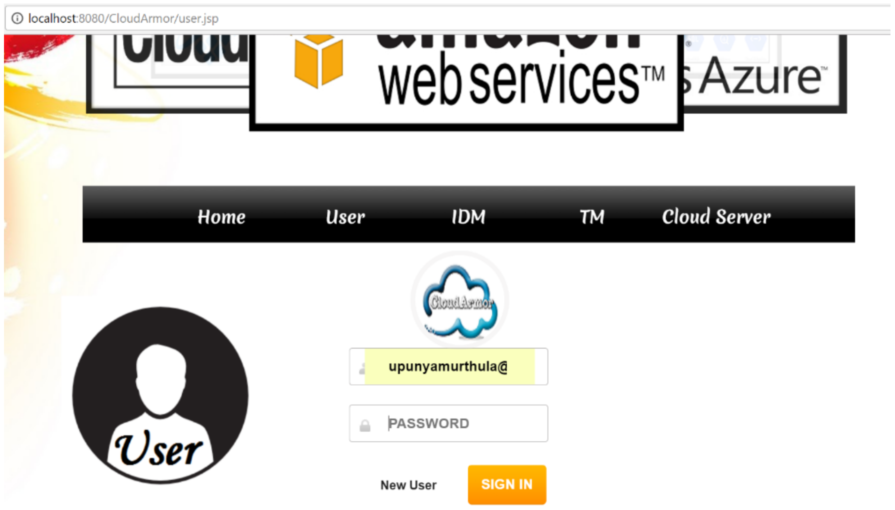


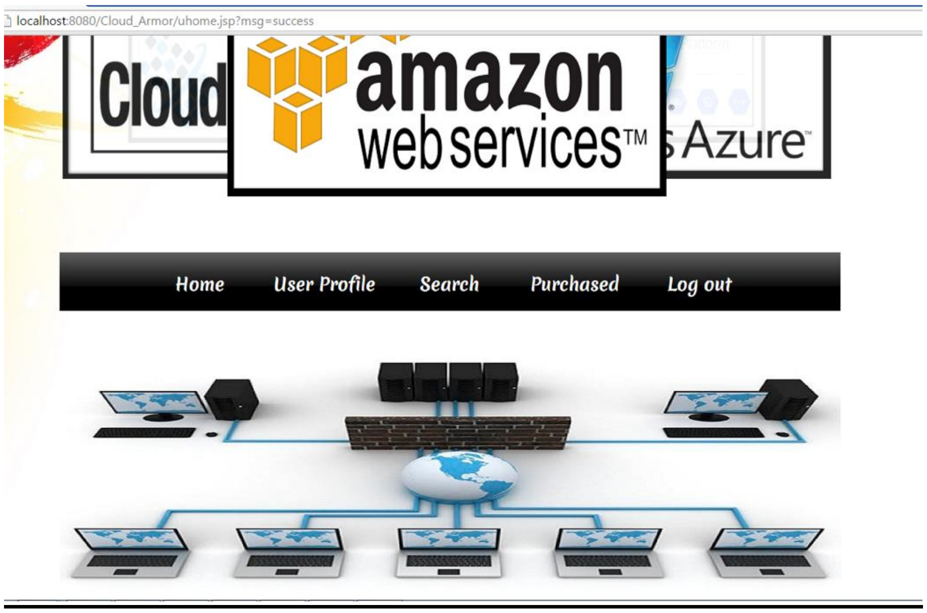
The page at localhost:8080 says:

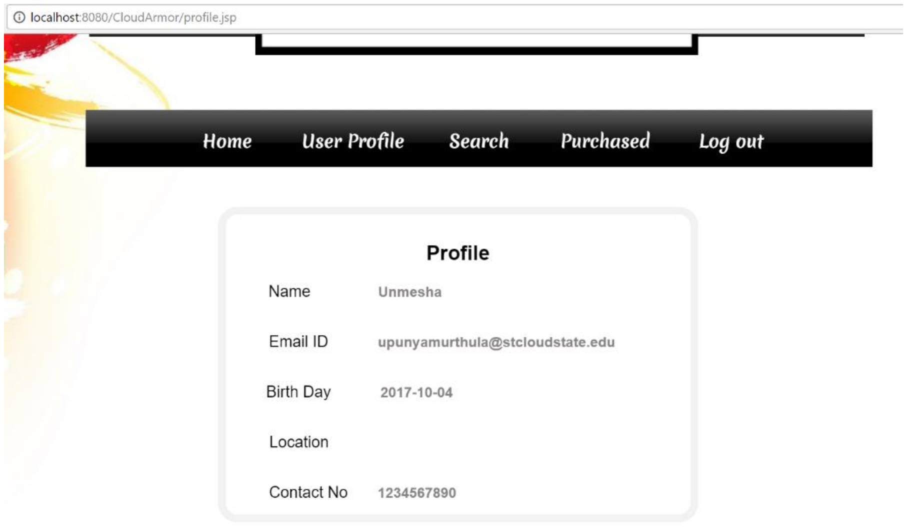
User Activated Successfully

☐ Prevent this page from creating additional dialogs.

User Login

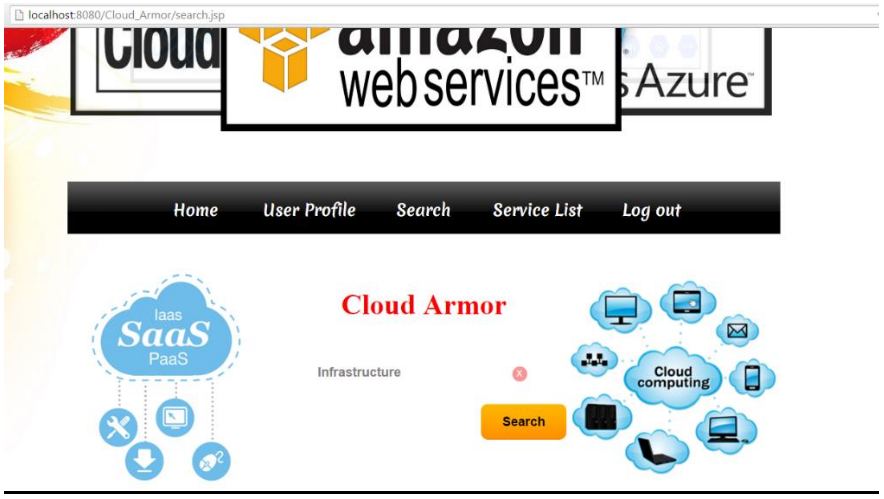

localhost:8080/CloudArmor/user.jsp
web services™
Azure
Home
User
IDM
TM
Cloud Server
CloudArmor
upunyamurthula@
PASSWORD
User
New User
SIGN IN

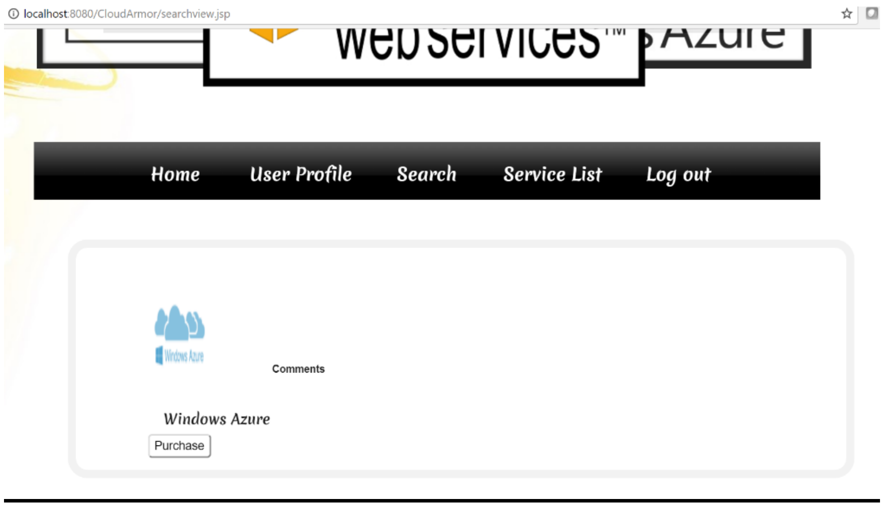

localhost:8080/Cloud_Armor/uhome.jsp?msg=success
Cloud
amazon web services™
Azure
Home
User Profile
Search
Purchased
Log out

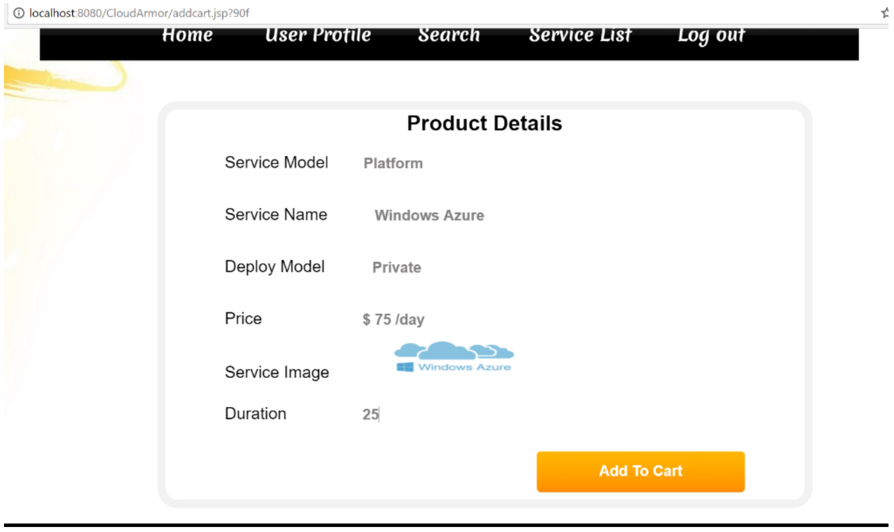


Profile

Name	Unmesha
Email ID	upunyamurthula@stcloudstate.edu
Birth Day	2017-10-04
Location	
Contact No	1234567890

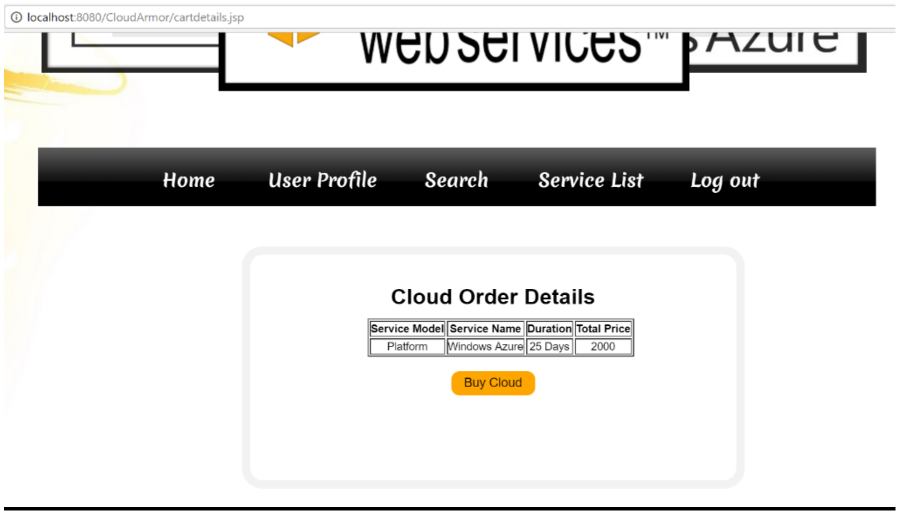






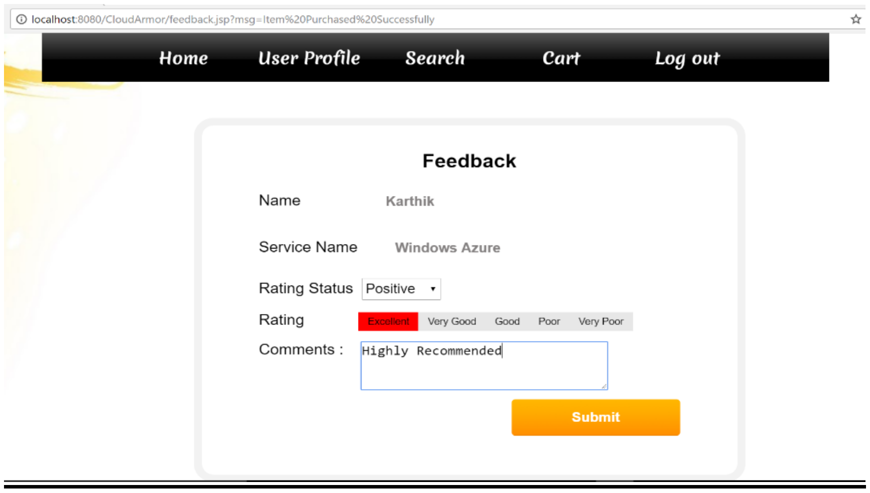
Product Details

Service Model	Platform
Service Name	Windows Azure
Deploy Model	Private
Price	$ 75 /day
Service Image	
Duration	25


localhost:8080/CloudArmor/cartdetails.jsp

Cloud Order Details

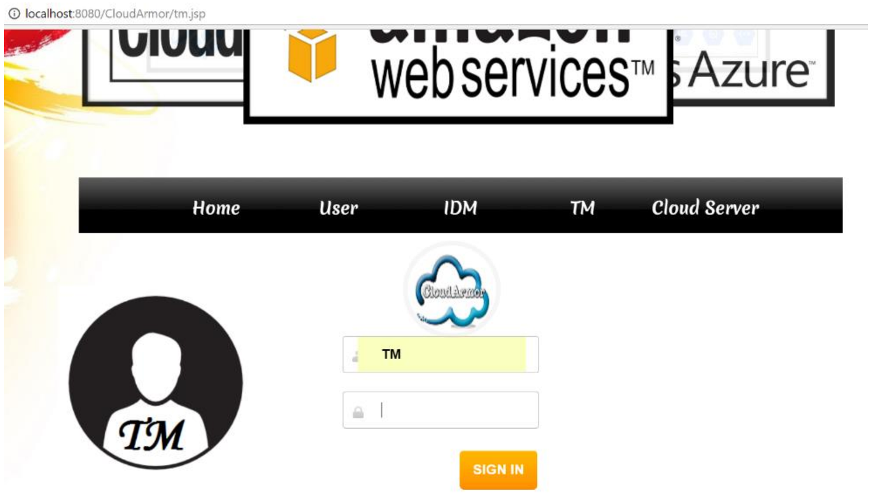
Service Model	Service Name	Duration	Total Price
Platform	Windows Azure	25 Days	2000

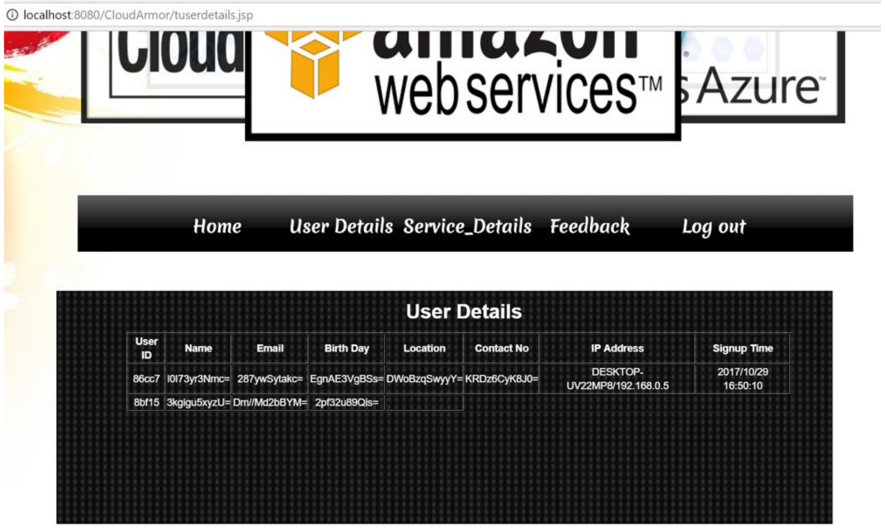

localhost:8080/CloudArmor/feedback.jsp?msg=Item%20Purchased%20Successfully

Feedback

Name	Karthik
Service Name	Windows Azure
Rating Status	Positive
Rating	Excellent
Comments :	Highly Recommended

TM





User Details

User ID	Name	Email	Birth Day	Location	Contact No	IP Address	Signup Time
86cc7	I0I73yr3Nmc=	287ywSytakc=	EgnAE3VgBSs=	DWoBzqSwyyY=	KRDz6CyK8J0=	DESKTOP-UV22MP8/192.168.0.5	2017/10/29 16:50:10
8bf15	3kgigu5xyzU=	Dm//Md2bBYM=	2pf32u89Qis=				