Spring 2018

Security risk assessment in cloud computing domains

Amartya Sen

Department: Computer Science

SECURITY RISK ASSESSMENT IN CLOUD COMPUTING DOMAINS

by

AMARTYA SEN

A DISSERTATION

Presented to the Graduate Faculty of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

2018

Approved by

Dr. Sanjay Madria, Advisor
Dr. Sajal Das
Dr. Wei Jiang
Dr. Simone Silvestri
Dr. Jagannathan Sarangapani

PUBLICATION DISSERTATION OPTION

This dissertation consists of the following six articles which have been submitted for publication as follows:

Paper I: Pages 31 to 67 was published in IEEE Transactions on Services Computing, 2017.

Paper II: Pages 68 to 83 was published in IEEE Cloud Computing, 2015.

Paper III: Pages 84 to 124 was submitted to IEEE Transactions on Services Computing, 2018.

Paper IV: Pages 125 to 153 was published in Proceedings of the 19th International Conference on Distributed Computing and Networking (ICDCN), 2018.

Paper V: Pages 154 to 191 was submitted to IEEE Transactions on Services Computing, 2018.

Paper VI: Pages 192 to 220 was submitted to the 37th IEEE International Symposium on Reliable Distributed Systems (SRDS), 2018.

ABSTRACT

Cyber security is one of the primary concerns persistent across any computing platform. While addressing the apprehensions about security risks, an infinite amount of resources cannot be invested in mitigation measures since organizations operate under budgetary constraints. Therefore the task of performing security risk assessment is imperative to designing optimal mitigation measures, as it provides insight about the strengths and weaknesses of different assets affiliated to a computing platform.

The objective of the research presented in this dissertation is to improve upon existing risk assessment frameworks and guidelines associated to different key assets of Cloud computing domains - infrastructure, applications, and users. The dissertation presents various informal approaches of performing security risk assessment which will help to identify the security risks confronted by the aforementioned assets, and utilize the results to carry out the required cost-benefit tradeoff analyses. This will be beneficial to organizations by aiding them in better comprehending the security risks their assets are exposed to and thereafter secure them by designing cost-optimal mitigation measures.

ACKNOWLEDGMENTS

First and foremost, I would like to acknowledge the support and guidance of my advisor, Dr. Sanjay Madria. I sincerely appreciate his faith in recruiting me, and being encouraging and patient during the tough times. His ample experience and knowledge will keep coming handy for me as I continue my academic career.

I would like to extend my humble gratitude towards the members of my dissertation committee, Drs. Das, Jiang, Silvestri, and Sarangapani, for their constructive feedback and advice in and beyond the context of this dissertation. I would also like to acknowledge the support of the Department of Computer Science staff and other faculty members during the course of my degree program.

Further, I would also like to acknowledge the help and contributions from the members of my research group, past and present. Special thanks to my friends, Dr. Kenneth Fletcher, Venkata Prashant Modekurthy, Dr. Roy Cabaniss, Shashank Kumar, Dr. Vimal Kumar, and Doug McGeehan, for their invaluable support during different stages of my PhD journey, both professionally and personally.

Finally, my deepest gratitude and respect to my parents, Dipika and Amit Sen for their unconditional love and support. They have been a part of the successes and hardships of this journey as much as I have, and I dedicate this dissertation to them.

TABLE OF CONTENTS

Page

LIST OF ILLUSTRATIONS

LIST OF TABLES

LIST OF ALGORITHMS

SECTION

1. INTRODUCTION

Cybersecurity is one of the primary concerns in any computing platform. Given the interconnectivity of billions of devices and their networks, the risks of attacks by unscrupulous minds have also been continuously going up. Such exploitations of any network end up with a huge adverse/devastating effect. This is evident in the fact that Cyber crime damage costs is estimated to reach $6 trillion annually by 2021 [1] which will prompt a cybersecurity spending increase of more than $1 trillion from 2017 to 2021[2]. This trend will only intensify with the notable rise in paradigms like Internet of Things, industrial IoT, vehicular security, and so forth. However, in terms of addressing cybersecurity incidents it is not sufficient to merely identify and analyze attacks and implement numerous security measures with the hopes that a network will remain secure. This is because, from a socio-economic perspective security is a non-functional requirement and given the budgetary constraints of organizations or clients, they are skeptical to invest in security measures.

Nonetheless, even in the absence of budgetary constraints it is not effective to simply implement numerous random security measures. Since, without an understanding of the risks associated to the platforms which hosts a client's or organization's application, developing and implementing security measures are futile leading to inefficient utilization of limited security resources. Therefore, addressing Cyber security threats by formulating efficient and effective security measures is a multi-step process which requires performing security requirements analysis, risk assessment, risk management and mitigation. These processes will also help clients understand the benefits of their investment in Cyber security.

[1] goo.gl/Ph76dh
[2] goo.gl/j7VLMF

Risk assessment lies in the crux of the multi-step process, enabling the incorporation of organization-wide assessment to determine the security threats, vulnerabilities and their impact on network security parameters like confidentiality, integrity, and availability. It also helps in prioritizing the risks that needs to be addressed thereby playing an important role aiding the allocation of security measures constrained by an organization's budget. However, performing idealistic risk assessments for a large scale organization can still be very expensive and may produce results which might require considerable domain knowledge in order to comprehend and generate useful insights. Therefore, the task of risk assessment is not trivial and requires careful consideration.

1.1. RISK ASSESSMENT METHODOLOGIES

The process of security risk assessment is guided by identifying security requirements and thereafter performing extensive evaluations of an organization's Cyber security prospects. These tasks involve assessing the network infrastructure for security threats due to vulnerabilities that are present and the attacks that can exploit them. Risk assessment also enables the evaluation of another important organization asset - the people. Users of an application and an organization's employees are also susceptible to different threats which can result in the Cyber security incidents. These threats cannot be detected by running a vulnerability scanner tool on the network, further necessitating the need of performing risk assessment. At the end, risk assessment provides a detailed report on the threats confronted by an organization in different domains (network, people, security policies, etc.) which can be used in the risk management and mitigation process to either patch security vulnerabilities and prevent certain attacks or to pro-actively develop prioritized security incident response plans to minimize the effect of an exploit. Therefore, risk assessment acts as an estimation and validation tool for an organization by shedding light into security return on

investment, get feedback on implemented security measures, and help to understand their cohesion with respect to best security practices, compliance and standards as laid down by regulatory bodies like NIST [77] or ENISA [78].

1.1.1. Security Risk Assessment Process. The security risk assessment process is used to determine the strengths and weaknesses of an organization's systems, identifying and minimizing threats below a threshold which is acceptable as per the security requirements of the organization. An example of an organization's security requirements could be ensuring a certain level of assurance in confidentiality, integrity, and availability of their application and the data it processes. Risk assessment generally focuses on evaluating the likelihood of an undesired event (for e.g. a data breach, or an unauthorized access) and the impact it will have on the exploitation of the system and the organization as a whole. Once this is evaluated, risk mitigation measures are designed and developed to minimize the likelihood and impact of the risks. In broader terms, risk assessment can be used to identify risks in different areas of an organization and not just related to Cyber security. For example, traditional risk assessment in the domain of information security and IT security can be used to evaluate systems and applications that support the functional services of an organization, its network and servers, physical security of the devices and premise, risks present (due) to employees of the organization. The process of risk assessment is normally (and must be) utilized during the conception of an IT service. Other than this, addition of new functionalities to an application/service; changes in the networking environment; change in technology (Software or Hardware updates) should also prompt the utilization of the risk assessment process.

Prior to performing any risk assessment process, organizations must follow through with the creation of risk assessment policies which outlines a blueprint to guide the risk assessment process to be carried out by an organization. Such a blueprint consists of guidelines establishing factors such as:

- When does an organization need to perform risk assessment

- How often should it be repeated
- What will be the scope of risk assessment (how comprehensive it is going to be)
- Who will be in charge of carrying out the task (internal or third party)
- What is expected from it (actionable insights)
- Prioritizing risk levels (what sort of risks will be acceptable and which ones will be deemed critical)
- What methods will be used to perform risk assessment (qualitative, quantitative or hybrid)

The risk assessment process is an exhaustive task and is initiated with the risk assessment policies document. It helps in defining the scope of the task and appointing personnels who will be responsible to carrying it out. Thereafter, risk assessment procedures are chosen (or developed) and a list of threats are identified which is followed up with identifying vulnerabilities. In this context, a *threat* is defined as an unwanted event that may cause harm; *Vulnerability* is defined as a weakness which may provide a way for a threat to materialize; *Impact* is defined as the consequence(s) of a threat that has materialized. After vulnerability identification, security measures are determined and evaluated which might either help to mitigate, transfer (getting insurance policies), or avoid the threat and its impact. Thereafter, all the aforementioned information are used to estimate probability values which will depict the likelihood of occurrence of the threat in the presence of evaluated security measures. This is generally done either by using experts with domain knowledge, historical logs of threats, or statistical analysis. For prioritizing, the estimated probability can be further categorized as *Very likely*, *Isolated incidents*, *Rare*, *Very unlikely*, and *Almost impossible*. These categorizations are subjective and typically depends on the organization and their risk assessment policies outline and followed up with sensitivity analyses like Monte-Carlo analyses.

Once probabilities are estimated, the damage is quantified in terms of the impact of the identified threats when they exploit the vulnerabilities. This followed up with risk level estimations which is defined by threat multiplied by its likelihood and impact probabilities. Although, organizations like NIST have a scale for categorizing risk levels (for e.g. 1.0: High, 0.5: Medium, 0.1: Low), it is a challenging task to be able to quantify and categorize risk levels based on monetary loss or loss of reputation. After risk level estimations, security measures are re-evaluated and suggested for implementation which are presented in the reports generated from the risk assessment process.

For traditional risk assessment many tools are available [13] [25] however any tool developed for carrying out risk assessment should have the some of the following features:

- Structured report generation to show the risk probabilities and their impacts
- Questionnaires and checklist to assess concerns related to compliance, policies, and best practices
- List of threats and Security measures that can be used to suppress them
- Software automation

1.1.2. Qualitative vs. Quantitative Risk Assessment. Risk assessment involves the comprehensive identification of different threats an organization will face and evaluate their likelihood and impact. There are two primary methods of doing this: qualitative and quantitative. Qualitative risk assessment involves subjectively evaluating the identified risk's impact or likelihood using metrics like High, Medium, or Low. The categorization of the identified risks is usually done based on organizational policies and their understanding of decisions like what constitutes of a high impact risk on the organization. This process is simplistic in nature and allows for organizations to develop a ball-park estimation of the overall threats faced by an organization.

Quantitative risk assessment in contrast involves numerical estimations and mathematical equations. This is done by using estimation techniques like Bayesian networks or Monte-Carlo analysis to determine the probability of success for the likelihood or impact of different threats. Although, an inherent challenge in this technique is to able to accurately obtain the initial probability or risk values of the assets which are exposed to the identified threats. However, tools and techniques are available like Common Vulnerability Scoring System (CVSS) [17] which can be used a foundation to customize the empirical estimation according to the needs of an organization. In doing so, quantitative risk assessment will allow for quantification based on different categories like inherent difficulty of exploiting the threat, priority given to different network security parameters, impact of initial security measures and so on. Nonetheless the goal of this method is to develop a framework that can lead to some actionable insights and not just provide an understanding of the threat level of the organization. The actionable insights may involve computing the security return on investment by performing cost-benefit tradeoff analysis, or evaluating the effectiveness of different security measures.

However, performing accurate and comprehensive quantitative risk assessment is ideally complex and costly, and therefore cannot be universally incorporated by small or medium scale organizations. A more pragmatic approach lies in using hybrid risk assessment methods that involves qualitative and quantitative techniques. Such a hybrid approach will involve initially estimating and prioritizing identified threats using qualitative assessment measures and then apply quantitative measures to design and develop actionable insights.

1.2. RISK ASSESSMENT IN CLOUD COMPUTING DOMAINS

The traditional IT services were hosted on organization's private networks. This bounded the extent of the risks that their applications were exposed to like traditional threats due to poor router controls, poorly configured firewall policies, and web browser security

concerns like phising, script injection and so forth. All the vulnerability categories like personnels, system hosting facilities, network communications, and software vulnerabilities were within the trust boundaries of the organization. Although in terms of Cyber security this gave the organizations more control over their systems, they had to own and maintain all the commodities and facilities required to host their application. This prospect was quite expensive, especially when there was a seasonal need to scale up resulting in hardware investments and facilities upgrade which might remain underutilized after the demands go down.

The Cloud computing paradigm was designed to address this challenge of having to own and maintain your own hardware and facilities wherein it pooled computing resources and provided it to clients based on a pay-as-you-use model. This was achieved through three different delivery models - Infrastructure-as-a-service (IaaS), Platform-as-a-service (PaaS), and Software-as-a-service (SaaS). IaaS rents out computing infrastructures on which clients can deploy and develop their applications, for e.g. Amazon Web Services[3]. In PaaS development platforms are rented which maybe required to develop certain applications, for e.g. Apprenda[4]. SaaS delivers ready made applications to clients which have been developed by cloud service providers, for e.g. Google Apps[5]. Even though such a paradigm was cost efficient, it resulted in an increase in security concerns since applications, data, physical facilities were no longer within the trusted boundaries of the client and their organization. This further made the aspects of risk assessment challenging because it was not straightforward for the clients to evaluate the cloud service provider's network, personnels, or physical facilities. Additionally, cloud computing introduced new paradigms like multi-tenancy and resource sharing which required risk assessment methodologies to evolve in order to adapt with the changes.

[3]aws.amazon.com/

[4]apprenda.com

[5]gsuite.google.com/products/

Along this direction, the scope of our dissertation is captured through the scenario summarized in Figure 1.1.

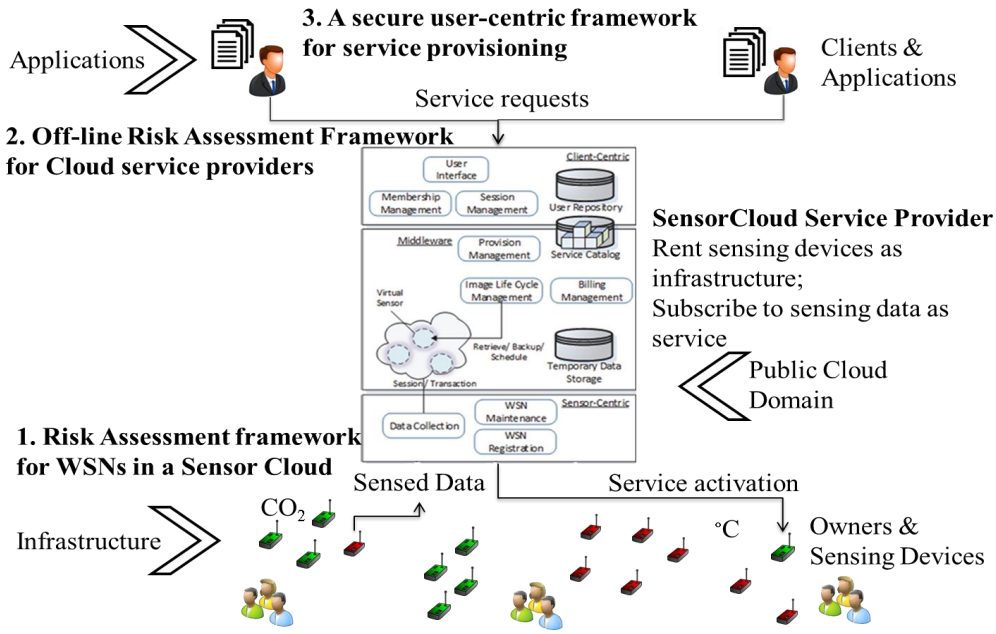

Figure 1.1. Key Assets related to the Sensor Cloud Computing Domain

In this context, there can be two set of users:

- owners - who possess infrastructure consisting of various sensing devices like wireless sensor networks, cellphones, wearable body sensors, etc. These sensory devices are capable of sensing environmental phenomena like temperature, humidity, atmospheric pressure, user's physiological attributes (heart rate, skin temperature), and so forth.

- clients - who possess different applications which require some kind of sensing attributes for their functionality for example applications related to the domain of remote health care, smart assisted living, industry, and so forth.

Traditionally, owners can use their infrastructure to run or support their applications in a dedicated fashion. Ideally, sensory infrastructure may be operational all the time, however this is seldom done in practical scenarios. This is because, applications requiring sensing capabilities utilize the devices based on certain sensing frequency and duration, which in turn might leave the infrastructure underutilized. Whereas, clients may have the infrastructure to host their applications, however they might not have the capabilities to own or maintain their sensory infrastructure. Therefore, to connect these two set of users - owners and clients, the Missouri S&T Sensor Cloud platform was designed and implemented [79]. Sensor Cloud facilitates infrastructure-as-a-service by allowing different owners to register their wireless sensor networks (WSNs) with its middleware. Further, clients can subscribe to the data originating from the Sensor Cloud infrastructure as a service thereby facilitating the concept of sensing-as-a-service, similar to software-as-a-service in the traditional cloud computing domains. In addition to renting the sensing infrastructure and subscribing to the data generated by them, clients may also opt to migrate their applications from their private networks to the middleware of a Sensor Cloud service provider hosted on a public cloud domain.

In such a scenario, the three key assets that will be vulnerable to different security exploits are the - infrastructure, applications, and clients. Therefore, in order to develop optimal security measures to protect these assets it is necessary to understand their strengths and weaknesses, thereby necessitating the need to perform security risk assessments. This is addressed via. this dissertation in the form of three security risk assessment frameworks (Figure 1.1):

1. Risk Assessment framework for wireless sensor networks in a Sensor Cloud.

2. Off-line risk assessment framework for cloud service providers.

3. A secure user-centric framework for service provisioning.

In the following sections, we discuss the challenges that necessitates the need for more efficient risk assessment frameworks with respect to the aforementioned assets.

1.2.1. Risk Assessment in Sensor Cloud. A Sensor Cloud framework consists of a collection of wireless sensor networks (WSNs) which is used to provide sensing-as-a-service to users via the cloud platform. These WSNs have different ownership entities and maybe heterogeneous in nature. Therefore, different WSNs may have different routing and security protocols, the way in which they connect with the cloud platform might also vary. In addition to understanding the standalone risks that are present in these WSNs, one must also be able to understand the logical co-relation between stand alone risks. Successful execution of an attack may leave a network vulnerable to further degenerate attacks. For example in a WSN, execution of a malware attack propagated from the cloud platform may lead to node subversion, which can be further exploited to execute Sinkhole or Sybil attack. Current risk assessment frameworks for WSNs do not take into account such kind of logical co-relationship between attacks. This requires the design of a risk assessment framework for these WSNs in a Sensor Cloud that takes into consideration such logical co-relation between the feasible set of attacks on a WSN. This can be done by using techniques such as attack graphs to depict cause-consequence relationship between the attacks and model it as Bayesian networks for quantification purposes. A quantitative risk assessment approach will help in determining the net threat level to WSN's security parameters like confidentiality, integrity and availability and in turn efficiently allocate security resources as well as schedule maintenance activities to check the largely unattended WSNs in a Sensor Cloud.

Attack graphs has been extensively studied in the literature for wired networks [80] [81] [82]. However, it cannot be applied as is to perform risk assessment for WSNs in a Sensor Cloud. This is because when an attack graph for a wired network is generated, the network is scanned using a vulnerability scanner tool such as Nessus [13]. These scanners

detect the list of vulnerabilities present on each system in a wired network which is then parsed into an attack graph generating tool [83]. However, no comparable vulnerability scanning tool currently exists for WSNs.

Even if such vulnerability scanning tool existed for WSNs, generating a list of vulnerabilities would be insufficient for the purpose of performing risk assessment. Since, sensor nodes in WSNs collaborate to achieve a common goal and suffer from inherent resource limitations which is the primary cause of WSNs vulnerabilities as they do not allow the implementation of desired security measures to safeguard the network. Hence the vulnerability list will be identical for all nodes and no concrete conclusions could be formulated from it. Thus, rather than focusing on vulnerabilities in a sensor node or network, an alternative will be to perform risk assessment by considering the feasibility of attacks on a particular WSN in a sensor cloud. The successful execution of different attacks will vary according to security measures used, tasks being carried out, and deployed environment of a WSN. Due to the above factors, risk assessment in WSNs under a sensor cloud is challenging task. However, these existing works can be adopted and applied to this domain which will help understand the cause-consequence relationship between WSN attacks and identify ways in which a particular WSN security parameter may be exploited.

1.2.2. Risk Assessment in Traditional Cloud Computing Domains. Traditional Cloud computing is a viable solution for applications which depend on scalability and reliability for an uninterrupted service. Cloud services in the form of Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS) and Infrastructure-as-a-Service (IaaS) cuts down application maintenance and development costs dramatically. However migration of applications to the cloud platform gives rise to security issues as applications are no longer within the secure domains of its organization. The security threats present varies from one application to another and as such the promise of generalized security measures provided by cloud service providers (CSPs) may not be sufficient to alleviate the security concerns. Hence, the security solutions provided by a CSP is like a big black box to its clients.

Currently the means to assess security of a CSP is by going through their security white papers, SLA agreement, or tendor notes. An organization can also employ the services of a third party to do the same.

However, frequent external security audits on a cloud service provider may expose the infrastructure and security details and if acted upon maliciously may result in adverse effects. Alternatively, organizations can internally undertake the risk assessment procedure by referring to guidelines established by the authorities like ENISA [78] or CSA [84] to assess the security provided by a cloud service provider. Additionally, standards such as PCI-DSS [85], HIPAA [86] lays down guidelines for cloud service providers hosting application related to specific fields like health care, applications comprising of financial transactions. However, accuracy of internal security audits requires considerable domain expertise. Further, risk assessment results either performed externally or internally, will also be impacted by the keenness of the evaluator. Additionally, the cloud platform is quite dynamic and security and/or service policies are subject to change on a short notice. Thus, personnels responsible for performing risk assessment need to keep themselves up-to-date with these changes which might not be pragmatic in the presence of multiple possible cloud service providers to host an organization's application. In addition to these challenges, risk assessment evaluations for cloud service providers need to account for the security requirements of an application to be migrated to the cloud platform. A security measure present in a cloud service provider may be applicable to one application, but it may not be sufficient for another application.

A more effective way to assess a cloud service provider's security will be to do it in contrast to the security threats present in an application. To do so, one must be able to identify the vulnerabilities that are present in an application and determine if a cloud service provider addresses it using their available security measures. One way of identifying application vulnerabilities is to use vulnerability scanner tools like Tenable [13], but this can be done once the application has been fully conceived and is operational. As such any

repairs/patching at this point could be an expensive task. Therefore, security vulnerabilities in an application should ideally be evaluated during the system design phase. In this regard, tools like EMC's Developer Driven Threat Modeling [26] and Microsoft's STRIDE [22] are present. Application threat modeling as discussed by Open Web Application Security Project (OWASP) [24] gives useful guidance in vulnerability detection and secure application development. It helps to identify, quantify and address the security risks associated with an application. Although these methods are quite suitable in terms of identifying vulnerabilities in an application, they are not sufficient. Since these applications will eventually be considered for cloud migration, design phase application risk assessment cannot be simply performed using traditional methods which consider applications to be hosted in an organization's private in-house hosting facilities. For example, if a client application uses a malicious virtual machine instance from the cloud repository it might lead to undesirable outcomes like data breach or unauthorized access. Therefore, application risk assessment methodology needs to adapt by including cloud security domains like encryption and key management, infrastructure and virtualization security, data center security and so on [87].

Additionally, security measures, cloud platform infrastructure and safeguards differs from one cloud service provider to another. The techniques used by a cloud service provider to address and mitigate threats on their cloud platform changes with time and without prior knowledge of consumers. If an application is hosted by a federation of cloud service providers, there could be every possibility of certain incompatibility issues amongst the different cloud service providers. Hence, the presence or absence of security solutions on the cloud platform is no longer sufficient to assess the security of an application on the cloud.

1.2.3. A Secure User-centric Framework for Service Provisioning. The services of the sensing infrastructure subscribed by the clients and their applications is facilitated by encapsulating their way of life with the *Big* data sensed by surrounding wireless devices. In

this context, service provisioning will involve developing frameworks which will identify a subset of wireless devices from the available infrastructure, tasked to serve different users based on their service requests. Generally, such frameworks consist of two primary layers - perception layer, comprising of sensory devices, and management layer like the Cloud or Edge platforms to process the sensed data.

Existing frameworks extend these two layers according to their client's or application's need, in different domains like industry [88], or health care [89]. Therein, they address specific issues either arising due to resource constrained wireless devices or interoperability challenges as a result of device heterogeneity [90], or discuss about making the management layer more efficient [91]. Although novel in their own regards, these frameworks lack the inclusion of user-centric behavior to dispense services accounting for a user's variable Quality of Service (QoS) and security preferences. The aspect of addressing user QoS during service discovery and selection has been presented in [92]. The primary focus there is to outline the process of dynamic identification of wireless sensory devices for service selection. However, they do not consider the satisfaction or variability of a user's QoS/seurity preferences.

In addition to QoS, users' security requirements from the participating devices serving their application, plays an important role. However, addressing security in this domain is a challenging task since the wireless devices are resource constrained (mainly in terms of memory, energy, and bandwidth) and thus, cannot support the well established resource intensive security protocols. Rullo et al. [93] address this challenge by leveraging concepts of optimization and game theory. However, the allocation of security resources is done from the infrastructure's perspective and do not include users in the loop. For the task of service provisioning composed of sensory wireless devices, the aspect of security cannot be treated as a constant parameter. This is because security attacks on a network (e.g denial of service) has an impact on the QoS parameters (e.g. response time). Therefore, similar

to network QoS, a user may have variable security preferences which must be taken into consideration, in addition to their QoS and functional preferences (e.g. region of interest, service duration) during service provisioning.

To address this requirement, a service provisioning framework such as the Sensor Cloud should be modeled in a secure user-centric fashion allowing its clients to specify their functional, variable QoS, and security preferences. This paradigm will include clients/users in the service provisioning loop instead of solely relying on the innate networking and security capabilities of the middleware/infrastructure. In turn, this will alleviate some of users' apprehension about the security provided by their subscribed services, thereby improving the overall user experience; something that existing frameworks do not address.

1.3. DISSERTATION OBJECTIVES

In the following sections, the outlines of the research objectives and contributions are discussed for the proposed security risk assessment frameworks designed for the aforementioned assets of a Sensor Cloud computing platform as shown in Figure 1.1.

1.3.1. Risk Assessment in a Sensor Cloud Using Attack Graphs. The primary objective in this research domain is to design and develop a risk assessment framework for wireless sensor networks (WSNs) in a Sensor Cloud environment by adapting the techniques of attack graphs from the domain of wired networks. This will help in elucidating the logical relationships between the feasible set of attacks that can exploit WSNs. Further, these attack graphs will be modeled using the concepts of Bayesian networks in order to apply to quantitative risk assessment such that we are able to compute the *net threat level* to different wireless sensor networks security parameters like confidentiality, integrity, availability. These results will be used along with continuous-time Markov modeling such that we are able to estimate different time frames estimating the probable degradation of the aforementioned network security parameters and scheduled maintenance activities for the largely unattended wireless sensor networks.

We outline our contributions to the proposed risk assessment framework in a Sensor Cloud as follows:

1. Design and develop attacks graphs for wireless sensor networks in a Sensor Cloud to depict the logical relationship between attacks on a wireless sensor network.
    - Develop attack patterns and attack module database for attacks in the domain of wireless sensor networks.
    - Create attack graphs for each network security parameter (confidentiality, integrity, and availability) by using the attack module.
2. Depict attack graphs as a Bayesian network quantifying the likelihood and impact of attacks on a wireless sensor network and the net threat level to network security parameters.
    - Design and develop severity ratings for attacks captured in its likelihood and impact estimations by adopting the guidelines established by Common Vulnerability Scoring System (CVSS) [17].
    - Compute probability of success of an attack using its frequency and impact estimations in order to evaluate the net threat level to network security parameters of confidentiality, integrity, and availability.
3. Compute time frames predicting the degradation of network security parameters by modeling the risk level estimations as a continuous-time Markov process.
    - Utilize impact estimations of attacks to design and develop service levels depicting the degradation level of a wireless sensor network's security parameter.
    - Utilize likelihood estimations of attacks to compute the transition probability from one service level to another.
    - Evaluate time frames to schedule maintenance by using the transition probability of the most degraded service level.

1.3.2. Offline Risk Assessment of Cloud Service Providers. The primary objective in this research domain is to develop a risk assessment framework which is able to assess the security provided by different cloud service providers by evaluating the security requirements of an application which will be migrated to their platform. To do so in a cost efficient fashion, we develop the risk assessment framework such that it can be applied during the design phase of the software development lifecycle (SDLC) of the application. The application risk assessment during design phase also incorporates various cloud security domains which was identified by analyzing Cloud Security Alliance's Security, Trust, & Assurance Registry (STAR) [87]. These evaluations produced the risk level estimations of an organization's application (client's security requirements) which will be migrated to the cloud platform of cloud service providers. The security provided by a cloud service provider was measured by contrasting their security policies gathered from publicly available documents with the client's security requirements. Finally, a cost-benefit tradeoff analysis was performed, designed as multi-objective optimization problem, in order to develop optimal cloud migration plans with a balance between security achieved and cost incurred, accounting for constraints arising due to factors such as budget, technical and legal issues, and legacy systems.

We outline our contributions to the proposed risk assessment framework for cloud service providers as follows:

1. Design and develop mission oriented risk assessment methodology for design phase application risk assessment.
    - Perform project assessment through system design represented using data flow diagrams.
    - Analyze system design for security threats arising due to traditional web application paradigm.

- Design and develop risk assessment methods to increase the client's certainty about cloud service provider's security policies by introducing cloud security metrics during an application's design phase risk assessment.

2. Design and develop cloud service provider risk assessment methodology in order to evaluate the security provided by various cloud vendors with respect to the client's security requirements.

    - Aggregate security measures provided by various cloud vendors through publicly available documents and other third party reviews, especially from Cloud Security Alliance's (CSA) Security, Trust, & Assurance Registry (STAR).
    - Perform descriptive and exploratory analysis of CSA STAR to better understand the context of answers from different cloud service provider for different cloud security domains.
    - Improve the STAR registry structure and develop a comprehensive knowledge base incorporating the aggregated security evaluations from STAR with the proposed risk assessment framework.

3. Design and develop methodologies to generate optimal cloud migration strategies.

    - Perform cost-benefit tradeoff analysis by modeling it as a multi-objective optimization problem.
    - Formulate different cloud migration scenarios accounting for hybrid cloud migration in a federated and non-federated cloud hosting.
    - Performance evaluation of different evolutionary algorithms to model our proposed cost-benefit tradeoff analysis.

1.3.3. A Secure User-centric Framework for Service Provisioning. The primary objective in this research is to develop a user-centric service provisioning framework for the clients/users of a Sensor Cloud service provider, designed to achieve variable security, QoS preferences and evaluate users' satisfaction, addressing the following challenges:

- incorporate user security preferences during service provisioning,
- allow users to linguistically express their preferences on different objectives of service provisioning, and
- evaluate a user's subjective satisfaction feedback from the provisioned service by analyzing the quality of experience information gathered from user's physiological data.

In doing so, the contributions to this research are as follows:

1. Introduce the concept of variable security composition paradigm to capture a user's variable preferences on their desired security protocols.
2. Model service provisioning by integrating concepts of multi-objective optimization and user elastic preference specification.
3. Design a multilayer perceptron to accurately validate users subjective satisfaction feedback by incorporating physiological sensor data and deep learning emotion classification methodologies.

2. LITERATURE REVIEW

The scope of this dissertation deals with three primary related work fields: (1) Risk assessment in wireless sensor networks (WSNs), (2) Risk assessment of a traditional web application, and (3) Risk assessment of Cloud computing platforms. In the first field, we discuss the various attacks that can be executed on WSNs, the different ways in which attack graphs are designed and developed for wired networks, the concept computing probabilistic likelihood and impact of attacks using Common Vulnerability Scoring System (CVSS), and representing attack graphs using Bayesian networks. In the second field, we outline the different methods which are used to perform application risk assessment during its design phase. Finally, we discuss some of the existing frameworks for carrying out risk assessment of Cloud computing platforms as well as some of the works that address Cloud migration techniques.

2.1. RISK ASSESSMENT IN WIRELESS SENSOR NETWORKS

The scope of attacks sustained by a WSN has been surveyed and discussed in [1] [2] [3]. The authors have assessed well-known sets of WSN attacks along with their countermeasures. They discuss various security issues related to the fields of secure data transfer, denial of service attacks and its counter measures by using techniques such spread spectrum frequency hopping. They also outline the limitations that sensor nodes have due to their inherent resource constraints resulting in limited energy and processing power which makes the application of traditional security measures like public key cryptography a very challenging, near impossible task. This gives rise to the need for efficient and secure techniques to manage key distribution. Additionally, WSNs also requires secure routing protocols enabling authentication of the messages that are broadcast between the sensor nodes and that to the base station.

Finally, challenges also exist in performing secure data aggregation which is essential to preserve the limited energy of sensor nodes. These works also give directions into the various threat models for WSNs, considering the aspects of insider and outsider attacks; active and passive attacks, helping to categorize the known attacks on WSNs like eavesdropping, jamming attacks, Sybil attack into these categories. They also present the different security requirements which is desired in a sensor network like secrecy, graceful degradation, efficient key management techniques, etc. However, these works did not discuss the inherent characteristics of an attack like how difficult is it to execute the attacks, how resourceful an attacker needs to be in order to execute the attack and so forth.

Walter [4] in his survey provides a more in-depth insight on security issues of a WSN establishing the parameters based on which security of a WSN is characterized. These parameters were confidentiality, integrity and availability. We are able to design the attack patterns and attack module from this information for our work and analyze the attacks on a broader perspective. Analysis of various attacks adopted by the adversary to exploit security parameters and ways in which they could be averted were also discussed in [4]. Although, it did not address the likelihood of exploitation of an attack. Wood [5] and Xu [6] gives exposition on the omnipresent denial of service (DoS) attack. DoS are not only hard to predict but also to counter. This helped us in understanding the nature of jamming attacks in WSNs. The absence of predictability and correlation with other attacks in case of DoS attack, is a drawback on the security administrator's part. Karlof [7], Kannhavong [8] and Newsome [9] gives an in depth analysis on routing layer attacks and the Sybil attack respectively. However, these attacks can be exploited by successful execution of attacks in different network layers. For this purpose we should identify the interdependencies between different feasible attacks. Mauw [10] and Phillips [11] demonstrated this kind of logical relationship via attack graphs or trees. Using the principles from the work of Lee [12], we were able to assess the risks to a network. But the drawback was that they were for a wired network scenario. Sheyner [14] discussed the various types of attack graph and models.

This contributed immensely towards the development of the attack graph model for WSN. In this context, the generation of traditional attack graphs whose nodes depict different wired network vulnerabilities, is a multi-step process in which the first step involves the generation of the attack patterns which outlines the definition of an attack/vulnerabilities and network security parameter that it intends to exploit/compromise. For traditional wired networks this can be identified by running vulnerability scanner tools like Nessus [13] which will output the list of vulnerabilities a particular system in the wired network is exposed to. This task is followed up with designing the data structure called the attack module which forms the basis for the generation of an attack graph. It consists of information related to an attack pattern such as pre and post conditions of the attack pattern. Pre-conditions illustrates the set of the network conditions/resources that need to be satisfied in order to successfully execute the attack or exploit the vulnerability. Whereas, post-condition are the set of network conditions/resources that becomes vulnerable due to the successful execution of an attack. Additionally, an attack module also holds the information about the join type of a particular attack pattern in the attack graph which can be one of two types - OR join, AND join. An OR join type in the attack graphs depicts that if any one of the child nodes of an attack pattern is successfully exploited, the parent node becomes exploitable. In contrast, an AND join illustrates that all of the child nodes of an attack pattern need to be successfully exploited in order to compromise the parent node.

Gallon [15] devised the methods to quantitatively assess the attack nodes in the attack graph; although they were not meant for a WSN. National vulnerability database [16] established the vectors to calculate the severity ratings of vulnerabilities in a wired network. Using the same principles, we calculated the severity ratings for the attacks on WSN. The CVSS metrics are used to quantitatively compute two specific characteristics of an attack/vulnerability: (1) misuse frequency (how likely is it to execute/exploit the attack/vulnerability), and (2) misuse impact (what will be the extent of the damage if the attack is executed or the vulnerability is exploited).

In order to compute misuse frequency, CVSS provides two core metrics known as *Base metric* and *Temporal metric*, whereas misuse impact is computed using Base metric and *Environmental metric*. Base metric is used to express the level of difficulty to exploit a vulnerability. Temporal and environmental metrics are used to express the effect of the network's deployment environment and surroundings in the exploitation of the vulnerability. This metrics are further made up of sub-categories like attack complexity, available tools and techniques for exploitation, security requirement, and so on. A detailed description of the CVSS scoring system is available in the CVSS scoring guide [17].

Thereafter, the work of Frigault [18] gave insight on implementing attack graphs as a Bayesian network. This gave us a better understanding about the adversary's capability, likelihood and impact of attacks for various attack scenarios. Dantu [19] and Liu [20] analyzed attacks by assigning probability values to the attack graph nodes. Furthermore, using the concepts of Bayesian networks on these probability values, they calculated potential attack paths and modeled network vulnerabilities. Nevertheless, these computations were not for an attack scenario of a WSN and as such could not be applied to the proposed attack graphs for a WSN. Houmb [21] proposed the methodologies of risk level estimations using the exploitation frequency and impact of vulnerabilities in a wired network. We adopted these concepts to identify the metrics necessary to compute net threat level to the root node of our attack graph when it is represented as a Bayesian network. This gave a degree of diversification and uniqueness to the WSNs with respect to quantitatively analyzing our attack graphs and using the results to estimate maintenance period for the largely unattended WSNs in a Sensor Cloud.

2.2. RISK ASSESSMENT OF TRADITIONAL WEB APPLICATIONS

The task of risk assessment for traditional web application is essential in order to identify the vulnerabilities and threats to the application. However, this process cannot be undertaken when an application is fully developed.

This is because the process of addressing the identified risk in a fully developed application can increase the costs incurred by clients. Therefore, in order to efficiently perform risk assessment on traditional one should consider doing it during the application design phase. This allows us to not only assess different architecture/outline for our application, but also assess the risks that will be associated with them. Thereby, helping to come up with security application architecture in which risks can addressed by optimal allocation of security resources.

With respect to performing design phase application risk assessment, some notable works are Microsoft's STRIDE [22] [23], and EMC's Developer Driven Threat Modeling (DDTM) [26]. Application threat modeling as discussed by Open Web Application Security Project (OWASP) [24] also gives some useful guidance in vulnerability detection and secure application development during its design phase. These methods help to identify, quantify and address the security risks associated with an application. STRIDE is an acronym for the different categories of vulnerabilities that may be present in an application. It stands for: **S**poofing, **T**ampering, **R**epudiation, **I**nformation disclosure, **D**enial of service, **E**levation of privilege. It operates by taking as input the application design in the form its data flow diagram (DFD). Data flow diagrams gives an overview of system functionalities; the data flow among the different system components by showing entry and exit data points as well as where data gets processed. This is passed onto Security Development Lifecycle (SDL) and Threat Modeling tool [25] which conceptualizes STRIDE to identify the set of vulnerabilities related to the elements of an application's DFD. This set however, is not application dependent but rather element dependent. To elaborate this notion, consider the various elements of a DFD - external entities, for e.g. users interacting with the application; process, for e.g. a particular functionality of the application; data store, for e.g. a database storing the data generated and processed by the application; data flow, for e.g. the flow of data between processes and to external entities. STRIDE produces a generic set of vulnerabilities for each element type.

For example, given a data store element, it will generate four possible set of vulnerabilities: {T,R,I,D}. Depending on the type of application and the protocols it uses all the four vulnerability category may not be applicable to its data store. For example, repudiation will not be feasible if all activities in and out of the data store is authenticated and logged by another element of the application. This action can be carried out by using the set of questionnaire provided in the SDL tool to narrow down the vulnerabilities. However it still requires domain knowledge of the application architecture and security expertise. For a professional application design with a DFD consisting of many levels and numerous elements this process can become very tedious. EMC's DDTM improves STRIDE by only considering the data flow elements of the DFD which considerably reduces the amount of elements which needs to be analyzed; the reasoning behind doing this is the the data at flow will only lead to vulnerabilities, otherwise it will be secure behind the boundaries of the application. The rest of assessment technique is the same as that of STRIDE.

Along these lines, the works in [27] [28] also supports the fact that risk analysis has become an integral part in the various stages of the software development process and not just the design phase. Building in security from the design to the testing phase of a software results in more secure software and makes the overall process cost-efficient. Several works present in the literature builds upon STRIDE to improve the risk assessment process like [29], [30], [31], [32]. Furthermore, authors in [33] have addressed the automation of the risk assessment models during the software design phases. The works like [34] have also presented techniques for inclusion of security mechanisms in the various stages of the agile software development process which is now one of popular models in the software development process. In addition to inclusion of security mechanisms in the software design and development stages, several works like [35], [36], [37] have addressed the security aspects in the software testing stages. Their primary focus in this domain being the automated generation of security aware software test cases.

2.3. RISK ASSESSMENT OF CLOUD COMPUTING PLATFORMS

Risk assessment of cloud computing platforms and the services offered by different cloud vendors has been one of the prominent fields of work related to the cloud computing domain. This is because of several factors like migrating the client's application from their private domain to the public domain of the cloud providers, partially losing control over their data, and uncertainty regarding the cloud provider's security policies and their assurance. Several works related to this field can be found in the literature, however, there are still numerous open and emerging security challenges and threats to the domain of cloud computing as identified in [38]. Further, the advent of Internet of Things (IoT) paradigm supported by cloud computing platforms will also introduce new security challenges [39].

This notion is further corroborated from the author's work presented in [40] which surveys risk assessments methodologies related to cloud computing platform. Some of the earlier works in this domain discusses the security impacts uniquely arising due to the cloud infrastructure like in [41], which investigates the threats occurring due to the setup and distribution of cloud services, outlining the threats in terms confidentiality, integrity, and availability of cloud infrastructure. Whereas, other risk assessment works discuss security concerns related to very specific domains such as network based attacks [42], and security concerns related to the data residing on the cloud platform [43]. The authors in [44] have also explored into the possibilities of providing risk assessment as a service to clients contemplating hosting their application on the cloud platform. Other works like [45], introduces security metrics to service level agreement (SLA) documents such that clients can not only assess the quality of service related to a cloud service they want to rent but also the security risks associated to it. Further with respect to assessing the security of cloud platforms a challenging aspect is to keep up with the dynamic environment of the cloud platform as well as the evolving needs of the clients. In this regard, authors in [46] have proposed a tool for dynamic and flexible service provisioning of cloud services to clients which can account for estimation of parameters such as trust, associated risks, and cost.

Such tools will help clients in predicting and efficiently adapting to unanticipated changes in resource requirements on the cloud platform. In more recent trends, authors in [47] propose a risk assessment framework for cloud computing platforms by assessing the SLA as presented by clients and evaluating from the cloud vendor's perspective to check the number of SLA violations that might take place in accordance to the demands presented by the clients in their SLA. Other recent works in risk assessment of cloud service provider as presented in [48], utilizes information gathered from clients and cloud vendors to assess different risk scenarios. They do so by proposing a machine learning framework which leverages the security evaluation documents of various cloud vendors publicly available on Cloud Security Alliance's (CSA) Security, Trust and Assurance Registry or STAR to present a quantitative rating output to describe the risk associated with different cloud services.

Furthermore, other instances of works that utilizes CSA STAR repository and its CAIQs can be found in [49]. In [50], authors discuss another quantitative model to evaluate and compare the security dispensed by CSPs which is based on the concept of developing security service level agreements (SecLAs). The authors make use of the Analytic Hierarchy Process (AHP) for its decision making purpose and evaluate the SecLAs qualitatively and quantitatively. Additionally, authors in [51] present the concepts of automatically enforcing SecLAs in the cloud platforms. The authors propose a model in this regard which engages the software life cycle of the components that is covered in the SecLAs to determine the associated constraints of the security components, the security requirements of the clients and follows it with automatic provisioning and configuration of the selected security resources. However, evaluating and addressing security concerns on the cloud platform is not the end game. Given the dynamic nature of the cloud, architecture and policies are susceptible to change on a short notice. Therefore, clients need assurance that the identified security policies are still in place. The context of security assurance has been discussed extensively in [52]. Along these lines, authors in [53] have proposed the concept of continuous auditing.

This will help to keep up with the dynamic cloud environments and evaluate the change in security policies and their assurances. Furthermore, authors in [54] propose techniques to dispense real-time cloud security assessment. Their approach, called the Moving Interval Process can help clients compare their security assessments with the security offerings of the CSPs and also help CSPs compare their services with other CSPs by understanding the needs of their clients with high accuracy and computational efficiency. Additionally, other works focus solely on evaluating particular types of delivery models like software-as-a-service [55], infrastructure-as-a-service [56], or a specific domain of the cloud security like data security [57].

Nevertheless, performing risk assessment requires an actionable output which its users can make use of. In this regard, quantitative estimates have leverage over qualitative analysis. Therefore, several works can be found across the literature both in terms quantitative risk assessment for software applications [58] and for cloud security [59], [60]. Finally, a pre-requisite to performing risk assessment is to be able to understand and estimate the types of security threat an application (or platform) might be exposed to. In this regards, security attack patterns coupled with misuse cases is good way to elicit the mechanisms in which an application might be attacked. Security attack patterns for software applications have been studied rigorously by the community [62] and its coupling with application abuse [64] and misuse cases [65], [66], [67] have well-found acceptance and feasibility. But in contrast to the well-studied attack and misuse patterns of software applications, cloud misuse patterns [61] are still in its dormant stages and requires further analysis to be able to create a comprehensive knowledge base which can be used in a modular fashion to analyze the risks associated to software applications which will be migrated to the cloud platform.

There have been several researches in the recent past in the direction of cloud migration. Most of which primarily aimed at cost reduction with identification of other critical factors such as organizational policies and some introductory identification of security issues which did not fully address the security concerns from the application's perspective.

The authors in [68] describe a hybrid cloud migration process with the primary goal of reducing total cost of ownership for large scale organizations. They present a comprehensive study, identifying numerous cost factors that govern cloud migration policies. In [69], authors present a case study showing the cost implications of migrating an application to the IaaS platform of Amazon EC2. They have also identified that cost is not the sole criteria affecting cloud migration as other non-financial aspects like reputability of CSPs, technical unfamiliarity play an important role in the decision making process. They extended their case study presenting a comprehensive modeling tool [69] that provides cost estimates of migrating to public IaaS cloud platform. The benefits are in its capability to compare and contrast the services provided by different CSPs. Authors in [70] present the challenges of migrating existing applications to the cloud platform like changes made to the software environment, programming models such that it can operate on the cloud platform. They present a model identifying the tasks involved in the cloud migration process and analyzed them according to the cost involved in performing them. This work helped in the formulation of our migration feasibility model. In [71], authors discuss the technical and non-technical challenges of migrating a web server to the cloud platform. They discuss the nontrivial aspects in cloud migration which involves taking into consideration many criteria, addressing all of which manually is challenging. As such, they propose an automated framework, CloudGenius, for helping clients in the decision making process for cloud migration. Authors in [72] present an overview of the challenges in performing cloud migration like financial aspects, security concerns arising due to cloud migration such as multi-tenancy and data confidentiality. The authors in [73] present a detailed cloud migration decision making framework. It takes into account the architecture of an application to be migrated along with its service requirements in terms QoS factors, security risks involved, and the financial aspects of the process. In [74] [75], the authors discuss some challenges of migrating preexisting and legacy application to the cloud platform. They propose some well-formulated and in-depth approaches to address the underlying challenges.

Authors in [75] perform an in-depth review of existing cloud migration techniques related to hosting legacy systems on the cloud platforms. Their literature review discusses the similarities between Service Oriented Architecture (SOA) and Cloud platforms. In [74], authors present an evolutionary, iterative approach to help small-medium enterprises (SME) is able to migrate their legacy applications to the cloud platform. The outlined solution aims in making the process less dependent on any particular kind of technology support or CSP. Through these works, we were able to identify and acknowledge the impact and challenges of migrating legacy applications to the cloud platform. The work in [76] briefs about the concept of fault tolerance to help high performance computing applications, which are allocated large number of virtual instances to address their computation intensive tasks.

PAPER

I. RISK ASSESSMENT IN A SENSOR CLOUD FRAMEWORK USING ATTACK GRAPHS

Amartya Sen and Sanjay Madria

Department of Computer Science

Missouri University of Science and Technology

Rolla, Missouri 65409–0050

Email: {asrp6,madrias}@mst.edu

ABSTRACT

A sensor cloud consists of various heterogeneous wireless sensor networks. These wireless sensor networks may have different owners and run a wide variety of user applications on demand in a wireless communication medium. Hence, they are susceptible to various security attacks. Thus, a need exists to formulate effective and efficient security measures that safeguard these applications impacted from attack in the sensor cloud. However, analyzing the impact of different attacks and their cause-consequence relationship is a prerequisite before security measures can be either developed or deployed. In this paper, we propose a risk assessment framework for wireless sensor networks in a sensor cloud that utilizes attack graphs. We use Bayesian networks to assess and analyze attacks on wireless sensor networks. The risk assessment framework will first review the impact of attacks on a wireless sensor network and estimate reasonable time frames that predict the degradation of wireless sensor network security parameters like confidentiality, integrity and availability.

Using our proposed risk assessment framework allows the security administrator to better understand the threats present and take necessary actions against them. The framework is validated by comparing the assessment results with that of the results obtained from different simulated attack scenarios.

Keywords: Attack graphs, security, risk assessment, sensor clouds, wireless sensor networks, bayesian networks

1. INTRODUCTION

Sensor cloud [14] consists of several different wireless sensor networks (WSNs). These WSNs are provided, as a service, to users through the sensor cloud platform. WSNs are comprised of low-cost nodes deployed in an ad-hoc fashion over a large area [22] [10] to collect temperature, humidity and other sensitive data, as requested by user applications. These deployments maybe done in hostile environments wherein they are not physically monitored for long time intervals. Additionally, with the integration of WSNs with different ownerships under a sensor cloud platform running a variety of user applications, possibility of attacks are more likely. As such, there is a need for a risk assessment mechanism to estimate the likelihood and impact of attacks on these WSNs in a sensor cloud. We should be able to answer questions like - how can we better secure these networks, what are the possible attacks on our network and their risk level in the presence or absence of various security measures. This will help in strengthening the network security before we integrate a WSN in a sensor cloud. Further, understanding the cause-consequence relationship between different attacks is also necessary. For example, a malware attack can lead to node subversion [8] and an adversary can use the compromised node to break the authentication scheme. This will prompt the execution of other degenerate attacks like sinkhole [18] or Sybil [17].

Although the complete safety of a network in sensor cloud is an idealistic scenario, being able to predict the degradation of WSN security parameters such as confidentiality, integrity and availability [27], and taking appropriate precautions such as redeploying the WSN using improved security measures is always a better alternative.

We studied the works in attack graph for wired networks [23] [4] [21] and research these ideas to adapt in WSNs within a sensor cloud framework. This will help understand the cause-consequence relationship between attacks and identify ways in which a particular WSN security parameter may be exploited. But establishing such relationships alone however, is not sufficient. Empirical evidence [26] suggests that risk evaluation requires a quantitative perspective. Instead of saying that a sensor network is secure, we are more interested in knowing the extent of this security vulnerability and should be able to numerically assess the likelihood and impact of an attack. The National Institute of Standards and Technology's (NIST) Common Vulnerability Scoring System (CVSS) [1] equips us with the means to calculate severity ratings of vulnerabilities in only wired networks. We adapt these mechanisms to quantify the attack nodes in our attack graph, merging them with the concepts of Bayesian networks [5]. Hence, we can numerically evaluate the risk of security parameters within a WSN. In addition to quantitative perspectives, time frames with respect to WSN uptime is another useful estimate that help us reason better. For this purpose, we adapt risk level estimations modeled as a continuous-time Markov process in Houmb's Misuse frequency model [7] for WSNs. This help in predicting the degradation of security parameters within a WSN.

Risk assessment for wired networks using attack graphs has been studied [23] [4] [21], but a sensor cloud consisting of WSNs is inherently different and the existing works on attack graphs cannot be applied. This is because when an attack graph for a wired network is generated, the network is scanned using a vulnerability scanner tool such as Nessus [1]. These scanners detect the list of vulnerabilities (Figure 1) present on each

[1] nvd.nist.gov

Figure 1. List of Vulnerabilities - Nessus Network Scan

system in a wired network which is then parsed into an attack graph generating tool [24]. However, no comparable vulnerability scanning tool currently exists for WSNs. Even if such vulnerability scanning tool existed for WSNs, generating such a list (Figure 1) would be insufficient. This is because sensor nodes in a WSN collaborate to achieve a common goal and suffer from inherent resource limitations. These limitations are the primary cause of WSN vulnerabilities as they do not permit the application of desired security protocols to safeguard the network. Hence the vulnerability list will be identical for all nodes and no concrete conclusions could be formulated. Thus, rather than focusing on vulnerabilities in a sensor node or network, we focus on the feasibility of attacks on a particular WSN in a sensor cloud. The successful execution of different attacks will vary according to security measures used, tasks being carried out, and deployed environment of a WSN.

Due to the above factors, risk assessment in WSNs under a sensor cloud is challenging and hence, we propose the following methodologies:

- We formulate attack graphs to depict the logical correlation between the attacks on WSNs [20] (Section 2.2). These attack graphs will then used to analyze how the attacks can exploit the WSN security parameters [27].

- We depict the attack graphs as a Bayesian network (Section 2.3), quantifying the likelihood and impact of the attacks in a WSN and the net threat level to WSN security parameters.

- We compute time frames predicting the degradation of WSN security parameters by modeling our risk level estimations as a continuous-time Markov process (Section 2.4). Given these time frames we can take precautionary measures and perform maintenance in an unattended WSNs before they reach an irreparable state.

2. PROPOSED METHOD

The proposed risk assessment framework will determine the likelihood and the impact of the attacks on a WSN. The likelihood of attacks is influenced by factors such as sensor node configuration, topology and routing measures. Additionally, execution of an attack increases the possibilities of other attacks [8]. These types of interdependencies between the attacks can be modeled using attack graphs. Quantifying these attacks based on CVSS parameters helps us to determine the feasibility of various attacks. When we merge these attack graphs with the principles of Bayesian networks, we can estimate the net impact of the feasible attacks on the WSN security parameters. A flow chart summarizing the risk assessment framework is illustrated in Figure 2.

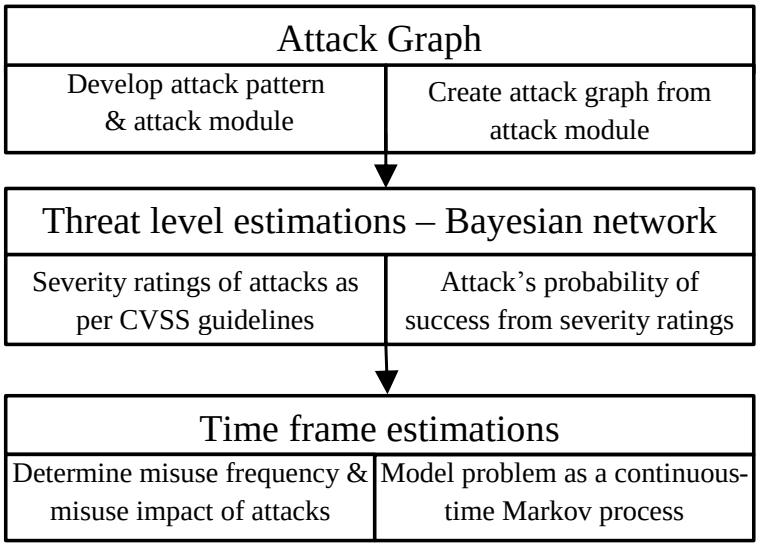

Figure 2. Proposed Risk Assessment Framework

2.1. The Sensor Cloud Network Architecture. A sensor cloud, composed of WSNs owned by different entities, provides sensing services to users. Sensor cloud separates network ownership from users, allowing multiple users to access the available WSNs simultaneously. This paradigm has been realized by the use of virtual sensors present on the cloud platform which are incorporated as an image of the actual physical sensors. The users deal with these virtual sensors which contains information about the actual physical sensors working for a user's application.

Sensor cloud consists of three layers, namely - client centric layer, middleware layer, and sensor-centric layer. The client centric layer deals with users and acts as an user interface for the sensor cloud. The middleware layer, is responsible for tasks such as - service negotiation management, integrating the communication between client centric layer and sensor centric layer, and managing the virtual sensors. Sensor centric layers deals with the actual physical sensors. Since risk assessment using attack graphs for wired networks are already present, we will not delve deep into this sphere. We would deal with the risk assessment for the physical WSN of a sensor cloud network. A WSN in a sensor cloud is considered as an undirected graph G = (V, E), where V is the set of vertices containing the sensor nodes and E is set of edges denoting connectivity between these nodes. For simplicity, we have assumed these WSNs to be static.

2.2. Attack Graphs for Wireless Sensor Networks. Attack pattern generation is a prerequisite in developing attack graphs. It gives insight about the goal of an attacker and allows us to avert the attack. An attack on a WSN will tend to exploit one or more of the WSNs security parameters viz. confidentiality, integrity, and availability. Thus, we can group the attacks according to the security parameter they tend to exploit. This will help in developing the attack pattern for attacks on a WSN [2].

Definition 2.1. Attack Pattern: *An attack pattern is a tuple $P_i = (s_i, \phi)$. Where, s_i, is the attack and, $\phi \in$ [Confidentiality, Integrity, Availability], is one of the exploited WSN security parameters.*

Attacks on a WSN can be categorized as active or passive attacks [19]. Active attacks, such as sinkhole, are executed to alter the network resources or operations such as routing protocol. Passive attacks, such as eavesdropping, are executed to gather information about the network which can then be used to execute Active attacks. The set of attacks along with their definition and attack pattern [27] [28] [2] is given in Table 1.

Table 1. Attack Definition and Attack Pattern

Attack	Definition	Target Security Parameter
Eavesdropping	Listening to communication between the nodes	C
Jamming & DoS	Disrupting network communication by jamming communication frequency or sending additional garbage packets the network cannot handle	A
Node Subversion	Adversary taking over a legitimate sensor node	C; I
Sybil	Adversary node creating false virtual node identities and making them seem like legitimate sensor nodes	C; I
Spoofing	Adversary node pretending to be a legitimate node of a WSN	C; I
Altering; Replay	Continuously changing the route of packets; Sending the same packets over and over again	I; A
Wormhole; Sinkhole; Blackhole	Adversary falsely advertising efficient route paths and rerouting traffic from actual paths - All traffic can now pass through the adversary & (s)he may choose to drop all the packets	C; I; A
Selective Forwarding	Once packet traffic gets rerouted through the adversary (s)he may decide to selectively drop some of the packets	C; I; A
Acknowledgment Spoofing	Falsifying acknowledgment during authentication procedure or when legitimate sensor nodes are trying to identify its neighbors	C; I
Node Malfunction	A legitimate sensor node functionally abnormally due to scare resources or a malware running on them	I; A
Node Replication	An adversary creating a rogue sensor node which is based on a legitimate sensor node	C; I
False Data Injection	Adversary introducing garbage packets within the actual packet transmission	I
Node Outage	When a legitimate sensor node is no longer able to function	A
Directed Physical Attack	Physical damage brought unto the sensor nodes	A
Hello Floods	Continuous Hello messages sent to sensor node, which makes them unable to handle any other messages and drains their resources as a result of constantly having to deal with these Hello messages	A
Desynchronization	Sensor nodes constantly trying to re-establish broken communication and not being able to do so	A
Malware Attack	Execution of a malicious code on a legitimate sensor node	C; I; A

Further, to understand the cause-consequence relationships between attacks on a WSN, we should be aware of the conditions required to execute a particular attack (pre-conditions of an attack) and the consequences of successful execution of an attack (post-conditions of an attack). If the post-conditions of an attack satisfies the pre-conditions of another attack then these two attacks will have a cause-consequence relationship (an edge in the attack graph). We develop an attack module that will capture the cause-consequence relationship between the attacks (Table 2).

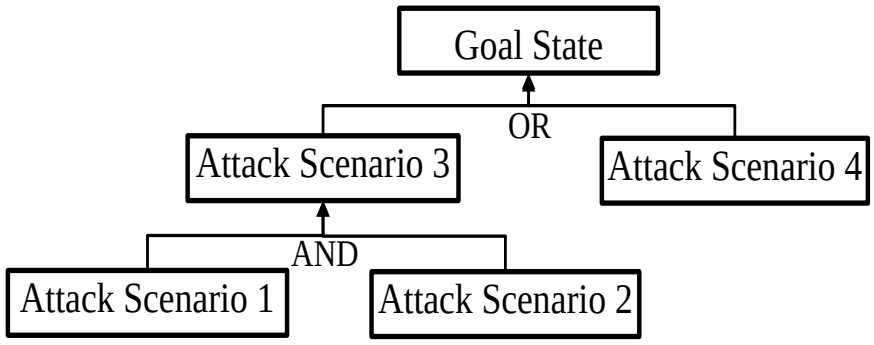

Figure 3. An Illustration of an Attack Graph

In some cases, pre-conditions of an attack may be satisfied by the post-conditions of a single attack, these kinds of attacks will be connected by an OR type join in the attack graph (Figure 3). Whereas, if post-conditions of two or more attacks are required simultaneously to satisfy the pre-conditions of an attack, they are connected by AND type joins.

Definition 2.2. Attack Module: *An attack module is defined as a tuple, (P_i, S_{pre}, S_{post}, ϵ), where P_i is the attack pattern, s_{pre} is the pre-conditions required to execute the attack, s_{post} are the post-conditions after the execution of the attack, and ϵ is the join type, $\epsilon \in$ [OR, AND].*

Once the attack module has been developed, we can use it to implement attack graphs for each of the WSN security parameters. By doing so, we can visualize the ways in which an attacker might exploit the WSN security parameters. Additionally, the root node of an attack graph is considered to be an attacker's ultimate goal. Hence, exploitation of one of the WSN security parameters - confidentiality, integrity or availability, will be the root node of an attack graph for a WSN. All other nodes in the attack graph will be intermediate states an attacker can take during the course of their attack. We assume that an attack state cannot be undone once it has been exploited, preventing any form of backtracking.

Table 2. Attack Module

Attack	Pre-condition	Post-condition	Join Type
Eavesdropping	Adversary node within physical reach of WSN & unencrypted communication	Acquire unencrypted data	AND
Jamming & DoS	Adversary within the reach of WSN	Disrupt network communication	OR
Node Subversion	Inability to authenticate; running malicious codes	Attacks that require bypassing authentication	OR
Sybil	Node Subversion; Inability to authenticate	Bypass authentication; breaking of key distribution	AND
Spoofing	Sybil; Node Subversion	Further degradation of authentication	OR
Altering; Replay	Sybil; Spoofing	Loss of energy; Loss of integrity	OR
Wormhole; Sinkhole; Blackhole	(Node Subversion)Sybil; Spoofing	Route control; Packet drop - partial or complete; communication disruption	AND-OR
Selective Forwarding	Wormhole; Sinkhole	Network more vulnerable to replay and rerouting	AND
Acknowledgment Spoofing	Node Subversion; Sybil	spoofing; wormhole; sinkhole	AND
Node Malfunction	Low Energy of the sensor nodes; Execution of Malicious codes on the sensor	Erroneous data in the network	OR
Node Replication	Physical Tampering	Adversary node legitimate part of WSN	AND
False Data Injection	Adversary node; In communication range of WSN	Presence of garbage data in the WSN; jamming due to more traffic	AND
Node Outage	Severe Energy drain; Directed Physical Attack ; Blackhole	Network disruption	OR
Directed Physical Attack	Topology discovery	Node Outage	OR
Hello Floods	Spoofing; Sybil	Energy drain; breaking route table due to continuous Hello Packet transmissions	OR
Desynchronization	Node Outage; Hello Floods; Replay; Reactive Jamming	Communication disruption	OR
Malware Attack	architecture of the sensor nodes; user can execute his/her code	Node Subversion	AND

Definition 2.3. Attack Graph: *An attack graph is a tuple containing the attributes (s_{root}, S, τ, ϵ), where s_{root} is the goal of the attacker - one of the WSN security parameters. S denotes the complete set of attacks (Table 1). τ denotes the set of pre- and post-conditions of all attacks in S. ϵ is the join type, $\epsilon \in$ [OR, AND].*

2.3. Risk Assessment Using Bayesian Networks. An attack graph's nodes maybe assigned with either true or false values implying that an attack state is either successfully executed or not executed at all. To better analyze an attack scenario, we can assign numerical values, like the probability of success of executing an attack s_i, $Pr(s_i)$. The probability of success of attacks can be derived from their severity ratings and can be computed by adopting the scoring metric established by CVSS for wired networks.

CVSS scores are based on three criteria; base metrics, temporal metrics and environmental metrics. Base metric is used to express the level of difficulty to exploit a vulnerability. Temporal and environmental metrics are used to express the effect of the network's deployment environment and surroundings in the exploitation of the vulnerability. A brief description about their sub-categories and attributes is given in Table 3.

Table 3. CVSS Metric Description

CVSS metrics	Sub-categories	Attributes	Description
Base	Exploitability sub-score	Access Vector Access Complexity Authentication Instances	Vulnerability exploitation from within network domain or remotely Difficulty in exploiting vulnerability Number of authentication measure that needs to be bypassed
	Impact sub-score	Confidentiality Integrity Availability	Impact on confidentiality Impact on integrity Impact on availability
Temporal	Exploitability tools & techniques (T_{E}) Remediation level (T_{RL}) Report Confidence (T_{RC})		Current state of available exploitation techniques Type of solutions available to fix the vulnerability Evidences available about existence of vulnerability
Envnmnt	Collateral Damage Security required		Impact of exploited vulnerability on organization's economy Amount of security required for organizational assets like confidentiality, integrity and availability

A detailed description of the CVSS scoring system is available in the CVSS scoring guide [16]. WSN attacks have not yet been corroborated by CVSS and the base metrics will be evaluated subjectively. We assumed that, although preventive measures for the attacks are available, they are the solutions that individual researchers have reported. Hence, based on the definition of *remediation level*, we have considered these solutions as *workaround* fixes. Environmental metrics are context specific varying for different organizations and being constant for any given organization. Houmb's misuse frequency model [7] performs risk level estimations as a conditional probability over the *Misuse frequency* (MF) and *Misuse Impact* (MI) estimates of an attack. MF and MI of an attack helps in depicting the likelihood and impact of an attack respectively taking into account the intrinsic attributes of the attack, network architecture, and security measures used. It is useful in estimating time frames predicting the degradation of organizational assets like confidentiality, integrity and availability. Hence, to compute the probability of success of attack nodes in our attack graph, we have adopted Houmb's misuse frequency model. MF of an attack is calculated using (1) - (3) and CVSS parameters specified in Table 4. A detailed depiction of the rating values in Table 4 can be found in [16]. Initial misuse frequency, MF_{init}, in (1) is calculated using exploitability sub-score under the base metrics (Table 3 and 4). We normalize the values of B_{AR}, B_{AC}, B_{AU} for the attack considered, to keep the final score between 0→1 (MF is a probability and cannot be over 1).

The MF of an attack however, may change over time according to the availability of security solutions and techniques for executing the attacks. These factors are reflected using temporal metrics, computed as MF_{uFac} (2).

$$MF_{init} = (\frac{1}{3}) \sum_{s_i \in S} (B\_\{AR\}, B\_\{AC\}, B\_\{AU\}) \tag{1}$$

$$MF_{uFac} = (\frac{1}{3}) \sum_{s_i \in S} (T\_\{E\}, T\_\{RL\}, T\_\{RC\}) \tag{2}$$

$$MF = (\frac{1}{2}) \sum_{s_i \in S} (MF_{init}, MF_{uFac}) \tag{3}$$

Table 4. CVSS vectors to calculate MF

CVSS metrics group	CVSS Attributes	Rating	Rating value
Base Metrics	Access vector (B_{AR})	Local (L)	0.395
		Adjacent network (A)	0.646
		Network (N)	1.00
	Attack complexity (B_{AC})	High (H)	0.35
		Medium (M)	0.61
		Low (L)	0.71
	Authentication instances (B_{AU})	Multiple (M)	0.45
		Single (S)	0.56
		None (N)	0.704
Temporal Metrics	Exploitability tools & techniques (T_{E})	Unproved (U)	0.85
		Proof-of-Concept (POC)	0.90
		Functional (F)	0.95
		High (H)	1.00
	Remediation level (T_{RL})	Official Fix (OF)	0.87
		Temporary Fix (TF)	0.90
		Workaround (W)	0.95
		Unavailable (U)	1.00
	Report Confidence (T_{RC})	Unconfirmed (UC)	0.90
		Uncorroborative (UR)	0.95
		Confirmed (C)	1.00

MF_{uFac} is then added to (MF_{init}) and the final misuse frequency (MF) is computed in (3). Similar computations are done to calculate MI using the impact sub-score under base metrics and environmental metrics (Table 5) [16] and (4) - (7). Initial MI estimate, MI_{init}, was estimated using impact sub-score of the base metrics in (4). This estimate is a vector depicting the effect of an attack on confidentiality, integrity, and availability of a network. MI_{init} was then updated on the basis of the collateral damage potential (E_CDP) in (5). The MI estimates were further updated as per the security requirements information in (6). Finally, the resulting MI estimate, MI, is obtained in (7).

Table 5. CVSS vectors to calculate MI

CVSS metrics group	CVSS Attributes	Rating	Rating value
Base Metrics	Confidentiality Impact (B_{C})	None (N)	0.00
		Partial (P)	0.275
		Complete (C)	0.660
	Integrity Impact (B_{I})	None (N)	0.00
		Partial (P)	0.275
		Complete (C)	0.660
	Availability Impact (B_{A})	None (N)	0.00
		Partial (P)	0.275
		Complete (C)	0.660
Environmental Metrics	Confidentiality Requirement (E_{CR})	Low (L)	0.50
		Medium (M)	1.00
		High (H)	1.51
	Integrity Requirement (E_{IR})	Low (L)	0.50
		Medium (M)	1.00
		High (H)	1.51
	Availability Requirement (E_{AR})	Low (L)	0.50
		Medium (M)	1.00
		High (H)	1.51
	Collateral Damage Potential (E_CDP)	None (N)	0.00
		Low (L)	0.10
		LowMedium (LM)	0.30
		MediumHigh (MH)	0.40
		High (H)	0.50

$$MI_{init} = [B\_\{C\}, B\_\{I\}, B\_\{A\}] \tag{4}$$

$$MI_{CDP} = E\_CDP[MI_{init}] \tag{5}$$

$$MI_{Env} = [B\_\{CR\}, B\_\{IR\}, B\_\{AR\}] \tag{6}$$

$$MI = MI_{CDP} \times MI_{Env} \tag{7}$$

Once we compute the MF, we can estimate an attack's probability of success, Pr(s_i), using (8):

$$Pr(s_i) = (1 - \mu)MF_{init} + \mu(MF_{uFac}) \tag{8}$$

where, μ is a constant and is defined as the security administrator's belief of the impact of the security measures on an attack's MF (base metrics) and temporal metrics. It can vary from [0,0.5]. If a security administrator is uncertain about the impact of security measures of an attack (like in cases of new or unknown attacks), then the probability of success is based on the base metrics, by taking μ as 0. After assigning attack graph's nodes with their probability of success, we depict it as a Bayesian network. This will help us ascertain the non-deterministic nature of the attacks for different network scenario with a reasonable amount of accuracy.

A Bayesian attack graph in our framework can be defined as:

Definition 2.4. Bayesian Attack Graph: *A Bayesian attack graph is a tuple containing the attributes (S, τ, ϵ, Pr), where, S, is the complete set of possible attacks on a WSN (Table 1), τ, is the set of the pre- and post-condition of all the attacks in S, ϵ is the join type, $\epsilon \in$ [OR, AND], for the attacks in S, and Pr, is the set of probability values specifying the success rate of an attack node in a Bayesian attack graph.*

For an attack $s_i \in S$, $Pa[s_i]$ denotes the parent set of attack s_i i.e. if an attack $s_j \in Pa[s_i]$, post condition of attack s_j will lead to pre-condition of attack s_i. As such, attack s_j will be the parent of attack s_i in the attack graph. The probability of each attack's success is captured in a Local Conditional Probability Distribution (LCPD) table. These values are assigned as per the subjective belief of the security administrator regarding their network. LCPD can be defined as:

Definition 2.5. Local Conditional Probability Distribution: *For a Bayesian attack graph containing the tuples (S, τ, ϵ, Pr), the local conditional probability distribution function of any $s_i \in S$ is given as $Pr(s_i|Pa[s_i])$ and is defined as,*

1. $\epsilon = AND$

$$Pr(s_i \mid Pa[s_i]) = \begin{cases} 0, \exists s_j \in Pa[s_i] \mid s_i = 0 \\ Pr(\bigcap_{s_i=1} s_i), otherwise \end{cases} \tag{9}$$

2. $\epsilon = OR$

$$Pr(s_i \mid Pa[s_i]) = \begin{cases} 0, \forall s_j \in Pa[s_i], s_i = 0 \\ Pr(\bigcup_{s_i=1} s_i), otherwise \end{cases} \tag{10}$$

2.3.1. Static risk assessment. The difficulty of executing an attack is given by its probability of success, $\Pr(s_i)\ \forall\ s_i \in S$, also known as the prior probability. With this set of prior probabilities captured in a node's LCPD, we can compute the unconditional probabilities. Consider the attack scenario described in Figure 4.

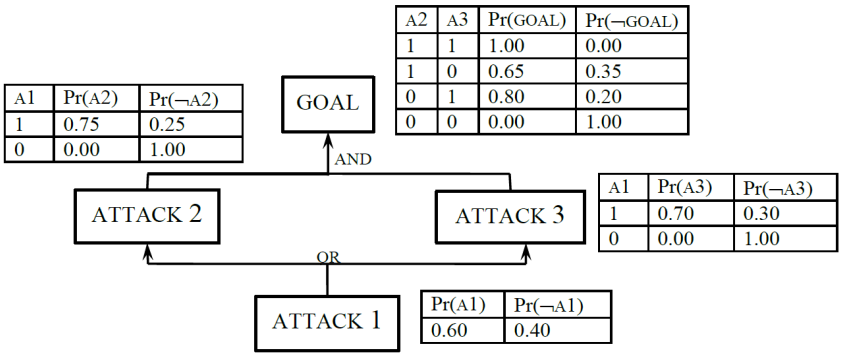

Figure 4. An Attack Graph Represented as a Bayesian Network

The probability of attack 1's success is assigned based on the subjective belief of a security administrator. Using the concepts Bayesian networks, we estimate the probability of success of the remaining nodes in the attack graph. Once we have the probability of success of every node, we calculate the joint probabilities of all the variables in a Bayesian network using the chain rule in (11). The unconditional probability of the goal state is then computed as the joint probability of all of the nodes effecting the goal state's outcome. Thus, in Figure 4, $Pr(Goal)$ will be computed as shown in eq. 12. The unconditional probability of each node is computed similarly by considering the sub-tree rooted at that node.

$$Pr(s_1, ..., s_n) = \prod_{i=1}^{n} Pr(s_i \mid Pa[s_i]) \tag{11}$$

$$\begin{aligned}
Pr(Goal) &= Pr(Goal, A3, A2, A1) \\
&= Pr(Goal \mid A3, A2) \times Pr(A2 \mid A1) \times Pr(A3 \mid A1) \\
&= \sum_{A3, A2, A1 \in \{T, F\}} [Pr(Goal \mid A3, A2) \times Pr(A2 \mid A1) \times Pr(A3 \mid A1) \times Pr(A1)] \\
&= (1.0 \times 0.75 \times 0.70 \times 0.60)_{TTT} + \\
&(0.65 \times 0.75 \times 0.30 \times 0.60)_{TFT} + \\
&(0.80 \times 0.25 \times 0.70 \times 0.60)_{FTT} \\
Pr(Goal) &\approx 0.49
\end{aligned} \tag{12}$$

2.3.2. Dynamic risk assessment. Static risk assessment is done by assuming non-zero prior probabilities of the attacks. However, once WSNs in a sensor cloud has been deployed, we may observe evidence of certain attacks. The probability of success of those attack node will become one, leading to re-evaluation of risk level estimations. We did this using the Bayesian inference techniques of forward and backward propagation. Successor of the attack node with probability 1 will be updated by forward propagation. The initial assumptions on all prior probabilities during static risk assessment will be corrected with backward propagation. The updated unconditional probabilities are known as posterior probabilities. Given a set of attacks $s_i^{'}$ for which we have evidence of exploit, the probability of success for those attack nodes is now 1. Thus, we need to determine the probability of success for the attack nodes that are affected by $s_i^{'}$, i.e. the set of $s_j \in \{S - s_i^{'}\}$. We compute the posterior probability, $Pr(s_j \mid s_i^{'})$ using Bayes theorem:

$$Pr(s_j \mid s_i^{'}) = \frac{[Pr(s_i^{'} \mid s_j) \times Pr(s_j)]}{Pr(s_i^{'})} \tag{13}$$

where $Pr(s_j)$ and $Pr(s_i^{'})$ are the prior unconditional probabilities of the corresponding nodes. The conditional probability of joint occurrence of $s_i^{'}$ given the states of s_j is denoted by $Pr(s_i^{'} \mid s_j)$. In Figure 4, If we have evidence of the goal state being compromised, we can compute its effect on attack 3 by assuming $Pr(Goal) = 1.0$ and eq. 13. Hence, the unconditional probability of attack 3 has gone up from 0.49 to 0.85 in the evidence of an exploit. Similar computations can be done for other nodes.

$$Pr(A3 \mid Goal) = \frac{[Pr(Goal \mid A3) \times Pr(A3)]}{Pr(Goal)}$$

$$Pr(Goal \mid A3) = \sum_{A2 \in \{T,F\}} [Pr(Goal \mid A2, A3 = T) \times Pr(A2)$$

$$= (1.0 \times 0.49)_T + (1.0 \times 0.51)_F$$

$$Pr(Goal \mid A3) = 1.0$$

$$Pr(A3) = 0.42 \quad \text{and} \quad Pr(Goal) = 0.49$$

$$\text{Therefore} \quad Pr(A3 \mid Goal) = 0.85$$

2.4. Time Frame Estimation. Predicting time frames for degradation of a WSN security parameter is a useful tool to carry out maintenance of an unattended WSN and take precautionary measures before they reach an unrepairable state. The MF and MI estimates of WSN attacks is used to build such a risk level estimation. We model our risk assessment framework as a continuous-time Markov process. Such a model consists of a finite state space E, having n service levels - SL_0 to SL_n. Each service level is a subset in E. We define a service level as:

Definition 2.6. Service Level: *Service level is a state composing of non-empty sets of attacks $s_k \in S$. Attacks belonging to a service level have equivalent misuse impact.*

Attacks are grouped based on their MI on WSN security parameters - confidentiality (C), integrity (I), and availability (A). This gives us the number of service levels in the state transition model. The first service level, SL_0, has no impact on a WSN security parameter, in contrast to the final service level, SL_x which has full impact. The time frame estimation will be a two-step process; (1) Develop state transition model from MI estimates: Creation of service levels. (2) Compute state transition rates from MF estimates using a rate transition matrix: the probability of transition from a service level with lower impact to that of a service level with higher impact. These two processes are elucidated in Section 3.4.

3. EXPERIMENTS AND RESULTS

In the following sections, we outline our experimental results and analysis. Thereafter, we present our observations regarding the feasibility of our proposed approach.

3.1. Network Deployment. The deployed sensor cloud network consisted of five WSNs. The deployment region of these networks were pre-defined. Three of the WSNs were deployed on the second floor of the Missouri S&T computer science department and the remaining two on the third floor. The networks on the second floor consisted of three sensor nodes each in addition to the base station and those on the third floor consisted of four and five sensor nodes respectively. The sensor cloud served three users, tasked to sense three different phenomena - temperature, humidity and light intensity. This network deployment was used to perform validation of the proposed risk assessment framework (Section 4).

3.2. Attack Graph for WSN Security Parameters. This section shows the representation of attack graphs using SeaMonster [2] security modeling software. They were modeled to depict the net threat level for individual WSN security parameters for one of the WSN deployed on the second floor (Section 3.1). We calculated the probability of success for each attack node in the attack graph to determine the threat level estimations (Section 2.3.1, eq. 11). This was done assuming that the WSN had no security measures and thus all attacks (Table 1) were feasible. This assumption helps in illustrating the complete set of attacks on a WSN through an attack graph. In the presence of security measures for a particular attack, the probability of success for that attack node will be zero (Section 4.1). For the illustrated attack graphs (Figure 5, 6, 7), the expected threat level for a particular security parameter was assumed to be 50% (expected probability of successful exploitation of the goal state). The expected threat level indicates the subjective belief of a security administrator about the chances of degradation of their WSN's security parameter due to an attack. In the absence of security measures, there is an equal likelihood of an attack being feasible or otherwise. Although one can argue that absence of security measures should prompt in a higher percentage of expected threat level, but to encompass the uncertainty of an attacker's attack, the assumption of 50% expected threat level is justified.

[2]www.sourceforge.net/projects/seamonster

The outcome of the net threat level, due to the executed attacks on a WSN security parameter, will vary based on the expected threat level. The effect on computation of net threat level as expected threat level increases is discussed in Section 3.3. The unconditional probability of success for each attack is given along with nodes of the attack graph. These values are used to compute the net threat level. They are computed adopting the techniques used to calculate the severity rating for vulnerabilities of wired networks established by CVSS (Section 2.3).

3.2.1. Confidentiality. We select the attacks from Table 1 with confidentiality of a WSN as their attack pattern. These attacks become the attack nodes in the attack graph for confidentiality. Then, we analyze the pre- and post-conditions of each of these attacks from the attack module (Table 2), building the logical correlation and depicting it via the attack graph. Once we have created the attack graphs, we need to assign the attack nodes with their probability of success. This is done by evaluating the Base metrics and Temporal metrics (Table 4) for the attacks. An instantiation of this evaluation is given in Table 6. Misuse frequency (MF) is then computed using (1) - (3). Confidentiality of a WSN can be exploited via eavesdropping attack node (Eav) by successful execution of topology discovery (TD) and adversary within communication range (ACR) attacks conjunctively (Figure 5). We compute the unconditional probability of eavesdropping attack node using (9) and MF of these three attack nodes (Eav: 0.82, TD: 0.74, ACR: 0.825).

Table 6. Evaluation of Misuse Frequency of attacks on WSN

Attack Name	Base_Metrics (B_AR,B_AC,B_AU)	Temporal_Metrics (T_E,T_RL, T_RC)	MF $_{init}$	MF$_{uFac}$
Eavesdropping	Adj,Low,None	F,W,C	0.686	0.966
Node Subversion	Network, Medium, Single	POC,W,C	0.723	0.95
Sinkhole/Selective Forwarding	Network,Medium,MI	FEE,W,C	0.686	0.966

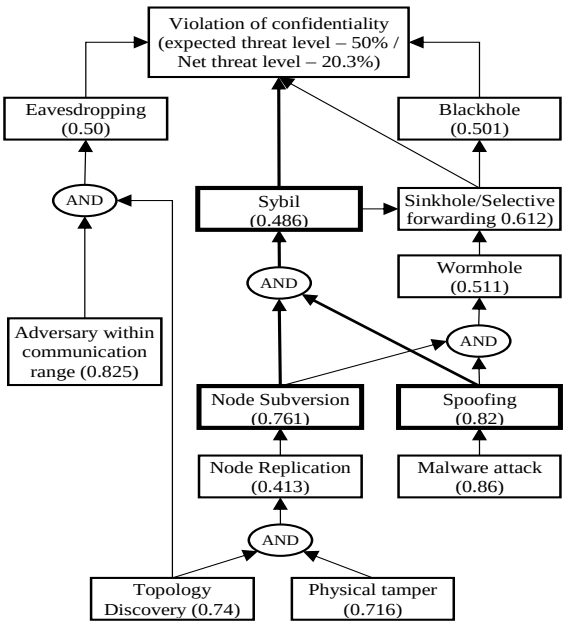

Figure 5. Attack Graph for Exploitation of Confidentiality

Then, we compute the unconditional probability for Eav using (9) as shown in eq. 14.

$$\begin{aligned} Pr(Eav)_{uncond.} &= Pr(Eav)_T * Pr(TD)_T * Pr(ACR)_T \\ &= (0.82 * 0.825 * 0.74) = 0.50 \end{aligned} \tag{14}$$

Similarly, node subversion (NS) can be exploited via successful execution of either malware attack (MA) or node replication (NR) attack node. The MF of NS, NR, and MA is 0.83, 0.413 and 0.86 respectively. Since, we have a disjunctive join in the attack graph, the unconditional probability for NS will be computed using eq. 10. Similar computations are done for other attack nodes. We see from Figure 5 that an attacker can exploit confidentiality of a totally unprotected WSN by executing Eavesdropping, Sybil, Blackhole, or Selective forwarding, either individually or in combination, giving them 2^4 attack options. But some of these combinations will not contribute towards the exploitation of confidentiality. For example, Blackhole attack is a successful consequence of Selective Forwarding.

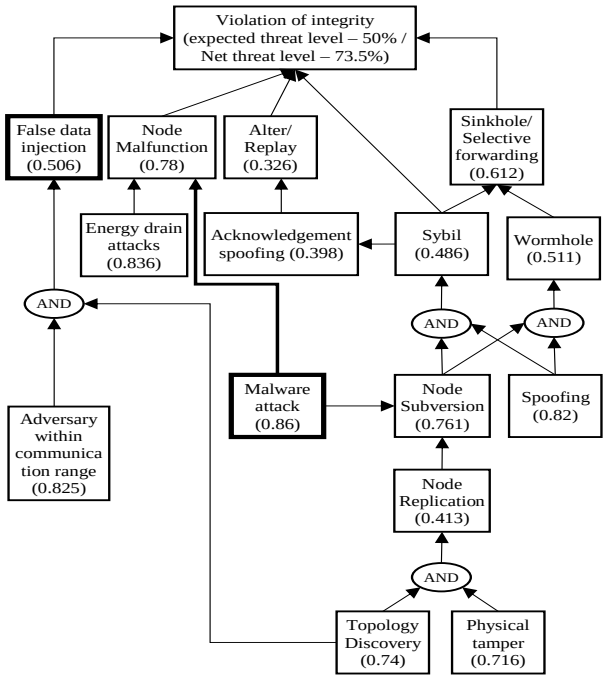

Figure 6. Attack Graph for Exploitation of Integrity

If Selective Forwarding does not contribute towards the exploitation of confidentiality then there will be no contribution from Blackhole. Hence for an expected threat level of 50%, the computed net threat level for confidentiality will be 20.3%. Since it is very hard to get hold of the information unless the adversary knows the location of the sensor nodes and can closely monitor and capture the traffic. Given the protective measures used, this can be challenging since the adversary must decipher the captured information.

$$\sum_{(NR,MA)\in\{T,F\}} (Pr(NS)_T * Pr(NR) * Pr(MA))$$

$$Pr(NS)_{uncond.} = 0.761$$

3.2.2. Integrity. Figure 6 depicts the exploitation of integrity of a WSN through the execution of five attack states, either individually or in combination. The attack states are - False data injection, Node malfunction, Alter/Replay attack, Sybil, and Sinkhole.

This gives an attacker 2^5 options to exploit integrity, out of which few will be nullified based on the unsatisfying JOIN criteria of an attack graph. For a node having AND join, if any one of its child nodes are not executed, then the probability of execution of that attack is zero. For example, if Sybil is not executed (Figure 6), the chances of exploiting integrity via Alter/Replay attack goes down to zero. Hence, for an expected threat level of 50%, the computed net threat level for integrity is 73.5%. The high threat level to integrity is as a result of the fact that even if an adversary cannot decipher the meaning of the captured information, they can very well corrupt the information such that even the destination cannot comprehend it.

3.2.3. Availability. The attack graph for availability in Figure 7 illustrates that an attacker can cause service disruption by exploiting either of three attack states - Frequency jamming, Node outage, and Desynchronization. Executing these attacks in combination, gives them up to 2^3 choices to cause permanent or partial breakdown of WSN services. Hence, for an expected threat level of 50%, the computed net threat level for availability is 49.3%. The means to exploit availability of WSN is quite simple (jamming, node outage) but quite difficult to counter. As such the expected threat level is close to net threat level.

3.3. Expected Threat Levels vs. Net Threat Levels. The computation for net threat level of WSN security parameters in previous section was done assuming expected threat level to be a probability of 0.5. In this section, we will depict the computation of net threat level by varying expected threat level from a probability of 0.0 to 1.0, incrementing it by 0.1 each time. The y-axis depicts the probability values of the net threat levels and the x-axis depicting the variation of expected threat level from 0.0 to1.0.

Figure 8 illustrates the trend of varying expected threat level versus that of net threat level for the three WSN security parameter - confidentiality, integrity and availability. As one increases the expected threat level, the security measures implemented will also be high. Increase in security measures makes the exploitation of confidentiality complex.

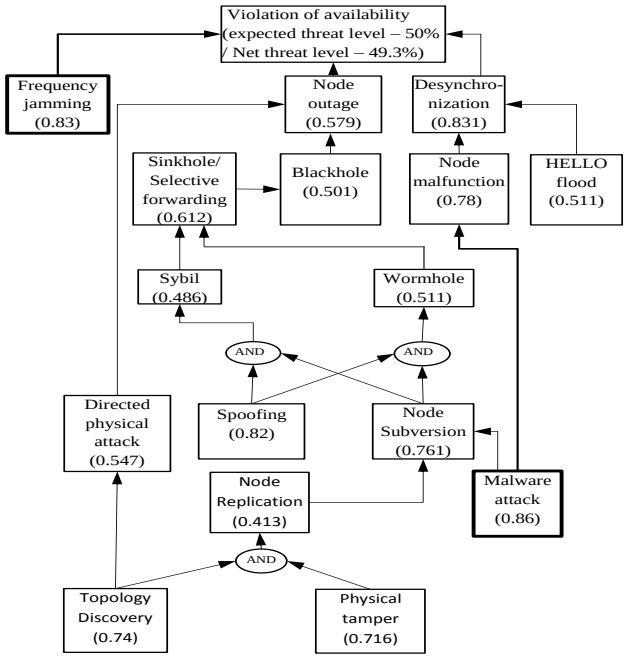

Figure 7. Attack Graph for Exploitation of Availability

Although, the net threat level towards integrity goes up. This is because an attacker does not need to decipher the captured information to exploit integrity. They can corrupt the information itself. Thus, as the belief for exploitation of integrity rises, we see that the net threat level goes higher than the expected value with probability of 0.7 being the threshold value depicting the non-deterministic nature in the estimation of net threat level for integrity. Similarly, availability can be exploited by jamming the sensor node's communication frequency or congesting the sensor nodes with a large number of packets (denial of service). These mechanism are simple to execute and difficult to counter. Further, being able to deterministically predict these attacks are difficult. As such the expected threat level and net threat level does not vary much.

3.4. Time Frame Estimation. We estimate reasonable time frames depicting the degradation of WSN security parameters in the absence of security measures. The first step in computing time frames of degradation is to create service levels (Section 2.4). These service levels are based on the Misuse Impact (MI) of the attacks.

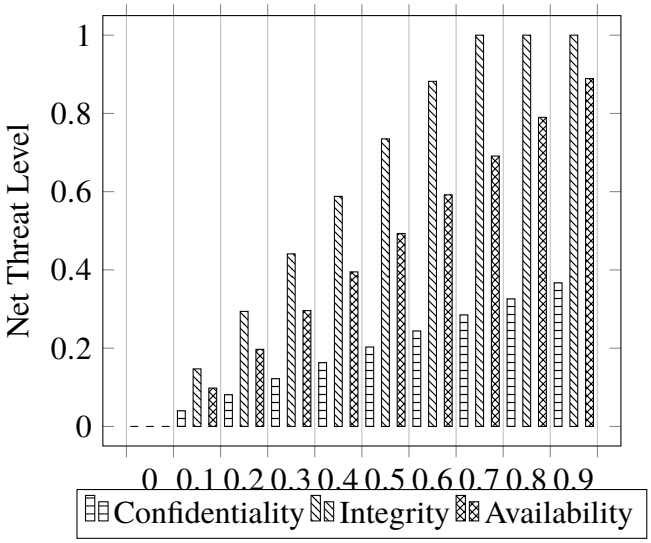

Figure 8. Expected Threat Level vs. Net Threat Level

The MI of attacks is computed using the environmental metric and impact category of base metric (Table 5). The environmental metric depends on an organization deploying the WSN. Assuming we have three different WSN - A, B, and C, their environmental metrics can be summarized as shown in Table 7. Based on these metric the times frame estimations will vary from one physical WSN to another as a result of different number of service levels. Given the metrics in Table 7, WSN_A has 4 service levels for confidentiality, 4 service levels for integrity and 5 service levels for availability. Similarly, WSN_B has 4 service levels for confidentiality (MI - 0,0.15,0.30,1), 3 for integrity (MI - 0,0.05,1) and 4 for availability(0,0.15,0.30,1). WSN_C has 4 service levels for confidentiality (MI - 0,0.16,0.39,1), 4 for integrity (MI - 0,0.16,0.39,1) and 4 for availability (MI - 0,0.05,0.13,1). Computation for time frames using Misuse Frequency (MF) is shown assuming the environmental metrics of WSN_A (Section 3.4.1).

3.4.1. Developing service levels. Computing the MI of the attacks whose attack pattern is confidentiality (Table 1), we have two sets of impact - 0.14 and 0.33 (Table 5 and (7)), along with service levels SL_0 (fully operational) and SL_x (total degradation). The attacks having an impact of 0.14 was grouped into service level SL_1 and those having an

Table 7. Environmental Metric for Physical WSN

Network	Confidentiality requisite	Integrity requisite	Availability requisite	Collateral Damage Potential
A	High	High	High	High
B	High	Low	High	Low-Medium
C	High	High	Low	Medium-High

impact of 0.33, was grouped into service level SL_2. Confidentiality degrades and reaches an irreparable state, as we traverse from SL_0 to SL_x. Similarly for integrity, there are two sets of impacts as well - 0.14 and 0.33. For availability, we have three sets of impact - 0.14, 0.33 and 0.50. The service for all three WSN security parameters is summarized in Table 1.

Table 8. Service Levels for WSN Security Parameters

Service levels	Attacks	Attack Pattern
SL0(0.0)	-	-
SL1(0.14)	Node Subversion, Spoofing, Node Replication, Malware attack, Wormhole, Selective Forwarding	C; I
SL2(0.33)	Eavesdropping, Sybil, Selective Forwarding, Spoofing, Alter/Replay, Acknowledgment Spoofing, Node Malfunction	C; I
SL3(0.50)	Frequency Jamming, Denial of Service	A
SLx(1.0)	-	-

3.4.2. Computing state transition rates. We compute the rate transition matrix using MF estimates (Table 4 and (3)) once we have created the service levels. The transition rates for the service level are illustrated in Table 9. We assume that a transition from a higher service level to a lower service level is not feasible. Also, we assume that a network cannot reach SL_x directly from SL_0 or SL_1, since, SL_0 is a fully operational level with no harmful attacks. Furthermore, execution of attacks in SL_1 will result in a transition to the next service level (SL_2 and not SL_x since, the impact of attacks in SL_1 is lower that of attacks in SL_2. The network would be functioning in SL_0 in the absence of attacks.

Execution of an attack belonging to SL_1, causes a traversal from SL_0 to SL_1 and so forth. The transition from SL_1 to SL_2 is dependent on the transition rate for SL_0 to SL_1 and is computed as MF(SL1SL2)| MF(SL0SL1).

Given a time frame of 30 days, a WSN in absence of security measures will have its confidentiality fully compromised in about 13 days. Since, the probability of full compromise, reaching SL_x, is around 0.44, which translates to 13 days (44 % of 30 days). We also conclude that in such a WSN, there is 66.27% chance that the data will be fully compromised. Integrity, in close co-relation to confidentiality, will also be lost in 13 days. Since, the probability of full compromise, i.e. reaching SL_x is around 0.44, which translates to 13 days (44 % of 30 days). This result makes sense because a network cannot maintain its integrity after confidentiality is fully compromised. Whereas for availability, there is a 20% chance that a WSN will face communication disruption beyond the point of recovery. Since, probability of full compromise (SL_x) is around 0.20. Thus, a complete degradation might occur in about 6 days (20 % of 30 days) if precautionary measures are not taken to protect the WSN. The rate transition matrix computation for all three WSN security parameters is summarized in Table 9.

Table 9. Rate Transition Matrix for WSN Security Parameters

	SL0	SL1	SL2	SL3	SLx
SL0	0	$(0.83)_{C,I,A}$	$(0.81)_{C,I}$ $(0.77)_A$	$(0.83)_A$	0
SL1	0	$(0.83)_A$	$(0.66)_{C,I}$ $(0.64)_A$	$(0.69)_A$	0
SL2	0	0	0	$(0.44)_A$	$(0.44)_{C,I}$
SL3	0	0	0	0	$(0.20)_A$
SLx	0	0	0	0	0

3.5. Complexity Analysis and Scalability. In this section, we will discuss the complexity involved in designing the proposed risk assessment framework and its scalability with respect to large scale sensor clouds.

The initial steps involves the creation of a database which contains the information on different WSN attacks regarding their attack patterns, pre- and post-conditions, and JOIN type in the attack graph. Creation of attack graphs requires extracting this information from the database. This process involves considering each attack attribute as a root node and traversing the remaining attack attributes in database and determining if the pre-condition of the attack attribute being considered as the root node matches the post-condition of the remaining attack attributes in the database.

We perform this step three times, once for each of the three WSN security parameters. Hence, this process is upper bounded by $O(n^2)$, where n being the number of WSN attacks taken into account. Creation of attack graphs is followed by scoring the attack nodes in the graph with their probability of success which takes a constant amount of time. We then compute the net threat level of the root node using the concepts of the Bayesian networks, computational complexity of which is upper bounded by constant time. Time frame estimations involves the creation of service levels from the Misuse Impacts and then computing the transition matrix from the Misuse frequency values, which is again upper bounded by a constant time. Hence the total computational complexity of our proposed risk assessment framework is squared.

The framework creates three attack graphs for a given WSN network. As such if N is the total number of WSNs in the sensor cloud, the total number of attack graphs that will be generated is $3N$. For larger values of N, the number of attack graphs that needs to generated and evaluated increases rapidly. Although this increase is linear and the generation of the attack graphs is not dependent on either the number of sensor nodes or the number of WSNs present in the sensor cloud. Hence, the proposed risk assessment can scale with respect to the increase in number of WSNs in the sensor cloud. However, a challenge lies with the security administrator to draw inferences from net threat level values generated by the attack graph and is something we will address as a future work.

4. VALIDATION

In this section we will study the impact of various attacks on deployed WSN (Section 3.1) in a sensor cloud environment in order to determine the effectiveness of our proposed risk assessment framework.

4.1. Initial Security Measures. Section 3.2 depicted the attack graphs for a deployed WSNs in the absence of security measures. This illustrated the full potential of the attack graphs since none of the attack nodes had their probability of success as zero. But for validation purposes we used a set of initial security measures for the deployed WSNs.

The implemented sensor cloud was supported with proxy re-encryption scheme. Although the encryption service is optional and depends on a user's preference. Thus, data coming from a WSN serving their application may or may not be encrypted. Our current sensor cloud architecture also supports ten different frequency bands for communication. The sensor node of a WSN is programmed with a unique frequency band during the pre-deployment phase to prevent collisions while communicating. However, we can reprogram these nodes with a backup frequency band. Thus, in case of a Denial of Service attack, if the primary frequency band is jammed the nodes can switch to the backup band to send out a distress signal, alleviating the effects of a DoS attack. We consider this as a partial solution to a DoS attack. Additionally, Node malfunction at times can result due to the lack of sensor node battery power. The sensor nodes having low battery might not be able to correctly perform its operations (sensing) and send incorrect or infrequent data. Our sensor cloud architecture has the capability of powering up the nodes using Ethernet cables and USB connectivity instead of the traditional AA batteries. Thus, if a deployed WSN is in close vicinity of a USB power source we consider this capability as a solution to attacks that rely on depleting a sensor node's energy. These are available to the WSN deployed on the second floor of the Missouri S&T computer science department.

Further, the five deployed WSN are sparse. This fact is relevant since the density of a WSN belonging to a sensor cloud determines the feasibility of attacks like node capture. Hence, if the deployment of a WSN is dense, two or more sensor nodes might have redundant sensing region, as such if a node is captured it will be harder to detect it than in a sparse network. Although attacks like node capture also depend on how easily the deployed networks can be discovered (nodes deployed for health monitoring can be easily discovered in contrast to the nodes deployed to track wildlife) and as such are susceptible to physical node capture. Finally, we have also assumed that addition of new nodes to existing WSN requires authentication.

4.2. Attack Models. We incorporate three attack scenarios on the deployed WSNs belonging to the sensor cloud. Each scenario is targeted to exploit a given WSN security parameter.

4.2.1. Attack model 1. The attacker in this scenario is an insider to the sensor cloud. As such they might be aware of the encryption keys and deployment regions. Users selecting encrypted way of data delivery still might be at risk from this kind of attacker. Additionally, the attacker might also be aware of authentication procedures and might add rogue nodes as legitimate nodes to the existing WSNs. The main motive here is to compromise data. To avoid disclosing their identity by getting caught, they will refrain from using attacks that will disrupt the availability of the network or cause inconsistencies in data received by legitimate users. The straightforward way to achieve this would be by adding a rogue node or subverting an existing one. As such the unconditional probability of node subversion becomes 1.0. As a result, further exploitation of attacks such as Sybil or wormhole eases up.

4.2.2. Attack model 2. The attacker in this scenario does not have the same capability as the previous attack model. As such, they will not be able to compromise data. However, they can corrupt the data by causing data injection attacks and will try to subvert the nodes not by physical capture (deployment region is not known) but by sending

malicious codes, which if executed will subvert the nodes and give them the capability of the attacker discussed in attack model 1 (escalation). In this context, the probability of success of False Data Injection will be 1.0. Additionally with respect to escalation, the attacker will also try to execute malware attack, making its probability of success as 1. This malware attack might lead to node malfunction and not node subversion. Escalations are a possibility, but the attacker does not have the required resources or expertise to cause it.

4.2.3. Attack model 3. The attacker here is not concerned about the network data. Their objective is to disrupt available services and as such jam communication frequencies, overload sensor nodes with spurious message packets and so on. In such a scenario, the most tempting attack will be to destroy the sensor nodes. But if the topology cannot be discovered, the next available option will be to jam the network communications. This can be achieved by jamming the wireless communication frequency around the approximate vicinity of the WSN. Additionally Desynchronization attacks can also be initiated through a malware attack, causing a node to malfunction and not be able to establish communication with neighboring nodes and transfer data.

4.3. Results and Observations. We performed risk assessment for the deployed WSNs, given the initial security measures as discussed in Section 4.1. After computing the net threat levels to the WSN security parameters, we simulated attacks according to the attack models described in Section 4.2. Given these attacks we utilize our risk assessment framework to re-compute the observed net threat levels. Once we have the estimated and observed net threat levels, we compare and contrast the results to evaluate the effectiveness of our proposed risk assessment framework.

4.3.1. Estimated net threat level. In this section, we assess the net threat level to the WSN security parameters, estimated during the pre-deployment phase of the WSNs under the sensor cloud. The probability of success for the attack node, physical tampering, is assumed to be zero. This is because the deployed WSNs are sparse and as such absence

of a node will be detected and addressed before the attack can be successfully executed. Similarly, given the ethernet-usb power sources for the sensor nodes, the probability of success of the attack node, energy drain attacks, is also assumed to be zero.

Confidentiality can be exploited from four attack nodes (Figure 5). If each of these attack nodes contributes equally in the exploitation of confidentiality, there is a 25% chance of exploitation from each node. But eavesdropping, can be suppressed if the user opts for the available encryption service for their application. Given the uncertainty in a user's encryption service selection, exploitation coming from successful execution of eavesdropping is not taken into account for expected threat level. Additionally, absence of physical tampering will reduce the probability of successful execution of node replication attack. This will reduce the expected threat level further (5% approximately, using the concepts of forward propagation in Bayesian networks). Hence, the expected threat level for confidentiality is estimated to be about 70%.

Similarly, integrity can be exploited from 5 different attack nodes (Figure 6). Considering equal contribution from each of these attack nodes, there is a 20% chance of exploitation towards integrity coming from these nodes. But the probability of success of one of the child nodes of the contributing attack node is assumed to be zero (energy drain attacks). This reduces the contribution of that node (Node Malfunction) by 10%. Hence, the expected threat level to integrity is estimated to be 90%. This zero probability of success for energy drain attacks will also be reflected in the attack graph for availability as well. Availability can be exploited by three attack nodes (Figure 7), approximately 33% contribution from each node. But the attack node, Desynchronization, has node malfunction as its child node, which in turn is the parent node of energy drain attacks. Hence, the expected threat level to availability was estimated to be 88%.

Given these expected threat levels, the estimated net threat level to confidentiality, integrity and availability was 56.14%, 74.95%, and 86.82% respectively. Computations performed were similar to the ones discussed in Section 3.2.

4.3.2. Observed net threat level. Dynamic risk assessment (Section 2.3.2) is used to compute the observed net threat level. These estimations are performed after the evidence of a successful attack execution. As such, the probability of success of attack nodes which has been successfully exploited becomes one and the net threat level to the root node of the attack graph (confidentiality, integrity or availability) for WSNs are re-computed.

The simulated attacks are based on the attack models described in Section 4.2. The observed attacks from attack model 1 are node subversion, spoofing, and Sybil. The observed net threat level due to which is estimated to be 45.19%. Observed attacks from attack model 2 is malware attack and data injection attack. The observed net threat level due to which is estimated to be approximately 72.00%. Similarly, the observed attacks from attack model 3 was frequency jamming. The estimated observed net threat level owing to this was approximately 28.38%. Each of these observed attack states are highlighted in Figure 5, 6, 7.

4.3.3. Analysis of the results. The estimations of the risk assessment framework can be utilized to allocate resources to reduce the net threat level during the deployment phase of the WSNs. As such, higher the value of the estimated net threat level, greater the amount of security resources that will be deployed. For example, if the nature of applications being hosted by the sensor cloud becomes sensitive (military applications), then using encryption becomes a necessity instead of an option. Similarly, the observed net threat level values can be used to evaluate whether or not the invested security resources is sufficient. If it is not then what all resources the security administrator must levy to reduce the observed net threat level and re-deploy the WSNs using the updated security resources. This allocation of resources can be traced back to the metrics used for computing the probability of success of the attack nodes in the attack graph. Variation of these metrics will help in identifying the resources that needs to be invested and their target instance

in the network. For example, whether the number of authentication instances needs to be increased or whether the way in which the network interacts with the outside domain needs to be made more secure and so on.

In the experiments performed, we observe that the estimated net threat levels for confidentiality and integrity were able to encompass the observed net threat levels due the attacks that were directed to exploit them from attack model 1 and 2. This is owing to the fact that the measures to exploit and counter confidentiality and integrity is bounded. There are some set ways to exploit these two WSN security parameters and safeguarding these ways reduces the chances of successful exploitation. Although, there was an over investment of security resources. For confidentiality, the over investment was 19.5% and that for integrity was 3.93%. But in contrast to confidentiality and integrity, although the estimated net threat level for availability was able to encompass the observed net threat level, the invested resources were under utilized to a large extent - 67.31%. This is because even though there was an observed instance of frequency jamming, it did not affect the entire sensor cloud network. There was partial disruption to one of the WSNs. But according to the proposed risk assessment framework, the unconditional probability of an attack node is taken to be 1.0 if it has been observed. This is a challenge that needs to be addressed in our framework with respect to assessing availability as it is not able to identify the extent of exploitation and help in re-allocating security resources accordingly. For example, if an observed attack phenomena has affected the sensor cloud network partially, the risk assessment framework will re-compute the net threat level considering that the observed attack has affected the entire network. This area will be addressed as a part of future work.

5. RELATED WORK

The scope of attacks sustained by a WSN has been surveyed and discussed in [28] [25] [19]. The authors have assessed well-known sets of WSN attacks along with their countermeasures. They were, however, oblivious about the attack's impact on a network

and efficiency of the countermeasures. Walter [27] in his survey on security issues of a WSN established the parameters based on which security of a WSN is characterized. These parameters were confidentiality, integrity and availability. We are able to design the attack patterns from this information for our work and analyze the attacks on a broader perspective. Analysis of various attacks adopted by the adversary to exploit security parameters and ways in which they could be averted were also discussed in [27]. Although, it did not address the likelihood of exploitation of an attack.

Wood [29] and Xu [30] gives exposition on the omnipresent denial of service (DoS) attack. DoS are not only hard to predict but also to counter. This helped us in understanding the nature of jamming attacks in WSNs. The absence of predictability and correlation with other attacks in case of DoS attack, is a drawback on the security administrator's part. Karlof [11], Kannhavong [9] and Newsome [17] gives an in depth analysis on routing layer attacks and the Sybil attack respectively. However, these attacks can be exploited by successful execution of attacks in different network layers. For this purpose we should identify the interdependencies between different feasible attacks. Mauw [15] and Phillips [20] demonstrated this kind of logical relationship via attack graphs or trees. Using the principles from the work of Lee [12], we were able to assess the risks to a network. But the drawback was that they were for a wired network scenario. Sheyner [24] discussed the various types of attack graph and models. This contributed immensely towards the development of the attack graph model for WSN. Gallon [6] devised the methods to quantitatively assess the attack nodes in the attack graph; although they were not meant for a WSN. National vulnerability database established the vectors to calculate the severity ratings of vulnerabilities in a wired network. Using the same principles, we calculated the severity ratings for the attacks on WSN. Frigault [5] gave insight on implementing attack graphs as a Bayesian network. This gave us a better understanding about the adversary's capability, likelihood and impact of attacks for various attack scenarios. Dantu [3] and Liu [13] analyzed attacks by assigning probability values to the attack graph nodes.

Furthermore, using the concepts of Bayesian networks on these probability values, they calculated potential attack paths and modeled network vulnerabilities. Nevertheless, these computations were not for an attack scenario of a WSN and as such could not be applied to the proposed attack graphs for a WSN. Houmb [7] proposed the methodologies of risk level estimations using the exploitation frequency and impact of vulnerabilities in a wired network. We adopted these concepts to identify the metrics necessary to compute net threat level to the root node of our attack graph when it is represented as a Bayesian network. This gave a degree of diversification and uniqueness to the WSNs with respect to quantitatively analyzing our attack graphs and using the results to estimate maintenance period for the largely unattended WSNs.

6. CONCLUSION AND FUTURE WORK

In this paper, we have presented a risk assessment framework for WSNs in a sensor cloud environment. We depicted the cause-consequence relationship for attacks on WSNs using attack graphs and perform quantitative assessment by representing them as Bayesian networks. Thus, we are able to compute the net threat level to WSN security parameters - confidentiality, integrity, availability and develop time frames estimating the degradation of these WSN security parameters.

Static risk assessment helps in identifying critical resources in the network. This information can be utilized to determine the effective placement of a intrusion detection system (IDS) to monitor these resources in the network. Further, static risk assessment takes into account the logical relationship between different attacks. Using this as a supplement to current IDS will benefit them in their monitoring tasks by not having to wait till the attack is taking place to raise an alarm. The proposed risk assessment will also be used to determine how efficient a security measure will be, which can be measured in terms of resource utilization and the capability to reduce the overall threat level to WSN security parameters.

We will use this as the basis to estimate the net threat level of overlay WSN formulated by using individual sensor nodes from different existing WSN in the sensor cloud. Since a substantial amount of overlay networks could be formed given the combination of sensor nodes that are available, the current risk assessment framework will take the aid statistical analysis to help the security administrator better analyze the obtained results. These facets will be taken into account as a part of our future work.

REFERENCES

[1] Nessus vulnerability scanner v6.0, 2014.

[2] S. Barnum. Attack patterns: Knowing your enemy in order to defeat them. *BlackHat DC, cigital (www.cigital.com)*, 2007.

[3] Ram Dantu, Kall Loper, and Prakash Kolan. Risk management using behavior based attack graphs. In *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04) Volume 2 - Volume 2*, ITCC '04, pages 445–, Washington, DC, USA, 2004. IEEE Computer Society.

[4] J Dawkins, C Campbell, and J Hale. Modeling network attacks: Extending the attack tree paradigm. In *Workshop on Statistical and Machine Learning Techniques in Computer Intrusion Detection*, pages 75–86, 2002.

[5] Marcel Frigault and Lingyu Wang. Measuring network security using bayesian network-based attack graphs. In *Proceedings of the 2008 32nd Annual IEEE International Computer Software and Applications Conference*, COMPSAC '08, pages 698–703, Washington, DC, USA, 2008. IEEE Computer Society.

[6] Laurent Gallon and Jean Jacques Bascou. Using cvss in attack graphs. In *Proceedings of the 2011 Sixth International Conference on Availability, Reliability and Security*, ARES '11, pages 59–66, Washington, DC, USA, 2011. IEEE Computer Society.

[7] S.H. Houmb and V. Nunes Leal Franqueira. Estimating toe risk level using cvss. In *Proceedings of the Fourth International Conference on Availability, Reliability and Security (ARES 2009 The International Dependability Conference)*, IEEE Conference Proceedings, pages 718–725, Los Alamitos, March 2009. IEEE Computer Society Press.

[8] Chen Wang K. Pongaliur and Li Xiao. Maintaining functional module integrity in sensor networks. *IEEE International Conference on Mobile Adhoc and Sensor Systems Conference*, 0:806, 2005.

[9] B. Kannhavong, H. Nakayama, Y. Nemoto, N. Kato, and A. Jamalipour. A survey of routing attacks in mobile ad hoc networks. *Wireless Communications, IEEE*, 14(5):85–91, 2007.

[10] A. Kapadia, S. Myers, XiaoFeng Wang, and G. Fox. Toward securing sensor clouds. In *Collaboration Technologies and Systems (CTS), 2011 International Conference on*, pages 280–289, 2011.

[11] Chris Karlof and David Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. In *In First IEEE International Workshop on Sensor Network Protocols and Applications*, pages 113–127, 2002.

[12] Jehyun Lee, Heejo Lee, and Hoh Peter In. Scalable attack graph for risk assessment. In *Proceedings of the 23rd international conference on Information Networking*, ICOIN'09, pages 78–82, Piscataway, NJ, USA, 2009. IEEE Press.

[13] Yu Liu and Hong Man. Network vulnerability assessment using bayesian networks. 5812:61–71, 2005.

[14] Sanjay Madria, Vimal Kumar, and Rashmi Dalvi. Sensor cloud: A cloud of virtual sensors. *IEEE Software*, 31(2):70–77, 2014.

[15] Sjouke Mauw and Martijn Oostdijk. Foundations of attack trees. In *ICISC'05*, pages 186–198, 2005.

[16] Peter Mell, Karen Scarfone, and Sasha Romanosky. *A Complete Guide to the Common Vulnerability Scoring System Version 2.0*. NIST and Carnegie Mellon University, 1 edition, June 2007.

[17] J. Newsome, E. Shi, D. Song, and A. Perrig. The sybil attack in sensor networks: analysis defenses. In *Information Processing in Sensor Networks, 2004. IPSN 2004. Third International Symposium on*, pages 259–268, 2004.

[18] E.C.-H. Ngai, Jiangchuan Liu, and M.R. Lyu. On the intruder detection for sinkhole attack in wireless sensor networks. In *Communications, 2006. ICC '06. IEEE International Conference on*, volume 8, pages 3383–3389, 2006.

[19] G. Padmavathi and D. Shanmugapriya. A survey of attacks, security mechanisms and challenges in wireless sensor networks. *CoRR*, abs/0909.0576, 2009.

[20] Cynthia Phillips. A graph-based system for network-vulnerability analysis. In *in Proceedings of the 1998 workshop on New security paradigms*, pages 71–79. ACM Press, 1998.

[21] Nayot Poolsappasit, Rinku Dewri, and Indrajit Ray. Dynamic security risk management using bayesian attack graphs. *IEEE Trans. Dependable Secur. Comput.*, 9(1):61–74, January 2012.

[22] Nayot Poolsappasit, Vimal Kumar, Sanjay Madria, and Sriram Chellappan. Challenges in secure sensor-cloud computing. In *Proceedings of the 8th VLDB international conference on Secure data management*, SDM'11, pages 70–84, Berlin, Heidelberg, 2011. Springer-Verlag.

[23] Indrajit Ray and Nayot Poolsapassit. Using attack trees to identify malicious attacks from authorized insiders. In *Proceedings of the 10th European conference on Research in Computer Security*, ESORICS'05, pages 231–246, Berlin, Heidelberg, 2005. Springer-Verlag.

[24] Oleg Sheyner and Jeannette Wing. Tools for generating and analyzing attack graphs. In *proceedings of formal methods for components and objects, lecture notes in computer science*, pages 344–371, 2004.

[25] E. Shi and A. Perrig. Designing secure sensor networks. *Wireless Communications, IEEE*, 11(6):38–43, 2004.

[26] Aven Terje. Trends in quantitative risk assessments. volume 5, pages 447–461, October 2009.

[27] John Paul Walters, Zhengqiang Liang, Weisong Shi, and Vipin Chaudhary. Wireless sensor network security: A survey,âĂİ in book chapter of security. In *in Distributed, Grid, and Pervasive Computing, Yang Xiao (Eds*, pages 0–849. CRC Press, 2007.

[28] Yong Wang, Garhan Attebury, and Byrav Ramamurthy. A survey of security issues in wireless sensor networks. *IEEE Communications Surveys Tutorials*, 8:2–23, 2006.

[29] A. Wood and J.A. Stankovic. Denial of service in sensor networks. *Computer*, 35(10):54–62, 2002.

[30] Wenyuan Xu, Ke Ma, Wade Trappe, and Yanyong Zhang. Jamming sensor networks: attack and defense strategies. *IEEE Network*, 20(3):41–47, 2006.

II. OFFLINE RISK ASSESSMENT OF CLOUD SERVICE PROVIDER

Sanjay Madria and Amartya Sen
Department of Computer Science
Missouri University of Science and Technology
Rolla, Missouri 65409–0050
Email: {madrias,asrp6}@mst.edu

ABSTRACT

The acceptance of cloud as an infrastructure to host applications has been a growing trend in the recent times. Facilitating and hosting applications on the cloud reduces the support and maintenance cost. However, the security concerns of these applications on the cloud is one of the primary reasons which makes organization avoid complete adoption of cloud services. Despite the fact that cloud provides standard security, they do not address it with respect to application's security requirements. Without clear understanding of the degree of security provided with respect to the scope of applications, organizations have been cautious about migrating their application onto cloud platforms. In this paper, we propose an off-line risk assessment framework to evaluate the security provided by a cloud service provider from the perspective of an application to be migrated on it. Once the most secure cloud service provider is identified for a given application, the proposed framework will perform a cost-benefit tradeoff analysis in terms of security dispensed and service costs to gauge an ideal cloud migration plan.

Keywords: Risk Assessment, Cloud Computing

1. INTRODUCTION

Cloud computing is a viable solution for applications which depend on scalability and reliability for an uninterrupted service. Cloud services in the form of Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS) and Infrastructure-as-a-Service (IaaS) cuts down application maintenance and development costs dramatically. However migration of applications on to the cloud platform gives rise to security issues as applications are no longer within the secure domains of its organization. The security threats present varies from one application to another and as such the promise of generalized security measures provided by cloud service providers (CSP) may not be sufficient to alleviate the security concerns. Hence, the security solutions provided by a CSP is like a big black box to its clients.

Currently the means to assess security of a CSP is by going through their security white papers, SLA agreement, or tendor notes. An organization can also employ the services of a third party to do the same. Guidelines established by the authorities like ENISA [8] or CSA [5] would help an individual assess the security of a CSP. Additionally, standards such as PCI-DSS [2], HIPAA [3] lays down guidelines for CSP hosting application related to specific fields. The drawbacks of these current measures is twofolds. First, a third party or self assessment depends on the keenness and expertise of the evaluator which may or may not be up-to-date with security standards laid down by a CSP. Secondly, such evaluations are not from the perspective of security requirements of an application to be migrated onto the cloud. A security measure present in a CSP may be applicable to one application, but it may not be sufficient for another. For example, a cloud service provider hosting e-health data needs to be HIPAA compliant like FireHost's cloud platform with the inclusion of explicit security measures like the one discussed in [9].

Thus, a better and an accurate way to assess a CSP's security will be to do it in contrast to the security threats present in an application. To do so, one must be able to identify the vulnerabilities that are present in an application.

Thereafter to evaluate if a CSP addresses it using the available security measures. There are several means to identify vulnerabilities in an application. Tenable [14] has the tools to check an application for security issues. But this can be done once the application has been deployed. As such any repairs at this point would be costly. We need to assess the security vulnerabilities in an application during the system design phase. In this regard, threat modeling tools like EMC's Developer Driven Threat Modeling [7] and Microsoft's STRIDE [15] are helpful. Application threat modeling as discussed by Open Web Application Security Project (OWASP) [13] gives some useful guidance in vulnerability detection and secure application development. It helps to identify, quantify and address the security risks associated with an application.

In this paper, we propose a suitable and dependable framework for assessing the security provided by a CSP from the perspective of an application to be migrated on it. Our framework will help in determining, given the risks that are present, whether or not it would be cost effective to perform cloud migration. The applicability of our framework is before the CSP selection phase and during the system design phase. It will help organizations identify the threats associated with the functional elements of their applications. This will be interpreted as the security requirements of their application. Given these security requirements, an organization will be able to determine the extent of security that a CSP will dispense for their applications.

Additionally, security measures, cloud platform infrastructure and safeguards differs from one CSP to another. The techniques used by CSPs to address and mitigate threats on their cloud platform changes with time and without prior knowledge of consumers. If an application is hosted by a federation of CSPs, there could be every possibility of certain incompatibility issues amongst the different CSPs. Therefore, the *presence* or *absence* of security solutions on the cloud platform is no longer sufficient to assess the security of an application on the cloud.

Our proposed framework will aggregate all security measures of various CSP and keep it up-to-date, taking into account the interaction between federation of CSP, while performing security assessment for an application to be migrated onto the cloud platform.

2. PROPOSED METHOD

The off-line risk assessment framework is composed of three modules viz. - Mission Oriented Risk Assessment, Cloud Service Provider Security Assessment and Cloud Adoption Strategies. A flow chart summarizing the off-line risk assessment framework is given in Figure 1. The objective of this work is to evaluate the security capabilities of

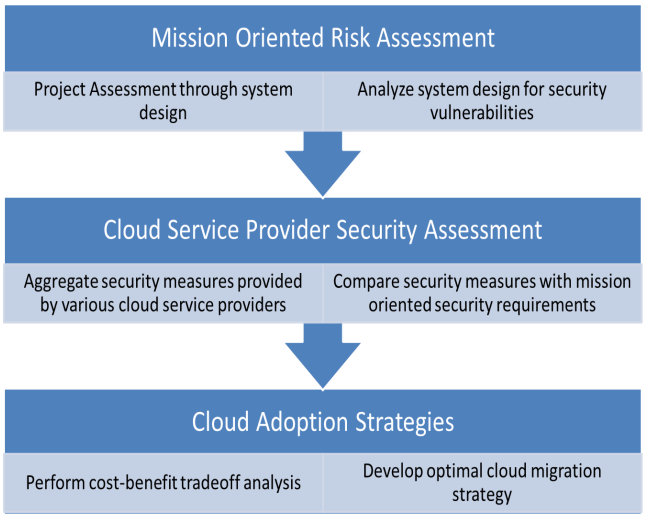

Figure 1. Proposed Off-line Risk Assessment Framework

various CSPs based on the security requirements of a client's application. This information will be then used to develop an optimal cloud migration strategy. In the following sections, we will discuss each module of the proposed framework in detail.

2.1. Mission Oriented Risk Assessment. The Mission Oriented Risk Assessment module is responsible for identifying the security threats that are present in an application and create the client's security requirement that has to be addressed by a CSP. It consists of two parts: (1) Project Assessment, (2) Risk Assessment.

2.1.1. Project assessment. Project Assessment helps in design and analysis of an application that aids in carrying out risk assessment during the system design phase. To perform project assessment, system design in the form of Data Flow Diagrams (DFD) [10] is given as an input to the framework. In this context, there can be two scenarios:
Scenario 1: An organization has already developed their application. They have their DFDs ready and is now considering cloud migration.
Scenario 2: An organization is yet to conceive their application. The DFDs are not ready but the organization is considering cloud migration once their application is developed.

In the first scenario, our framework will gather an application's goals and objectives by analyzing their DFDs. For the second scenario, our framework can assist in the creation of mission ontology in the form of Context Flow Diagrams (CFD) and DFD. Once the mission ontology is available, the information is passed through a vulnerability detection process. For this purpose, we use the concepts of Microsoft's STRIDE [15] in our framework. STRIDE can analyze a system architecture and identify vulnerabilities and threats associated with the functional elements of a system (elucidated by DFD). Hence our framework can identify the exact locations of threats in an application along with the required mitigation measures. Thus, we can not only identify the threats but also where they are present. Impact of these threats are then computed by ranking them. Ranking helps in determining the high impact risks and prioritize their mitigation. It is more beneficial to integrate risk assessment to the process of system design since mitigating faults in the system design phase is more economical than after its deployment (Applicable only to Scenario 2).

2.1.2. Risk assessment of system design using STRIDE. This is applicable only to Scenario 2 discussed in Section 2.1.1. Mission ontologies and system designs are generated in the form of several DFDs. Data flow diagrams gives an overview of system functionalities; the data flow among the different system components by showing entry and exit data points as well as where data gets processed. DFDs are easier to understand by technical and non-technical audiences alike [10].

Hence, for the purpose of system design generation and analysis, we have considered DFD of an application as an input to our framework. The mission ontologies are then passed through Security Development Lifecycle (SDL) and Threat Modeling tool [12]. SDL conceptualizes STRIDE to identify the threat related to the elements of a DFD.

Analyzing System Design for Vulnerabilities. STRIDE is an acronym for the different categories of vulnerabilities [15] that may be present in an application. It stands for: **S**poofing, **T**ampering, **R**epudiation, **I**nformation disclosure, **D**enial of service, **E**levation of privilege. For each element in a DFD, STRIDE gives a set of feasible vulnerabilities. This set however, is not application dependent but rather element dependent. For example, consider the DFD of an Online Movie Streaming Application shown in Figure 2.

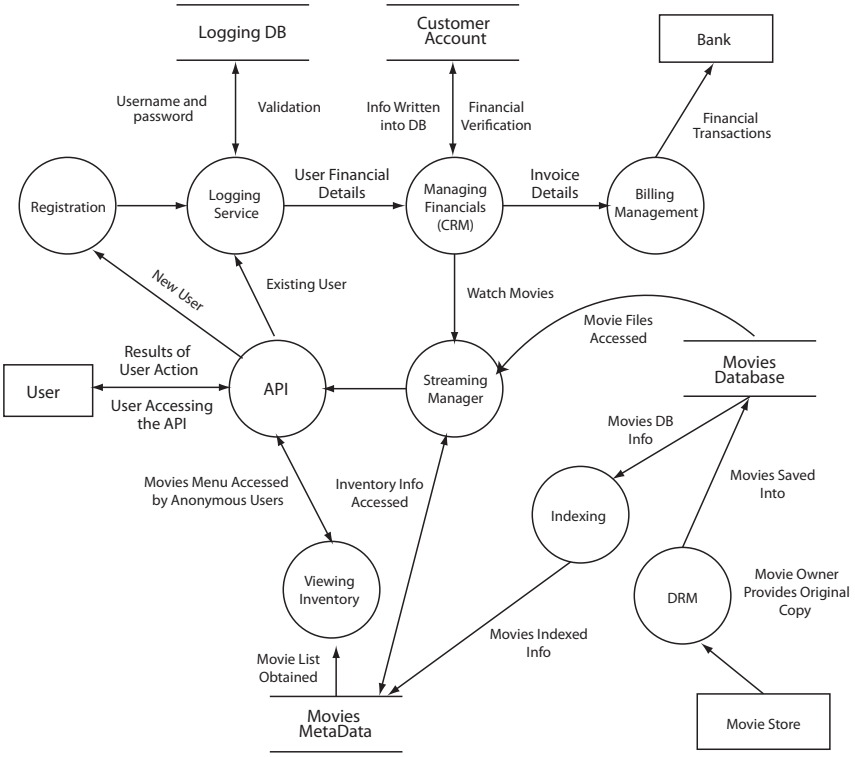

Figure 2. DFD for Online Movie Streaming Service

It was drawn in Microsoft SDL. For *process* elements in a DFD, STRIDE generates five possible sets of vulnerabilities (*spoofing, tampering, repudiation, information disclosure, denial of service and elevation of privilege*).

Whereas for *data store* elements it generates four possible sets of vulnerabilities (*tampering, information disclosure, repudiation and denial of service*). Depending on the type of application, these set of vulnerabilities may or may not be applicable. For instance in the online movie streaming service, the sets of possible vulnerabilities that STRIDE generates for "Customer Account" data store are "*tampering, information disclosure, repudiation and denial of service*". However, in this domain, "*repudiation*" would not be feasible since all transactions within "Customer Account" are authenticated and logged using a trusted third party - "Bank", hence it will be omitted. Similarly, considering "Movies Metadata" data store, the data in this store is made available for public consumption. Therefore "*information disclosure*" will not be applicable. Similarly, the possible set of vulnerabilities is narrowed down to the ones that are applicable. This process is manual as it requires human intervention (system designer) to understand the functionality of the elements of an application.

Identifying Attacks. Once the set of possible vulnerabilities in the system components(s) are narrowed down, our framework proceeds to identify the attacks that can exploit these vulnerabilities. In order to do so, we use MITRE's Common Attack Pattern Enumeration and Classification (CAPEC) [4] which is an attack database containing information on conditions that are required to execute an attack, its mitigation measures, and impacts due to successful execution. CAPEC's database, available in XML format, is imported into our framework. We map the set of STRIDE vulnerabilities to the attack pattern categories of CAPEC. The mapping is illustrated in Table. 1.

Additionally, each CAPEC attack pattern category has sub-attack pattern categories. We depict this relationship using a tree structure (Figure 3). To find an attack that can exploit an identified vulnerability, our framework finds the attack pattern category that the vulnerability maps onto (Figure 1). This attack pattern category then becomes the root node of a tree. Any attack pattern category or attack that has the root node as its parent (information provided in imported XML file) will be grouped under it.

Table 1. Mapping STRIDE Vulnerabilities to CAPEC Attack Pattern Categories

STRIDE Vulnerability	CAPEC Attack Pattern Category
Spoofing	Spoofing
Tampering	Data Structure Attacks, Injection, Remote Code Inclusion
Repudiation	Attack categories of Spoofing and Tampering
Information Disclosure	Data Leakage Attacks, Path Traversal, Functionality Misuse
Denial of Service	Resource Depletion Attacks
Elevation of Privilege	Exploitation of Authentication, Exploitation of Privilege or Trust, Privilege of Escalation

The tree structure is populated in this way; for attacks which are not the parent of any other attack become the leaf nodes. Our framework traverses down from the root to leaf level, registering feasible attacks and deleting attacks which are infeasible due to the presence of initial security measures in the application. This gives a specific set of attacks related to each functional element of the system. For example, our framework would not just output that a *process* is vulnerable to *spoofing attack*, but it will give the specific instantiation of spoofing attack pattern category (Phising, Man-in-the-middle, etc.). Similarly for rest of the vulnerability types our framework traverses through the CAPEC database and identifies attacks that will exploit the identified vulnerabilities. While registering attacks, our framework will also import their proposed security mitigations as provided by CAPEC. These mitigation strategies will form the basis of client's security requirements. This process is semi-automated. In events of absence or lack of information about initial security measures present, our framework will ask the user a set of heuristically developed yes/no questions in order to ascertain whether a registered attack is feasible.

Ranking Identified Attacks. Once the relevant attacks have been identified our framework ranks their impact on the application. For the purpose of ranking, we use DREAD [15] ranking system. Other ranking techniques such as Attack Surface Metrics [11] can also be used which identifies the threats in an application's executable code.

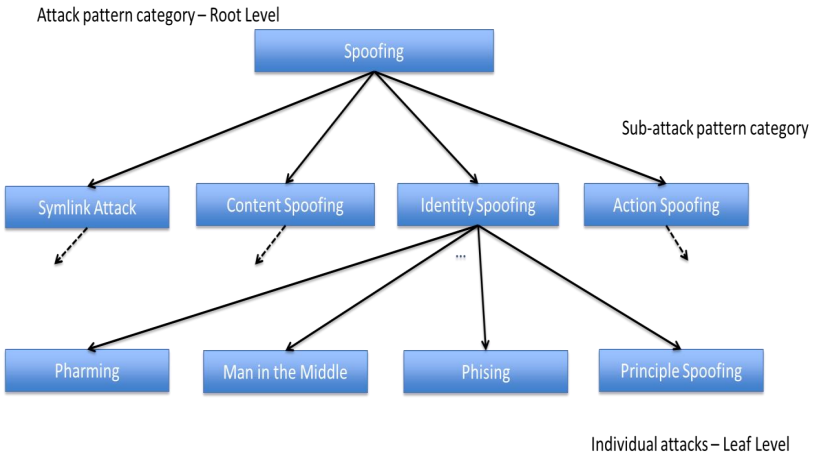

Figure 3. Tree Structure of CAPEC DB

Thereafter the identified threats are ranked based on their damage potential. However, it requires the executable code of an application and as such cannot be utilized during the system design phase. DREAD, a logical extension to STRIDE analysis, is an acronym and stands for,

- Damage (D): Impact of the attack,
- Reproducibility (R): How easy is it to reproduce the attack,
- Exploitability (E): How complex is the execution of the attack,
- Affected users (A): How many people will the attack impact, and
- Discoverability (Di): How easy is it to discover the threat.

Parameters such as "Affected users" depends on an organization and hence, is subjective in nature. For rest of the attributes, CAPEC's attack description quantifies them as high, medium or low. Depending on the scale an organization will choose, 0-3 or 0-10 or 0-100, they can quantify high, medium and low. For example, if the organization opts for the scale 0-100, they can quantify "low" as [0-30], (30-70] as "medium" and "high" as (70-100]. It is evident that this type of quantization is subjective in nature.

Since, it depends on the scale an organization will choose and how they will interpret high, medium and low. In this regard, our framework uses the scale of 0-10. To find the final ranking of an attack, our framework sums up the scores of each DREAD attribute and normalizes it so as to keep the final ranking in between 0 and 10. For example, if we have the following scores for the DREAD attributes of a registered attack X,

$(D : 8; R : 10; E : 6; A : 6; Di : 5)$

Its rank will be given as,

$Rank(X) = ((D + R + E + A + Di)/5)$

$Rank(X) = 7$

In this way the rank of all registered attacks are computed. Once the ranks are obtained, they are sorted in decreasing order. Based on a threshold, chosen by an organization, security mitigation measures are exported for those registered attacks whose ranks are above the set threshold. This becomes the client's security requirement and input to module 2 of our framework, cloud service provider security assessment.

2.2. Cloud Service Provider Security Assessment. Cloud service providers mostly use a low-touch self-service model to implement SaaS, PaaS and IaaS. These self-service models are economical with respect to renting the services provided by the cloud platform. Nonetheless it impedes effective evaluation of cloud's infrastructure security.

Additionally most details regarding cloud platform security are presented in a way that are obscure to the clients. Otherwise they are cajoled within the facts of advertisement. As such, a security evaluator has to carefully extract the relevant security information from all such documents regarding security. This module of our framework evaluates various CSP's security measures as per the client's security requirements.

We address facts like given a set of client's security requirements, how much security coverage is guaranteed by a CSP. Our framework evaluates a client's security requirements as described in Section 2.1. To perform security assessment of a CSP, our framework collects public information of different CSPs regarding their security measures.

These public information are available in the form of tendor notes, security white papers and other third-party statistical report. For evaluation purposes, we consider three cloud service providers namely - Amazon, Microsoft and Defense Information Systems Agency's Department of Defense Cloud Broker (DISA: DoD Cloud Broker).

Security evaluations based on public information regarding a CSP's security measures can answer questions like,

1. What kind of security measures does a CSP have?

2. Whether or not a CSP adheres to certain standard compliant like SAS-70 [1], PCI-DSS [2], or HIPAA [3]?

3. What does a CSP guarantee with respect to application security through its Service Level Agreement (SLA)?

Even though such questions may help in assessing a CSP's security, it digresses from the client's security requirements. But in contrast to such traditional means of risk assessment, our proposed framework keeps the client's security requirements in mind. Hence, the security assessment extends from project to cloud infrastructure. Our framework after collecting security information of different CSPs, contrasts it with mission-oriented risk assessment. This is achieved by comparing the resources (conditions) required to execute the registered attacks with that of the resources (conditions) secured by the security measures dispensed by a CSP. If the resources (conditions) required by an attack is safeguarded by a security measure, that particular attack is suppressed by a CSP. We denote this as the security coverage provided by a CSP. A CSP's security measures is mapped onto the prioritized registered attacks (output of module 1). Each mapping is then scored by a risk reduction factor denoted by α_{ij}. For a CSP's security measure, M_i, and an attack, T_j, (M_i,T_j) is a security coverage if M_i reduces the attack execution probability Pr(T_j) with a reduction factor α_{ij}. The concept of security coverage is depicted via Figure 4.

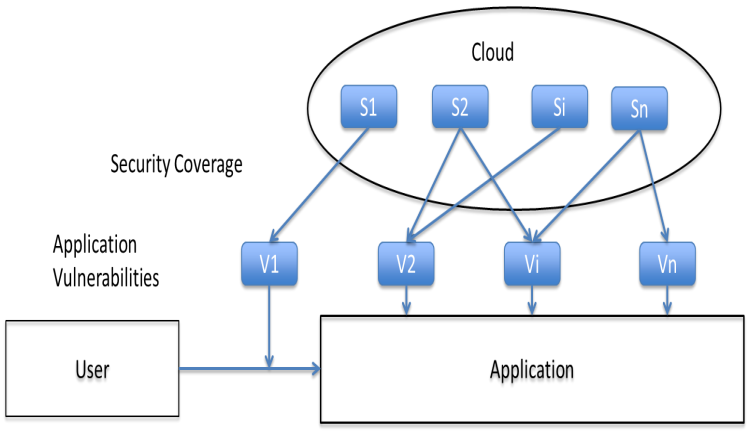

Figure 4. Cloud Service Provider Security Coverage

Cloud security measures, $S_1,...,S_n$, is mapped onto application vulnerabilities, $V_1,...,V_n$. This mapping need not always be one-to-one. It can either be conjunctive or disjunctive in nature. In conjunctive form, a single security threat requires more than one security measure to be suppressed. Whereas in disjunctive form a single security measure can suppress a given threat. For example, we can see that the mapping of V_2 and V_i are conjunctive in nature and that of V_1 and V_n is disjunctive in nature (Figure 4). The reduction in the execution probability of an attack will vary based on the conjunctive or disjunctive nature of coverage. This is elucidated via "risk reduction factor".

Thus, given the security coverage and risk reduction factor of a CSP, we compute its trust based on its capability to reduce the threats present in an application to be hosted on it. The output of this module will then be given as the most trustworthy CSP amongst all the available options to host an application.

2.3. Cloud Adoption Strategies. Module 2 of our framework gives the security coverage provided by the most trustworthy CSP. However, there are possibilities that these security coverages might not envelop all the threats present in an application. As such the security coverages that are not available needs to be imposed by the organization.

To accomplish such a task requires a cost-benefit tradeoff analysis. The analysis will determine which cloud-based solution will be economical in terms of providing required service and ensuring complete application security.

This module of our framework computes a CSP's service cost and security control cost incurred by the client to impose their security requirements, also known as *client's share of responsibilities*. Given the migration of all or few of the functional elements of an application, module 3 estimates the security coverage provided by a CSP for the threats present in the functional elements. The threats which are not covered by a CSP's security coverage identifies the security controls that an organization needs to implement. It then estimates the total cost (cloud service cost and security control implementation cost) due to cloud migration. Based on such an analysis it develops a suitable cloud migration strategy - complete, partial or none. In other words, migrating all functional element of the application onto the cloud platform, migrating some functional element of the application onto the cloud platform, or not migrating any functional element of the application onto the cloud platform.

To this extent, mission ontology is considered as a graph G=(V,E). V is the set of nodes representing the system elements. E is the set of edges representing the functional dependencies. A series of what-if analysis will be performed on this graph, assessing the consequences of outsourcing a group of vertices onto the cloud platform. A cloud adoption plan will consist of varying combination of more than one vertices. Then for each cloud adoption plan formed, cloud migration is considered. Security coverage for the threats present in each cloud adoption plan is assessed. The threats which are secured by the cloud is known as *provider's share of security responsibility*. We then compute *client's share of responsibility* by identifying the cost of implementing the security controls for the threats that are not present in the security coverage provided by the cloud for that particular cloud adoption plan.

Therefore, given a cloud adoption plan $\overrightarrow{P}$, the cost assessment is computed as,

$$PlanAssessment(\overrightarrow{P}) = VenServ(\overrightarrow{R}) \oplus CliCst(\neg\overrightarrow{R}) \quad (1)$$

where $VenServ(\overrightarrow{R})$ denotes provider's service and security coverage cost and $CliCst(\neg\overrightarrow{R})$ denotes client cost for implementing the security measures of threats not covered in the security coverage of the cloud.

Cost-benefit tradeoff analysis is performed in a similar way for rest of the developed cloud adoption plan. Organization will choose a plan that minimizes $VenServ(\overrightarrow{R}) \oplus CliCst(\neg\overrightarrow{R})$. But chances are, the provider cost and client cost can be of different units. To address this issue of multi-objective problem, the concept of Pareto-optimality [6] can be used.

3. CONCLUSIONS

The security provided by cloud service providers has not been addressed adequately in terms of the security requirement of an application to be migrated on to it. As such organizations have been uptight in completely migrating their applications to the cloud platform. In this paper, we have presented an off-line risk assessment framework for evaluating the security of various cloud service providers (CSP) as per the security requirements of an application to be migrated on to it. This gives an organization a better understanding about the security of their application on the cloud platform rather than generically evaluating the security of a CSP. Our framework helps in developing (if used during software development phase) and identifying the vulnerabilities associated with an application. It then evaluates the security of different available CSPs by contrasting it with the application's security requirements. Once the most suitable (secure) CSP is identified, the framework performs a cost-benefit tradeoff analysis.

This is done to determine the most optimal cloud migration plan. In other words, to identify the application's components whose migration onto the cloud platform will be beneficial both in terms of cost and security.

Currently, we are building the tool by implementing the off-line risk assessment framework and validating the obtained results. The work will then be extended to the risk assessment domain when the application is running on the cloud platform; online risk assessment of cloud service providers. The online risk assessment of cloud service providers will take into account the security issues arising due to multi-tenancy and interoperability.

REFERENCES

[1] Au section 324 service organizations: Source sas no. 70; 78; 88; 98. www.aipca.org/Research/Standards/AuditAttest/DownloadableDocuments/AU-00324.pdf.

[2] Payment card industry (pci) data security standard, v3.0. PCI Security Standards Council, LLC. 2013. https://www.pcisecuritystandards.org/documents/PCI_DSS_v3.pdf

[3] The health insurance portability and accountability act of 1996 (hippa) privacy and security rules. www.hhs.gov/ocr/privacy/hippa/administrative/privacyrule/adminsimpregtext.pdf, 2006.

[4] S. Barnum. Common attack pattern enumeration and classification (capec) schema description. Cigital, Inc, Tech. Rep, 2008.

[5] G. Brunette and R. Mogull. Security guidance for critical areas of focus in cloud computing v2.1. *Cloud Security Alliance*, 2009.

[6] Rinku Dewri, Indrajit Ray, Nayot Poolsappasit, and Darrell Whitley. Optimal security hardening on attack tree models of networks: a cost-benefit analysis. *Int. J. Inf. Sec.*, 11(3):167–188, 2012.

[7] Danny Dhillon. Developer-driven threat modeling: Lessons learned in the trenches. *IEEE Security & Privacy*, 9:41–47, 2011.

[8] European Network and Information Security Agency (ENISA). Cloud computing: Benefits, risks and recommendations for information security, 2009.

[9] Jiankun Hu, Hsiao-Hwa Chen, and Ting-Wei Hou. A hybrid public key infrastructure solution (hpki) for hipaa privacy/security regulations. *Special Issue on Information and Communications Security, Privacy and Trust: Standards and Regulations*. Computer Standards & Interfaces, Elsevier, volume 32, pages 274–280, 2010.

[10] Jr. Le Vie, Donald S. Understanding data flow diagrams. *Society for Technical Communications (STC)*, 2000.

[11] Pratyusa K. Manadhata and Jeannette M. Wing. An attack surface metric. *IEEE Transactions on Software Engineering*, 37(3):371–386, 2011.

[12] C. Mockel and A.E. Abdallah. Threat modeling approaches and tools for securing architectural designs of an e-banking application. In *Information Assurance and Security (IAS), 2010 Sixth International Conference on*, pages 149–154, Aug 2010.

[13] Marco M. Morana. Managing software security risks using application threat modeling. *IMI Security Symposium and Expo*, 2008.

[14] Steve Piper. Definitive guide to next generation vulnerability management. CyberEdge Group LLC, 2013.

[15] Adam Shostack. Experiences threat modeling at microsoft. Modeling Security Workshop, Toulouse, 2008.

III. SENSITIVITY ANALYSIS OF A CLOUD MIGRATION FRAMEWORK FOR OFFLINE RISK ASSESSMENT OF CSPS

Amartya Sen and Sanjay Madria

Department of Computer Science

Missouri University of Science and Technology

Rolla, Missouri 65409–0050

Email: {asrp6,madrias}@mst.edu

ABSTRACT

Organizations find it beneficial to host their applications on the public cloud platform because it reduces infrastructure ownership as well as maintenance cost. However, they are skeptical about completely hosting their applications on these cloud platforms due to security concerns. The security assured by cloud providers is not specific to the requirements of user applications hosted on these cloud platforms. To address this important issue, we proposed an off-line risk assessment framework for cloud service providers. The proposed framework assesses the security provided by a cloud service provider with respect to the security risks present in an organization's application to be hosted on the selected cloud platform. In this paper, we present a cloud migration strategy for our proposed off-line risk assessment framework to perform a comprehensive sensitivity analyses of the factors that play a vital role in cloud migration. We do so by modeling our cloud migration strategy as a multi-objective optimization problem and applying it to a cloud migration use case scenario. We discuss in details the relationship between different conflicting cloud migration parameters like cost, application security, migration constraints, and scalability along with a performance evaluation study to compare the impact of different evolutionary algorithms to model a cloud migration framework.

Keywords: Cloud Migration, Sensitivity analyses, Multi-objective optimization, Genetic Algorithm, Risk Assessment

1. INTRODUCTION

Cloud computing platforms can host clients' applications helping in reducing ownership and maintenance liabilities by not having to acquire hardware, software, and facilities to operate their applications. This benefit clients as they can reduce their revenue expenditures by hassle free use of publicly available cloud platforms. Clients can also use cloud services according to their pay-as-you-use model which makes scaling up and down relatively easier in comparison to a private cloud hosting. However, these benefits are not devoid of challenges and risks since cloud migration moves a client's application from the secure boundaries of their private network to more or less an untrusted domain. Therefore, security is a primary concern that prevent clients from fully migrating to the cloud.

The process of cloud migration involves moving a client's application and associated data to the cloud platform of different available cloud service providers (CSPs). In doing so, several objectives are to be considered, most notable of them being cost and security. The frameworks presented in [16] [13] [22] [15] address the cost component of cloud migration, whereas other frameworks like [2] [17] [8] which have discussed aspects related to application and cloud platform security. Additionally, other works have introduced non-financial facets like technical/legal concerns [16] [27], or legacy applications [9] which play a critical role in determining the migration of a client's application to a cloud platform. The decision making process of cloud migration involves several such parameters, and it becomes more challenging as some of these vital parameters are conflicting in nature, for e.g. strengthening the security of an application hosted on the cloud platform by implementing additional security measures will increase the overall cost. Similarly, consider an instance when a part of an organization's application is not migrated to the cloud platform either due to legal or technical issues.

In such instance, organizations are responsible for hosting, maintaining, and securing the partial application on their private networks. They will also have to address any associated scalability issues. Given such a scenario, the question is, if it will be beneficial for organizations to migrate and address their legal or technical issues, or should they host their partial application on their private networks. Further, organizations also need to consider if they should host their entire application on a single cloud service provider (CSP) or distribute their application on multiple CSPs. To be able answer such challenging questions, requires us to understand the relationships that exists between these different parameters involved in the decision making process of performing cloud migration. As such, there is a need to perform a comprehensive sensitivity analyses.

Along this direction, we develop a holistic cloud migration framework built on top of our previously proposed off-line risk assessment framework [20]. The security of hosted applications on the public cloud platform is given high priority. Notable works like [7] address the challenges of performing security risk assessment for the cloud platform by assessing the SLA presented by clients and evaluating from the CSP's perspective the number of SLA violations that might happen based on the demands presented by the clients in their SLA. Similarly, authors in [4] present a risk assessment framework of CSPs which utilizes information gathered from clients and cloud vendors to assess different risk scenarios. They do so by proposing a machine learning framework which leverages the security evaluation documents of various CSPs publicly available on Cloud Security Alliance Security, Trust and Assurance Registry to output quantitative ratings describing the risks associated with different cloud services. However, such evaluations are not from the perspective of the security requirements of an application to be migrated onto the cloud. A security measure present in a CSP may be applicable to one application, but it may not be sufficient for another. This is addressed via the off-line risk assessment framework by introducing the concepts of *security coverage, total risk reduction, and share of security responsibilities*.

We incorporate these novel concepts to design the security model of our cloud migration framework, and adopt other parameters like cost, migration feasibility from existing cloud migration frameworks [16] [13] [9]. Additionally, we have introduced a new concept of *fault tolerance*, which will allow clients to group parts of their application with similar functionality and host them separately, thereby ensuring that the entire application is not compromised in the event of an attack belonging to a specific attack pattern.

Overall, the sensitivity analyses of our cloud migration framework is modeled as an multi-objective optimization problem with the help of Non-Dominated Sorting Algorithm - II (NSGA-II) [6]. Although there are many ways to solve a multi-objective optimization problems [23] [5] [25], the selection of NSGA-II was based on our study of other evolutionary algorithms (Section 5). The presented results in this regard will be beneficial to extend our proposed cloud migration framework using other evolutionary algorithms for different cloud migration scenarios and design. Our contributions in this work are as follows:

- Performed an extensive sensitivity analyses of the vital parameters involved in cloud migration by proposing a cloud migration framework built on top of the offline risk assessment framework of CSPs (Section 2).
- Introduced a novel security model for cloud migration which helps in evaluating the security of different CSPs as per the security risks present in an application.
- Introduced a new concept of *fault tolerance* in performing cloud migration allowing clients to specify grouping of parts of their application with similar functionality.
- Presented the performance evaluation of different multi-objective evolutionary algorithms to model our cloud migration framework and statistically validating our selection of NSGA-II, and provided insights into modeling our framework with other evolutionary algorithms (Section 5).

2. PROPOSED CLOUD MIGRATION FRAMEWORK

The most cost-efficient CSP might not be the ideal host for a client's application while performing cloud migration. Similarly, the most secure CSP might not be the best option for all or some of the application's elements. Since, with the guarantee of the desired security, if the migration costs more than what it would cost to host an application privately then the migration is not that fruitful. But another contrasting scenario could be that a client might consider to pay a high cost if the security dispensed is more than that of the private domain and alleviates the responsibilities of maintaining the security patches. Hence, an ideal CSP for hosting a client's application needs to provide the desired security at the cost which a client is willing to pay. In such cases where multiple objectives are involved in a cloud migration process, decision making is not a trivial task and some cost-benefit tradeoff analysis needs to be performed.

In our proposed cloud migration framework, we have modeled the cost-benefit tradeoff analysis as a multi-objective optimization problem to solve multiple conflicting objectives [5]. In our case, these objectives are reducing security risks, costs incurred by clients, and the number of application elements not hosted on the cloud platform. Solving such problems requires to find a set of decision variables satisfying the constraints established by the problem and concurrently optimizing a vector function, $\overrightarrow{x}$ representing all the involved objectives. The evaluation of two or more objectives in $\overrightarrow{x}$ produces a solution vector $\overrightarrow{y}$. To compare any two solutions in $\overrightarrow{y}$ (say $\overrightarrow{y_1}$ and $\overrightarrow{y_2}$) a dominance criteria is defined, also known as the *Pareto* criteria as shown in eq. 1. The *Pareto* criteria states that a feasible solution vector, $\overrightarrow{y_1}$, is said to dominate another solution vector, $\overrightarrow{y_2}$ ($\overrightarrow{y_1} < \overrightarrow{y_2}$), if no component of $\overrightarrow{y_2}$ is greater than the corresponding components of $\overrightarrow{y_1}$, and there exists at least one component of $\overrightarrow{y_1}$ which is greater than the components of $\overrightarrow{y_2}$ (eq. 1).

$$\begin{aligned} &1. \quad f_i(\overrightarrow{y_1}) \leq f_i(\overrightarrow{y_2}), \forall i \in \{1, 2, ..., k\} \text{ and,} \\ &2. \quad f_j(\overrightarrow{y_1}) < f_j(\overrightarrow{y_2}), \exists j \in \{1, 2, ..., k\} \end{aligned} \tag{1}$$

In order to solve this multi-objective optimization problem, we have used the Non-dominated Sorting Algorithm - II (NSGA-II) [6] based on our study discussed in Section 5.

2.1. Problem Statement. The objective of our cloud migration framework is to find a cloud migration plan consisting of an application's elements such that it maximizes overall security of the application and minimizes the cost. We formally define a cloud migration plan as follows:

Definition 2.1. Cloud Migration Plan: *Given an application's data flow diagram G = {V,E}, where V is the set of elements in the data flow diagram,*

$$V = \{v_1, v_2, ..., v_N\} \tag{2}$$

for a total of N elements, and E is a $|V| \times |V|$ adjacency matrix representing edges between these elements,

$$E_{ij} = \begin{cases} 1, \exists v_i \rightarrow v_j \\ 0, Otherwise \end{cases} \tag{3}$$

A cloud migration plan can be defined as a vector, $\overrightarrow{P_j}$,

$$\overrightarrow{P_j} = \{v_1, v_2, ..., v_q\} \tag{4}$$

where, $1 \leq q \leq N$ and N is the total elements in the data flow diagram. And $1 \leq j \leq 2^N$, is the maximum possible cloud migration plans that can be formulated. $\overrightarrow{P_j} = true$ if all the elements in $\overrightarrow{P_j}$ are migrated to the cloud platform.

In order to formally establish the problem of cloud migration as a multi-objective optimization problem using NSGA-II, we need to develop our objective functions. For our proposed cloud migration framework, we consider the objectives of minimizing (1) security threats present in the application, (2) overall expenditure of a client, and (3) the number

of application elements that are not migrated to the cloud platform. Along these lines, we devise three models; a *cost* model, a *security* model, and a *migration feasibility* model. We comprehensively discuss these models in the following sections.

2.2. Cost Model. Costs incurred by an organization can be on many different fronts. In terms of cloud migration, it will cost to avail the services of a CSP and for any service negotiations (security related or otherwise) which are not present in the service level agreements. Costs will be incurred by an organization if they choose not to migrate an application on the cloud platform since they have to acquire resources (hardware and software) to operate their application and will also be responsible for securing them. Considering these facets, we categorize the cost that a client might incur as - Client Cost (C_{Client}), Security Control Cost (C_{SCC}), Vendor Service Cost (C_{Vendor}).

The category of *Client Cost* covers all costs that a client will have to bear if they choose to host some of the application's elements by themselves. Such costs encompasses but are not limited to buying - hardware, licensed software, physical location to host the hardware, electricity to operate the hardware, and Internet services [15]. Some of these costs like physical location to host the hardware, licensed software, Internet can be fixed whereas others like, electricity or hardware might vary based on usage or decisions to scale up or down. Additionally, self-hosting will also include labor costs, software and hardware maintenance costs. We have not taken into consideration the labor cost, since we reckon that for either private, cloud or hybrid hosting labor costs will be involved and will be subjective in nature but it is constant for any given organization. Hence, it will not affect the migration scenario. Corrective maintenance cost is assumed to be 20% of software and hardware cost. As such, the cost of self-hosting an application's element is as follows:

$$C_{Client} = \sum_{\forall v \in (1-\overrightarrow{P_j})} C_{H/W} + C_{S/W} + C_M \tag{5}$$

where, $C_{H/W}$ is the cost of all required hardware and physical resources, $C_{S/W}$ is the cost of buying all required licensed software, and C_M is the corrective maintenance cost. Additionally, if some elements of a client's application are legacy systems, there might be certain compatibility issues if these elements are migrated to the cloud platform since the available resources on the cloud platform do not provide support for legacy systems [27]. Thus, if clients consider migrating a legacy system then they have to rewrite the system using resources whose support will be available on the cloud platform. This will have a cost of its own. We formalize this as follows:

Constraint 2.1. Legacy Systems: *If a node $v \in V$, in a cloud migration plan, $\overrightarrow{P_j}$, being considered for cloud migration is a legacy system, then it needs to be rewritten using software or hardware resources provided by the cloud. The cost of rewriting such a legacy system is depicted as follows:*

$$\sum_{i=1}^{q} \lambda_i \times C_{legacy}(v_i) \tag{6}$$

where λ_i is the legacy constraint flag for a node $v_i \in \overrightarrow{P_j}$ and,

$$\lambda_i = \begin{cases} 1, \text{if } \mathrm{v}_i = \text{legacy system} \\ 0, \text{Otherwise} \end{cases} \tag{7}$$

In eq. 6, q is the total number of application elements in $\overrightarrow{P_j}$, and C_{legacy} is the cost of rewriting the legacy system.

Security Control Cost is the cost that clients have to invest in to reduce the security risks present in their application. In this regard, there can be two different kinds of cost. First, if an application's element having a security risk is not migrated to the cloud platform then it has to be patched or suppressed by the clients. Secondly, if an application's element is migrated to the cloud platform but the cloud needs to implement a specific security measure (which is not a part of SLA) to suppress the security risk (and are willing to do so),

then the client has to pay for that extra service. For example, a particular hardware on the cloud platform hosts multiple virtual instances. If a client's application requires a security measure to be incorporated for the entire hardware then the cloud cannot ask the users of other virtual instances to pay for it. Since other user's application might not be vulnerable to such a security risk and do not require the security measure. As such, the cost of the security measure needs to be paid by the client who has requested for it. This scenario is adopted from [11] and can be expressed as follows:

Constraint 2.2. Security Measure Implementation: *The security control cost required to mitigate a security risk in an application's element is to be implemented by the client if the element is not migrated on to the cloud platform. Otherwise, it will be implemented by the cloud.*

$$C_{VenSCC} = \sum_{i=1}^{q} \sum_{\forall T(v_i)} \begin{matrix} \eta_i \cdot (C_{SCC})_{Vendor} \times \\ (1-\eta_i) \cdot (C_{SCC})_{Client} \end{matrix} \tag{8}$$

where for a cloud migration plan consisting of q number of application elements and each element having a set of threats $T(v_i)$, η_i is the security share flag for v_i given by:

$$\eta_i = \begin{cases} 1, v_i \text{ is migrated to cloud} \\ 0, \text{Otherwise} \end{cases} \tag{9}$$

thus, the total security control cost (C_{SCC}) is given by:

$$C_{SCC} = \sum_{\forall v \in (1-\overrightarrow{P_j})} C_{SCC}(v)) + C_{VenSCC} \tag{10}$$

where $C_{SCC}(v)$ is the cost of implementing security measures for those application elements that were not migrated to the cloud and had some associated security risks.

Vendor Service Cost is composed of the costs that a client has to pay in order to avail the services provided by a CSP. These costs are expressed in terms of the cloud computing service models that will be used by a client and varies for different CSPs. For example, if a client is going to use Infrastructure-as-a-Service (IaaS) [16], the Vendor Service Cost is expressed in terms of number of virtual instances used, number of cores used, storage devices required, their type and billing option. An instantiation of such cost option can be found on Amazon EC2 pricing page.

Additionally, we identify scaling up as an aspect that will attribute towards the vendor service cost. The size by which the system might expand will increase the size and number of required instances and feasibility of private hosting. Thus, the cost associated with respect to scalability is as follows:

Constraint 2.3. Cost of Scalability: *For any node of an application, $v \in \overrightarrow{P_j}$, such that $\overrightarrow{P_j} = true$, has a factor of scalability associated with it, then it adds to the vendor cost as follows:*

$$C_{Vendor} = \sum_{i=1}^{q} C_{Vendor}(v_i) \times \sigma \tag{11}$$

where σ ($\sigma \geq 1$) is a factor for scalability and will be determined by an organization based on their policies. Thus, the total cost (TC) for a cloud migration plan ($\overrightarrow{P_j}$) is:

$$TC(\overrightarrow{P_j}) = C_{Client} + C_{SCC} + C_{Vendor} \tag{12}$$

2.3. Security Model. Security provided by various CSPs can be evaluated by different methods like security level agreements [10], generating knowledge bases to assess threats confronted by Cloud assets [14], or developing security service level agreements (SecSLAs) and individually analyzing the trust of different Cloud providers [8]. These methods are able to identify the security threats and their counter measures present on a CSP but does not answer how relevant they will be to a particular client's application.

To do so, we extend the notions of security evaluation from these works and incorporate it with our previously proposed offline risk assessment framework [20] to output the ranked set of security risks that a client needs to mitigate when migrating their application to the Cloud. In this regard, we introduce and define the concepts of *security coverage, total risk reduction,* and *share of security responsibility* to build our proposed cloud migration framework's security model.

Definition 2.2. Security Coverage: *Let T be a set of threat and M be a set of cloud security measures. Security coverage, SC, is defined as a mapping* $SC : M \times T \longrightarrow \{0,1\}$. *Then, given a security measure,* $m_l \in M$, *and a threat,* $t_h \in T$, m_l *is a security coverage of threat* t_h, $(m_l,t_h) \in SC$, *if probability of exploiting threat* t_h, $Pr(t_h)$, *is reduced by applying* m_l, *i.e.* $Pr(t_h \mid m_l) = \alpha_{lh} * Pr(t_h)$ *with a reduction factor* $0 \leq \alpha_{lh} < 1.0$

The above definition states that if a security measure employed by a CSP suppresses a security risk in a client's application then that particular CSP is providing security coverage to the client's application. The security coverage provided by a CSP will lead to reduction of the security risks present in an application. We express this in terms of total risk reduction $R(t_h)$ as follows:

Definition 2.3. Total Risk Reduction: *Given a threat* $t_h \in T$ *and a security coverage SC, total risk reduction* $R(t_h)$ *is measured as:*

$$R(t_h) = \begin{cases} \prod\limits_{\forall m_l \in M \mid (m_l,t_h) \in SC} (\alpha_{lh}), \textit{if } t_{lh} \textit{ is disjunctive} \\ 1 - \left(\prod\limits_{\substack{\forall m_l \in M \mid \\ (m_l,t_h) \in SC}} (1 - \alpha_{lh}) \right), \textit{otherwise} \end{cases} \tag{13}$$

Security coverage along with the total risk reduction gives clients an idea on how secure their application will be if migrated to a particular CSP. Along these lines, the client expects that if an application element is migrated to a CSP then it should provide security for that element. We define this using *Share of Security Responsibility* in Definition 2.4.

Definition 2.4. Share of Security Responsibility: *Given a cloud adoption plan $\overrightarrow{P_j}$, and a set of threats T, let Θ represent a set of chosen application elements to be migrated on the cloud platform according to the plan $\overrightarrow{P_j}$. We define provider's share of security responsibility as:*

$$\sum_{\theta \in \Theta} \sum_{t_h(\theta) \in T} Pr(t_h(\theta)) \tag{14}$$

where θ is an application element in set Θ and $t_h(\theta)$ is the threat(s) associated with that element.

Thus, the client's share of security responsibility, $R^{'}$, is equal to the aggregated impact of security threats present in application elements which do not belong in Θ,

$$R^{'} = \sum_{\substack{(\delta \in V) \\ \wedge (\delta \notin \Theta)}} \sum_{t_h(\delta) \in T} Pr(t_h(\delta)) \tag{15}$$

where δ is an application element in the set of elements V. The objective while performing cloud migration will thus be to minimize a client's share of security responsibilities, $R^{'}$, for a given cloud migration plan, $\overrightarrow{P_j}$. The value of $R^{'}$ is based on the security rating quantization scale selected by an organization and is subjective in nature. Generally, the rating scales can vary as [0,1], [0,10], or [0,100]. Based on the adopted rating scale, security rating of high, medium, and low will be quantized appropriately. For example, if the rating scale is [0,10], low threat can be interpreted as [0,3], medium as (3,7], and high as (7,10]. Hence, for a security scale of [0,10] if probability of exploitation of a threat ($Pr(t)$) is say, 0.56, then it will be scored as 5.6 (0.56×10).

2.4. Migration Feasibility Model. In this model, we assess any restrictions clients have while performing cloud migration. First, if migration of certain elements like data store, results in a violation of legal or organizational policies like privacy of the users then it might not be feasible to migrate such elements beyond the domain of an organization's private network. For example, some CSPs have their data centers located across various

countries where the IT laws are different. Some of it might require the CSPs to share a copy of their data with government agencies. Secondly, if a particular application element cannot leave the trust boundaries of a client's private network as a result of performing some private operations or hosting sensitive data, then it cannot be considered for cloud migration. We depict these constraints as follows:

Constraint 2.4. Migration Constraint: *An application's element, $v \in V$, can be considered for cloud migration if and only if,*

$$\underset{1 \leq i \leq q}{\forall} v_i \in \overrightarrow{P_j}, \quad \tau_{v_i} \wedge \gamma_{v_i} = false \tag{16}$$

where τ_{v_i} depicts whether or not an application element, v_i, has any organizational trust issues related to cloud migration. And, γ_{v_i} depicts whether or not there could be any legal issues if v_i is migrated to the cloud platform. As such, minimizing the number of application elements that are not migrated to the cloud platform can be expressed as:

$$Mig(\overrightarrow{P_j}) = v_N - v_q \tag{17}$$

where v_N is the total number of application elements and v_q is the total number of application's elements present in a cloud migration plan, $\overrightarrow{P_j}$.

2.5. Problem Formalization. We can now formalize the cloud migration framework as a multi-objective optimization problem:

For an application's data flow diagram depicted as a graph $G = \{V, E\}$, devise a cloud migration plan, $\overrightarrow{P_j}$ composed of elements of the application such that,

$$\textbf{minimize:} \qquad TC(\overrightarrow{P_j}), \quad R^{'}(\overrightarrow{P_j}), \quad Mig(\overrightarrow{P_j}) \tag{18}$$

subject to:

$$
\begin{aligned}
& TC(\overrightarrow{P_j}) \leq TC(threshold) \\
& R^{'}(\overrightarrow{P_j}) \leq \overrightarrow{R^{'}}(threshold) \\
& \sum_{i=1}^{q} \lambda_i \times C_{legacy}(v_i) \\
& \sum_{i=1}^{q} \sum_{\forall T(v_i)} \eta_i \cdot (C_{SCC})_{Vendor} \times (1-\eta_i) \cdot (C_{SCC})_{Client} \\
& C_{Vendor} = \sum_{i=1}^{q} C_{Vendor}(v_i) \times \sigma \\
& \underset{1 \leq i \leq q}{\forall} v_i \in \overrightarrow{P_j}, \tau_{v_i} \wedge \gamma_{v_i} = false
\end{aligned}
\tag{19}
$$

3. CLOUD MIGRATION SCENARIOS AND ALGORITHM DESIGN

Our proposed cloud migration framework will determine which of the elements of an application is most suitable for hosting on a cloud platform taking into account the aspects of cost and security. In this regard, many different scenarios can be feasible. The two broad categories that are applicable is as follows; (1) *Non-Federated Migration*: The most ideal elements migrated on a selected cloud service provider's platform and the rest on a client's private network. (2) *Federated Migration*: The most ideal elements migrated across a federation of cloud service providers and the rest on a client's private network.

Under both of these categories, we follow the concept of hybrid hosting and in the best case, the entire application may be hosted on a CSP or a federation of CSPs. Additionally, our proposed framework also considers the concept of *fault-tolerance*. We elaborate on the aspect of fault-tolerance for our scenario using the following example. Consider a simplistic web application requiring web servers, backend servers and database. All of these elements can be hosted on a single large physical (virtual) instance on the cloud platform. Although in the event of a hardware failure, the entire functionality of the application might become unavailable.

Alternatively, one can host the same application using a small instance and two medium instances. In such a deployment, hardware failures might only result in a partial breakdown of the application's functionality. Hence, we introduce a level of fault-tolerance by hosting the application elements on separate physical (private hosting) or virtual (cloud hosting) instance based on their functionalities. Thus, we extend the previously stated cloud migration scenarios as follows; (1) Non-Federated Migration without fault-tolerance. (2) Non-Federated Migration with fault-tolerance. (3) Federation Migration with fault-tolerance.

The capability to specify the extent of fault-tolerance is available to the users via an input field, *clustering label*. All application elements belonging to the same clustering label will be hosted on the same physical (virtual) instance. We further discuss these concepts using the algorithms for some of the major functionalities of our framework which we have divided into two broad categories; Initialization (Algorithm 1) and Migration Computations (Algorithm 2).

Algorithm 1 Cloud Migration Framework - Driver Method

Require: Set of Application Elements, *MigrationConst* flags, *UniqueClusters* in V
1: **for all** $v_i \in V$ **do**
2: **if** (*MigrationConst* == 1) **then**
3: Mig-Const-Flag += 1
4: **end if**
5: **end for**
6: **for all** $v_i \in V$ **do**
7: Aggregate Number of *UniqueClusters*
8: **end for**
9: Create Two Arrays
10: PrivateHosting[*UniqueClusters*+1]
11: CloudHosting[*UniqueClusters*+1]
12: **for all** $v_i \in V$ **do**
13: **invoke** MigrationComputations(v_i, $v^i_{Mig-Const-Flag}$)
14: TotalCost +=Temp-Cost
15: TotalResDmg += Temp-ResDmg
16: **end for**
17: *//eq. 18 and 19*
18: Declare *Objectives*(TotalCost, TotalResDmg)
19: Declare *Constraints*(legacy,elasticity,security,migration)
20: **invoke NSGA-II**(*Objectives*, *Constraints*)
21: **return** Cloud migration plans

The initialization of the framework requires users to specify the input in a text file format containing rxw attributes where r denotes the number of framework input (e.g. eq. 10, 11) associated to a single or multiple CSPs under consideration and w denotes

the number of elements in an application's data flow diagram (DFD). Additional inputs are required corresponding to total number of application elements (N), cost ($TC(\overrightarrow{P_j})$, eq. 12) and client's share of security responsibility ($R^{'}$) threshold. The threshold values are specific to an organization and are to be determined based on the organizational policies and requirements. As depicted in Algorithm 1, in the beginning of initialization process, our framework goes through all the specified application elements and stores any migration constraints that might be associated with them in the form of a migration constraint flag (line 1-5, eq. 16). Then it aggregates the total number of unique clustering label information for all the elements (line 6-8). This is followed by a creation of two arrays whose size is determined by the total number of clusters specified as shown in line 10 and 11. These arrays will be utilized to make sure that during the computations our framework takes into account the cost of a cluster only once. After all preliminary information are read and mapped correctly, our framework traverses through every node invoking the migration computation method passing it the node information and migration constraint flag (line 12-13). The migration computation method returns the cost and $R^{'}$ values (line 14-15) which is then incorporated in the objective functions and constraints function (line 18-19). The objectives and constraints are then passed to the MOEA framework's NSGA-II package to return the optimal cloud migration plans to the user (line 20-21).

The algorithm for migration of computation (Algorithm 2) is responsible for computing the net cost and client's share of security responsibility ($R^{'}$, eq. 15) value for each application element. For federated scenario, this is done for multiple CSPs in contrast to a single CSP in non-federated scenario. Cloud hosting computations (line 3-19) begins with aggregating the total cost of hosting an application element on the cloud platform. It starts by checking if the application element is coded using legacy software and if there are any additional costs in re-coding the element using the technology which is supported by the cloud platform (line 4, eq. 6). Our framework then computes the vendor service costs by taking into account the scalability factor (σ, eq. 11) of the application element.

Algorithm 2 MigrationComputations(v_i, $v^i_{Mig-Const-Flag}$)

Require: Set of Application Elements, Cost and Residual Damage of elements, Clustering Labels, Legacy, Security Coverage, and Scalability Flag of elements in V

```
1: Temp-Cost = 0
2: //Hosted on Cloud
3: if (v_i == 1 && v^i_Mig-Const-Flag == 0) then
4:     C-legacy-temp[i] = C-legacy[i]*legacyFlag[i]
5:     if (clustering-label[i] == 1) then
6:         C-vendor-temp = C-vendor[i]*scalability[i]*60
7:     else
8:         if (!CloudHosting[clustering-label[i]]) then
9:             C-vendor-temp = C-vendor[i]*scalability[i]*60
10:            CloudHosting[clustering-label[i]] = true
11:        end if
12:    end if
13:    if (security-coverage[i] == 0) then
14:        C-ven-scc = (C-scc-client-cloud[i]*(1-cloud-SC-flag[i])) + (C-scc-vendor[i]* cloud-SC-flag[i])
15:    else
16:        C-ven-scc = 0
17:    end if
18:    C-cloud[i] = C-vendor-temp[i] + C-legacy-temp[i] + C-ven-scc[i]
19:    Temp-Cost = C-cloud[i]
20: else
21:    Cost-Client[i] = C-scc-client[i]
22:    Temp-ResDmg += Security-Risk-Impact-Level[i]
23:    //For Database Elements
24:    if (clustering-label[i] == 1) then
25:        Cost-Client[i] += C-sw[i] + C-hw[i]
26:    else
27:        //Other Element Clusters
28:        if (!PrivateHosting[clustering-label[i]]) then
29:            Cost-Client += C-sw[i] + C-hw[i]
30:            Temp-Cost = Cost-Client
31:            PrivateHosting[clustering-label[i]] = true
32:        end if
33:    end if
34: end if
35: return (Temp-Cost, Temp-ResDmg)
```

This is done for a period of sixty months (line 6). In doing so, it takes into account any specified clustering label information (line 8). For security implementation costs, the framework will check if security coverage (SC, Definition 2.2) is present for the identified security risks. If security coverage is present, the client does not incur any additional costs (line 16). In private hosting computations (line 20-33), the framework aggregates the cost of software ($C_{S/W}$), hardware ($C_{H/W}$), security implementations (C_{SCC}, eq. 10) for a given node. Security risks in private hosting falls under client's share of security responsibilities ($R^{'}$, eq. 15). Hence, our framework aggregates the $R^{'}$ value, if any. While computing costs, our framework accounts for the differences arising due to specification of clustering labels (line 23-26 and line 28-31). The complexity of our framework algorithm is guided by the number of elements in the DFD being assessed (Algorithm 1, line 1). Every other operation taking place after that takes constant time.

As such, the time complexity of our framework is equivalent to $O(N)$, where N is the number of elements in a DFD. Although, large scale applications might not be constrained to level zero DFD and for an application having DFDs upto level h, the net time complexity of our framework is governed by $O(hN)$.

4. FRAMEWORK EVALUATION

The objective of our cost-benefit tradeoff analysis framework is to find an optimal cloud migration plan for our use-case application depicted in Figure 1. For this purpose, we have considered three different semi-synthetic cloud service providers (CSPs). This is because the publicly available documents about the security measures promised by different available real CSPs are almost same. As such, if we consider real life CSPs, the assessment of our proposed framework will be reduced to finding a cloud service dispensing the minimum cost. Hence, in order to exhibit a variation in the security policies used by different CSPs and to show the complete applicability of our proposed framework, we are considering semi-synthetic CSPs.

Although, we have modeled the cloud service costs according to the structure provided by real CSPs. We believe that if an extensive documentation about the security measures of real CSPs is available, then there will be some variations and the assessment for such scenario will be governed by what we present here. We have used the open source Java framework, MOEA version 2.10's *Executor* method to implement our cloud migration framework using NSGA-II. The Executor method takes as input NSGA-II relevant parameters (Table 2) and the *problem class* to be solved (algorithm 1 and 2). It further invokes other notable methods like *Evaluate* and *Solution* whose input arguments are our framework objectives and constraints (eq. 18, 19.

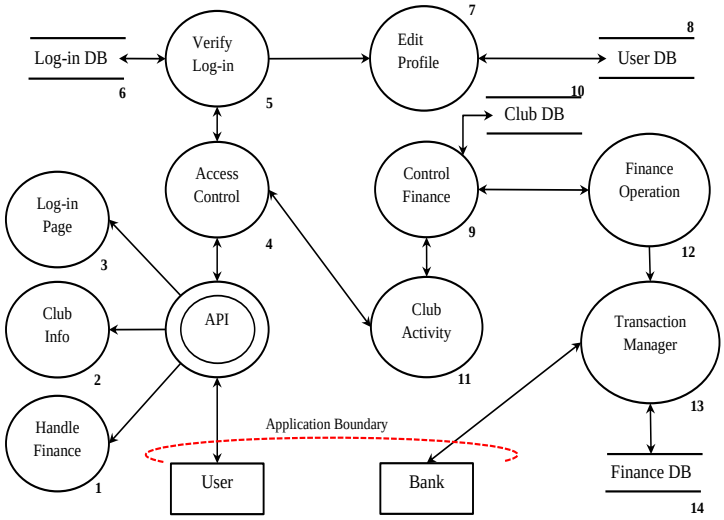

Figure 1. University Club Manager Application

These methods then results in the generation of optimal cloud migration plans (Table 3). MOEA's documentation[3] and our implemented code is available on GitHub[4]. Experiments were run on a machine hosting a Linux operating system having 8GB of RAM and core i5 processor.

4.1. Use-Case Scenario Description. The data flow diagram depicted in Figure 1 will form the basis of our use-case application for which we plan to estimate an optimal cloud migration strategy. The application shown in Figure 1 is a web application for managing and registering club activities of a university. Users can access the API which consists of three functionalities: user log-in, club information, and managing financing operations. Access to the log-in or financing operations page will follow an access control and log-in verification operation. The authentication data is stored in a Log-in database. Once logged in, a user can modify their user profile. This information gets stored in a User database. Further operations after authentication phase might involve updating the activities of a club and/or managing its financing operations.

[3]moeaframework.org
[4]goo.gl/KqTQ5t

The activities related to a club are stored in the Club database. Financing operations are logged in a Finance database and performed with the help of a trusted external entity, Bank. We assume that our university club manager web application will require 10 web servers, 2 middle tier servers and 4 databases. The databases will run Microsoft SQL server, enterprise edition. We also assume that for private hosting the application will require acquiring new software licenses. The application user interface will require high availability (99% and above). The application will not require any disaster recovery. We further assume for private hosting, hardware needs to be acquired; servers, routers, firewall. Storage area (room, electricity, Internet services) are assumed to be already available. Corrective maintenance costs are considered to be 20% of total software and hardware costs [1]. Labor cost is not taken into account in this scenario. The typical hardware refresh rate for an on premise hardware is between 36 months to 60 months. In our case, we have considered it to be 60 months. Since most of the in-house servers will start to lose their ability to adapt to increasing workload after 48 months. The generic IT infrastructure refresh rate is thus about 60 months. The specifications of our private hosting servers and their approximate cost are listed in Table 1.

Table 1. Hardware & Software Specifications for use-case scenario

Type	**Specification**	**Cost**
Server	Dell Precision T7610	$2,969
Router	Linksys LRT224	$178
UPS	APC-BR 1500G	$171
Storage Area Network	Dell PowerVault MD3400	$4409
Rack Cost	Tripp lite SR420B	$930
Load Balancer	Cisco Localdirector 416	$450
Windows Server 2012	2 cores	$700
Microsoft SQL Server 2014	8 core	$4924

In terms of cloud hosting, we have considered three unique CSPs. The service and cost model of these CSPs has been designed in accordance to Amazon EC2, Microsoft Azure, and Rackspace.

Although, due to the absence of a fully detailed publicly available security documentation of these CSPs, we have made some assumptions regarding their security coverage (Section 2.3). This is done in order to depict some diversity of our proposed cloud migration framework.

Hence, we consider our CSPs for the use-case scenario to be semi-synthetic, with a real-world cost and service model and a synthetic security model. The minimum requirements to host our use-case application on the cloud platform will require about 10 small instance for web servers, 2 medium instances for the 2 middle tier servers and 4 databases. Alternatively, all of these can be hosted on a single large instance. This decision is based on the client and is guided by their selection of cloud migration scenario as discussed in Section 3. The input to our framework is specified in text file containing all the detailed cost and security values for the given application and the CSP under consideration. The output of our framework for non-federated cloud migration scenarios (with or without fault-tolerance) will be a binary string. Each bit of this binary string will correspond to an application element in the data flow diagram. The order needs to be pre-determined and for our use-case scenario it is depicted numerically alongside the DFD nodes in Figure 1. A bit value of 1 will correspond to the application element being migrated to the cloud platform and a bit value of 0 will correspond to private hosting. In federated migration scenario, the output will be an integer encoded string with bit values ranging from 0 to C, where 0 to $C-1$ will represent various unique CSP under consideration and C represents private hosting. In the following sections, we present the results of our framework evaluation and sensitivity analysis.

4.2. Use-Case Scenario Evaluation. This section presents the results of our framework applied to our use-case application. We have assumed a cost threshold of $300,000 for a five year period and a client's share of security responsibility ($R^{'}$) threshold of 40 for our use-case application. The $R^{'}$ value is based on our security rating quantization scale, which we have assumed as: [0,10] where a value of zero indicates no security impact, and

ten represents highest security impact. As such, for our use-case application containing fourteen elements (excluding the external entities), the net $R^{'}$ value will fall in the range of [0,140]. Hence, a $R^{'}$ threshold of 40 units elucidates low security risk. This value is subjective in nature and is proportional to the security rating quantization scale.

We estimate cloud migration plans for different migration scenarios as depicted in Section 3. The effect of variation of threshold values and other input parameters on our framework output is shown and discussed in Section 4.3.

Further, our use-case application has been modeled using NSGA-II which is dependent on several parameters like *evaluation number*, *population size*, *mutation* and *crossover* rate. In addition to the search space, these parameters play an important role in obtaining the desired solution set. This is because the parameters (mutation and crossover rate) can take probabilistic values. As such, there are chances that a model output might be sensitive to the values these parameters take. We studied the sensitivity of our cloud migration framework output due to the variation of NSGA-II input parameters. In doing so, we varied the value of one NSGA-II parameter at a time, keeping the rest constant, and analyzed the obtained model output. The population size determines the starting search space. Bigger the population size, better the obtained diversity. Evaluation number dictates the total iteration of searches performed to reach optimality of solution set. Crossover rate determines the creation of child population using the encoding of the previous population set. Mutation rate determines the chances of modifying the encoding of a given individual in the population to generate a new individual. Mutation brings about diversity in the population and has an evident effect on models having lower population size. Although mutation brings about diversity, it prevents the search process from converging to an optimal solution set. Hence, it is not desirable to have a very high mutation rate.

In our framework, we determined that the output is most sensitive to evaluation number. On an average, for all different cloud migration scenarios, after evaluation number passes 1200 mark, the solutions on the pareto front becomes constant.

Further, they are unaffected by the variation of other NSGA-II parameters. The population size is the second most dominating criteria, yielding optimal results between the 175 to 250 mark. After this point, increasing population size had no significant effect on the obtained solutions. We did not observe any significant effect on our framework output due to the remaining NSGA-II parameters. Based on these analyses, we finalized the NSGA-II parameter values as summarized in Table 2.

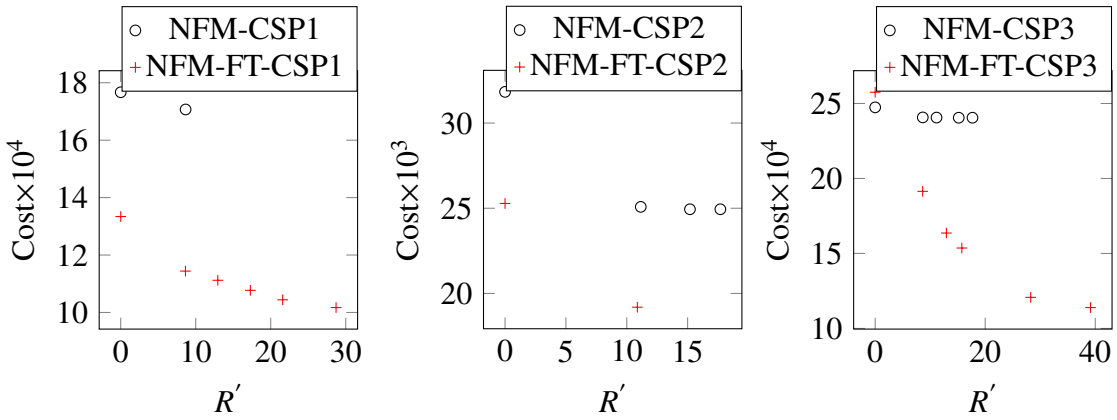

Figure 2. Non-Federated Migration Plans CSP1, CSP2, and CSP3

Table 2. NSGA-II Parameters for Use-case Application Evaluation

Parameter	**Value**	**Parameter**	**Value**
Evaluations	1200	Population Size	250
Crossover Rate	0.6	Mutation Rate	0.1
Crossover Distribution Index	15.00	Mutation Distribution Index	10.00

Figure 2 shows the output of our cloud migration framework for non-federated migration (NFM) with and without fault-tolerance. Figure 3 shows the output for federated migration with fault tolerance (FM-FT). In these figures, the y-axis shows the cost incurred by the clients and x-axis shows $R^{'}$ value of the estimated cloud migration plans. The obtained solution sets depicted in Figure 2 and 3 is also summarized in Table 3.

In the cloud migration plans for the two scenarios (Figure 2 NFM - circle marked and NFM-FT - cross marked), we note that some of the R' values are in close proximities. For example, R' in NFM-CSP1 is 0.00 and 8.65 (first two coordinates) and R' in NFM-FT-CSP1 is 0.00 and 8.65. For a R' value of 0.00, the cloud migration plans in both scenarios is all 1 bit which means all elements of an application are migrated to a cloud platform.

Although, the cost varies in the two scenarios; For a 0.00 R' value in NFM scenario, the migration plan costs $176,750 for a five year period, whereas for NFM-FT scenario it is $133,430 for five years (Table 3). Given our description of fault-tolerance (Section 3), in NFM-FT scenario, we envision the hosting of application elements in a set of small and medium instances according to their functionality, whereas in NFM we try to host all elements in a single large instance. This reduces the cost in NFM-FT scenario, but may introduce other overheads like maintaining these separate small clusters. Additionally, we reckon that such setups may also introduce latency due to transfer of data between clusters of application elements (if they end up being hosted on separate physical instances on the cloud platform). Such overheads are not depicted in terms of its monetary value in Figure 2 and is something a client needs to consider while contrasting between NFM and NFM-FT migration scenarios.

Further, in the NFM-FT scenario the number of solutions obtained is more than in NFM scenario. We can attribute such behavior to the change in cost in NFM-FT scenario computations which is inherently less than that of NFM scenarios. As such, a migration plan in the NFM-FT-CSP1 (Cost - $111,238 and R' - 12.94) having R' value in close proximity (as in the case for R' of 8.63 and 8.65) when projected to NFM scenario, will get dominated by presented output due to an increase in cost. Similar behavior can be seen for migration output of CSP2 and CSP3. On an average, NFM-FT scenario gives cost reductions of at least 20% as compared to the NFM scenario.

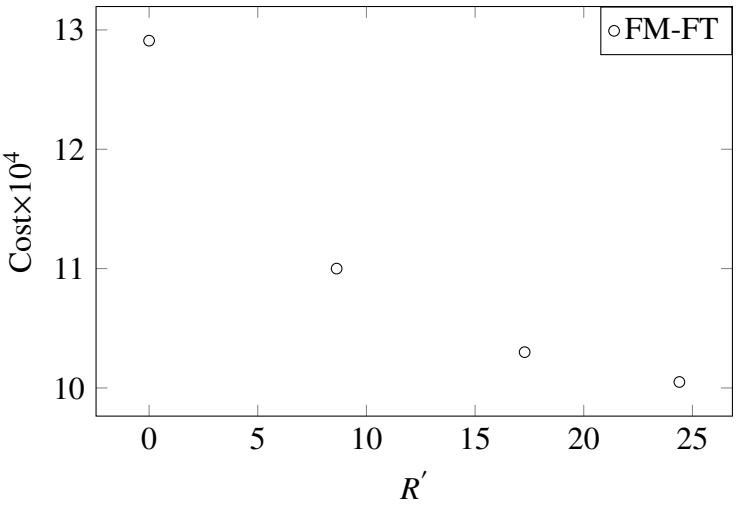

Figure 3. Federated Migration Plans

Figure 3 shows the cloud migration plans for FM-FT migration scenario. Unlike non-federated cloud migration, we can observe a much more uniform spread in the solution set for FM-FT migration scenarios. This is because FM-FT migrations take into consideration the hosting capabilities of all CSPs for every individual application element. This capability along with different available clustering specification due to fault-tolerance option helps in obtaining better solutions. When compared to the solutions obtained in NFM-FT scenario, for a $R^{'}$ value of 0.00, FM-FT gives a cost reduction of approximately 3.6% for CSP1 and 50% for CSP3. In case of CSP2, the costs are considerably lower than that for CSP1 and CSP3 since databases requiring Microsoft services cost much less when hosted in Microsoft Azure. As such, ideally one of the federated cloud migration plans could have been to host the entire application on CSP2 (cost modeled after Microsoft Azure). But given the search design of our framework, the tradeoffs in finding security coverage for application elements and thus emphasizing in reducing the responsibilities of clients to enforce and look after security of the application does not yield such an ideal output. Although, CSP2 for 0.00 $R^{'}$ valued cloud migration plan, is hosting 50% of the application's elements and as such striking a balanced tradeoff between cost and security features.

Another uniform observation across all cloud migration scenarios is that one can nullify R' (client's share of security responsibilities) for the use-case application. Although such an option will cost clients the most with respect to the other feasible migration plans. For example, considering cloud migration plans for NFM-FT-CSP3 scenario (Figure 2, circle marked and Table 3) the cost value of 0.00 R' is \$247,475. For our assumed R' threshold of 40 units, and considering R' of 0.00 to be 100%, the R' of the second plan is approximately 78.5% $(100 - (8.65/40)\%)$, at the expense of a cost reduction of 2.81% $(((\$247,475 - \$240,732)/\$240,732)\%)$.

Similarly, for the next cloud migration plans having R' of 11.15 (72%), 15.19 (62%), and 17.69 (56%), there is not ample cost differences (0.015%, 0.044%, 0.015%) as compared to the first two plans with R' of 0.00 and 8.65. With all the migration plans being pareto optimal, clients can select from any one of them, but certain considerations need to be made when it comes to making a decision to select from the first two plans. Selection of the second migration plan having a R' of 8.65 will save the users a considerable amount of revenue and they can choose to patch the security risks themselves. But, such actions will not account for any additional costs that might apply in future due to security upgrades or new patches. Decisions for such a scenario need to be made in accordance to the policies followed by a client's organization.

Further, the cost aspect for CSP2 is significantly less than that for CSP1 and CSP3, since our use-case application assumes the use of Microsoft SQL servers. For Microsoft Azure platform there is no additional licensing cost for the same. This significantly cuts down the overall cost as compared to hosting on Amazon EC2 or Rackspace modeled CSPs. The decision to migrate in such scenarios is based on some additional factors which we have discussed in Section 4.

4.3. Framework Input Sensitivity Analysis. Evaluation frameworks that model real world scenario are generally complex in nature and consist of many input parameters. In this regard, our proposed cloud migration framework consists of nineteen input parameters.

Table 3. Solution Set for Different Cloud Migration Scenario

Scenario	$ **Cost** (5 years)	**Residual Damage**	**Migration Plan**
NFM-CSP1	176,750	0.00	11111111111111
	170,007	8.65	11111111111101
NFM-CSP2	31,837	0.00	11111111111111
	25,087	11.15	10111111111101
	24,949	15.19	11011111111101
	24,942	17.69	10011111111101
NFM-CSP3	247,475	0.00	11111111111111
	240,732	8.65	11111111111101
	240,695	11.15	10111111111101
	240,587	15.19	11011111111101
	240,550	17.69	10011111111101
NFMFT-CSP1	133,430	0.00	11111111111111
	114,498	8.63	11111111111110
	111,238	12.94	11111011111110
	107,755	17.28	11111111111100
	104,495	21.59	11111011111100
	101,716	28.71	11111011101100
NFMFT-CSP2	25,282	0.00	11111111111111
	19,191	10.87	11111101111101
NFMFT-CSP3	257,520	0.00	11111111111111
	191,428	8.63	11111111111110
	163,628	12.94	11111011111110
	163,607	15.16	11111001111110
	120,752	28.27	11111010101110
	113,988	39.14	11111000101100
FM-FT	128,640	0.00	01011110102100
	109,753	8.63	22002110202013
	103,345	17.28	21022110101233
	100,186	24.40	01011110232233

These input parameters can be broadly categorized as cost values, security risk level values, and discrete binary values (true or false) to depict presence or absence of security measures. In most cases, some of these input parameters play an important role in determining the output. A straightforward way to evaluate the same is to vary one input parameter and keep rest of the input parameters constant [19]. The challenge of this approach lies in the fact that in real world scenario multiple input parameters can vary simultaneously and affect each other in determining the output. In order to address this challenge, we utilized Sobol sensitivity analysis [26].

The Sobol sensitivity analysis takes into account the interactions between input parameters while computing their effect on model output. It is a useful technique for being able to measure the effect of an input or a group of input on the output of a model. The model input parameters are passed through a sample generator to generate a set of random

input values while performing Sobol sensitivity analysis for a framework. This is done by specifying a range using the minimum and maximum values a given input parameter can take. The randomly generated input set is then used to evaluate the model output and produce two basic results in the form of first-order sensitivities for measuring the effect of individual input on output and total-order sensitivity accounting effect of group of input on the output. Given the challenge of performing sensitivity analysis for real world models as discussed earlier in this section, we are interested in the total-order sensitivities for our model. The total-order sensitivity value ranges from 0 to 1, with a value of more than 0.8 considered as significant in terms of affecting the model output. A step-by-step application of Sobol sensitivity analysis can be found in [29]. We identified two of our input parameters, cost threshold and $R^{'}$ threshold, to be the most important criteria. Since, these parameters not only act as input, but are also a part of the constraints. Thus, they determine the feasible search space for finding the model output. The input parameter specifications for our Sobol analysis is summarized in Table 4.

Table 4. Framework Input Sensitivity Analysis Parameters

Attribute	**Value**	**Attribute**	**Value**
Initial Samples	5,000	Cost Distribution	100,000-500,000
$R^{'}$ Distribution	10-140	Total Runs per plan	30,000
Redundant Runs	10		

From Table 4, one can note that, for 5,000 initial samples, the random sampler generates 30,000 input samples in the specified input distribution range. The simulations were run ten times with a different random seed in every run. Thus, the obtained results are averaged over 300,000 simulation runs. Figure 4 shows the total-order sensitivities of cost and $R^{'}$ values for all of our proposed cloud migration scenarios.

In most cases, the threshold value of cost dominates $R^{'}$ (NFM-CSP1, NFM-CSP3, NFM-FT-CSP2). This can be interpreted as the availability of security coverage on the cloud platform and as such the clients do not have to invest in security measures.

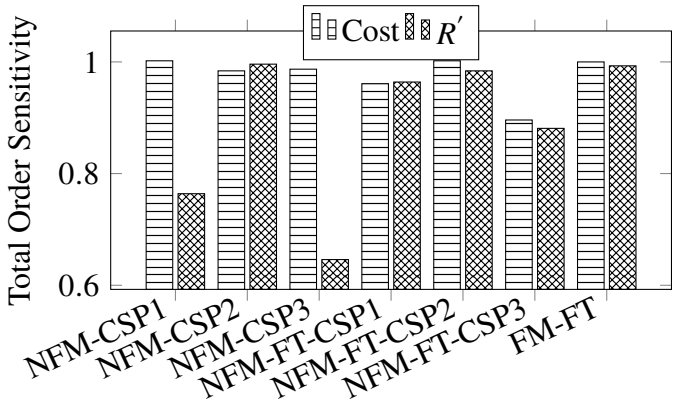

Figure 4. Total Order Sensitivities

This leaves cost threshold as the primary criteria in the formulation of a feasible cloud migration plan. The absence of security coverage is reflected when $R^{'}$ threshold is equally sensitive as cost threshold (NFM-FT-CSP1) and at times dominates it (NFM-CSP2). Although a continuous distribution can be specified for our remaining input parameters like vendor cost, security risk of individual application elements and so forth, we have not included these parameters in our Sobol analysis. This is because variances of the values for these input parameters are influenced by other inputs which in our model have discrete values of either 0 or 1 and is subjective in nature. Hence, their inclusion in Sobol analysis is not fully applicable. These relationships between the input parameters of our framework is shown through a dependency chart in Figure 5. We note from Figure 5 that discrete parameters such as Scalability, Security Coverage, Migration Constraint are the primary factors that determine the variance of other input parameters like cost of Software or Hardware, vendor services, implementing security measures and so on. Given the auto scale up and down options present in most of the cloud platforms, scalability will primarily affect the software and hardware costs incurred during private hosting. Security Coverage directly affects $R^{'}$, which might result in costs related to patching any potential security risks. Migration constraint also plays an important role in determining costs incurred by clients for private hosting.

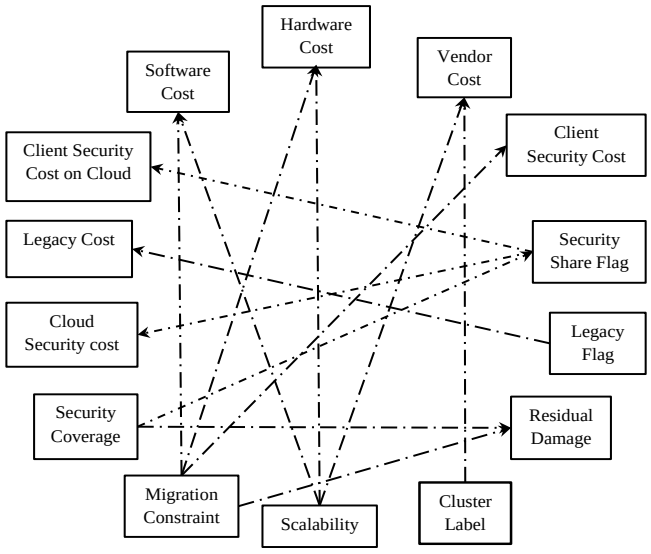

Figure 5. Dependency between Framework Input Parameters

This in turn might increase the financial implications if coupled with scalability factor. Migration constraints can also influence $R^{'}$, since one cannot migrate an element to a cloud platform even if it has security coverage present there, thereby increasing total cost of ownership. Thus, for our framework, we reckon that migration constraint plays the most important role in determining the obtained output. This is followed by security coverage along with cost threshold, $R^{'}$ threshold, and finally scalability along with the other remaining input parameters.

5. EVOLUTIONARY ALGORITHMS PERFORMANCE COMPARISON

In this section, we compare different evolutionary algorithms that could have been used to model our proposed cloud migration framework. Our objective here is to statistically validate our selection of NSGA-II to model our proposed cloud migration framework and to provide insights into feasibility of other evolutionary algorithms to model our framework. The comparison is based on criterion such as *Hypervolume*, *Maximum Pareto Front Error*,

and *Generational Distance* [18]. We have considered six different evolutionary algorithms. Their selections were made based on the rationale used by them to solve multiobjective problems. The list of algorithms and their solution strategy is summarized in Table 5.

Table 5. Evolutionary Algorithms Solution Strategies

Algorithm	**Type**	**Algorithm**	**Type**
NSGAIII	Reference Point	GDE3	Differential Evolution
PAES	Evolutionary Strategy	eNSGAII	ϵ-Dominance
OMOPSO	Particle Swarm	NSGAII	Genetic Algorithm

Figure 6 shows the Hypervolume of the algorithms specified in Table 5. Hypervolume is one of the indicators to measure the closeness of the estimated pareto front to the desired pareto front in the solution space. In this regard, larger the size of Hypervolume, better is the performance of the algorithm.

From our experiments, we note that NSGA-II and ϵNSGA-II have the best values (~0.51), followed by GDE3, NSGA-III, PAES, and OMOPSO. Further, NSGA-II and ϵNSGA-II are statistically indifferent from each other.

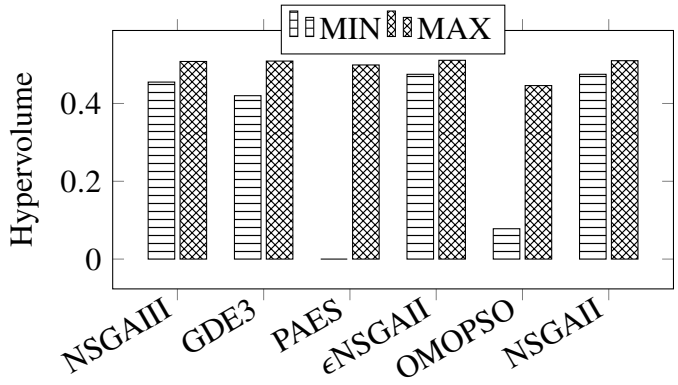

Figure 6. Hypervolume

Figure 7 shows the Maximum Front Error of the different algorithms under consideration. By definition, smaller the value of the Maximum Front Error, better the performance of the algorithm. From our analysis, we note that OMOPSO and PAES have the maximum

pareto front error, followed by GDE3. NSGA-III and ϵNSGA-II have almost the same Maximum Pareto Error, with ϵNSGA-II having marginally lesser value. NSGA-II has the least Maximum Pareto Front error amongst all other algorithms.

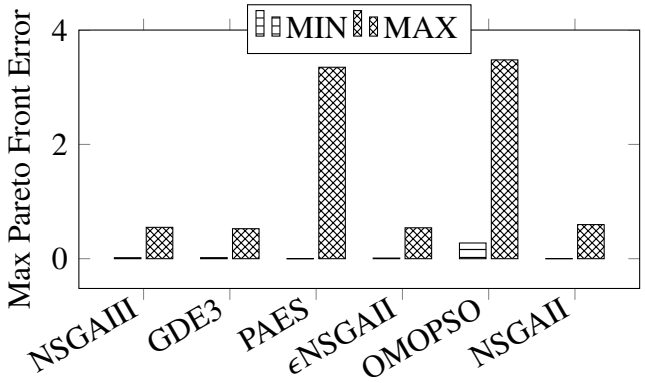

Figure 7. Maximum Pareto Front Error

Figure 8 shows the Generational Distance comparison for the algorithms. Generational Distance, like Hypervolume is the measure of the pareto front generated and its closeness to the desired ideal pareto front. Smaller the value of Generational Distance better is the performance of an algorithm. We note from the obtained results in Figure 8 that PAES and OMOPSO have the maximum generational distances. The performance of the remaining four algorithms are in close proximities of each other with ϵNSGA-II having the best results. Although, NSGA-II is statistically indifferent as compared to ϵNSGA-II.

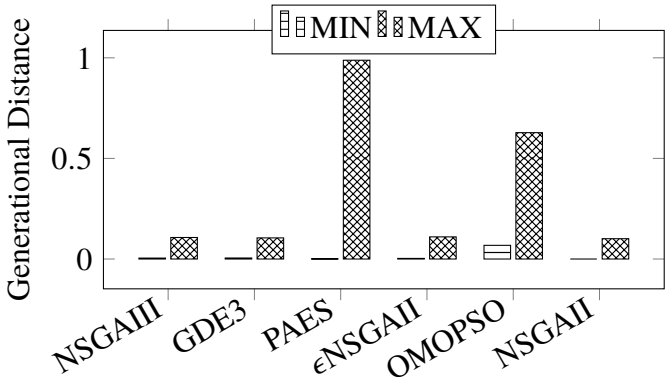

Figure 8. Generational Distance

To summarize the findings above, our currently proposed cloud migration framework yields optimal results when modeled using NSGA-II, which in most cases is statistically insignificant when compared to ϵNSGA-II and NSGA-III. Although, we reckon that the performance of ϵNSGA-II will improve over traditional NSGA-II if the number of objective functions increases for our proposed cloud migration framework. For example, if attributes of the objective function for cost is broken down to create individual objective functions. Along these lines, reference point evolutionary strategy will give better results if clients fine tune the migration constraints along with cost and security requirements by specifying desired cloud migration plans as reference point for the cloud migration framework. In such a scenario, the framework will try to find optimal cloud migration strategies by taking into consideration the client's desired cloud migration plans. The particle swarm evolution strategy in contrast to genetic algorithm strategy does not utilize crossover and mutation.

Particle swarm also does not employ any sorting techniques as used by genetic algorithm, giving it a computational advantage over genetic algorithm strategy. Although, this benefit of particle swarm is not fully utilized in our case where maximum population size for non-federated migration scenario can be approximately 2^N (where N is the number of elements in the data flow diagram) and $N = 14$. But, considering the cases of industrial applications where data flow diagrams (DFD) will not be restricted to level zero DFD, the maximum population size could be approximately 2^{hN} (where h is the depth of the DFD). Further, this population size in the case of federated migration consisting of C number of cloud providers will become, C^{hN}. In such cases, we reckon particle swarm will be a better option in modeling the cloud migration framework than genetic algorithm. In the case of differential evolution, its uniqueness lies in the way it generates child population and its application of crossover, mutation and selection parameters as compared to genetic algorithm. It will perform better than genetic algorithm when applied exclusively for our federated cloud migration framework having large number of available cloud providers.

This is because federated migration with large number of cloud providers will result in a solution vector encoding composed of real numbers with adequate diversification, which is the forte of differential evolution strategy.

6. OBSERVATIONS

In terms of performing cloud migration, our proposed framework performs cost-benefit tradeoff analysis by giving priority to the security threats present in an application instead of cost and security features of available CSPs. This is evident from the obtained results as some of the output for client's share of security responsibility ($R^{'}$) values across different migration scenarios are the same ($R^{'}$ of 0.00 and 8.00). But, there is also uniqueness in the solution set for different cloud migration scenarios. This is due to the characteristics of different CSPs. Thus, our framework extends its assessment from the client to cloud platform as claimed in our previous work [20].

Intuitively, increasing cost incurred by a client (for services and security measures) should tend to reduce $R^{'}$. But, we note from our output for scenarios like, NFM-CSP2 and NFM-FT-CSP1, that this is not always the case. This can be attributed to security coverage on a given CSP which does not require clients to invest in security patching and as such reducing $R^{'}$ without an increase in total cost. Another factor that might also attribute to such a behavior is the amount of fault-tolerance (clustering label) that could be specified by a client which can add to either the service cost or security cost (in case security coverage is not present).

Further, federated migration scenarios inherently reduce cost as it assesses and selects the best option for an application element from various available CSPs. Although, in doing so clients will have to keep in check the service policies of multiple CSPs at any given instant. These service policies are susceptible to change on a short notice. Thus, clients can reap the cost and security benefits that federated migration has to offer but should also be able to keep themselves up-to-date with the changes in CSP's policies and

if necessary, react to it. Solely in terms of cost-benefits, we note that CSP2 has the best output since it has been modeled after Microsoft Azure and our use-case application has a reliance on Microsoft software. Overall, the percentage of migration achieved for our use-case scenario on an average is about 80% (see Table 3). But, we reckon that this can be affected by changes in the model input (Figure 5). Although, our framework being modeled as a multi-objective optimization problem will still provide users with a set of alternative solutions regardless of the migration scenario. This will enable them to compare and contrast the obtained migration plans and assess it for various organizational factors that might be applicable. For example, Quality of Service parameter like performance of the application can be evaluated by observing the elements that has been suggested for migration [3] and comparing it to the migration status of the elements that directly interact with the later.

This aspect is very useful if users opt for federated migration scenario. Further, given the modular structure of our framework, users can add additional input factors and objectives (or constraints) to tailor model output to meet their comprehensive needs.

7. RELATED WORK

The task of cloud migration benefit clients as it reduces the inconvenience of owning and maintaining software or hardware to run their applications. However, the task itself has several challenges and careful considerations are required with respect to various parameters most notably cost effectiveness, security of applications on the cloud platform, and migration constraints due to legacy applications. In the following paragraphs we summarize some compelling works related to cloud migration, categorized by the aforementioned parameters they take into consideration.

The authors in [15] describe a hybrid cloud migration process with the primary goal of reducing total cost of ownership for large scale organizations. They present a comprehensive study, identifying numerous cost factors that govern cloud migration policies. In [16],

authors present a case study showing the cost implications of migrating an application to the IaaS platform of Amazon EC2. They have also identified that cost is not the sole criteria affecting cloud migration as other non-financial aspects like reputability of CSPs, technical unfamiliarity play an important role in the decision making process. They extended their case study presenting a comprehensive modeling tool [17] that provides cost estimates of migrating to public IaaS cloud platform. The benefits are in its capability to compare and contrast the services provided by different CSPs.

Authors in [27] present the challenges of migrating existing applications to the cloud platform like changes made to the software environment, programming models such that it can operate on the cloud platform. They present a model identifying the tasks involved in the cloud migration process and analyzed them according to the cost involved in performing them. This work helped in the formulation of our migration feasibility model.

In [21], authors discuss the technical and non-technical challenges of migrating a web server to the cloud platform. They discuss the non-trivial aspects in cloud migration which involves taking into consideration many criteria, addressing all of which manually is challenging. As such, they propose an automated framework, *CloudGenius*, for helping clients in the decision making process for cloud migration. Authors in [2] present an overview of the challenges in performing cloud migration like financial aspects, security concerns arising due to cloud migration such as multi-tenancy and data confidentiality. The authors in [13] present a detailed cloud migration decision making framework. It takes into account the architecture of an application to be migrated along with its service requirements in terms QoS factors, security risks involved, and the financial aspects of the process.

Authors in [12] perform an in-depth review of existing cloud migration techniques related to hosting legacy systems on the cloud platforms. Their literature review discusses the similarities between Service Oriented Architecture (SOA) and Cloud platforms. In [9], authors present an evolutionary, iterative approach to help small-medium enterprises (SME) is able to migrate their legacy applications to the cloud platform. The outlined solution

aims in making the process less dependent on any particular kind of technology support or CSP. Through these works, we were able to identify and acknowledge the impact and challenges of migrating legacy applications to the cloud platform. Authors in [28] discuss the impact of various NSGA-II algorithmic parameters on the optimization process and the obtained solution set. The experiments and results presented in [28] helped us form our basis for NSGA-II sensitivity analysis. Our proposed cloud migration framework extends our previously proposed risk assessment framework [20] and addresses the optimality between most notable factors like client cost and application security while performing cloud migration. In doing so, we have extended and refined cost parameters as presented in [15] [16] and introduced the costs related to addressing the security risk by implementing the security measures. For security evaluations we utilized our risk assessment framework which introduces the novel concepts.

These are *security coverage, risk reduction factor,* and *client's share of security responsibility*, extending application risk assessment hosted on the client's private network to one or multiple CSPs. This sort of security evaluation is more economical and outlines the relevance of the security policies of different CSPs, giving more confidence to clients in the security aspect of cloud migration as compared to existing works using security SLAs [8] or security checklist [17]. Additionally, our proposed cloud migration framework also incorporates other challenges of performing cloud migration like hosting existing legacy applications, migration constraints due to technical or legal concerns, and elasticity [9] [27]. We have also introduced the concept of fault-tolerance which differentiates from existing approaches of providing cloud migration reliability [24] by using hosting based on application functionality instead of replication. The nature of our cloud migration framework is generic, integrating several different parameters and is not specific to any particular clients like small or medium enterprises [22]. Our framework modeling is done

using genetic algorithm (NSGA-II) in an attempt to semi-automate the challenging task of hybrid cloud migration for federated and non-federated scenarios which to the best of our knowledge is the first attempt in this direction.

8. CONCLUSION

In this paper, we have presented cost-benefit analysis of a cloud migration framework modeled as a multi-objective optimization problem for optimizing the cost of migration and client's share of security responsibilities. We have proposed various migration scenarios like non-federated and federated cloud migrations supported by a newly introduced concept of fault-tolerance which allows to separately host application elements based on their functionality. We have applied our proposed migration framework to a use-case application scenario and analyzed the output as well as the sensitivity of input parameters used in our framework. Furthermore, we have evaluated the performance of different evolutionary algorithms to model our proposed framework.

This helped in statistically validating the selection of NSGA-II and provided insights into the applicability of other evolutionary algorithms for cloud migration problem. These insights can be used to model different migration scenarios (Section 5) which relies on application design.

Nevertheless, we reckon that cost and security cannot be the only factors to determine a suitable CSP for a client's application. Once an optimal cloud migration plan has been generated and adopted, the decision on its continuity is met with various challenging aspects. In this regard, one should be able to supplement the proposed cloud migration framework with subjective information like reputability of a CSP and objective information like availability, throughput, and other performance metrics. Hence, as an extension to our current cloud migration framework, we plan to formulate an On-line Trust Evaluation

Framework to address these challenges by collectively associating the subjective and objective parameters to quantify the suitability of selected cloud service providers to continue hosting a client's application.

REFERENCES

[1] Accurately estimate your software maintenance costs. galorath.com/software_maintenance_cost. Available Online.

[2] Vasilios Andrikopoulos, Steve Strauch, and Frank Leymann. Decision Support for Application Migration to the Cloud: Challenges and Vision. In *Proceedings of the 3rd International Conference on Cloud Computing and Service Science (CLOSER'13)*, pages 1–7. SciTePress, May 2013.

[3] Sean Kenneth Barker, Yun Chi, Hyun Jin Moon, Hakan Hacigümüs, and Prashant J. Shenoy. "cut me some slack": latency-aware live migration for databases. In *15th International Conference on Extending Database Technology, EDBT*, pages 432–443, 2012.

[4] Erdal Cayirci, Alexandr Garaga, Anderson Santana de Oliveira, and Yves Roudier. A risk assessment model for selecting cloud service providers. *Journal of Cloud Computing*, 5(1):14, 2016.

[5] Carlos A. Coello. A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information Systems*, 1(3):269–308, 1999.

[6] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Trans. on Evolutionary Computations*, 6(2):182–197, April 2002.

[7] K. Djemame, D. Armstrong, J. Guitart, and M. Macias. A risk assessment framework for cloud computing. *IEEE Transactions on Cloud Computing*, 4(3):265–278, July 2016.

[8] N. Ghosh, S. K. Ghosh, and S. K. Das. Selcsp: A framework to facilitate selection of cloud service providers. *IEEE Transactions on Cloud Computing*, 3(1):66–79, Jan 2015.

[9] Alexander Gunka, Stepan Seycek, and Harald Kühn. Moving an application to the cloud: An evolutionary approach. In *Proceedings of the 2013 International Workshop on Multi-cloud Applications and Federated Clouds*, MultiCloud '13, pages 35–42.

[10] Matthew L. Hale and Rose Gamble. Secagreement: Advancing security risk calculations in cloud services. In *8th IEEE World Congress on Services*, 2012.

[11] P. Jamkhedkar, J. Szefer, D. Perez-Botero, T. Zhang, G. Triolo, and R. B. Lee. A framework for realizing security on demand in cloud computing. In *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*, volume 1, pages 371–378, Dec 2013.

[12] P. Jamshidi, A. Ahmad, and C. Pahl. Cloud migration research: A systematic review. *IEEE Trans. on Cloud Computing*, 1(2):142–157, July 2013.

[13] Bjorn Johnson and Yanzhen Qu. A holistic model for making cloud migration decision: A consideration of security, architecture and business economics. In *10th IEEE International Symposium on Parallel and Distributed Processing with Applications, ISPA*, pages 435–441, 2012.

[14] Patrick Kamongi, Mahadevan Gomathisankaran, and Krishna Kavi. Nemesis: Automated architecture for threat modeling and risk assessment for cloud computing. In *The 6th ASE International Conference on Privacy, Security, Risk and Trust (PASSAT), Cambridge, MA, USA,*, Dec 2014.

[15] Mohammad Mahdi Kashef and Jörn Altmann. A cost model for hybrid clouds. In *Economics of Grids, Clouds, Systems, and Services: 8th International Workshop*, pages 46–60, 2012.

[16] A. Khajeh-Hosseini, D. Greenwood, and I. Sommerville. Cloud migration: A case study of migrating an enterprise IT system to IaaS. In *IEEE 3rd International Conference on Cloud Computing*, pages 450–457, July 2010.

[17] A. Khajeh-Hosseini, I. Sommerville, J. Bogaerts, and P. Teregowda. Decision support tools for cloud migration in the enterprise. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 541–548, July 2011.

[18] M. Li, S. Yang, and X. Liu. Diversity comparison of pareto front approximations in many-objective optimization. *IEEE Transactions on Cybernetics*, 44(12):2568–2584, Dec 2014.

[19] Daniel P. Loucks. Model sensitivity and uncertainty analysis. In *Water Resources Systems Planning and Management*, pages 255–290. UNESCO, 2005.

[20] S. Madria and A. Sen. Offline risk assessment of cloud service providers. *IEEE Cloud Computing*, 2(3):50–57, May 2015.

[21] M. Menzel, R. Ranjan, L. Wang, S. U. Khan, and J. Chen. Cloudgenius: A hybrid decision support method for automating the migration of web application clusters to public clouds. *IEEE Transactions on Computers*, 64(5):1336–1348, May 2015.

[22] Nicolas Nussbaumer and Xiaodong Liu. Cloud migration for SMEs in a service oriented approach. *IEEE 37th Annual Computer Software and Applications Conference Workshops.*, 2013.

[23] Antonin Ponsich, Antonio LÃşpez Jaimes, and Carlos A. Coello Coello. A survey on multiobjective evolutionary algorithms for the solution of the portfolio optimization problem and other finance and economics applications. *IEEE Trans. Evolutionary Computation*, 17(3):321–344, 2013.

[24] W. Qiu, Z. Zheng, X. Wang, X. Yang, and M. R. Lyu. Reliability-based design optimization for cloud migration. *IEEE Transactions on Services Computing*, 7(2):223–236, April 2014.

[25] Margarita Reyes-sierra and Carlos A. Coello Coello. Multi-objective particle swarm optimizers: A survey of the state-of-the-art. *International Journal of Computational Intelligence Research*, 2(3):287–308, 2006.

[26] I. M. Sobolá. Global sensitivity indices for nonlinear mathematical models and their monte carlo estimates. *Math. Comput. Simul.*, 55(1-3):271–280, feb. 2001.

[27] Van Tran, Jacky Keung, Anna Liu, and Alan Fekete. Application migration to cloud: A taxonomy of critical factors. In *Proceedings of the 2nd International Workshop on Software Engineering for Cloud Computing*, SECLOUD '11, pages 22–28, 2011.

[28] Wafa Ben Yahia, Omar Ayadi, and Faouzi Masmoudi. A sensitivity analysis of multi-objective cooperative planning optimization using NSGA-II. In *Proceedings of the Multiphysics Modelling and Simulation for Systems Design Conference*, pages 327–337, 2015.

[29] X-Y Zhang, MN Trame, LJ Lesko, and S Schmidt. Sobol sensitivity analysis: A tool to guide the development and evaluation of systems pharmacology models. *CPT Pharmacometrics Syst. Pharmacol.*, (4):69–79, 2015.

IV. DATA ANALYSIS OF CLOUD SECURITY ALLIANCE'S SECURITY, TRUST AND ASSURANCE REGISTRY

Amartya Sen and Sanjay Madria

Department of Computer Science

Missouri University of Science and Technology

Rolla, Missouri 65409–0050

Email: {asrp6,madrias}@mst.edu

ABSTRACT

The security of clients' applications on the cloud platforms has been of great interest. Security concerns associated with cloud computing are improving in both the domains; security issues faced by cloud providers and security issues faced by clients. However, security concerns still remain in domains like cloud auditing and migrating application components to cloud to make the process more secure and cost-efficient. To an extent, this can be attributed to a lack of detailed information being publicly present about the cloud platforms and their security policies. A resolution in this regard can be found in Cloud Security Alliance's Security, Trust, and Assurance Registry (STAR) which documents the security controls provided by popular cloud computing offerings. In this paper, we perform some descriptive analysis on STAR data in an attempt to comprehend the information publicly presented by different cloud providers. It is to help clients in more effectively searching and analyzing the required security information they need for the decision making process for hosting their applications on cloud. Based on the analysis, we outline some augmentations that can be made to STAR as well as certain specific design improvements for a cloud migration risk assessment framework.

Keywords: Cloud Computing, Data Analyses, CSA STAR, Risk Assessment

1. INTRODUCTION

The cloud platforms have received a gradual acceptance in the community due to its beneficial services provided through well-known delivery models like software-as-a-service (SaaS), platform-as-a-service (PaaS), and infrastructure-as-a-service (IaaS). However, security of client's application on the cloud platform is still one of the primary concerns that make them reluctant in completely adopting their services. A client's requirement in these regards has been to procure from cloud service providers (CSPs) transparent and concrete information regarding the security of their platform across different delivery models. To bridge this gap, several works from industry and academia alike have been proposed in the recent past. Some of these works [6] [21] [14] provide guidelines for clients to help them perform security assessment of various CSPs while others [1] [15] help in establishing best practices regarding application security on the cloud platform.

Nevertheless, comprehending cloud security is not a trivial task as the process becomes convoluted in the presence of multiple dimensions across which security needs to be addressed like networking, application, data storage, and so forth. Additionally, CSPs are reluctant to publicly publish detailed internal security information of their platforms due to market competitiveness and the risk of enhancing malicious activities. A stellar effort in publicly making available the security assessments of numerous CSPs with a certain degree of transparency and accuracy can be found in Cloud Security Alliance's (CSA) Security, Trust and Assurance Registry (STAR) [3]. The security evaluations of CSPs in STAR is conceived using the Consensus Assessments Initiative Questionnaire (CAIQ) which is based on CSA's Cloud Controls Matrix (CCM) [2]. These security evaluations are addressed across different cloud platform dimensions like - identity and access management, encryption and key management, auditing, and so forth which CSA has coined as *control groups*. These control groups pose a set of assessment questions.

In CAIQ version 3.0.1, these questions can be answered by a CSP in a *yes, no,* or *not applicable* fashion. Further, they can also provide supplementary *notes* to the *yes or no* answers and specify the compliances that they are following for a control group.

Several works are present in the literature which have addressed the concerns of security assessment of CSPs by utilizing CSA STAR [6] [9]. The primary objectives of these works have been to analyze the control groups answered by various CSPs to create quantitative security rating scales helping clients to objectively understand the security provided by different CSPs. These kinds of quantitative approaches benefits the clients as it could be quite tedious to manually peruse the entirety of STAR registry looking for CSPs that might synchronize with their security requirements. However, we reckon that these approaches do not fully alleviate the security concerns that clients might have once their applications are hosted on the cloud platform. This is because the objective security assessment performed in these approaches does not account for the security threats present in a client's application and how the underlying security practices of a CSP will nullify them. To address this challenge, the offline risk assessment of CSPs framework was proposed in [19]. The notable features of the offline risk assessment framework can be summarized in its capability to perform application risk assessment during the design phase including cloud security metrics, aggregating the cloud security metrics of various CSPs, and then performing a cost-benefit tradeoff analysis to develop an optimal cloud migration plan for the client's application. The tradeoff analysis is an important aspect of the framework in contrast to other cloud security assessment approaches since security is one of the primary concerns but not the only one which impedes complete adoption of cloud services.

Considering the outlines of the offline risk assessment framework, our contributions in this work are as follows:

- Perform a descriptive and exploratory analysis of CSA's STAR registry to better understand the pattern and context of answers from different CSPs.

- Use the results of our analysis to suggest improvements to the structure of CAIQ and CCM which is used to populate the STAR registry.

We envision that the results of this study will further help in improving the design of the offline risk assessment framework. This is because analyzing the pattern and context with which CSPs have responded to the security assessment questions will help clients further improve the design of their application and reduce some of their impending security concerns.

The rest of the paper is organized as follows - Section 2 discusses some of the works in the literature that utilizes CAIQ and STAR to perform security assessments of CSPs. We present our descriptive and exploratory analysis of STAR in Section 3. Section 4 discusses the enhancements we suggest to CAIQs and STAR registry from the results obtained through our analysis. Section 5 outlines the improvements that can be made to the offline risk assessment framework by using our results. We conclude our work in Section 5.

2. RELATED WORK

Risk assessment of cloud computing platforms and the services offered by different cloud vendors has been one of the prominent fields of work related to the cloud computing domain. This is because of several factors like migrating the client's application from their private domain to the public domain of the cloud service providers (CSPs), partially losing control over their data, and uncertainty regarding the cloud provider's security policies and their assurance. Several works related to this field can be found in the literature, however, there are still numerous open and emerging security challenges and threats to the domain of cloud computing as identified in [8]. Further, the advent of Internet of Things (IoT) paradigm supported by cloud computing platforms will also introduce new security challenges [20].

The authors in [6] proposed a risk assessment model evaluating CSPs utilizing the public information available in CSA's STAR registry and built a quantitative framework profiling different CSPs based on their security, privacy and services. Other instances of works that utilizes CSA STAR repository and its CAIQs can be found in [9]. In [21], authors discuss another quantitative model to evaluate and compare the security dispensed by CSPs which is based on the concept of developing security service level agreements (SecLAs). The authors make use of the Analytic Hierarchy Process (AHP) for its decision making purpose and evaluate the SecLAs qualitatively and quantitatively. Additionally, authors in [5] present the concepts of automatically enforcing SecLAs in the cloud platforms. The authors propose a model in this regard which engages the software life cycle of the components that is covered in the SecLAs to determine the associated constrains of the security components, the security requirements of the clients and follows it with automatic provisioning and configuration of the selected security resources. However, evaluating and addressing security concerns on the cloud platform is not the end game. Given the dynamic nature of the cloud, architecture and policies are susceptible to change on a short notice. Therefore, clients need assurance that the identified security policies are still in place. The context of security assurance has been discussed extensively in [4]. Along these lines, authors in [12] have identified that in order to keep up with changing cloud environments and evaluate the change in security policies and their assurances there is a need for continuous auditing. Furthermore, authors in [14] propose techniques to dispense real-time cloud security assessment. Their approach, called the Moving Interval Process can help clients compare their security assessments with the security offerings of the CSPs and also help CSPs compare their services with other CSPs by understanding the needs of their clients with high accuracy and computational efficiency. Additionally, other works focus solely on evaluating particular types of delivery models like software-as-a-service [22], infrastructure-as-a-service [11], or a specific domain of the cloud security like data security [7].

3. DESCRIPTIVE AND EXPLORATORY ANALYSIS OF CSA STAR

In this section, we will give a brief overview of the CSA STAR registry followed by our document pre-processing techniques that we performed on the datasets collected from the STAR registry. Thereafter, we present the results of our descriptive and exploratory analysis and discuss some of the lessons learnt from them which will be used as the foundation to suggest enhancements to the STAR registry such that it can be optimally integrated with the offline risk assessment framework for cloud service providers (CSPs).

3.1. The STAR Registry: Background. A good publicly available source in terms of aggregating security measures employed by different CSPs can be found on Cloud Security Alliance's (CSA) Security, Trust & Assurance Registry (STAR) [3]. This registry is based on CSA's Cloud Controls Matrix (CCM) document [2], which provides security assessment questions across sixteen cloud security domains (referred to as control groups). These security assessment questions also known as the Consensus Assessments Initiative Questionnaire (CAIQ), can be used by CSPs to provide security assessment of their services and by client's to evaluate the security guidelines present with a particular CSP. Some examples of the control groups used in CCM are *Application and Interface Security*, *Audit Assurance and Compliance*, *Identity and Access Management*, and so on. These domains are further broken down into their subcategories which help in elaborating the kind of security elements that should be addressed in these control groups. CCM also lists the impact of these control groups on cloud platform's architectural domains like, *Physical infrastructure, Networking, Computing, Storage, Application, and Data* along with its applicability to delivery models like, *SaaS, PaaS, and IaaS*. It also specifies if the exploitation of a control group solely affects the service provider and/or clients along with its impact on corporate governance. Finally, it lists the appropriate compliance and standards that CSPs should follow and be certified with (if applicable) like, *HIPAA, COBIT, ISO 27001* and so forth. We consider CCM version 3.0.1 which consists of 133 total assessment questions in general (16 control groups and their subcategories).

Out of these, 71 assessment questions affect clients (in union with service providers; no exclusivity) whereas, 61 exclusively affects service providers. Some of the assessment questions are further sub-categorized by CSPs while answering them in the STAR registry. In such cases, there is a net total of 295 assessment questions. In CAIQ version 3.0.1, which we have considered for our analysis purposes, CSPs discretely address them according to *yes, no, or not applicable* answers along with some optional *notes* elaborating or providing supplementary information to their answers. The 16 control groups and their acronyms as outlined in CCM is summarized in Table 1.

Table 1. CCM Control Groups

Control Group		**Control Group**	
App. and Interface Security	AIS	Human Resources	HRS
Audit Assurance and compliance	AAC	Identity and Access Management	IAM
Business Continuity Management and operational resilience	BCR	Infrastructure and Virtualization Security	IVS
Change, Control and Configuration Management	CCC	Interoperability and Portability	IPY
Data Security and Information Lifecycle Management	DSI	Mobile Security	MOS
Data center Security	DCS	Security Incident Management, E-discovery and Cloud	SEF
Encryption and Key Management	EKM	Supply Chain Management, Transparency and Accountability	STA
Governance and Risk Management	GRM	Threat and Vulnerability Management	TVM

3.2. Document Pre-processing. We scrapped the STAR registry on CSA's website to collect CAIQ documents answered by 201 CSPs. Majority of these were present in `*.xls` format with some in `*.pdf`. Out of these, we found 138 of them (all in *.xls format) to be in accordance with CAIQ version 3.0.1. Although, these documents followed a similar structure (CAIQ and CCM documentations), their answering techniques varied for some CSPs. To elaborate this, while answering questions as *yes, no* or *not applicable*, some CSPs have used check marks like `x`, while others use characters like `Y`, or strings like `Yes`. We cleaned all the collected documents with the objective of creating a universal structure of the answering columns. To address such challenges and others like these, we imported all the collected documents into `R` studio. We cleaned all the extra rows and columns due to the import process of an excel document to a `R` data frame. Thereafter, for the answering column inconsistencies, we replaced an affirmation response, either in *yes*, *no*, or *not applicable*

column, with a numerical 1 and remaining answering columns (of the same row) with a 0. We further broke down each document import into two separate data frames; one (let us refer to this as dfGen) consisting of the *control ID* of the CAIQ questions, its description, *notes* columns (all of which were character datatype), and the *yes/no/not applicable* answering columns (numerical). The other data frame (let us refer to this as dfComp) of a CSPs import consisted of the control ID, its description and *compliance* information (all character data types). Additionally, document import also introduced NAs which were addressed and removed from the created data frames. However, the pre-processing techniques could not be applied to all of the 138 documents because some of the documents were write protected by the CSPs due to which we could not make certain changes in the original xls documents which were required by the cleaning procedures written in R. This reduced our sample size for descriptive and exploratory analysis from 138 to 108 CSPs.

3.3. Analysis and Lessons Learned. We first present a summarized analysis of the CCM document. The objective behind it is to get a general understanding about facts like how many assessment questions belong to a control group and how they affect different cloud platform domains like *network, storage, data*, and so forth. The total number of security assessment questions posed in each of the 16 control domains in the Cloud Control Matrix's (CCM) Control Group Structure is shown in Figures 1 and 2. In these figures, the x-axis represents the 16 different control groups and the y-axis represents the number of assessment questions belonging to each of these control groups. The question we ask ourselves from this is - whether the quantity of questions presented in a control group a direct indicator of its emphasis in the cloud security. In other words, does a control group like Mobile Security (MOS) consisting of the largest number of questions (20) reflects the sentiments of the cloud community in terms of emphasizing its importance with respect to cloud security. Such notions will then convey the ideology that the community emphasizes on MOS category

more than Encryption and Key Management (EKM) which has 4 assessment questions (Figure 1). However, basing these notions on a single criterion (total number of questions in a control group) will not help in depicting the complete picture.

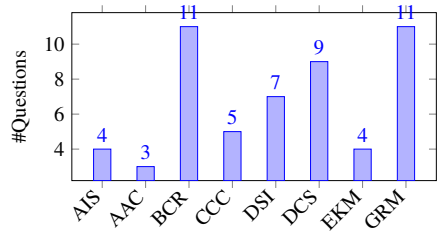

Figure 1. Cloud Control Matrix's Control Group Structure - Part 1

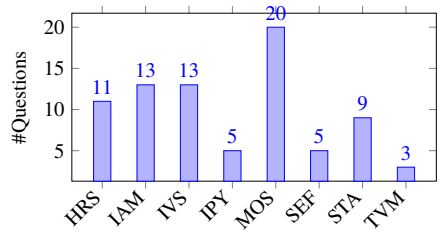

Figure 2. Cloud Control Matrix's Control Group Structure - Part 2

We further explored the relative impact of these 16 control groups on the different aspects of the cloud platform as shown in Table 2 and 3. In these tables, the column

Table 2. Impact of Control Groups on Cloud Platform Domains - 1

	AIS	AAC	BCR	CCC	DSI	DCS	EKM	GRM
Physical	25	100	73	20	0	100	0	46
Network	75	100	64	20	14	44	75	46
Compute	100	100	64	100	71	44	25	55
Storage	100	100	64	100	86	56	100	55
App	100	100	55	100	86	33	75	55
Data	100	100	64	80	100	33	100	55
CorpRelevance	50	100	82	100	71	22	75	100
Tenant	50	67	36	40	57	22	50	82

represents the control groups proposed in CCM and the row shows the different cloud platform domains that these control groups affect. Each cell of these tables exhibits the relative percentage impact of the control groups on the cloud platform domains.

Table 3. Impact of Control Groups on Cloud Platform Domains - 2

	HRS	IAM	IVS	IPY	MOS	SEF	STA	TVM
Physical	64	62	54	20	40	100	100	0
Network	55	69	77	100	25	100	100	100
Compute	55	85	100	80	40	100	100	100
Storage	55	85	69	80	15	100	100	67
App	64	92	69	80	65	100	100	100
Data	91	85	69	80	35	100	100	33
CorpRelevance	100	31	46	40	90	100	100	33
Tenant	100	62	46	20	45	100	22	68

For example, in Table 2, an impact value of 25% of Application and Interface Security (AIS) control group on *Physical* infrastructure domain of the cloud platform elucidates that given all the security assessment questions and their sub-categories belonging to the AIS control group, a quarter of them impacts the *Physical* domain. Analyzing Table 2 and 3 row-wise, we can perform categorical analyses to further (and more accurately) understand how different cloud platform domains are impacted by the control groups as exhibited by the CCM document structure. For example, consider *Storage* domain; the highest impact is 100% from 6 control groups (AIS, AAC, CCC, EKM, SEF, STA). The next impact category can be considered to be 80% and above which will have 3 control groups (DSI, IAM, IPY). The remaining impact categories can be 50% to 80% and below 50%, having 6 (BCR, DCS, GRM, HRS, IVS, TVM) and 1 (MOS) control groups, respectively. Additionally, column-wise analysis of these tables by aggregating the relative impact scores of each control group across all cloud domains can shed light from another perspective into the relevance of the control groups in terms of security assessments of CSPs. While using the STAR registry for security evaluation of the CSPs, users can take into account such information to have a better comprehension of the assessments provided and further improve upon it by adding their requirements. For example, does a user give more emphasis on the *Storage* domain over *Compute* for their application. We will discuss more on this in the enhancement to STAR registry in Section 4.

We then performed exploratory analysis on the `dfGen` data frame which consisted of *yes, no,* or *not applicable* responses for 295 net total CAIQ questions from 108 CSPs. The objective behind this was to understand the pattern of yes, no, and not applicable answers from different CSPs. The summarized answering response scores for *yes, no,* and *not applicable* are shown in Figure 3, 4, and 5, respectively. The min and max scores for

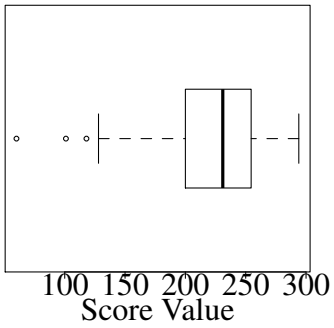

Figure 3. Distribution of Yes Responses in STAR Registry

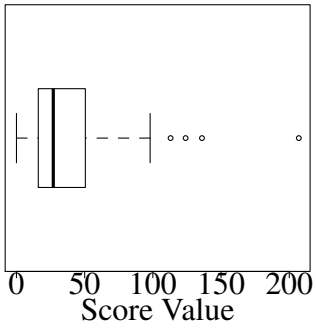

Figure 4. Distribution of No Responses in STAR Registry

yes responses (Figure 3) are 128 and 294, respectively. Its mean lies approximately at 236. It also consists of 3 outliers (score - 60,101,118). The *no* response scores (Figure 4) have a min and max of 0 and 98, respectively with mean value approximately 26 and consisting of 4 outliers (score - 113,124,136,207). Finally, for the *not applicable* responses (Figure 5) the min and max scores are 0 and 90, respectively. The mean score is approximately 26, with 3 outliers (score - 96,116,192).

Overall, we observed that some of the CSPs that are the outliers of the *no* and *not applicable* response set are mostly SaaS vendors. They offload most of the infrastructure tasks to other vendors. This results in high count of no or not applicable with respect control groups like DCS or IVS since currently there is no way to nullify these control groups from the CAIQ (We discuss this further in Section 4). Additionally some of the CSPs have either answered *yes or no* (and ignored the not applicable column altogether), or in some instances they have answered *yes* and *no* for an assessment question. Hence, intuitively, the outliers of the *yes* response set are outliers in either the *no* or *not applicable* response set. In addition

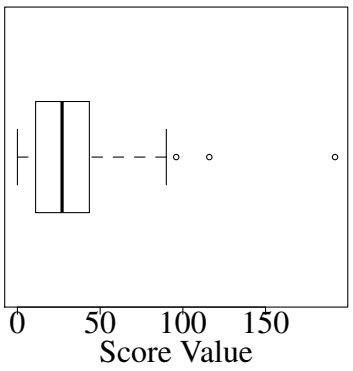

Figure 5. Distribution of NA Responses in STAR Registry

to the scores, the `dfGen` data frame consists of *Notes* from CSPs explaining some of the vendor specific control group parameters. One of the considered attempts to comprehend these notes in a (semi-)automated way was to make a compilation of the entire notes set for a CSP. However, doing so would have made the whole analysis too broad like reading through a policies whitepaper without any focus as to what the reader might be looking for, thereby making the process generic in nature. Alternatively, one can aggregate only the notes of those control group assessment questions which has been responded either with a *No* or *Not applicable*. Although, as we considered the highest score outlier for the *No* responses (Figure 4) we could not find any supporting notes. Further, in these cases most of the notes were documented as “in progress” or “can be ordered”.

Similarly, for the highest score outlier for *not applicable* responses, no notes were provided. In contrast, many vendors who answered *Yes* have provided notes which were much more detailed. As such, we reckon that notes should be present when CSPs respond with a *no or not applicable* and when available it can be followed up with manual analysis.

However, for one of the assessment questions - BCR-06.1: "Are any of your data centers located in areas having high probability of environmental risk"; responding it with a no is positive affirmation. This has also been pointed out in other works like [6]. Although, this is the only question of its kind amongst the 295 questions so while scoring using version 3.0.1, one cannot simply count the number of Yes and No responses. Nevertheless, this should not be sufficient grounds to reject the usage of the version 3.0.1 and when it comes to (semi)automated analysis, being aware of the context of the question will be essential. For example, in CGID BCR-1.2 - "Do you provide tenants with infrastructure service failover capability to other providers", if a vendor responds with *no*, should this be rated negatively and reduce the categorical rating of the vendor? Since, in notes they explain why they do not provide the specified control and the counter/failsafe to the requested control is provided. As such, solely relying on aggregating *no* scores will negatively impact the scoring and we need to able to integrate context aware analysis for the assessment questions. For example, to accurately rate BCR-1.2's answer, one needs to be able to assess the kinds of counter/failsafe options that are provided and if they are comparatively better than the control specification stated in the assessment question. Automated analysis of such kind of information without or minimal human intervention is not a trivial task, but it is something that needs to be addressed. Furthermore, when assessment questions are responded with a yes, it may induce follow up questions like for AAC-2.4 - "Do you conduct internal audits regularly as prescribed by industry best practices and guidance". The response to these questions is typically yes, but at times it either does not mention which industry best practices they follow, or clarify if those best practices are universally applicable across all CSPs (considering delivery and deployment models).

Although, at times CSPs cannot reveal all the information in details on a public forum due to several reasons like market competitiveness and/or presence of malicious entities which can then compromise their security. We will discuss ways to address this in Section 4.

Additionally, we analyzed assessment question responses of the 16 control groups individually to see their pattern and check if there were any similarities across different CSPs (aggregating by their delivery model and so forth) and contrasting it with the analyses of the original CCM structure as presented in Table 2 and 3.

Table 4. Score Responses from CSPs on Individual CAIQ Control Groups

	AIS	AAC	BCR	CCC	DSI	DCS	EKM	GRM
Yes%	82	86	72	86	75	82	65	85
No%	8	11	16	7	13	8	21	12
Not Applicable%	10	3	12	7	12	10	14	3
	HRS	IAM	IVS	IPY	MOS	SEF	STA	TVM
Yes%	85	79	79	70	49	80	80	85
No%	9	15	6	11	17	16	11	7
Not Applicable%	6	6	15	18	34	4	9	8

From Table 4, we note that on average the control groups of AAC and CCC have the maximum *yes* responses. The maximum *no* responses belongs to the control group EKM and maximum *not applicable* responses to MOS control group, which also has the minimum *yes* responses. The minimum *no* response belongs to the IVS control group and thereafter the minimum *not applicable* belongs to the control groups AAC and GRM. If we hypothesize the importance of a control group based on its relevance impact across different cloud platform domains (Table 2 and 3), we would say that control groups like AIS, AAC, CCC, SEF, and STA will be the top contenders which the CSPs will address and respond with *yes* to the assessment questions belonging to these control groups. However, considering the summarized results in Table 4 and taking into consideration the survey performed by RightScale in their 2017 state of the art cloud report [16], our hypothesis is

partially true. To further elaborate on this, according to our hypothesis it does turns out that control groups of AAC and CCC are of importance and majority of CSPs respond positively to the assessment questions in this domain.

This notion is also affirmed by RightScale's survey that as the maturity of cloud platform models and understanding of the client base keep on increasing, issues like compliance and audits take priority over raw security aspects of the cloud platform. Therefore, CSPs seems to increasingly address these areas more positively leaving very little or no ambiguities (minimum not applicable response for AAC). As far as security itself goes, one may state that the notions outlined in control groups of IAM and EKM is of value and would be addressed by CSPs as positively as compared to AAC or CCC control groups. However, this does not seem to be the case as we saw from our analysis that EKM consisted of the maximum *no* responses from the CSPs and IAM also has a leaning towards the minimal bounds. We attribute this behavior to two reasons, (1) as RightScale's survey show there is reduction in the overall security concerns of clients compared to the previous years with the maturity of the cloud models and, (2) we reckon that CSPs believe that the aspects of these control groups will be more effectively handled by clients themselves giving them more control over the security of their hosted application. Furthermore, quite a few of the CSPs from our collection did not have IaaS provisioning or borrowed the services of other CSPs for performing storage and computation aspects. This could also attribute towards the high *no* count for domains like EKM which otherwise seems to be of importance when it comes to security on the cloud platform.

The control group of MOS itself has the maximum number of assessment questions as compared to the other control groups. However in terms of its relative impact with respect to the cloud platform domains, it is lackluster if compared to AAC, CCC, SEF, SEF, and so forth. When it comes to responding to this control group, as we can see from Table 4, it has the minimum *yes* responses and maximum *not applicable* responses. Its *no* responses is also the second highest in the spectrum.

According to RightScale's survey the notion of this control group is not touched upon extensively. We strongly believe that with the rising trend in the concept of Internet of Things (IoT), CSPs should pay more attention to this area since, its eventual exploitation will impact other control groups. Finally, based on our CCM structure analysis we hypothesized that the control groups of SEF and STA have considerable impact on all the considered cloud platform domains and it will be addressed positively on average by the CSPs. However, this hypothesis is not reflected from the results in Table 4. On the contrary, SEF has the third highest *no* responses.

Another valuable trove of information is embedded in the *notes* column of the CSP's CAIQ responses. Generally, these notes are textual, unstructured information providing supplementary information regarding the CSP's yes, no, or not applicable responses to the different control group questions posed in accordance to CSA's CAIQ pattern, however certain CSP's did not provide any supporting information for their CAIQ responses. To understand the context of the textual information present in the *notes* column, we performed some basic text mining operations on them. In this direction, we stripped all the 108 CSP's notes information and created a free flowing text documents, one for each CSP. We then created a corpus for all these 108 text documents in `R` and applied various text mining document pre-processing steps. This involved the general pre-processing steps like removing punctuations, whitespaces, English stop words, and stemming the documents. We also removed certain custom words applicable to our domain, for example removing the name of the CSPs. After finishing the pre-processing steps, we acquired a corpus with 108 text documents, containing 883 unique terms and a 0% sparsity. We followed this with the creation of the document term matrix to initially identify the most frequent terms which is summarized in Table 5. As a result of document stemming, terms like security, secured, securing, and so forth are reduced to its core form "secur". We can note from the results in Table 5 that custom, system, security, and access are the top four most frequent terms.

This indicates that CSPs while elaborating on their policies related to the 16 different control groups as proposed in the CAIQ emphasizes on security of the system that are in place to support the infrastructure. Additionally there is information related to customizations of the client's applications that are hosted on their platform, their policies in terms of access control and so forth. This gives some insights into what the CSPs believe the clients should be aware of when it comes to the policies about their cloud platforms.

However, solely relying on term frequency for understanding the context can be misleading. This is because a particular term like security, systems, or customization can be repeated in several different contexts. Therefore, to obtain a deeper level of granularity in terms of context of the supporting *notes* we performed tokenization on our corpus, considering phrases built from individual terms and identifying their frequencies. In this regard, we utilized the `ngram` analysis. The ngram analysis shows how frequently a particular term is followed by another term. We generated bi-grams, tri-grams, and four-grams from our corpus of 108 CSP text documents. The top results for our obtained bi-grams, tri-grams, and four-grams are summarized in Table 6, Table 7, and Table 8 respectively. For a given row in these tables, the first column illustrates the generated n-gram phrase followed by the term that most frequently follows that specific n-gram in the first column. The last column shows the frequency of occurrence (across all the 108 documents) of the generated n-gram. A few of the frequent contexts from the bi-grams (Table 6) can be associated to groups and employees; their assignment for access controls and encryption keys; CSP service zones.

Table 5. Most Frequent Terms in the Corpus

Term	**Frequency**	**Term**	**Frequency**
custom	16740	system	13284
secur	11448	access	10368
inform	9180	respons	8964

In addition to that, the uptime of the cloud computing infrastructure, details of communicating and discussing security related incidents and policies, and descriptions on security information and event management (SIEM) systems and their protocols.

However, as we increase our tokenization terms i.e. move from bi-grams to four-grams, we notice that the context of presented textual information shifts focus from encryption, identity and access control (as in the case of bi-grams) towards application interface management and security, performing audits, data management and its security (as in the case of tri-grams shown in Table 7), and finally towards actual physical security of the cloud computing infrastructure and their employees (as in the case of four-grams shown in Table 8). Therefore, we can encapsulate the contexts mined from the textual data in the *notes* column into three broad categories: (1) access control and application security, (2) protocols (security and otherwise) used by CSPs and how often they are reviewed, (3) information related to the physical security of cloud infrastructure. Coupling a tokenization feature to the STAR registry will benefit both the CSPs that participate and the clients who will access it. We will discuss the outline of such a prototype and its benefits to CSPs and their client in Section 4.

Mining the textual information present in the *notes* for different CSPs gave us an understanding of the contexts that were presented in them. However, we also wanted to ascertain if the CSP's responses (yes, no, or not applicable) were statistically significant when compared to each other. For this assessment we performed `t-tests` on our data frames accounting for the 295 assessment questions.

Table 6. Top Bi-grams and Their Frequencies

bi-grams	**Next Term**	**Frequency**
group employe	assign	216
zone provid	oper	108
up system	generat	324
communic discuss	secur	108
replic siem	system	324

Table 7. Top Tri-grams and Their Frequencies

tri-grams	Next Term	Frequency
interfac use content	creator	108
perform period review	iptabl	108
data real time	configur	108
begin state implement	tool	108

Table 8. Top Four-grams and Their Frequencies

four-grams	Next Term	Frequency
manag process evidenc within	inform	108
exterior physic secur personnel	enter	108
secur oper personnel anomali	detect	108

Our null hypothesis was if the mean of the yes (or no/not applicable) answers were the same for any two given CSPs in the collection of our 108 CSPs then they will be statistically significant to each other. The t-values were too large to support the null hypothesis and the alternative hypothesis was accepted depicting that there was no significant statistical significance between any of the CSPs with respect to their responses to the assessment questions.

4. ENHANCEMENTS TO STAR

Based on our analysis presented in Section 3.3, in this section we will discuss some enhancements to the STAR registry and the CAIQ documentation which can improve it in general and will also be used to improve some of the outlines of offline risk assessment framework. More specifically, the offline risk assessment framework aims at integrating cloud security domains to risk assessment procedures performed during the design phase of an application which might be migrated to the cloud platform in the future. Therefore, according to the CCM structure (CAIQ v.3.0.1) and the answering patterns of the CSPs, combined with the 2017 user surveys from RightScale, we outline the following evaluation

criterion that should be integrated to the STAR registry framework - (1) Compliance applicability, (2) Client's security emphasis on cloud platform domains, and (3) Relevance of a control group (CGID) and its assessment questions.

The increase in maturity of the cloud platform models have seen a shift in client concerns from security towards cloud compliance, audit reviews, and cost efficiency. Compliances are currently listed in a comprehensive fashion in the CCM documents. A client who is new to this domain will not make much sense of "NIST SP 800-53 R3 SC-7 (1)". This will require them to manually go over the compliance standards which are documented elsewhere. This process, for all the compliances listed for one given CSP, can be tedious and may deter newcomers from the process of cloud migration. In order to better understand compliance applicability, STAR registry can be implemented with a phrase-based description database of its compliance list. This can be supplemented with a schema for comparing it with the CSP's compliances giving further insights into the applicability of compliances with respect to the functionality of a client's application. The aforementioned criterion can be further coupled with client input like - (1) Application functionality (health, e-commerce, government, and so forth), (2) Delivery model (SaaS, PaaS, and IaaS) and Deployment model (public, private, and hybrid).

Application functionality can be used to narrow down the compliance applicability estimation process. This will also make the process a bit more client-centric then what it current state. The process of selection of a CSP to host an application given the category of that application and the type of data it stores and processes, certain compliance standards need to be present on the CSP. For example, a CSP hosting an educational application needs to be FERPA compliant. Similarly, medical applications need HIPAA compliant CSPs and e-commerce needs PCI DSS compliance standards. As such in the evaluation category for compliance applicability, a risk assessment framework should populate the available list of

compliances in CCM documentation with tag phrases elucidating their functionality. These tag phrases coupled with client input for application category will be used to identify the CSPs which are following the required compliances.

Delivery and deployment model client input will be used to shortlist the set of control groups that impact clients, changing the evaluation criteria from the CSPs perspective to the client's requirements. This will be further useful because in the current CAIQ version used by the STAR registry, CSPs which are not IaaS providers have to respond with *no* or *not applicable*. As such control groups not relevant to a deployment model should be nullified. Thereby, companies do not have to answer *not applicable* if the response is based on the fact that they do not provide such functionalities. This will help to focus on the not applicable responses which are actually relevant. Given this improvement, if a CSP states that they are IaaS providers and there is a control group related to IaaS, but the CSP has answered not applicable or no to it; then one can better focus and analyze the impact of such responses. Once the control groups have been shortlisted, a risk assessment framework can use it to better perform a (semi)automated analysis of the evaluation criterion - Relevance of a control group and the answer to its assessment question. In doing so, more preference is given towards the control groups that impact clients more and then perform generic CSP evaluations. To this extent we have analyzed the impact of control groups on cloud platform domains along with its impact on client and/or service provider (Table 2 and 3). Given the nature of the delivery and deployment models used, the division of responsibilities between the client and CSP will vary and a much fine grained evaluation can be performed using the STAR registry by incorporating this information.

The information present in the *notes* section of the CAIQ responses of a CSP in the STAR registry provide details regarding the actions a CSP has taken with respect to an assessment question belonging to a control group. Manual analysis of such information will help a client shortlist the set of CSPs that might meet their functional and non-functional requirements. Nonetheless, this in itself is a tedious task and sometimes this

information might have partial details (or not be present at all) which are not sufficient to make accurate decisions. We reckon one of the reasons behind this could be attributed to the market competitiveness because of which CSPs might be reluctant to provide complete information about their CAIQ assessment. Therefore a workaround to this concern could be to perform privacy preserving document comparisons by encrypting the *notes* section of the STAR registry. Clients can perform analysis on the encrypted information, without having the CSPs to reveal the information to other competing CSPs and the clients itself. Alternatively, one may argue that clients can get such information from CSPs by contacting them. However, CSPs may be reluctant to reveal all their cloud platform information.

If they do, they might have the clients sign a non-disclosure agreement, which ideally is not foolproof. Therefore, CSPs can encrypt some of their more detailed information and outsource it to the STAR registry. With the STAR registry acting as a trusted third party, clients can submit their queries which can then be used to perform document similarity threshold detection, or querying of k nearest CSPs that matches with certain functional requirements of the clients and so forth. In this regard, several works in the literature are present [23] [18] which can adopted and applied to the schema of the STAR registry to obtain the desired outcomes. Furthermore, submission of information using encrypted means can be beneficial with respect to physical security of the cloud infrastructure. The CSPs can make their employees undergo a training routine, the aggregate scores of which can be shared to the STAR registry through encrypted mechanisms. The clients can then query this information to determine physical security aspects like which CSP's employees have scored better or between multiple CSPs when was the last security training carried out.

Additionally, the tokenization performed as a part of our text mining (end of Section 3.3) gave us an insight into the context presented through the textual information by CSPs in the STAR registry. This process can be extended and utilized to implement a queryable data database for STAR registry. Such a registry will enable clients to enter the phrases for which they seek information like application management or access control

policies and so forth. The query feature can be supported by n-gram dictionaries as a result of which the search process can narrow down the set of CSPs which present information whose context matches with that of the client's query context. This kind of process will make the client's assessment task of CSPs much more efficient and effective as compared to manually analyzing the entire STAR registry. Further, n-gram dictionaries can also lay the foundation for prediction algorithms. Thereby, clients who are not fully aware of their exact query terms can be suggested (via. prediction) a set of phrases which will reduce their search uncertainties. This process of a query-able database through tokenization can also benefit CSPs since they can use the client's query to estimate contexts that clients are typically looking to assess and reformat and better align their responses (textual ones) to the needs of the client base.

5. IMPROVEMENTS TO OFFLINE RISK ASSESSMENT FRAMEWORK

The offline risk assessment framework evaluates the security provided by different cloud service providers based on the security risks present in a client's application [19]. Based on its evaluations, it suggests a pareto-optimal cloud migration strategy for the client's application. In doing so, it also takes into account any migration constraints that may arise due to client's finances, legal and technical concerns. A summary of the functional objectives of the offline risk assessment framework is shown in Figure 6.

Traditional risk assessment during design phase of an application addresses the security concerns present in them. To validate a CSP's security, third party security audits can be performed on them, which may increase the service costs and too many third party audits can expose CSP's infrastructure information to outsiders [15] compromising the cloud platform. However, if clients have a way to safeguard their application related to cloud domains that are either within or beyond their control, it will reduce their level of uncertainty about a CSP's security policies. This can be done by performing risk assessments during design phase addressing security threats from hosting an application on

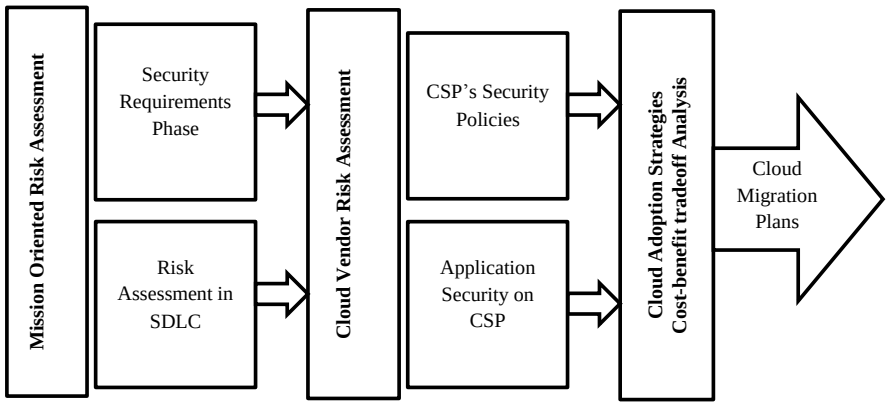

Figure 6. Offline Risk Assessment for CSPs

the cloud. Analysis of the STAR registry gives us insight into how a CSP has responded to questions raised in the CAIQ for a particular control group. Incorporating this information in the risk assessment framework will make the assessment process more effective and cost efficient. This can be achieved by including cloud security issues during the design phase, belonging to a control group of the CAIQ that has not been addressed in detail by the CSP. To do so, data models like *exploit*, *include*, *threaten*, *mitigate* can be developed capturing the application design with incorporated security measures and misuse patterns, different feasible attack patterns, and cloud misuse patterns [10]. The correlation between these data models can be established using techniques such as misuse case diagrams [17]. This can be further extended to include new models developing transition of security threats arising from the cloud misuse patterns into the applications hosted on it. For example, consider the scenario shown in Figure 7 which shows the risk assessment of a hotel reservation web application hosted on a cloud platform using misuse patterns of the web application in isolation and that of the cloud platform.

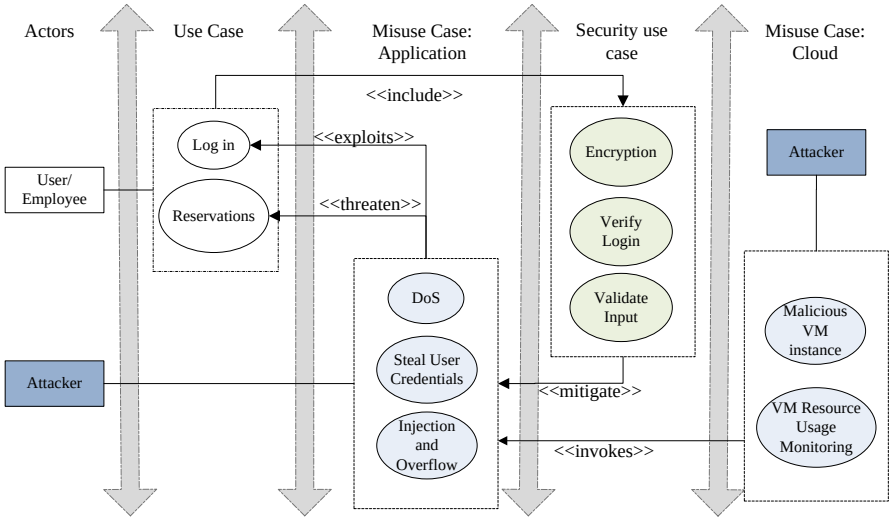

Figure 7. Cloud Security Domain in Design Phase Application Risk Assessment

As we transition from cloud misuse patterns like using *Malicious virtual machine instances* (see *invokes* in Figure 7) to application attack patterns (Malware injection leading to possible theft of user credentials), we will be able to identify threats which might not have been considered if an application was hosted on a private domain. As such, this kind of assessment facilitated by STAR registry evaluations as shown in Table 4 will make clients relatively more prepared in terms of (securely) designing their applications with plans of future cloud migration. This will considerably cut down cost and uncertainties related to cloud security while performing migration.

Furthermore, the Internet of Things (IoT) paradigm has been growing at an unprecedented rate. According to IHS estimates, in 2015, there were 15.4 billion connected devices and it is projected to rise up to 30.7 billion and 75.4 billion by 2020 and 2025, respectively [13]. This will have a notable effect on platforms like Cloud (along with edge and fog computing) computing whose essential feature lies in its capability to pool various computing resources and provide it as a pay-as-you-use model. The presence of vast amount of mobile and embedded devices will change the ways in which clients connect to the cloud computing infrastructures. Therefore, a divergence from accessing the cloud

computing resources from traditional devices to mobile or embedded devices will increase the attack surfaces of the cloud computing infrastructure as well as the client's applications hosted on them. Securing these attack surface rising as a result of these mobile devices becomes even more challenging since many of these devices are resource (energy and computing) constrained. Further concerns arise due to the lack of proper standardization and compliance documentation related to these interconnected mobile and embedded devices. The emphasis to such mobile security concerns has been foreshadowed by CAIQ's mobile security (MOS) control group, which amongst all the 16 control groups has the highest set of questions. However, based on the responses provided by the participating CSPs it is evident that mobile security control group is not a top concerns when compared to control groups like audit, assurance, and compliance (AAC) or datacenter security (DCS).

Current cloud risk assessment frameworks, including the offline risk assessment framework for CSPs, do not take into account the aspect of mobile security which may result in overlooking various security concerns that will eventually arise due to the growth of IoT paradigm affecting the cloud computing infrastructure and applications hosted on it. Thus, we outline the inclusion of the mobile security (as established by cloud control matrix's CAIQ) to improve the functional output of the offline risk assessment framework. Additionally, as discussed in Section 4, some of the enhancements to the STAR registry like tokenization, privacy preserving document comparison, and phrase-based description of compliance applicability can also be applied to improve the functionalities of the offline risk assessment framework. A summary of our analysis results and actionable insights to improve the STAR registry and risk assessment frameworks is given in Table 9.

6. CONCLUSION AND FUTURE WORK

In this paper, we have presented exploratory and descriptive analysis of data from Cloud Security Alliance's (CSA) Security, Trust and Assurance Registry (STAR). We analyzed the answering pattern of various CSPs related to the control groups that has been

Table 9. Analysis Results and Actionable Insights - Summary

Analysis	**Action**	**Benefits**
No specific support for delivery models	Breakdown CAIQ based on question relevance to delivery models	CSPs and Clients
In terms of responses, lack of emphasis on Mobile security	Increase emphasis to Mobile security	CSPs and Clients
Many CSPs make use of other CSP's services	Integrate responses from externally referred CSPs	Clients
Compliances have no information to elaborate functionality	Elucidate compliances with annotations or tag words to evaluate relevance to application	Clients
Manual analysis of notes section of CAIQ can be tedious for a very large sample space	Introduce tokenization and data mining to express context of textual data to narrow down search scope	Clients
No interaction between CSPs and Clients	Incorporate query-able framework wherein clients can express their needs to narrow down search scope	Clients
No interaction between CSPs and Clients	Allow CSPs to access client's query to understand client requirements	CSPs
Lack of detailed information in some sections to make accurate decisions	Present sensitive information in encrypted format and query on it	CSPs and Client
An insight into the emphasis given by CSPs to different security aspect of the Cloud	Incorporate cloud security metrics to design phase application risk assessment	Clients

outlined in the Consensus Assessments Initiative Questionnaire (CAIQ) like auditing, data security, encryption and key management. We also analyzed the context provided in the textual information presented by the CSPs in their STAR assessments by performing n-gram analysis. In doing so, we were able to identify several features that can be incorporated in the STAR registry to help clients more effectively analyze and search for the required security (or related to other domains like auditing) information they need for the decision making process for hosting their applications on a cloud provider's platform. Along with also helping the CSPs better comprehend the needs of clients and customize the information they present to clients accordingly. Additionally, through our analysis we have also outlined some design optimizations that can be incorporated in the offline risk assessment framework to make its evaluations of a CSP more accurate, further reducing the security uncertainties of the client. As a future work, we would like to integrate this with the off-line assessment tool for the complete risk assessment of an application.

REFERENCES

[1] Cloud Security Alliance. Security guidance for critical areas of focus in cloud computing v3.0, 2011.

[2] Cloud Security Alliance. Candidate mappings of iso 27002/27017/27018 to version 3.0.1 of the csa cloud controls matrix (ccm), 2016. [online; accessed 14-Sep-2017].

[3] Cloud Security Alliance. Csa security, trust and assurance registry (star), 2016. [online; accessed 14-Sep-2017].

[4] Claudio A. Ardagna, Rasool Asal, Ernesto Damiani, and Quang Hieu Vu. From security to assurance in the cloud: A survey. *ACM Computing Survey*, 48(1):2:1–2:50, jul 2015.

[5] V. Casola, A. De Benedictis, M. Rak, J. Modic, and M. Erascu. Automatically enforcing security slas in the cloud. *IEEE Transactions on Services Computing*, PP(99):1–1, 2016.

[6] Erdal Cayirci, Alexandr Garaga, Anderson Santana de Oliveira, and Yves Roudier. A risk assessment model for selecting cloud service providers. *Journal of Cloud Computing*, 5(1):14, Sep 2016.

[7] Victor Chang and Muthu Ramachandran. Towards achieving data security with the cloud computing adoption framework. *IEEE Transactions on Services Computing*, 9(1):138–151, 2016.

[8] Luigi Coppolino, Salvatore D'Antonio, Giovanni Mazzeo, and Luigi Romano. Cloud security: Emerging threats and current solutions. *Computers and Electrical Engineering*, 59(Supplement C):126 – 140, 2017.

[9] Sheikh Mahbub Habib, Sebastian Ries, and Max Mühlhäuser and Prabhu Varikkattu. Towards a trust management system for cloud computing marketplaces: using caiq as a trust information source. *Security and Communication Networks*, 7:2185–2200, 2014.

[10] Keiko Hashizume, Nobukazu Yoshioka, and Eduardo B. Fernandez. Misuse patterns for cloud computing. In *Proceedings of the 2nd Asian Conference on Pattern Languages of Programs*, AsianPLoP '11, pages 1–6, 2011.

[11] K. Kritikos, K. Magoutis, and D. Plexousakis. Towards knowledge-based assisted iaas selection. In *2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 431–439, Dec 2016.

[12] S. Lins, S. Schneider, and A. Sunyaev. Trust is good, control is better: Creating secure clouds by continuous auditing. *IEEE Transactions on Cloud Computing*, PP(99):1–1, 2017.

[13] Sam Lucero. Complimentary whitepaper: Iot platforms - enabling the internet of things, Mar 2016.

[14] Jolanda Modic, Ruben Trapero, Ahmed Taha, Jesus Luna, Miha Stopar, and Neeraj Suri. Novel efficient techniques for real-time cloud security assessment. *Computers and Security*, 62:1–18, 2016.

[15] European Network and Information Security Agency (ENISA). Cloud computing: Benefits, risks and recommendations for information security, 2009.

[16] RightScale. State of the cloud report, 2017. [online; accessed 14-Sep-2017].

[17] Lillian Rostad. An extended misuse case notation: Including vulnerabilities and the insider threat. *In Proc. 12th Working Conf. Requirements Eng.: Foundation for Software Quality (R EFSQ)*, 2006.

[18] Bharath K. Samanthula, Yousef Elmehdwi, Gerry Howser, and Sanjay Madria. A secure data sharing and query processing framework via federation of cloud computing. *Journal of Information Systems, Elsevier*, 48(C):196–212, mar 2015.

[19] A. Sen and S. Madria. Off-line risk assessment of cloud service provider. In *2014 IEEE World Congress on Services*, pages 58–65, June 2014.

[20] J. Singh, T. Pasquier, J. Bacon, H. Ko, and D. Eyers. Twenty security considerations for cloud-supported internet of things. *IEEE Internet of Things Journal*, 3(3):269–284, June 2016.

[21] Ahmed Taha, Ruben Trapero, Jesus Luna, and Neeraj Suri. Ahp-based quantitative approach for assessing and comparing cloud security. *2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 00:284–291, 2014.

[22] Changlong Tang and Jiqiang Liu. Selecting a trusted cloud service provider for your saas program. *Journal of Computers and Security*, 50(C):60–73, may 2015.

[23] C. M. Yu, C. Y. Chen, and H. C. Chao. Privacy-preserving multikeyword similarity search over outsourced cloud data. *IEEE Systems Journal*, 11(2):385–394, June 2017.

V. DESIGN PHASE RISK ASSESSMENT OF APPLICATIONS USING CLOUD SECURITY DOMAINS

Amartya Sen and Sanjay Madria

Department of Computer Science

Missouri University of Science and Technology

Rolla, Missouri 65409–0050

Email: {asrp6,madrias}@mst.edu

ABSTRACT

Security of applications on the cloud platform has been one of the primary issues that prevent clients from completely adopting the services provided by various cloud vendors. Although there is a general notion of security present on the cloud platform, it is not addressed with respect to the security threats present in a client's application. To address this concern, the offline risk assessment of cloud service providers framework was proposed earlier. Nonetheless, while performing risk assessment during the software design phase of an application it neither considers cloud security domains as an assessment category nor identifies resources that needs to be protected in the likelihood of a successful attack. In this paper, we present a framework that will address these challenges during the design phase of an application, making client applications adequately prepared for cloud hosting and reducing some of their prevalent security uncertainties related to migrating their application on the cloud platform. Furthermore, the proposed framework will aim to reduce a client's demand for having cloud vendors validate their provided security by performing multiple third party security audits. We present a use-case study showing the applicability of our proposed framework along with discussions about its usefulness.

Keywords: Cloud Computing, Risk Assessment, Software Development Lifecycle, Misuse Patterns

1. INTRODUCTION

Cloud computing platforms through its deployment models (IaaS, PaaS, and SaaS) have provided clients with many notable benefits [21]. These benefits range from not having to own or maintain computing infrastructure to integrating built-in software APIs with a client's application. Although these advantages are well-received in the community, clients are still skeptical to migrate their applications on a cloud platform. This is because of the security concerns arising due to applications being migrated from the secure boundaries of a client's private network to a more or less untrusted domain of a cloud service provider (CSP). The CSPs do provide a notion of security but it does not address the fact that how relevant they will be with respect to the threats present in a client's application. This concern was addressed by proposing the offline risk assessment framework for CSPs in [35].

The mission oriented risk assessment methodology in the offline risk assessment framework addresses the security evaluation of an application during its software development lifecycle (SDLC) phase. This is done by evaluating different components of an application like processes, databases, or users interacting with the application; collectively known as *application elements*, using well known techniques like Microsoft's STRIDE[5]. Security risk evaluation during SDLC phase is economical in nature as it allows clients to effectively evaluate the consequences of different implementation architectures. However, before one can start the secure SDLC process the security requirements specifications phase [17] needs to be addressed. In this phase, design and security teams specify the critical assets of an application, their assumed asset level (high, medium, or low), and security measures that should be implemented.

Nevertheless, there are two potential shortcomings in this method. First, it does not consider the successful execution of an attack due to which multiple application elements can be left exposed to further exploits. As such formulation of security measures should consider the spread of attacks in an application and identifying critical elements.

[5]msdn.microsoft.com/ee823878.aspx

These elements needs to be secured in a way such that it halts the spread of attacks and minimizes damage. Secondly, assumed asset level outlines the security priority of different application elements. However, existing technique [23] of using qualitative classification (high, medium, or low) is not capable of answering the following - “In the event of a security exploit, should we give all the assets belonging to the same asset level equal priority?”

Furthermore, when an application is migrated on the cloud platform, CSPs provide some basic level of security related to domains that are beyond the clientâĂŹs control [8]. A way for clients to validate this is by leveraging third party security audits. This might give rise to two foreseeable concerns. First, CSPs might have to spend extra dollars to have these audits performed which in turn may increase their service costs. Secondly, performing too many third party audits will reveal information related to cloud infrastructure (security and otherwise) to many outsiders [1] and if used maliciously, may compromise the cloud platform. Although, if clients have a way to safeguard their application related to the domains that are beyond their control, it will reduce the level of their uncertainty about a CSP’s security protocols. This can be achieved by securely designing an application accounting for the threats originating from the cloud platform.

To address the aforementioned challenges, we propose a novel design phase risk assessment framework extending the mission oriented risk assessment methodology by including cloud security domains during the SDLC phase. We do so by considering semi-automated generation of cloud misuse patterns as proposed in [18] and integrating it with security evaluation of an application’s data flow diagram (DFD). Along this direction, we can identify those application elements which needs to be protected in the event of a successful attack either arising from the cloud platform or due to the application’s inherent nature of being a web-application. Our proposed framework will help in identifying critical application elements, providing a deeper level of granularity to the asset level classification technique and thereby improving the design of early intervention policies in the event of a successful attack on an application.

Our contributions in this work are as follows:

- Designing and developing a novel design phase risk assessment framework by introducing cloud security metrics.
- Identifying critical application elements to be protected in the event of different attacks, and providing a deeper level of granularity to asset level classification technique.
- Simulating the proposed framework on a use case scenario analyzing the effectiveness of considering cloud security metrics during the design phase risk assessment of an application.

The rest of the paper is organized as follows. In Section 2 we introduce our use case scenario and briefly discuss the outlines of the previously proposed offline risk assessment framework. In Section 2.1, we present the outlines of our proposed framework discussing our threat model and relevant algorithms. Thereafter, we show the applicability of our proposed framework by presenting the obtained results from our use case scenario and discussing its physical significance in Section 3. We further discuss the usability of our proposed framework in Section 4. In Section 2 we discuss some of the significant literature with respect to risk assessment performed for software applications and the cloud platform. Finally, we conclude the paper in Section 5.

2. PROPOSED FRAMEWORK

An effective way to assess the security provided by different CSPs is to compare them with respect to the security risks present in an application that will be hosted on their platform. To do so, one must be able to identify the security risks present in an application and determine if CSPs address them using their security measures. This concept is addressed by the previously proposed offline risk assessment framework for CSPs [35].

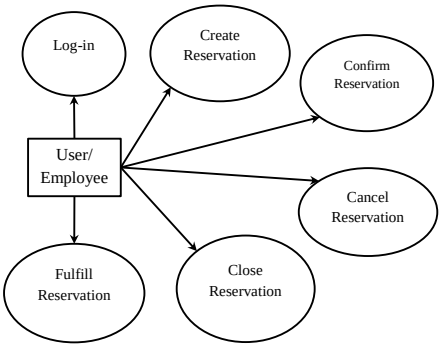

Figure 1. Hotel Reservation - Use Case Diagram

The framework identifies security risks that are present and determines whether it would be cost efficient to perform cloud migration on a CSP. These tasks are facilitated with the help of three independent but correlated modules: (1) mission oriented risk assessment, (2) cloud vendor security assessment, and (3) cloud adoption strategies.

Our proposed risk assessment framework's primary objective is to address the challenges in mission oriented risk assessment methodology as discussed in Section 1. We have designed our framework to be generic enough such that it can applied to any application scenario, however for a better understanding we will elaborate it using a use case scenario that depicts a web application framework for a hotel management, in particular the process of reserving a hotel room. Users or hotel management employees can access the API which consists of three functionalities: log-in, room information, and managing reservations. Access to the log-in page by users will follow a log-in verification operation. The authentication data is stored in a Log-in database. Once logged in, a user can perform room reservations. This information gets stored in a Reservation database. Operations during the reservation phase involve creating, confirming, canceling, fulfilling, and/or closing the room reservations. In case a reservation is fulfilled, the payment information (credit card) is validated through a trusted external entity, Bank. The actions that can be taken by legitimate users and employees are summarized in the use case diagram shown in Figure 1 and its corresponding data flow diagram is given in Figure 2.

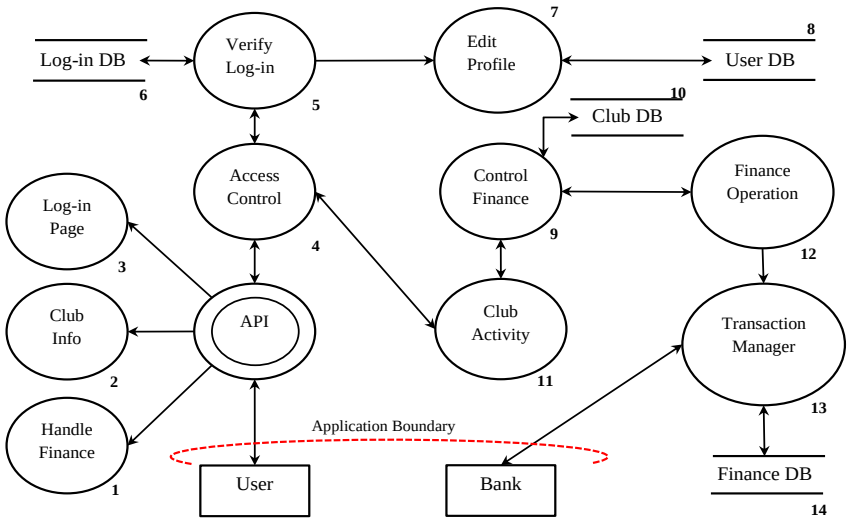

Figure 2. Use Case Scenario: Data Flow Diagram for Hotel Room Reservation Web Application

In the following sections, we will be referring to this application scenario while describing in detail the different components of our proposed framework.

2.1. Mission-Oriented Risk Assessment. The mission oriented risk assessment module [35] semi-automates performing risk assessment during the design phase of an application. Nevertheless, the whole approach is based on an optimistic outlook. In other words, its security requirements phase only collects information primarily related to the domains of security objectives, critical security assets, their asset levels, and security measures to be implemented. Such an optimistic outlook does not consider identifying critical application element(s) which needs to be secured during a successfully executed attack scenario to minimize the incurred damages. This assertion is based on the fact that one cannot guarantee the complete safety of an application. Although the implemented security measures (identified through risk assessments performed in the design and testing phase) will be able to protect against some attacks, it does not assure that other (unknown)

attacks with similar exploitation objectives will not be able to bypass the implemented security measures. To understand this motivation, let us consider a very näive example of cross-site scripting (XSS) attacks on web applications.

XSS attack [3] in its simplest form, can be used to inject javascript code in input fields from client side which is executed on the server side to cause malicious activities. For example, an attacker can use the *Log-in Page* (Figure 2) to inject malicious scripts which needs to be submitted and validated against the *Log-in DB*. If the attack remains unchecked it can return log in credentials leading to elevation of privilege. An example of such an injected script is shown in eq. 1. Although it alerts the message *Hello, World*, attackers can inject much more malicious code. To counter this, a security measure could be to apply techniques such as *filtering* or *obfuscation* as shown in eq. 2. For example in filtering, double quotes within the script tags will be filtered out thereby nullifying the attack attempt. This security measure can be bypassed by using ASCII characters instead of using characters like single and double quotes or backslashes therefore bypassing the implemented filtering technique (eq. 3). With this example, our intention is not to demonstrate that XSS can be bypassed as other security measures are available to prevent it. However, we use this instantiation to depict that no set of security measures (either used disjunctively or conjunctively) can assure complete safety. As such, clients should be prepared with certain *early intervention policies* (EIPs) to minimize the damage in the event of a successful attack execution.

$$< script > alert(\text{“}Hello, World\text{”}) < /script > \tag{1}$$

$$< script > alert(\textbf{filtered-empty body}) < /script > \tag{2}$$

$$\begin{aligned} &< script > String.fromCharCode(\\ &\quad 97, 108, 101, 114, 116, 40, 34, 72, 101, 108, 108 \\ &\quad 111, 44, 32, 87, 111, 114, 108, 100, 34, 41) \\ &< /script > \end{aligned} \tag{3}$$

Nonetheless, clients operate under budgetary constraints which limits the amount of security resources that can be incorporated in an application. Therefore, a pre-requisite to developing early intervention policies is to identify application elements prioritized according to their level of exploitation in the event of successful attacks. Hence, through our proposed framework, our objective is to perform analysis identifying critical applications elements that needs to be protected for different attack patterns such that the limited security resources can be allocated in timely fashion minimizing the spread and damage due to the attacks.

To identify these critical applications elements (CAEs), we incorporate attack pattern category database such as Common Attack Pattern Enumeration and Classification (CAPEC) along with the application design to generate possible misuse cases. These misuse cases coupled with implemented security measures will be used to develop probabilistic state transitions which will be further modeled using the concepts similar to that of *percolation centrality* [31]. Percolation centrality (PC) of a node v in a network with N number of total nodes at a time interval t is computed as follows:

$$PC_v^t = \frac{1}{N-2} \sum_{s \neq v \neq r} \frac{\sigma_{s,r}(v)}{\sigma_{s,r}} \frac{x_s^t}{[\sum x_i^t] - x_v^t} \tag{4}$$

where s and r are the source and destination nodes, respectively. $\sigma_{s,r}(v)$ is the number of shortest paths between s and r passing through v. $\sigma_{s,r}$ is the number of shortest paths between s and r. x_s^t $(0 < x_s^t < 1)$ is the state of infection or spread at the source node during the time interval t. x_v^t is the spread of infection at node v during t. Such concepts are helpful in identifying the relevance of a node in a network with respect to the spread of a virus instead of solely relying on their position in the network topology. Figure 3 summarizes the outlines of our proposed framework with respect to identifying CAEs.

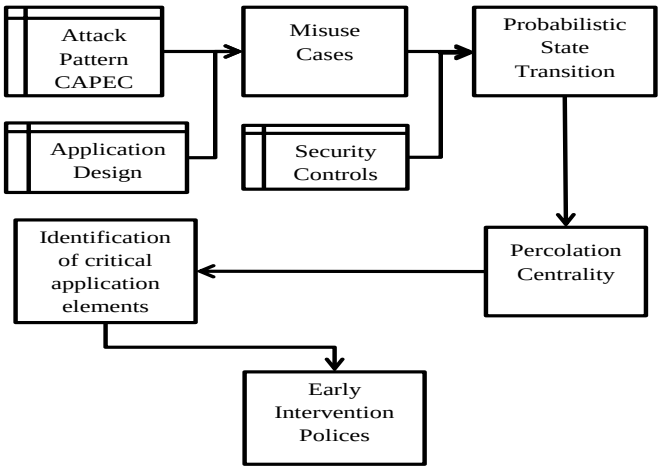

Figure 3. Critical Application Element Assessment

The CAPEC database archives the list of known attack patterns that can be used in order to exploit an application. We have used it to model our misuse cases because even if an attack is unknown (uses a different exploitation mechanism), its overall pattern and objectives will be similar to that of a known attack. Hence, while generating misuse cases we group the attack pattern categories based on their exploitation objective (or pattern). In this regard, we account for CAPEC's *domains of attacks* (view ID: 3000) which lists the different attack categories. The application design block in Fig 3 will consist of input information such as the application DFD (Fig 2), external entities interacting with the application and the use cases associated with them (Figure 1). Additionally, we consider the security threats resulting due to the characteristics of the cloud platform (multi-tenancy, shared resources and so forth). To do so, we incorporate misuse cases of the cloud platform adopted from the works of [18].

2.1.1. Threat model. The threat model with respect to our use case scenario considers the attacker to have intermediary expertise and access to tools and resources (bot scripts) which can help in carrying out the attacks. The exploitation objective of the attacker would be to either compromise the services provided by the hotel management web application (denial of service, CAPEC ID#469) or steal legitimate user credentials in the

form of their identity or financial information. For the latter, attacker might use a collection of different attacks - man in the middle (CAPEC ID#94), command injections (CAPEC ID#248), Simple Script Injection (CAPEC ID#63), Exploitation of Trusted Credentials (CAPEC ID#21; parent of session replay and cross-site request forgery) and so forth.

In the cloud computing platform, the attacker can execute attacks from beyond the network boundaries or pretend to be a legitimate user (malicious insider). As such, attackers may rent virtual instances collocated in the same physical infrastructure as that of our hotel management web application. They can then monitor the resources utilized by the legitimate application and choose to disrupt services by demanding more resources during its peak utilization hours resulting in denial of service. Attackers may also create and submit malicious virtual machine instances to the cloud repository [18] which in turn if used by the legitimate user may result in malware injection. These malwares can then monitor the virtual machine (VM) activities, steal user credentials and relay the information back to the attacker.

Finally, we assume the cloud provider to be a trusted party and they will not collude with the attacker. In accordance to these attributes of our threat model, an instantiation of the generated misuse cases for our use case scenario is shown in Figure 4. The notations used in our misuse case diagram like *include, threaten, exploits,* and *mitigate* has been adopted from the works of [32]. Their utilization is further explained in Algorithm 3 shown in Section 2.1.2.

2.1.2. Integration of cloud domains to application risk assessment. The generation of our misuse case diagrams requires three key input parameters - *AppDesign*; information about the application elements, *AttPattern*; application attack patterns generated from CAPEC, and *CloudExploit*; misuse cases for threats and vulnerabilities specific to the cloud computing platform. Each column in these tables consists of the information shown in Table 1, and the rows represent elements of the application. *AttPattern* consists of attacks exploiting a web application hosted on a private network represented by ID_{wa}.

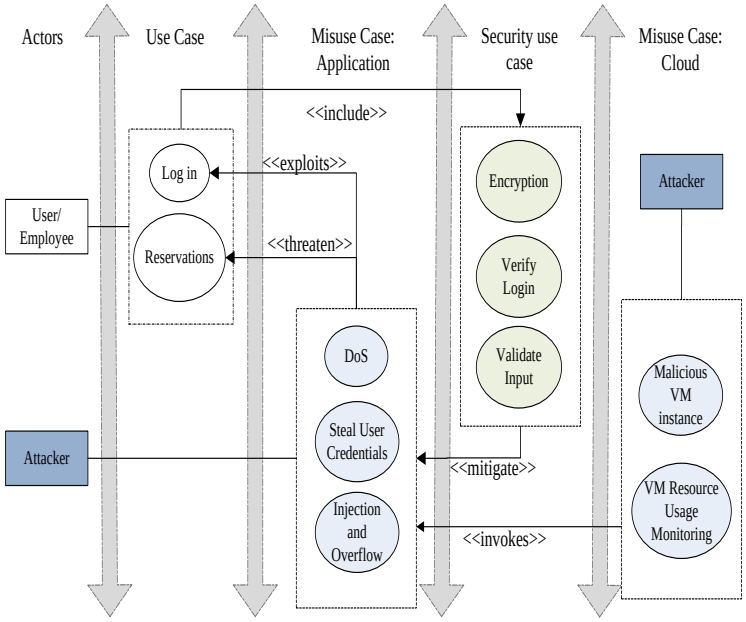

Figure 4. Misuse Case Generation

CloudExploit is composed of attacks originating from the cloud platform which is encapsulated in ID_{cc}. Using the information in AppDesign and AttPattern, and CloudExploit we create five more data structures which will be required to generate the misuse case diagrams in our proposed risk assessment framework. The four traditional well-known misuse pattern notations [32] are *exploit*, *include*, *threaten*, and *mitigate*.

Table 1. AppDesign, AttPattern, and CloudExploit Attributes

AttPattern	**AppDesign**	**CloudExploit**
ID_{wa}, Name	Element Name, Type	ID_{cc}, Name
Prerequisite Resources	Asset level (High,Medium,Low)	Prerequisite Resources
Likelihood and Severity	Interaction (forms, file upload)	Consequences
Mitigation	Security Measures	Mitigation

Additionally, we introduce a fifth one - *invoke* (see Figure 4) which will be used for the integration of the threats originating from the cloud platform into the application. The generation of these data structures is shown in Algorithm 3.

Algorithm 3 Misuse Case Generation

Require: AppDesign, CloudExploit, and AttPattern
Ensure: Misuse case data structures
1: $<< include >>$:
2: **for all** elem1 ∈ AppDesign **do**
3: **for all** elem2 ∈ AppDesign **do**
4: **if** elem1 → elem2 **then**
5: *include* ← (elem1, elem2)
6: **end if**
7: **end for**
8: **end for**
9: $<< invoke >>$:
10: **for all** ID_{cc} ∈ (CloudExploit) **do**
11: **for all** ID_{wa} ∈ AttPattern **do**
12: **if** (ID_{cc}.*Consequence* == ID_{wa}.*PrereqResource*) **then**
13: *invoke* ← (ID_{cc}, ID_{wa})
14: **end if**
15: **end for**
16: **end for**
17: $<< exploit >>$:
18: **for all** elem ∈ AppDesign **do**
19: **for all** ID_{wa} ∈ AttPattern **do**
20: **if** (*PrereqResource* == *Interaction*) **then**
21: *exploit* ← (ID_{wa}, elem)
22: **end if**
23: **end for**
24: **end for**
25: $<< threaten >>$:
26: **for all** elem2 ∈ *exploit* && *include* **do**
27: *include* ← (ID_{wa}, elem1)
28: **end for**
29: $<< mitigate >>$:
30: **for all** elem ∈ *exploit* && AppDesign **do**
31: **if** (SecurityMeasure → elem) && (SecurityMeasure → prereqResource) **then**
32: *mitigate* ← (elem, ID_{wa})
33: **end if**
34: **end for**
35: **return** *include, invoke, exploit, threaten, mitigate*

The *include* data structure is generated by traversing all the elements in AppDesign. If there is an edge between the application elements - $elem1$ and $elem2$, then those element pairs are registered in *include* (lines 2 to 8). We then traverse the misuse patterns present in CloudExploit and estimate if the consequences of their successful exploitation ($ID_{cc}.Consequence$) might result in the satisfying the prerequisite of any attack pattern in AttPattern ($ID_{wa}.PrereqResource$) (lines 10 to 12). If it does, we register the cloud misuse pattern and application attack pattern in *invoke* (line 13). For the *exploit* data structure, our framework traverses all the attack patterns stored in AttPattern for each application element in AppDesign (line 18 and 19). For a given attack pattern and application element, if the resource required to execute the attack is available in the interaction type of the element

(line 20) then the attack pattern along with the application element is registered in the exploit data structure (line 21). The *threaten* data structure is then created by using *exploit* and *include*. Given an element present in *exploit*, if it also present in *include*, then the corresponding attack pattern and connected element is added to the threaten data structure (lines 26 to 28). Finally to create *mitigate*, our algorithm uses AppDesign and *exploit* to identify the security measures that the elements have and checks if those security measures suppresses the required resources to execute the attacks exploiting those elements (line 31). If true, the element (along with its security measure) and corresponding attack patterns are added to the mitigation data structure (line 32).

The inclusion of the *invoke* data structure introduces the cloud security domains to the application security assessment procedure. As we transition from the cloud misuse patterns to the application attack patterns ($<< invoke >>$ in Figure 4), we can identify the threats which might not have been considered if an application was hosted on the client's private network. As such, this kind of assessment will make clients relatively more prepared in terms of (securely) designing their applications with plans of future cloud migration. This will in turn considerably cut down the cost and uncertainties related to cloud security.

Once the misuse case diagrams have been generated, we model them as state transition diagrams to estimate the probabilistic spread of an attack in the event of their successful execution. This is done by taking into consideration the severity and likelihood information of the attack patterns present in AttPattern (Table 1). The state transition diagram shown in Figure 5 for our use case scenario (Figure 2) depicts the routes that an attacker (or legitimate user) can take across the application elements present in the system. In our use case scenario for a hotel management web application's reservation process, the usage starts with the log in state for a user (if used by employee, they can directly access the next states with their credentials). From log in, one can transition to creating a room reservation (or accessing an already created one). This can be followed up with confirming the reservation, canceling it, or fulfilling it.

Confirming a reservation can be followed up with either fulfilling it or closing it. Once a reservation is fulfilled, it will be closed. From the canceled or closed states the application usage reaches the stop state.

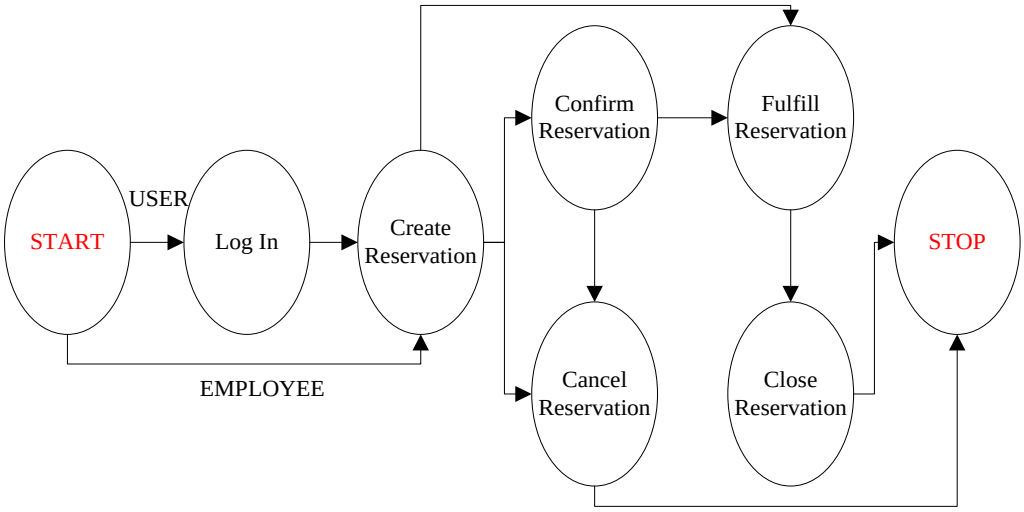

Figure 5. State Transition Diagram for Use Case Scenario in Figure 2

The state transition diagram will then be used to depict the flow of different attacks in the application. This information will be used to determine that - given a particular attack pattern, which application elements are most susceptible (or critical) and thus, should be protected in the event of an attack detection belonging to that pattern. These kinds of assessments are necessary since it helps in the decision making process of effectively allocating security measures for resource constrained clients. An intuitive approach in this assessment would be to identify the nodes (application elements represented as DFD) which have the highest connectivity, i.e. to measure the betweenness centrality of nodes. Although, relying just on the topography of the graph (DFD) will not yield comprehensive results since there is also a need to identify the nodes which are attributing most towards the spread of the attack. In order to evaluate this we model our state transition diagram (Figure 5) by adopting techniques similar to the concepts of *percolation centrality* [31] as defined previously in eq. 4.

2.1.3. Identification of critical application elements. For our framework, in order to model our state transition diagram (Figure 5) estimating the level of exploitation on a particular node, we need to assess the following factors:

- Level of exploitation for a node.
- Transmission probability of an attack from one node to another.

To estimate the above factors, we will use the qualitative information related to the likelihood and impact of attacks available from CAPEC and store it in our *AttPattern* table (Table 1). In CAPEC, likelihood of an attack depicts the probability of success of an attack, taking into consideration factors like attack prerequisite, skills and resources required, available and implemented security measures and so on. Similarly, attack severity defines the average impact of an attack on the targeted software which in conjunction with confidentiality, integrity, and availability (CIA) impact can be used to determine the overall impact of a successful attack. These CAPEC categories are qualitatively scored as *very low, low, medium, high,* and *very high*. We will use attack severity and CIA impact to estimate the level of exploitation of the nodes in our transition diagram (Figure 5) and the attack likelihood to estimate the transmission probabilities. As such, we do not assume that if a safe node is in contact with an exploited node, then it will also be compromised; we account for the attack's complexity and application design for such evaluations.

To quantitatively use the CAPEC's qualitative ratings in our percolation centrality computations, we assume a subjective rating scale of $0 \rightarrow 1$ and adopt the concepts similar to [28] to generate the required rating scales as shown in Table 2. Given our rating scale ($0 \rightarrow 1$), we have assumed an error delta of ± 0.1 units. We model the state of the attacks on a node in the state transition diagram (Figure 5) based on the concepts presented in [30] for infection spread across a network. In our context, we assume that at any given time interval for a node attacks can be in one of the following states - *exploited, safe,* or *exposed*. An attack in exploited state conveys that it has been successfully executed on that node.

Table 2. Quantitative Ratings for CAPEC Qualitative Scores

Qualitative Score	Quantitative Scale	Rating value
Very Low	0-0.2	0.1
Low	0.2-0.4	0.3
Medium	0.4-0.6	0.5
High	0.6-0.8	0.7
Very High	0.8-1	0.9

A safe state will indicate that a node has not been compromised by the attack, and exposed state depicts that there has not been any evidence of a successful attack, but the chances are high. We summarize these states and their state transition matrix setup in Figure 6.

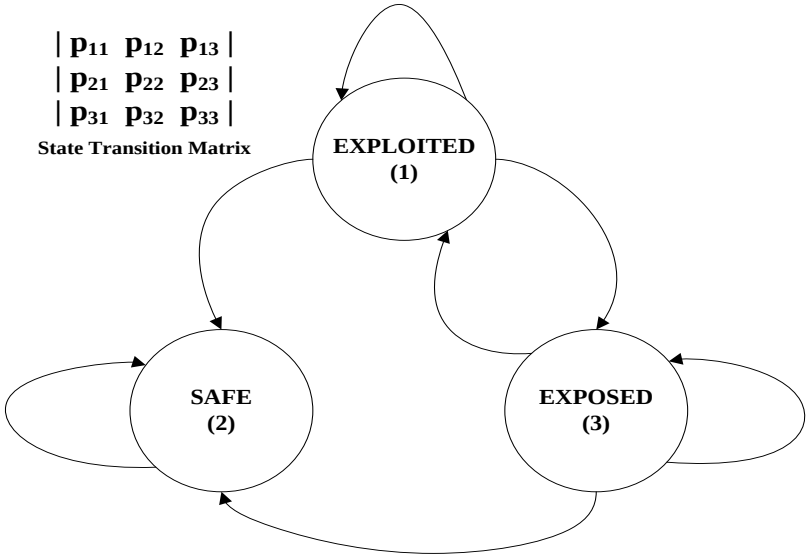

Figure 6. Attack States and Transitions

For the sake of simplicity in our estimations, we assume that in each time interval, an attacker will traverse a distance of one-hop in the state transition diagram (Figure 5) and there will be no backtracking (an attack on a node in exploited state will not transition to safe or exposed state).

Accordingly, projecting Figure 6 on Figure 5; for a node with an attack in the exploited state in a given time interval, can either transition to exploited, safe or exposed states on its connected forward node. If an attack on a node is in safe state it will also be in safe state on the connected forward node in the next time interval. Finally, for an attack on a node in exposed state can transition into either the exposed, safe, or exploited state on the connected forward node. Hence, we can update our state transition matrix as:

$$\begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ 0 & p_{22} & 0 \\ p_{31} & p_{32} & p_{33} \end{bmatrix} \tag{5}$$

In eq. 5, a value of $p_{ij} = 0$ indicates that there can be no transition from state i to j. The remaining transition probabilities will be estimated using the likelihood of success (given by l) of an attack k computed as $(P(l)_k)$, a subjective *security bias* factor (μ_j) estimating the impact of a security control to suppress an attack on a node j, and a *balancing weight* (ω_i) based on the state (exploited, safe, or exposed) of the attack on the previously connected node(s) i, as shown below:

$$P(s)_{ij} = \omega_i \times \left(P(l)_k * \mu_j\right), \forall k \in A_K \tag{6}$$

where $P(s)_{ij}$ is estimated transition probability of an attack between two nodes, v_i and v_j having states i to j and A_K is the total number of attacks under consideration for the current node, v_j. The security bias factor μ_j ranges between $0 \rightarrow 1$, where 0 conveys the belief that the implemented security measure can suppress the likelihood of the attack(s) in

A_K completely and 1 indicates that the security measures has no impact on the attack(s). Practically, the impact of μ_j can be evaluated following the guidelines laid down for security measures in [2]. The balancing weight $\omega_i \in \{Ex, S, Ep\}$, where Ex, S, Ep correspond to Exploit, Safe, and Exposed state respectively. The estimate of $P(s)_{ij}$ will lie between $0 \rightarrow 1$.

We subjectively divide this into three ranges to determine the ω_i value of the current node. Since each of the nodes will correspond to an application element, one can determine these ranges based on the specified *asset level* (Table 1 - high, medium, or low) of the element during the security requirements phase. For example, an instantiation of these ranges for a *high* security asset could be as follows:

$$P(s)_{ij} = \begin{cases} 0 \rightarrow 0.1, \omega_j \in Safe \wedge \omega_j = 0 \\ 0.1 \rightarrow 0.3, \omega_j \in Exposed \wedge \omega_j = 0.5 \\ 0.3 \rightarrow 1, \omega_j \in Exploited \wedge \omega_j = 1 \end{cases} \tag{7}$$

These ranges can be adjusted according to the security objectives of a client. Therefore based on $P(s)_{ij}$, if an attack's state on the current node v in a time interval t is estimated to be in exploited state, then the level exploitation (x_v^t) can be estimated as follows:

$$\frac{1}{A_K}\left(\sum_{\forall A_K} Sev_{v_j} \times \left(\sum_{\forall a_k \in A_K} NodeImpact\right)\right) \tag{8}$$

where A_K is the total attacks under consideration for the current node, v_j. Its quantitative value can be estimated based on the rating scale shown in Table 2. a_k is an attack in A_K. Sev_{v_i} and Sev_{cur} is the net exploitation amount of the previous and current node, respectively. The obtained value of x_v^t is normalized over the total number of attacks under consideration in order to keep the final output in the range of $0 \rightarrow 1$. $NodeImpact$ is an

application specific criteria which can be estimated as follows:

$$AssetLevel \times \left(\frac{1}{A_K} \sum_{\forall a_k \in A_K} \frac{C_{a_k} + I_{a_k} + Av_{a_k}}{3} \right) \tag{9}$$

where $AssetLevel \in \{high, medium, low\}$, is the security relevance of the current node as specified by the clients. $C_{a_k}, I_{a_k}, Av_{a_k} \in \{high, medium, low\}$ is the successful impact of the current attack $a_k \in A_K$ on confidentiality (C), integrity (I), and availability (Av). This is further normalized to keep the final estimate within the selected quantitative scale. It can be noted that if the previous node's $\omega_i \in Safe$, $Sev_{v_j} = 0$, i.e. severity of the current node connected to a safe node will be zero.

Finally, we need to update the level of exploitation computed in eq. 8 by including the impact of the cloud platform misuse patterns which invokes the application attack patterns (Figure 4). To do so, we use the result of the surveys conducted by cloud security alliance (CSA) with regards to the top 12 cloud security threats and how they affect different cloud security domains as listed in CSA's cloud controls matrix documentation. A good publicly available source in this domain can be found on Cloud Security Alliance's (CSA) Security, Trust & Assurance Registry (STAR)[6]. This registry is based on CSA's Cloud Controls Matrix (CCM) document [4], which provides security guidelines across sixteen cloud security domains. These guidelines can be used by CSPs to provide security assessment of their services and by client's to evaluate the security guidelines present with a particular CSP. Some examples of the security domains used in CCM are *Application and Interface Security*, *Audit Assurance and Compliance* to *Identity and Access Management* and so on. Each of these domains are broken down into their subcategories accompanied by their control specification which helps in elaborating what kind of security measures and compliance should addressed in these domains. CCM also lists the impact of these control specifications on the cloud platform's architectural domains like, *Physical infrastructure,*

[6]https://cloudsecurityalliance.org/star/

Networking, Computing, Storage, Application, and Data as well as its applicability to delivery models like, *SaaS, PaaS, and IaaS*. We consider CCM version 3.0.1 which consists of 133 total control specifications (16 domains and their subcategories). For more detailed analysis, readers are kindly referred to work in [36].

We utilized the results presented in [36] to create a list (Table 3) showing the importance of cloud security domains in terms of their security relevance. The *score*

Table 3. Cloud Domains Security Relevance

Domain	**Score**
Identity and Access Management	8
Virtualization, Encryption and Key Management, Application Security, Information Management and Data Security	7
Incident Response, Traditional Security Business Continuity and Disaster Recovery, Cloud Computing Architectural Framework	5
Data Center Operations, Governance and Enterprise Risk Management	4
Interoperability and Portability, Legal Issues	2
Security as a Service, Compliance and Audit Management	1

column in Table 3 shows how many of the 12 cloud security threats affects the given cloud security domains. For most of these cloud security domains, a cloud service provider (CSP) does provide some level of security. Nonetheless, a client does have some responsibilities related to them. We shortlist these domains and create our own ranked list in Table 4 using the *scores* given in Table 3.

Table 4. Cloud Security Domain Ranked List

Domain	**Rank**
Identity and Access Management	5
Virtualization, Encryption and Key Management, Application Security, Information Management and Data Security	4
Disaster Recovery	3
Legal Issues	2
Compliance	1

We estimate the impact of our cloud misuse patterns on these cloud security domains and update eq. 8 to compute the net exploitation level of an application element by multiplying the normalized *cloud security domain impact* or $CSDI$ using the ranks of the cloud security domain as follows:

$$(x_v^t)_{net} = \frac{(CSDI \times x_v^t)}{TotalDomains} \tag{10}$$

where $TotalDomains$ is the total number of unique cloud security domains affected in the computation of x_v^t for a node v in the application. The resulting output of our framework will then be a ordered list of application elements having a $(x_v^t)_{net}$ value ranging from $0 \rightarrow 5$. This value will depict their importance in terms of security evaluation after taking into consideration the feasibility of attacks that can exploit it when either hosted on a private network or a cloud platform. We demonstrate the results of applying our proposed framework to the use case scenario in Section 3.

2.2. Complexity Analysis and Scalability. In this section, we discuss the complexity involved in designing our proposed risk assessment framework and its scalability with respect to large scale application scenarios.

The initial steps involves the input of the application design using data flow diagrams and creation of a database using algorithm 3. The database consists of information like *include, invoke, exploit, threaten,* and *mitigate*. The generation of these data structures is upper bounded by $O(n^2)$ (for e.g. algorithm 3, lines 4 to 18). Although for an application having a set amount of security measures and assuming that the number of entries in attack pattern together with cloud exploits (Table 1) over a period of time remains constant, we can state that the generation of our database takes place in a constant time. The generation and input of these information is followed by the estimations of the net exploitation levels ($(x_v^t)_{net}$) by the traversing state transition diagrams as shown in Figure 5. These state transition diagrams can be considered to be directed acyclic graphs and given the methodologies

used by our proposed risk assessment framework, an optimal way to traverse it will be to apply breadth first search. For a graph consisting of V vertices and E number of edges, the complexity of breadth first search is upper bounded by $O(|V| + |E|)$ which is linear time. As we visit each node, we compute the transition probabilities ($P(s)_{ij}$) and net exploitation level of each node which takes constant time. Hence, assuming uniform attack patterns and cloud exploits, our framework scales linearly as the size and complexity of the applications increase.

On the contrary if we keep the application design constant and vary attack patterns, cloud exploits, and security measures by taking into consideration different threat models simultaneously, then the complexity and scalability of our risk assessment framework will be squared. In a more pragmatic setup, we reckon that an application's functionality and security objectives drive the establishment of the attack models and not vice versa. Therefore, it is more likely that future iterations of the design process will add more application elements while keeping uniform attack models and security objectives.

3. SIMULATION AND RESULTS

In this section, we present the simulation results of our proposed risk assessment framework applied to the use case scenario for a hotel management web application as depicted in Figure 2. The primary input to our framework will be the state transition diagram of our use case scenario which is obtained as explained in Section 2.1.2 and shown in Figure 5. According to algorithm 3, we generate misuse patterns for our simulations (Figure 4) which results in the set of the attacks as shown in Table 5.

The listed attacks having the superscript *1* originates in the application due to cloud misuse pattern - malicious virtual machine instance. Whereas, attacks with superscript *2* is as a result of the misuse pattern - resource monitoring by adversary of collocated virtual machines on the cloud platform. Remaining attacks can occur in the application due its

Table 5. Attack Set for Simulation

Attack	**CAPEC ID**	**Attack**	**CAPEC ID**
Targeted Malware Attack[1]	542	Footprinting[1]	169
Directed Internal Reconnaissance[1]	529	Man in the Middle[2]	94
XSS through log files[2]	106	DoS - Excessive resource allocation[2]	130
Embedding scripts in non-script element	18	Embedding scripts in http headers	86
Command injection OS command injection	88	Command injection LDAP injection	136
Command injection SQL injection	66	Http DoS (outside cloud boundaries)	469
Simple Script Injection	63	MitM (outside cloud boundaries)	94

nature of being a web application, irrespective of being hosted privately or on the cloud platform. We aggregate the severity, likelihood and CIA impact values of these attacks from CAPEC and obtain their quantitative estimates according to our rating scales (Table 2).

While applying the risk assessment methodologies to Figure 5, we assume the *start* node (application API) is susceptible to the attack set (A_K) (Table 5). As such, initially the *start* node is assumed to be in the *exploited* state. Even though our framework will help clients evaluate the impact of their incorporated security measures (μ_j in eq. 6), its primary focus is the prioritized identification of critical application elements which needs to be protected in the event of different attack executions. The feasibility of this evaluation relies on assuming the *start* node to be in the exploited state for all the attacks in A_K. The remaining transition probabilities ($P(s)_{ij}$) is computed according to eq. 6, which for transition of an attack $k \in A_K$ between nodes, $v_i \rightarrow v_j$, considers the likelihood of the attack ($P(l)_k$), state of attack in v_i, $\omega_i \in \{S, Ep, Ex\}$ (eq. 7), and the security control μ_j present for attack k on node v_j.

Table 6. Asset Type and ω_i Ranges for Attacks

	High	**Medium**	**Low**	ω_i
$P(s)_{ij}$	$0 \rightarrow 0.1$	$0 \rightarrow 0.3$	$0 \rightarrow 0.7$	S
	$0.1 \rightarrow 0.3$	$0.3 \rightarrow 0.7$	$0.7 \rightarrow 0.9$	Ep
	$0.3 \rightarrow 1$	$0.7 \rightarrow 1$	$0.9 \rightarrow 1$	Ex

For the sake of simplicity in our simulations, we have randomly selected the value of μ_j for the attacks from a continuous probability distribution of $0 \rightarrow 1$, where a value of 0 will indicate that the security measure is able to suppress the attack completely and a value of 1 means that the security measure has no impact on the attack. For practical scenarios, one can estimate μ_j more accurately according the concepts of security measure evaluation as given in [2]. Based on the estimates of the transition probabilities for the attacks and the asset level of the node v_j, we will shortlist the set of attacks that can exploit v_j, expose it, or reside in the safe threshold. This shortlist is created according to eq. 7 and we present the ranges for *high, medium*, and *low* assets used for our simulations in Table 6.

In our simulation, the estimations of the assumed asset levels for our nodes in Figure 5 is given in Table 7 along with the reasoning of our selections given the threat model discussed in Section 2.1.1. All the remaining estimation parameters involved in our risk assessment simulation along with their quantitative boundary values are summarized in Table 8.

Let us consider the assessment of the following attacks from Table 5 - *Embedding scripts XSS in http headers*, *Footprinting*, and *XSS through log files*. The likelihood $(P(l)_k)$ for these attacks are 0.7, 0.7, and 0.5 respectively. The estimated μ_j values for these attacks on a simulation run are 0.02, 0.75, and 0.27 respectively. This indicates amount of security on log in node for the considered attacks. Assuming an ω_i value of 1 (estimated state of attacks on *start* node), the transition probability from *start* node to *Log in* node for say *XSS in http headers* attacks will be computed according to eq. 6 as shown in eq. 11. Similarly, transition probabilities for *Footprinting* and *XSS through log files* attacks are - 0.525 and

Table 7. Assumed Asset Levels for Use Case Scenario

Node	**Asset Level**	**Notes**
Log in	High	Coupled with user credentials; source node
Create Reservation	High	If spoofed can occupy empty room slots, denying service to legitimate users
Confirm Reservation	Medium	Not of same value as create; but unwarranted confirmations can still lead to unwanted consequences
Cancel Reservation	Low	Reduces negative outcome even for spoofed reservations. Not much value to adversary
Fulfill Reservation	High	Involves user financial transactions; destination node
Close Reservation	Low	Since it comes after fulfill, at this point there is nothing to exploit

0.135. Once transition probabilities values are obtained, our framework estimates the state of the attacks on the node they transitioned to. This is based on the specified asset level of the nodes (Table 7) and ranges to determine the state (Table 6).

$$\begin{aligned} \omega_i \times \left(P(l)_k * \mu_j\right) &= 1.0 * 0.7 * 0.02 \\ &= 0.014 \end{aligned} \tag{11}$$

Table 8. Simulation Attributes

Attribute	**Notation**	**Values**
Attack likelihood (Continuous)	$P(l)_k$	$0 \rightarrow 1$
Security bias (Continuous)	μ_j	$0 \rightarrow 1$
Balancing weight (Discrete)	ω_i	{0,0.5,1}
Transition probability (Continuous)	$P(s)_{ij}$	$0 \rightarrow 1$
Exploitation level (Continuous)	$(x_v^t)_{net}$	$0 \rightarrow 5$
Asset level (Discrete)	$AssetLevel$	{0.7,0.5,0.3}
CIA impact (Discrete)	C, I, Av	{0.7,0.5,0.3}
Cloud security domain impact (Discrete)	$CSDI$	{1,2,3,4,5}

For our use case scenario, *Log in* node's asset level is *high*. Hence, *XSS in http headers* will fall into the safe state and thus, will not be considered for further transitions. *Footprinting* and *XSS through log files* will fall in exploited and exposed categories, and will be considered as the new attack set from which attacks can transition to the next nodes. Further, in order to compute the net exploitation of a node ($(x_v^t)_{net}$) in state transition diagram (eq. 8, 9, and 10), we consider only the attacks that are present in exploited state for a given node. As such exploitation due to *Footprinting* attack will be computed as shown in eq. 12. These computations will involve normalizations if multiple attacks are present in the exploited attack set. Further, if the attacks in the exploited state are initiated due to the misuse cases of the cloud platform then we update the level of exploitation computed above.

This is done by including the *cloud security domain impact* factor (eq. 10), for e.g. *Footprinting* attack is invoked as a result of *malicious virtual instances*. This misuse pattern can impact the identity and access management domain by injecting malware causing identity theft in the application or leading to misconfiguration of the application's virtual machines.

$$\begin{aligned} x_v^t =& Sev_{v_j} \times \left(AssetLevel \times \left(\frac{C_{a_k} + I_{a_k} + Av_{a_k}}{3} \right) \right) \\ =& 0.1 * \left(0.7 * \left(\frac{(0.5 + 0.3 + 0.3)}{3} \right) \right) \\ =& 0.0252 \end{aligned} \tag{12}$$

Thus, we recompute the level of exploitation by using the ranks of these cloud security domains (Table 4) as follows:

$$\begin{aligned} (x_v^t)_{net} =& \frac{(CSDI \times x_v^t)}{TotalDomains} = ((5 + 4) * 0.0252)/2 \\ =& 0.1134 \end{aligned}$$

Table 9. Simulation Results

Node	**Severity** (x_v^t)	**Cloud Domains** ($CSDI$)	**Net Exploitation** ($(x_v^t)_{net}$)
Log in	0.30	5,4	1.33
Create reservation	0.33	5,4	1.47
Confirm reservation	0.32	4	1.26
Cancel reservation	0	NA	0
Fulfill reservation	0.39	4	1.53
Close reservation	0	NA	0

These processes are repeated until no more attacks remain in the attack set to transition onto the next node or all the nodes in the state transition diagram have been traversed. For cases where a node may have multiple child nodes, like create reservation can be transitioned to from log in as well as start node (Figure 5), our framework will consider the maximum of the ω_i values of an attack from the multiple transitions. To elaborate this, consider an attack which transitions from start node to log in node and gets categorized as safe or exposed state. If categorized as safe, this attack will not be considered for transition from log in node to create reservation node. For exposed, the ω_i value will be taken as 0.5, reducing the transition capabilities of the attack. But if the create reservation node can be directly reached from the start node, the adversary can bypass the security measures present in the log in node. Further, absence of adequate security measures create reservation node, may result in exploit state. In doing so, if this attack was in safe state from log in to create reservation it will be added to the attack set (maximum of $\omega_i \in \{0, 1\}$) or category will be updated to exploited (by taking maximum of $\omega_i \in \{0.5, 1\}$).

Once we have computed x_v^t, we update it by adding the cloud domain impact by identifying the attacks in the attack set belonging the exploited category originating as a result of the cloud misuse patterns. Accordingly, the final obtained results of $(x_v^t)_{net}$ value for our use case scenario nodes are given in Table 9. The *net exploitation* column in Table 9 translates to the amount of threat an application element is susceptible to in the event of

attacks executed from the set of attacks A_K. Sorting *net exploitation* in descending order will yield a prioritized list of critical application elements which needs to be secured in the event of security exploits on the application.

With regards to our use case scenario, the risk assessment framework gives notable relevance in securing *fulfill* reservation. This is followed by *create, log in, and confirm* reservation. Finally, no emphasis is given to *cancel and close* reservation elements. We reckon that this behavior is based on the nature of our use case application wherein the fulfill reservation element is responsible for handling financial transactions. Create reservation is given the second highest relevance for two foreseeable reasons. First, it is the crux of the entire room reservation process making sure that service availability is present for legitimate users. Secondly, it can be directly accessed by employees of the hotel as well as by external users through the log in process. Given the possibilities of outsider and insider attacks on one node in contrast to just outsider attacks on log in node, create reservation in our opinion gets rated higher than log in node. The third node emphasized in the output is log in, for being the entry point of all outsider attacks and also hosting user's authentication facilities. This makes it prone to attacks that would steal user credentials. Confirm reservation is the last on the list which is given some security emphasis. Attacks on this node will aggravate the denial of service situation which can start from the create reservations node. But we reckon that the reason it gets a lower relevance compared to the other nodes is because higher capabilities are required by an attacker to exploit this node as they will need to commit some form of identity to confirm a reservation. Finally, no emphasis is given to cancel and close reservation which could be because an attacker does not stand to gain anything by exploiting these nodes with regards to our assumed threat model.

With respect to our use case scenario, such output validates the security objectives specified in the security requirements phase but it also provides some additional insights into the scenario. First, our framework provides deeper granularity of security emphasis to a set of application elements belonging to the same asset level category. To elaborate

this, let us consider the traditional approach to allocate security resources. It is done based on the asset level (high, medium, or low) of different application elements specified during the security requirements phase. But then the question is - for all elements belonging to a particular asset level should we allocate equal amounts of security resource? One can answer this question by performing a bunch of what-if analysis or using other similar methods. Although, our assessment framework semi-automates this approach. Considering our use case scenario, all three nodes - log in, create reservation, and fulfill reservation are categorized as high asset level. After the output of our framework, we are presented with a more prioritized output - fulfill reservation ($(x_v^t)_{net}$ = 1.53) > create reservation ($(x_v^t)_{net}$ = 1.47) > log in ($(x_v^t)_{net}$ = 1.33) . In terms of decision making of security resource allocation and further assessments, this will provide more clarity to clients.

Secondly, the output also factors in the threats arising due to the cloud platform. The *severity* column in Table 9 shows the security relevance given to the application elements while considering it solely as a web application. In this direction, the impact on confirm reservation (x_v^t = 0.315) is higher than that of log in (x_v^t = 0.296) which in a way is contradictory to the asset level specification of these elements in our use case scenario - confirm reservation: medium and log in: high. However, once we include the effect of the attacks originating from the cloud platform which can exploit the application in the form of *cloud security domain impact* (Table 4, eq. 10), we see that log in ($(x_v^t)_{net}$ = 1.33) gets a higher priority than confirm reservation ($(x_v^t)_{net}$ = 1.26). Hence, aligning the obtained output with the specified security objectives.

4. DISCUSSIONS

In contrast to traditional approaches of hosting an application on a private network, it is more cost efficient for clients to migrate their application to a cloud platform. However, this move also entails security concerns on the client's part regarding their application.

Taking such challenges into consideration, our proposed framework has the added inclusion of cloud security domains in terms of performing risk assessment for their application during the design phase of the software development lifecycle process.

Clients can use our proposed framework to comprehensively assess the security threats it might be exposed to as result of being a web application which might be hosted on a cloud platform in the future. Such kinds of assessment performed during the design phase makes the development process economical for the clients and also helps them prepare against the security uncertainties of hosting their application on a cloud platform. The relevance of our proposed framework can be expressed in two salient features.

First, to help clients estimate the impact of preventative security measures they plan to incorporate in their application. Secondly, to identify critical assets in the application that needs to be protected in the event of successful attack exploitations. The latter being the primary focus of our framework, clients are presented with a list of application elements prioritized according to their level of exploitation by taking into account a feasible set of attacks. Such output list will help in the decision making process of allocating security resources which may be constrained by a client's budget.

Furthermore, if multiple application elements are declared as *high* asset levels during the security requirements phase, it is imperative for clients to decide whether to allocate them with equal amounts of security resources. One way to address this is to use traditional mechanisms like annualized loss expectancy (ALE) [40]. But coupling these mechanisms such as ALE with the output of our framework can be used to gain a deeper understanding of the security impacts on the application elements belonging to the same asset level and how much of that impact is contributed by threats from the cloud platform. These outputs can be further used in conjunction with the functionality of the application elements in question, which will help in making more informed decisions related to security measures allocation.

Additionally, clients can also estimate how much security impact the cloud will have on their application elements. It will help them analyze scenarios like - will the risk of an application element solely increase because of being on the cloud. Knowing what elements could be vulnerable on the cloud at an earlier stage of software development will make clients much more prepared once they migrate their application to the cloud platform. This is beneficial in contrast to patching or rewriting their application elements or having to solely rely on the security provided by cloud vendors.

5. RELATED WORK

In this section, we will briefly discuss the literature review related to our proposed risk assessment framework. In doing so, we have categorized them into two domains - (1) risk assessment of applications in different stages of the software development lifecycle process, and (2) risk assessment of cloud computing platforms.

As presented in [40] [41], risk analysis has become an integral part in the various stages of the software development process. Building in security from the design to the testing phase of a software results in more secure software and makes the overall process cost-efficient. One of the most notable risk assessment works in this regard is [5] leading to the maturation of Microsoft's STRIDE process which is now one of the widely accepted risk assessment standards in the community. Several works followed thereafter to improve the risk assessment process like [38], [25], [6], [42]. Further, authors in [14] have addressed the automation of the risk assessment models during the software design phases. Along these lines, works like [27] have also presented techniques for inclusion of security mechanisms in the various stages of the agile software development process which is now one of popular models in the software development process. In addition to inclusion of security mechanisms in the software design and development stages, several works like [29], [12], [7] have addressed the security aspects in the software testing stages. Their primary focus in this domain being the automated generation of security aware software test cases.

Numerous works are present in the literature that addresses risk assessments methodologies related to the cloud computing domain as surveyed in [20]. Some of the earlier works in this domains discusses the security impacts uniquely arising due to the cloud infrastructure like in [19], which investigates the threats occurring due to the setup and distribution of cloud services, outlining the threats in terms confidentiality, integrity, and availability of cloud infrastructure. Whereas, other risk assessment works discusses security concerns related to very specific domains such as network based attacks [24], and security concerns related to the data residing on the cloud platform [33]. The authors in [39] have also explored into the possibilities of providing risk assessment as a service to the clients contemplating hosting their application on the cloud platform.

Other works like [16], introduces security metrics to service level agreement (SLA) documents such that clients can not only assess the quality of service related to a cloud service they want to rent but also the security risks associated to it. Further with respect to assessing the security of cloud platforms a challenging aspect is to keep up with the dynamic environment of the cloud platform as well as the evolving needs of the clients. In this regard, authors in [13] have proposed a tool for dynamic and flexible service provisioning of cloud services to clients which can account for estimations of parameters such as trust, associated risks, and cost. Such tools will help clients in predicting and adapting efficiently to unanticipated changes in resource requirements on the cloud platform.

In more recent trends, authors in [10] propose a risk assessment framework for cloud computing platforms by assessing the SLA as presented by clients and assessing from the cloud vendor's perspective to check the number of SLA violations that might take place in accordance to the demands presented by the clients in their SLA. Other recent works in risk assessment of cloud service provider as presented in [9], utilizes information gathered from clients and cloud vendors to assess different risk scenarios. They do so by proposing a machine learning framework which leverages the security evaluation documents of various

cloud vendors publicly available on Cloud Security Alliance Security, Trust and Assurance Registry or STAR to present a quantitative rating output to describe the risk associated with different cloud services.

Nevertheless, performing risk assessment requires an actionable output which users can utilize. In this regard, quantitative estimates have leverage over qualitative analysis. Therefore, works can be found across the literature both in terms quantitative risk assessment for software applications [43] and for cloud security [34], [15]. Finally, a pre-requisite to performing risk assessment is to understand and estimate the types of security threat an application (or platform) might be exposed to. In this regards, security attack patterns coupled with misuse cases is a good way to elicit the mechanisms in which an application might be attacked. Security attack patterns for software application have been studied vehemently by the community [26] and its coupling with application abuse [22], [44] and misuse cases [37], [11], [32] have well-found acceptance and feasibility. But in contrast to the well-studied attack and misuse patterns of software applications, cloud misuse patterns [18] are still in its dormant stages and requires further analysis to create a comprehensive knowledge base which can be used in a modular fashion to analyze the risks associated to software applications which will be migrated to the cloud platform.

In comparison to the aforementioned works, our proposed framework attempts to bridge the gap between performing risk assessment for software applications and the cloud platform. In doing so, we integrated cloud security domains to the risk assessment procedure of a software application during its design phase. Our framework not only helps clients to assess the security measures they plan to incorporate in their application but also to identify application elements which needs to be protected in the event of a successful attack. Such estimations will also help to single out the impact of the cloud platform thereby boosting the client's confidence in terms of the security of their application on a cloud platform.

6. CONCLUSION AND FUTURE WORK

In this paper, we have presented a risk assessment framework for applications during its design phase by introducing cloud security domains as an assessment metric. This has been done with the objective of benefiting clients who might consider migrating their application to the cloud platform in the future. In doing so, clients will not only understand the threats that their application might be exposed to, but also how much of that threat is due to hosting of their application on a cloud platform. This will help clients design and develop applications with much more preparedness towards the security uncertainties of the cloud platform.

The presented risk assessment framework will give the clients a list of application elements that needs to be secured in the event of different attack exploits. Using this output clients will be able to allocate security resources appropriately in case they are limited by budgetary constraints. To our knowledge, there is no prior work addressing the design phase risk analysis of applications to be hosted on cloud platforms.

As a part of our future work, we would like to develop a much more comprehensive cloud misuse pattern database and extend our risk assessment framework to develop and suggest efficient security measures addressing the economics involved in the security spending process. We also intend to develop the design phase risk assessment tool in future.

REFERENCES

[1] *Cloud computing: Benefits, risks and recommendations for information security*. European Network and Information Security Agency (ENISA), 2009.

[2] *NIST Special Publication 800-53 Revision 4 Recommended Security Controls for Federal Information Systems and Organizations*. 2012. [online; accessed 31-May-2017].

[3] *OWASP Top 10: The Top 10 Most Critical Web Application Security Threats*. Open Web Application Security Project, CreateSpace Independent Publishing Platform, USA, 2014. [online; accessed 31-May-2017].

[4] *Candidate Mappings of ISO 27002/27017/27018 to version 3.0.1 of the CSA Cloud Controls Matrix (CCM)*. Cloud Security Alliance Summit San Francisco 2016, 2016. [online; accessed 31-May-2017].

[5] Marwan Abi-Antoun, Daniel Wang, and Peter Torr. Checking threat modeling data flow diagrams for implementation conformance and security. In *Proceedings of the Twenty-second IEEE/ACM International Conference on Automated Software Engineering*, ASE '07, pages 393–396. ACM, 2007.

[6] Mohamed Almorsy, John Grundy, and Amani S. Ibrahim. Automated software architecture security risk analysis using formalized signatures. In *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, pages 662–671. IEEE Press, 2013.

[7] J. Bozic and F. Wotawa. Security testing based on attack patterns. In *IEEE 7th International Conference on Software Testing, Verification and Validation Workshops*, pages 4–11, March 2014.

[8] G. Brunette and R. Mogull. *Security guidance for critical areas of focus in cloud computing v2.1*. Cloud Security Alliance, 2009.

[9] Erdal Cayirci, Alexandr Garaga, Anderson Santana de Oliveira, and Yves Roudier. A risk assessment model for selecting cloud service providers. *Journal of Cloud Computing*, 5(1):14, 2016.

[10] K. Djemame, D. Armstrong, J. Guitart, and M. Macias. A risk assessment framework for cloud computing. *IEEE Transactions on Cloud Computing*, 4(3):265–278, July 2016.

[11] Mohamed El-Attar. Using smcd to reduce inconsistencies in misuse case models: A subject-based empirical evaluation. *J. Syst. Softw.*, 87:104–118, January 2014.

[12] Michael Felderer, Christian Haisjackl, Ruth Breu, and Johannes Motz. Integrating manual and automatic risk assessment for risk-based testing. In *Software Quality. Process Automation in Software Development: 4th International Conference, SWQD 2012, Vienna, Austria, January 17-19*, pages 159–180, 2012.

[13] Ana Juan Ferrer, Francisco HernáNdez, Johan Tordsson, Erik Elmroth, Ahmed Ali-Eldin, Csilla Zsigri, RaüL Sirvent, Jordi Guitart, Rosa M. Badia, Karim Djemame, Wolfgang Ziegler, Theo Dimitrakos, Srijith K. Nair, George Kousiouris, Kleopatra Konstanteli, Theodora Varvarigou, Benoit Hudzia, Alexander Kipp, Stefan Wesner, Marcelo Corrales, Nikolaus Forgó, Tabassum Sharif, and Craig Sheridan. Optimis: A holistic approach to cloud service provisioning. *Future Gener. Comput. Syst.*, 28(1):66–77, January 2012.

[14] M. Frydman, G. Ruiz, E. Heymann, E. Csar, and B. P. Miller. *Automating Risk Analysis of Software Design Models*. The Scientific World Journal, 2014.

[15] Nelson Gonzalez, Charles Miers, Fernando Redigolo, Tereza Carvalho, Marcos Simplicio, Mats Naslund, and Makan Pourzandi. A quantitative analysis of current security concerns and solutions for cloud computing. In *Proceedings of the 2011 IEEE Third International Conference on Cloud Computing Technology and Science*, pages 231–238, 2011.

[16] Matthew L. Hale and Rose Gamble. Secagreement: Advancing security risk calculations in cloud services. In *8th IEEE World Congress on Services*, 2012.

[17] Charles B. Haley, Jonathan D. Moffett, Robin Laney, and Bashar Nuseibeh. A framework for security requirements engineering. In *Proceedings of the 2006 International Workshop on Software Engineering for Secure Systems*, SESS '06, pages 35–42. ACM, 2006.

[18] Keiko Hashizume, Nobukazu Yoshioka, and Eduardo B. Fernandez. Misuse patterns for cloud computing. In *Proceedings of the 2nd Asian Conference on Pattern Languages of Programs*, AsianPLoP '11, pages 12:1–12:6, 2011.

[19] A. U. Khan, M. Oriol, M. Kiran, M. Jiang, and K. Djemame. Security risks and their management in cloud computing. In *4th IEEE International Conference on Cloud Computing Technology and Science Proceedings*, pages 121–128, Dec 2012.

[20] Rabia Latif, Haider Abbas, Saïd Assar, and Qasim Ali. Cloud computing risk assessment: A systematic literature review. In *Future Information Technology: FutureTech*, pages 285–295, 2014.

[21] Sean Marston, Zhi Li, Subhajyoti Bandyopadhyay, Juheng Zhang, and Anand Ghalsasi. Cloud computing - the business perspective. *Decision Support Systems*, 51(1):176 – 189, 2011.

[22] John McDermott and Chris Fox. Using abuse case models for security requirements analysis. In *Proceedings of the 15th Annual Computer Security Applications Conference*, ACSAC '99, pages 55–. IEEE Computer Society, 1999.

[23] Peter Mell, Karen Scarfone, and Sasha Romanosky. *A Complete Guide to the Common Vulnerability Scoring System Version 2.0*. NIST and Carnegie Mellon University, 1 edition, June 2007.

[24] Rui Miao, Rahul Potharaju, Minlan Yu, and Navendu Jain. The dark menace: Characterizing network-based attacks in the cloud. In *Proceedings of the 2015 Internet Measurement Conference*, IMC '15, pages 169–182, 2015.

[25] M. R. Naeem, W. Zhu, A. A. Memon, and A. Khalid. Using v-model methodology, uml process-based risk assessment of software and visualization. In *Proceedings of 2014 International Conference on Cloud Computing and Internet of Things*, pages 197–202, Dec 2014.

[26] Hironori Washizaki Nobukazu Yoshioka and Katsuhisa Maruyama. A survey on security patterns. *Progress in Informatics*, (5):35–47, 2008.

[27] Lotfi Ben Othmane, Pelin Angin, Harold Weffers, and Bharat K. Bhargava. Extending the agile development process to develop acceptably secure software. *IEEE Transactions on Dependable and Secure Computing.*, 11(6):497–509, 2014.

[28] Samir Ouchani and Gabriele Lenzini. Attacks generation by detecting attack surfaces. *Procedia Computer Science*, 32:529 – 536, 2014.

[29] Maragathavalli Palanivel and Kanmani Selvadurai. Risk-driven security testing using risk analysis with threat modeling approach. *SpringerPlus*, 3(1):754, 2014.

[30] José Roberto C. Piqueira and Vanessa O. Araujo. A modified epidemiological model for computer viruses. *Appl. Math. Comput.*, 213(2):355–360, July 2009.

[31] Mahendra Piraveenan, Mikhail Prokopenko, and Liaquat Hossain. Percolation centrality: Quantifying graph-theoretic impact of nodes during percolation in networks. *PLOS ONE*, 8(1):1–14, 01 2013.

[32] Lillian Rostad. An extended misuse case notation: Including vulnerabilities and the insider threat. *In Proc. 12th Working Conf. Requirements Eng.: Foundation for Software Quality (R EFSQ)*, 2006.

[33] Amit Sangroya, Saurabh Kumar, Jaideep Dhok, and Vasudeva Varma. *Towards Analyzing Data Security Risks in Cloud Computing Environments.*, volume 54 of *Communications in Computer and Information Science*, pages 255–265. 2010.

[34] Prasad Saripalli and Ben Walters. Quirc: A quantitative impact and risk assessment framework for cloud security. In *2010 IEEE 3rd International Conference on Cloud Computing*, pages 280–288, 2010.

[35] A. Sen and S. Madria. Off-line risk assessment of cloud service provider. In *2014 IEEE World Congress on Services*, pages 58–65, June 2014.

[36] Amartya Sen and Sanjay Madria. Data analysis of cloud security alliance's security, trust & assurance registry. In *Proceedings of the 19th International Conference on Distributed Computing and Networking*, ICDCN '18, pages 42:1–42:10, 2018.

[37] Guttorm Sindre and Andreas L. Opdahl. Eliciting security requirements with misuse cases. *Requirements Engineering*, 10(1):34–44, 2005.

[38] T. Sommestad, M. Ekstedt, and H. Holm. The cyber security modeling language: A tool for assessing the vulnerability of enterprise system architectures. *IEEE Systems Journal*, 7(3):363–373, Sept 2013.

[39] Marianthi Theoharidou, Nikolaos Tsalis, and Dimitris Gritzalis. In cloud we trust: Risk-assessment-as-a-service. In *IFIP International Conference on Trust Management(IFIPTM), Malaga, Spain, June 3-7*, pages 100–110, 2013.

[40] D. Verdon and G. McGraw. Risk analysis in software design. *IEEE Security Privacy*, 2(4):79–84, July 2004.

[41] J. Whitmore, S. Türpe, S. Triller, A. Poller, and C. Carlson. Threat analysis in the software development lifecycle. *IBM Journal of Research and Development*, 58(1):2:6–2:6, jan 2014.

[42] Imano Williams and Xiaohong Yuan. Evaluating the effectiveness of microsoft threat modeling tool. In *Proceedings of the 2015 Information Security Curriculum Development Conference*, InfoSec '15, pages 9:1–9:6. ACM, 2015.

[43] Artsiom Yautsiukhin, Riccardo Scandariato, Thomas Heyman, Fabio Massacci, and Wouter Joosen. Towards a quantitative assessment of security in software architectures. *Proceedings of the The Third International Conference on Availability, Reliability and Security, ARES 2008*, pages 921–928, 2008.

[44] Xiaohong Yuan, Emmanuel Borkor Nuakoh, Imano Williams, and Huiming Yu. Developing abuse cases based on threat modeling and attack patterns. *Journal of Software*, 10(4):491–498, 2015.

VI. A SECURE USER-CENTRIC FRAMEWORK FOR SERVICE PROVISIONING IN IOT ENVIRONMENTS

Amartya Sen, Kenneth K Fletcher*, and Sanjay Madria

Department of Computer Science

Missouri University of Science and Technology

Rolla, Missouri 65409–0050

Email: {asrp6,madrias}@mst.edu

*Department of Computer Science

University of Massachusetts Boston, USA

Email: kenneth.fletcher@umb.edu

ABSTRACT

The Internet of Things (IoT) environment is characterized by heterogeneous devices which collaborates to facilitate various sensing services to users in different domains. Currently, these services are provisioned to users by having them specify their functional and quality of service (QoS) preferences, similar to traditional QoS-driven web services selection methods. However, adopting conceptually similar web services selection framework for IoT environments is not sufficient because they do not allow users to specify their security preferences. Similar to QoS preferences, a user's security preferences may vary and change over time. Hence, there is a need for a framework that will allow users to explicitly specify their varying security preferences for IoT services. In this paper, we propose a user-centric framework for secure service provisioning in IoT environments by allowing users to express their variable security preferences and QoS requirements from the network that will support their applications. Additionally, we present a multilayer perceptron to validate the users' satisfaction feedback from the provisioned services.

This is done by utilizing deep learning techniques and quality of experience (QoE) information captured from the physiological sources like wearable body sensors. In this regard, we simulate our framework on a case scenario, discuss its applicability and challenges.

Keywords: Internet of Things; User-centric modeling; Security Preferences; Variable Security Compositions;

1. INTRODUCTION

The Internet of Things (IoT) paradigm is becoming a norm across many different user-centric applications. It facilitates services to users by encapsulating their way of life with the *Big* data sensed by surrounding wireless devices. Service provisioning for IoT-based applications involve developing frameworks which will identify a subset of wireless devices, tasked to serve different users. Generally, such frameworks consist of two primary layers - perception layer, comprising of IoT sensory devices, and management layer like the Cloud or Edge platforms to process the sensed data.

Existing IoT-based frameworks extend these two layers according to their application's need, in different domains like industry [20], health care [10]. Therein, they address specific issues either arising due to resource constrained IoT devices or interoperability challenges as a result of device heterogeneity [1], or discuss about making the management layer more efficient [16]. Although novel in their own regards, these frameworks lack the inclusion of user-centric behavior to dispense IoT-based services accounting for a user's Quality of Service (QoS) and security preferences. The aspect of addressing user QoS during service discovery and selection has been presented in [21]. The primary focus there is to outline the process of dynamic identification of IoT devices for service selection. However, they do not consider the satisfaction or variability of a user's QoS/seurity preferences.

In addition to QoS, users' security requirements from the participating devices in various IoT based application scenarios plays an important role. However, addressing security in this domain is a challenging task. This is because IoT devices are resource constrained (mainly in terms of memory, energy, and bandwidth) and thus, cannot support the well established resource intensive security protocols. Rullo et al. [17] address this challenge by leveraging concepts of optimization and game theory. However, the allocation of security resources is done from the infrastructure's perspective and do not include users in the loop. For the task of service provisioning in IoT environments, the aspect of security cannot be treated as a constant parameter. This is because security attacks on a network (e.g denial of service) has an impact on the QoS parameters (e.g. response time). Therefore, similar to network QoS, a user may have variable security preferences which must be taken into consideration, in addition to their QoS and functional preferences (e.g. region of interest, service duration) during service provisioning.

To elaborate this scenario, let us consider an instantiation of an IoT service for disaster management integrated with a Cloud platform. It can consist of an IoT infrastructure layer enumerated by devices like cellphones, sensor nodes, wearable body sensors, gathering and relaying data such as temperature, CO_2, user heart rate, images and video feeds. The management layer can be the traditional Cloud platform or Sensor Cloud [14], a variant of the Cloud platform to provide sensing-as-a-service. In a typical scenario, normal users may query for k nearest safe zones or evacuation routes. In contrast, rescue workers and first responders can analyze and query incoming sensory data (or image/video feeds) to gauge the disaster affected regions. Service provisioning parameters (functional, QoS and security) for both regular and first responder category users may not be the same. Furthermore, it is typical for users, even within the same user category, to have different needs. For example, one user may give more emphasis to privacy and authentication of data whereas another may give more priority to availability and latency.

In such a scenario, a framework modeled in a secure user-centric fashion will allow its users to specify their functional, and variable QoS and security preferences. This paradigm will include users in the service provisioning loop instead of solely relying on the innate networking and security capabilities of the IoT infrastructure. In turn, this will alleviate some of users' apprehension about the security provided by their subscribed services, thereby improving the overall user experience; something that existing IoT frameworks do not address.

The output of a user-centric service provisioning framework is an IoT network provisioned to a user based on their functional and variable QoS and security preferences. The effectiveness of this output is evaluated based on the satisfaction of subscribed users with respect to using the provisioned services. In traditional web services selection domain, this is measured based on a subjective satisfaction feedback provided by the user [8] [4]. Although, such a feedback maybe either prone to bias, or inaccurate due to users' lack of service familiarity leading to sub-optimal results [22]. To address this challenge during service provisioning in IoT environments, a novel way is to incorporate quantitative evaluation of users' quality of experience (QoE) information [7]. This can be obtained from users' physiological data like Electrocardiogram (ECG), Galvanic skin response (GSR), or skin temperature, which can help elucidate different user emotions [18] and validate the subjective satisfaction feedback provided by the users.

Our objective in this work is to address the aforementioned challenges by proposing a user-centric service provisioning framework for IoT environments, designed to achieve variable security, QoS preferences and users' satisfaction. Our contributions in this regard are as follows:

- Enable users to specify their personalized security preferences during service provisioning, thereby modeling network security as a variable parameter similar to network QoS;

- Introduce the concept of variable security composition paradigm to capture a user's variable preferences on their desired security protocols;
- Model service provisioning by integrating concepts of multi-objective optimization and user elastic preference specification; and
- Design a multilayer perceptron to accurately validate users subjective satisfaction feedback by incorporating physiological sensor data and deep learning emotion classification methodologies.

The rest of the paper is outlined as follows. In Section 2 we discuss some of the notable and significant works in the literature. We present our application scenario and proposed method in Section 3 and follow it up with our evaluations and analysis in Section 4. Finally, we conclude our paper and discuss some of the challenges which we will address as a part of our future work in Section 5.

2. RELATED WORK

The task of service provisioning in IoT domain is of importance due to the presence of numerous heterogeneous energy and resource-constrained IoT devices, collaborating to provide various services to users. This task becomes especially challenging when the services have to abide by some quality of service (QoS) and security requirements. This is because of the dynamic and uncertain nature of the IoT environment in which the services may span across multiple platforms.

Along these directions, [12] has proposed a probabilistic approach to evaluate the dependability and cost of service composition in IoT environments. They propose a finite state machine (FSM) to model the functional preferences and extend their FSM to use Markov decision processes to model the costs of the IoT services.

[3] has also proposed a method that utilizes social network paradigm to manage IoT devices and dispense services in a distributed fashion. They make use of RESTful web services to abstract the device specifications, thereby providing a transparent layer to invoke their services. Other works like [15] and [11] discuss the topics of conservation of energy of IoT devices while service provisioning. In this regard, [15] has outlined an evolutionary game approach which aims at avoiding the congestion of different services on the same set of IoT devices thereby increasing application hosting lifetime. In [11], in addition to energy consumption, it also addresses the optimal management of QoS. To do so, it proposed an energy-centered QoS-aware service selection algorithm (EQSA) which identifies IOT services based on QoS preferences requested by a user They subsequently evaluate the preferences based on Pareto optimality of their energy consumption and user's QoS preferences. In a similar direction, [21] has proposed an event-aware framework for dynamic services discovery and selection in IoT domain. Their primary focus in this regard is the dynamic discovery of the available services in an unpredictable IoT environment along with accounting for the user's QoS requirements.

These proposed models and approaches are not comprehensively user-centric as they do not address the satisfaction and variability of the considered users' QoS. Further, users' security preferences, similar to their QoS requirements may vary, and needs to be considered while developing a user-centric framework for service provisioning. Additionally, optimization techniques and game theory approaches may yield optimal results, but it leaves users with the task of identifying the best plan from the Pareto optimal services. We address this challenge by introducing ranking of optimized service plans by allowing users to specify their trade-off on optimization objectives like security, QoS, and cost. This notion is adopted from web services selection which has been heavily based on ranking services according to their QoS factors as shown in [8]. Since, combining multiple QoS factors makes QoS-based service selection a difficult task, as users struggle to find the right service with an optimal QoS factors combination [8].

Recently, the utilization of users' physiological data to create personalized contents by evaluating their emotional levels has seen a rising trend in different domains. [9] has proposed a model that relies on users' physiological data to analyze their experience related to video games and created user-tailored contents to increase their levels of engagement. [5] also proposed a model for self-adaptive software systems which collects several user data like electrodermal activity, physical activity, and skin temperature. In addition, [2] has proposed a method to use physiological sensor data to make decisions in the health care domain. However, these works tend to use an emotion classification index which is binary in nature i.e. presence or absence of emotions like joy, stress, anxiety, etc. This was addressed in [18], where a model using pscyho-physiological user data simulated for a mobile gaming environment using a naive Bayes approach to classify a user's frustration on multiple levels has been proposed. Our proposed multilayer perceptron to validate user's satisfaction feedback in this work is inspired by the techniques presented in [18]. We classify multiple levels of user satisfaction using deep learning techniques and supplement users' physiological data with their functional, QoS, and security preferences used for service provisioning in IoT environments.

3. PROPOSED FRAMEWORK

Our proposed user-centric service provisioning framework in IoT environments, for providing on-demand variable security and QoS comprises of two main stages: (1) IoT service provisioning, and (2) user satisfaction feedback validation.

In the first stage, service provisioning output for a given user will be a Pareto optimal set of IoT networks, composed of different combination of IoT devices, formulated based on three primary user inputs: (1) functional preferences (2) QoS preferences and (3) security protocol preferences. The service provisioning stage is first modeled as a multi-objective optimization problem, employing existing genetic algorithms like NSGA-II [6], with the goal of minimizing cost incurred by users subscribing to the IoT services (Section 3.1),

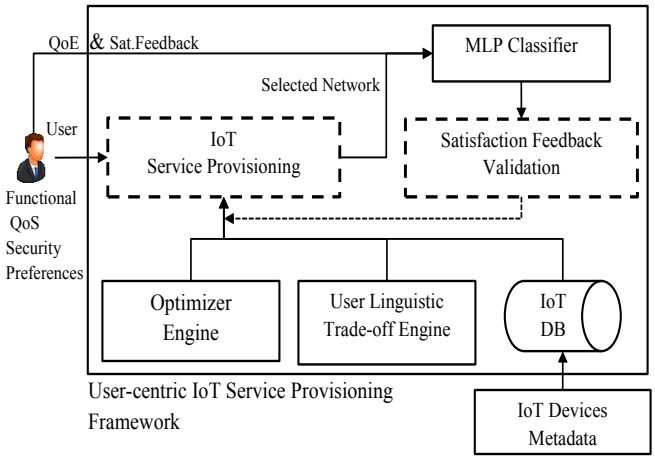

Figure 1. User-centric IoT Service Provisioning Framework Summary

and maximizing the satisfaction of users' QoS (Section 3.2.1) and security preferences (Section 3.2.2). However, optimization approaches minimize the individual objectives without any regard of the relationship that exists between them.

For instance, security preferences (e.g. a heavyweight encryption scheme) and QoS (e.g. response time) are conflicting and therefore, both might not be satisfied at the same time. In such scenarios, users should be able to explicitly trade-off amongst the optimization objectives, making their overall preferences known to IoT service providers. To do this, we have adopted the method outlined by authors in [8].

In the second stage, users will subscribe to, and use one of the IoT networks from the service provisioning stage. While doing so, their physiological data like electro cardiograph (ECG), galvanic skin response (GSR), and skin temperature will be collected and propagated to our framework. After the service duration ends, users will provide a subjective satisfaction feedback on a scale of *very high, high, medium, low, very low*. This feedback will be used as ground truth and the collected physiological data along with the input parameters of stage 1 will be passed through our multilayer perceptron (MLP) (Section 3.4), to accurately validate the users' satisfaction feedback. A summary of our proposed framework is shown in Figure 1.

The service provisioning stage is aided by the optimizer engine and the user preference trade-off engine (Figure 1), which will help to output a ranked list of optimal IoT networks, conforming to a user's explicit trade-off strategy with respect to cost, QoS, and security preferences. The user satisfaction feedback validation stage is supported by our proposed MLP classifier that helps to validate a user's satisfaction feedback by using their quality of experience (QoE) information. Further, our framework also utilizes an IoT device metadata database which is detailed in Section 4.

In the following sections, we outline the utility models for cost, QoS, and security preferences that is used at the IoT service provisioning stage, in conjunction with the optimization engine (see Figure 1), to generate optimal networks that meets the request of a particular user.

3.1. Functional Preferences and Cost Model. The IoT service provisioning stage of our proposed framework first takes a user's functional preferences input, and matches it against the IoT metadata database, to identify a candidate set of existing IoT devices, which can facilitate service for the user. In this regard, a user's functional preference input consists of a string containing:

1. The desired sensing attributes, networking, and data protocols;
2. Four coordinates representing their desired region of interest;
3. Sensing frequency and service duration;
4. Cost, QoS, and security satisfaction thresholds; and
5. User preference trade-off strategy on the optimization utilities.

Our framework first performs a functional preference match to filter a set of relevant IoT devices that are present in the IoT database. Thereafter, from this set, devices which can provide sensing service in the user's desired region of interest, are selected as the candidate set. This candidate set will be passed onto the optimizer engine. In this engine, the utilities

of QoS, Security preferences (Section 3.2.1 and 3.2.2), and costs (Section 3.1.1) will be evaluated against the threshold values specified by the user, as his/her functional preference (trade-off analysis discussed in Section 3.3).

3.1.1. Cost model. In our framework, we compute the cost (C) to users for subscribing to the services of the provisioned IoT networks based on the volume of data traffic (ν) that will generated and transmitted to the user. This will be directly proportional to the sensing frequency (f), service duration (τ), and number of devices (η), constituting the IoT network serving a particular user, as shown below:

$$\begin{aligned} C \propto & (f, \tau, \eta) \\ \nu = & f \times \tau \times \eta \\ C = & K \times \nu \end{aligned} \tag{1}$$

where K is a constant, representing the subjective service provider specific cost value per byte of data transmitted. This will be one of the optimization utilities that our framework's optimization engine will attempt to minimize. A notable thing in this regard is that, since the cost C is directly proportional to the number of participating devices in the network (η), attempts to minimize C will be directly affected by minimizing η, as the other two factors are user specific constants. Further, if C_T is the cost threshold that a user is willing to pay for a service, we define the cost satisfaction degree iC_d as follows:

$$iC_d = (C_T - C)/C_T \quad | \quad (iC_d < 1) \& (C_T, C) > 0 \tag{2}$$

As iC_d approaches 1, better is the cost satisfaction degree. A negative value for iC_d indicates that the cost of provisioned network is greater than what the user is willing to pay (C_T). Finally, iC_d cannot be 1, since this will mean that there is no service cost involved with the provisioned network. The objective of our optimization framework will be to maximize the utility of cost satisfaction degree. In addition to the cost utilities, for each of the networks

formulated using the IoT devices in the candidate set, our framework computes the utilities of their QoS and security preferences satisfaction degrees. These computations are done based on a user's non-functional preference input and our QoS and security preference models. We outline the same in the following section.

3.2. Non-functional Preferences and QoS-Security Models. The input to our QoS and security preference satisfaction models is a user's non-functional preference. This input will be a string containing: (1) user's preference trade-off strategy on QoS attributes; (2) desired upper and lower bound values indicating the user's variable QoS preferences; and (3) list of variable security protocols and their preference weights. The first two inputs will be used to compute the QoS satisfaction degree, whereas the third input will be used to compute the security preference satisfaction degree of a provisioned network. We detail these computations in the following subsections.

3.2.1. User preference on QoS factors. As mentioned in Section 3.1, functional attributes in the domain of IoT can be captured through parameters like region of interest, sensing frequency, and service duration. On the other hand, non-functional attributes like QoS preferences, are typically captured based on a combination of factors like *availability*, *reliability*, *throughput*, and *response time*. However, with the availability of several devices in a given region of interest with similar functionality, users may end up with multiple networks that can facilitate a requested service. Therefore, the challenge is to select the service, via an IoT network, that best satisfies a user's requirements. To address this challenge, users can personalize their preferences on QoS factors, which provides a distinction among competing IoT networks. Although, this personalization comes with additional challenges such as having to deal with conflicting QoS attributes and being unable to explicitly specify their trade-offs among QoS factors to make their preferences known to service providers.

To ensure that users' personalized preferences on QoS factors are fully captured, we employ a method based on fuzzy logic [8]. The method allows users to represent their variable QoS requirements and associated importance using linguistic terms to specify their personalized trade-off strategies. Users' personalized QoS attribute requirements are first captured through linguistic terms, then by users' membership function, and finally users' importance on QoS factor(s). For instance, a user may specify response time QoS factor as follows: *The response time for my IoT network must be HIGH*, which is a linguistic term in fuzzy logic whose membership function specifies a user's preference on the response time QoS factor. Formally, let $S = \{s_1, s_2, s_3, ..., s_n\}$ denote a set of services representing IoT networks formulated using a combination of IoT devices in the identified candidate set, $U = \{u_1, u_2, u_3, ..., u_m\}$ be a set of users, and $Q = \{q_1, q_2, q_3, ..., q_k\}$ a set of QoS factors describing each provisioned IoT network. The personalized QoS preference $P_{U_i}^{Q_j}$ for a user U_i on their QoS factor Q_j is a membership function δ and a weighting factor ϖ given as:

$$P_{U_i}^{Q_j} = \delta(U_i, Q_j) \times \varpi(U_i, Q_j) \tag{3}$$

Generally, there are several QoS factors that describe a service. Therefore, the overall personalized service requirement, R_{U_i}, can be specified using individual personalized QoS preference, $P_{U_i}^{Q_j}$, and an aggregation operator, $\bigsqcup$ as shown in eq. 4. Here, $\bigsqcup$ is any of the fuzzy connective operators $\wedge$, $\vee$ or $\otimes$. Given a user U_i, their preferences on QoS factors Q_1 and Q_2, $P_{U_i}^{Q_1}$ and $P_{U_i}^{Q_2}$ the fuzzy connective operators are computed as shown in eq. 5. In our proposed framework, the following QoS factors are considered - Response time (RT), Availability (A), Throughput (T), and Reliability (RE). Therefore, for a user U_i, an instantiation of eq. (4) using his/her personalized QoS preferences $P_{U_i}^{Q_j}$ is shown in eq. 6. Given S, Q, and R, the service provisioning process in the IoT environment can be modeled as a ranking in terms of the satisfaction of requirement R.

Therefore, for any two services S_i and S_j composed of a combination of IoT devices, the concept in eq. 7 should hold true.

$$R_{U_i} = \bigsqcup_{j=1}^{k} P_{U_i}^{Q_j} \tag{4}$$

$$\bigsqcup = \begin{cases} \wedge, & min(P_{U_i}^{Q_1}, P_{U_i}^{Q_2}) \\ \vee, & max(P_{U_i}^{Q_1}, P_{U_i}^{Q_2}) \\ \otimes, & mean(P_{U_i}^{Q_1}, P_{U_i}^{Q_2}) \end{cases} \tag{5}$$

$$\begin{aligned} R_{U_i} &= P_{U_i}^{RT} \wedge P_{U_i}^{A} \vee P_{U_i}^{T} \otimes P_{U_i}^{RE} \\ R_{U_i} &= min(P_{U_i}^{RT}, P_{U_i}^{A}) \vee mean(P_{U_i}^{T}, P_{U_i}^{RE}) \\ R_{U_i} &= max(min(P_{U_i}^{RT}, P_{U_i}^{A}), mean(P_{U_i}^{T}, P_{U_i}^{RE})) \end{aligned} \tag{6}$$

$$S_i \succ S_j \iff Sat_R(S_i) \geq Sat_R(S_j) \tag{7}$$

Where $Sat_R(S_i)$ (eq. 6 and 7) represents the QoS satisfaction degree of service S_i with respect to some user requirement R. Hence, using eq. (3) and (4) our framework will compute the QoS satisfaction utility (Sat_R) which will be maximized by the optimization engine of our framework during service provisioning. Nevertheless, QoS preferences do not completely model the secure user-centric paradigm of our proposed framework. In addition to QoS preferences, an essential component is to account for the user's variable security preferences which we discuss in the next section.

3.2.2. Variable security preferences. The different layers of a typical IoT framework may have different security requirements for different application domain (e.g. health care, smart cities, etc). The different security requirements stems from the presence of diverse range of participating devices and protocols, which makes the incorporation of

end-to-end security a challenging task. Furthermore, with respect to developing an IoT framework to account for users security preferences, it is challenging to rely on a universally defined security policy. This is because a user's security requirements may vary with time. For example, they may initially have a higher preference on confidentiality of data, but at a later stage may have more emphasis on network availability. These circumstances necessitate the presence of variable security composition (VSC) paradigm, that will be flexible according to users' varying security preferences and device capabilities to support them.

In our proposed framework, VSC will be characterized using two parameters: (1) users' security protocol preferences (SPP), and (2) IoT network's security protocols used. The first parameter will be influenced by the nature of the IoT application (e.g. health care vs. military), type of user (e.g. normal users vs. first responders), service requests (e.g. real-time vs. off-line data requests). Additionally, our framework will impose minimum security requirements (incorporated by the second parameter) which could be in the directions of having accurate data, available with some level of confidentiality and integrity. The effect of a user's preferences on security will play a role in scenarios such as real-time data access, where users may give a higher priority to aspects such as data stream rate. As a result, users may prefer a lightweight encryption scheme with alternate data packet authentication. Although, if users are more concerned about their privacy, they will give higher preferences to heavyweight encryption schemes and authenticating every data packet.

At any given instant of time, a single IoT device can be a part of multiple IoT networks providing services to different users simultaneously. As such, there is a possibility that an IoT device might be hosting a composition of security measures and inter-operating between them. These compositions of security measures hosted by an IoT device will be self contained within a *capsule* and the emphasis of a capsule will be determined by the

needs and capabilities of the service provisioning networks it is a part of. The number of capsules in a device will not exceed host device's processing power and available resources. We formalize our proposition for this scenario as follows:

Definition 3.1. Variable Security Compositions: *Variable security compositions is denoted by a set* $\{\Psi_1, \Psi_2, \cdots, \Psi_L\}$. *Where L is the number of layers present in the IoT framework, and* Ψ_i *is a security composition hosted by a device belonging to a layer i, which can be represented by a set* $\{\vartheta_1, \vartheta_2, \cdots, \vartheta_m\}$. *Where* ϑ *is a security* capsule *composed of a specific composition of security measures either addressing confidentiality (AES, PES, Hummingbird2.0), integrity (SHA-2, MD5), availability (channel hopping, message source authentication, solving challenges), or authentication (digital signatures) (*CIA_vA_u*) of a network.*

Here, m is the number of users hosted by the IoT framework. At any given instance, a security *capsule* ϑ_i will address the minimum security requirements (in terms of $\mathrm{CIA}_v\mathrm{A}_u$) of the network(s). Further, if maximum resource utility threshold of a device hosting Ψ_i is Λ_T, then the additive utilization of resources from all $\vartheta_i \in \Psi_i$ is less than or equal to Λ_T as follows:

$$\sum_{i=1}^{m} (\Lambda_{\vartheta_i} + \Lambda_{S_i}) \leq \Lambda_T \tag{8}$$

where Λ_{S_i} is the resource utilization as a result of service provisioning S_i for a user U_i.

To practically interpret this notion, let $\Lambda_T = 1.0$, and Λ_{S_i}=80% and sum of Λ_{ϑ_i} be 20% of Λ_T; then each ϑ_i will have the resource utility between $0 \leq \Lambda_{\vartheta_i} \leq 1$ (1 here is 20% resource utilization of Λ_T). Thus, we can say that a Ψ_i is *composed* of 66% of ϑ_1 and 34% of ϑ_2 where ϑ_1, ϑ_2 are any two security capsules. The emphasis given to a ϑ_i in Ψ_i (the percentage of Λ in a particular ϑ), can be determined by evaluating the parameters of VSC as mentioned above.

Our proposed framework requires users to first specify a set of preferences on their variable security protocols, on which their requested service will be provisioned. We denote this set as SPP, which is a personalized weighted input for different security protocols, that can be used to enforce the security categories of confidentiality (C), integrity (I), availability (A_v), and authentication (A_u). The set, SPP_i, for a user U_i can either be null (no security preferences) or a list of z number of security protocols ($p_1, p_2, ..., p_z$), with their preferences as shown in eq. 9. In this equation, user preferences (ω) is a five-scale subjective quantitative rating between one and five. An example instantiation of SPP for two users U_1 and U_2 is shown in eq 10.

$$SPP_i = \begin{cases} \phi \\ \{\omega.p_1, \omega.p_2, ..., \omega.p_z\} \quad | \quad \omega \in 1, \cdots, 5 \end{cases} \tag{9}$$

$$\begin{aligned} SPP_{U_1} =& \{5.AES-128, 3.AES-256, \\ & 5.Deffie-Hellman, 2.SHA-2, 5.TLS\} \\ SPP_{U_2} =& \{3.AES-256, 1.ECC-128, 4.ElGamal \\ & 2.Schnorr, 2.IPSec\} \end{aligned} \tag{10}$$

For user U_1, the maximum security preference weight is 5 for the security protocols AES-128 bit encryption, Deffie-Hellman (key exchange method for AES encryption), and TLS (communication security protocol). The remaining preference weights are 3 (AES-256) and 2 (SHA-2 scheme). One can interpret this as U_1 having more preference on an encryption scheme with lower key size, which can be substituted with a stronger encryption if not available, thereby relaxing the strength of integrity scheme. Further, U_1 also gives more emphasis to communication channel security (in contrast to U_2). On the other hand, U_2 overall has higher preferences on stronger encryption and lower integrity and communication security. Note that the security preferences have impact on the QoS.

Therefore, it is to be also noted that users are capable of changing their security preferences during the course of service usage, in which case our framework will re-compute the networks to be used for service provisioning. Once the user's security preferences are specified, it be will be cross-validated against the devices present in the network(s) (represented by S_i) selected for service provisioning, in order to evaluate a utility *IoT Reputability* ($iRep_{U_i}^{S_i}$), which will be maximized by the optimization engine of our framework, as shown in eq. 11. The equation states that if devices in a selected network for service provisioning constitutes of a ϑ_i which is able to satisfy the highest user preferences in SPP_{U_i}, its $iRep$ score is 1 (case 1, eq (11)).

Whereas, if it does not match any of the preferences ($\{\omega p_1, \omega p_2, ..., \omega p_z\}$) specified by the user, an $iRep$ score of 0 is assigned. Finally, if some of the security preferences are met but not the highest preference ones, then a $iRep$ score between 0 to 1 is computed. In case the user does not specify any security preference ($SPP_{U_i} = \phi$), the provisioned network's default security protocols will be used and it will not impact the $iRep$ computations.

$$\forall SPP_{U_i} \neq \phi,$$
$$iRep_{U_i}^{S_i} = \begin{cases} 1 & |\exists \vartheta_i \iff \omega_i.p_i \ \ \& \ \ \omega_i = max(\omega_i) \\ 0 & |\nexists \vartheta_i \iff \omega_i.p_i \\ (0,1) & |\exists \vartheta_i \iff \omega_i.p_i \ \ \& \ \ \omega_i \neq max(\omega_i) \end{cases} \tag{11}$$

3.3. Tradeoff in Security, QoS, & Cost Objectives. In the development of IoT frameworks for any application scenario, it is imperative to perform a trade-off analysis considering parameters like network security, QoS, and cost, owing to the heterogeneous and resource constrained nature of IoT devices. This will help to achieve optimal performance and security standards within the budgetary constraints of the users. In order to address these considerations and develop a holistic trade-off analysis and optimization method, we model

our IoT service provisioning by formulating networks as a multi-objective optimization problem. This will help in developing networks composed of IoT devices which will optimize the security, QoS, and cost objectives conforming to its respective constraints.

In our framework, optimization approaches like evolutionary algorithms may result in networks with optimal objectives. However, as mentioned in Section 3, these objectives are optimized in isolation without regard of the relationship that exist between them. For instance, QoS and Security are conflicting in nature and therefore, satisfaction for both may not be satisfied if the constraints are too strict. In such a case, users need the capability to explicitly specify their trade-off amongst the optimization objectives such that selection from the optimized result set are customized according to the personalized preferences of individual users. In this regard, we introduce the method outline in [8] and shown in eq. (6) on top of our optimization modeling. The final output will be a personalized ranked set of optimized IoT networks capable of provisioning services to a user.

3.3.1. Optimization problem formulation. Our envisioned multi-objective problem formulation is as follows: Generate a network S_i using N number of IoT devices which satisfies a user's functional preferences in the given user's region of interest, hosting multiple *VSC* such that it minimizes the objectives shown in eq. 12.

$$\textbf{minimize:} \quad iC_d^{'} \quad Sat_R^{'}(S_i) \quad iRep^{'} \tag{12}$$

The specified objectives are the inverse of cost satisfaction degree (iC_d, eq. 2), QoS satisfaction degree ($Sat_R(S_i)$, eq. 6), and security preference satisfaction degree ($iRep$, eq. 11). These objectives will be subject to the constraints as outlined in eq. 13. Where, the total cost ($C_{network}$) involved in service provisioning for a user should be less than or equal to their budgetary constraints (C_T). $QoS_{network} \geq QoS_T$ states that the provisioned network's QoS satisfaction degree should be greater than or equal to the user's specified QoS satisfaction threshold and similarly for security preferences satisfaction degree. $\sum_{i=1}^{N}(\Lambda_{\vartheta_i} + \Lambda_{S_i}) \leq \Lambda_T$

depicts the resource utilization (Λ) of the device, supporting a variable security composition (Ψ), composed of numerous capsules (ϑ_i) and running the services (performing sensing and transmitting data etc.) should not exceed the threshold resource utilization capacity (Λ_T).

Constraints:

$$
\begin{aligned}
&C_{network} \leq C_T \\
&QoS_{network} \geq QoS_T \\
&Security_{network} \geq Security_T \\
&\sum_{i=1}^{N} (\Lambda_{\vartheta_i} + \Lambda_{S_i}) \leq \Lambda_T
\end{aligned}
\tag{13}
$$

3.4. Validation of User Satisfaction Feedback. In developing a user-centric framework, the optimality of the selected output is evaluated by a user's satisfaction from engaging with the provisioned service. This will give insights about the performance of a user-centric framework which can be used to assist future users to specify their input preferences for QoS and security.

In traditional service selection approaches, the degree of satisfaction is typically evaluated as shown in Figure 2 where the y-axis represents the estimated satisfaction degree of a user ($0 \rightarrow 1$) vs. the user's preference on a given QoS parameter (e.g., response time). It is assumed in these traditional approaches that as a QoS parameter degrades, the satisfaction degree of a user will also linearly degrade. However, practically, this need not be completely accurate in the IoT environment. This is because satisfaction degree may be influenced by the service that is being used and need not be directly proportional to the individual QoS parameters. Further, the service itself is a culmination of various QoS and security preferences, and all of these parameters will not equally contribute towards the degradation or improvement of a user's satisfaction.

Therefore, to better validate the evaluation of users' satisfaction feedback, in our framework we have proposed the incorporation of QoE data through analysis of user's physiological sensor data using deep learning framework's multilayer perceptron (MLP) classification.

Users' physiological sensor data can be characterized with attributes such as electrocardiograph (ECG), blood volume pulse (BVP), heart rate (HR), electrodermal activity (EDA/GSR), and so forth. Traditionally, their usage has been researched in fields like health care and in more recent endeavors to develop software which can adapt to assure a basic level of QoS and positive user experience [5]. These physiological data are used to analyze different human feelings like stress, anxiety, excitement, and can be collected in an non-intrusive fashion by wearable devices like the E4 wristbands [5]. The aggregation of these physiological data for user QoE (Quality of experience) measurement will be done in a continuous fashion during their service usage.

However, an important challenge to note in this regard is that the collected data needs to correlate to the service usage and not be influenced by external environmental factors or other psychological aspects which is not related to the service. This is beyond the scope of our current work and is something we will address in the future.

The collected physiological data will be processed by extracting its statistical features like mean, maximum, minimum, standard deviations, etc. Then, it will be used along with the user's specified QoS and security preferences to create an input vector which will be passed into our multi-layer perceptron (MLP) for multi-class classification. The classification will be for user's satisfaction feedback based on a qualitative five-scale rating - *very low, low, medium, high, very high*. Users will also provide this qualitative rating for their satisfaction as an input after the service usage which will be used as a baseline to train and test our designed MLP.

Table 1. IoT Device Profile for Simulations

Field	**Attribute**
Device GeoLocation	(38.627,-74.005) to (40.712,-90.199)
Device Type	1:Normal Sensor,2:Media Sensor,3:Base Station
Communication Range	(20,70) meters
Sensing Range	(10,50) meters
Device Energy	(1E04,5E04) Joules
Sensing Attributes	Temperature, Humidity, Light Intensity, Image
Network Protocols	6LoWPAN, IPv4/IPv6, RPL
Data Protocols	MQTT, CoAP, AMQP
Radio type	WiFi, Bluetooth, ZigBee
Security Protocols	AES-128,AES-256,ECC-128,ECC-256,SHA-2,MD-5,ElGamal,Schnorr,PGP,TLS,IPSec,Deffie-Hellman

4. EVALUATION AND ANALYSIS

In the following sections we present the outlines of our synthetic dataset and results of simulating our proposed approach.

4.1. Dataset Generation. All our simulations have been performed on a Windows 7, 3.10Ghz core i5 machine with 8GB RAM. The dataset has been generated synthetically for 150 users, comprising of their functional, QoS, and security preferences, physiological sensing values, and satisfaction feedback label. We randomly deployed following a uniform distribution, 200 IoT devices between maximum and minimum latitude of 40.712775 and 38.627003 respectively, and maximum and minimum longitude of -90.199404 and -74.005973 respectively. Each deployed IoT device consisted of a set of attributes (randomly selected) as displayed in Table 1 and stored in our IoT DB (Figure 1). The QoS values for users of our framework has been generated following a random uniform distribution whose lower and upper bounds is summarized in Table 2.

The remaining user preferences like functional and security preferences (region of interest, sensing attributes, network protocols, security preferences, etc.) has been generated by keeping the values within the bounds specified in Table 1.

Table 2. QoS Value Data Bounds

QoS Type	Bounds (Lower, Upper)
response time	(0.0, 15.0)
availability	(85,100)
throughput	(0,24)
reliability	(3,18)

With respect to physiological values, ECG has been simulated using the concept of Daubechies wavelet [13], which is a rough approximation to a real, single, heart beat signal. For each signal, we have considered the beats per minute rate (which is 60 while resting and 180 during exercise, usually for a healthy athletic person), and also added a gap after the signal to simulate when the heart is resting. Finally, we concatenated the number of heart beats needed and added a random Gaussian distributed noise to obtain our simulated ECG data. Whereas, we simulated our GSR data by using Poisson distribution. We did this by calculating the first few moments of the distribution and then added a gap. We concatenated an arbitrary number of peaks needed to obtain our simulated GSR data.

4.2. Service Provisioning Results. The goal of our service provisioning simulation experiment is to generate a set of optimal IoT networks, ranked according to a user's linguistic trade-off strategy on optimization objectives. Our optimization approach is modeled using the NSGA-II package in MOEA framework[7] version 2.10. The algorithmic parameters used to execute NSGA-II, identified through empirical estimations are summarized in Table 3.

Table 3. NSGA-II Algorithm Parameters

NSGA-II Parameter	**Value**	**NSGA-II Parameter**	**Value**
Max. Evaluations	2000	Population Size	500
Crossover rate	0.6	Crossover distribution index	15.00
Mutation rate	0.1	Mutation distribution index	10.00

[7]moeaframework.org

We present the service provisioning results for 2 random users, consisting of their unranked and ranked optimal IoT networks as shown in Figure 3 and 4. The unranked output (Figure 3 and 4, Left) consists of an optimal list of IoT networks and their satisfaction degrees for security preference (eq. 11), QoS (eq. 6), and cost (eq. 2). It is to be noted from the figures that a higher cost utility (iC_d) indicates a lower service cost (C). All IoT networks in each figure are optimal with respect to the constraints specified in our proposed framework (eq. 13), and aligns with each user's functional and nonfunctional preferences. The title of the plots in Figure 3 and 4 outlines the respective user's linguistic trade-off on the optimization objectives along with their specified threshold values.

Figure 3 left and right, shows the unranked and ranked IoT output networks for user 1's request respectively. It consists of eleven optimal IoT networks, unranked (Figure 3, Left), as returned by the optimization engine of our framework. The vertical axis of the figure denotes the satisfaction degree of each optimization objective while the horizontal axis represents the list of networks. The output of Figure 3, right side, uses the user preferences trade-off engine of our framework and shows optimally ranked IoT networks (horizontal axis) based on user 1's trade-off strategy (title of Figure 3, Right). The trade-off strategy of user 1 gives more emphasis to security preferences (dotted pattern) and compromises between cost (vertical lines pattern) and QoS (horizontal lines) of the optimally provisioned networks which is reflected in our framework's output. By allowing user 1 to explicitly specify his/her overall trade-off strategy on the optimal objectives, our framework outputs a ranked list of IoT networks, a process which would have otherwise been manually performed by user 1 to determine which of the eleven optimal services would be suitable for his/her needs. Along this direction, the IoT network provisioned at rank 5 (Ntwk5, Figure 3, Right) would seem as the best choice because the satisfaction degree on cost objective is the highest. However, given the overall trade-off strategy of the user 1, explicitly giving more emphasis to security preference, overall output of our framework suggests it as the fifth plan that should be selected by the user.

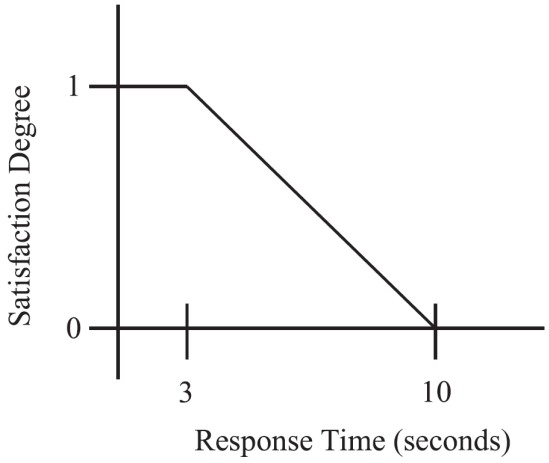

Figure 2. Traditional User Satisfaction Feedback Evaluation [8]

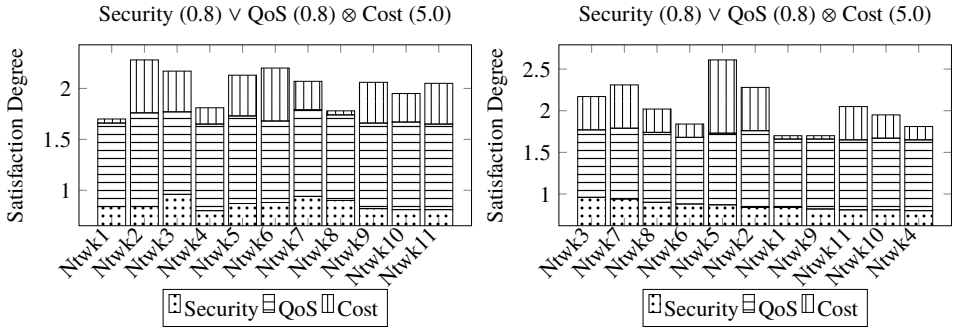

Figure 3. User1 Optimal Services:Unranked (Left), Ranked (Right)

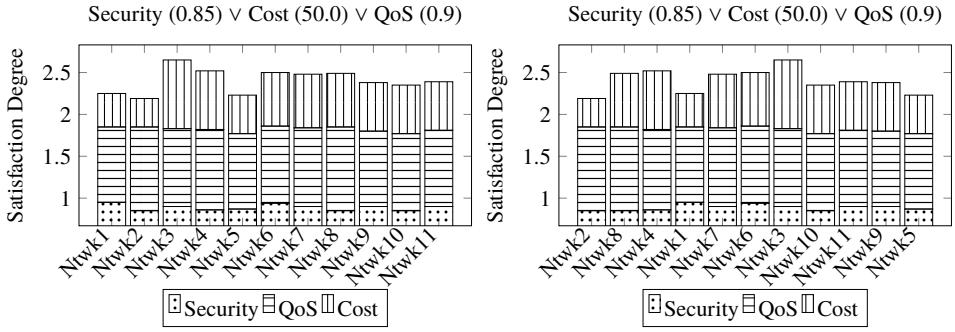

Figure 4. User 2 Optimal Services:Unranked (Left), Ranked (Right)

Whereas based on user 2's trade-off strategy (title of Figure 4), our framework first selects all services with higher QoS; a 1.0 QoS satisfaction degree is computed for rank 1 and rank 2 (Ntwk2 and Ntwk8) which is the highest satisfaction amongst all three objectives. Thereafter, from rank 6 (Ntwk6) onwards, the highest satisfactions amongst the three objectives is for security preferences. As a result of such a trade off strategy, there is no increasing or decreasing trend within the cost satisfaction degree of the ranked IoT network services.

4.3. Satisfaction Feedback Evaluations. Our multilayer perceptron (MLP) has been designed using Keras[8] framework running on top of TensorFlow[9] backend. It comprises of a sequential deep learning model with input vector comprising of 18 dimensions consisting of 6 statistical features (Section 3.4), each for ECG and GSR values; one value for skin temperature, four QoS preferences (Section 3.2.1), and security preferences satisfaction degree.

Further, the sequential model has 3 hidden layers, each comprising of fully connected 128 neurons connected to an output layer with 5 states. The rest of parameters used for training and testing our MLP is summarized in Table 4. During the training phase, we stripped the user input of the satisfaction feedback labels.

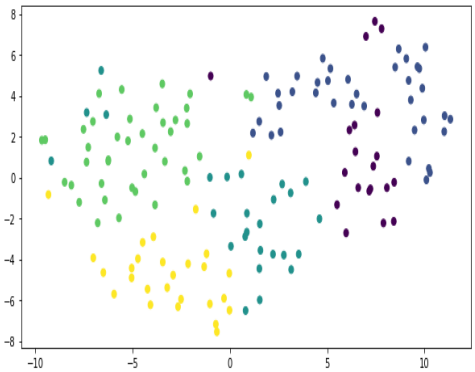

Figure 5. t-SNE Plot for Train Dataset

[8]keras.io
[9]tensorflow.org

Table 4. Simulation Parameters for Multilayer Perceptron

Attribute	**Value**
train_test split	80,20
hidden layer activation	relu
output layer activation	softmax
categorical data transformation	one-hot encoding
Dropout	0.2
Loss	categorical crossentropy
optimizer	rmsprop
metrics	accuracy
epochs	200
batch size	32
validation	KFold Cross validation

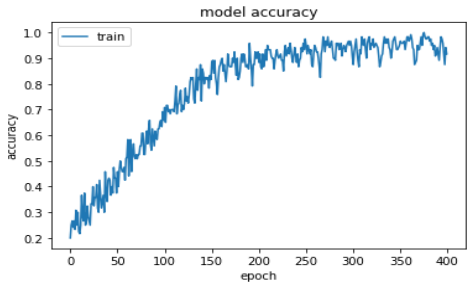

Figure 6. Training Dataset Accuracy Variation

We then analyzed the dataset to identify representational user records from each of the five satisfaction feedback categories (very low → very high). Thereafter, using these 5 user records as centroids, we applied KMeans clustering (python sklearn package) and predicted the labels of the remaining user records. The t-SNE plot [19] for the same is shown in Figure 5. t-SNE plots are typically used to convert high dimensional data and visualize them in two dimensional space. In Figure 5, each individual color represents one of five user satisfaction labels to be used by our MLP model during its training. This training dataset was then used to compile our MLP model. The net accuracy was estimated to be approximately 90%, with a loss of approximately 0.30. The variations of accuracy and loss through the training epochs are shown in Figure 6 and 7.

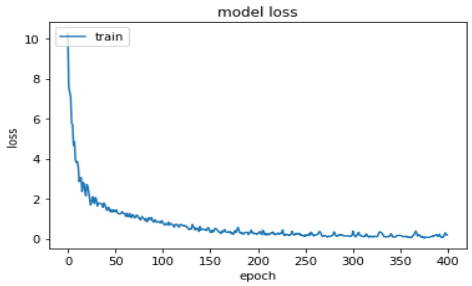

Figure 7. Training Dataset Loss Variation

5. CONCLUSION AND FUTURE WORK

In this paper, we have presented the outlines of a user-centric framework for secure service provisioning in IoT environments. In doing so, our framework accounts for the optimal satisfaction of a user's variable quality of service (QoS) preferences through linguistic fuzzification, and security preferences by introducing the concept of *variable security composition* paradigm. Finally, our framework further assists the users to efficiently select from the optimally provisioned services by allowing them to specify their linguistic trade-off on the optimization objectives. Additionally, we have also presented the design of a multilayer perceptron which will be able to validate the user's satisfaction feedback from using the provisioned service by utilizing the quality of experience collected from physiological body sensors and deep learning techniques.

As a part of our future work, we would develop a test-bed implementing our synthetic simulation environment and collecting physiological data from real users and thereafter, applying our proposed framework to a real-world scenario. This will enable the accurate modeling of the dynamic and uncertain nature of the IoT environment and carry out dynamic service provisioning. Further, we will carry out extensive sensitivity analysis with respect to our framework's optimization approach, compare and contrast NSGA-II with other available evolutionary algorithm techniques.

REFERENCES

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys Tutorials*, 17(4):2347–2376, Fourthquarter 2015.

[2] Ahmed MU. Begum S, Barua S. Physiological sensor signals classification for healthcare using sensor data fusion and case-based reasoning. *Sensors (Basel, Switzerland)*, 14:11770–11785, 2014.

[3] G. Chen, J. Huang, B. Cheng, and J. Chen. A social network based approach for iot device management and service composition. In *2015 IEEE World Congress on Services*, pages 1–8, June 2015.

[4] X. Chen, Z. Zheng, X. Liu, Z. Huang, and H. Sun. Personalized qos-aware web service recommendation and visualization. *IEEE Transactions on Services Computing*, 6(1):35–47, First 2013.

[5] Nelly Condori-Fernandez and Franci Suni Lopez. Using emotions to empower the self-adaptation capability of software services. In *Proceedings of the 2nd International Workshop on Emotion Awareness in Software Engineering*, SEmotion '17, pages 15–21, 2017.

[6] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, Apr 2002.

[7] Mianxiong Dong, Takashi Kimata, Komei Sugiura, and Koji Zettsu. Quality-of-experience (qoe) in emerging mobile social networks. *IEICE Transactions on Information and Systems*, E97.D(10):2606–2612, 2014.

[8] Kenneth K. Fletcher, Xiaoqing F. Liu, and Mingdong Tang. Elastic personalized nonfunctional attribute preference and trade-off based service selection. *ACM Trans. Web*, 9(1):1:1–1:26, January 2015.

[9] Theodosis Georgiou and Yiannis Demiris. Adaptive user modelling in car racing games using behavioural and physiological data. *User Modeling and User-Adapted Interaction*, 27(2):267–311, Jun 2017.

[10] S. M. R. Islam, D. Kwak, M. H. Kabir, M. Hossain, and K. S. Kwak. The internet of things for health care: A comprehensive survey. *IEEE Access*, 3:678–708, 2015.

[11] M. E. Khanouche, Y. Amirat, A. Chibani, M. Kerkar, and A. Yachir. Energy-centered and qos-aware services selection for internet of things. *IEEE Transactions on Automation Science and Engineering*, 13(3):1256–1269, July 2016.

[12] L. Li, Z. Jin, G. Li, L. Zheng, and Q. Wei. Modeling and analyzing the reliability and cost of service composition in the iot: A probabilistic approach. In *2012 IEEE 19th International Conference on Web Services*, pages 584–591, June 2012.

[13] Jean-Marc Lina and Michel Mayrand. Complex daubechies wavelets. *Applied and Computational Harmonic Analysis*, 2(3):219 – 229, 1995.

[14] Sanjay Madria, Vimal Kumar, and Rashmi Dalvi. Sensor cloud: A cloud of virtual sensors. *IEEE Software*, 31(2):70–77, 2014.

[15] J. Na, K. J. Lin, Z. Huang, and S. Zhou. An evolutionary game approach on iot service selection for balancing device energy consumption. In *2015 IEEE 12th International Conference on e-Business Engineering*, pages 331–338, Oct 2015.

[16] M. Nardelli, S. Nastic, S. Dustdar, M. Villari, and R. Ranjan. Osmotic flow: Osmotic computing + iot workflow. *IEEE Cloud Computing*, 4(2):68–75, March 2017.

[17] Antonino Rullo, Daniele Midi, Edoardo Serra, and Elisa Bertino. Pareto optimal security resource allocation for internet of things. *ACM Trans. Priv. Secur.*, 20(4):15:1–15:30, October 2017.

[18] Brandon Taylor, Anind Dey, Daniel Siewiorek, and Asim Smailagic. Using physiological sensors to detect levels of user frustration induced by system delays. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '15, pages 517–528, 2015.

[19] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[20] L. D. Xu, W. He, and S. Li. Internet of things in industries: A survey. *IEEE Transactions on Industrial Informatics*, 10(4):2233–2243, Nov 2014.

[21] A. Yachir, Y. Amirat, A. Chibani, and N. Badache. Event-aware framework for dynamic services discovery and selection in the context of ambient intelligence and internet of things. *IEEE Transactions on Automation Science and Engineering*, 13(1):85–102, Jan 2016.

[22] L. Zhang, S. Wang, R. K. Wong, F. Yang, and R. N. Chang. Cognitively adjusting imprecise user preferences for service selection. *IEEE Transactions on Network and Service Management*, 14(3):717–729, Sept 2017.

SECTION

3. SUMMARY AND CONCLUSIONS

This dissertation presents three frameworks to carry out security risk assessments for key assets related to the Sensor Cloud computing domain, namely - infrastructure, application, and users. The proposed techniques are informal in nature which extends and improves upon existing approaches outlined by notable organizations like National Institute of Standards and Technology (NIST), European Union Agency for Network and Information Security (ENISA), and Cloud Security Alliance (CSA).

The infrastructure of a Sensor Cloud computing domain typically consists of several heterogeneous wireless sensor networks (WSNs). The research outlined for carrying out security risk assessment with respect to such infrastructure introduces the usage of attack graphs to model the net threat levels to network security parameters (confidentiality, integrity, and availability) for WSNs in Sensor Cloud. The notable feature of the approach lies in its utilization of feasible set of known attacks on WSNs to model the attack graphs instead of the number of sensor nodes in the WSN. This aids in bypassing combinatorial explosion problem which is persistent in the traditional attack graph approaches. The outlined framework further models the attack graphs using quantitative approaches of Bayesian networks in order to estimate time frames which can be utilized by security administrators to schedule maintenance and repair activities. This is beneficial since the infrastructure of these kind of environments are generally deployed in an ad-hoc fashion which is largely unattended in nature.

Further, a client's application can be migrated and hosted on a Sensor Cloud service provider's platform. This scenario is similar to hosting applications on traditional cloud computing platforms like Amazon AWS or Microsoft Azure.

The security risk assessment framework proposed in this dissertation for client's application which may be migrated to a cloud service provider's (CSP) platform assess the security provided by a CSP with respect to the security risks present in the client's application. This is carried out in two notable phases, namely - application risk assessment using cloud security domains, and CSP's security assessment. The first phase introduces the cloud security domains while performing risk assessment for a client's application during its design phase. This makes the outlined techniques economical in nature, allows client's to test and try out different application architecture, and reduces their apprehensions of application security when they migrate to a CSP's cloud platform. The seconds phase aggregates and analyses the security provided by different CSPs in the market by perusing their publicly available security documents in registries like CSA's Security, Trust, and Assurance Registry (STAR). The proposed framework can further aid clients in developing optimal cloud migration minimizing cost and security risks by accounting for the outputs from phase one and two.

Finally, clients/users are supposed to be present in the center of any service provisioning. Therefore, infrastructure and application security cannot be treated as constant parameter based on their architecture and security policies. They need to account for the variable security preferences of clients and accordingly adapt to the specified preferences. This paves the foundation to customizing security measures and risk assessment techniques, and it is the premise of the research presented in this dissertation for addressing security assessment related to the users. The proposed framework introduces the variable security composition paradigm enabling users to specify their security preferences and optimally provision networks composed of feasible devices from the infrastructure. The framework also introduces a multilayer perceptron utilizing deep learning emotion classification techniques and user's quality of experience information gathered from their physiological data to accurately validate the satisfaction feedback from using the provisioned services.

REFERENCES

[1] Y. Wang, G. Attebury, and B. Ramamurthy, "A survey of security issues in wireless sensor networks," *IEEE Communications Surveys Tutorials*, vol. 8, pp. 2–23, 2006.

[2] E. Shi and A. Perrig, "Designing secure sensor networks," *Wireless Communications, IEEE*, vol. 11, no. 6, pp. 38–43, 2004.

[3] G. Padmavathi and D. Shanmugapriya, "A survey of attacks, security mechanisms and challenges in wireless sensor networks," *CoRR*, vol. abs/0909.0576, 2009.

[4] J. P. Walters, Z. Liang, W. Shi, and V. Chaudhary, "Wireless sensor network security: A survey, in book chapter of security," in *in Distributed, Grid, and Pervasive Computing, Yang Xiao (Eds*. CRC Press, 2007, pp. 0–849.

[5] A. Wood and J. Stankovic, "Denial of service in sensor networks," *Computer*, vol. 35, no. 10, pp. 54–62, 2002.

[6] W. Xu, K. Ma, W. Trappe, and Y. Zhang, "Jamming sensor networks: attack and defense strategies." *IEEE Network*, vol. 20, no. 3, pp. 41–47, 2006.

[7] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: Attacks and countermeasures," in *In First IEEE International Workshop on Sensor Network Protocols and Applications*, 2002, pp. 113–127.

[8] B. Kannhavong, H. Nakayama, Y. Nemoto, N. Kato, and A. Jamalipour, "A survey of routing attacks in mobile ad hoc networks," *Wireless Communications, IEEE*, vol. 14, no. 5, pp. 85–91, 2007.

[9] J. Newsome, E. Shi, D. Song, and A. Perrig, "The sybil attack in sensor networks: analysis defenses," in *Information Processing in Sensor Networks, 2004. IPSN 2004. Third International Symposium on*, 2004, pp. 259–268.

[10] S. Mauw and M. Oostdijk, "Foundations of attack trees," in *ICISC'05*, 2005, pp. 186–198.

[11] C. Phillips, "A graph-based system for network-vulnerability analysis," *in Proceedings of the 1998 workshop on New security paradigms*. ACM Press, 1998, pp. 71–79.

[12] J. Lee, H. Lee, and H. P. In, "Scalable attack graph for risk assessment," in *Proceedings of the 23rd international conference on Information Networking*, ser. ICOIN'09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 78–82.

[13] Steve Piper. Definitive guide to next generation vulnerability management. CyberEdge Group LLC, 2013.

[14] O. Sheyner and J. Wing, "Tools for generating and analyzing attack graphs," in *proceedings of formal methods for components and objects, lecture notes in computer science*, 2004, pp. 344–371.

[15] L. Gallon and J. J. Bascou, "Using cvss in attack graphs," in *Proceedings of the 2011 Sixth International Conference on Availability, Reliability and Security*, ser. ARES '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 59–66.

[16] "The national vulnerability database (nvd)," 2005. [Online]. Available: https://nvd.nist.gov/

[17] P. Mell, K. Scarfone, and S. Romanosky, *A Complete Guide to the Common Vulnerability Scoring System Version 2.0*, 1st ed., NIST and Carnegie Mellon University, June 2007. [Online]. Available: http://www.first.org/cvss/v2/guide

[18] M. Frigault and L. Wang, "Measuring network security using bayesian network-based attack graphs," in *Proceedings of the 2008 32nd Annual IEEE International Computer Software and Applications Conference*, ser. COMPSAC '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 698–703.

[19] R. Dantu, K. Loper, and P. Kolan, "Risk management using behavior based attack graphs," in *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04) Volume 2 - Volume 2*, ser. ITCC '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 445–.

[20] Y. Liu and H. Man, "Network vulnerability assessment using bayesian networks," vol. 5812, pp. 61–71, 2005.

[21] S. Houmb and V. Nunes Leal Franqueira, "Estimating toe risk level using cvss," in *Proceedings of the Fourth International Conference on Availability, Reliability and Security (ARES 2009 The International Dependability Conference)*, ser. IEEE Conference Proceedings. Los Alamitos: IEEE Computer Society Press, March 2009, pp. 718–725.

[22] Adam Shostack. Experiences threat modeling at microsoft. Modeling Security Workshop, Toulouse, 2008.

[23] M. Abi-Antoun, D. Wang, P. Torr, Checking threat modeling data flow diagrams for implementation conformance and security, in: Proceedings of the Twenty-second IEEE/ACM International Conference on Automated Software Engineering, ASE '07, ACM, 2007, pp. 393–396.

[24] Marco M. Morana. Managing software security risks using application threat modeling. *IMI Security Symposium and Expo*, 2008.

[25] C. Mockel and A.E. Abdallah. Threat modeling approaches and tools for securing architectural designs of an e-banking application. In *Information Assurance and Security (IAS), 2010 Sixth International Conference on*, pages 149–154, Aug 2010.

[26] Danny Dhillon. Developer-driven threat modeling: Lessons learned in the trenches. *IEEE Security & Privacy*, 9:41–47, 2011.

[27] D. Verdon, G. McGraw, Risk analysis in software design, IEEE Security Privacy 2 (4) (2004) 79–84.

[28] J. Whitmore, S. Türpe, S. Triller, A. Poller, C. Carlson, Threat analysis in the software development lifecycle, IBM Journal of Research and Development 58 (1) (2014) 2:6–2:6.

[29] T. Sommestad, M. Ekstedt, H. Holm, The cyber security modeling language: A tool for assessing the vulnerability of enterprise system architectures, IEEE Systems Journal 7 (3) (2013) 363–373.

[30] M. R. Naeem, W. Zhu, A. A. Memon, A. Khalid, Using v-model methodology, uml process-based risk assessment of software and visualization, in: Proceedings of 2014 International Conference on Cloud Computing and Internet of Things, 2014, pp. 197–202.

[31] M. Almorsy, J. Grundy, A. S. Ibrahim, Automated software architecture security risk analysis using formalized signatures, in: Proceedings of the 2013 International Conference on Software Engineering, ICSE '13, IEEE Press, 2013, pp. 662–671.

[32] I. Williams, X. Yuan, Evaluating the effectiveness of microsoft threat modeling tool, in: Proceedings of the 2015 Information Security Curriculum Development Conference, InfoSec '15, ACM, 2015, pp. 9:1–9:6.

[33] M. Frydman, G. Ruiz, E. Heymann, E. Csar, B. P. Miller, Automating Risk Analysis of Software Design Models, The Scientific World Journal, 2014.

[34] L. B. Othmane, P. Angin, H. Weffers, B. K. Bhargava, Extending the agile development process to develop acceptably secure software, IEEE Transactions on Dependable and Secure Computing. 11 (6) (2014) 497–509.

[35] M. Palanivel, K. Selvadurai, Risk-driven security testing using risk analysis with threat modeling approach, SpringerPlus 3 (1) (2014) 754.

[36] M. Felderer, C. Haisjackl, R. Breu, J. Motz, Integrating manual and automatic risk assessment for risk-based testing, in: Software Quality. Process Automation in Software Development: 4th International Conference, SWQD 2012, Vienna, Austria, January 17-19, 2012, pp. 159–180.

[37] J. Bozic, F. Wotawa, Security testing based on attack patterns, in: 2014 IEEE Seventh International Conference on Software Testing, Verification and Validation Workshops, 2014, pp. 4–11.

[38] L. Coppolino, S. D'Antonio, G. Mazzeo, and L. Romano, "Cloud security: Emerging threats and current solutions," *Computers and Electrical Engineering*, vol. 59, no. Supplement C, pp. 126 – 140, 2017.

[39] J. Singh, T. Pasquier, J. Bacon, H. Ko, and D. Eyers, "Twenty security considerations for cloud-supported internet of things," *IEEE Internet of Things Journal*, vol. 3, no. 3, pp. 269–284, June 2016.

[40] R. Latif, H. Abbas, S. Assar, Q. Ali, Cloud computing risk assessment: A systematic literature review, in: Future Information Technology: FutureTech, 2014, pp. 285–295.

[41] A. U. Khan, M. Oriol, M. Kiran, M. Jiang, K. Djemame, Security risks and their management in cloud computing, in: 4th IEEE International Conference on Cloud Computing Technology and Science Proceedings, 2012, pp. 121–128.

[42] R. Miao, R. Potharaju, M. Yu, N. Jain, The dark menace: Characterizing network-based attacks in the cloud, in: Proceedings of the 2015 Internet Measurement Conference, IMC '15, 2015, pp. 169–182.

[43] A. Sangroya, S. Kumar, J. Dhok, V. Varma, Towards Analyzing Data Security Risks in Cloud Computing Environments., Vol. 54 of Communications in Computer and Information Science, 2010, pp. 255–265.

[44] M. Theoharidou, N. Tsalis, D. Gritzalis, In cloud we trust: Risk-assessment-as-a-service, in: IFIP International Conference on Trust Management(IFIPTM), Malaga, Spain, June 3-7, 2013, pp. 100–110.

[45] M. L. Hale, R. Gamble, Secagreement: Advancing security risk calculations in cloud services, in: 8th IEEE World Congress on Services, 2012.

[46] A. J. Ferrer, F. HernáNdez, J. Tordsson, E. Elmroth, A. Ali-Eldin, C. Zsigri, R. Sirvent, J. Guitart, R. M. Badia, K. Djemame, W. Ziegler, T. Dimitrakos, S. K. Nair, G. Kousiouris, K. Konstanteli, T. Varvarigou, B. Hudzia, A. Kipp, S. Wesner, M. Corrales, N. Forgó, T. Sharif, C. Sheridan, Optimis: A holistic approach to cloud service provisioning, Future Gener. Comput. Syst. 28 (1) (2012) 66–77.

[47] K. Djemame, D. Armstrong, J. Guitart, M. Macias, A risk assessment framework for cloud computing, IEEE Transactions on Cloud Computing 4 (3) (2016) 265–278.

[48] E. Cayirci, A. Garaga, A. Santana de Oliveira, Y. Roudier, A risk assessment model for selecting cloud service providers, Journal of Cloud Computing 5 (1) (2016) 14.

[49] S. M. Habib, S. Ries, and M. Mühlhäuser and Prabhu Varikkattu, "Towards a trust management system for cloud computing marketplaces: using caiq as a trust information source," *Security and Communication Networks*, vol. 7, pp. 2185–2200, 2014.

[50] A. Taha, R. Trapero, J. Luna, and N. Suri, "Ahp-based quantitative approach for assessing and comparing cloud security," *2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, vol. 00, pp. 284–291, 2014.

[51] V. Casola, A. D. Benedictis, M. Rak, J. Modic, and M. Erascu, "Automatically enforcing security slas in the cloud," *IEEE Transactions on Services Computing*, vol. PP, no. 99, pp. 1–1, 2016.

[52] C. A. Ardagna, R. Asal, E. Damiani, and Q. H. Vu, "From security to assurance in the cloud: A survey," *ACM Computing Survey*, vol. 48, no. 1, pp. 2:1–2:50, jul 2015.

[53] S. Lins, S. Schneider, and A. Sunyaev, "Trust is good, control is better: Creating secure clouds by continuous auditing," *IEEE Transactions on Cloud Computing*, vol. PP, no. 99, pp. 1–1, 2017.

[54] J. Modic, R. Trapero, A. Taha, J. Luna, M. Stopar, and N. Suri, "Novel efficient techniques for real-time cloud security assessment," *Computers and Security*, vol. 62, pp. 1–18, 2016.

[55] C. Tang and J. Liu, "Selecting a trusted cloud service provider for your saas program," *Journal of Computers and Security*, vol. 50, no. C, pp. 60–73, may 2015.

[56] K. Kritikos, K. Magoutis, and D. Plexousakis, "Towards knowledge-based assisted iaas selection," in *2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, Dec 2016, pp. 431–439.

[57] V. Chang and M. Ramachandran, "Towards achieving data security with the cloud computing adoption framework," *IEEE Transactions on Services Computing*, vol. 9, no. 1, pp. 138–151, 2016.

[58] A. Yautsiukhin, R. Scandariato, T. Heyman, F. Massacci, W. Joosen, Towards a quantitative assessment of security in software architectures, Proceedings of the The Third International Conference on Availability, Reliability and Security, ARES 2008 (2008) 921–928.

[59] P. Saripalli, B. Walters, Quirc: A quantitative impact and risk assessment framework for cloud security, in: 2010 IEEE 3rd International Conference on Cloud Computing, 2010, pp. 280–288.

[60] N. Gonzalez, C. Miers, F. Redigolo, T. Carvalho, M. Simplicio, M. Naslund, M. Pourzandi, A quantitative analysis of current security concerns and solutions for cloud computing, in: Proceedings of the 2011 IEEE Third International Conference on Cloud Computing Technology and Science, 2011, pp. 231–238.

[61] K. Hashizume, N. Yoshioka, and E. B. Fernandez, "Misuse patterns for cloud computing," in *Proceedings of the 2nd Asian Conference on Pattern Languages of Programs*, ser. AsianPLoP '11, 2011, pp. 1–6.

[62] H. W. Nobukazu Yoshioka, K. Maruyama, A survey on security patterns, Progress in Informatics (5) (2008) 35–47.

[63] J. McDermott, C. Fox, Using abuse case models for security requirements analysis, in: Proceedings of the 15th Annual Computer Security Applications Conference, ACSAC '99, IEEE Computer Society, 1999, pp. 55–.

[64] X. Yuan, E. B. Nuakoh, I. Williams, H. Yu, Developing abuse cases based on threat modeling and attack patterns, Journal of Software 10 (4) (2015) 491–498.

[65] G. Sindre, A. L. Opdahl, Eliciting security requirements with misuse cases, Requirements Engineering 10 (1) (2005) 34–44.

[66] M. El-Attar, Using smcd to reduce inconsistencies in misuse case models: A subject-based empirical evaluation, J. Syst. Softw. 87 (2014) 104–118.

[67] L. Rostad, An extended misuse case notation: Including vulnerabilities and the insider threat, In Proc. 12th Working Conf. Requirements Eng.: Foundation for Software Quality (R EFSQ).

[68] M. M. Kashef and J. Altmann, *A Cost Model for Hybrid Clouds*. Springer Berlin Heidelberg, 2012, pp. 46–60.

[69] A. Khajeh-Hosseini, D. Greenwood, and I. Sommerville, "Cloud migration: A case study of migrating an enterprise IT system to iaas," *CoRR*, vol. abs/1002.3492, 2010.

[70] V. Tran, J. Keung, A. Liu, and A. Fekete, "Application migration to cloud: A taxonomy of critical factors," in *Proceedings of the 2nd International Workshop on Software Engineering for Cloud Computing*, ser. SECLOUD '11, pp. 22–28.

[71] M. Menzel and R. Ranjan, "Cloudgenius: Decision support for web server cloud migration," in *Proceedings of the 21st International Conference on World Wide Web*.

[72] V. Andrikopoulos, S. Strauch, and F. Leymann, "Decision Support for Application Migration to the Cloud: Challenges and Vision," in *Proceedings of the 3rd International Conference on Cloud Computing and Service Science (CLOSER'13)*. SciTePress, May 2013, pp. 1–7.

[73] B. Johnson and Y. Qu, "A holistic model for making cloud migration decision: A consideration of security, architecture and business economics," in *10th IEEE International Symposium on Parallel and Distributed Processing with Applications, ISPA*, 2012, pp. 435–441.

[74] A. Gunka, S. Seycek, and H. Kühn, "Moving an application to the cloud: An evolutionary approach," in *Proceedings of the 2013 International Workshop on Multi-cloud Applications and Federated Clouds*, ser. MultiCloud '13, pp. 35–42.

[75] P. Jamshidi, A. Ahmad, and C. Pahl, "Cloud migration research: A systematic review," *IEEE Trans. on Cloud Computing*, vol. 1, no. 2, pp. 142–157, July 2013.

[76] I. P. Egwutuoha, S. Chen, D. Levy, and B. Selic, "A fault tolerance framework for high performance computing in cloud." in *CCGRID*. IEEE Computer Society, 2012, pp. 709–710.

[77] Nist and Aroms, Emmanuel, "NIST Special Publication 800-53 Revision 3 Recommended Security Controls for Federal Information Systems and Organizations". CreateSpace, Paramount, CA, 2012.

[78] European Network and Information Security Agency (ENISA). Cloud computing: Benefits, risks and recommendations for information security, 2009.

[79] S. Madria, V. Kumar, and R. Dalvi, "Sensor cloud: A cloud of virtual sensors," *IEEE Software*, vol. 31, no. 2, pp. 70–77, 2014.

[80] I. Ray and N. Poolsapassit, "Using attack trees to identify malicious attacks from authorized insiders," in *Proceedings of the 10th European conference on Research in Computer Security*, ser. ESORICS'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 231–246.

[81] J. Dawkins, C. Campbell, and J. Hale, "Modeling network attacks: Extending the attack tree paradigm," in *Workshop on Statistical and Machine Learning Techniques in Computer Intrusion Detection*, 2002, pp. 75–86.

[82] N. Poolsappasit, R. Dewri, and I. Ray, "Dynamic security risk management using bayesian attack graphs," *IEEE Trans. Dependable Secur. Comput.*, vol. 9, no. 1, pp. 61–74, Jan. 2012.

[83] O. Sheyner and J. Wing, "Tools for generating and analyzing attack graphs," in *proceedings of formal methods for components and objects, lecture notes in computer science*, 2004, pp. 344–371.

[84] G. Brunette and R. Mogull. Security guidance for critical areas of focus in cloud computing v2.1. *Cloud Security Alliance*, 2009.

[85] Payment card industry (pci) data security standard, v3.0. PCI Security Standards Council, LLC. 2013. https://www.pcisecuritystandards.org/documents/PCI_DSS_v3.pdf

[86] The health insurance portability and accountability act of 1996 (hippa) privacy and security rules. hhs.gov/sites/default/files/privacysummary.pdf, 2006.

[87] Cloud Security Alliance, "CSA security, trust and assurance registry (star)," 2016, [online; accessed 14-Sep-2017]. [Online]. Available: cloudsecurityalliance.org/star

[88] L. D. Xu, W. He, and S. Li, "Internet of things in industries: A survey," *IEEE Transactions on Industrial Informatics*, vol. 10, pp. 2233–2243, Nov 2014.

[89] S. M. R. Islam, D. Kwak, M. H. Kabir, M. Hossain, and K. S. Kwak, "The internet of things for health care: A comprehensive survey," *IEEE Access*, vol. 3, pp. 678–708, 2015.

[90] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys Tutorials*, vol. 17, pp. 2347–2376, Fourthquarter 2015.

[91] M. Nardelli, S. Nastic, S. Dustdar, M. Villari, and R. Ranjan, "Osmotic flow: Osmotic computing + iot workflow," *IEEE Cloud Computing*, vol. 4, pp. 68–75, March 2017.

[92] A. Yachir, Y. Amirat, A. Chibani, and N. Badache, "Event-aware framework for dynamic services discovery and selection in the context of ambient intelligence and internet of things," *IEEE Transactions on Automation Science and Engineering*, vol. 13, pp. 85–102, Jan 2016.

[93] A. Rullo, D. Midi, E. Serra, and E. Bertino, "Pareto optimal security resource allocation for internet of things," *ACM Trans. Priv. Secur.*, vol. 20, pp. 15:1–15:30, Oct. 2017.

VITA

Amartya Sen was born in Kolkata, India in 1990. His initial schooling took place in three different schools, spread across three states in India. He earned his baccalaureate degree in Computer Science Engineering from National Institute of Technology, Durgapur, India in 2012.

Amartya joined Missouri University of Science and Technology, formerly University of Missouri at Rolla, Missouri, in 2012. He received a Master of Science in Computer Science from Missouri University of Science and Technology in May 2018. He received a PhD in Computer Science from Missouri University of Science and Technology in July 2018. During his course as a PhD candidate, he worked under the supervision of Dr. Sanjay Madria in the broad areas of security risk assessment techniques, Cloud security, and security in IoT environments. Amartya also mentored multiple undergraduate students during NSF funded summer research camps, and supported the Department of Computer Science undergraduate course curriculum as a graduate teaching assistant for three years.

He was the recipient of the Best PhD Forum presentation award in the 16th IEEE International Conference on Mobile Data Management, Pittsburgh, 2015, and also led his team to the national data challenge finals hosted by Teradata University Network at PARTNERS conference, Atlanta, 2016.