

Chapter One: Introduction

1.1. Background

The essence of security is to protect assets or resources of an organization. A database contains a company's most valuable asset, data [1, 2] that requires resources to be invested for its protection. In the last few years the world has experienced a global increase in data generated from a variety of sources for instance transactions, sensor devices, websites, social networks and so forth. In a few years data has grown from hundreds of gigabytes (GB), crossing into hundreds of terabytes (TB) and into petabytes (PB) [70]. This has given birth to the term Big Data to describe these massive volumes of data and their associated management technologies. "Big Data is data whose scale, diversity, and complexity require new architecture, techniques, algorithms, and analytics to manage it and extract value and hidden knowledge from it..." [3, 4]. Big data storage techniques have presented a breakthrough in achieving scalability, cost reduction, performance and flexibility in the management of Big Data. However, security has remained a daunting challenge in this technology as it was never prioritized in software development [5]. It is imperative that profound attention be directed to exploring the Security Requirements of Big Data environments.

Security is a characteristic of software systems which ensures prevention of circumstances leading to the loss of confidentiality, integrity, and availability of data [6]. These three abbreviated CIA are the goals at the core of software systems security as identified by several authors [6, 7, 8]. According to [9] Databases have the highest rate of breaches among all business assets. Security challenges in databases have been discovered to be even more profound in the Big Data environments. Big Data Databases assimilate a variety of data making them a target for attack by intruders, malicious crackers and other threats due to the criticality of data stored there in. This has led to serious security breaches leading to loss of data confidentiality,

integrity and availability, a situation with adverse impact on business operations. The systems may not be acceptable to customers as they cannot be depended upon [11] as this can lead to failure and subsequently losses being incurred by organizations [12, 13]. Most research work has attributed software failure to lack of consideration of security requirements for the complete system and neglecting that implementation of security mechanisms cannot be done randomly [7, 15, 16].

According to Bruce [14], "Security is a chain; it's only as secure as the weakest link. Security is a process, not a product."

Scrutinizing this statement, we identify that security should be taken as a process which covers the whole SDLC to ensure there is a holistic consideration of security requirements leading to ultimate implementation of adequate security mechanisms in a software system. Software Engineering principles have now been commonly adopted by the software engineering community in their work. However the focus has mainly been on the functional requirements which are given more gravity in the SDLC. Security has not been prioritized in software engineering to a greater extent. There is no cognizance to the fact that different software systems have different security needs and thus different security requirements which need to be thoroughly explored to ensure proper mechanisms are built into each unique software product.

Several authors have given various definitions and classifications of security requirements [8]. In [10] security requirements are defined as "constraints on the functions of the system, where these constraints operationalize one or more security goals". They are not functional requirements but rather constraints on them. They are classified under non-functional or quality requirements in [17, 18, 19].

Furkani et al [8] argues that the classification of Security Requirements as non-functional requirements brings about security limitations as a result of inadequate techniques to model the requirements. It is observed from various study that securing a system, there is need to model both the security requirements and functional requirements of the system. However, if security requirements are classified as functional requirement, calls for a complete modification

of techniques applied in the system development. Several authors agree that considering Security Requirements early in the SDLC is the way to ensure a secure and quality software is developed [19, 33]. Generally, Security Requirements Engineering is the process that encompasses security requirements elicitation, analysis, prioritization and specification. The output of Security Requirements Engineering is a security requirements specification document. This document helps the design engineers in selecting appropriate security mechanisms to be implemented.

1.2.Related Work

Several researches have been carried out on Security Requirements Engineering [10, 11, 19, 21, 22, 23, 24, 25]. However, none of these researches presented work in the Big Data domain. Most of security-related studies focus on trends of vulnerabilities and breaches, but Security Requirements Engineering practices that are actually adopted in industrial projects are unclear [22] more so in Big Data environments. According to [23], Security Requirements Engineering is important for developing secure systems, as it offers “techniques, methods, standards and systematic and repeatable procedures for tackling security requirement issues throughout the system development lifecycle”. In [11] a Security Requirements Engineering process is presented aimed at considering security requirements early in the system design by incorporating the Common Criteria into the SDLC. Various methodologies are proposed in literature for supporting Security Requirements Engineering activities especially elicitation and analysis [26, 27, 28, 29, 30, 31]. According to [27], “Security Requirements analysis is a key process of Security Requirements Engineering which includes the elicitation, analysis, verification and management of security requirements”. In a model proposed by Sommerville et al. [32], the elicitation activities are requirements discovery, analysis and negotiation while requirements specification is considered to be a modeling activity.

Few researches have considered security requirements for database related domains. Soler et. al [33] introduces a security requirement model for data warehousing and also presents a three-step process for modeling security requirements. They focus on information and quality-of-service requirements (including security) which they combine in an approach based on Model

Driven Architecture. Bertino and Sandhu [34] discuss database security focusing mainly on various access control models. They carry out their research on relational databases, object oriented databases and XML. Similarly [35, 36] discuss database security in terms of access control models. However, none of these papers explores Big Data security requirements. This scenario presents clear evidence that in as far as Security Requirements Engineering study is concerned, Big Data still remains unexplored yet it is one of the most vulnerable areas to security breaches in present day. It is highlighted in [37] that enforcing security in Big Data environments has been shown to be quite a mammoth task. This is due to several reasons including that the schema-less nature of aggregated data from multiple sources in distributed environments makes establishing access control quite a challenge. Also, ensuring integrity is very difficult because of the nature of Big Data. Change controls are difficult to implement. Aggregation of data across the enterprise means various types of data including sensitive data is in the same repository. In order to secure a Big Data store, “security controls need to be applied as close to the data as possible to protect from both external and internal threats” [38].

1.3.Motivation

The software engineering community worldwide has found a great deal of interest in research on security requirements. Quite a significant number of approaches and frameworks have been proposed for application of security requirements in different domains. Many researchers advocate for building security into the system early in the SDLC as it ultimately reduces costs and results in a product able to withstand attacks [24]. Even though this research area has matured in the past few years, there is still very little focus on emerging areas like Big Data. This is a fairly new area of research and thus has not been thoroughly explored as shown by scarcity of relevant research papers in International journals focusing on Security Requirements Engineering in the Big Data domain. Literatures that have attempted to explore security issues in Big Data environments have not explored the security requirements in this area according to Security Requirements Engineering principles.

Databases have recently been at the top of assets targeted and compromised by various threats, as shown in Figure 1. This shows that database security needs to be given extra attention in systems security considerations. Moreover when dealing with Big Data environments, database security is a challenge due to complexity introduced by the increased volume, velocity and variety characteristics of the data involved as well as the distributed nature of the environment. This scenario has led to an increase in vulnerabilities and threats owing to the increased attack surface imposed by environments.

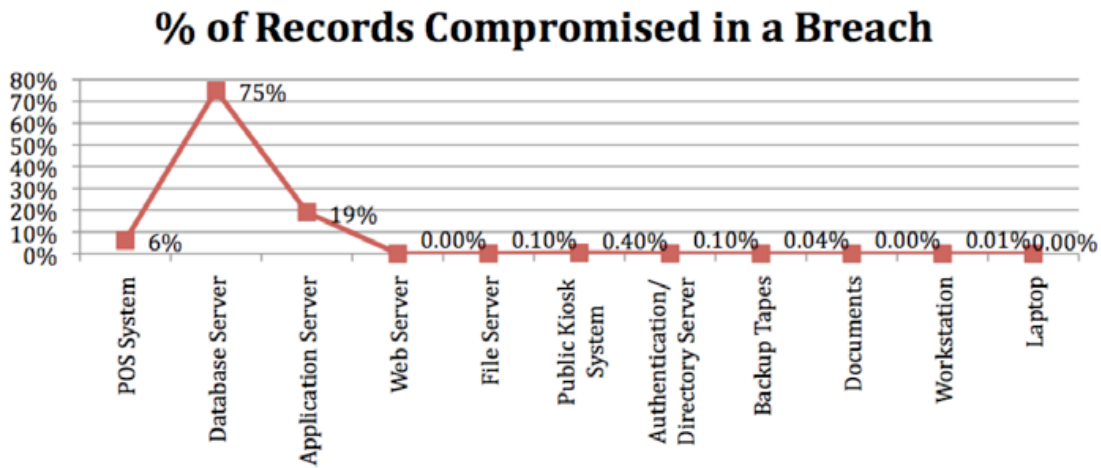


Figure 1: Percentage of Records Compromised per Asset Type [37]

Most research work on security requirements has focused mainly on extracting threats and corresponding cryptographical techniques to mitigate those threats. However, from our research on Big Data, we identified that these environments have numerous vulnerabilities which need to be addressed foremost so that threats may be reduced. Unless the data stored in the Big Data databases is secured, then no matter how we try to secure the front end loopholes still exist which may lead to security breaches. Thus as alluded before, controls ought to be enforced close to the data source which implies moving away from only investing on protecting the perimeter of the network to protecting data at the source. Our motivation in carrying out this work is derived from the realisation that proper Security Requirements Engineering in the wake of Big Data is the only way we can have confidence that data stored in these massive repositories is secure.

1.4. Research Problem

Many researchers have proposed techniques for elicitation and analysis of security requirements. However very few have gone further to prioritise and specify these security requirements. Additionally even fewer have developed tool support for Security Requirements Engineering activities to support their proposals. In addition to this problem, most of the work is also lacking validation by use of case studies to demonstrate applicability of the proposals in real use cases. This has created a problem where some of the requirements for different domains are not clear and adequate. If security requirements are not completely, consistently and clearly specified then the system designers may fail to incorporate fully the correct security mechanisms in the design leading to a product which may not be secure enough as required and may succumb to security breaches.

Database security requirements have not been given thorough consideration as most researches have focused primarily on application systems. This situation has been more profound in Big Data environments as this area is still fairly new with many complex emerging technologies integrated in its implementation. There has not been enough attention given to the security of Big Data database systems. This is highly inadequate as more resources will need to be invested in the applications so as to secure data which may still not prevent breach of security from the backend.

Various researches on Big Data databases have revealed that security has not been prioritized in the design of these massive databases. Focus has mainly been in addressing the scalability, performance and flexibility issues brought on by the increase of data generated by various systems. As a result the onus for securing the data has been left to the applications using these databases, a situation which is highly inadequate and ultimately expensive. Complete frameworks for Security Requirements Engineering for Big Data environments are still not available. Existing frameworks have not been applied in this area creating a gap that urgently needs to be closed.

So in this thesis we develop a framework for Security Requirements Engineering suitable for securing Big Data environments. We validate our work with two case studies and we also develop a tool to support the process.

Our proposed framework will help software engineers working on Big Data projects to elicit correct security requirements for the databases so that security can be built within these systems. This will in turn make application development more cost effective as fewer resources will need to be invested in securing the front end if the back end is secured.

1.5.Scope of Work

This work intends to provide a systematic and well defined way of eliciting, analyzing, prioritizing and specifying security requirements suitable for Big Data Databases. The thrust of our work is mainly on elicitation of security requirements through extraction of vulnerabilities inherent in database operations. Also usage of sequence diagrams in identifying vulnerable points in the generic actor interactions.

Our proposed framework first identifies generic actors for a database system, database operations are identified, vulnerabilities along with related threats are extracted by drawing sequence diagrams for actor interactions in each operation based on predefined relationships maintained in our repository, true security requirements are then elicited to mitigate all the threats and vulnerabilities. Security requirements proposed by Firesmith [16] will be adopted for use in this work. Analysis and prioritization of security requirements will follow. Finally with the aid of our tool support, specification of the requirements will be done. Our framework is applied on a case study of MongoDB and Cassandra which are instances of Big Data databases in the so called NoSQL family as will be explained in chapter 3. The vulnerabilities and threats will be extracted from a repository we maintain. Thus the scope of the work can be stated as follows:

- i. A framework for Security Requirements Engineering**
- ii. A technique for elicitation of security requirements for Big Data**

- iii. **Analyze and prioritize security requirements**
- iv. **Develop a software tool called SeCRUD that supports specification of security requirements in line with the proposed approach**

1.6. Thesis Structure

The rest of this thesis is organized as follows:

Chapter two gives an overview of Security Requirements Engineering, various approaches proposed in literature, modeling techniques available and an insight on some developed tools for supporting various Security Requirements Engineering activities. A summary is drawn to show the extent to which various research materials provide methodology to elicit, analyze and specify security requirements. In addition to that we also consider availability of case studies to validate these proposals and as well as tool support.

Chapter three explores the Big Data domain in depth. It touches on characteristics, technologies and challenges of Big Data environments. An insight is given on available storage options. Security issues of Big Data environments are discussed.

Chapter four focuses on our proposed framework for Security Requirements Engineering in the Big Data environments. Techniques for elicitation, analysis and prioritization of security requirements in the framework are explained in detail.

Chapter five features our case studies for validation of our proposed framework. The case studies are based on two popular Big Data databases MongoDB and Cassandra.

Chapter six will discuss our implementation in the form of a tool for supporting Security Requirements Engineering framework. This tool will automate all the steps in our framework to assist software engineers to carryout Security Requirements Engineering easily.

Chapter seven will give the conclusion and future work.

Chapter Eight will provide our Publication from this thesis

Chapter Two: An Overview of Security Requirements Engineering

In this chapter we explore the area of Security Requirements Engineering in depth. We first make an insight into what security requirements entail. Next we look at Security Requirements Engineering broadly. We consider different modeling techniques as applied to Security Requirements Engineering. Various Security Requirements Engineering approaches are also explored. We then look at tools that support Security Requirements Engineering process. Finally, in the last section we draw a summary of findings from different research papers in Security Requirements Engineering area.

2.1 Security Requirements

Security requirement are defined as “a manifestation of a high-level organizational policy into the detailed requirements of a specific system” in [39]. According to [10], adequate security requirements are those which ensure satisfaction of a system’s security goals. The contrary may lead to the system being unacceptable to the customers. Moreover, adequate Security Requirements ought to satisfy security requirements definition, assumptions and satisfaction criteria [10]. Firesmith [16] identified several security objectives and defined corresponding security requirements as identification, authentication, authorization, immunity, integrity, intrusion detection, non-repudiation, privacy, security auditing, survivability, physical protection and system maintenance security requirements. We define these with respect to database security.

2.1.1. Identification Requirements

Identification requirement is any security requirement that specifies the extent to which the database system shall describe a user or entity. It is assigning a unique label/identity to a user or entity involved in a database interaction to distinguish it from the rest.

2.1.2. Authentication Requirements

Authentication requirement is any security requirement that specifies the extent to which the database system shall be able to verify and make a confirmation of the identity of an entity wishing to access some resource in the database.

2.1.3. Authorization Requirements

Authorization requirement is any security requirement that specifies the extent to which the database system shall verify permissions and rights to access of the user to the requested resource.

2.1.4. Immunity Requirements

Immunity requirement is any security requirement that specifies the extent to which the database system shall provide an internal ability to defend itself from corruption and attack by malicious software.

2.1.5. Integrity Requirements

Integrity requirement is any security requirement that specifies the extent to which the database system shall ensure that data is protected from unauthorized modification through insertion, deletion or update of data.

2.1.6. Intrusion Detection Requirements

Intrusion detection requirement is any security requirement that specifies the extent to which the database system shall provide internal ability to monitor and identify attempts to gain unauthorized access to it. If proper audit trails are in place, they can aid intrusion detection.

2.1.7. Non-repudiation Requirements

Non-repudiation requirement is any security requirement that specifies the extent to which the database system shall be able to record all activities such that there will be no future denial of access to the database by any user or process. This requirement ensures that tamper-proof records are maintained to prevent denial of interactions that have occurred between users involving the database.

2.1.8. Privacy Requirements

Privacy requirement is any security requirement that specifies the extent to which the database system shall ensure personal control over data stored in the database. This means owner determines who sees what information and use it for what reason according to their discretion.

2.1.9. Security Auditing Requirements

Security auditing requirement is any security requirement that specifies the extent to which the database system shall monitor system for violations of security policy, malicious activities and recording there-of. “It is a systematic, measurable technical assessment of how the organization's security policy is employed at a specific site.”[17] Database security audit can assist in achieving several security-related objectives, for instance intrusion and fraud detection.

2.1.10. Survivability Requirements

Survivability requirement is any security requirement that specifies the extent to which the database system shall make a tradeoff between availability and integrity if database attack occurs. In [18] it is stated as, “we evaluate the survivability of a database system based mainly on the proportion of data objects that are not altered or destructed by intrusions and are available for user access at the same time.” This means integrity and availability may not be achieved 100% when the database system is operating in degraded mode. One will be lower than the other.

2.1.11. Physical Protection Requirements

Physical protection requirement is any security requirement that specifies the extent to which the database system shall protect itself from physical conditions and procedures that could

cause serious losses or damage to an organization. This includes protection from fire, natural disasters, theft and destruction.

2.1.12. System Maintenance Security Requirements

A system maintenance security requirement is any security requirement that specifies the extent to which the database system shall prevent any system modification be it authorized or not from disrupting its deployed security mechanism. Thus all security mechanism implemented in the database system ought to be reviewed timely.

2.2 Security Requirements Engineering

The main aim of Security Requirements Engineering is identifying all security requirements early in the software development process so that precise mechanisms may be incorporated in the overall design of the system being developed. According to Mellado et. al [24], Security Requirements Engineering is crucial to achieve secure software system in the SDLC. It comprises four phases that is: Security Requirement Elicitation, Security Requirement Analysis, Security Requirement Prioritization and Security Requirement Specification as shown in Figure 2.

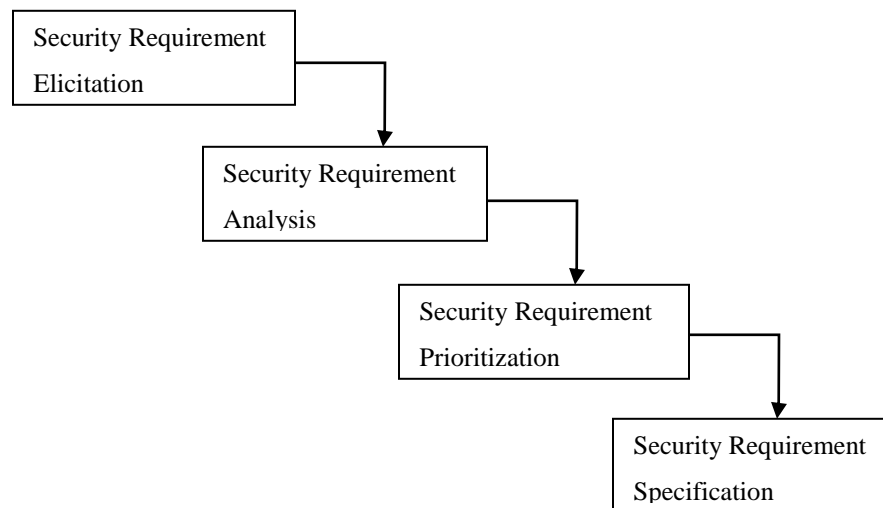


Figure 2: Security Requirements Engineering Process

2.2.1 Security Requirement Elicitation

This is the discovery of security requirements using various methods or techniques. The security requirements are drawn from different models and organisational policies on security [41].

2.2.2 Security Requirement Analysis

This phase is related to quality control since there is consideration of some quality attributes of the gathered security requirements. Grouping of the security requirements defined above is performed. Any ambiguities present are removed, analysis is made to check for consistency and completeness of the requirements. Risk analysis is performed using various methods for instance CRAMM, AHP [40].

2.2.3 Security Requirement Prioritization

After security requirements have been analyzed, prioritization is performed on the basis of risk measure. If the budget of the project is low only medium to high risk security requirements may be given implementation priority. The rest of the requirements may only be considered on availability of resources.

2.2.4 Security Requirement Specification

The prioritised requirements are specified so that we can keep track of all security requirements. Specification involves use of various methods for documenting the security requirements obtained in the prior phases [41]. Documents about the security requirements will be used as the basis for design and later phases. The output which is a security requirements specification document will be used in the following phases of the SDLC.

2.3 Security Requirements Modeling Techniques

Several modeling techniques are proposed for modeling misuse, threats, abuse, attacks and so forth in security requirements elicitation and analysis activities. Some have been applied individually while some papers have proposed combining several of these techniques for instance [42] proposes a combination of attack trees and misuse cases. This is aimed at improving the outcome of the Security Requirements Engineering activities.

2.3.1 Attack trees

Attack trees present method of representing the system security threats in a hierarchy on the basis of the nature of attacks possible and the different ways they can occur [27] [43] [44]. According to Schneier [45] attack trees model security threats by putting focus on attackers and the various means by which they may perpetrate an attack on the systems. A closely related concept of threat trees is suggested in [46], these use a tree structure for modeling attack goals and how to achieve them as shown in Figure 3.

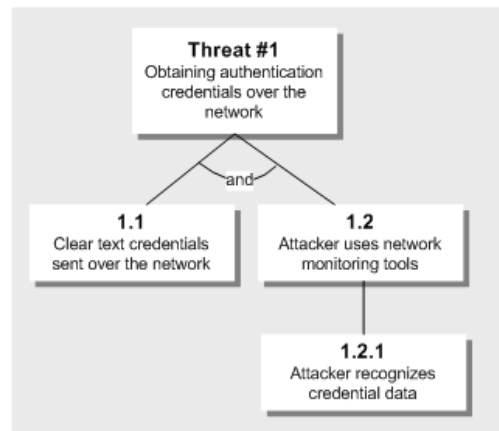


Figure 3: Attack Tree

2.3.2 Misuse Case

A use case is used to model desired system interaction with a user which makes them irrelevant to model functional requirements. In recent study, several authors have investigated the applicability of use cases in connection with security requirements and found them falling

short in this area [29][47-49]. Misuse cases extend the use case, in modeling behavior that is not desired by system owner. A misuse case is represented similarly to a use case except that the nature of interaction portrayed is undesirable in the former. However, a mis-actor as given in [29] is the actor who negatively interacts with the system in the misuse cases. In [48] a security requirements process for elicitation of security requirements with misuse cases is proposed. This process is then embedded in software development process. Advantages of Misuse Case are its ability to model a use case together with the threats which makes it very clear to see what threats are inherent in the system while the disadvantage is that it does not show the defense by the system. In [44] it was observed that Misuse Cases are simple to utilize, but the output is difficult to comprehend. Figure 4 below shows an example of a Misuse Case.

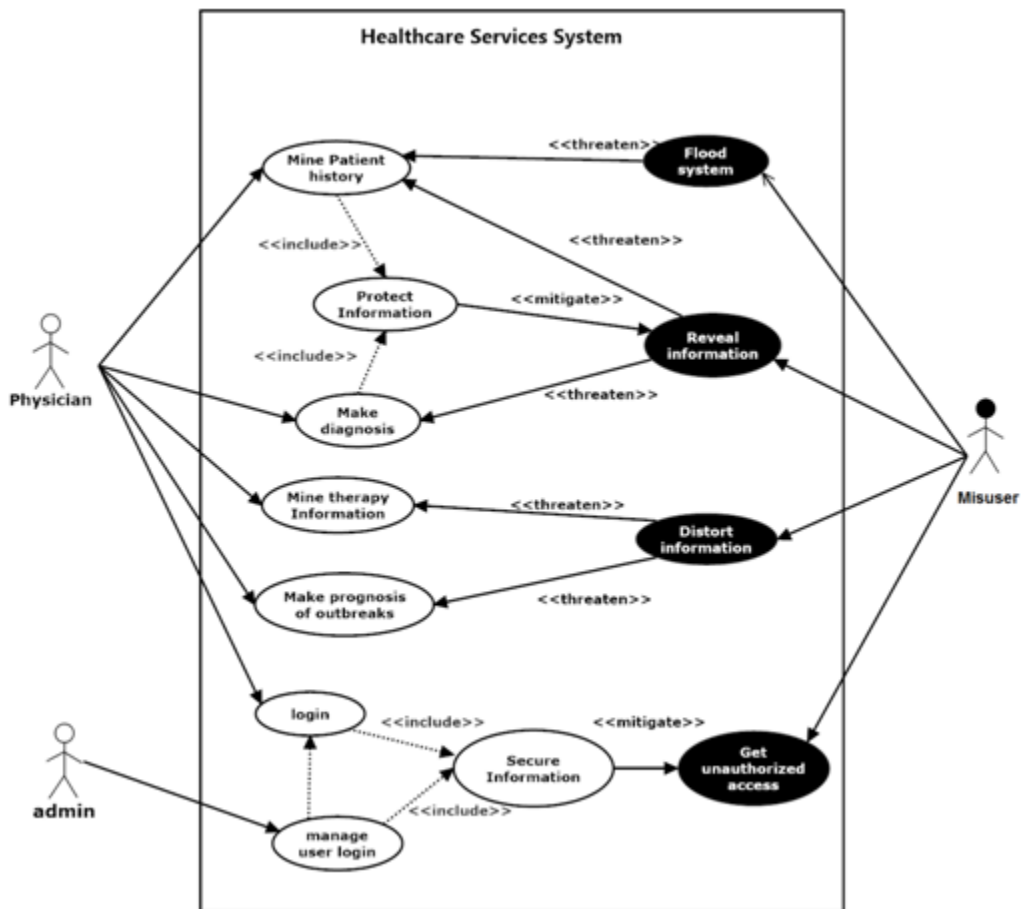


Figure 4: Misuse Case

2.3.3 Abuse Case

An abuse case is defined as a “specification of a type of complete interaction between a system and one or more actors, where the results of the interaction are harmful to the system, one of the actors, or one of the stakeholders in the system” [30]. An abuse case should describe the abuse of privilege in the behavior portrayed. Abuse cases can be described using the same strategy as for use cases [30]. Whereas use case diagrams show what system must do in terms of functionality, abuse case diagrams show the exact opposite that is what should not happen in the system. The advantages of an abuse case model are that it is simple and is easy to understand. Its disadvantage is that it does not clearly show the relationship between functionality and abuse. An abuse case is shown in Figure 5 below.

A similar approach using “problem frames as a means of defining system boundaries to provide a focus for early security threat analysis” is presented in [31]. They explore the concept of anti-requirement as the objective of a malicious user. Anti-requirements characterize undesirable behavior by a malicious user which alters the system state to be inconsistent with its requirements. An abuse frame represents a security threat.

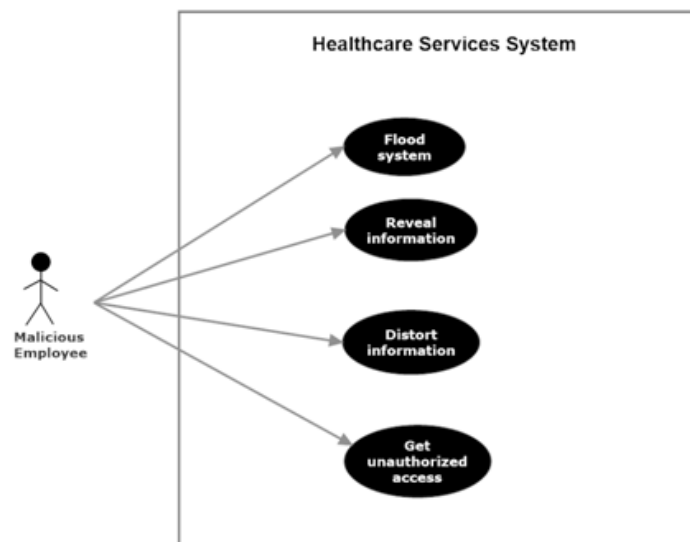


Figure 5: Abuse Case

2.3.4 Security Use Case

Security Use Cases are derived from Use Cases and are specifically used for elicitation of security requirements. The purpose of a security use case is specification of security requirements, prevention of an attack and mitigation of the impact of an attack if it happens. The difference between normal use cases and security use cases is that the former provide support for system interactions with users while the later provide system protection against attacks [50]. According to [51], “security use cases should be used to specify requirements that the application shall successfully protect itself from its relevant security threats”. Security Use Case has the advantage of associating functionality with the security requirement in the model. Figure 6 below shows a security Use Case.

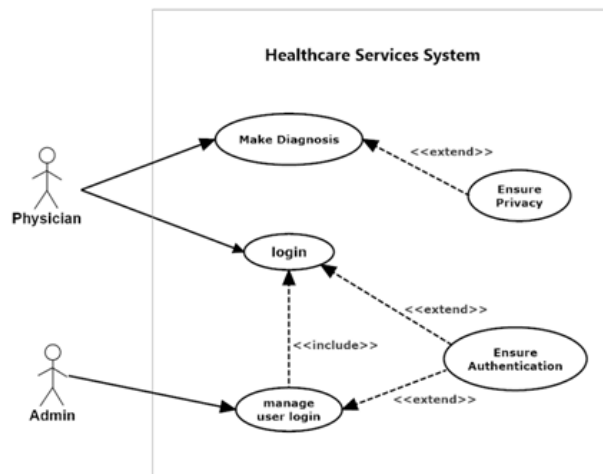


Figure 6: Security Use Case

2.3.5 Weaving Scenarios

Scenario consists of descriptions of the behavior of users, system and interaction between users and system [52][53]. In [52] a technique for weaving scenario fragments on the basis of security evaluation criteria suite is proposed. It considers successful scenarios; validate behavior there in and finally elicitation of the security requirements. In this paper an evaluation is made of this technique regarding inadequate behaviors resulting in vulnerabilities. Aspect-oriented technique for software development is applied to scenario specification as it is useful in security requirements elicitation. This method is very useful as it uses security evaluation criteria and its incorporation of validation in the requirements elicitation.

2.3.6 Formal Methods

According to [54], “Formal methods are typically used in the specification and verification of secure systems. From a life-cycle viewpoint, the specification typically represents either formal requirements or a formal step between informal requirements and design”. The Common Criteria (CC) is a standard for security requirements elicitation, specification, and analysis that was developed by the National Institute for Standards and Technology. It has two main components which are Protection Profiles and Evaluation Assurance Levels. “A Protection Profile (PPro) defines a standard set of security requirements for a specific type of product, such as a firewall. The Evaluation Assurance Level (EAL) defines how thoroughly the product is tested” [15]. According to [44] the Common Criteria has the disadvantage of difficulty in learning and using, but have the advantage of easy of analysis. Ware et al. [55] uses combination of use case diagrams and the common criteria for the elicitation of Security Requirements. They make use of actor profiling defined for all actor using the following seven fields as shown in Table 1:

Table 1: Common Criteria Actor Profile

Actor Profile	Description
Actor	Name of the actor.
Use Case	Name of Use Case
Type	Human, Cooperative Or Autonomous
Location	Location from which actor is interacting which can be local or remote
Private Exchange	True or false depending on whether private information is being exchanged or not
Secret Exchange	True or false depending on whether confidentiality of the information being exchanged is required
Association	Actor association with use case e.g. Read/Write/Read_Write/Retrieve/Send etc.

2.3.7 An Evaluation

As highlighted above, these models are not independent at all. There is great overlap which results in better security requirements development. Tøndel et al. in [42] suggest an approach for combining attack trees and misuse cases for use in the requirements engineering part of software engineering projects. The advantage for this is that integrated techniques provide an enhanced technique which may produce better security requirements models. Also tool support is very important in security requirements modeling. However, their proposal is not

applied to any particular system for validation thus it is not easy to ascertain its effectiveness. Dialo et al. [44] presented a comparison of three important approaches to security requirements which are Attack Misuse, Cases Trees and Common Criteria. Upon application of these approaches to the same area, they found out that each one of these approaches has pros and cons, and that together they can be complimentary. In [56] an approach is presented that reduces attack surface by effectively combining STRIDE for identifying threats and DREAD for calculating the risk involved.

2.4 Security Requirements Development Approaches

“Security requirements are usually specified to prevent any activities that may pose a threat to either the stakeholders or the system itself” [26]. Various authors have proposed many different approaches to Requirements Engineering in general. Some have been applied in the area of Security Requirements specifically. However some are applicable to functional and non-functional requirements thus making them applicable to Security Requirements also. These approaches are not particularly independent of each other but overlap in most instances.

2.4.1 Process Oriented Approach

Process oriented approaches are concerned with defining appropriate guidelines for users [41]. Methodology known as Security Quality Requirements Engineering (SQUARE) was designed by Mead et al. [19]. It is a Security Requirements development process methodology with nine steps and is used for elicitation, categorization, and prioritization of security requirements for IT systems and applications. Its main objective is to incorporate security into the SDLC early. The Accelerated Requirements Method (ARM) is a methodology designed for elicitation, categorization, and prioritization of security requirements [27]. Analytic Hierarchy Process (AHP) is a methodology proposed in [57] to prioritize requirements based on relative value and cost of security requirements. It is used in [27] to prioritize security requirements. Both ARM and AHP are used in SQUARE [14]. In [58] the SQUARE Method is compared with other methods like Attack trees, Tropos, Misuse case, Abuse case and formal methods. Security Requirements Engineering Process (SREP) is a process oriented as well as reuse based approach

proposed by Mellado et .al in [24]. It incorporates the Common Criteria in the SDLC while also providing a security resources. Haley et .al [10] present a process oriented framework for elicitation and analysis of Security Requirements. These approaches have an advantage of being easy to follow.

2.4.2 Re-use oriented Approach

Re-use oriented Approach is useful for more rapid security requirements specification which may be cost effective [58]. A 4 step process known as SIREN approach is proposed by Toval et al. [59]. Its steps are i) Asset identification, ii) Vulnerability identification, iii) Risk analysis, iv) Choosing countermeasures. Sindre et al. [60] proposed a reuse-based methodology almost similar to SIREN for analysing misuse cases and specifying security requirements. It has the following steps: i) Identify critical and/or vulnerable assets, ii) Determine security goals for each asset, iii) Identify threats to each asset iv) Analyze risk for each threat, v) Determine requirements.

2.4.3 Agent-oriented approach

Agent oriented approaches are important for understanding software systems, their environments, and users interacting with each other. The an agent-oriented approach to requirements engineering known as i* Framework was proposed by Liu et .al in [62]. It is a strategic modeling framework that helps in elicitation, identification and analysis of security requirements. In [27], two proposals for analysis of the security requirements of Information Systems are made. The first one uses i* framework to model security requirements supported by the si*-tool. The second one utilizes techniques such as use/misuse case, attack tree, risk assessment and so on in an eight steps process to obtain the final categorized and prioritized security requirements. i* framework is also used in Liu et.al [28] and also in [63]. The advantages of the framework are that it focuses on strategic analysis, which explores relationships at an intentional level and also has tool support that makes security requirements development easier.

In [64] a development framework named Tropos is proposed. It has four phases i) Early requirements ii) Late requirements iii) Architectural design iv) Detailed design. In [62], Tropos

adopts the concepts offered by i^* such as actor, and social dependencies among them. Tropos has the advantages of using same concepts across all phases of development. This methodology can be easily integrated with other approaches, and it actually complements i^* framework. Deriving from these advantages of Tropos, in [65] an extension to Tropos methodology, Secure Tropos is presented which is a comprehensive process, comprising various modeling activities and allows incorporation of security across the development process of multi-agent systems. It has the advantage of having tool support (T-Tool)

2.4.4 Goal Oriented Approach

Goal-oriented approaches to requirements engineering helps to expose objectives of users of a software system [22]. Knowledge Acquisition in autOmedated Specification (KAOS) [61] is a goal oriented requirements analysis method, developed by University of Oregon and university of Louvain. It provides a formal modeling of all software requirements in terms of for instance goals, constraints, assumptions, etc. KAOS focus on realizing and indicating the business goals, then specifying the requirements that infer to the business goals. The main advantage of KAOS over other requirements analysis methods is its ability to align requirements to business goals and objectives. FADES [66] is another goal-oriented approach to requirements and includes formal specifications refined to design and implementation of secure software systems. FADES has the advantages of enforcing security consideration from the early phases and throughout the SDLC which makes it possible to identify security vulnerabilities early.

2.4.5 Aspect Oriented Approach

Aspect-oriented approaches are useful in the elicitation and analysis phases of security requirements development [52]. Aspect-orientated approach aims at capturing the “crosscutting nature of threats and mitigations so that systems requirements are elicited in a more structured way” [67]. An aspect-oriented approach is presented In [67][68] to demonstrate the separation of security and functional requirements. Based on the use case-driven development, a specification of security threats and corresponding mitigations as aspects that encapsulate pointcuts and advice is made. As aspects for classes in aspect-oriented programming [69], threat and mitigation aspects are specified at the meta-level of use case descriptions. This has the advantages of

facilitating separation of security requirements from functional requirements and improves reuse of security solutions.

2.4.6 Viewpoints-Oriented Approach

Viewpoints are defined in Sommerville et al. [32] as “entities which may be used to structure the process of requirements elicitation and to structure the requirements specification”. They go on to elaborate that the approach considers that system requirements cannot be elicited by viewing the system from a single outlook instead requirements should be gathered from a number of different perspectives. In [32] viewpoints are categorized as viewpoints associated with system stakeholders and viewpoints associated with organizational and domain knowledge. They propose the Viewpoint Oriented Requirements Definition process which is a more generalized approach. Agarwal et al. [40] proposes an approach for security requirements using viewpoints based on the spiral model of software development. They call their approach Viewpoints Oriented Security Engineering Process (VOSREP). Its Requirements Engineering stage has the following three main phases: Requirements discovery and definition, Analysis and prioritization of the requirements and Management of the requirements.

2.5 Security Requirements Development Tools

Several tools are available to aid the SRE process by supporting different activities thereof. However some tools are not specifically designed for security requirements per se but are also applicable for supporting RE process in general.

2.5.1 Si*-tool

Si*-tool is based on the i* framework. In [27], si*-tool is used to model security requirements graphically and thus it supports security requirements development.

2.5.2 SRS-tool

Sang-soo et al. in [71] proposed a “security functional requirement specification development tool for application information system of organization” known as SRS-tool. This tool is proposed to aid their security requirements development process.

2.5.3 T-Tool

An automated tool, T-Tool is proposed in [72]. It provides automated support for the consistency validation of specifications verifies adherence to specific desired security characteristics. T-Tool also includes animations that provide immediate feedback to the user on its implications. In [65] it is used to support Secure Tropos.

2.5.4 SREPPLineTool

In paper [25] SREPPLineTool was developed which is a prototype of a CARE tool. SREPPLineTool prototype lets one apply the security quality requirements engineering process for software product lines (SREPPLine) process in a software Product Line development by offering activity support required. The tool implements the Security Reference Meta Model [73] using dynamic repositories of security artefacts, and guides in the execution of the process in a sequential way. Thus, it offers suggestions for suitable security artifacts at activities of SREPPLine process.

2.5.5 Suraksha

An Open Source Security Designers’ Workbench tool named Suraksha was proposed in [99]. Suraksha supports a range of features such as assets identification and prioritization, Textual representation of Misuse cases using Misuse case template, Co-representing Use and Misuse Cases, attack tree development.

2.5.6 NALASS

The authors in [74] present NALASS, which is a tool that provides automated support for the Natural Language Syntax and Semantics Requirements Engineering (NLSSRE) methodology. The NLSSRE methodology supports the major activities of Requirements

Engineering including Requirements Discovery, Analysis and Specification. However, limitation of this paper is that it does not consider security requirements. Its focus is merely on the automatic generation of the SRS and corresponding diagrams.

2.5.7 STORM

STORM, a UML based software engineering tool designed for the purpose of automating as much of the requirements specification as possible presented in [75]. Its goal was to create a tool capable of handling of text aspects of requirements specification and use case modeling. STORM provides support for requirements engineering and use case modeling. It is motivated by the need created by inexistence of CASE Tools for capturing requirements as well as textual descriptions of use cases and scenarios. STORM is not very different from NALASS highlighted in [74]. Both tools are not designed for security requirements specifically. However we cannot rule out their applicability for security requirements development until they have been applied to that area unsuccessful.

2.6 Summary of Security Requirements Engineering research findings

We draw a summary in Table 2 from some of the different research papers we studied. This shows to what extend proposed approaches cover the whole Security Requirements Engineering process and whether techniques used are risk based or not. We also consider availability of tool support and if there is a case study to support the research.

Table 2: Security Requirements Engineering research findings

Reference	Technique used (Risk-based)	Elicitation	Analysis	Specification	Tool Support	Case Study or Experiment
Donald Firesmith, 2003	Security Use Cases, No risk analysis	No elicitation mentioned	Analysis using security use case	Specification	None	Short Reference to UNIX and CORBA
Luncheng Lin, Bashar Nuseibeh, Darrel Ince, Michael Jackson, Jonathan Moffett, 2003	Abuse Frames, No risk analysis	Elicitation using	Analysis using Abuse Frames	No specification	None	traffic light system
Jing-Song CUI, Da ZHANG, 2007	I* framework, Artifacts, Risk Assessment	Use of ARM for eliciting requirements	Main focus on Requirements analysis	No specification	Si*-tool	Disaster Recovery System
Hiroga Itoga, Atsushi Ohnishi, 2007	Weaving Scenarios, No risk analysis	Security Requirements Elicitation	No reference to analysis	No specification	None	Book Selling System, two experiments performed
Ashish Agarwal, Daya Gupta 2008	Viewpoint Oriented Security Engineering Risk analysis	Elicitation of security requirements together with functional requirements	Analysis and prioritization	Specification	None	Online System
Charles B. Haley, Robin Laney, Jonathan D. Moffett, 2008	A Framework for elicitation and Analysis of Security Requirements, Some risk consideration	Elicitation	Analysis	No specification	None	Air traffic control technology evaluation project
Daniel Mellado, Eduardo Fernández-Medina, Mario Piattini, 2008	SREP-Security Requirements Engineering Process, Risk assessment	Some elicitation	Some analysis in security requirements scoping, prioritization	No specification	SREPPLine Tool	Software Product Line development
John McDermott and Chris Fox, 2009	Abuse Case, No risk analysis	Elicitation	Analysis	No specification	None	Internet based Information Security Lab
S. B. G, V. K. Maurya, E. Jangam, M. S. V, A. K. Talukder, and A. R. Pais, 2009	Security Designers' Workbench	Elicitation of security requirements together with functional requirements	Some reference to analysis	No specification	Suraksha	E-commerce application
Daniel Mellado, Jesús Rodríguez, Eduardo Fernández-Medina and Mario Piattini,	Some risk assessment	Security quality requirements engineering process for software product lines	Some analysis in security requirements scoping, prioritization also	No specification	SREPPLine Tool	Customer Relationship Management system

2009		(SREPPLine)					
Inger Anne Tøndel, Jostein Jensen, Lillian Røstad, 2010	Combination of misuse cases and attack trees	Approach for security requirements elicitation	No reference to analysis	No specification	tool proposed for future work	No application to any system for validation	
R. Hassan , M. Eltoweissy, S. Bohner S. El-Kassas, 2010	FADES, No risk analysis	No focus on elicitation	Uses KAOS requirements model for analysis	B method is used for specification	FADES tool	Controlled experiment Practitioners and Experts studies	
Kenneth Kofi Fletcher, Xiaoqing (Frank) Liu, 2011	High-order object oriented modeling technique, no risk analysis	Elicitation from prevention and mitigation options	Context Object Diagram is used for analysis and relative priority analysis	Specification included	None	Pervasive Health Monitoring System	
Peter Karpati, Andreas L. Opdahl, Guttorm Sindre, 2011	Misuse case maps, No risk analysis	Elicitation of security vulnerabilities	Comparative analysis	No specification	None	Experiment to identify vulnerabilities and mitigations	

2.7 Conclusion

There is no universal definition of security requirements as shown by the varied explanations given by different authors on the topic. Several modeling techniques and approaches exist for Security Requirements Engineering phases. The approaches are not independent but are interleaved and overlap to a greater extent. Different modeling techniques are available for elicitation and analysis of security requirements. These include attack trees, misuse case, abuse case, security use case etc. Tool support is paramount as they assist in generating the models as well as providing the process for security requirements development.. It was observed that case studies are required to validate the existing techniques. Only when proposed methodology and techniques have been applied to a specific domain can it be evaluated for usefulness. Most researches were found to elicit and analyze security requirements only without specifying them.

Chapter Three: An Analysis of Big Data

In this chapter, we look at what the term Big Data entails. We highlight its characteristics according to what different authors have suggested. We explain the technologies for storing, manipulating and analysing Big Data. We explore Hadoop framework for processing, storage and analysis of Big Data. We give an overview of the NoSQL databases as a storage technology for Big Data and go on to discuss the security issues associated with Big Data environments.

3.1 Big Data Overview

It is important to note: 1 petabyte = 1 thousand terabytes = 1 million gigabytes = 10^{15} bytes

The term Big Data has created a massive hype and caught the interest of not only researchers but also business enterprises across the globe. For some it has become a major opportunity while yet for others quite a pain to deal with as it means investing a lot of money to avail required infrastructure and support. “Big Data is data whose scale, diversity, and complexity require new architecture, techniques, algorithms, and analytics to manage it and extract value and hidden knowledge from it...” [76][77]. Big Data refers to substantial volume of structured, semi- structured and unstructured data derived from various sources such as social data, machine generated data, traditional enterprise which is so large that it is difficult to process with traditional database and software techniques. Existing conventional software tools become inadequate to handle and process large data sets [78]. This massive data is not just more streams of data, but also entirely new ones [79]. Big Data today often deals with very large unstructured data sets, Examples of these are data produced by social networking or ecommerce organisations including Facebook, Google, LinkedIn, Twitter or Amazon, which analyse user statuses or search terms to trigger targeted advertising on user pages. Big Data represents large scale of data spread across hundreds or thousands of physical storage servers or nodes. The data is in different

types which include Relational Data (Tables/Transaction/Legacy Data), Text Data (Web), Semi-structured Data (XML), Graph Data (Social Network, Semantic Web), Streaming Data. Figure 7 below gives an overall view of what Big Data is all about.

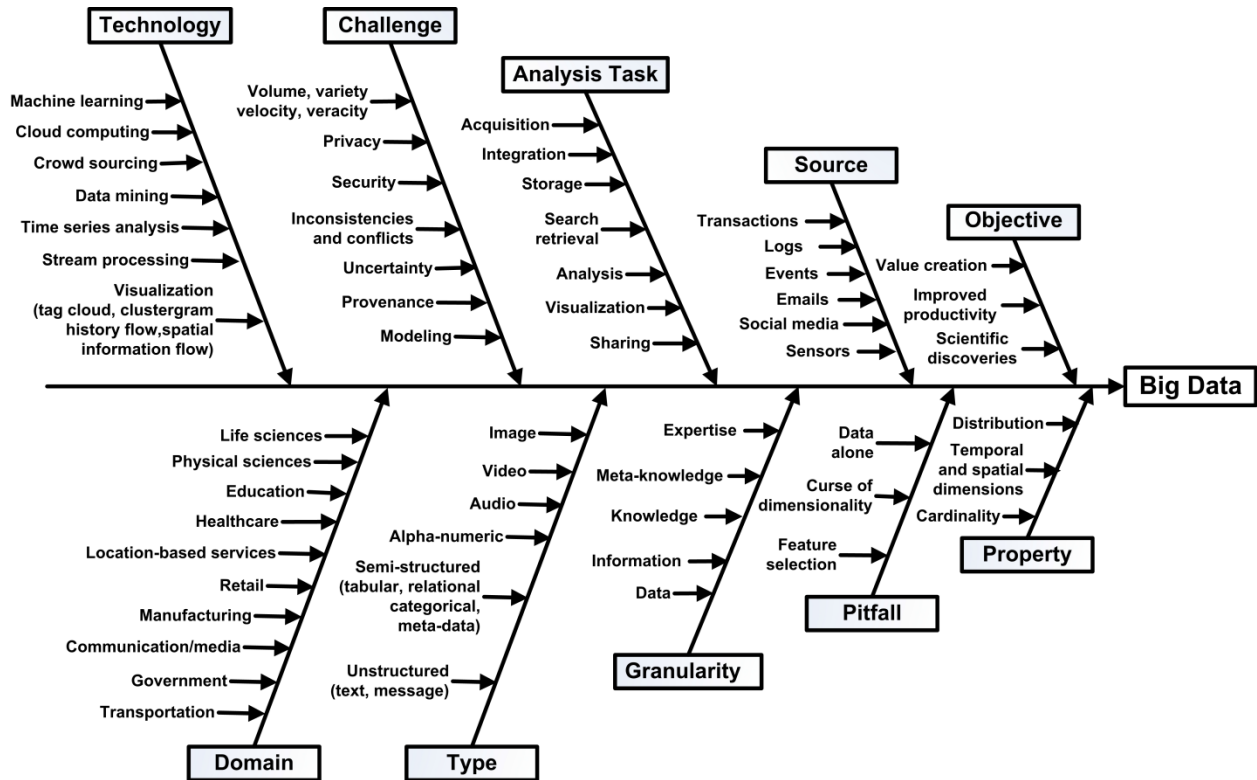


Figure 7: Dimensions in Big Data [80]

3.2 Characteristics of Big Data

Doug Laney in his note [81] specified that current business conditions and mediums are pushing traditional data management principles to their limits, giving rise to novel, more formalised approaches. He goes on to define three dimensions along which organisations have experienced exploded data management challenges as Volume, Velocity and Variety depicted in Figure 8. These dimensions have become ubiquitous more than ten years later and widely known as the “3Vs” framework for understanding and dealing with “big data”.

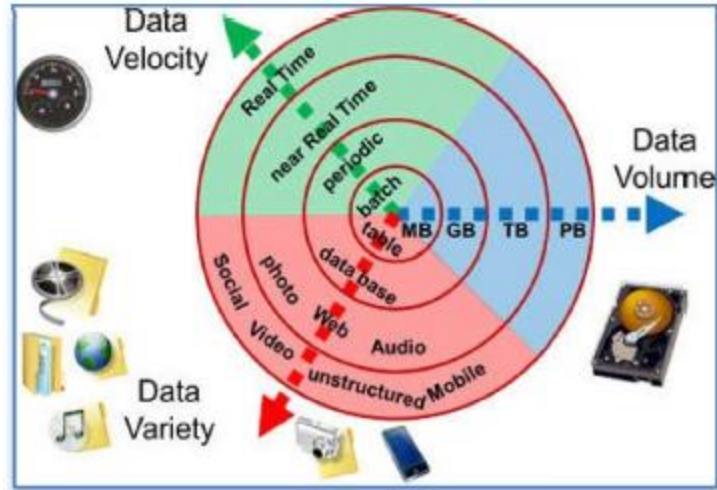


Figure 8: The 3Vs that define Big Data [82]

3.2.1 Volume

Volume refers to the amount of data stored for analysis (data at rest). E-commerce channels increase the depth and breadth of data available about a transaction (or any point of interaction) [81]. Volume means vast amount of data generated in every second and it is a scale characteristic [76]. Machine generated data are examples for these characteristics. Data may be generated from various sources and combine to increase the volume of data that has to be analyzed or stored. This data may grow to the tune of several Terabyte, Petabyte, Exabyte and Zettabyte in the near future. The volume also includes numerous data sources each of which may be holding very large quantities of data which when combined contributes to substantial volumes of data.

3.2.2 Velocity

Velocity refers to the speed of collecting or acquiring or generating or data processing (data in motion like streaming data). E-commerce has also increased point-of-interaction (POI) speed [81]. Data is generated at a very fast rate and therefore needs to be processed and analyzed in real time for it to be useful. Many of the Big Data sources are very dynamic [83].

3.2.3 Variety

Variety refers to the number of types of data (Data in many forms). Variety of incompatible data formats, non-aligned data structures, and inconsistent data semantics. A single application can be generating/collecting many types of data in various formats, types, and structures for instance text, numerical, images, audio, video, sequences, time series, social media data, multi-dimensional arrays and so forth [76]. It may be static data or streaming data. Complexity is brought on by having to integrate all these various data for analysis. In relation to the variety of data, Big Data can also be categorized as:

- **Structured data:** This type describes data which can fit well into a fixed relational schema within a standard SQL database. Structured data can be easily stored, manipulated and analyzed.
- **Semi-structured data:** This is a form of structured data that does not conform to an explicit and fixed schema. Examples include weblogs and social media feeds.
- **Unstructured data:** This type of data consists of formats which cannot easily be indexed into relational tables for manipulation. “Unstructured data represents almost every kind of data being produced like social media interactions, to recorded meetings, to handling of PDF documents, fax transfers, to emails and more” [9]. Examples include X-Ray images, audio and video files.

According to [1], “the 3Vs together describe a set of data and a set of analysis conditions that clearly define the concept of big data”. Many researchers have studied and some expanded on these three characteristics of Big Data. Additional Vs have been suggested for instance Veracity (uncertainty due to data inconsistencies), Value (important information can be deduced from the data). Complexity has also been added to the characteristics of Big Data.

3.3 Big Data Categories

There are several categories of Big Data mainly differentiated by the sources. Some have been summarized in Table 3 below.

Table 3: Categories of Big Data

Categories	Example
Web and social media data	Clickstream and interaction data from social media such as Facebook, Twitter, LinkedIn, and blogs. It can also include health plan websites, smartphone apps, etc.
Machine-to-machine data	Readings from sensors, meters, and other devices.
Big transaction data	Health care claims and other billing records increasingly available in semi-structured and unstructured formats.
Biometric data	Fingerprints, genetics, handwriting, retinal scans, and similar types of data. This would also include X-rays and other medical images, blood pressure, pulse and pulse-oximetry readings, and other similar types of data.
Human-generated data	Unstructured and semi-structured data such as electronic medical records (EMRs), physicians' notes, email, and paper documents

3.4 Big Data Use Cases

According to Zhang [80], “the objectives of big data analysis are varied. They are largely aligned with the objectives of big data stakeholders. These can translate into creating values in healthcare, accelerating the pace of scientific discoveries for life and physical sciences, improving the productivity in manufacturing, developing a competitive edge for business, retail, or service industries, and innovating in education, media, transportation, or government.” Business data is analyzed for many purposes for instance a company may perform system log analytics and social media analytics for risk assessment, customer retention, brand management, and so on.

3.4.1 Log Analytics

Logs and trace data generated from IT solutions are referred to as data exhaust and have concentrated value [86]. Big Data helps to store and extract value from these data exhausts. It

helps in identifying previously unreported areas for performance optimisation. Problems may also be discovered from analysing these logs which may prevent a crisis from happening.

3.4.2 Social Media

Insights into social media data can help figure out what customer sentiments are about a product or service. Analysis of customer feedback is very important for a business for decision making and to leverage competitive advantage over its competitors [86].

3.4.3 Healthcare

Useful insights can be derived from Electronic Medical Records including prognosis of disease outbreaks, effectiveness of certain treatment, and trends in infections in areas or age groups. These insights may help the health sector in making decisions that may prevent loss of life in the future by putting preventative measures in place.

3.4.4 Risk Modelling and analysis

Financial institutions benefit from Big Data analysis through risk modelling and analysis [86]. Underutilisation of data inhibits full realisation of value in the data. Risk modelling may help businesses in forecasting financial impact of decisions made which might have adverse effects.

3.4.5 Fraud Detection

Fraud detection using Big Data analytics has applications in varied areas including Healthcare, Financial institutions, Ecommerce, Education etc. There is a high requirement for applying fraud detection on streaming data to reduce latency in the processing as fraud needs to be detected immediately before damage is done.

3.5 Challenges of big data

Big Data has brought numerous challenges to the enterprises handling it including its capturing, storage, sharing, analysis and visualization of the massive data etc. According to the 3Vs model, the challenges of big data management result from the expansion of all three properties, rather than just the volume alone -- the sheer amount of data to be managed [2]. Big data systems are still in their early stages thus are not yet well understood. Owing to this, big data systems are more complicated to deal with compared to previous systems. Several challenges exist in Big Data systems.

3.5.1 Architecture and Infrastructure

The issue of how to handle compatibility of big data and the associated new technologies and techniques with existing legacy systems remains quite a challenge to many organizations. This voluminous amount of data has created an urgent need for organisation to either upgrade or adapt their technologies to growing data demands. The limitations imposed by the inadequacy of the existing systems hinder the ability to fully realise value in the data thus inhibiting the organisation's ability to leverage Big Data.

3.5.2 Storage

Key storage requirements for big data are that it can handle substantial amounts of data, can scale out to keep up with growth, and that it can reliably deliver data to analytics tools that is to provide the necessary processing efficiency required. Increasingly, outsourcing the data to cloud is being adopted as an option for resolving various challenges of Big Data management. However, uploading this large amount of data in cloud will not happen too quickly. Moreover, to extract important insights from the Big data requires collecting all the data and then linking it which may not happen in real time [9].

3.5.3 Big Data Integration

Big Data Integration involves assimilating various types of data from various sources. In the current systems, the proportion of Big Data that is relational (accounting for mainly structured data) is quite limited. The integration of raw, unstructured, schema-less and complex Big Data world of database and semantic web is a big challenge.

3.5.4 Inconsistencies in Big Data Analysis

How to properly handle various types of inconsistencies during data pre-processing and analysis is another challenge [80]. In environments where Big Data are generated, gathered, assimilated, transformed, or represented, inconsistencies in large datasets may arise. This can be due to human factors or applied process. Inconsistencies in Big Data can adversely affect the quality of the outcomes in the analysis process thereof.

3.5.5 Security

The privacy of Big Data is a growing concern. Customer or user Information is collected and used for value addition to the organization without the awareness of the person. Integration of large datasets involving personal information may lead to the inference of new facts about that person that may be confidential and it's possible that these kinds of facts about the person are secretive and the person might not want them to be known [9]. Some Big Data application areas like Healthcare have strict laws governing privacy of data like the HPPA while for some it is less stringent. Disclosure of personal health information of a patient can have permanent effects which cannot be undone. Patients may be stigmatized if their HIV status is involuntarily disclosed. However, all stakeholders have a responsibility to maintain privacy of sensitive data.

3.5.6 Skills and expertise

There is a huge skills gap in Big Data management. This is because the area is still developing thus technical knowhow is limited. Also most organizations take time to adopt new technologies thus taking even longer to upgrade employees to support new technologies.

Universities have started incorporating Big Data in their curriculum. However because of the varied nature of technologies implemented by different organization, specific training is required which does not come cheaply in most cases.

3.5.7 Technical Challenges

Achieving scalability, fault tolerance and high performance are some of the aims for Big Data management [9]. However these properties are not easily achievable. They require massive parallelism not just inter-node but also intra-node. New technologies are required for storage for instance solid state Drives which is a move from traditional Hard Disk Drives. Integrating masses of unstructured, heterogeneous data is another challenge.

3.6 Big Data technologies and implementations

Following the rapid growth of data, traditional data management and business analytics tools and technologies have also become inadequate in handling Big Data due to the strain brought on by the added weight thereof. As a result new approaches are emerging to help organizations gain actionable insights from Big Data. According to [87] “...20th century privacy laws are not keeping up with 21st Century technology when it comes to protecting our most sensitive data”. There are many technologies available now for handling Big Data. These range from storage technologies to manipulation and analysis of the data. The most popular is the Hadoop framework which is a platform for Big Data storage and analysis. NoSQL (Not Only SQL) is another Big Data storage technology for storing and manipulating multi-structured data. Massively parallel analytical databases are another technology more suited for structured data.

3.6.1 Hadoop

The idea that led to the creation of Hadoop was inspired by MapReduce, a user-defined function developed by Google for indexing the Web. Hadoop was then developed by Doug Cutting at Yahoo based on that idea. Hadoop is now a project of the Apache Software Foundation. Hadoop is a highly scalable analytics platform for processing large volumes of

structured and unstructured data [3]. Hadoop technology is not a single entity but rather, it consists of different open source products such as HDFS (storage) and MapReduce (analysis). The open source Hadoop framework is based on Google's Map Reduce software and can process large data sets at a granular level. It offers analytics at a low cost and high speed that some analysts say can't be achieved any other way. A very essential element to the effectiveness of Hadoop is the Hadoop Distributed File System (HDFS), which allows parallel processing by spanning data over different nodes in a single cluster and provides fault tolerance [4].

3.6.2 Hadoop Distributed File System

Hadoop comes with a distributed File System called HDFS (Hadoop Distributed File System). It is a File System designed for storing very large files with streaming data access patterns, running on clusters on commodity hardware. HDFS block size is much larger than that of normal file system i.e. 64 MB by default. The reason for this large size of blocks is to reduce the number of disk seeks. A HDFS cluster has two types of nodes i.e. namenode (master) and number of datanodes (workers). The namenode manages the file system namespace, maintains the file system tree and the metadata for all the files and directories in the tree. The datanode stores and retrieve blocks as per the instructions of clients or the namenode. The data retrieved is reported back to the namenode with lists of blocks that they are storing. The namenode it is very essential for accessing the file thus it is very important to make namenode resilient to failure [9] as it constitutes a single point of failure. Figure 9 gives the architecture of the HDFS.

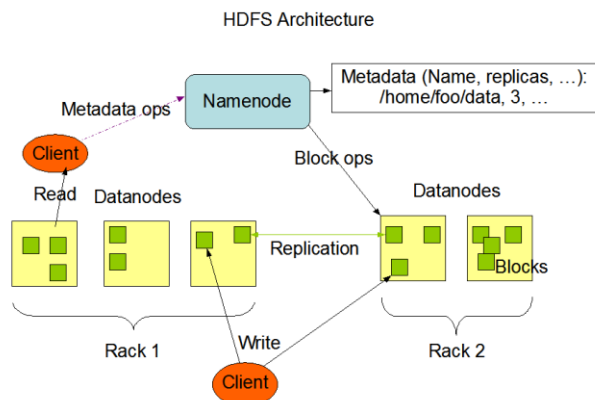


Figure 9: HDFS Architecture [84]

3.6.3 MapReduce

MapReduce is the programming paradigm/model allowing massive scalability and high performance in data processing. The MapReduce basically performs two different tasks i.e. Map Task and Reduce Task [9]. These are two separate but complimentary tasks. A map-reduce computation execution, Map tasks are given input from distributed file system. The map tasks produce a sequence of key-value pairs from the input and this is done according to the code written for map function. These value generated are collected by master controller and are sorted by key and divided among reduce tasks. The sorting basically assures that the same key values ends with the same reduce tasks. The Reduce tasks combine all the values associated with a key working with one key at a time. Again the combination process depends on the code written for reduce job. Different Hadoop components are shown in Table 4 below.

Table 4: Hadoop components

Component	Description
HDFS	A highly fault tolerant distributed file system that is responsible for storing data on the clusters.
MapReduce	A powerful parallel programming technique for distributed processing on clusters
HBase	A scalable, distributed database for random read/write access. It provides database capabilities for Hadoop, allowing you to use it as a source or sink for MapReduce jobs.
Pig	A high level data processing system for analyzing data sets that occur a high level language
Hive	A data warehousing application that provides a SQL like interface and relational model
Sqoop	A project for transferring data between relational databases and Hadoop
Avro	A system of data serialization.
Oozie	A workflow for dependent Hadoop jobs
Chukwa	A Hadoop subproject as data accumulation system for monitoring distributed systems.
Flume	A reliable and distributed streaming log collection.
ZooKeeper	A centralized service for providing distributed synchronization and group services

3.6.4 NoSQL Database

Not Only Structured Query Language (NoSQL) is a class of “schema-less” database management systems that have been designed with more relaxed data models as compared to RDBMS. The term is not meant to discredit SQL but rather to indicate that these databases can provide additional solutions where relational ones fall short. They neither use SQL nor relational data model. These engines usually come with a query language that provides a subset of what SQL can do, plus some additional features [88]. Okman et.al [5] summarized the common features of NoSQL databases as: “ high scalability and reliability, very simple data model, very simple (primitive) query language, lack of mechanism for handling and managing data consistency and integrity constraints maintenance(e.g., foreign keys), and almost no support for security at the database level”.

3.6.4.1 categories of NoSQL databases

There are several categories of NoSQL databases. Four main ones are Key-Value Stores, Column Family, Document Oriented and Graph Databases [38]. These databases generally have lower administration requirements, are cheaper to manage and offer very high performance.

i. Key Value Stores

Provide a way of storing schema-less data by means of a distributed index for object storage. The key (data-type) will be displayed on the left and the corresponding value (actual data) on the right. Key/Value store is best applicable where write performance is of highest priority since its schema-less structure allows for fast storage of data.

ii. Column Oriented Databases

Provide a data store that resembles relational tables but also adds a dynamic number of attributes to the model. They use keys but they point to multiple tables.

iii. Document Oriented Databases

Data is treated as independent objects and their attributes which are stored as separate documents. Each document contains unique information pertaining to a single object. Document stores recognise the structure of the objects stored. Read and writes can be accomplished at once thus making it faster in performance. Schema-less structure gives flexibility in the wake of changing technologies. Documents are described using JSON or XML or derivatives.

iv. Graph Databases

These are databases that are based on the graph theory. Graph databases store data in a graph structure with nodes, edges and properties to represent the data. The nodes represent entities in the database. Edges are connecting lines between two nodes representing their relationships. Properties are the attributes of the entities. Graph databases are more applicable in social networks and intelligent agencies as they efficiently show relationships between entities and provide a way to access data in sites with heavy workloads (predominantly reads). Table 5 below shows a summary of four main categories of NoSQL databases.

Table 5: Categories of NoSQL Databases [38]

	Key Value Stores	Column Databases	Family Document Databases	Graph databases
Based on	Dynamic Hash Tables, Dynamo DB	Google's Bigtable	Lotus Notes, encoding include JSON, XML	Euler's Graph Theory
Data Model	Key/Value pairs	Columns	Key/Value Collections	Graph structure- Nodes, Edges and Properties
Applicability	Handling massive load	Distributed file systems	Web applications, full text searches and updates, information ranking	Semantic web, Social Networks, Intelligent Agencies
Advantages	Simple and easy to implement	Fast querying of data, storage of very large quantities of data	Accepts partially complete data, allows efficient querying	Easy scaling of complex data across distributed systems.
Disadvantages	Inefficient in querying/ updating part of a database	Very low-level API	No standard query language	Traversal of entire graph to give correct results
Examples	Redis, Project Voldermort	Cassandra, HBase	MongoDB, CouchDB	Neo4J, InfoGrid

3.6.4.2 NoSQL database Design Theorems

NoSQL databases make use of CAP (Consistency, Availability and Partition tolerance) and BASE (Basically Available, Soft State, Eventual Consistency) theorems in their design as opposed to the RDBMSs which are based on the ACID (Availability, Consistency, Isolation, Durability). In order to achieve high performance and scalability, NoSQL databases generally trade off consistency and security. Owing to the unstructured nature of the data stored in these databases, security may be difficult to enforce making them more vulnerable to security threats [38].

i. CAP Theorem

CAP theorem was put forward by Eric Brewer. “The theorem implies that within a large-scale distributed data system, there are three requirements that have a relationship of sliding dependency: Consistency, Availability, and Partition Tolerance” [89].

Consistency: Implies that all database clients will read the same value for the same query, even given concurrent updates.

Availability: Implies that all database clients will always be able to read and write data.

Partition Tolerance: Implies that the database can be split into multiple machines; it can continue functioning in the face of network segmentation breaks.

Only two of these three requirements can be achieved at a time. CA means the system primarily supports Consistency and Availability and will block when a network partition occurs. CP means the system primarily supports Consistency and Partition Tolerance but there is still the possibility of some data being unavailable if nodes go down. AP means the system primarily supports Availability and Partition Tolerance, the system may be inconsistent, but the system will always be available, even though operating in degraded mode due to network partition [89].

ii. BASE Theorem

Another alternative to CAP model known as BASE was introduced in line with the move towards NoSQL databases as a way to counter the challenges posed by the ACID model. The acronym BASE represents:

Basically Available: This ensures that the database is accessible even when the system is operating in degraded mode due to failure of a part of the system. This is enabled by partitioning of data across several servers by means of a technique known as sharding. Replication of data across several nodes results in high availability of the data regardless of possible failures.

Soft state: This is the property that means data is constantly changing. It enables transactions to proceed even though updates may take time to propagate to all data stores owing to system disturbances or failure. Inconsistencies are tolerated to a certain extent but the end result will be eventual consistency. A refresh of the data results in its update, otherwise the data becomes stale.

Eventually consistent: BASE relaxes the requirement of strict consistency at the end of every operation and only guarantees that the data stores will come to a consistent state at a later stage. Thus at any given time all entities may not necessarily have the same view of the data.

3.7 Big Data Security Issues

We review security built-into the Big Data environment including NoSQL databases and evaluate the weaknesses of these systems. Our goal is to uncover security problems inherent in the Big Data environments. The amount, diversity and rate of data being generated for processing and storage results in sheer masses of data that need to be safely secured. Big Data, in the hands of organizations is highly valuable, and subject to privacy laws and compliance regulations, and must be protected. It can be established that the various significant ways in which these database systems are deployed have impact on the security of the Big Data environment. The following are some security issues associated with Big Data.

3.7.1 Threats posed by Distributed nature of the Big Data environments

Nodes within the Big Data environment are distributed subsequently resulting in massive parallel computation [56]. This creates increased attack surface across several distributed nodes which makes it very complex to secure the Big Data environment. Another issue is deciding where to grant database system Access, whether at the Clients home locations or at the remote location which increases the probability of unauthorized access. Enforcing privacy of data across these distributed nodes is a challenge since encryption and decryption may bring a latency that may create a performance bottleneck.

3.7.2 Safeguarding Integrity of data

The protection of integrity is much harder in Big Data environments because of its heterogeneous nature than in homogeneous environments. There is absence of central control and its schema-less nature makes it difficult to enforce integrity constraints.

3.7.3 Communication between nodes

All communication protocols as nodes interact rely on RPC over TCP/IP. A Remote Procedure Call (RPC) abstraction incorporates both the Client Protocol and the DataNode Protocol in the distributed Big Data environments. Big Data environments with RPC ports exposed to the Distributed environment are especially vulnerable. Security concerns emanate as nodes interact through message passing, because communication is not secure.

3.7.4 Fragmentation and sharing of data

NoSQL databases horizontally segments slices of data and share them across multiple servers in a process called sharding. Data from a variety of nodes move to and from in the Big Data environment which is distributed across multiple servers. Movement of this data to multiple locations is automated for large inter/intra-clustering employing MapReduce parallel copying mechanism in copying portions of the source data into the destination file system. The

maintenance of replicated shards of data that includes passwords is computationally expensive, more prone to error and increases the risk of theft. This model is not centralized which poses difficulties in securing data as it gets replicated and moves in many places as needed.

3.7.5 Lack of central management security

Clients accessing NoSQL databases are in contact with resource managers and various nodes directly. In situations where malicious data gets propagated from a single compromised location, the entire system is compromised. Protecting nodes, name servers and those clients becomes difficult especially when there is no central management security point.

3.7.6 Encryption of data

Most NoSQL databases are found wanting when it comes to protecting data in storage, only a few categories of NoSQL databases provide mechanisms to protect data at rest by employing encryption techniques. Encryption is widely regarded as the de-facto standard for safeguarding data in storage. Malicious intruders who intend to then steal from archives or with intention to read directly from the disk will find the data unintelligible. Encrypted data will be accessed by users with decryption keys, but however most industry solutions offering encryption services lack horizontal scaling and transparency required in the Big Data environment.

3.7.7 Enforcing access control

The NoSQL database's schema-less structure makes Role-based access control difficult to enforce. We take for instance the Key-Value store that store data by means of a distributed index for object storage. In this type of database different data are stored in one huge database. This becomes a challenge as heterogeneous data is stored together in one database as opposed to relational models which conform to defined schemas and tables that store only related data. This leaves NoSQL databases dependent on 3rd party solutions or application middleware to provide better access control.

3.7.8 Firewall Breaches

Firewalls cannot fully protect data at rest or in-transit within the Big Data environment [56]. If a firewall gets breached, the database is immediately exposed to attacks. Firewall breaches emanating from the firewall perimeter cannot be avoided like attackers who get into data centers physically or electronically can get access to data.

3.7.9 Authenticating Clients

Kerberos can be used to authenticate clients, Data Node, Name Node in the Big Data environment. Malicious Clients and Nodes can gain unauthorized access to the Big Data environments upon stealing or duplicating the Kerberos ticket. These credentials can be obtained from system snapshots as well as virtual images. The situation has worsened in this Big Data environment where exact copies, clones and imposter nodes can be used to generate malicious services into the databases environment.

3.7.10 Audit and logging

Audits and logs are performed to aid in discovery of malicious activities in the database system. However without actually looking at the data and developing policies to detect malicious activities, logging is not useful. Also the frequency at which the Audits are carried out can have impact on their effectiveness. If audits are performed say quarterly that means malicious activities can occur which can result in serious problems for the organization. This may be discovered too late when the damage has already occurred.

3.7.11 Monitoring, input validation, and blocking

Big data collects data from many different sources. Existing Big Data monitoring tools lack the capability to identify malicious queries, misuse activities and subsequently block them. Monitoring undertaken by several tools in the Big Data environment mostly perform their task at the API. There is an assumption that all access by client connections will pass through the same

path that authenticate clients through Kerberos, which results in a performance constrains. Also advanced threats may bypass the central Kerberos authentication.

3.7.12 API security

APIs can be subjected to several attacks such as Code injection, buffer over flows, command injection as they access the NoSQL databases [56]. The APIs for big data clusters need to be protected from code and command injection, buffer overflow attacks, and every other web services attack. This responsibility often left to the application that uses the cluster which creates problems.

3.7.13 Inference problem

Big data management usually involves applying data mining and analytics. This brings many security concerns related to sharing big data analytics as there is risk of loss of privacy and confidentiality of data. If there are no proper control this may create an inference problem where despite de-identification of data, some identities may still be deduced from released analytics data.

3.8 Advantages of NoSQL databases

NoSQL database systems have addressed scaling and performance challenges inherent in traditional RDBMS by exploiting partitions, relaxing heavy strict consistency protocols and by way of distributed systems that can span data centers while handling failure scenarios without a hitch [38]. Making the right choice of a NoSQL database for a specific application can be a difficult task. However, some important factors to consider are as follows:

i. Scalability

Adopting the sharding technique can be useful in achieving scale regardless of the database technology in use. Sharding employs horizontal partitioning which is a database concept in which rows of a database table are stored in separate locations. Quick seamless scaling at any time has become a determinant factor in Web traffic that has on and off surges. Resource contention between servers like disk, memory and CPU is removed. Intelligent parallel processing and maximization of CPU/Memory per database instance can be done.

ii. Performance

In web applications latencies are required to be very low. Read and write latencies of NoSQL databases are very low. Performance is achieved by sharding, replication and parallel processing of data.

iii. Availability

Availability of a NoSQL database is of paramount importance. Nature of associated applications does not go with disruption of service even for a short time. The system is required to be always available irregardless of a network partition.. If your application is down, you are simply losing money. The database system should generally allow for online maintenance and upgrades without affecting availability of the system.

iv. Ease of development

Schema-less nature of NoSQL databases makes it possible to make changes to the application without altering the database and vice-versa which cannot be done in RDBMS.

3.9 Disadvantages of NoSQL database technologies.

- i. Challenges of NoSQL databases include lack of standardized models and expertise for efficient database support.
- ii. There is unavailability of common tools and techniques. Query languages are varied.
- iii. The majority of the solutions for NoSQL use command-line interface or simple shell tools which most users are not comfortable with especially those more familiar with graphic user interfaces common to RDBMS.
- iv. Dependent on third parties for security, analytics and other data management requirements.

3.10 Conclusion

From our research on Big Data, we have identified that there are numerous vulnerabilities which are brought forward by the characteristics and distributed nature of the Big Data environments. Our interest in exploring security of Big Data environments emanates from identifying security as one ominous challenge in Big Data. As alluded before, security should be applied as near to the data source as possible. Thus in this Thesis, we concentrate on databases in the Big Data environments (NoSQL databases). Unless the data stored in the Big Data databases is secured, then no matter how we try to secure the front end loopholes still exist which may lead to security breaches.

Chapter Four: Framework for Security Requirements Engineering For Big Data

In this chapter we propose a framework for Security Requirements Engineering for Big Data stores. We elicit the security requirements from vulnerabilities inherent in the Big Data stores based on generic operations (Create, Read, Update, and Delete) performed on the database. Our proposed framework first identifies generic actors, database operations, and then vulnerabilities with related threats by drawing sequence diagrams for operations, security requirements are elicited to mitigate all the vulnerabilities and threats, analysis, prioritization and specification steps are carried out. If the database is secure then fewer resources will be invested in securing the application making application development more cost effective for the organisations.

4.1 Overview of the proposed framework

Our proposed framework is aimed to be a generic model that can be applied to development of any Big Data database. In this light, activities involved are Security Requirements Elicitation, Security Requirements Analysis, Security Requirements Prioritization and Security Requirements specification. Elicitation activities start with identification of generic actors for database. Next the generic CRUD database operations are used. Sequence diagrams are drawn for each actor interaction with the database. Vulnerabilities inherent in points of interactions are identified. These are related to threats from a vulnerability and threat database we maintain. Security requirements are elicited from the Vulnerability and Security requirements mapping derived from our repository. After Security Requirements Elicitation we analyse the elicited security requirements by grouping and checking for completeness and consistency. Prioritisation involves mapping vulnerability to threats then calculating risk. Security requirements are then prioritized based on the risk levels. Lastly, we specify the security requirements. Our framework is shown in Figure 10 below.

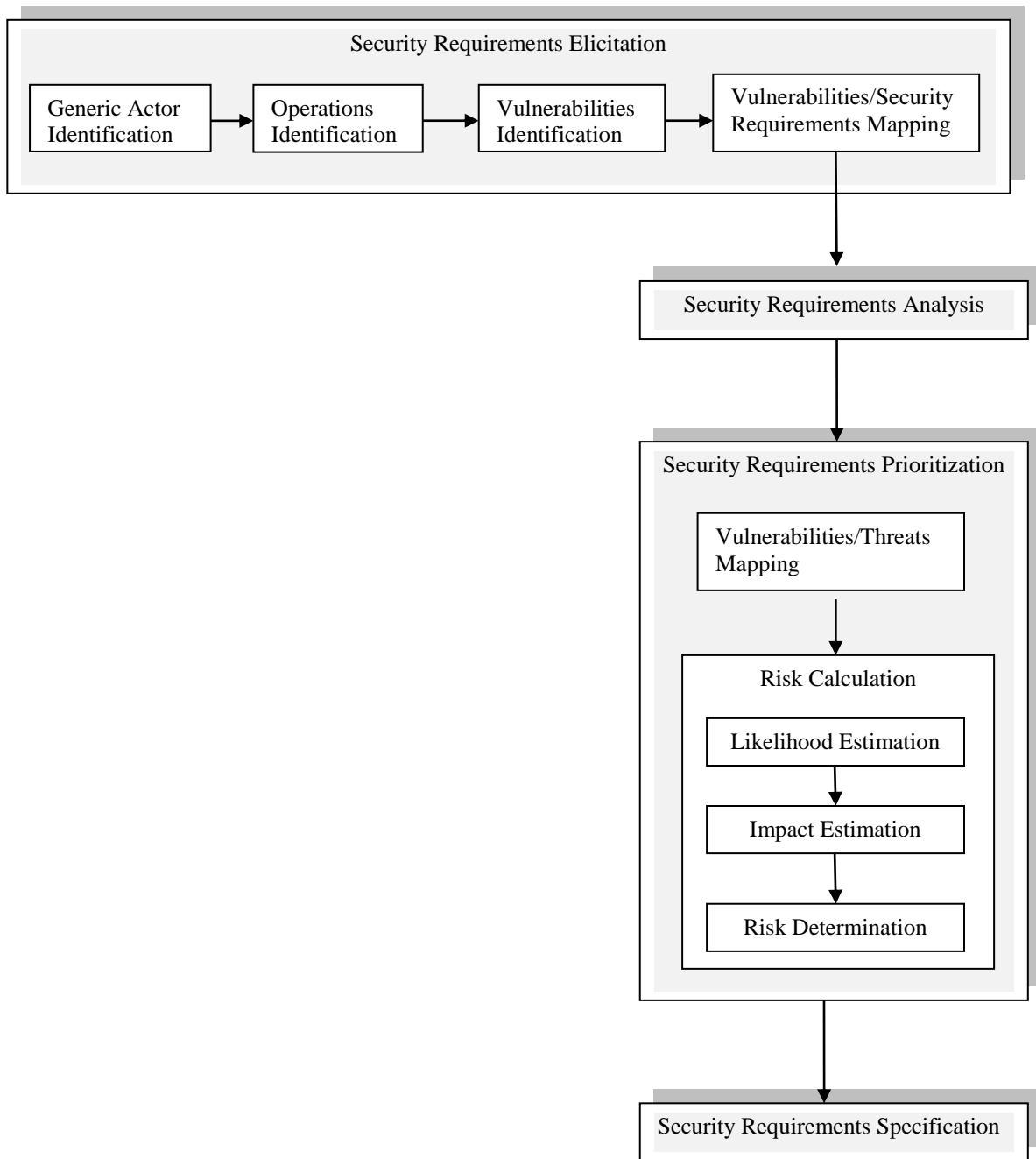


Figure 10: Proposed Framework for Security Requirements Engineering for Big Data

4.2 Security Requirements Elicitation

The security requirements elicitation encompasses generic actor identification, operations identification, sequence diagrams for database operations are used in the identification of vulnerabilities, and vulnerabilities/security requirements mapping is done. This step consists of the following sub steps:

A. Generic Actor Identification

Actors are those which interact with the system such as user, administrator, management and so on. User can be human, cooperative (such as a DBMS) or autonomous actor (such as standalone computational software) [90]. We make use of generic actors as our focus is only on database security. Only two generic actors User (human) and Database are considered for illustration in this thesis.

B. Operations Identification

There are four basic functions of persistent storage which are referred by the acronym CRUD standing for Create, Read, Update and Delete in big databases. These are also known as database operations [91]. These operations are used for manipulating data in the databases. Depending upon the type of database in use, these operations may be applied using different queries. They are defined in Table 6 below:

Table 6: Generic CRUD Operations

Operations	Description
Create	Adding new entries to the database
Read	Retrieve, search, or view existing entries without changing the data
Update	Edit or modify existing entries (changes the data values by insertion, deletion or update)
Delete	Remove or deactivate existing entries

C. Vulnerability identification

Vulnerability is a flaw or weakness in the system environment [97] that a malicious attacker could exploit to cause damage to the system. Vulnerabilities could exist in various parts of the system environment including system design, business operations, installed software, and network configurations. It is the vulnerability that enables a threat to be exercised within the system. Hence vulnerabilities should be identified together with the threats. In a Big Data environment vulnerabilities arise due to complexity brought on by the type and distributed nature of data involved. All vulnerabilities are depicted with a prefix V. For Big Data stores, vulnerabilities are identified by drawing and analysing the sequence diagram for weak points where threats can occur. Sequence diagrams are shown in Figure 11-14.

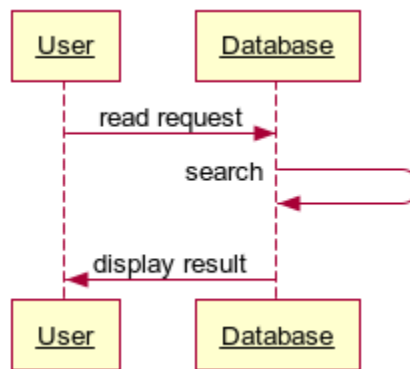


Figure 11: READ Sequence Diagram

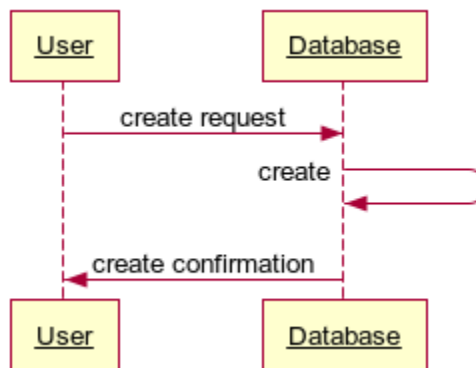


Figure 12: CREATE Sequence Diagram

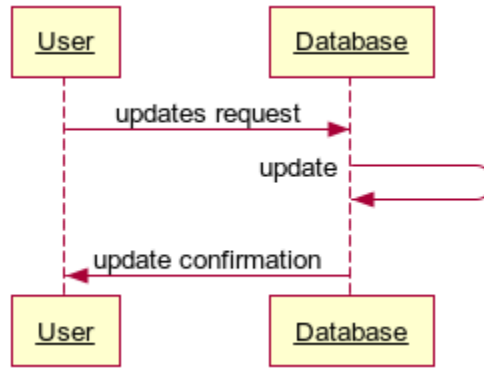


Figure 13: UPDATE Sequence Diagram

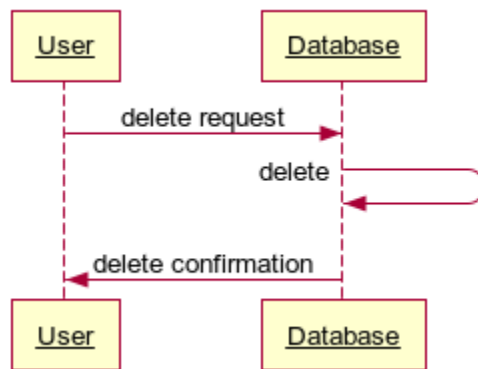


Figure 14: DELETE Sequence Diagram

Table 7 below shows the vulnerabilities for CRUD Operations for a generic human user

Table 7: Extraction of vulnerabilities for CRUD Operations for Human User			
Operation	Actors	Interaction	Vulnerability
CREATE	User-> Database	Create Request	1. V.Weak_Access_Control 2. V.Untrained_Users 3. V.Unencrypted_Data 4. V.Unsecured_Network 5. V.Monitoring_Absence 6. V.Network_Partition 7. V.Breached_Firewall 8. V.Inadequate_Logging
	Database->Database	Create	1. V.Unencrypted_Data 2. V.Breached_Firewall 3. V.Monitoring_Absence 4. V.Physical_Security 5. V.Misconfigurations 6. V.Unsecured_API
	Database-> User	Return Confirmation	1. V.Network_Partition
READ	User -	Read Request	1. V.Weak_Access_Control

	>Database		<ol style="list-style-type: none"> 2. V.Untrained_Users 3. V.Monitoring_Absence 4. V.Network_Partition 5. V.Inadequate_Logging
	Database->>Database	Search	<ol style="list-style-type: none"> 1. V.Unsecured_API 2. V.Unencrypted_Data 3. V.Misconfigurations 4. V.Breached_Firewall 5. V.Monitoring_Absence 6. V.Physical_Security
	Database->User	Display result	<ol style="list-style-type: none"> 1. V.Unencrypted_Data 2. V.Unsecured_Network 3. V.Monitoring_Absence 4. V.Network_Partition 5. V.Untrained_Users 6. V.Inadequate_Logging
Update	User ->Database	Update Request	<ol style="list-style-type: none"> 1. V.Weak_Access_Control 2. V.Untrained_Users 3. V.Unencrypted_Data 4. V.Unsecured_Network 5. V.Monitoring_Absence 6. V.Network_Partition 7. V.Breached_Firewall 8. V.Inadequate_Logging 7. V.Untrained_Users
	Database->>Database	Update	<ol style="list-style-type: none"> 1. V.Unencrypted_Data 2. V.Breached_Firewall 3. V.Monitoring_Absence 4. V.Physical_Security 5. V.Misconfigurations 6. V.Unsecured_API 7. V.Obsolete_System
	Database->User:	Return Confirmation	<ol style="list-style-type: none"> 1. V.Network_Partition
Delete	User ->Database	Delete Request	<ol style="list-style-type: none"> 1. V.Weak_Access_Control 2. V.Untrained_Users 3. V.Unsecured_Network 4. V.Monitoring_Absence 5. V.Network_Partition 6. V.Inadequate_Logging
	Database->>Database	Delete Data	<ol style="list-style-type: none"> 1. V.Monitoring_Absence 2. V.Physical_Security 3. V.Misconfigurations 4. V.Inadequate_Logging
	Database->User	Return Confirmation	<ol style="list-style-type: none"> 1. V.Network_Partition

D. Mapping of Vulnerabilities to Security Requirements

After vulnerabilities are identified using sequence diagrams, security requirements as defined in [16] are elicited to mitigate the vulnerabilities according to the criteria given below:

i. Identification, authentication, authorisation requirements

These requirements work together to protect the database by enforcing access control [97]. All database users should be identified, authenticated and authorised to access some database resource [100]. Vulnerabilities such as Weak access control and untrained users may lead to database compromise. Furthermore, if users are not properly trained to be aware of database threats, consequences of negligence and the need to safeguard their credentials, It may lead to potential attack. Threats associated with these requirements are impersonate, insider, social engineer, credential theft, phishing, spoofing and data theft [102]. These threats are associated with managing to acquire legitimate credentials leading to compromise of the database thus the need for the three security requirements.

ii. Immunity Requirements

Immunity requirement provide an internal ability to defend against corruption and attack. Vulnerabilities associated with immunity requirements include firewall breach, unvalidated input, unsecured API [98] and unsecured network. These pose threats such as malware, insider, outsider, technical failure and injection attacks. Immunity is required to safeguard the database against compromise arising from these vulnerabilities and threats.

iii. Integrity Requirements

Integrity requirement ensure that data is protected from unauthorized modification [97]. Unauthorized modification can occur due to the following vulnerabilities: breached firewall, untrained users, unencrypted data, unsecured API and unsecured network. These vulnerabilities may lead to insider, outsider and change data threats.

iv. Intrusion Detection Requirements

Intrusion detection requirement provide internal ability to monitor and identify attempts to gain unauthorized access to the database. Vulnerabilities such as breached firewall, unsecured API and unsecured network may lead to compromise of the database in the absence of mechanism for monitoring, intrusions. These intrusions, if not detected may allow threats such as malware [102], data theft, deny service, credential theft, spoofing, insider and outsider to be exercised on the system.

v. Non-repudiation Requirements

Non-repudiation requirement enables keeping tamper-proof records of all database activities to prevent future denial of access to it by any user or process. Absence of adequate logs may lead to violation of non-repudiation [98] leading to threats such as repudiate receive and repudiate send.

vi. Privacy Requirements

Privacy requirement ensures personal control over data stored in the database. Vulnerabilities including untrained user , unencrypted data, firewall breach, unsecured API and unsecured network. All these weaknesses may lead to unauthorised disclosure of confidential data [97]. Disclose data, privacy violated, eavesdropping, outsider, insider and data theft [102] are all threats related to violation of privacy requirements.

vii. Security Auditing Requirements

Security auditing requirement ensures monitoring of database system for violations of security policy, malicious activities and recording there-of. Vulnerabilities including-misconfigurations, breached firewall, unsecured API, lack of input validation, monitoring absence and inadequate logging may be revealed by security auditing [98, 102]. Threats such as repudiation, insider, credential theft, fraud, injection attack, deny service and data theft can result from absence of mechanisms for ensuring security auditing.

viii. Survivability Requirements

Survivability requirement ensures that the database system makes a tradeoff between integrity and availability in the face of database attack. Survivability of a database system cushions it from impact of a network partition. Absence of mechanisms for ensuring survivability of database system leads to database unavailability [97], technical failure and hardware failure.

ix. Physical Protection Requirements

Physical protection requirement ensures facility protection [97] from physical conditions and procedures that could cause serious losses or damage to an organization. Physical security is required to be in place for the database system to be protected from vandalism and outsider threats.

x. System Maintenance Security Requirements

A system maintenance security requirement ensures prevention of any system modification be it authorized or not from disrupting its deployed security mechanism [101]. Vulnerabilities include misconfigurations [102], obsolete system and breached firewall. Associated threats include outsider, technical failure, and hardware failure result from absence of system maintenance.

A correlation matrix is formulated from studies [39] on Big Data for security requirements elicitation. Table 8 shows the vulnerabilities and Security Requirements Correlation Matrix as explained above.

Table 8: Vulnerabilities and Security Requirements Correlation Matrix

Security Requirement \ Vulnerabilities	Identification	Authentication	Authorization	Immunity	Integrity	Intrusion Detection	Non-reputationation	Privacy	Security Auditing	Survivability	Physical Protection	System Maintenance
V.Misconfigurations									X			X
V.Weak_Access_Control	X	X	X		X							
V.Unencrypted_Data					X			X				
V.Breached_Firewall				X	X	X		X	X			X
V.Unsecured_API				X	X	X		X	X			
V.Unsecured_Network				X	X	X		X				
V.Network_Partition										X		
V.Unvalidated_Input				X					X			
V.Untrained_Users	X	X	X		X			X				
V.Monitoring_Absence						X			X			
V.Inadequate_Logging							X		X			
V.Physical_Security											X	
V.Obsolete_System												X

E. Security requirements for Big Data

The elicited security requirements for CRUD operation as applied to big data such as MongoDB and Cassandra databases are shown in Table 9 below. Once security requirements elicitation process is done it will help in analysis, prioritization and specification phases of the framework.

Table 9: Security Requirements elicited based on CRUD Operations

Operations	Description	Vulnerable interaction Sequences	Vulnerabilities	Security Requirements
Create	Create operations are those that add new records	User-> Database Create Request	1. V.Weak_Access_Control 2. V.Untrained_Users 3. V.Unencrypted_Data 4. V.Unsecured_Network 5. V.Monitoring_Absence 6. V.Network_Partition 7. V.Breached_Firewall 8. V.Inadequate_Logging	1. Identification 2. Authentication 3. Authorization 4. Integrity 5. Privacy 6. Immunity 7. Intrusion Detection 8. Security Auditing 9. System Maintenance 10. Non-reputationation
		Database->Database: Create	1. V.Unencrypted_Data 2. V.Breached_Firewall 3. V.Monitoring_Absence 4. V.Physical_Security 5. V.Misconfigurations 6. V.Unsecured_API	1. Security Auditing 2. Intrusion Detection 3. Integrity 4. Privacy 5. Immunity 6. System Maintenance 7. Physical Protection
		Database-> User: Return Confirmation	1. V.Network_Partition	1. Survivability
Read	Read operations are those that	User-> Database Create Request	1.V.Monitoring_Absence	1.Security Auditing 2.Intrusion Detection 5.Identification 6.Authentication

	retrieve records.		2.V.Network_Partition 3.V.Inadequate_Logging 4.V.Weak_Access_Control 5.V.Untrained_Users	3.Survivability 4.Non-repudiation	7.Authorisation
		Database->Database: Search	1. V.Unencrypted_Data 2. V.Misconfigurations 3. V.Breached_Firewall 4. V.Monitoring_Absence 5. V.Physical_Security 6. V.Unsecured_API	1. System Maintenance 2. Security Auditing 3. Intrusion Detection 4. Integrity	5. Privacy 6. Immunity 7. Physical Protection
		Database-> User: Return Result	1. V.Unencrypted_Data 2. V.Unsecured_Network 3. V.Monitoring_Absence 4. V.Network_Partition 5. V.Inadequate_Logging 6. V.Untrained_Users	1. Integrity 2. Privacy 3. Immunity 4. Intrusion Detection	5. Security Auditing 6. Survivability 7. Non-repudiation
Update	Update operations are those that modify/edit existing records	User->Database: Update Request	1. V.Unencrypted_Data 2. V.Unsecured_Network 3. V.Monitoring_Absence 4. V.Network_Partition 5. V.Breached_Firewall 6. V.Inadequate_Logging 7. V.Untrained_Users 8. V.Weak_Access_Control	1. Integrity 2. Privacy 3. Immunity 4. Intrusion Detection 5. Security Auditing	6. Survivability 7. Identification 8. Authentication 9. Authorization 10. System Maintenance
		Database->Database: Update	1. V.Unencrypted_Data 2. V.Breached_Firewall 3. V.Monitoring_Absence 4. V.Physical_Security 5. V.Misconfigurations 6. V.Unsecured_API 7. V.Obsolete_System	1. Integrity 2. Privacy 3. Intrusion Detection 4. Immunity	5. Security Auditing 6. Survivability 7. Physical Protection 8. System Maintenance
		Database->User: Return Confirmation	1. V.Network_Partition	1. Survivability	
Delete	Delete operations are those that remove or deactivate existing entries	User->Database: Delete Request	1. V.Unsecured_Network 2. V.Monitoring_Absence 3. V.Network_Partition 4. V.Inadequate_Logging 5.V.Weak_Access_Control 6.V.Untrained_Users	1. Immunity 2. Integrity 3. Privacy 4. Intrusion Detection 5. Security Auditing	6. Survivability 7. Non-repudiation 8. Identification 9. Authentication 10.Authorisation
		Database->Database: Delete Data	1. V.Monitoring_Absence 2. V.Physical_Security 3. V.Misconfigurations 4. V.Inadequate_Logging	1. Intrusion Detection 2. Security Auditing 3. Physical Protection 4. Intrusion Detection	5. Security Auditing 6. System Maintenance 7. Non-repudiation 8. Security Auditing
		Database->User: Return Confirmation	1. V.Network_Partition	1. Survivability	

4.3 Security Requirements Analysis

Once security requirements have been elicited, next stage is analysis of security requirements. Security Requirements Analysis involves grouping and checking the identified security requirements for consistency and completeness. If any conflicts are present, they ought to be resolved to reach an agreement to avoid any further conflicts.

i. Grouping

Grouping means combining vulnerabilities that are mitigated by the same security requirements as shown in Table 10.

Table 10: Grouped vulnerabilities and security requirements

Security Requirements	Vulnerabilities
Identification	V.Weak_Access_Control V.Untrained_Users
Authentication	V.Weak_Access_Control V.Untrained_Users
Authorization	V.Weak_Access_Control V.Untrained_Users
Immunity	V.Unsecured_API V.Unvalidated_Input V.Unsecured_Network V.Breached_Firewall
Integrity	V.Weak_Access_Control V.Unencrypted_Data V.Breached_Firewall V.Unsecured_Network V.Untrained_Users V.Inadequate_Logging
Intrusion Detection	V.Breached_Firewall V.Monitoring_Absence V.Unsecured_Network V.Unsecured_API
Non-repudiation	V.Inadequate_Logging
Privacy	V.Unencrypted_Data V.Breached_Firewall V.Untrained_Users V.Unsecured_Network V.Unsecured_API
Security Auditing	V.Monitoring_Absence V.Inadequate_Logging V.Misconfigurations V.Breached_Firewall V.Unsecured_API V.Unvalidated_Input
Survivability	V.Network_Partition
Physical Protection	V.Physical_Security
System Maintenance	V.Obsolete_System V.Misconfigurations V.Breached_Firewall

ii. Completeness

Completeness means all services required by the user should be defined. “The notion of ‘completeness’ in requirements definition is problematic. There is no simple analytical procedure for determining when the users have told the developers everything that they need to know in order to produce the system required.” [62].

iii. Consistency

Consistency means requirements should not have contradictory definitions. A state of a specification that is inconsistent means there are parts of it that conflict. Inconsistency may be defined in terms of relationships that should hold which may refer to syntactic, semantic aspects of the specification as well as process relationships [57].

4.4 Security Requirements Prioritization

Analysed security requirements are prioritised so that depending on resources available, high priority requirements get implemented first. We adopt risk-based security requirements prioritisation. A risk-based framework for Security Requirements Engineering is proposed in [54]. They defined risk as the combination of the probability of occurrence of harm and its severity. In [61], IT risk is given by the formula:

$$Risk = Threat * Vulnerability * Impact.$$

“In this way, risk is characterized by the opportunity of exploiting one or multiple vulnerabilities, from one or many entities, by a threatening element using an attack, causing an impact on business assets” [61]. A related model by OWASP [93] provides basic customizable risk rating methodology. They apply:

$$Risk = Likelihood * Impact.$$

We adopt and redefine the steps of the OWASP model which is customisable to suit our framework. Below are the steps we follow in prioritising security requirements. In this section we will briefly describe threats, mapping of threats to vulnerabilities and steps to prioritise the security requirements

A. Threat

Vulnerability leads to threats. According to [11], “A threat is the potential for a particular threat-source to successfully exercise a particular vulnerability”. [55] Defines a threat as any “circumstance or event with the potential to adversely impact organizational operations, assets, or individuals through an information system via unauthorized access, destruction, disclosure, modification of information, and/or denial of service”. Threat sources include human factors, natural disasters, and infrastructure failures [38]. Threats can be intentional or unintentional and consist and may lead to potential loss [92]. Different types of threats described in Common Criteria [15] and additional threats are depicted in Table 9 with a prefix T. They can be mitigated using the security requirements. For example:

- i. T.Change_Data means unauthorized modification of data.
- ii. T.Spoofing means an unauthorized user accesses resources by masquerading as a legitimate user through forgery of data.

T.Change_Data is mitigated using Integrity security requirement. T.Spoofing can be mitigated using Identification, Authentication, Authorisation and Intrusion Detection security requirements. Table 11 shows threats and Security Requirements Correlation Matrix also as explained above.

Table 11: Threats and Security Requirements Correlation Matrix

Security Requirement Threats	Identification	Authentication	Authorization	Immunity	Integrity	Intrusion Detection	Non-repudiation	Privacy	Security Auditing	Survivability	Physical Protection	System Maintenance
T.Change_Data					X							
T.Data_Theft	X	X	X			X		X	X			
T.Deny_Service						X			X			
T.Disclose_Data								X				
T.Impersonate	X	X	X									
T.Injection_Attack				X					X			
T.Fraud			X						X			
T.Privacy_Violated								X				
T.Eavesdropping								X				
T.Credential_Theft	X	X	X			X			X			
T.Social_Engineer			X									
T.Phishing	X	X	X									
T.Spoofing	X	X	X			X						
T.Repudiate_Receive							X		X			
T.Repudiate_Send							X		X			
T.Insider			X	X	X	X		X	X			
T.Outsider				X	X	X		X			X	X
T.Technical_failure				X						X		X
T.Hardware_Failure										X		X
T.Vandalism											X	
T.Malware				X		X						
T.Unavailability									X	X		

B. Vulnerabilities/Threats Mapping

We have maintained a threat and vulnerability database from where mapping of various vulnerabilities to threats is done as justified under step 1 above. Threat and Vulnerability list for Big Data Environments is shown in Table 12 below. Mapping of vulnerability and corresponding threats is shown in Figure 15.

Table 12: Vulnerability and Threat List for Big Data Environments

Vulnerability List		Threat List	
V.Misconfigurations	V.Physical_Security	T.Change_Data	T.Social_Engineer
V.Weak_Access_Control	V.Obsolete_System	T.Data_Theft	T.Phishing
V.Unencrypted_Data		T.Deny_Service	T.Spoofing
V.Breached_Firewall		T.Disclose_Data	T.Repudiate_Receive
V.Unsecured_API		T.Impersonate	T.Repudiate_Send
V.Unsecured_Network		T.Injection_Attack	T.Insider
V.Network_Partition		T.Fraud	T.Outsider
V.Unvalidated_Input		T.Flooding	T.Technical_failure
V.Untrained_Users		T.Privacy_Violated	T.Hardware_Failure
V.Monitoring_Absence		T.Eavesdropping	T.Malware
V.Inadequate_Logging		T.Credential_Theft	T.Vandalism
		T.Brute_Force	T.Unavailability

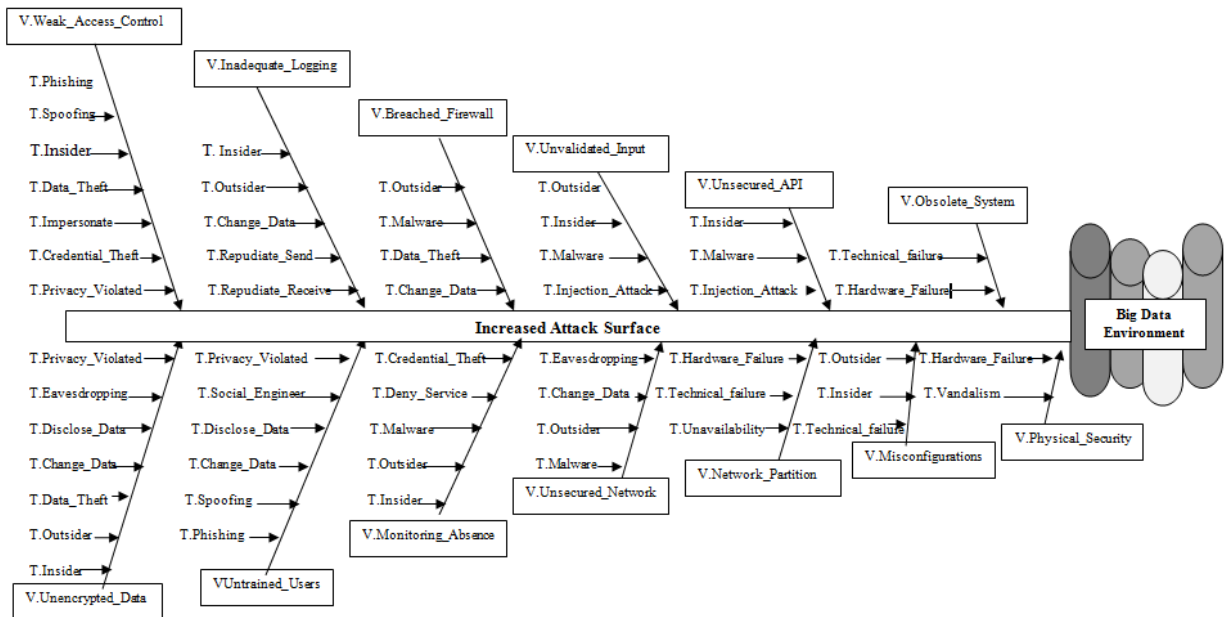


Figure 15: Vulnerabilities and Threats for Big Data Environment

C. Steps for security requirements prioritization

We use the formula given in the OWASP model for risk calculation:

$$Risk = Likelihood * Impact.$$

Steps involved in the risk calculation are given below.

i. Likelihood Estimation

Likelihood is a rough measure of how likely a particular vulnerability is to be uncovered and exploited by an attacker (threat). Likelihood ratings are estimated for vulnerabilities and threats based on degree of satisfaction of vulnerability and threat factors. This makes the process more formal and repeatable. The accuracy of the estimation will depend on the experience and knowledge of the team in charge of the project. Threat factors are evaluated according to skill level, motive, opportunity and size while likelihood is evaluated on ease of discovery, ease of exploitation, awareness and intrusion detection, each having an estimated score in the range 0 to 9. We then average the scores to get one value for likelihood. Using our vulnerability/threat mapping, we input the average scores as shown in Table 13.

Table 9: Likelihood Estimation

Vulnerability Threats	V.Misconfigurations	V.Weak_Access_Control	V.Unencrypted_Data	V.Breached_Firewall	V.Unsecured_API	V.Unsecured_Network	V.Network_Partition	V.Invalidated_Input	V.Untrained_Users	V.Monitoring_Absence	V.Inadequate_Logging	V.Physical_Security	V.Obsolete_System
	T.Change_Data		8	8	3		3			6		2	
T.Data_Theft		6	7	4									
T.Deny_Service										4			
T.Disclose_Data			8						2				
T.Impersonate		8											
T.Injection_Attack					2			2					
T.Fraud		5									6		
T.Privacy_Violated		9	9						2				
T.Eavesdropping			6			3							
T.Credential_Theft		8								3			
T.Social_Engineer									2				
T.Phishing		2							2	2			
T.Spoofing		2							2	2			
T.Repudiate_Receive											1		
T.Repudiate_Send											1		
T.Insider	2	8	3		3			2		3	7	3	
T.Outsider	4		7	5		7		3		5	1	1	
T.Technical_failure	1						2						2
T.Hardware_Failure							3					1	2
T.Vandalism												2	
T.Malware				3	4	4		3		8			
T.Unavailability							7						

ii. Impact Estimation

Impact refers to consequences of a successful exploit. We estimate impact ratings for threats and vulnerabilities. The factors considered are technical impact and business impact. Technical factors include loss of confidentiality, integrity, availability and accountability. Business impact includes financial damage, reputation damage, non-compliance and privacy violation. Next, we calculate the average impact and indicate the scores as shown in Table 14.

Table 10: Impact Estimation

Vulnerability Threats	V.Misconfigurations	V.Weak_Access_Control	V.Unencrypted_Data	V.Breached_Firewall	V.Unsecured_API	V.Unsecured_Network	V.Network_Partition	V.Unvalidated_Input	V.Untrained_Users	V.Monitoring_Absence	V.Inadequate_Logging	V.Physical_Security	V.Obsolete_System
T.Change_Data		9	9	9		9			7		7		
T.Data_Theft		8	8	8									
T.Deny_Service										8			
T.Disclose_Data			7						8				
T.Impersonate		8											
T.Injection_Attack					7			7					
T.Fraud		8									7		
T.Privacy_Violated		7	7						7				
T.Eavesdropping			6			7							
T.Credential_Theft		9								9			
T.Social_Engineer									8				
T.Phishing		7							9	9			
T.Spoofing		7							9	9			
T.Repudiate_Receive											3		
T.Repudiate_Send											2		
T.Insider	7	9	7		7			6		7	7	7	
T.Outsider	9		9	8		9		8		9	8	9	
T.Technical_failure	9						7						3
T.Hardware_Failure							5					4	2
T.Vandalism											5		
T.Malware				5	5	5		5		5			
T.Unavailability							8						

iii. Risk Determination

We calculate average scores for both likelihood and impact and assign levels Low, Medium or High on a scale of 0 to <3, 3 to <6, 6 to 9 respectively as shown in Table 15. We use the given

Severity Table 16 to determine risk for each vulnerability according to likelihood and impact ratings as shown in Table 17.

Table 11: Likelihood and Impact Levels

Likelihood and Impact Levels	
0 to <3	LOW
3 to < 6	MEDIUM
6 to 9	HIGH

Table 12: Overall Risk Severity

Overall Risk Severity				
Impact	HIGH	Medium	High	High
	MEDIUM	Low	Medium	High
	LOW	Low	Low	Medium
		LOW	MEDIUM	HIGH
	Likelihood			

Table 13: Risk Calculation

Vulnerabilities	Likelihood	Impact	Risk
V.Misconfigurations	$(2+4+1)/3 = 2.33 = \text{Low}$	$7+9+9 = 8.33 = \text{High}$	Medium
V.Weak_Access_Control	$(8+6+8+5+9+8+2+2+8)/9 = 6.22 = \text{High}$	$9+8+8+8+7+9+7+7+9 = 8.00 = \text{High}$	High
V.Unencrypted_Data	$8+7+8+9+6+3+7 = 6.85 = \text{High}$	$9+8+7+7+6+7+9 = 7.57 = \text{High}$	High
V.Breached_Firewall	$3+4+5+3 = 3.75 = \text{Medium}$	$9+8+8+5 = 7.50 = \text{High}$	High
V.Unsecured_API	$2+3+4 = 3.00 = \text{Medium}$	$7+7+5 = 6.33 = \text{High}$	High
V.Unsecured_Network	$3+3+7+4 = 4.25 = \text{Medium}$	$9+7+9+5 = 7.50 = \text{High}$	High
V.Network_Partition	$2+3+6 = 3.67 = \text{Medium}$	$7+5+8 = 6.67 = \text{High}$	High
V.Unvalidated_Input	$2+2+3+3 = 2.50 = \text{Low}$	$7+6+8+5 = 6.50 = \text{High}$	Medium
V.Untrained_Users	$6+2+2+2+2+2 = 2.67 = \text{Low}$	$7+8+7+8+9+9 = 8.00 = \text{High}$	Medium
V.Monitoring_Absence	$4+3+2+2+3+5+8 = 3.86 = \text{Medium}$	$8+9+9+9+7+9+5 = 8.00 = \text{High}$	High
V.Inadequate_Logging	$2+6+1+1+7+1 = 3.00 = \text{Medium}$	$7+7+3+2+7+8 = 5.67 = \text{Medium}$	Medium
V.Physical_Security	$3+1+1+2 = 1.75 = \text{Low}$	$7+9+4+5 = 6.25 = \text{High}$	Medium
V.Obsolete_System	$2+2 = 2 = \text{Low}$	$3+2 = 2.50 = \text{Low}$	Low

We now prioritize security requirements based on the risk rating of vulnerabilities described in subsection C above. Table 18 shows how we derive overall priorities for the security requirements. Where a single vulnerability is associated with one security requirement, we assign the priority of the vulnerability to the security requirement. However, where there are numerous

vulnerabilities associated with one security requirement, we select an average value for the likelihood and also for the impact and resolve the overall priority for the security requirement.

Table 14: Basic security requirements prioritization

Security Requirements	Vulnerabilities	Likelihood	Impact	Priority	Overall Priority
Identification Authentication Authorization	V.Weak_Access_Control	6.22	8.00	High	High
	V.Untrained_Users	2.67	8.00	Medium	
Immunity	V.Unsecured_API	3.00	6.33	High	High
	V.Unvalidated_Input	2.50	6.50	Medium	
	V.Unsecured_Network	4.25	7.50	High	
	V.Breached_Firewall	3.75	7.50	High	
Integrity	V.Weak_Access_Control	6.22	8.00	High	High
	V.Unencrypted_Data	6.85	7.57	High	
	V.Breached_Firewall	3.75	7.50	High	
	V.Unsecured_Network	4.25	7.50	High	
	V.Untrained_Users	2.67	8.00	Medium	
	V.Inadequate_Logging	3.00	5.67	Medium	
Intrusion Detection	V.Breached_Firewall	3.75	7.50	High	High
	V.Monitoring_Absence	3.86	8.00	High	
	V.Unsecured_Network	4.25	7.50	High	
	V.Unsecured_API	3.00	6.33	High	
Non-repudiation	V.Inadequate_Logging	3.00	5.67	Medium	Medium
Privacy	V.Unencrypted_Data	6.85	7.57	High	High
	V.Breached_Firewall	3.75	7.50	High	
	V.Untrained_Users	2.67	8.00	Medium	
	V.Unsecured_Network	4.25	7.50	High	
	V.Unsecured_API	3.00	6.33	High	
Security Auditing	V.Monitoring_Absence	3.86	8.00	High	High
	V.Inadequate_Logging	3.00	5.67	Medium	
	V.Misconfigurations	2.33	8.33	Medium	
	V.Breached_Firewall	3.75	7.50	High	
	V.Unsecured_API	3.00	6.33	High	
	V.Unvalidated_Input	2.50	6.50	High	
Survivability	V.Network_Partition	3.67	6.67	High	High
Physical Protection	V.Physical_Security	1.75	6.25	Medium	Medium
System Maintenance	V.Obsolete_System	2.00	2.75	Low	Medium
	V.Misconfigurations	2.33	8.33	Medium	
	V.Breached_Firewall	3.75	7.50	High	

Table 15 below shows prioritized security requirements for CREATE operation for the generic model.

Table 19: Prioritized security requirements for CREATE operation for the generic model

Operations	Description	Vulnerable interaction Sequences	Security Requirements	Vulnerabilities	Priority
Create	Create operations are those that add new records	User-> Database Create Request	Identification	V.Weak_Access_Control V.Untrained_Users	High
			Authentication	V.Weak_Access_Control V.Untrained_Users	High
			Authorization	V.Weak_Access_Control V.Untrained_Users	High
			Integrity	V.Weak_Access_Control V.Unencrypted_Data V.Breached_Firewall V.Unsecured_Network V.Untrained_Users V.Inadequate_Logging	High
			Privacy	V.Unencrypted_Data V.Breached_Firewall V.Untrained_Users V.Unsecured_Network V.Unsecured_API	High
			Immunity	V.Unsecured_API V.Unvalidated_Input V.Unsecured_Network V.Breached_Firewall	High
			Intrusion Detection	V.Breached_Firewall V.Monitoring_Absence V.Unsecured_Network V.Unsecured_API	High
			Security Auditing	V.Monitoring_Absence V.Inadequate_Logging V.Misconfigurations V.Breached_Firewall V.Unsecured_API V.Unvalidated_Input	High
			System Maintenance	V.Obsolete_System V.Misconfigurations V.Breached_Firewall	Medium
		Non-repudiation	V.Inadequate_Logging	Medium	
		Database->Database: Create	Security Auditing	V.Unencrypted_Data V.Breached_Firewall V.Monitoring_Absence V.Physical_Security V.Misconfigurations V.Unsecured_API V.Inadequate_Logging	High
			Intrusion Detection	V.Breached_Firewall V.Monitoring_Absence V.Unsecured_API	High
			Integrity	V.Unencrypted_Data V.Breached_Firewall	High

			Privacy	V.Unencrypted_Data V.Breached_Firewall V.Unsecured_API	High
			Immunity	V.Unsecured_API V.Unvalidated_Input V.Unsecured_Network V.Breached_Firewall	High
			System Maintenance	V.Obsolete_System V.Misconfigurations V.Breached_Firewall	Medium
			Physical Protection	V.Physical_Security	Medium
		Database-> User: Return Confirmation	Survivability	V.Network_Partition	High

Our priorities are shown to be mostly High and few Medium ratings. Low ratings are not available. This is due to the fact that we are considering security of data at the source which we have indicated to be most paramount as it is at the highest risk of compromise. Thus no security requirement can have low priority as we cannot afford to leave out any in the database implementation. This risk model may be customized to cater for different systems as we recognize that different systems may have different vulnerabilities and risks.

4.5 Security Requirements Specification

Security Requirements Specification is the final phase in our framework. It involves proper documentation of the requirements. We implement a tool (SeCRUD Tool) for automating the whole process depicted in our proposed framework. Using this tool, we will be able to completely specify all the Security requirements in the form of a Security requirements Specification document. We wish to make the document in a format that can be incorporated in the IEEE standard Software Requirement Specification document format. This document will be used as an input for the design phase.

Chapter Five: Case Study for Security Requirements for Big Data Environments

In this chapter we apply our proposed framework explained in the previous chapter to instances of Big Data storage environments using case studies of MongoDB and Cassandra NoSQL databases. This will help us to validate our framework to show its applicability to the Big Data environment. Also we seek to find out whether our proposed framework is truly generic and can be applied to any Big Data storage environment seamlessly.

5.1 MongoDB Overview

MongoDB is a highly flexible, scalable, schema-less, document-oriented database developed in C++ programming language at 10Gen by Geir Magnusson and Dwight Merriman. The database handles sets of “schema-less JSON-like documents that allow data to be nested in complex hierarchies and still be query-able and index-able“ [5]. In terms of conformance to CAP theorem, MongoDB is a CP store meaning it ensures primarily consistency and partition tolerance while sacrificing a little on availability when necessary. MongoDB claims to put together features of RDBMS, document databases, key_value stores and object databases. MongoDB is chosen here because it is one of the most widely used NoSQL databases by a large spectrum of organizations such as SourceForge, Bit.ly, Foursquare, GitHub, Shutterfly, Evite, The New York Times, Etsy, and many more. Another reason we have chosen MongoDB is that it has very clear and organized documentation available freely on the internet which explains everything. MongoDB architecture is shown in Figure 16 below.

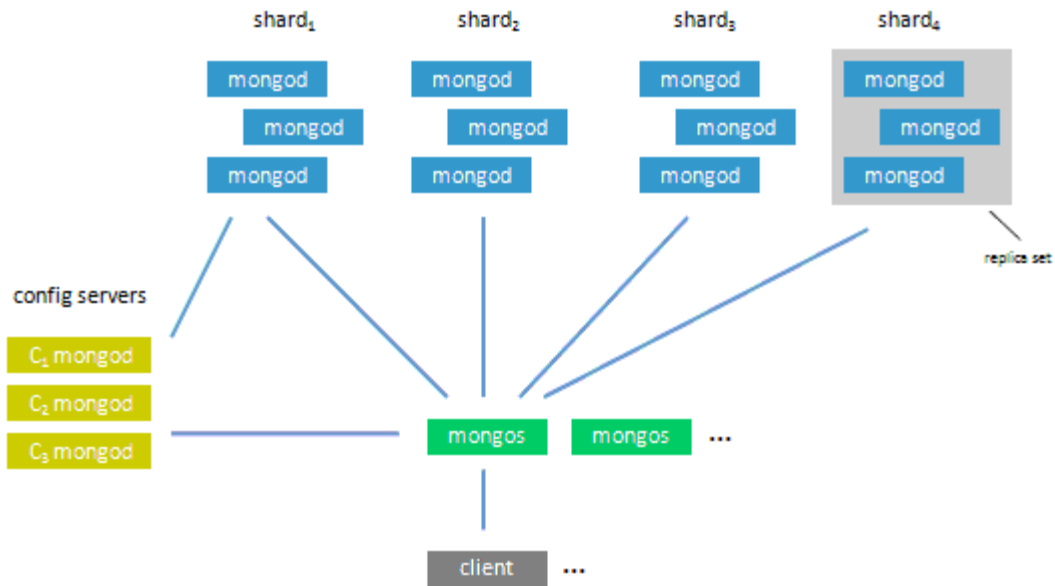


Figure 16: MongoDB Architecture [5]

5.1.1 MongoDB database security

Study on MongoDB has shown that it is also ridden with many security weaknesses common to most NoSQL databases. Okman et al [5] explored some of the security issues in this database. Below are some of the security weaknesses associated with the MongoDB database environment among others which are common to all Big Data environments.

- i. Unencrypted data files
- ii. Unsecured API
- iii. No input validation creating potential for injection attacks.
- iv. No Authentication in sharded mode
- v. No Authorization in sharded mode
- vi. Weak Access control
- vii. Passwords encrypted with MD5 algorithm which is not very secure
- viii. No auditing mechanisms in place

Other security weaknesses that may affect all Big Data Environments

- i. Misconfigurations due to human errors may lead to system failure and various security breaches.
- ii. Breached Firewall enables outsiders to gain access to data
- iii. Unsecured Network exposes data to eavesdroppers.
- iv. Network Partition causes system to be unavailable
- v. Obsolete System may cause both hardware and technical failure which may lead to unavailability of the system
- vi. Physical Security may not be strong enough to safeguard the system
- vii. Untrained Users may make mistakes causing unauthorized modification of data. They may also be targeted by attackers through social engineering, spoofing etc.

5.2 Applying proposed framework to MongoDB

5.2.1 Security Requirements Elicitation

i. Generic Actor Identification

Generic actors we use for MongoDB are:

- a) User
- b) Database

ii. Operations Identification

There are variations in the application of database operations among various available NoSQL databases based on their specific query languages. Table 20 below gives an overview of the MongoDB CRUD Operations [94].

Table 16: MongoDB CRUD Operations

Operations	Description	Method
Create	Create add new records or documents to a collection in MongoDB	insert - The insert() is the primary method to insert a document or documents into a MongoDB collection updates with the upsert option- The update() operation in MongoDB accepts an “upsert” flag that modifies the behaviour of update() from updating existing documents, to inserting data.
Read	Read operations retrieve records or documents from a collection in MongoDB.	find- The find() method is the primary method to select documents from a collection. The find() method returns a cursor that contains a number of documents. findOne The findOne() method selects a single document from a collection and returns that document. findOne() does not return a cursor
Update	Update operations modify existing records or documents in a MongoDB collection. Update operation modifies an existing document or documents in a collection.	update - The update() method is the primary method used to modify documents in a MongoDB collection. By default, the update() method updates a single document, but by using the multi option, update() can update all documents that match the query criteria in the collection. The update() method can either replace the existing document with the new document or update specific fields in the existing document. save- The save() method performs a special type of update(), depending on the _id field of the specified document.
Delete	Delete operations remove documents from a collection in MongoDB	The remove() method in the mongo shell provides this operation, as do corresponding methods in the drivers. Use the remove() method to delete documents from a collection.

iii. Vulnerabilities identification

Sequence diagram for Create operation in MongoDB are shown in Figure 17. It is covering two methods for create operation as mentioned in [90]. Table 21 shows all the output of step 3. Here threats are not mentioned in the table, associated threats are directly extracted from our repository according to the vulnerabilities identified.

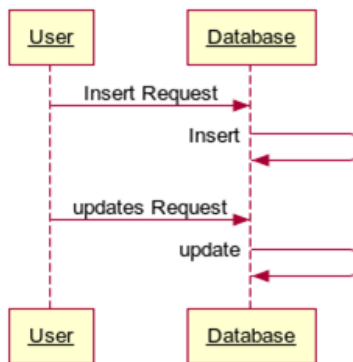


Figure 17: Sequence Diagram for Create operation in MongoDB

iv. Mapping of Vulnerabilities to Security Requirements

Elicited security requirements for create operation in MongoDB are shown in Table 21.

Table 17: MongoDB CREATE Operations

Operations	Method	Vulnerable interaction Sequences	Vulnerabilities	Security Requirements
Create	insert()	User->Database: Insert Request	<ol style="list-style-type: none"> V.Weak_Access_Control V.Unencrypted_Data V.Unsecured_Network V.Monitoring_Absence V.Network_Partition V.Breached_Firewall V.Inadequate_Logging Untrained_Users V.Unsecured_API V.Unvalidated_Input 	<ol style="list-style-type: none"> Identification Authentication Authorization Integrity Privacy Immunity Intrusion Detection Security Auditing System Maintenance Non-repudiation
		Database: ->Database: Insert	<ol style="list-style-type: none"> V.Unencrypted_Data V.Breached_Firewall V.Monitoring_Absence V.Physical_Security V.Misconfigurations V.Unsecured_API V.Physical_Security 	<ol style="list-style-type: none"> Security Auditing Intrusion Detection Integrity Privacy Immunity System Maintenance Physical Protection
		Database ->:User: Return Confirmation	<ol style="list-style-type: none"> V.Network_Partition 	<ol style="list-style-type: none"> Survivability
	update()	User->Database: updates Request	<ol style="list-style-type: none"> V.Weak_Access_Control V.Unencrypted_Data V.Unsecured_Network V.Monitoring_Absence V.Network_Partition V.Breached_Firewall V.Inadequate_Logging Untrained_Users V.Unsecured_API V.Unvalidated_Input 	<ol style="list-style-type: none"> Identification Authentication Authorization Integrity Privacy Immunity Intrusion Detection Security Auditing System Maintenance Non-repudiation
			Database ->Database: update	<ol style="list-style-type: none"> V.Unencrypted_Data V.Breached_Firewall V.Monitoring_Absence V.Physical_Security V.Misconfigurations V.Unsecured_API V.Obsolete_System V.Physical_Security
		Database ->:User: Return Confirmation	<ol style="list-style-type: none"> V.Network_Partition 	<ol style="list-style-type: none"> Survivability

5.2.2 Security Requirements Analysis

We check for completeness that is whether all database operations have been explored and all security requirements elicited. All security requirements are checked for consistency to eliminate all ambiguities.

5.2.3 Security Requirements Prioritization

The prioritized security requirements are shown in Table 22 below.

Table 18: MongoDB security requirements prioritization

Operations	Method	Vulnerable interaction Sequences	Security Requirements	Vulnerabilities	Priority
Create	insert()	User->Database: Insert Request	Identification Authentication Authorization	V.Weak_Access_Control V.Untrained_Users	High
			Integrity	V.Weak_Access_Control V.Unencrypted_Data V.Breached_Firewall V.Unsecured_Network V.Untrained_Users V.Inadequate_Logging	High
			Privacy	V.Unencrypted_Data V.Breached_Firewall V.Untrained_Users V.Unsecured_Network V.Unsecured_API	High
			Immunity	V.Unsecured_API V.Unvalidated_Input V.Unsecured_Network V.Breached_Firewall	High
			Intrusion Detection	V.Breached_Firewall V.Monitoring_Absence V.Unsecured_Network V.Unsecured_API	High
			Security Auditing	V.Monitoring_Absence V.Inadequate_Logging V.Misconfigurations V.Breached_Firewall V.Unsecured_API V.Unvalidated_Input	High
			System Maintenance	V.Obsolete_System V.Misconfigurations V.Breached_Firewall	Medium
			Non-repudiation	V.Inadequate_Logging	Medium
		Database: -	Security Auditing	V.Breached_Firewall	High

		>Database: Insert		V.Monitoring_Absence V.Inadequate_Logging V.Misconfigurations V.Unsecured_API	
			Intrusion Detection	V.Breached_Firewall V.Monitoring_Absence V.Unsecured_API	High
			Integrity	V.Unencrypted_Data V.Breached_Firewall	High
			Privacy	V.Unencrypted_Data V.Breached_Firewall V.Unsecured_API	High
			Immunity	V.Unsecured_API V.Unvalidated_Input V.Unsecured_Network V.Breached_Firewall	High
			System Maintenance	V.Obsolete_System V.Misconfigurations V.Breached_Firewall	Medium
			Physical Protection	V.Physical_Security	Medium
		Database - >:User: Return Confirmation	Survivability	V.Network_Partition	High
	update()	User- >Database: updates Request	Identification Authentication Authorization	V.Weak_Access_Control V.Untrained_Users	High
			Integrity	V.Weak_Access_Control V.Unencrypted_Data V.Breached_Firewall V.Unsecured_Network V.Untrained_Users V.Inadequate_Logging	High
			Privacy	V.Unencrypted_Data V.Breached_Firewall V.Untrained_Users V.Unsecured_Network V.Unsecured_API	High
			Immunity	V.Unsecured_API V.Unvalidated_Input V.Unsecured_Network V.Breached_Firewall	High
			Intrusion Detection	V.Breached_Firewall V.Monitoring_Absence V.Unsecured_Network V.Unsecured_API	High
			Security Auditing	V.Monitoring_Absence V.Inadequate_Logging V.Misconfigurations V.Breached_Firewall V.Unsecured_API V.Unvalidated_Input	High
			System Maintenance	V.Obsolete_System V.Misconfigurations V.Breached_Firewall	Medium
			Non-repudiation	V.Inadequate_Logging	Medium

		Database - >Database: update	Security Auditing	V.Breached_Firewall V.Monitoring_Absence V.Inadequate_Logging V.Misconfigurations V.Unsecured_API	High
			Intrusion Detection	V.Breached_Firewall V.Monitoring_Absence V.Unsecured_API	High
			Integrity	V.Unencrypted_Data V.Breached_Firewall	High
			Privacy	V.Unencrypted_Data V.Breached_Firewall V.Unsecured_API	High
			Immunity	V.Unsecured_API V.Unvalidated_Input V.Unsecured_Network V.Breached_Firewall	High
			System Maintenance	V.Obsolete_System V.Misconfigurations V.Breached_Firewall	Medium
			Physical Protection	V.Physical_Security	Medium
		Database->: User: Return Confirmation	Survivability	V.Network_Partition	High

5.2.4 Security Requirements Specification

We will be able to completely specify all the Security requirements in the form of a Security requirements Specification document using our tool (SeCRUD tool).

5.3 Cassandra database Overview

Cassandra is a NoSQL database management system that has properties of high scalability, high availability with no single point of failure, tuneable consistency, support for replication, flexible schema, very high write throughput and good read throughput. Cassandra was originally written at Facebook to solve their Inbox Search problem. The code was released as an open source Google Code project in 2008. In 2009, it was started as an Apache Incubator project. Cassandra is used at Facebook, Twitter, Cloudkick, Cisco, IBM, eBay, GitHub, GoDaddy, Hulu, Instagram, Intuit, Netflix, Rackspace, Reddit, The Weather Channel, SimpleGeo, Ooyala, and OpenX and many more companies. “The largest known

Cassandra cluster has over 300 TB of data in over 400 machines” according to [95]. Cassandra is shaped by Google’s BigTable (based on Google’s distributed file system) and Amazon’s Dynamo (based on a distributed hash table). Cassandra combines the data structure of BigTable, and the high availability of Dynamo [5]. In terms of conformance to CAP theorem, Cassandra is an AP data store. That means it primarily focuses on availability and partition tolerance. However, it has tuneable consistency. Advantages of Cassandra are shown in Table 23 below.

Table 19: Advantages of Cassandra

Advantage	Description
Proven	Wide usage by companies with large, active datasets
Distributed	Cassandra is distributed across multiple nodes while transparent to the user.
Fault Tolerant	Automatic replication of data to multiple nodes for fault tolerance
Performance	Cassandra’s architecture allows for high read and write performance
Decentralised	There is no single point of failure
Durable	Data is managed in Cassandra with durability thus it is very ideal for critical applications Use of a commit log enforces durability so that data is not lost in the event of system failure
Elastic	Read and write throughput both increase linearly as new nodes are added
Tuneable Consistent	Cassandra enables the user to adjust the level of consistency according to their requirements

5.3.1 Cassandra database security

Okman [5] in their study of security in NoSQL databases also considered Cassandra Security issues and some of the issues are listed below:

- i. Data at rest is unencrypted
- ii. Client communication-no encryption available
- iii. Authentication –available solution not production ready
- iv. Authorization –available solution not production ready
- v. Cassandra uses Log4J for its logging utility.
- vi. Intercluster network Communication – encryption available
- vii. Injection Attacks – possible in CQL
- viii. Weak Access control
- ix. Passwords encrypted with MD5 algo not very secure

Other security weaknesses that may affect all Big Data Environments

- ii. Misconfigurations due to human errors may lead to system failure and various security breaches.
- iii. Breached Firewall enables outsiders to gain access to data
- iv. Unsecured Network exposes data to eavesdroppers.
- v. Network Partition causes system to be unavailable
- vi. Obsolete System may cause both hardware and technical failure which may lead to unavailability of the system
- vii. Physical Security may not be strong enough to safeguard the system
- viii. Untrained Users may make mistakes causing unauthorized modification of data. They may also be targeted by attackers through social engineering, spoofing etc.

5.4 Applying proposed framework to Cassandra

5.4.1 Security Requirements Elicitation

i. Generic Actor Identification

Generic actors we use for Cassandra are:

- a) User
- b) Database

ii. Operations Identification

There are variations in the application of database operations among various available NoSQL databases based on their specific query languages. Table 24 below gives an overview of the Cassandra database Operations and their variations

Table 20: Cassandra Operations

Operation	Variations	Description
CREATE	CREATE Keyspace	Creates a new top-level namespace (Keyspace)
	CREATE Column Family	Creates new column family namespaces
	CREATE Index	Creates new automatic secondary index on the column family for the named column
READ	Get	Use the GET operation to retrieve columns or super columns, using a column path to access them
	getCol	Get a single key-column value
	getMulti	Do multiple gets
	SELECT	SELECT is used to read one or more records from a column family
UPSERT	INSERT	INSERT is used to write one or more columns to a new record in a column family
	UPDATE	UPDATE writes one or more column values to existing columns in a Cassandra column family. No results are returned.
REMOVE	DELETE	DELETE Removes entire rows or one or more columns from one or more rows.
	TRUNCATE	TRUNCATE statement results in the immediate, irreversible removal of all data in the named column family.

It can be noted that the CREATE operation in Cassandra is used differently from the way it is used in MongoDB. In Cassandra UPSERT operation is used similarly as CREATE in MongoDB. Thus for better comparison we will use UPSERT operation to apply our framework to Cassandra.

iii. Vulnerabilities identification

Sequence diagram for Create operation in Cassandra are shown in Figure 18. Table 25 shows all the output of step 3. Here threats are not mentioned in the table, associated threats are directly extracted from our repository according to the vulnerabilities identified.

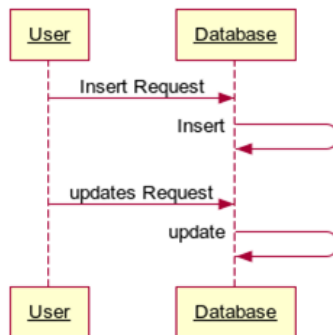


Figure 18: Sequence Diagram for UPSERT operation in Cassandra

iv. Mapping of Vulnerabilities to Security Requirements

Elicited security requirements for UPSERT operation in Cassandra are shown in Table 25.

Table 21: Cassandra UPSERT Operations

Operations	Method	Vulnerable interaction Sequences	Vulnerabilities	Security Requirements
Upsert	insert() (Where record does not exist)	User->Database: Insert Request	<ol style="list-style-type: none"> V.Weak_Access_Control V.Unencrypted_Data V.Unsecured_Network V.Monitoring_Absence V.Network_Partition V.Breached_Firewall V.Inadequate_Logging Untrained_Users V.Unsecured_API V.Unvalidated_Input 	<ol style="list-style-type: none"> Identification Authentication Authorization Integrity Privacy Immunity Intrusion Detection Security Auditing System Maintenance Non-repudiation
		Database: ->Database: Insert	<ol style="list-style-type: none"> V.Unencrypted_Data V.Breached_Firewall V.Monitoring_Absence V.Physical_Security V.Misconfigurations V.Unsecured_API V.Physical_Security 	<ol style="list-style-type: none"> Security Auditing Intrusion Detection Integrity Privacy Immunity System Maintenance Physical Protection
		Database ->:User: Return Confirmation	<ol style="list-style-type: none"> V.Network_Partition 	<ol style="list-style-type: none"> Survivability
	update() (Where record exists)	User->Database: updates Request	<ol style="list-style-type: none"> V.Weak_Access_Control V.Unencrypted_Data V.Unsecured_Network V.Monitoring_Absence V.Network_Partition V.Breached_Firewall V.Inadequate_Logging Untrained_Users V.Unsecured_API V.Unvalidated_Input 	<ol style="list-style-type: none"> Identification Authentication Authorization Integrity Privacy Immunity Intrusion Detection Security Auditing System Maintenance Non-repudiation
		Database ->Database: update	<ol style="list-style-type: none"> V.Unencrypted_Data V.Breached_Firewall V.Monitoring_Absence V.Physical_Security V.Misconfigurations V.Unsecured_API V.Obsolete_System V.Physical_Security 	<ol style="list-style-type: none"> Security Auditing Intrusion Detection Integrity Privacy Immunity System Maintenance Physical Protection
		Database ->:User: Return Confirmation	<ol style="list-style-type: none"> V.Network_Partition 	<ol style="list-style-type: none"> Survivability

5.4.2 Security Requirements Analysis

We check for completeness that is whether all database operations have been explored and all security requirements elicited. All security requirements are checked for consistency to eliminate all ambiguities. Grouping is performed on the basis of vulnerabilities mitigated by same security requirements.

5.4.3 Security Requirements Prioritization

Table 26 below shows the prioritized security requirements for Cassandra.

Table 22: Cassandra security requirements prioritization

Operations	Method	Vulnerable interaction Sequences	Security Requirements	Vulnerabilities	Priority
Upsert	insert()	User->Database: Insert Request	Identification Authentication Authorization	V.Weak_Access_Control Untrained_Users	High
			Integrity	V.Weak_Access_Control V.Unencrypted_Data V.Breached_Firewall V.Unsecured_Network V.Untrained_Users V.Inadequate_Logging	High
			Privacy	V.Unencrypted_Data V.Breached_Firewall V.Untrained_Users V.Unsecured_Network V.Unsecured_API	High
			Immunity	V.Unsecured_API V.Unvalidated_Input V.Unsecured_Network V.Breached_Firewall	High
			Intrusion Detection	V.Breached_Firewall V.Monitoring_Absence V.Unsecured_Network V.Unsecured_API	High
			Security Auditing	V.Monitoring_Absence V.Inadequate_Logging V.Misconfigurations V.Breached_Firewall V.Unsecured_API V.Unvalidated_Input	High
			System Maintenance	V.Obsolete_System V.Misconfigurations V.Breached_Firewall	Medium
			Non-repudiation	V.Inadequate_Logging	Medium

		Database: - >Database: Insert	Security Auditing	V.Breached_Firewall V.Monitoring_Absence V.Inadequate_Logging V.Misconfigurations V.Unsecured_API	High
			Intrusion Detection	V.Breached_Firewall V.Monitoring_Absence V.Unsecured_API	High
			Integrity	V.Unencrypted_Data V.Breached_Firewall	High
			Privacy	V.Unencrypted_Data V.Breached_Firewall V.Unsecured_API	High
			Immunity	V.Unsecured_API V.Unvalidated_Input V.Unsecured_Network V.Breached_Firewall	High
			System Maintenance	V.Obsolete_System V.Misconfigurations V.Breached_Firewall	Medium
			Physical Protection	V.Physical_Security	Medium
		Database - >:User: Return Confirmation	Survivability	V.Network_Partition	High
	update()	User- >Database: updates Request	Identification Authentication Authorization	V.Weak_Access_Control Untrained_Users	High
			Integrity	V.Weak_Access_Control V.Unencrypted_Data V.Breached_Firewall V.Unsecured_Network V.Untrained_Users V.Inadequate_Logging	High
			Privacy	V.Unencrypted_Data V.Breached_Firewall V.Untrained_Users V.Unsecured_Network V.Unsecured_API	High
			Immunity	V.Unsecured_API V.Unvalidated_Input V.Unsecured_Network V.Breached_Firewall	High
			Intrusion Detection	V.Breached_Firewall V.Monitoring_Absence V.Unsecured_Network V.Unsecured_API	High
			Security Auditing	V.Monitoring_Absence V.Inadequate_Logging V.Misconfigurations V.Breached_Firewall V.Unsecured_API V.Unvalidated_Input	High
			System Maintenance	V.Obsolete_System V.Misconfigurations V.Breached_Firewall	Medium

			Non-repudiation	V.Inadequate_Logging	Medium
		Database - >Database: update	Security Auditing	V.Breached_Firewall V.Monitoring_Absence V.Inadequate_Logging V.Misconfigurations V.Unsecured_API	High
			Intrusion Detection	V.Breached_Firewall V.Monitoring_Absence V.Unsecured_API	High
			Integrity	V.Unencrypted_Data V.Breached_Firewall	High
			Privacy	V.Unencrypted_Data V.Breached_Firewall V.Unsecured_API	High
			Immunity	V.Unsecured_API V.Unvalidated_Input V.Unsecured_Network V.Breached_Firewall	High
			System Maintenance	V.Obsolete_System V.Misconfigurations V.Breached_Firewall	Medium
			Physical Protection	V.Physical_Security	Medium
		Database->: User: Return Confirmation	Survivability	V.Network_Partition	High

5.4.4 Security Requirements Specification

We will be able to completely specify all the Security requirements in the form of a Security requirements Specification document using our SeCRUD tool.

Chapter Six: Implementation and Results

In this chapter we present implementation and results of our work. We have developed a tool which we call SeCRUD, derived from our consideration of security requirements based on CRUD database operations for Big Data in our framework. This tool is intended to support the four phases of our proposed framework that is Security Requirements Elicitation, Analysis, Prioritization and Specification. This tool can be used by software engineers working on Big Data Projects to automate the Security Requirements Engineering in software development.

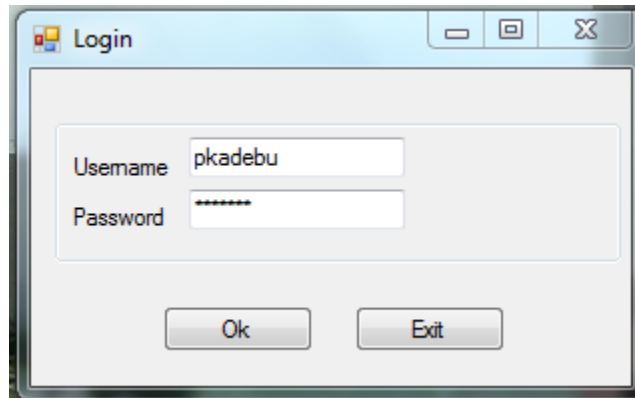
6.1 Tool Overview

SeCRUD developed using .NET framework and implemented in C# programming language, using a SQL Server 2005 database and SQL Server Management Studio Express. The C# language is a simple, modern, general-purpose, object-oriented programming language. C# was chosen because the language, and implementations thereof, provide support for software engineering principles such as strong type checking, array bounds checking, detection of attempts to use uninitialized variables, and automatic garbage collection.

Microsoft SQL Server 2005 is a relational database management system developed by Microsoft. SQL Server Management Studio Express is a GUI tool included with SQL Server 2005 and later for configuring, managing, and administering all components within Microsoft SQL Server. The tool includes both script editors and graphical tools that work with objects and features of the server. This was chosen over alternatives like Access because of the following benefits: improved reliability, better performance, reduced network traffic and increased scalability. We opted for a relational database management system (RDBMS) for our tool development because the data we are working with is structured and very limited. Thus it is quite manageable using an RDBMS.

6.2 Tool Snapshots

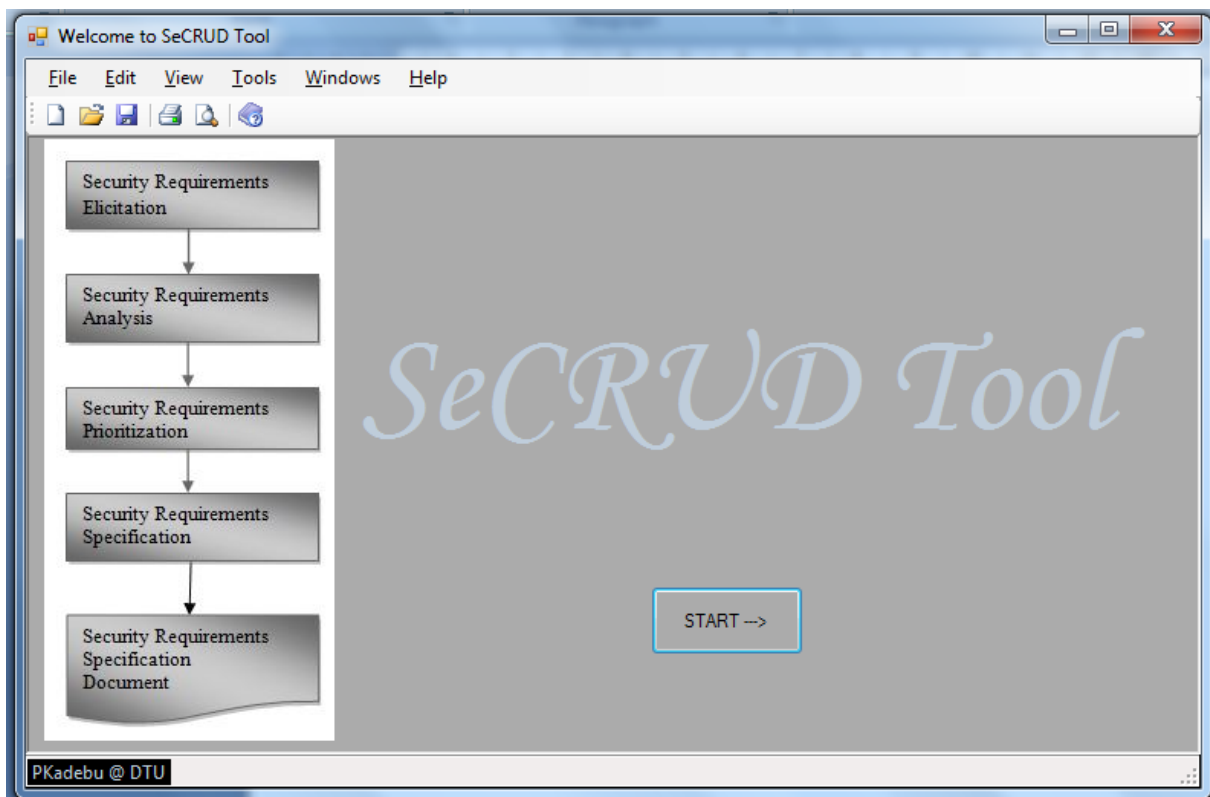
Security in SeCRUD is provided by Access Control using a username and password that is created by the administrator shown in Figure 19.



The screenshot shows a standard Windows-style dialog box titled "Login". It contains two text input fields: "Username" with the text "pkadebu" entered, and "Password" with a masked password represented by seven asterisks. Below the input fields are two buttons: "Ok" and "Exit".

Figure 19: Login Form

Main Form is shown in Figure 20.



The screenshot shows the main window of the "SeCRUD Tool". The title bar reads "Welcome to SeCRUD Tool". The menu bar includes "File", "Edit", "View", "Tools", "Windows", and "Help". On the left side, there is a vertical flowchart with five steps: "Security Requirements Elicitation", "Security Requirements Analysis", "Security Requirements Prioritization", "Security Requirements Specification", and "Security Requirements Specification Document". The main area of the window is a large grey rectangle with the text "SeCRUD Tool" in a large, light blue, cursive font. At the bottom right of this area is a button labeled "START -->". The status bar at the bottom left shows "PKadebu @ DTU".

Figure 20: Main Form

Initial stage in security requirements elicitation is shown in Figure 21 below.

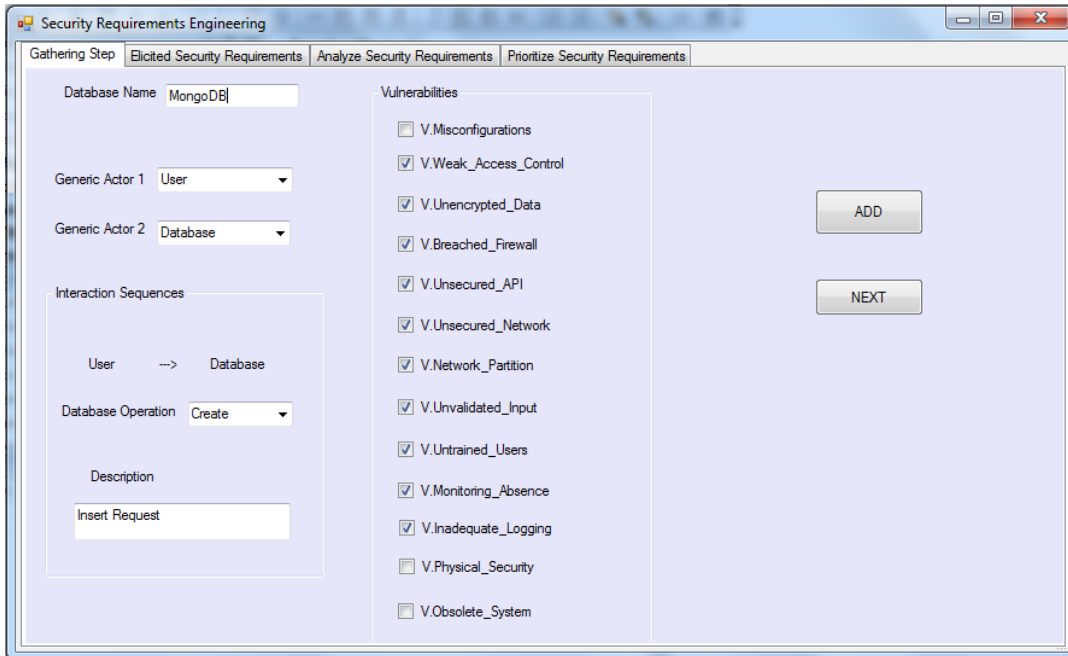


Figure 21: Initial stage of security requirements elicitation

Figure 22 below shows the elicited security requirements.

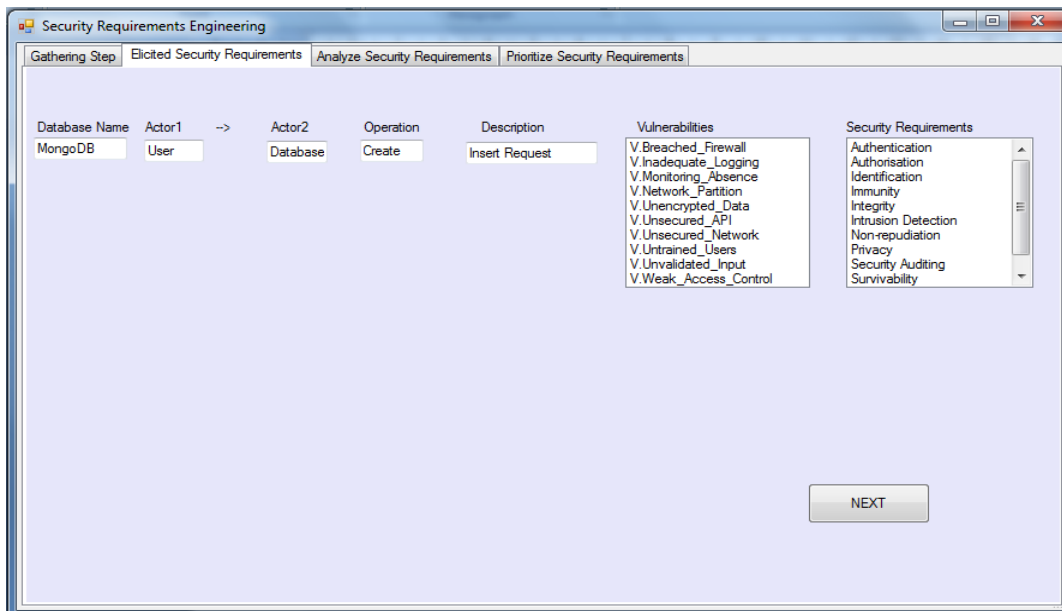


Figure 22: Elicited Security Requirements

Figure 23 below shows the analysis of security requirements

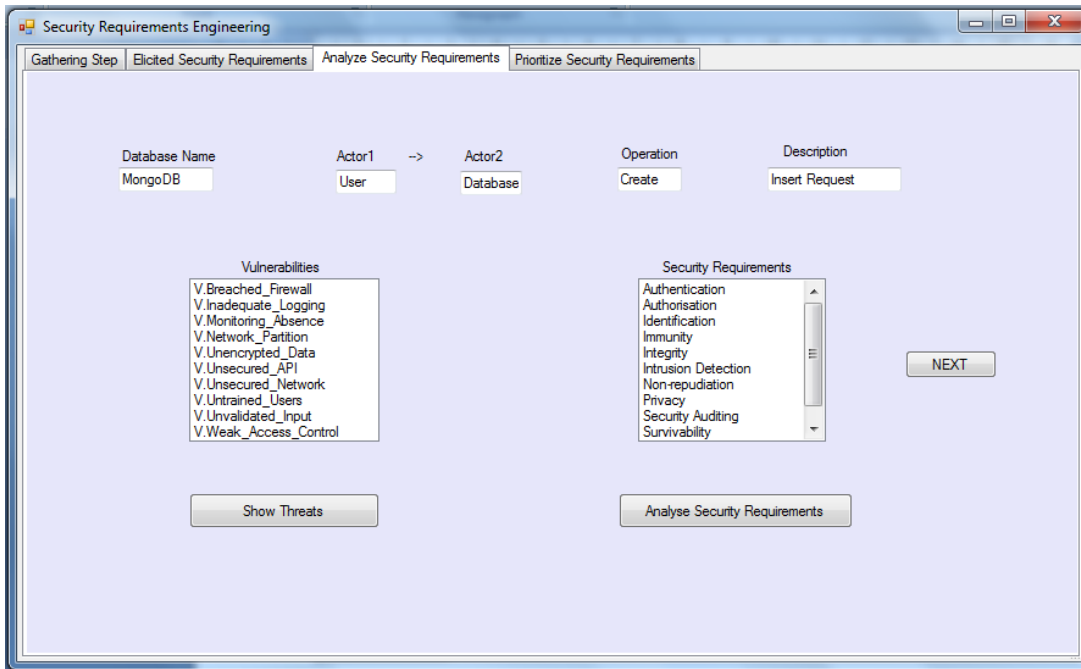


Figure 23: Analysis of Security Requirements

Figure 24 below shows the vulnerability/threat analysis popup.

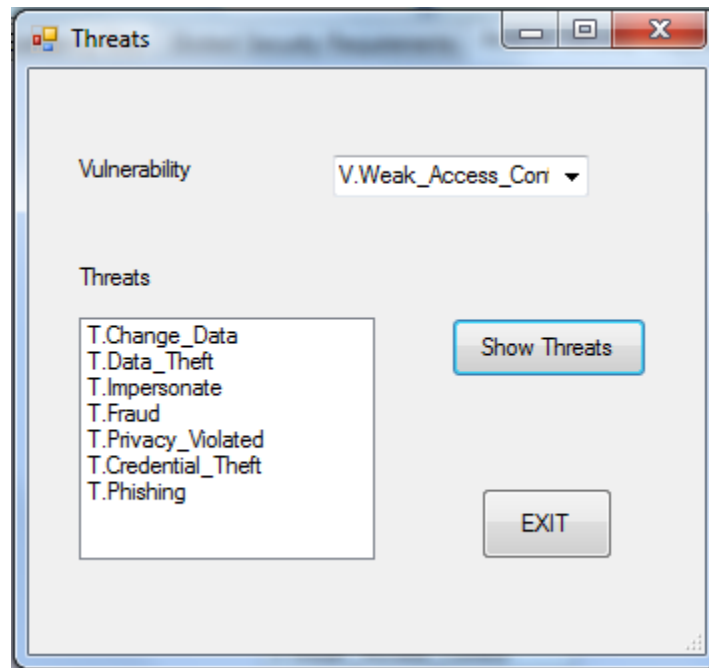


Figure 24: Vulnerabilities and Threats Popup

Figure 25 below shows the Security Requirements analysis popup.

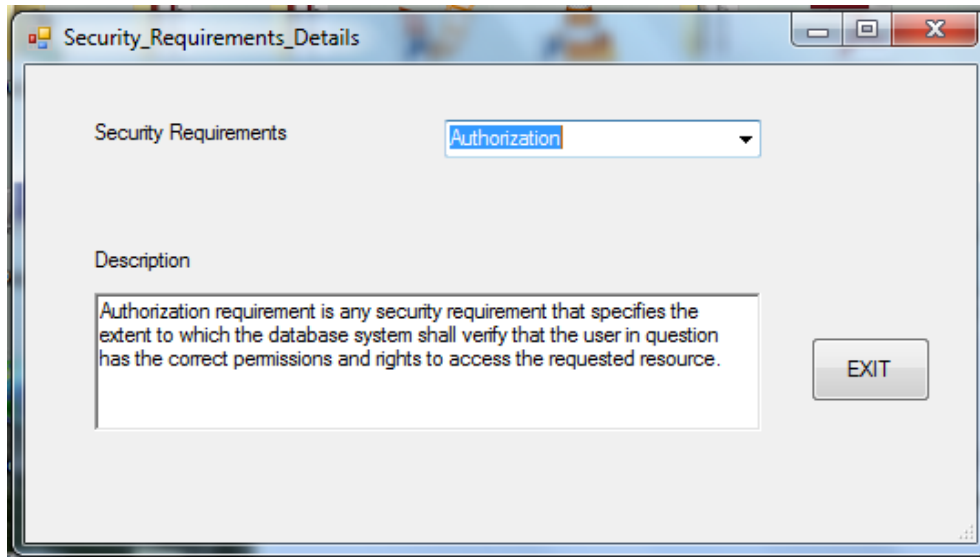


Figure 25: Security Requirements Details

Figure 26 below shows the prioritized security requirements.

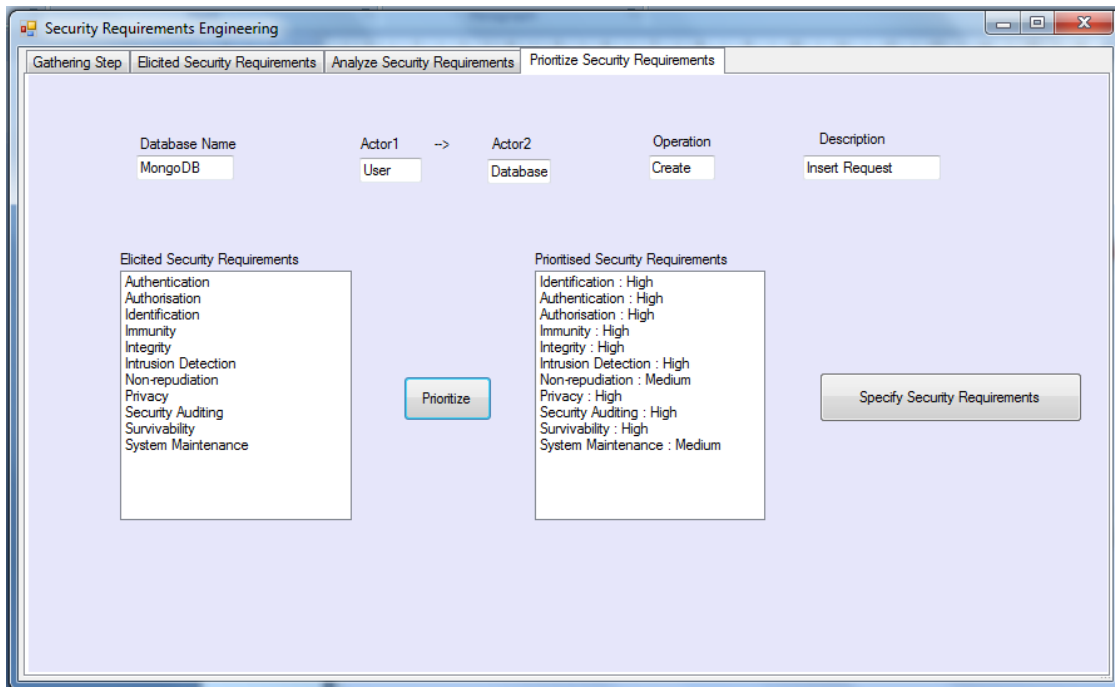


Figure 26: Security Requirements Prioritization

Figure 27 shows the Security Requirements Specification.

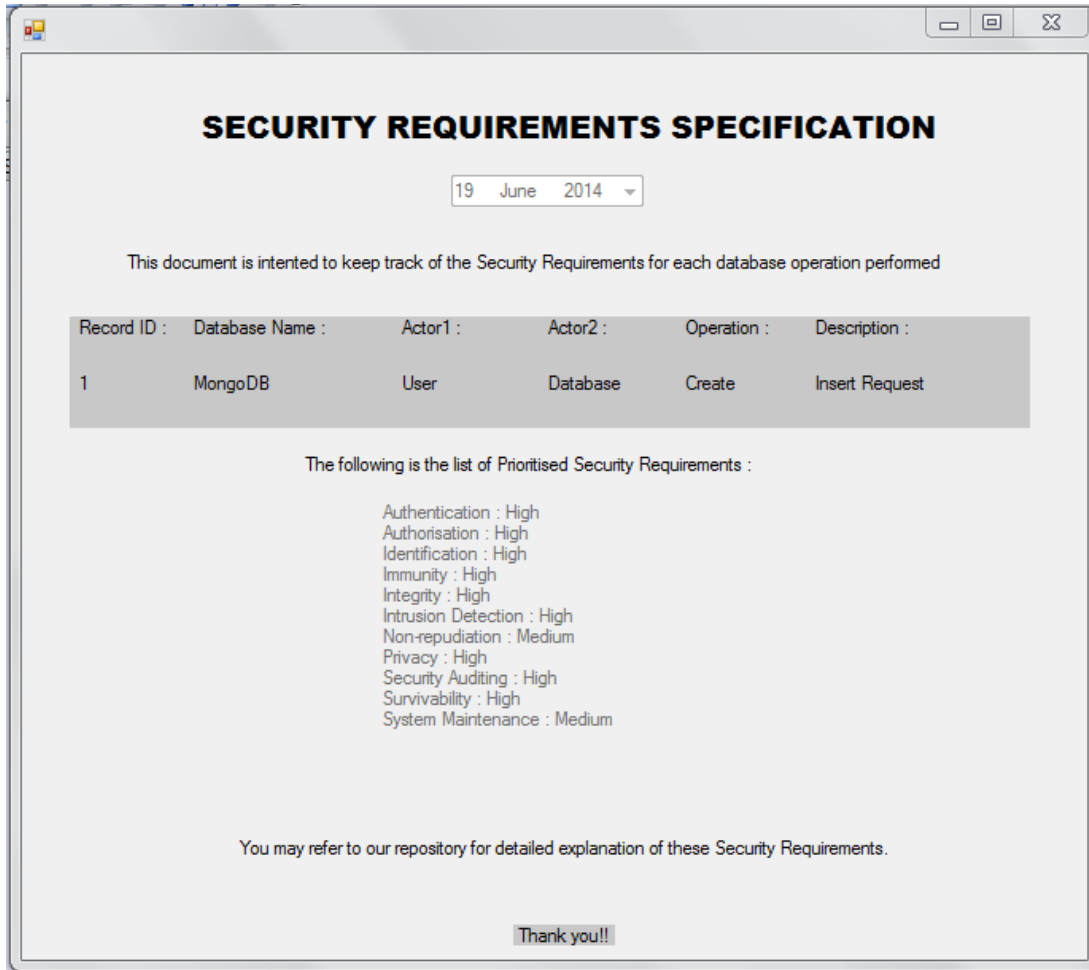


Figure 27: Security Requirements Specification

6.3 Results

We have shown in this thesis that securing Big Data environments is not an easy task due to complexity of these environments. However careful consideration of security requirements in development of Big Data databases ensures that data is protected at the source. In this thesis we were able to come up with true security requirements suitable for Big Data to mitigate inherent vulnerabilities as given below.

- i. **V.Weak_Access_Control**
Identification, Authentication, Authorization and Integrity security requirements are used to mitigate V.Weak_Access_Control.
.
- ii. **V.Unencrypted_Data**
Privacy security requirements and Integrity security requirements are used to mitigate V.Unencrypted_Data.
.
- iii. **V.Misconfigurations**
System Maintenance, Security Auditing security requirements are used to mitigate V.Misconfigurations..
- iv. **V.Breached_Firewall**
Privacy, Integrity, Intrusion Detection, Immunity, Security Auditing and System Maintenance security requirements are used to mitigate V.Breached_Firewall.
- v. **V.Unsecured_API**
Privacy, Integrity, Intrusion Detection, Immunity and Security Auditing security requirements are used to mitigate V.Unsecured_API.
- vi. **V.Unsecured_Network**
Privacy, Integrity, Intrusion Detection and Immunity security requirements are used to mitigate V.Unsecured_Network.
- vii. **V.Network_Partition**
Survivability security requirements are used to mitigate V.Network_Partition.
- x. **V.Unvalidated_Input**
Security Auditing and Immunity security requirements are used to mitigate V.Unvalidated_Input.

- xi. **V.Untrained_Users**
Identification, Authentication, Authorization, Integrity and Privacy security requirements are used to mitigate this vulnerability.

- xii. **V.Monitoring_Absence**
Security Auditing and Intrusion Detection security requirements are used to mitigate V.Monitoring_Absence.

- xiii. **V.Inadequate_Logging**
Non-repudiation and Security Auditing security requirements are used to mitigate V.Inadequate_Logging.

- xiv. **V.Physical_Security**
Physical Protection security requirements are used to mitigate V.Physical_Security.

- xv. **V.Obsolete_System**
System Maintenance security requirements are used to mitigate V.Obsolete_System.

Chapter Seven: Conclusion and Future Works

There is no universal definition of security requirements as shown by the varied explanations given by different authors on the topic. However all explanations converge to the fact that Security Requirements Engineering is of paramount importance in software engineering and in building Big Data database systems that are robust and able to withstand various forms of threats and attacks. If Security Requirements Engineering is properly performed it can assist the software engineering team in making correct decisions on mechanisms to be considered in the Design Phase for Big Data databases.

Generally the security vulnerabilities inherent in MongoDB and Cassandra databases were found to be almost similar. It is clear that the basic designs of these databases are highly lacking in security. Third party products are being availed that can help alleviate the inadequacy of security in these databases. This situation may seem to be good enough. However, it means more cost to the organisations using these databases to ensure their data is secured. It will be worthwhile to have adequate security built into the databases as they are developed which ultimately lowers costs.

Our framework may be adopted as a generic model for providing security in different Big Data databases as shown in the MongoDB and Cassandra case studies. The elicited security requirements for MongoDB and Cassandra which have their own methods for applying the CRUD operations are same as the generic big database operations. We concur that controls should be applied as close to the data as possible to protect from both external and internal threats. Thus, if the database is secure then it will be easier and less costly for application developers to secure the application program.

7.1 Limitations and Future Work

- i. Due to time constraints we were unable to incorporate UML diagram generation in our tool to show the sequence diagrams. Rather we took a descriptive approach only.
- ii. The whole process is somewhat winding which may be a bit cumbersome to apply.
- iii. Our Security Requirements Specification is not detailed enough and the format is not up to the standard we had initially intended.

As a part of our future work, more case studies of the Big Data stores as well as the Relational databases will be explored to verify whether our framework can be applied to any database. We will validate our methodology in various emerging domains. We will work on the security requirements specification format which can be easily incorporated in a Software Requirements Specification document. Furthermore, we will associate operation with functionality for an application for instance in Healthcare e.g. Read->Browse patient history to come up with a more detailed approach that covers both the frontend and the backend security.

Chapter Eight: Publication from this Thesis

8.1 Communicated Paper

1. Prudence Kadebu , Shruti Jaiswal , Daya Gupta, “Security Requirements Elicitation Process for Big Data: A Case Study of MongoDB”, in International Conference on Data Mining and Intelligent Computing 2014

Appendix

1. Abbreviations

CC	Common Criteria
SQUARE	Security Quality Requirements Engineering
STRIDE	Spoofing Identity, Tampering with data, Repudiation, Information disclosure, Denial of service, Elevation of privilege
DREAD	Damage potential, Reproducibility, Exploitability, Affected users, Discoverability
MUC	Misuse Cases
UML	Unified Modelling Language
CARE	Computer Aided Requirements Engineering
IT	Information Technology
SQL	Structured Query Language
NoSQL	Not Only Structured Query Language
RDBMS	Relational Database Management System
VOSREP	Viewpoint Oriented Security Requirements Elicitation process
CRUD	Create, Read, Update, Delete
JSON	JavaScript Object Notation
XML	eXtensible Markup Language
3Vs	Volume, Velocity and Variety

2. Prioritized security requirements for READ, UPDATE and DELETE

Prioritized security requirements for READ operation for the generic model

Operations	Description	Vulnerable interaction Sequences	Security Requirements	Vulnerabilities	Priority
Read	Read operations are those that retrieve records.	User-> Database Read Request	Identification	V.Weak_Access_Control V.Untrained_Users	High
			Authentication	V.Weak_Access_Control V.Untrained_Users	High
			Authorization	V.Weak_Access_Control V.Untrained_Users	High
			Security Auditing	V.Monitoring_Absence V.Inadequate_Logging V.Misconfigurations V.Breached_Firewall V.Unsecured_API V.Unvalidated_Input	High
			Intrusion Detection	V.Breached_Firewall V.Monitoring_Absence V.Unsecured_Network V.Unsecured_API	High
			Survivability	V.Network_Partition	High
			Non-repudiation	V.Inadequate_Logging	Medium
		Database->Database: Search	Security Auditing	V.Unencrypted_Data V.Breached_Firewall V.Monitoring_Absence V.Physical_Security V.Misconfigurations V.Unsecured_API V.Inadequate_Logging	High
			System Maintenance	V.Obsolete_System V.Misconfigurations V.Breached_Firewall	Medium
			Intrusion Detection	V.Breached_Firewall V.Monitoring_Absence V.Unsecured_API	High
			Integrity	V.Unencrypted_Data V.Breached_Firewall	High
			Privacy	V.Unencrypted_Data V.Breached_Firewall V.Unsecured_API	High
			Immunity	V.Unsecured_API V.Unsecured_Network V.Unvalidated_Input V.Breached_Firewall	High
			Physical Protection	V.Physical_Security	Medium
			Database->User: Return result	Integrity	V.Unencrypted_Data V.Breached_Firewall
		Privacy		V.Unencrypted_Data V.Breached_Firewall V.Unsecured_API	High

			Immunity	V.Unsecured_API V.Unsecured_Network V.Unvalidated_Input V.Breached_Firewall	High
			Intrusion Detection	V.Breached_Firewall V.Monitoring_Absence V.Unsecured_API	High
			Security Auditing	V.Unencrypted_Data V.Breached_Firewall V.Monitoring_Absence V.Physical_Security V.Misconfigurations V.Unsecured_API V.Inadequate_Logging	High
			Survivability	V.Network_Partition	High
			Non-repudiation	V.Inadequate_Logging	Medium

Table 19: Prioritized security requirements for UPDATE operation for the generic model

Operations	Description	Vulnerable interaction Sequences	Security Requirements	Vulnerabilities	Priority
Update	Update operations are those that modify/edit existing records	User->Database: Update Request	Identification	V.Weak_Access_Control V.Untrained_Users	High
			Authentication	V.Weak_Access_Control V.Untrained_Users	High
			Authorization	V.Weak_Access_Control V.Untrained_Users	High
			Integrity	V.Weak_Access_Control V.Unencrypted_Data V.Breached_Firewall V.Unsecured_Network V.Untrained_Users V.Inadequate_Logging	High
			Privacy	V.Unencrypted_Data V.Breached_Firewall V.Untrained_Users V.Unsecured_Network V.Unsecured_API	High
			Immunity	V.Unsecured_API V.Unvalidated_Input V.Unsecured_Network V.Breached_Firewall	High
			Intrusion Detection	V.Breached_Firewall V.Monitoring_Absence V.Unsecured_Network V.Unsecured_API	High

			Security Auditing	V.Monitoring_Absence V.Inadequate_Logging V.Misconfigurations V.Breached_Firewall V.Unsecured_API V.Unvalidated_Input	High
			System Maintenance	V.Obsolete_System V.Misconfigurations V.Breached_Firewall	Medium
			Survivability	V.Network_Partition	High
		Database->Database: Update	Security Auditing	V.Unencrypted_Data V.Breached_Firewall V.Monitoring_Absence V.Physical_Security V.Misconfigurations V.Unsecured_API V.Inadequate_Logging	High
			Intrusion Detection	V.Breached_Firewall V.Monitoring_Absence V.Unsecured_API	High
			Integrity	V.Unencrypted_Data V.Breached_Firewall	High
			Privacy	V.Unencrypted_Data V.Breached_Firewall V.Unsecured_API	High
			Immunity	V.Unsecured_API V.Unsecured_Network V.Unvalidated_Input V.Breached_Firewall	High
			System Maintenance	V.Obsolete_System V.Misconfigurations V.Breached_Firewall	Medium
			Physical Protection	V.Physical_Security	Medium
			Survivability	V.Network_Partition	High
		Database-> User: Return Confirmation	Survivability	V.Network_Partition	High

Table 19: Prioritized security requirements for DELETE operation for the generic model

Operations	Description	Vulnerable interaction Sequences	Security Requirements	Vulnerabilities	Priority
Delete	Delete operations are those that remove or deactivate existing entries	User->Database: Delete Request	Identification	V.Weak_Access_Control V.Untrained_Users	High
			Authentication	V.Weak_Access_Control V.Untrained_Users	High
			Authorization	V.Weak_Access_Control V.Untrained_Users	High
			Integrity	V.Weak_Access_Control V.Unencrypted_Data V.Breached_Firewall V.Unsecured_Network V.Untrained_Users V.Inadequate_Logging	High
			Privacy	V.Unencrypted_Data V.Breached_Firewall V.Untrained_Users V.Unsecured_Network V.Unsecured_API	High
			Immunity	V.Unsecured_API V.Unvalidated_Input V.Unsecured_Network V.Breached_Firewall	High
			Intrusion Detection	V.Breached_Firewall V.Monitoring_Absence V.Unsecured_Network V.Unsecured_API	High
			Security Auditing	V.Monitoring_Absence V.Inadequate_Logging V.Misconfigurations V.Breached_Firewall V.Unsecured_API V.Unvalidated_Input	High
			System Maintenance	V.Obsolete_System V.Misconfigurations V.Breached_Firewall	Medium
		Survivability	V.Network_Partition	High	
		Database->Database: Delete	Security Auditing	V.Unencrypted_Data V.Breached_Firewall V.Monitoring_Absence V.Physical_Security V.Misconfigurations V.Unsecured_API V.Inadequate_Logging	High
		Intrusion Detection	V.Breached_Firewall V.Monitoring_Absence V.Unsecured_API	High	

			Integrity	V.Unencrypted_Data V.Breached_Firewall	High
			Privacy	V.Unencrypted_Data V.Breached_Firewall V.Unsecured_API	High
			Immunity	V.Unsecured_API V.Unsecured_Network V.Unvalidated_Input V.Breached_Firewall	High
			System Maintenance	V.Obsolete_System V.Misconfigurations V.Breached_Firewall	Medium
			Physical Protection	V.Physical_Security	Medium
			Survivability	V.Network_Partition	High
			Non-repudiation	V.Inadequate_Logging	Medium
		Database-> User: Return Confirmation	Survivability	V.Network_Partition	High

References

- [1] Diya Soubra, "The 3Vs that define Big Data", Posted on July 5, 2012 on Data Science Central, <http://www.datasciencecentral.com/forum/topics/the-3vs-that-define-big-data>, Accessed on 05/04/2014
- [2] 3Vs (volume, variety and velocity), Posted by Margaret Rouse, <http://whatis.techtarget.com/definition/3Vs>, Accessed on 05/04/2014
- [3] Chris Evans, "Big data storage: Hadoop storage basics", published on October 2013 <http://searchbusinessintelligence.techtarget.in/feature/Big-data-storage-Hadoop-storage-basics>
- [4] Complete guide to Hadoop technology and storage <http://searchstorage.techtarget.com/essentialguide/Complete-guide-to-Hadoop-technology-and-storage> Accessed on 05/04/2014
- [5] Lior Okman, Nurit Gal-Oz, Yaron Gonen, Ehud Gudes, Jenny Abramov, Security Issues in NoSQL Databases, 2011 International Joint Conference of IEEE TrustCom-11/IEEE ICSS-11/FCST-11
- [6] Amoroso, E. G., Fundamentals of Computer Security Technology, Prentice-Hall, ISBN: 0-13-305541-8, 1994
- [7] T M Kiran Kumar, "A Road Map to the Software Engineering Security", IEEE 2009.
- [8] Toktam Ramezani Farkhani, Mohammad Reza Razzazi Examination and Classification of Security Requirements of Software Systems", 2006 IEEE
- [9] Avita Katal, Mohammad Wazid, R H Goudar, "Big Data: Issues, Challenges, Tools and Good Practices", IEEE 2013
- [10] Charles B. Haley, Robin Laney, Jonathan D. Moffett, "Security Requirements Engineering: A Framework for Representation and Analysis", IEEE Transactions On Software Engineering, Vol. 34, No. 1,), pp.133-153. January/February 2008
- [11] Hui Wang, Zongpu Jia, Zihao Shen, "Research on Security Requirements EngineeringProcess", IEEE 2009
- [12] A. Finkelstein, H. Fuks, "Multiparty Specification", Proc.Fifth Int'l Workshop Software Specification and Design, pp.185-195, 1989.
- [13] "Matter of Card Systems Solutions Inc.," Washington, D.C., Federal Trade Commission, 2006.
- [14] B. Schneier, "Secrets & Lies", John Wiley & Sons, Inc., 2000.
- [15] "Common Criteria for Information Technology Security Evaluation", version 3.1, reversion 1, Sep. 2006.
- [16] Donald Firesmith: Engineering Security Requirements, in Journal of Object Technology, vol. 2, no. 1, January-February 2003, pages 53-68.
- [17] M. Glinz, "Rethinking the Notion of Non-Functional Requirements," Proc. Third World Congress for Software Quality, vol. II, pp. 55-64, 2005.
- [18] L. Chung, B. Nixon, E. Yu, and J. Mylopoulos, Non-Functional Requirements in Software Engineering. Kluwer Academic, 2000.
- [19] N. R. Mead, T. Stehney, "Security Quality Requirements Engineering (SQUARE) Methodology," Proc. of the 2005 workshop on software engineering for secure systems-building trustworthy applications, Missouri, USA, 2005, pp.1-7.

- [20] Yu, E.: Agent-Oriented Modelling: Software Versus the World (2001), Agent-oriented software engineering AOSE-2001 Workshop Proceedings, LNCS 2222, pp. 206-225.
- [21] D. Gordon, T. Stehney, N. Wattas and E. Yu, "Quality Requirements Engineering (SQUARE): Case Study on Asset Management System," Phase II (CMU/SEI-2005-SR-005). Pittsburgh, PA, Software Engineering Institute, Carnegie Mellon University, 2005.
- [22] Golnaz Elahi, Eric Yu, Tong Li, Lin Liu "Security Requirements Engineering in the Wild: A Survey of Common Practices", 2011 35th IEEE Annual Computer Software and Applications Conference
- [23] Daniel Mellado, Eduardo Fernández-Medina, Mario Piattini, "Security Requirements Engineering Process for Software Product Lines: A Case Study", The Third International Conference on Software Engineering Advances IEEE 2008
- [24] Daniel Mellado , Eduardo Fernandez-Medina , and Mario Piattini: Applying a Security Requirements Engineering Process D. Gollmann, J. Meier, and A. Sabelfeld (Eds.) ESORICS 2006, LNCS 4189, pp. 192–206, 2006 c Springer-Verlag Berlin Heidelberg 2006
- [25] Daniel Mellado, Jesús Rodríguez, Eduardo Fernández-Medina and Mario Piattini, "Automated Support for Security Requirements Engineering in Software Product Line Domain Engineering", 2009 International Conference on Availability, Reliability and Security.
- [26] Sojan Markose, Xiaoqing (Frank) Liu, and Bruce McMillin, "A Systematic Framework for Structured Object-Oriented Security Requirements Analysis in Embedded Systems", 2008 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing
- [27] Jing-Song CUI, Da ZHANG, "The Research and Application of Security Requirements Analysis Methodology of Information Systems"
- [28] Lin Liu, Eric Yu, John Mylopoulos: Security and Privacy Requirements Analysis within a Social Setting. in: IEEE Joint International Conference on Requirements Engineering 2003, California, USA. pp. 151-161.
- [29] G. Sindre and A. L. Opdahl, "Eliciting Security Requirements by Misuse Cases," presented at TOOLS Pacific 2000, Sydney, 2000.
- [30] John McDermott and Chris Fox, "Using Abuse Case Models for Security Requirements Analysis", IEEE 2009
- [31] Luncheng Lin, Bashar Nuseibeh, Darrel Ince, Michael Jackson, Jonathan Moffett, "Introducing Abuse Frames for Analysing Security Requirements", Proceedings of the 11th IEEE International Requirements Engineering Conference 2003
- [32] Ian Sommerville and Pete Sawyer, "Viewpoints: principles, problems and a practical approach to requirements engineering", Annals of Software Engineering 3 (1997) 101–130
- [33] Emilio Soler, Veronika Stefanov, Jose-Norberto Mazón, Juan Trujillo, Eduardo Fernández-Medina, Mario Piattini, "Towards Comprehensive Requirement Analysis for Data Warehouses: Considering Security Requirements", The Third International Conference on Availability, Reliability and Security, IEEE 2008

- [34] Elisa Bertino, and Ravi Sandhu,, “Database Security—Concepts, Approaches, and Challenges”, Transactions On Dependable And Secure Computing, Vol. 2, No. 1, IEEE 2005
- [35] Premchand B. Ambhore,B.B.Meshram,V.B.Waghmare, “A Implementation of Object Oriented Database Security”, Fifth International Conference on Software Engineering Research, Management and Applications, IEEE 2007
- [36] Leon Pan, “A Unified Network Security and Fine-Grained Database Access Control Model”, Second International Symposium on Electronic Commerce and Security, IEEE 2009
- [37] secure and protect big data environments, www.ibm.com, secure and protect big data environments-
- [38] Prudence Kadebu, Innocent Mapanga, “A Security Requirements Perspective Towards A Secured Nosql Database Environment”, International Journal of Advance Research and Innovation, February 2014
- [39] Emilio Soler, Veronika Stefanov, Jose-Norberto Maz´on, Juan Trujillo, Eduardo Fern´andez-Medina, Mario Piattini, “Towards Comprehensive Requirement Analysis for Data Warehouses: Considering Security Requirements”, The Third International Conference on Availability, Reliability and Security, IEEE 2008
- [40] Ashish Agarwal, Daya Gupta, “Security Requirements Elicitation Using View Points for Online System”, IEEE 2008
- [41] Jing Du, Ye Yang, Qing Wang, “An analysis for understanding Software Security Requirement Methodologies”, Third IEEE International Conference on Secure Software Integration and Reliability Improvement 2009
- [42] Inger Anne Tøndel, Jostein Jensen, Lillian Røstad, “Combining misuse cases with attack trees and security activity models”, 2010 International Conference on Availability, Reliability and Security IEEE
- [43] Myagmar, S., A. J. Lee, and W. Yurcik, Threat Modeling as a Basis for Security Requirements.2005: SREIS 2005.
- [44] M. H. Diallo, J. Romero-Mariona, S. E. Sim, T. A. Alspaugh, and D. J. Richardson, “A comparative evaluation of three approaches to specifying security requirements,” in 12th Working Conference on Requirements Engineering: foundation for Software Quality (REFSQ’06), 2006.
- [45] B. Schneier, “Attack trees,” Dr. Dobb’s Journal, December 1999.
- [46] F. Swiderski and W. Snyder, Threat modeling. Microsoft Press Redmond, WA, USA, 2004.
- [47] G Sindre and A Opdahl: Templates for Misuse Case Description, 2001
- [48] Guttorm Sindre, Andreas L. Opdahl,” Eliciting Security Requirements by Misuse Cases”, Springer-Verlag London Limited 2004
- [49] Peter Karpati, Andreas L. Opdahl, Guttorm Sindre, “Experimental Comparison of Misuse Case Maps with Misuse Cases and System Architecture Diagrams for Eliciting Security Vulnerabilities and Mitigations”, 2011 IEEE
- [50] Security Use Cases - OPFRO www.opfro.org/Components/WorkProducts/.../SecurityUseCases.html

- [51] Donald Firesmith: "Security Use Cases", in Journal of Object Technology, vol. 2, no. 3, May-June 2003, pp. 53-64.
- [52] Hiroga Itoga, Atsushi Ohnishi, "Security Requirements Elicitation via weaving Scenarios based on security Evaluation Criteria", Seventh International Conference on Quality Software IEEE 2007
- [53] Ian F Alexander, Neil Maiden,"Scenarios, Stories, Use Case-through the Systems Development Life Cycle", John Wiley & Sons, 2004
- [54] Nancy R. Mead , "How To Compare the Security Quality Requirements Engineering (SQUARE) Method with Other Methods", August 2007, CMU/SEI-2007-TN-021
- [55] Ware M S., Bowles J.B., Eastman C.M., "Using the Common Criteria to Elicit Security Requirements with Use Cases." SoutheastCon. IEEE, pp 273-278, 2006.
- [56] Adrian Lane, "Securing Big Data: Security Recommendations for Hadoop and NoSQL Environments", Securosis October 12, 2012
- [57] Saaty T.L. How to make a decision: The analytic hierarchy process, European Journal of Operational Research, 1990 : 9-26
- [58] T. Biggerstaff and C. Richter, "Reusability Framework, Assessment and Directions," IEEE Software, vol. 4, pp. 41-49, 1987.
- [59] A. Toval, J. Nicolas, B. Moros, and F. Garcia, "Requirements Reuse for Improving Information Systems Security: A Practitioner's Approach," Requirements Engineering Journal, vol. 6, pp. 205-219, 2002.
- [60] Guttorm Sidre, Donald G. Firesmith, and Andreas L. Opdahl, "A Reuse-Based Approach to Determining Security Requirements", 9th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'03), Austria, 2003.
- [61] A. van Lamsweerde and E. Letier, "Handling Obstacles in Goal-Oriented Requirements Engineering," IEEE Transactions on Software Engineering, vol. 26, pp. 978-1005, 2000.
- [62] Lin Liu, Eric Yu, John Mylopoulos:Analyzing Security Requirements As Relationships among Strategic Actors.in: 2nd Symposium on Requirements Engineering for Information Security (SREIS), 2002.
- [63] Olawande Daramola, Yushan Pan, Peter Karpati, Guttorm Sindre, "A Comparative Review of i*-based and Use Case based Security Modelling Initiatives", 2011 IEEE
- [64] Jaelson Castro, Manuel Kolp, John Mylopoulos, "Towards Requirements-Driven Information Systems Engineering: The Tropos Project"
- [65] Haralambos Mouratidis, Paolo Giorgini, "Secure Tropos: a security-oriented extension of the Tropos methodology", international journal of software engineering and knowledge engineering, available at disi.unitn.it/~pgiorgio/papers/ijseke06-1.pdf
- [66] R. Hassan , M. Eltoweissy, S. Bohner S. El-Kassas, "Formal analysis and design for engineering security automated derivation of formal software security specifications from goal-oriented security requirements", IET Softw., 2010, Vol. 4, Iss. 2, pp. 149–160
- [67] Dianxiang Xu, Vivek Goel, and Kendall Nygard, "An Aspect-Oriented Approach to Security Requirements Analysis", IEEE 2006

- [68] I. Jacobson and W. Ng, *Aspect-Oriented Software Development with Use Cases*, Addison Wesley, 2005.
- [69] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J. Loingtier, and J. Irwin. "Aspect-Oriented Programming". In *Proc. of ECOOP'97*, pp. 220-242.
- [70] Joseph McKendrick, Research Analyst, "The Petabyte Challenge: 2011 IOUG Database Growth Survey", Produced by Unisphere Research, A Division of Information Today, Inc. August 2011
- [71] Sang-soo Choi, Soo-young Chae, and Gang-soo Lee, "SRS-Tool: A Security Functional Requirement Specification Development Tool for Application Information System of Organization", *Computational Science and Its Applications*, Berlin, 2005.
- [72] P. Giorgini, F. Massacci, J. Mylopoulos, and N. Zannone. *Requirements Engineering meets Trust Management: Model, Methodology, and Reasoning*. In *Proceedings of the Second International Conference on Trust Management (iTrust 2004)*, Lecture Notes in Computer Science 2995, pages 176-190. Springer-Verlag Heidelberg, 2004.
- [73] D. Mellado, E. Fernandez-Medina, and M. Piattini, "Security Requirements Variability for Software Product Lines", *Symposium on Requirements Engineering for Information Security (SREIS 2008)* co-located with ARES 2008, pp. 1413-1420, 2008.
- [74] M. Georgiades, A. Andreou, and C. Pattichis, "A Requirements Engineering Methodology based on Natural Language Syntax and Semantics," *Proc. of the 13th IEEE International Requirements Engineering Conference (RE'05)*, August 29-September 02, 2005, Paris, France, pp. 473-474
- [75] S Dascalu, E Fritzinger, K Cooper, N Debnath, "A software tool for requirements specification on using the STORM environment to create SRS document", 2007
- [76] Smitha T, V. Suresh Kumar, "Application of Big Data in Data Mining", *International Journal of Emerging Technology and Advanced Engineering*, Volume 3, Issue 7, July 2013
- [77] James Boyd, "BIG DATA :National data linkage infrastructure", Centre for Data Linkage Curtin University
- [78] Tavo De León, "Big: Data New Frontiers for IT Management", www.BigDataArchitecture.com
- [79] Big-Data's-Impact-In-The World, http://www.nytimes.com/2012/02/12/sunday-review/big-datas-impact-in-the-world.html?pagewanted=all&_r=0
- [80] Du Zhang, "Inconsistencies in Big Data", *Proc. 12th IEEE Int. Conf. on Cognitive Informatics & Cognitive Computing (ICCI*CC'13) IEEE 2013*
- [81] D. Laney, "3-D Data Management: Controlling Data Volume, Velocity and Variety. Application Delivery Strategies; <http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf>.
- [82] D. Klein, P. Tran-Gia, M. Hartmann "Big Data", *Informatik- Spektrum*, vol 36, issue 3, pp319-323, June 2013
- [83] Xin Luna Dong, Divesh Srivastava, "Big Data Integration", *ICDE Conference IEEE 2013*
- [84] HDFS Architecture Guide, http://www.hadoop.apache.org/docs/r1.2.1/hdfs_design.html, Posted 4 August 2013, Accessed 4 February 2014

- [85] Institute for Health Technology Transformation, "Transforming Health Care Through Big Data: Strategies for leveraging big data in the health care industry", The Institute for Health Technology Transformation
- [86] Paul. C Zikopoulos, Chris Eaton, Dirk deRoss, Thomas Deutsche, George Lapis, "Understanding Big Data: Analytics for Enterprise Class Hadoop and streaming Data", MacGraw-Hill, ISBN 978-0-07-179053-6, 2012
- [87] Kris Alman, "Big Data and Lawful Threats to Privacy",
<http://www.thelundreport.org/content/big-data-and-lawful-threats-privacy>, accessed on 28 February 2014
- [88] A "MongoDB vs Oracle - database comparison", IEEE 2012
- [89] Eben Hewitt, Cassandra: The Definitive Guide, 2011
- [90] Michael S. Ware John B. Bowles Caroline M. Eastman, "Using the Common Criteria to Elicit Security Requirements with Use Cases"
- [91] www.wikipedia.org/wiki/Create,_read,_update,_and_delete Accessed on 12/05/2014
- [92] Tanya Baccam, "Making Database Security an IT Security Priority", A SANS Whitepaper – November 2009
- [93] OWASP Risk Rating Methodology, http://www.owasp.org/index.php/OWASP_risk_rating_methodology ,
Accessed on 30 May 2014
- [94] MongoDB CRUD Operations Introduction Release 2.2.7, April 15, 2014
- [95] <http://cassandra.apache.org/> Accessed on 30/05/2014
- [96] Tanya Baccam, "Making Database Security an IT Security Priority", A SANS Whitepaper – November
- [97] Gary Stoneburner, Alice Goguen, and Alexis Feringa, "Risk Management Guide for Information Technology Systems: Recommendations of the National Institute of Standards and Technology", NIST Special Publication 800-30 July 2002
- [98] "OWASPD – Open Web Application Security Project. Top ten most critical web application vulnerabilities," 2005.
- [99] S. B. G, V. K. Maurya, E. Jangam, M. S. V, A. K. Talukder, and A. R. Pais, "Suraksha: A security designers' workbench," in Proceedings of Hack.in 2009, March 2009, pp. 59–66.
- [100] Sylvester Ngoma, "Vulnerability of IT Infrastructures: Internal and External Threats", March 04, 2012
- [101] Colwill C, "Human factors in information security: The insider threat- Who can you trust these days?", Information Security Technical Report, 14, 186-196. 2010
- [102] Critical Controls for Effective Cyber Defense Version 4.1, March, 2013