A

Dissertation

On

An Engineering Framework to Implement Security in Software System

Submitted in fulfillment of the requirements for the degree of

Doctor of Philosophy

(Computer Engineering)

By

SHRUTI JAISWAL

(University Roll no. 2K11/Ph.D./CO/10)

Under the supervision of:

Dr. Daya Gupta

Professor, Dept. of Computer Engineering, Delhi Technological University (DTU)

To the

DEPARTMENT OF COMPUTER ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

NEW DELHI -110042

2018

DECLARATION

I, Shruti Jaiswal, Ph.D. student (Roll No. 2K11/PhD/CO/10), hereby declare that the thesis entitled "*An Engineering Framework to Implement Security in Software System*" which is being submitted for the award of the degree of Doctor of Philosophy in Computer Engineering, is a record of bonafide research work carried out by me in the Department of Computer Engineering, Delhi Technological University. I further declare that the work presented in the thesis had not been submitted to any University or Institution for any degree or diploma.

Date: ____________

Place: New Delhi

Shruti Jaiswal
(Candidate)
2K11/Ph.D./CO/10
Department of Computer Engineering
Delhi Technological University (DTU)
New Delhi -110042

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)

CERTIFICATE

Date: ___________

This is to certify that the work embodied in the thesis entitled ***"An Engineering Framework to Implement Security in Software System"*** submitted by ***Shruti Jaiswal*** with **R*oll no. 2K11/PhD/CO/10*** as a *full-time* research scholar in the Department of Computer Engineering, Delhi Technological University, is an authentic work carried out by her under my guidance and is submitted to Delhi Technological University for the award of the Degree of **Doctor of Philosophy**.

This work is original research and has not been submitted, in part or full, to any other University or Institute for the award of any degree.

Supervisor

Dr. Daya Gupta
Professor,
Department of Computer Engineering
Delhi Technological University
Delhi-110042

SHAHBAD DAULATPUR, BAWANA ROAD, DELHI-110042, INDIA
OFF.:91-11-27871018 FAX: +91-11-27871023 WEBSITE: www.dtu.ac.in

ACKNOWLEDGEMENT

After an intensive journey of my dissertation, first and foremost I address my deepest thank to Almighty God, for his showers of blessings that provide me patience, calmness and motivated me throughout the dissertation.

I would like to express my sincere gratitude to my research supervisor, **Dr. Daya Gupta**, for giving me the opportunity to do research and providing invaluable guidance throughout this research. Her guidance, vision, and motivation have deeply inspired me. Throughout my thesis, she provided encouragement, sound advice, deep guidance and lots of good ideas that help me to improve myself and my work.

I extend my gratitude to **Prof. S.K. Garg** (DRC Chairman), **Prof. H.C. Taneja** (Dean, PG), **Dr. Anil Kumar** (AR, PG), **Prof. Yogesh Singh** (Vice Chancellor), **Prof. O.P.Verma** and other faculty members and staff of the Computer Engineering Department for providing me an healthy environment to complete my research at Delhi Technological University.

I am extremely grateful to my fellow researchers and my M.Tech scholars, Mr. Ashish Kumar Tripathi, Mr. Shubham Jain, Mr. Surya Kant Josyula, Ms. Aanchal Punia, and Mr. Gautam Verma for their valuable assistance, co-operation and great source of learning. I would also like to thank my roommates, Shagufta Khan and Nikita Gupta, who supported and inspired me during my stay in Delhi Technological University. They always made me intuitive and inquisitive which helps me a lot. I am likewise thankful to who have directly or indirectly helped me to finish my dissertation study.

Last but not the least I am extremely grateful to my loving and caring husband, Mr. Ankur Gupta. My parents in law, Mrs. Geeta Rani, Dr. Rakesh Gupta, and parents, Mrs. Kusum Jaiswal, Mr. Shiva Shanker, for their support, love, care and sacrifices for educating and preparing me for my future endeavor.

Shruti Jaiswal
May, 2018

ABSTRACT

Secure software development is the need of today's world because of the rise in the incidences of security breach like data theft, man in the middle attack, and others. Many techniques are available for handling security issues in software systems. However, it is recognized that none of the current proposals has complete engineering process to implement security in software system development. Also, a seamless integration of security engineering in the software development process is required.

This thesis addresses the issues by providing a methodology to develop a secure software system. The solution starts by developing a three-phase framework for security engineering, namely security requirements engineering, security design engineering, and security testing. Phases of the novel proposal are not independent they work together to achieve our goal of secure software development.

During the security requirements engineering phase, in addition to functional and non-functional requirements security requirements are also elicited, analyzed, prioritized and specified. During the design phase, efficient algorithms for implementation of security requirements are selected based on the applicable domain constraints. After that, testing of system security level is done to ensure that most of the security threats are mitigated.

The proposed generic framework is applied to new emerging domains like cloud computing, internet of things and big data databases. These new domains are vulnerable to various new threats and issues. These fields are becoming very common, as they have numerous functionalities, assets, threats, etc. To cater the need

of handling issues, repositories are made for easy access instead of tedious manual task.

Further, a tool to automate the process of secure software development is built to help the users in knowing system security needs and standards.

PUBLICATIONS FROM THE THESIS

International Journal:

1. Jaiswal, S., and Gupta, D. (2017). Engineering and validating security to make cloud secure. *International Journal of System Assurance Engineering and Management*, pp. 1-23, DOI: 10.1007/s13198-017-0612-x.

   [Scopus Indexed]

2. Jaiswal, S., and Gupta, D. (2017). Security Engineering Methods: In-Depth Analysis. *International Journal of Information and Computer Security*, 9 (3), 180-211.

   [Scopus Indexed]

3. Jaiswal, S., and Gupta, D. (2016). Measuring Security: A step towards Enhancing Security of System. *International Journal of Information Systems in the Service Sector (IJISSS). (accepted)*

   [ESCI, Scopus Indexed]

International Conference:

4. Gupta, D., Chatterjee, K., and Jaiswal, S. (2013). A Framework for Security Testing. *ICCSA- 2013,* ***published by*** *Springer-Verlag in Lecture Notes for Computer Science.* **[Scopus Indexed]**

5. Jaiswal, S., and Gupta, D. (2014). Security Requirements Engineering: A Challenge for Cloud System. *Proceedings of 'Second International Conference on Emerging*

Research in Computing, Information, Communication and Applications'(ERCICA-14), by Elsevier.

6. Jaiswal, S., and Gupta, D. (2016). Security Requirements for Internet of Things (IOT). *ComNet-2016 Issues and Challenges with IOT Revolution', published in ASIC series by Springer.* **[Scopus Indexed]**

LIST OF ABBREVIATIONS

AAA	Authentication, Authorization, Accounting
ACID	Atomicity, Consistency, Integrity, Durability
ACL	Access Control List
AES	Advance Encryption Standard
AOMDV-IoT	Ad-hoc on demand Multipath Distance Vector routing protocol for IoT
API	Application Program Interface
CBC	Cipher Block Chaining
CC	Common Criteria
CCTA	Central Computer and Telecommunications Agency
CIA	Confidentiality, Integrity, and Availability
CMS	Content Management System
CP	Cloud Provider
CPU	Central Processing Unit
CRAMM	CCTA Risk Analysis and Management Method
CRUD	Create, Read, Update, Delete
CSA	Cloud Security Standard
CSP	Cloud Service Provider
CSRF	Cross Site Request Forgery
CSU	Cloud Service User
CVE	Common Vulnerabilities and Exposures
DAC	Discretionary Access Control
DB	Data Base
DBMS	Database Management System

DDoS	Distributed Denial of Service
DES	Data Encryption Standard
DoS	Denial of Service
DSA	Digital Signature Algorithm
EC2	Elastic Computer Cloud
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
ECG	Electrocardiogram
ECIES	Elliptic Curve Integrated Encryption Scheme
ENISA	European Network and Information Security Agency
EARA	Energy-aware Ant Routing Algorithm
FAST	Facilitated Application Specification Technique
GAE	Google App Engine
GCM	Galois/Counter Mode
GSM	Global System for Mobile
HECC	Hyper Elliptic Curve Cryptography
HECDSA	Hyper Elliptic Curve Digital Signature Algorithm
HIPAA	Health Insurance Portability and Accountability Act
HIV	Human Immunodeficiency Virus
HR	Human Resource
IaaS	Infrastructure as a Service
IDC	International Data Corporation
IDS	Intrusion Detection System
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol

IoT	Internet of Things
IPS	Intrusion Prevention System
IPSec	Internet Protocol Security
ISAE	International Standard in Assurance Engagements
ISO	International Organization for Standardization
JSON	Java Script Object Notation
KDC	Key Distribution Center
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
MAC	Mandatory Access Control
MD5	Message Digest 5
MITM	Man in the Middle
NoSQL	No Structured Query Language
NIST	National Institute of Standard Technology
OS	Operating System
OWASP	Open Web Application Security Project
PaaS	Platform as a Service
PAIR	Pruned Adaptive IoT Routing
PB	Peta Bytes
PCI-DSS	Payment Card Industry Data Security Standard
PKI	Public Key Infrastructure
RBAC	Role-Based Access Control
RDBMS	Relational Database Management System
RE	Requirements Engineering
REL	Routing protocol based on Energy and Link Quality

RFID	Radio Frequency Identification
RnR	Roles and Responsibility
RPC	Remote Procedure Call
RPL	Routing protocol over low power and lossy networks
RSA	Rivest, Shamir, and Adleman
SaaS	Software as a Service
SAS	Statement on Auditing Standards
SCRAM	Salted Challenge Response Authentication Mechanism
SDLC	Software Development Life Cycle
SHA1	Secure Hash Algorithm 1
SI	Security Index
SLA	Security Level Agreement
SMRP	Secure Multihop Routing Protocol
SR	Security Requirement
SSAE	Statement on Standards for Attestation Engagement
SSL	Secure Socket Layer
StaaS	Storage as a Service
SQL	Structured Query Language
SQUARE	Software Quality Requirements Engineering
SREP	Software Requirements Engineering Process
SRR	Security Resource Repository
SRS	Software Requirements Specification
TB	Tera Bytes
TCP	Transmission Control Protocol
TLS	Transport Layer Security

UID	Unique Identification Scheme
VP	Viewpoint
WAN	Wireless Area Network
WLAN	Wireless LAN
WSN	Wireless Sensor Network
XML	eXtensible Markup Language
XSS	Cross Site Scripting

CONTENTS

Acknowledgement……………………………………………………………….III

Abstract……………………………………………………………...…...V

Publications from the Thesis………………………………………………...….VII

List of Abbreviations…………………………………………………….....IX

Contents……………………………………………………………………XIV

List of Figures………………………………………………………….....XIX

List of Table…………………………………………………………........XXII

Chapter 1 1

Introduction...1

1.1 Software Development Life Cycle...1

1.2 Requirements Engineering...3

1.2.1 Requirements 3

1.2.2 Requirements Engineering Process...6

1.3 Design Engineering...9

1.4 Testing...10

1.5 Security 11

1.6 Security Requirements 12

1.7 Related Work 19

1.8 Synthesizing the "Problem Statement" 24

1.9 Proposed Solution 29

1.10 Outline of the Thesis 33

Chapter 2 35

Literature Survey...35

2.1 Overview....35

2.2 Use Case Based Approach....36

2.3 Goal Oriented Approach....42

2.4 Process Oriented Approach....48

2.5 Evaluation of Security Engineering Methods....55

Summary....58

Chapter 3 61

Framework for Security Engineering....61

3.1 Security Engineering Framework....61

3.2 Security Requirements Engineering Phase....63

 3.2.1 Security Requirements Elicitation....65

 3.2.2 Security Requirements Prioritization and Specification....72

3.3 Security Design Engineering....76

3.4 Security Testing....81

3.5 Case Study: Content Management System....84

 3.5.1 Security Requirements Engineering....85

 3.5.2 Security Requirements Prioritization and Specification....93

 3.5.3 Security Design Engineering....94

 3.5.4 Security Testing....98

3.6 Case Study of Open Source Software: WordPress....99

Summary....106

Chapter 4 109

Security Framework for Cloud Systems....109

4.1 Security Issues in Cloud....109

 4.1.1 Cloud Architecture....110

4.1.2 Security Issues 112

4.2 Existing Proposals for Cloud-based Systems 115

4.3 Need for New Framework 117

4.4 Proposed Framework for Cloud-based systems....................................118

4.4.1 Specification 119

4.4.2 Prioritization 132

4.4.3 Implementation and Validation...134

4.5 Application of Proposed Framework on Cloud Storage Model..........................138

4.5.1 Security Requirements Specification..138

4.5.2 Prioritization 142

4.5.3 Implementation and Validation...143

4.6 Evaluation of the Existing Cloud Storage Models..................................147

4.7 Case Study of Open Source Software: ownCloud 151

Summary...153

Novel Contribution of the Chapter 154

Chapter 5 156

Security Framework for IoT systems..156

5.1 Internet of Things...156

5.1.1 IoT Architecture..158

5.1.2 Difference between IoT Security and Network Security..........................159

5.2 Security Issues in IoT...161

5.3 Existing Proposals in IoT Security 163

5.4 Need for New Framework 166

5.5 Security Engineering framework for IoT based Systems 167

5.5.1 Identification and Specification 167

5.5.2 Design and Validation....183

5.6 Case Study: Patient Monitoring System....188

5.6.1 Identification and Specification....189

5.6.2 Design and Validation....197

Summary....198

Novel Contribution of the Chapter....199

Chapter 6....201

Security Engineering Framework for Big Data Databases....201

6.1 Security Issues in Big Data Databases....201

6.1.1 Overview....202

6.1.2 Security Issues....204

6.2 Need for New Framework....206

6.3 Proposed Security engineering Methodology for Big Data Databases....207

6.3.1 Security Requirements Engineering....207

6.3.2 Security Design Engineering....215

6.4 Security Analysis of MongoDB....220

Summary....225

Novel Contribution of the Chapter....225

Chapter 7....226

Implementation....226

7.1 Overview....226

7.2 Security Requirements Elicitation Prioritization Specification....227

7.3 Security Design engineering....234

Summary....237

Chapter 8....238

Conclusions, Contributions and Future Scope...238

Contributions of the Thesis...242

Future Scope and Open Problems..245

References..246

Biography of Author...257

LIST OF FIGURES

Figure 1.1	Different types of Stakeholders	7
Figure 1.2	Classification of Security Requirements	12
Figure 2.1	Abuse Case for Actor Traveler in Railway Reservation System	37
Figure 2.2	Misuse Case for Railway Reservation System	39
Figure 2.3	Security Use Case for Railway Reservation System	40
Figure 2.4	Traveler Profile for Book Ticket Functionality	41
Figure 2.5	Refinement of Security Goals as Security Requirements	43
Figure 2.6	Attack Tree for Login Functionality	44
Figure 2.7	Intentional Anti Model for Railway Reservation System	45
Figure 2.8	Secure Tropos for Railway Reservation system	47
Figure 3.1	Security Engineering Framework	62
Figure 3.2	Security Requirements Engineering Process	63
Figure 3.3	Scenario using Sequence Diagram for ‘Login’ Functionality	67
Figure 3.4	Security Design Engineering Process	77
Figure 3.5	Categorization of Domain Constraints	80
Figure 3.6	Process of Security Testing	81
Figure 3.7	Test Scenario for ‘Login’ Functionality	82
Figure 3.8	Sample Test Report	84
Figure 3.9	Test Report for Content Management System	99
Figure 3.10	Vulnerabilities over the year for WordPress	100
Figure 3.11	Main Page of showing Dashboard	100
Figure 3.12	Theme Purchase for Repair on Website	101
Figure 3.13	Showing the price information for different users	101
Figure 3.14	Showing Services provided by Website	102

Figure 3.15	Another screen of website	102
Figure 3.16	Screen Showing the Items available with its price	103
Figure 3.17	Checkout Screen	103
Figure 3.18	Accunetix Scan Result Window	104
Figure 4.1	Cloud Reference Architecture	110
Figure 4.2	Proposed Framework for Cloud-Based Systems	120
Figure 4.3	Vulnerability over the years for ownCloud	151
Figure 5.1	Architecture of IoT	158
Figure 5.2	Proposed Framework for IoT-Based Systems	168
Figure 5.3	Process for Prioritization of Threats	178
Figure 5.4	Remote Patient Monitoring System	189
Figure 6.1	Security engineering Framework for Big Data Databases	208
Figure 6.2	Vulnerabilities over the years for MongoDB	223
Figure 7.1	Architecture of SREPS	227
Figure 7.2	Login Page of Tool	228
Figure 7.3	Window showing fields required for elicitation process	229
Figure 7.4	Elicited Security Requirements	229
Figure 7.5	Prioritized Security Requirements	230
Figure 7.6	Specified Security Requirements	230
Figure 7.7 (a)	Back end Architecture of SREPS	231
Figure 7.7 (b)	Back end Architecture of SREPS	231
Figure 7.8	Threats corresponding to selected Vulnerability	232
Figure 7.9	Vulnerability-Threat Mapping Repository	233
Figure 7.10	Repository of Threats affecting Assets	233
Figure 7.11(a)	Architecture of SDE	234

Figure 7.11(b)	Architecture of SDE	234
Figure 7.12	Repository of Security Mechanisms and Security Requirements Mapping	235
Figure 7.13	Selection of Domain Constraints	236
Figure 7.14	Suggested Security algorithm for Implementation	237
Figure 8.1	Generic Security Engineering Framework	240

LIST OF TABLES

Table 2.1	Summary of Contributions	56
Table 3.1	Vulnerability Mapping Table for Web-based System	70
Table 3.2	Criteria for Assigning Threat Rating	74
Table 3.3	Comparison of SRS	76
Table 3.4	Repository of Algorithm	78
Table 3.5	Functional and Non-Functional Requirements	86
Table 3.6	Vulnerabilities, Threats, Affected Assets and Security Requirements for CMS	87
Table 3.7	Threats and its Rating	92
Table 3.8	Assets Evaluation for CMS	93
Table 3.9	Risk Calculation and Security Requirements Prioritization	94
Table 3.10	Mapping of Security Requirements to Security Services	95
Table 3.11	Threat Analysis Repository	96
Table 3.12	Domain Constraint Analysis	97
Table 3.13	Design Template	97
Table 3.14	Threat Comparison	104
Table 4.1	Functionality-Asset Association for Cloud-based Systems	121
Table 4.2	Vulnerabilities-Threats Mapping for Cloud Systems	125
Table 4.3	Category of Threats Based on Risk Values	134
Table 4.4	Mapping of Security Requirements with Security Services	135
Table 4.5	Constraints for Cloud-Based System	137
Table 4.6	Functional Requirements for different Cloud Actors	139
Table 4.7	Vulnerabilities, Threats, Assets for Customers Functionalities	140

Table 4.8	Calculation of Security Requirements Priority	143
Table 4.9	Attack Analysis Repository for Cloud Systems	145
Table 4.10	Security Guidelines for Cloud-Based Systems	146
Table 4.11	Security Algorithm Employed by Cloud Storage Systems	148
Table 4.12	Threat Comparison	152
Table 5.1	Comparison of IoT Security and Network Security	160
Table 5.2	Assets at Various Layers of IoT	169
Table 5.3	Identified Vulnerabilities for Assets	173
Table 5.4	Vulnerability-Threats Mapping Table	174
Table 5.5	Threats and Affected Assets	179
Table 5.6	Mapping of Threats to Security Services and Mechanism	184
Table 5.7	Available Security Mechanisms	185
Table 5.8	Threats to Mechanism Mapping	186
Table 5.9	Constraints in Different Layers of IoT	187
Table 5.10	Involved Assets with their Roles	189
Table 5.11	Vulnerabilities and Threats for Assets	190
Table 5.12	Evaluation of Assets	192
Table 5.13	Calculation of Risk Value for Potential Threats	192
Table 5.14	Categorized Threats	194
Table 5.15	Elicited and Prioritized Security Requirements	196
Table 5.16	Recommended Security Techniques	197
Table 6.1	List of Direct Stakeholders	209
Table 6.2	Operations for Stakeholders	209
Table 6.3	Vulnerabilities for CRUD operation	210
Table 6.4	Vulnerabilities-Threats Mapping Tables for Big Data Databases	212

Table 6.5	Security Requirements Mapping to Threats	213
Table 6.6	Mapping of Security Services with Security Requirements	216
Table 6.7	Metric Showing the Constraints and Possible Values	217
Table 6.8	Source of Origin of Data Analysis	218
Table 6.9	Decision Template	219
Table 6.10	Guidelines for Stakeholders to Handle Security Properly	220
Table 6.11	Security Features in MongoDB	222
Table 6.12	Comparison of Threats	223

CHAPTER 1

INTRODUCTION

The present software development process follows conventional methodologies that induce security gaps while the software development is going through various phases. A solution must exist and amalgamated with the existing, software development processes to address such security vulnerabilities. The present chapter gives an overview of the current software development life cycle processes followed by the definitions and importance of Requirements Engineering, Requirements, Requirements Engineering Process, Design Engineering and Testing. Then, the need for security is presented, followed by an overview of Security Requirements and presents related work in the area of Security Engineering. Using this discussion, the problem statement for the thesis is captured.

1.1 Software Development Life Cycle

Software Development Life Cycle (SDLC) is a process used to develop and engineer high-quality software. High quality means software should meet customer expectations, completes within time and budget. SDLC consists of a detailed plan describing how to develop, maintain, change and modify or improve the specific software. A typical Software Development life cycle consists of the following phases:

Phase 1: Requirements Engineering: Requirements engineering refers to the process of gathering, analyzing, defining, documenting and maintaining software requirements from client. The goal of this phase is to develop and maintain refined and descriptive Software Requirements Specification (SRS) document.

Phase 2: Design: Objective of the design phase is to translate SRS into blueprint which can be used for Software Construction. It is a multi-step process that focuses on (i) Creation of Data Structure required to implement Software; (ii) Building Software Architecture which considers how the subsystems are making up the system and their relationship, modular framework of a Software; (iii) Interface Design that considers how software communicates with itself in each subsystem and how it interface with another subsystem; (iv) Procedure Detail which includes the procedural description of software components.

Phase 4: Building or Developing: In this phase, actual development starts and the product is built. The code is generated as per the decision was taken during the design phase. If the designing is done in a detailed and organized manner, code generation can be accomplished without much hassle.

Phase 5: Testing: It is usually a subset of all the phases. In the modern SDLC models, the testing activities are mostly involved in all the phases of SDLC. However, this phase refers to the testing of only those stages of the product where defects are reported, tracked, fixed and retested, until the product reaches the quality standards as defined in the SRS.

Various process models are available for software development some are linear such as waterfall model; others are iterative like the incremental model and spiral model. Among these models, Spiral model is very popular as it does the project planning and considers risk.

1.2 Requirements Engineering

Definitions

Definition 1: Requirements Engineering (RE) is defined by IEEE (IEEE Standard, 610.12, 1990) as "the systemic process of developing requirements through an iterative cooperative process of analyzing the problem, documenting the resulting observations in a variety of representation formats and checking the accuracy of understanding gained."

Definition 2: Requirements engineering defined in (Sommerville, 2004) as "The process of establishing services required by the customers from a system and constraints under which it operates and developed. The requirements themselves are the descriptions of the system services and constrictions that are generated during the requirements engineering process."

1.2.1 Requirements

Definition 1: A requirement as defined in (IEEE Standard, 610.12, 1990) is "(i) A condition or capability needed by a user to solve a problem or achieve an objective, (ii) A condition or capability that must be met or possessed by a system or system components to satisfy a contract, standard, specification or other formally imposed documents, (iii) A document representation of a condition as in (1) or in (2)."

Requirements thus arise from the user, general organization, standards, and government bodies. These requirements are then documented.

Definition 2: As defined by Davis (Hickey & Davis, 2003) "Requirements are high-level abstractions of the services the system shall provide and the constraints imposed on the system."

Types of Requirements:

Requirements have been classified as:

- Functional Requirements
- Non-Functional Requirements
- Domain Requirements

- **Functional requirements**

  Functional requirement as defined by IEEE (IEEE Standard, 610.12, 1990) "is a requirement that specifies a function that a system or system component must be able to perform."

  Functional requirements depend on the type of software, expected users and the type of system where the software is used.

- **Non-functional requirements**

  Non-functional Requirements are those that define system properties and constraints. For example, reliability, response time, storage requirements, input-output device capability, and system representations.

  "Non-functional requirements may be more critical than functional requirements. If these are not met, the system is useless."

  Non-functional requirements are further categorized into three classes:

  - **Product requirements**

    Requirements which specify that the delivered product must behave in a particular way e.g. performance requirements (execution speed), reliability, and portability.

  - **Organizational requirements**

    Requirements which are a consequence of organizational policies and procedures e.g. process standards used, implementation requirements, etc.

  - **External requirements**

    Requirements which arise from factors external to the system and its development process e.g. interoperability requirements, legislative (privacy, safety) requirements, etc.

- **Domain Requirements**

  Domain requirements are derived from the application domain and describe system characteristics and features that reflect the domain. Domain requirements can be new functional requirements, constraints on existing requirements or define specific computations. Again the problem with domain requirements is if they are not satisfied, the system may be unworkable.

Requirements play a vital role in software development; software requirements are a description of features and functionalities of the target system. Requirements convey the expectations of users from the software product. The requirements can be obvious or hidden, known or unknown, expected or unexpected from client's point of view.

1.2.2 Requirements Engineering Process

Requirements engineering is a difficult task, and any fault or error in this activity will lead to the development of the software systems that will either not work properly or may fail under some circumstances. Software systems developers realized that inadequate attention paid to requirements formulation is a major factor that leads to systems failure (Coman & Ronen, 2010) (Flyvbjerg & Budzierkdsjfk, 2011). Boehm and Papaccio (Boehm & Papaccio, 1988) said that the cost of correcting requirements errors is five times when done during the design phase, ten times during implementation phase, twenty times during testing and two hundred times after the system has been delivered. Thus, the process of requirements engineering should be done properly to develop a high quality and reliable system.

Requirements Engineering Process consist of following activities:

- ***Requirements Elicitation*****:** Operational capability needed from the system, the applicable constraints and the characteristics of the environment are identified. Additionally, it may bring out quality factors or risk management factors. The goal of this activity can be achieved by using traditional techniques such as ***interviews, brainstorming, introspection, questionnaires, Delphi technique, and FAST.*** But, traditional techniques suffer from certain disadvantages such as, the interaction between the requirements engineer and the stakeholder can be based on assumptions (Goguen & Linde, 1993), possibility of ambiguity in understanding the questions (Suchman & Jordan, 1990), and a tendency of over-analyzing may result in the system too constrained (Hickey & Davis, 2003). The disadvantages seen with traditional techniques are noticed when one moves to systems with increasing complexity (Lapouchnian, 2005). So, new techniques are developed such as ***viewpoint-oriented approach, Goal-oriented approaches, Agent-oriented approaches, Aspect-oriented approaches.*** The viewpoint approach for requirements elicitation which is used in the thesis is described in detail here, and others are omitted due to lack of space.

- **Viewpoint-Oriented Approach**

  In viewpoint-oriented approach (Kotonya & Sommerville, 1996) (Sommerville, 2004), a different class of stakeholders is identified. The requirements from these stakeholders are collected. They structure the requirement, group related requirement, and organize them into coherent requirements. A *Stakeholder* is a person or group who is affected by the system directly or indirectly. Different categories of actors are specified as shown in Figure 1.1.

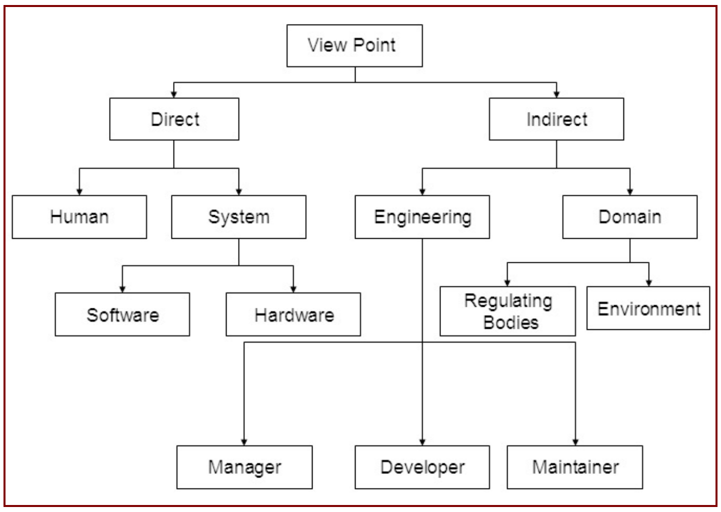


Figure 1.1 Different Types of Stakeholders (Kotonya & Sommerville, 1996)

  - **Direct Actor:** These actors directly interact with the system to receive services and may send control information and data to the system. For example, customer, manager, staff, and database operator.

  - **Indirect Actor:** These actors have an interest in system services, but they do not interact directly with it. For example, security staff, marketing manager, hardware and software maintenance engineer.
  - **Domain Actor:** These actors deal with the domain characteristics and constraints which influence the requirements. For example, Regulatory bodies, people who make standards, etc.

- ***Requirements Analysis:*** Requirements Analysis is the process of analyzing customers' needs to acquire a complete understanding of the system to be developed. During requirements analysis following points are checked:
  - **Domain:** Analyze application domain for different requirements.
  - **Conflict:** Finding out and resolving various conflicts when several stakeholders are associated with a requirement.
  - **Completeness:** Are all functions required by the customer included?
  - **Validity:** Does the system provide the functions which best support the client's needs?
  - **Realism:** Can the requirements be implemented in given available budget and technology?
  - **Conformance to standards:** Review the system as per security standards.

- ***Definition and Specifications*:** Requirements are documented for use in the subsequent development process. Formal specification languages, knowledge representation, are used to document requirements. Any inconsistencies, missing requirements found during the validation phase can also be fed back into this task. From the Specification and documentation task, one can go to the negotiation task (e.g. if conflicting requirements were found) or elicitation task (if more information is required).

- ***Requirements Validation*****:** Here requirements document is reviewed by the project manager to ensure, system provide the functions which best support customer needs.

1.3 Design Engineering

Definition 1: A Design is defined in (IEEE Standard, 610.12, 1990) as “The process of defining the architecture, components, interfaces, and other characteristics of a system or component.”

It focuses on four distinct attributes of software that are Data Structure, Software Architecture, Interface Design and Procedure Details. It is the basis of detailed implementation; here important decisions are made which affect the quality of software such as minimize the coupling, maximize the cohesion based on domain/ environment. It is a multilevel process where the system is designed from high-level view and progressively refined into the more detailed design.

To develop a complete design specification, four design models are needed. These models are listed below.

- **Data Design:** Data structures are specified for implementing the software by converting data objects and their relationships which are identified during the analysis phase.

- **Architectural Design:** The relationship between the structural elements of the software, architectural styles, design patterns, and the factors affecting the way in which architecture can be implemented are specified.
- **Component-level Design:** Provides the detailed description of how structural elements of software will be implemented.
- **Interface Design:** Depicts how the software communicates with the end-users' and the system that interoperates with it.

1.4 Testing

Definition 1: Software testing is a process of executing a program or application with the intent of finding the software bugs. It can also be stated as the procedure of validating and verifying that a software program or application meets the business and technical requirements that guided its design and development.

Goals of testing are (1) Detect faults; (2) Establish confidence in software; (3) Evaluate properties of software such as Reliability, Performance, Memory Usage, Security, and Usability.

Activities in Testing process consists of Test Planning, Test Case design, Running the Test Case, and Evaluating the Test Result. The testing process is also known as levels of testing that are:

- **Unit Testing.** Individual units of software are tested to find the differences between specified units and their implementation. Test cases designed for the interface, local data structures, independent paths, boundary conditions, and error handling.

- **Integration Testing.** After testing the modules individually, modules are integrated as per the integration plan, and the partially integrated system is tested at each integration step. The objective here is to expose problems arising from the combination.
- **System Testing.** It focuses on the testing of product as a whole by verifying its working with the original requirements specification document. Also, non-functional requirements such as performance, usability, security are tested.
- **Acceptance Testing.** Acceptance Testing is used when the software is developed for a specific customer. A series of tests are conducted to enable the customer to validate all requirements. These tests are managed by the end user /client and may range from ad-hoc tests to well-planned series of tests. The objective of this testing is to ensure that end users are satisfied.

1.5 Security

Security of Software systems is defined as technological and managerial procedures applied to computer systems to ensure CIA (Confidentiality, Integrity, and Availability) of system resources.

Security is gaining much popularity in recent years because of increased penetration of ubiquitous devices in software systems. Almost every real-world computation and transaction has become dependent on these devices. These devices are prone to get infected by intruders, virus, malicious crackers and other threats because they are handling security in an ad-hoc manner. Damage caused to system assets is not affordable, as they are carrying business and mission critical data. Also, the need to focus on security has become a priority task because of a sudden rise in the security attacks on organizational assets is reported in various reports (Department for Business Innovation & Skills, 2014) (Verizon, 2014) (Symantec, 2013). Also, new domains such as Cloud Computing, Internet of Things, and Big Data have emerged to connect people by providing a vast range of services. However, people are not using them because these new domains are vulnerable to security attacks.

Traditionally, security in software systems is handled during the design phase of the development process, which may cause performance related issues or sometimes it may cause failure of the system. To overcome these difficulties, software engineering community suggests that the security requirements should be elicited and then implemented to ensure the better implementation of security. It has led to the emergence of a new dimension in the classification of requirements known as **Security Requirements** as shown in Figure 1.2.

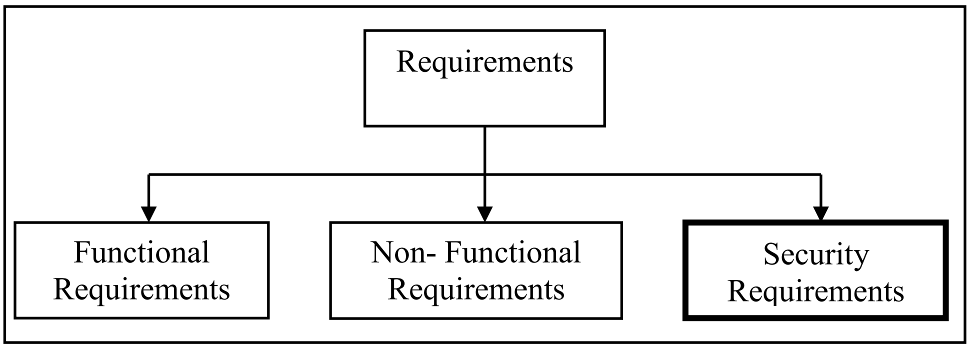


Figure 1.2 Classification of Security Requirements

1.6 Security Requirements

Definition: Security requirements are defined in (Firesmith, Engineering Security Requirements, 2003) as "high-level requirements that give a detailed specification of any system behavior that is not acceptable."

Initially, security requirements are considered to be part of non-functional requirements. However, due to increased use of ubiquitous devices attacks have grown multifold. Therefore, security requirements need to be addressed adequately. Also, Security Requirements should be differentiated from architectural constraints which are the security techniques to eliminate threats such as encryption, access control, authentication, etc.

Security requirements are related to functional requirements and are different from architectural constraints:

- Security requirements are related to functional requirements, as it represents the threats on assets which are used by functional requirements.
- Security requirements are different from architectural constraints which represent the protection measures implemented to mitigate the threats. These architectural constraints should be optimally specified; otherwise, it can make the system unnecessarily constrained and insecure.

Different types of security requirements proposed by researcher Firesmith (Firesmith, Engineering Security Requirements, 2003) are:

- **Identification Security Requirement**

  Identification security requirement signifies the extent to which software systems should identify its external entities before allowing them to interact or use its services.

For Example

- The software system shall identify all the stakeholders, external application beforehand, who will either interact with the system or with whom system will interact.
- Architectural constraint to implement the requirement: ‘Users are provided with user identifier and password or any digital possession.’

- **Authentication Security Requirement**

  Authentication security requirement signifies that software systems should verify the identity of its externals. Verification is done to have confidence that externals are actually who or what they claim to be.

  For Example

  - The software system shall verify the identify all the stakeholders who will interact with the system beforehand.
  - The application shall verify the identity of all the external software or application that are going to use them, or they are using it.
  - The application shall verify the identity of each entity who are going to do any addition/ modification/ deletion to either their information or system capabilities.
  - Architectural constraint to implement the requirement: ‘User shall be authenticated by his name and social security number.’ or ‘The system shall verify the user password twice before allowing it to use the system.’

- **Authorization Security Requirement**

  Authorization security requirement signifies that only authenticated externals can access software system only if they have been explicitly authorized to do so by the administrator of the application.

  For Example

  - Access right to use or update system capabilities shall be defined.
  - A user can have access to his data only not of others.
  - Corresponding architectural constraints to implementing the requirement: ‘System shall maintain authorization database or commercial intrusion prevention system to prevent infection by intruders.’

- **Immunity Security Requirement**

  Immunity security requirement represents that software system shall protect itself from infection by illegal, undesirable programs (e.g., computer viruses, worms, and Trojans).

  For Example

  - Software application shall protect itself from infection by computer viruses, Trojan horses, worms and other related harmful programs.
  - Architectural constraints to implementing the requirements: ‘the use of commercial antivirus programs, Firewall’; ‘Prohibition of type-unsafe languages such as C that allow buffer overflow’; ‘Use of Programming standards.’

- **Integrity Security Requirement**

  Integrity security requirement represents that any data in software system does not get corrupted via unauthorized creation, deletion, and modification.

  For Example

  - Software system shall prevent unauthorized corruption of data (email attachment, data from external sources or present in the data center) in the system.
  - Integrity Requirement can be implemented: 'Using Cryptographic Techniques or Hash Functions.'

- **Intrusion Detection Security Requirements**

  Intrusion detection security requirement represents that if the intruders have attacked the software system, then it should be detected and recorded so that the administrator can take necessary action.

  For Example

  - The system shall find and record all failed attempts to the identification, authentication, and authorization requirements.
  - The architectural constraint to implement requirements: 'The use of Alarms, event reporting, intrusion detection system (IDS) and Intrusion Prevention System (IPS).'

- **Non-repudiation Security Requirements**

  Non-repudiation security requirement represents that either the external should not deny after interacting (e.g. message, transaction) with all or part of the system.

For Example

- The system shall keep the complete time-based record of all the activities encountered by the system.
- The architectural constraint for implementation: 'The use Digital Signature (to identify the parties), Timestamps (to capture dates and times).'

- **Privacy Security Requirements**

  Privacy security requirement represents that the software system should keep its data and communications private from unauthorized individuals and programs.

  For Example

  - The software system shall keep a record of its user information, communication and stored data from unauthorized individuals.
  - Architectural constraints for implementation: 'The use of public or private key cryptography techniques.'

- **Security Auditing Security Requirements**

  Security auditing security requirement represents that an application shall enable security personnel to audit the status and use of its security mechanisms.

  For Example

  - The system shall collect, organize and summarize the status of security requirements and make report regularly.
  - The architectural constraint to implement the requirements: 'The use of event log and Audit trails.'

- **Survivability Security Requirements**

  Survivability security requirement represents that an application should work possibly in degraded mode even if some destruction has occurred in the application.

  For Example

  - Some backup mechanism shall be deployed which can be used in case of failure.
  - The architectural constraint to implement the requirement: 'Use of Hardware redundancy, data center redundancy, and failover software.'

- **System Maintenance Security Requirements**

  System Maintenance security requirement represents that how the authorized modifications in the system can be done without affecting the deployed security mechanism.

  For Examples

  - The system shall not affect its security requirements because of any upgrade in data, software or hardware components.
  - The architectural constraint for implementation: 'The use of maintenance and enhancement procedures, training program related to system maintenance.'

- **Physical Protection Security Requirements**

  Physical Protection security requirement represents how an application shall protect itself from substantial damage.

For Example

- Protection of hardware components from physical damage, theft, and destruction shall be provided.
- Architectural constraints to implementing the requirements: 'Deploying locks, security guards at entry points.'

1.7 Related Work

Various security methodologies were present in the literature for handling security issues of software systems. These methodologies are categorized into three classes namely use case-based approaches, goal-oriented approaches, and process-oriented approaches.

Use case based security methodologies are the extension of Use Case diagrams. Methodologies under this class are used to elicit threats and broadly suggests constraints to mitigate threats. Some methodologies in this category are abuse cases, misuse cases, common criteria, security use cases. Abuse cases (McDermott & Fox, 1999) represents the malicious functionality (threats) executed by unintended actors in the system. Misuse Cases (Alexander, 2003) (Sindre & Opdahl, 2005) models the threats and specifies some use cases to prevent and detect the threats occurring in the system. Common Criteria (Ware, Bowles, & Eastman, 2006) identifies threats applicable to functionality and suggests security mechanism to mitigate threats. Security Use Cases (Firesmith, Security Use Cases, 2003) identifies threats/ attacks associated with use-cases and represent it as misuse case. Further, Security Use Cases such as ensure privacy, control access, ensure integrity, ensure repudiation are specified to protect assets from security threats. **These methodologies focus on identification of threats and help in elicitation of security requirements.**

Goal-oriented methodologies represent the attacks or requirements as a goal. **Goals** are intended behavior and perspective statement of intent about the system. ***Security Goals*** are the abstract representation of security concerns to the assets of the system. For example, 'Privacy' security goal signifies that asset 'Patient Health Information' should remain private and confidential from misusers. Security goals are classified as Integrity, Confidentiality, and Availability goals (Lamsweerde, 2004). Security goal defines very general statement about the security of assets. For instance, the patient in the Hospital system has Confidentiality goal which expresses 'his health parameters should remain confidential.' Similarly, Integrity goal by Hospital Authority conveys that 'malicious actors should not change doctor assigned to the patient.' ***Security Requirements*** are a detailed description of security concern, or they give detailed statement about system behavior that is not acceptable. Security goals are refined as security requirements. For example, Security Goal Confidentiality can be refined as Identification, Authentication, and Authorization Security Requirements. Some Methodologies based on goal-oriented approach are attack trees, intentional anti-model, secure tropos. Attack Trees (Ellison, 2005) models the attack scenario in the form of a tree where root represents the attacker goal, and leafs shows the vulnerable point of attacks. The goal of the attacker is decomposed until the leaf node representing the exact vulnerable point where an attack can occur is reached. Intentional Anti-model (Lamsweerde, 2004), models the scenario of a threat to security goals at the application layer. Threats are represented as antigoals such as, 'achieve' is an antigoal showing 'thief comes to know payment medium of the client.'

This antigoal is the negation of relevant instance of the goal, ‘avoid’ is a goal which expresses ‘Account Number and PIN is unknown to Unauthorized Agents’ of the system. They refine the antigoals by building attack trees to get the detailed vulnerabilities. They end up specifying the security measures to mitigate antigoals. For instance, ‘put a limitation on a number of PIN entries attempt’ to avoid exhaustive PIN search by a thief. Secure tropos methodology (Mouratidis H., 2002), is an extension of Tropos methodology. They define the security goals and identify the threats associated with it. During the early requirements phase, security goal classified as privacy, safety, accountability, availability, and integrity of the system are identified. Afterward, during the late requirements phase, threats/ attacks to security goals are explored. For example, spoofing, man-in-the-middle attacks are possible to security goal ‘privacy.’ Finally, security mechanisms like encryption and decryption are listed to achieve the security goal privacy.

The goal-oriented methodologies define security goals and origin of security goals which are attacks and vulnerabilities. They end up specifying possible broad measures to achieve the security goal, without specifying the security requirements.

Recently, Process-Oriented approaches have emerged which deal with identification, analysis, and prioritization of security requirements. Some methodologies under this category are SQUARE, SREP, a framework by Haley et al. and a proposal by Fabian et al. Software Quality Requirements Engineering (SQUARE) (Mead, 2005) process starts with the identification of security goals such as Confidentiality, Accuracy, and Integrity of the system. Next, the goals are analyzed to get vulnerable points and threats using diagrams like the architectural diagram, use cases, and misuse cases. Afterward, based on the identified threats, security requirements are defined in the form of constraints. For example, ‘The system is required to have strong authentication measure in place at all system gateway/ entrance points.’ After that, security requirement is categorized as the system level, software level, etc. Finally, prioritization is done based on categorized requirements and risk assessment results using methods such as Triage, Win-Win, etc. Security Requirements Engineering Process (SREP) (Mellado, Medina, & Piattini, 2007) uses common criteria approach to identify the security requirements. A Security Resource Repository (SRR) of threat, vulnerability and related information corresponding to the assets from existing systems are maintained. SRR is used for identifying the vulnerable points and threats of critical assets of the current system. If the asset is not present in SRR, then threats and vulnerable points are modeled using some use case based technique such as misuse case, abuse case, etc. Information related to the new asset is added to the SRR. After identification of threats, risk assessment is done, and security requirements are defined and prioritized. In the framework by Haley et. al (Haley, Laney, Moffett, & Nuseibeh, 2008) for security requirements elicitation process is as follows: First, the functional requirements are defined. Secondly, security goals to protect assets associated with functionality are defined. Lastly, security measures are specified in the form of constraints to functional requirements for achieving the security goals, such as the system shall ***provide personal information*** **(function)** ***only to*** **(constraint)** members of HR department. Fabian et al. (Fabian, Gurses, Heisel, Santen, & Schmidt, 2010) have also presented a framework for elicitation and analysis of security requirements. Here, first, different stakeholders are identified. After that, functional requirements with corresponding assets and non-functional requirements associated with stakeholders are identified. Security goals are defined, which are further refined as security requirements based on the assets, its misuser and its context of use. Once the security requirements are defined at stakeholder level, they are compiled at the system level to avoid the conflict because of a different viewpoint. Finally, based on risk analysis results they suggest broad security mechanisms to implement the security requirements.

Thus, process-oriented approaches focus on defining security goals, derives vulnerability/ threats which are the cause of security goals. They end up specifying security requirements in the form of architectural constraint, which are mainly security measures to mitigate the threats. Also, prioritization of security goals based on the risk associated with threats in the system is done. None of these proposals explicitly specifies the security requirements.

Thus, all the above methodologies focus on the identification of security goals, originating from threats, attack, vulnerability to the asset. They specify the security requirements in the form of protection measures which can deploy the security goals. **They are not specifying the security requirements which are a refinement of security goals such as identification, authentication, authorization, non-repudiation, etc. defined by Firesmith (Firesmith, Engineering Security Requirements, 2003). Further, they do not recommend any particular security mechanism to mitigate the threats associated with the security goals. In addition, they do not consider domain constraints while suggesting security algorithms. Also, none of them does the testing of system security level.**

1.8 *Synthesizing the "Problem Statement"*

As implementation of security in a software system is a complex task, therefore, an engineering approach is necessary for efficient implementation of security during the software development process, where first the security requirements are elicited, analyzed, prioritized, and specified. Based on various constraints an efficient algorithm for implementation should be chosen, and finally, the system security level should be validated. Consequently, a new field of Security Engineering has emerged, which is concerned with the development of processes and methods for implementing security in software systems. In other words, it is an engineering approach that identifies the security requirements and provides/ suggest the algorithms to implement them efficiently (Chatterjee, Gupta, & De, 2013). Ideally, it should consist of three phases namely security requirements engineering, security design engineering, and security testing. These phases should be tightly coupled with the three phases of traditional development approach. During Requirements Engineering phase, the Security Requirements should be elicited based on threats/ attacks on the asset related to the functionality of the system, after that these requirements may be analyzed for consistency, correctness and then prioritized. Finally, they should be specified explicitly. During the Design Phase, for efficient implementation of the security requirements, security measures/ algorithms should be suggested based on the various applicable constraints such as memory, encryption speed, complexity, power, etc. Finally, in Testing phase, the system should be evaluated for security goals.

From the preceding section, it is drawn that:

- None of the present proposals include all the activities of requirements engineering that is elicitation, analysis, specification, and prioritization. Use case approach focuses on elicitation of threats; they do not analyze, prioritize or specify the security requirements. The goal-oriented approach identifies the security goals, threats, vulnerable point and security requirements in the form of security measures such as encryption, password, etc. They do not specify, analyze and prioritize the requirements or goals. Process-oriented approaches attempt to address the different task of the security requirements engineering, but none of them explicitly specifies the security requirements. They express the security requirements as a constraint to achieving the security goals. In addition to this some of them like SREP, SQUARE do not associate the security requirements with the functional requirements. Even, Haley et. al does not prioritize the security requirement. Though Fabian et al. associate security goals with functional and non-functional requirements, but they specify the security requirements as an architectural constraint. As it is mentioned in section 1.6 that security requirements are different from architectural constraints. Therefore, security requirements which are a refinement of the security goals should be explicitly specified before specifying the security measures.

As described in Section 1.6 Security Requirements are not independent of functional and non-functional requirements. Therefore, they should be elicited, analyzed and prioritized along with the system requirements. In this way, Security Requirements Engineering process can be embedded into traditional Requirements Engineering process.

- Current proposals specify broad protection measures to implement security requirements. They should consider constraints such as environment, memory, power, computation speed, etc. while choosing the security mechanism such as cryptography algorithms for attaining the security goals like confidentiality, privacy, authentication, etc. Selecting an algorithm without considering the domain parameters may result in unnecessary constraints to the system. Therefore, optimal security techniques/ algorithms must be identified during the design phase by considering domain constraints.

- None of the approaches validates security goals at the end. Analogous to the traditional software engineering the system must be validated for the embedded security.

- Most of the methodologies focus on web-based systems; they do not consider the emerging domains like Cloud Computing, Internet of Things (IoT), Big Data Databases. Cloud has specific security issues such as Shared Environment, Multi-location Data placement, and Trust. Similarly, IoT which also has additional security issues such as Data Freshness, Trust, and Liability. Also, big data databases have various constraints such as data is unstructured, the rate of data generation is very high, etc. The current proposal in these areas is suggesting the security algorithms for implementation of threats without considering different domain constraints. Hence, it calls for a generic methodology which can be adapted to these new emerging areas.

Hence, the problem statement of the thesis is:

Develop a Security Engineering framework that focuses on the development of Secure Software Systems, which can be integrated into conventional software development process.

This problem statement consists of the following research goals:

i) Security Requirements Engineering Framework

The first research goal is to extend the traditional requirements engineering process of SDLC to include the elicitation of security requirements. Also, like other functional and non-functional requirements, they should be analyzed for consistency, correctness, and completeness. Since all requirements cannot be implemented due to the budget and time constraint, therefore, they should be prioritized. **Hence, the first research goal is to develop "Security Requirements Engineering process" which can become an integral part of traditional Requirements Engineering Process. Thus, it should have activities of identification, analysis, prioritization, and specification of security requirements along with the functional and non-functional requirements.**

ii) Security Design Engineering Framework

During the design phase of any traditional SDLC, the objective is to divide the system into an optimal number of components/ sub-modules for meeting the principle of minimizing the coupling and maximizing the cohesion. Also, decide the algorithms for each component based on the applicable domain constraints so that system can correctly and efficiently implement all the functions. Therefore, design engineering needs to be extended for incorporating the selection of cryptography algorithms to implement the security requirements efficiently. **Hence, the second research goal of the thesis is to develop "Security Design Engineering process" that will choose the appropriate security algorithms to attain security goals based on different domain constraints.**

iii) Security Testing

Finally, it is essential to validate the security of the system. If a metric can be defined that would measure the level of security embedded in the system, it would assure the developer and users to take further necessary steps while developing or using the system. **Hence, the third research goal of the thesis is to develop "Security Testing process" that will evaluate the security of the system by generating a metric.**

iv) Application of Security Engineering to Cloud Systems

In the recent past, cloud system has become a popular media for information sharing. Besides many issues in successful cloud systems, security is a big issue as stakeholders are sharing information, infrastructure, and software. Compared to web-based systems there are new challenges like trust, multi-location data placement, etc. in the cloud system. Existing proposals did not consider the new security issues and defined the security measures without considering the domain constraints. **Hence, the fourth research goal of the thesis is to "adapt Security Engineering Framework for Cloud-based System" that should address new security issues related to the cloud-based system and suggests efficient algorithms pertaining to the domain constraints and finally, evaluate the system security.**

v) Application of Security Engineering to Internet of Things (IoT)

Recently Internet of Things (IoT) has gained popularity with increased deployment of IoT devices in various domains such as e-Health, e-Home, and e-Commerce. Security issues like Data Freshness, Liability, Trust are left unexplored in previous studies. Also, compared to network security, providing security to the IoT-based network is difficult because devices in IoT system have constraints. Again existing proposals do not consider new security issues and domain constraints pertaining to IoT systems. **Hence, the fifth research goal of the thesis is to "adapt Security Engineering Framework for the development of Secure IoT-based Systems." This process should address new security issues, consider domain constraints during the design phase and finally, test the deployed security.**

vi) Application of Security Engineering to Big Data Databases

Researchers have been focusing on security in big data databases because of the rise in the use of social networking, data analytics, etc. Data generated from these sources has volume, velocity, variety, has no defined structure, etc. These characteristics make it difficult to incorporate security into the system. **Hence, the sixth research goal of the thesis is to "adapt our Generic Framework of Security Engineering for Big Data Databases."**

1.9 Proposed Solution

The above sub-problems are handled by developing a three-phase framework of Security Engineering, that becomes an integral part of any software development process. Phases of proposed framework are not independent but are related to each other, and each of these works towards guiding research to its goal. Also, the proposed framework is applied to new domains such as Cloud Computing, Internet of Things and Big Data Databases.

i) **Development of Security Requirements Engineering Phase.** The thesis attempts to establish/ propose a methodology, which consists of different activities starting from identification of stakeholders to specification of the security requirements. In this phase different stakeholders are identified using viewpoint-oriented approach (Sommerville, 2004) (Kotonya & Sommerville, 1996), then functional requirements of each direct stakeholder are conceptualized. Based on the functionality execution sequence, various vulnerable points are identified with potential threats. These identified threats are now specified in terms of security requirements as defined by researcher Firesmith (Firesmith, Engineering Security Requirements, 2003). Then, the specified security requirements are analyzed and prioritized to get a concrete set of requirements which represents the overall scenario of security in the system.

ii) **Development of Security Design Engineering Phase.** In the process of developing a Security Engineering Framework, the thesis attempts to modify the existing design process by incorporating activities for selecting/ suggesting the Security Algorithm for implementation of Security Requirements. Based on the identified and prioritized Security Requirements, Security Mechanisms (Cryptography Algorithm) are selected, to protect the system assets. The process starts with the mapping of prioritized Security Requirements with the defined Security Services; this mapping would eventually help in implementing the system security. Now among the available security algorithms for implementing different security services, set of best algorithms are chosen based on: (1) number of threats they mitigate (to achieve this repository of attack mitigated by different algorithm is maintained) and (2) consideration of domain constraints (communicational constraints such as bandwidth, energy, etc.; computational constraints such as memory, encryption speed, etc. and type of device (Low-end or High-end)).

iii) **Development of Security Testing Phase.** Based on the necessity to evaluate the system security goals an attempt is made to measure the effectiveness of selected Security Algorithm. Security Testing is done to validate if the selected Security Algorithms can mitigate the potential threats to the system. A metric is generated showing the effectiveness of selected security algorithm. The metric would help in making a decision whether the selected algorithms are sufficient to achieve the security goals or revision is required. If the revision is needed, it backtracks to the design phase and select another applicable algorithm or modifies the applicable algorithms to mitigate the live threats. Also, this metric evaluates or measure the security of developed systems.

iv) **Application of Security Engineering to Emerging Domains**

Based on the need for handling security issues in emerging domains, we have adopted our generic proposal for different software systems namely Cloud Systems, IoT, and Big Data Databases. Therefore, the proposal of Security Engineering is modified to make it adaptable for various domains:

a) **Cloud Computing.** After analysis of cloud architecture provided by NIST (Liu, et al., 2011), thesis finds that security issues in cloud system are complex due to the presence of numerous functionality and assets. To deal with the situation, various repositories functionality-assets, vulnerabilities-threats mapping are developed that would help in identification of assets, vulnerability, and threats. Again during design phase, different domain constraints applicable to cloud systems are drawn, and a suitable algorithm is chosen. Our approach is illustrated for cloud-based storage system such as Dropbox, Mega, etc. For these cloud systems, it is illustrated that deployed security algorithms are not efficient pertaining to domain constraints.

b) **IoT.** With the automation of various sectors, a shift to IoT systems has been made. IoT systems have layered architecture, and each layer has its own complexity which gives rise to various security issues. To deal with these problems our Generic Framework for Security Engineering is adapted where repositories of assets at each layer with its applicable vulnerabilities and threats have been developed. The repository will help in the identification and handling of security requirements. The Security Engineering framework is applied to the IoT-based healthcare system. Further, it is drawn that existing security algorithms are not suitable. Hence, new security algorithms are suggested.

c) **Big data Databases.** With the tremendous increase in the use of social networking, e-commerce and related domains need to handle a huge amount of data generated from these sources arise. Also, data generated from these sources may not be structured, the speed of data generation is very high, etc. issues are present. To cater the above needs of data handling and processing NoSQL databases (a Big Data Database) have evolved. Hence, to handle the issues in a structured manner, thesis adapted the generic Security Engineering Framework for NoSQL Databases. The process steps have been applied to MongoDB a NoSQL database.

1.10 Outline of the Thesis

In **Chapter 2**, we present a *state-of-art review* of various methodologies available for handling security issues in software development. All the existing methodologies are critically evaluated along with the various parameters like Security Engineering activities covered (Security Requirements Engineering, Security Design Engineering, and Security Testing), and application domain. The chapter would serve as a backbone for our proposal of security engineering and the rest of the chapters.

In **Chapter 3**, we discuss our proposal for Security Engineering. The chapter starts with the elaboration of Security Requirements Engineering phase of our framework and then Security Design Engineering phase. Finally, we come to Security Testing phase. We will highlight each activity of our proposal in detail with the help of examples.

In **Chapter 4**, we have adapted our generic security engineering framework for Cloud Computing. The chapter starts with the brief introduction of cloud computing, then an overview of existing work available in the literature is provided. After that, elaboration of our proposed framework for cloud systems with highlights of modifications required in our generic framework to make it adaptable for cloud systems is presented.

In **Chapter 5**, we have adapted our generic framework of security engineering for Internet of Things (IoT). Chapter starts with the brief introduction of IoT, previous work available in the area of security is discussed. Then, adaption of our proposed framework for IoT system is presented. Also, we will highlight modifications required in our generic proposal to make it adaptable for IoT systems.

In **Chapter 6**, we have adapted our framework for Big Data Databases. We start with the brief introduction of big databases, a brief overview of existing work, then adaptation of our proposed framework for Big Data Database is discussed. Also, we will highlight modifications required in our basic proposal to make it adaptable for big data databases.

In **Chapter 7,** the implementation of a tool to help the developer is presented. The approach is based on the models and techniques discussed in Chapters 3, 4, 5 and 6. Explanation of working on the different components of the tools is described. To get a feel of the tools several screenshots have also been included.

In **Chapter 8,** we summarize the contribution of the thesis, present the conclusions and discuss the future scope of our work.

References: This section gives the references details of the thesis.

CHAPTER 2

LITERATURE SURVEY

Present chapter provides insight of security engineering methodologies available in the literature. Firstly, the overview of Use case-based methodologies is provided, followed by a discussion on Goal-oriented approaches, after that Process-oriented approach is discussed. Finally, each of the discussed methodologies is evaluated.

2.1 Overview

Security is gaining more attention because of a surge in security attacks. Therefore, handling of security attacks along with the activities of information system development process becomes an indeed activity. Various methodologies were present in the literature for handling security such as Secure Tropos (Mouratidis H., 2002), SQUARE (Mead, 2005), SREP (Mellado, Medina, & Piattini, 2007), Haley et.al framework (Haley, Laney, Moffett, & Nuseibeh, 2008) are to name few. Available methodologies are categorized according to the type of development approach they follow. Here three class of development approaches are taken:

- Use Case-based Approach
- Goal-Oriented Approach
- Process-Oriented Approach

Methodologies available in the literature has their procedure to handle security issues. Some are having detailed steps for elicitation, analysis, and prioritization of security requirements. Few of them suggest broad security measures, and some methodologies only model the security threats.

In the forthcoming sections, each of the mentioned approaches is explained with methodologies that fall in particular class. Moreover, each of the methodologies would be elaborated and explained with a common case study of the Railway Reservation System.

2.2 Use Case Based Approach

The term ‘use cases’ were first introduced by Ivar Jacobson in object-oriented development methodology, later Larry Constantine and others have used use cases for requirements analysis and design. In a use case diagram, use cases describe a circumstance in which a user interacts with the system to accomplish some task. Use cases are extended to elicit security threats and sometimes broadly suggest mechanisms to mitigate threats. It mainly helps in threat modeling and risk analysis. Various Security Engineering Methodologies are available in the literature based on use case-based approach. Some of them are Abuse Cases (McDermott & Fox, 1999), Misuse Cases (Alexander, 2003) (Sindre & Opdahl, 2005), Security Use Cases (Firesmith, Security Use Cases, 2003), and Common Criteria (Ware, Bowles, & Eastman, 2006). Each of the mentioned methodologies is now discussed below in detail:

(i) Abuse Cases –Abuse case is an extension of ‘Use Cases,’ Abuse Cases are explained using the same notations as used for use cases. Proper labeling of diagrams distinguishes the two models. An Abuse Case gives complete knowledge of the interaction between a system and actors, where the result of the interaction is harmful to system resources (assets). The abuse case provides a description of each actor with what is required and what is not required (threats) from the system. The actor's description consists of required resources (assets), skills and objectives. Abuse Cases are drawn by having total control of the goal machine and modifying the system software. A brief note of the harm caused because of the attacker is also included.

Use Cases are used to identify and represent the functional requirements of actors, whereas Abuse Cases are used to represent and identify the possible harmful interaction (threats) present in the system due to malicious actors. It also includes the actor's descriptions which help in security analysis. Abuse case approach has following steps:

a) Identify Malicious actors
b) Identify functional (harmful) requirements of malicious actors
c) Include short description of harm with malicious actor descriptions

Figure 2.1 shows an abuse case for the Railway Reservation System, where 'Malicious Traveler' is the abuser (attacker) and the oval depicts the threats that can be made by the attacker.

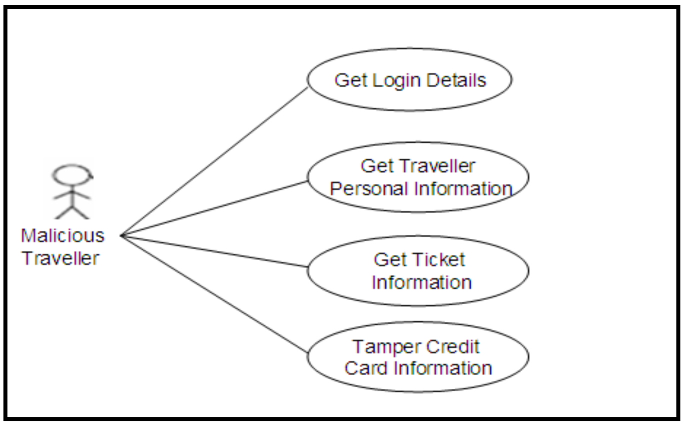

Malicious Traveller
Get Login Details
Get Traveller Personal Information
Get Ticket Information
Tamper Credit Card Information

Figure 2.1 Abuse Case for Actor Traveler in Railway Reservation System

Analysis: It is a traditional approach which depicts the malicious actors and threats executed by them. They do not specify security requirements and security mechanism to mitigate the threats. Only a short description of the attacker and abuse cases which includes their skills and resources is added.

(ii)Misuse Cases –Misuse Cases are also derived from Use Case diagrams. The Use Case diagram is an effective tool for eliciting and describing functional requirements of system-to-be, whereas Misuse Cases captures the threats based on the behavior of the system that is undesirable. The Use Cases are derived from goals of the system, on the other hand, Misuse Case are derived from system threats. Misuse Case approach has following steps:

a) Identify the actors and their functional requirements (use cases).
b) Identify the misusers and their functionalities (misuse cases); misuse cases are black in color to distinguish it from use cases.
c) Identify the new use cases which are related to existing use cases with ‘include’ relationship and misuse cases using two new relations ‘Detect’ or ‘Prevent.’ New relations will show whether a new use case can detect or prevent the activation of misuse cases respectively.
d) New use-cases are documented as security requirements.

Figure 2.2 shows a Misuse Case for the Railway Reservation System, where white color ovals represent the functionalities executed by the intended user and the black color ovals represent the threats that can be executed by attackers (misuser).

Analysis: Misuse cases models the broad security measures in the form of use cases. These use cases are related to the functional use cases with include relation and misuse cases with the relation detect or prevent. They do not identify the security requirements but identifies the threats on functionality that are then used to propose security measures in an ad-hoc manner. The new use case introduced in step (c) such as 'encrypt,' represents an ad-hoc mechanism to mitigate threats.

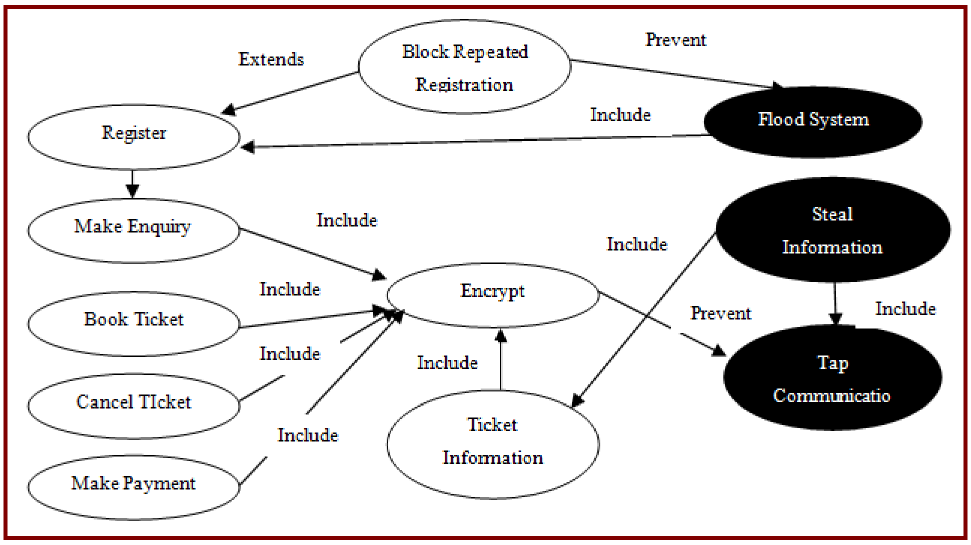


Figure 2.2 Misuse Case for Railway Reservation System

(iii)Security Use Cases – Security Use Cases are proposed by researcher Firesmith; which are used for analysis and specification of security requirements. It protects the application from security threats by eliciting the security requirements for implementation. It has following steps:

a) Identify the Actor and their use cases
b) Identify the misusers and functionality (misuse cases) executed by them.
c) Identify the security use cases required to protect the system from misuse cases/ threats.

Figure 2.3 shows a Security Use Case for the Railway Reservation System, which represents the functionalities executed by the actor 'Traveler,' threats executed by the misusers as misuse cases and the security use cases showing how the functionalities can be protected from threats.

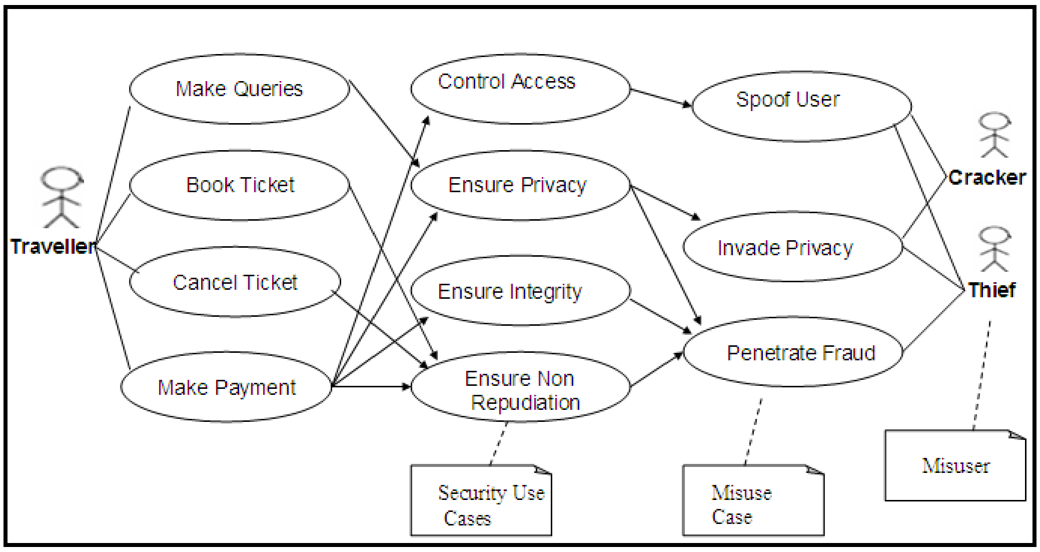


Figure 2.3 Security Use Case for Railway Reservation System

Analysis: It depicts the security requirements in the form of use cases and relates it to functional use cases and misuse cases. The process of identification of security requirements is systematic, but it does not analyze or prioritize the requirements. It is assumed that during the design phase they would select appropriate security algorithms to implement security use cases.

(iv) Common Criteria (CC) – This methodology relates Common Criteria (Common Criteria Implementation Board, 1999) with Use Cases, to handle security issues in Information Technology products during the Software Engineering Process. The approach followed by the process is as follows:

a) Identify the primary actor and supporting actor corresponding to the functionality.

b) Fill the actor profile consisting of seven fields (actor, use case, type, location, secret exchange, private exchange, and association).

c) Based on actor profile threats are extracted from a predefined repository.

d) Security techniques are defined to mitigate the identified threats, such as Non-Repudiation security technique is elicited to mitigate threat Repudiate Send.

An example, Traveler functionality "Book ticket" of the Railway Reservation System is considered. The actor (Traveler) profile is shown in Figure 2.4.

Following threats are extracted:

- T.Flooding
- T.Privacy Violated
- T.Change Data
- T.Repudiate Receive

Now threats are mapped to security techniques, such as Flooding is mapped to security technique O.Check User id and O.Password.

Actor	Traveler
Use case	Book Ticket
Type	Human
Location	Remote
Secret Exchange	Yes
Private Exchange	Yes
Association	Read Write

Figure 2.4 Traveler Profile for Book Ticket Functionality

Analysis: In common criteria approach, the threats on the functional requirements are identified systematically; the clear relation of a threat to the functionality is depicted. Security techniques are defined to mitigate identified threats are very abstract. No formal specification of security requirements is done; neither the security requirements are analyzed or prioritized.

Summary Use Case-based Approach: Based on analysis following conclusions is drawn about the Use Case-based approach:

a) These are traditional and basic approaches.
b) They mainly identify the functional requirements and possible threats to requirements.
c) Some methodology namely security use cases and common criteria under this approach broadly suggests the security algorithms to mitigate threats.
d) They do not prioritize the requirements.
e) No domain constraints (environmental and device constraints) are considered while defining the security algorithm.

2.3 Goal Oriented Approach

Goal-Oriented approach views the target system and its environment as a collection of active components. It helps in early requirements analysis; where goals represent the complete set of requirements. Goal refinement provides traceability from high-level strategic objectives to low-level technical requirements. It helps in:

- Structuring complex requirements documents.
- Detection and management of conflicts among requirements.
- Communicating with the customers regarding requirements.

- Choosing among the available alternatives.

Goal-oriented methodologies represent the attacks or requirements as a goal. **Goals** are intended behavior and perspective statement of intent about the system. ***Security Goals*** are an abstract representation of security concerns to the assets of the system. For example, ‘Privacy’ security goal signifies that asset ‘Patient Health Information’ should remain private and confidential from misusers. Security goals are classified as Integrity, Confidentiality, and Availability goals (Lamsweerde, 2004). Security goal defines very general statement about the security of assets. For instance, the patient in the Hospital system has Confidentiality goal which expresses ‘his health parameters should remain confidential.’ Similarly, Integrity goal by Hospital Authority conveys that ‘malicious actors should not change doctor assigned to the patient.’ ***Security Requirements*** are a detailed description of security concern, or they give detailed statement about system behavior that is not acceptable. Further, Security Goals are refined as Security Requirements as shown in Figure 2.5. For example, Security Goal Confidentiality can be refined as Identification, Authentication, and Authorization. Some Methodologies based on goal-oriented approach are attack trees, intentional anti-model, secure tropos. The detail of each approach is as follows:

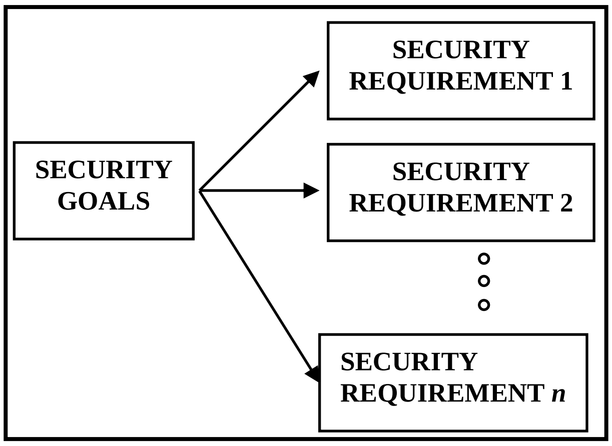


Figure 2.5 Refinement of Security Goals as Security Requirements

(i) Attack Trees –Attacks Trees are used to identify the possible attacks which further helps in risk analysis and in turn can be used for security analysis of the system. Attack Tree is a formal method for describing the security of a system based on varying attacks. The goal of the attacker is placed at the root node and ways of achieving goals as leaf nodes. Satisfaction of goal is represented by either satisfaction of all leaves (AND) or by the satisfaction of a single leaf (OR). Here, the overall goal of the attacker with preconditions, steps followed for attack and post conditions are specified. Figure 2.6 shows an Attack Tree for Login functionality in the Railway Reservation System.

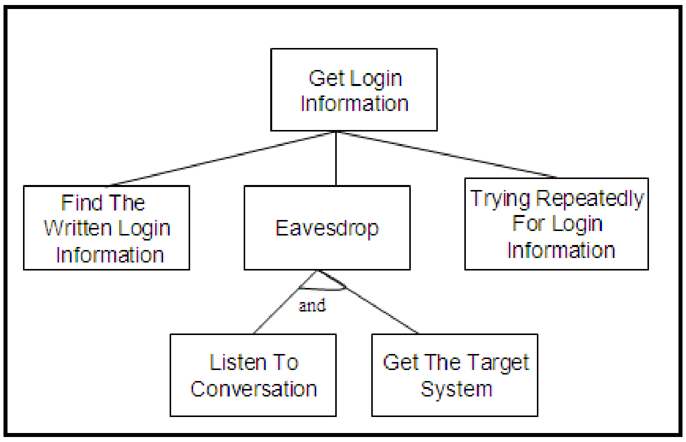


Figure 2.6 Attack Tree for Login Functionality

Analysis: In Attack Trees methodology, attacks are represented in the form of a tree; it depicts how an intruder can reach its goal by exploiting the vulnerable points of the system. They do not specify, analyze, and prioritize the security requirements.

(ii) Intentional Anti-models – It is an incremental methodology which focuses on defining the security measures for mitigation of threats to the system. The process starts with the elaboration of attacker goal by building two different models iteratively and concurrently: (a) A model of the system-to-be that covers software and its environment with their goals, objects, agents, requirements, operations, and assumptions. (b) An intentional anti-model which starts with an anti-goal that is a threat to the initial goal of the model described in (a). After that, the trees are derived systematically through anti-goal refinement until leaf nodes representing either the vulnerabilities observable by the attacker or anti-requirements implementable by the attacker is reached. Then, the original model is enriched with new security measures derived from the anti-model. The process is repeated iteratively till it encapsulates all security goals. An example, Figure 2.7 shows an Intentional Anti-Model for the Railway Reservation System.

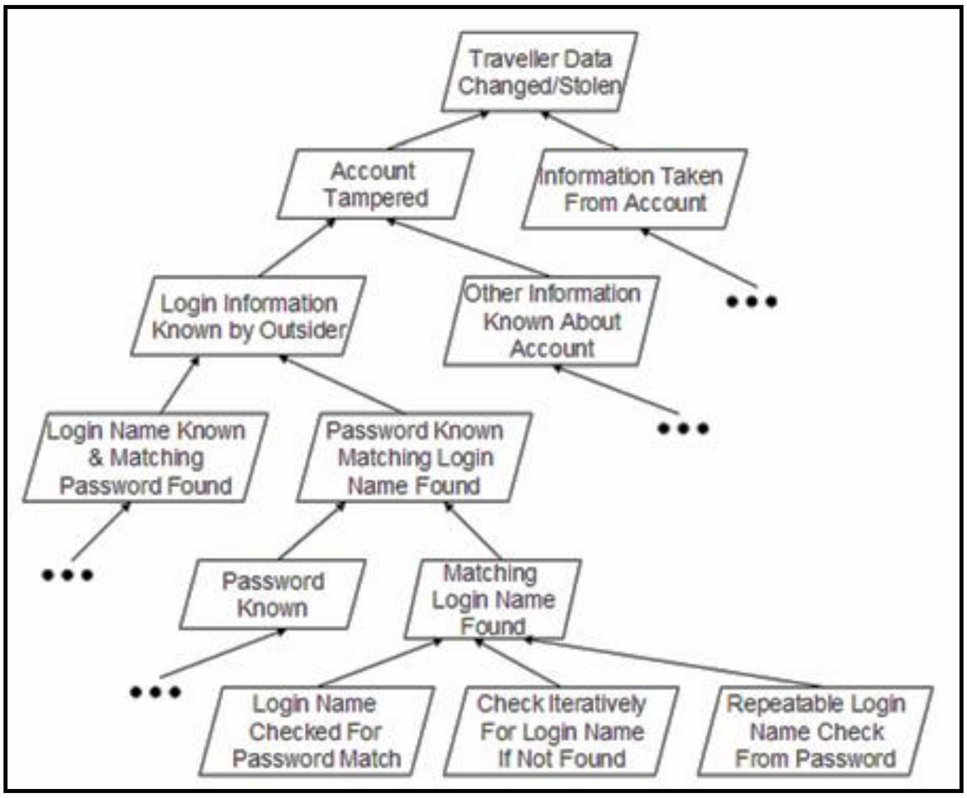

Traveller Data Changed/Stolen
Account Tampered
Information Taken From Account
Login Information Known by Outsider
Other Information Known About Account
Login Name Known & Matching Password Found
Password Known Matching Login Name Found
Password Known
Matching Login Name Found
Login Name Checked For Password Match
Check Iteratively For Login Name If Not Found
Repeatable Login Name Check From Password

Figure 2.7 Intentional Anti-Model for Railway Reservation System

For instance, vulnerability:

‘Repeatable Password Check from Login Detail’ is identified against the anti-requirements:

‘Login Name Known & Matching Password Found’

Moreover, then the original system is modified by adding new security goal:

Avoid [Repeatable Password Check from Login Detail].

Analysis: In Intentional Antimodel methodology, first the goals and corresponding antigoals are identified. These antigoals are decomposed iteratively to get the detailed vulnerabilities or anti-requirements which exist in the system. Then new security measures are added to the system to protect vulnerable points. No prioritization is done, and no domain constraints are considered while defining the broad security measures.

(iii) Secure Tropos – It is an extension of Tropos methodology and adopts a hierarchical approach for security implementation. The methodology has the following steps:

- **a)** Identify the actors, their functionalities, and dependency.
- **b)** During early requirement phase actors specify the security goals such as privacy, authentication.
- **c)** In Late requirement phase Security diagram is constructed which depict the following information:
    - a. Security goals as the starting point (root)
    - b. Possible attacks to security goal are added to the diagram.
    - c. Protection measures are identified and added to the diagram to mitigate the attacks.

d) Test scenarios are generated at design time for testing the security of the system as mentioned in their later work (Mouratidis & Giorgini, 2007).

Security diagram using Secure Tropos for the Railway Reservation System is shown in Figure 2.8. Oval represents the actors, where the traveler is related to railway management for the execution of functionality 'Book Ticket' (represented as a rounded rectangle). Cloud represent the security goal that traveler has imposed on the system for login functionality.

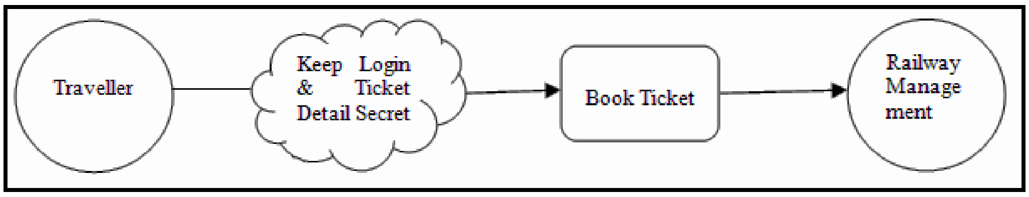


Figure 2.8 Secure Tropos for Railway Reservation System

Analysis: The given approach effectively identifies the actors, functionalities, and dependencies that exist between them. The security goals are initially specified by the actors. Later they are refined, analyzed with the help of security diagram, and broad security measures are defined to protect the system from threats. However, they do not prioritize the security requirements which are expressed as security goals and does not take any domain constraints into consideration while suggesting the security measures for implementation.

Summary Goal Oriented Approach: Based on the analysis following conclusions is drawn about the goal-oriented approach:

a) Identifies the goals, and attacks/ threats to the goal of the system.

b) Identifies the detailed security vulnerabilities present in the system.

c) Attack trees and Intentional antimodel methodologies only model the vulnerable point in the system that may be used in defining the security measures.

d) Secure tropos recommends broad security measures to attain the security goals of the system.

e) They do not analyze or prioritize the security goals. Even they are not specifying the security requirements.

f) None of the above proposals consider domain constraints while defining the security measures.

2.4 Process Oriented Approach

The approach has detailed steps for Information System development. It contains all necessary information regarding how and when data is moved and processed by an Information System. Some methodologies under this category are Software Quality Requirements Engineering (SQUARE) (Mead, 2005), Security Requirements Engineering Process (SREP) (Mellado, Medina, & Piattini, 2007), a framework for security requirements elicitation presented by Haley et. al (Haley, Laney, Moffett, & Nuseibeh, 2008), and framework by Fabian et al. (Fabian, Gurses, Heisel, Santen, & Schmidt, 2010) for elicitation and analysis of security requirements. Now mentioned methodologies are elaborated:

(i)Security Quality Requirement Engineering (SQUARE) - SQUARE methodology has steps for elicitation, categorization, and prioritization of Security Requirements of software projects. The process has the following steps:

a) Security goals such as Confidentiality, Accuracy, and Integrity are laid down that conforms to organization overall business goal.
b) Security goals are analyzed to get the vulnerable points and threats present in the system using an architectural diagram, use cases, misuse cases, etc.
c) Based on identified threats various security requirements are defined as a constraint to the system.
d) Security requirements are categorized according to levels (system, software, etc.).
e) Security requirements are prioritized based on risk assessment results using methods such as Triage, Win-Win, etc.

An example, SQUARE process for the functionality 'Book Ticket' executed by stakeholder 'Traveler' for Railway Reservation System:

- Business Goal: To provide an application that supports ticket booking.
- Security Goals
    - The reservation system will be available for ticket booking and other functions.
    - Management will control system configuration and usage.
- Elicited Security Requirements
    - (R1) Availability of System
    - (R2) Authentication and Authorization of users
    - (R3) Privacy should be maintained
- Prioritized Security Requirements based on risk assessment
    - R3>R2>R1

Analysis: A systematic process for elicitation and prioritization of security requirements as a constraint to the system is defined. Security requirements are defined as a constraint to the system. No security algorithm/ mechanisms are suggested for implementation of security threats.

(ii) Security Requirements Engineering Process (SREP) –SREP methodology is proposed and applied by Daniel M. et al. It consists of various activities; which are repeatedly applied and specified to the iteration of the Unified Process model. SREP embeds the concept of Common Criteria which helps in Threat Modeling and identification of Security measures. They are using repository named security resource repository (SRR) which is created using already developed systems. SRR contains security related information like vulnerable point, threats, security goals, etc. related to assets.

SREP has the following set of activities:

a) Identify the Critical Assets: Identify the assets of the system. Assets can be extracted from SRR based on similarity of the domain, function, etc. of the current system.
b) Identify Security Goals: Firstly, SRR is checked if the asset is already available in it, then security goal is retrieved directly from SRR else, the security goal are determined for each asset.
c) Identify the Vulnerabilities and Threats: If the asset is already present in SSR, then vulnerabilities and threats are retrieved directly from it otherwise use cases based techniques like misuse cases, abuse cases, etc. are used for vulnerability and threat identification.

d) Risk Assessment: Risk is calculated using the probability of threat occurrences.

e) Elicit Security Requirements: Security Requirements are defined as constraints to the system by analysis of security goals and related threats.

f) Prioritize Requirements: Security Requirements are ranked based on the results of risk assessment.

g) Requirements Inspection: Verification of security requirements is done to check the correctness, completeness, unambiguity, etc.

h) Repository Improvement: At the end of each phase of Unified Process repositories are extended with new information.

An example, the above process for the functionality 'Book Ticket' executed by stakeholder 'Traveler' for Railway Reservation System:

- Vulnerable/ Critical Assets:
  - Traveler Information
  - Ticket Information
  - Card Information
- Security Goals
  - System Availability
  - Privacy/ Confidentiality of Information
  - Authentication/ Authorization of persons involved
- Threats Identified
  - Unavailability of network
  - Password Leak
  - Information Theft

- Security Requirements
    - The session should expire when the system is unattended for specified amount of time.
    - Block repeated login attempt.
    - Ensure Privacy of information.

At last, prioritization is done based on risk assessment results, and new findings are updated in the repositories.

Analysis: It covers elicitation, analysis, and prioritization of security requirements. Here security requirements are expressed as constraints which are architectural constraints. They do not analyze the various domain constraints.

(iii)Framework by Haley et al. –An iterative process presented by Haley et al. has the following steps:

a) Identify the functional requirements and associated assets.

b) Security goals are determined to protect the assets.

c) Threats corresponding to security goals are determined.

d) Security Requirements are identified as constraints to the functional requirements for achieving the security goals.

An example, the above process for the functionality ‘Book Ticket’ executed by stakeholder ‘Traveler’ for Railway Reservation System:

- Functional requirements and assets
    - Book Ticket.

    - Assets (traveler information, ticket information, credit card details) shall be protected.
- Security Goals
    - (SG1) Traveler and ticket details should be private.
- Threats
    - (T1) Data gets leaked
    - (T2) Data is not available
- Security Requirements (constraints)
    - (SR1) Traveler and Ticket data is given only to members of booking system.

Analysis: They identify the functional requirements, assets and corresponding security goals. They elicit the security requirements as constraints to security goals. They do not prioritize the security goals. No security algorithm is suggested/ chosen to implement the security requirements, and no domain constraints are considered.

(iv)Framework by Fabian et al. –Fabian et al. have proposed a framework for security requirements elicitation and analysis. The proposed framework has the following steps:

a) First, the involved stakeholders of the system are identified.

b) Functional requirements and non-functional requirements with associated assets are identified for each stakeholder.

c) Next security goals are defined to protect the asset.

d) Identified security goals are refined and represented as security requirements.

e) Security requirements are defined and analyzed based on the assets, its misusers and its context of use.
f) Once the security requirements are defined at stakeholder's level, they are compiled at the system level by combining their viewpoints, to resolve the conflict and to arrive at a consistent set of system requirements.
g) Risk analysis is done by considering the vulnerable points, attackers, and related attacks. Based on the results of risk analysis security measures are defined to implement the security requirements.

An example, the above process for the functionality 'Book Ticket' executed by stakeholder 'Traveler' for Railway Reservation System:

- Functional Requirements and assets
    - Get Ticket Booked
    - Assets: Ticket information, Traveler information, Credit Card information shall be protected
- Security Goals
    - Confidentiality
    - Availability
- Security Requirement
    - Customer ticket details should not be disclosed to the third party.
- All functional and other requirements are documented.

Analysis: Security Requirements are elicited and analyzed firstly at the individual level and then at the system level. The framework effectively deals with the conflict between the requirements arising due to different viewpoints of stakeholders. It identifies the vulnerable points, threats with security measures in an ad-hoc manner. No prioritization of security requirements is done, and security measures are defined without considering the environmental and device constraints.

Summary of Process Oriented Approach: Based on the analysis following conclusions is drawn about the process-oriented approach:

a) Identifies the security goal.
b) Identifies the vulnerabilities and threats.
c) Identifies the security requirements in the form of constraint to functional requirements. Formal specification or categorization of security requirements is missing. A formal specification of security requirements as done by Firesmith (Firesmith, Engineering Security Requirements, 2003) is necessary to compute the generic form of SRS by adding one more section.
d) SQUARE and SERP methodologies prioritize the security requirements.
e) Some methodologies namely framework by Fabian et. al and proposal by Haley et. al suggests broad security measures such as provide access control, perform encryption, etc. for implementation but do not consider the various domain constraints while defining security algorithms.

2.5 Evaluations of Security Engineering Methods

In this section, we evaluate the foregoing methodologies on different parameters of security engineering. We classify our evaluation into three phases namely security requirements engineering, security design engineering, and security testing. The security requirements engineering phase is further classified as elicitation, analysis, prioritization, threat modeling, and risk analysis. Security design engineering phase is classified as consideration of domain constraints and suggestion of security algorithm. Security testing phase whether the testing activity to check security embedded in the system is done or not. Another parameter domain of application is also considered. Table 2.1 shows a summary of contributions of available frameworks.

Table 2.1 Summary of the Contributions

Methodology	**Security Engineering Activities**			**Domain of Application**	**Main Contribution**
	Security Requirements Engineering	**Security Design Engineering**	**Security Testing**		
Abuse Cases	**Elicitation:** No	**Environment:** No	They did not consider the security testing part.	Internet-based Information Security Lab	• It is only helpful in identifying threats.
	Analysis: No	**Device:** No			
	Prioritization: No	No mechanisms are suggested to mitigate threats.			
	Threat Modeling: Yes				
	Risk Analysis: No				
Misuse Cases	**Elicitation:** Abstract	**Environment:** No	They did not consider the security testing part.	E-commerce Systems	• It helps in the modeling of threats and a broad mechanism to implement them are specified.
	Analysis: No	**Device:** No			
	Prioritization: No	Broad security mechanisms are suggested to mitigate threats in the form of new use cases.			
	Threat Modeling: Yes				
	Risk Analysis: No				
Security Use Cases	**Elicitation:** Yes	**Environment:** No	They did not consider the security testing part.	Banking System	• A novel method for elicitation of Security Requirements.
	Analysis: No	**Device:** No			
	Prioritization: No	Done during the design phase (assumption).			
	Threat Modeling: Yes				
	Risk Analysis: No				
Common Criteria	**Elicitation:** Yes	**Environment:** No	They did not consider	Student Management System	• Threats are modeled based on
	Analysis: No	**Device:** No			

	Prioritization: No **Threat Modeling:** Yes **Risk Analysis:** No	Abstract security measures for implementation are specified.	the security testing part.		functional requirements and its user, and abstract measures are identified to mitigate threats.
Attack Trees	**Elicitation:** No **Analysis:** No **Prioritization**: No **Threat Modeling:** Yes **Risk Analysis:** No	**Environment:** No **Device:** No No consideration of this phase.	Testing part is not covered.	Web-based System	• Identify the vulnerable points of attack.
Intentional Anti-model	**Elicitation:** No **Analysis:** No **Prioritization**: No **Threat Modeling:** Yes **Risk Analysis:** No	**Environment:** No **Device:** No No consideration of this phase.	No clues for doing security testing.	Web Based Banking System	• Only threats and vulnerable points are identified and analyzed. • Broad measures of implementing security are listed.
Secure Troops	**Elicitation:** Yes **Analysis:** No **Prioritization**: No **Threat Modeling:** Yes **Risk Analysis:** No	**Environment:** No **Device:** No Broad security mechanisms are specified.	Attack scenarios are generated at design time to help testers during testing.	e-SAP deals with facilities provided to older persons.	• Security Requirements are identified as constraints to the system. • Broad measures are suggested for implementation. • Test scenarios are generated at design time.
SQUARE	**Elicitation:** Yes **Analysis:** No **Prioritization**: Yes **Threat Modeling:** Yes **Risk Analysis:** Yes	**Environment:** No **Device:** No Done in an ad-hoc manner	Testing part is not covered.	Asset Management System	• This methodology has steps for Elicitation and Prioritization. But they specify the security requirements as constraints.

SREP	**Elicitation:** Yes	**Environment:** No	Inspection is done to check the correctness, consistency, etc. of security requirements.	No case study considered	• Vulnerable assets are considered. • Various threats applicable to assets are identified, and risk assessment is done. Security requirements are identified and prioritized.
	Analysis: No	**Device:** No			
	Prioritization: Yes	Broad measures are suggested.			
	Threat Modeling: Yes				
	Risk Analysis: Yes				
Framework by Haley et al.	**Elicitation:** Yes	**Environment:** No	Testing part is not covered.	Air Traffic Control	• Security requirements are defined as constraints to the functional requirement.
	Analysis: No	**Device:** No			
	Prioritization: No	Broad mechanisms are specified.			
	Threat Modeling: Yes				
	Risk Analysis: No				
Framework by Fabian et al.	**Elicitation:** Yes	**Environment:** No	No provision of verification or validation is adopted.	No case study considered	• Security Requirements are identified and analyzed both at stakeholder level and system level to avoid conflicts. • Security measures are defined in an ad-hoc manner.
	Analysis: Yes	**Device:** No			
	Prioritization: No	Broad measures are defined to implement the security.			
	Threat Modeling: Yes				
	Risk Analysis: Yes				

Summary

- Use case-based methodologies: They focus on identification of threats associated with the functional requirements. They do not analyze or prioritize the threats.

- Goal-oriented methodologies: Deal with the identification of security goals (confidentiality, integrity, authentication), attacks, vulnerable points and sometimes may suggest broad measures of security. They do not analyze and prioritize the security goals. Possible attacks, antigoals, vulnerable points to the security goals are identified. They are not either doing the prioritization of security threats or security goals. Some of them (Lamsweerde, 2004) (Mouratidis H., 2002) specify the broad security measures for mitigation of threats.

- Recent process-oriented approaches: Attempt to address the different task of requirements engineering such as elicitation, analysis, prioritization, specification of security requirements. The major gap is that they specify the security requirements as a constraint but fails to formally specify the security requirements or categorize them. In addition to this, some of them (Mellado, Medina, & Piattini, 2007) (Fabian, Gurses, Heisel, Santen, & Schmidt, 2010) suggests broad measures without considering the different domains.

- From Table 2.1 we can see that most of the methodologies focus on web-based systems, they do not consider the new emerging domains such as cloud computing, Internet of Things (IoT) which has specific security issues such as Shared Environment, Multi-location Data placement, Trust, Data Freshness, Trust, and Liability.

Hence it is established that existing proposals define the security mechanism without considering the design constraints. This may unnecessarily constrain the system. Secondly, none of the proposals evaluate the embedded security. If security can be measured, it can help in mitigation of live threats by physical measures or by new security algorithm. Also, these proposals focus on web-based applications. In the next chapter, we propose a security engineering framework which will address these problems.

Publication from this work

1. Jaiswal, S., and Gupta, D. (2016). Security Engineering Methods: In-Depth Analysis. *International Journal of Information and Computer Security* (*in press*).

[Scopus Indexed]

CHAPTER 3

FRAMEWORK FOR SECURITY ENGINEERING

With the increase in the use of software system, security becomes an emergent area of study. Most of the software engineering processes deals with security issues during the design or implementation phase which may result into unnecessary constrained system. So, there is a need for a new process which deals with security issues from requirement engineering phase and then selects appropriate algorithms for implementation of security issues. So, here in this chapter, we provide a framework for Security Engineering for handling issues in a structured manner. First, the explanation of novel framework for handling security issues in software systems is given. Then the different phases of proposed framework are explained. After that, a case study of Content Management System is presented to explain the activities of our novel framework in detail.

3.1 Security Engineering Framework

The proposed novel framework of security engineering for identification and handling of security issues is depicted in Figure 3.1. The proposed framework consists of three phases:

- Security Requirements Engineering
- Security Design Engineering
- Security Testing

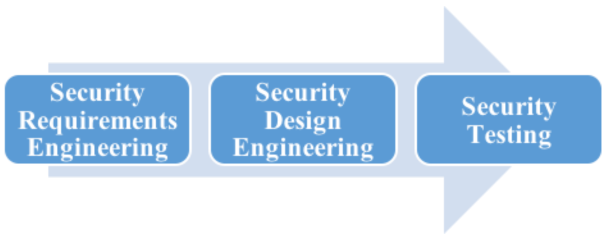

Security Requirements Engineering
Security Design Engineering
Security Testing

Figure 3.1 Security Engineering Framework

- **Security Requirements Engineering.** In this phase, security requirements required to mitigate threats are identified, analyzed and prioritized along with the functional and non-functional requirements. Activities of this phase start with the identification of actors, their functional requirements, non- functional requirements and associated assets. Next, threats applicable to system assets at vulnerable points are identified. After that, identified threats are represented in the form of security requirements. Then, the threats are evaluated using risk analysis method and based on it security requirements are prioritized.

- **Security Design Engineering.** In this phase, efficient mechanisms to implement the security requirements are identified. These mechanisms are basically security algorithms which are identified to implement the security requirements or to mitigate the threats of the system. Algorithms are suggested based on number of threats mitigated by the algorithm and various domain constraints namely encryption speed, bandwidth, memory requirements, power, throughput, etc. Finally, a template is generated showing the details related to this phase.

- **Security Testing.** In this phase, deployed algorithms are tested to check if all potential threats are mitigated. Overall security assessment of the system is done by generating a metric showing system security level.

Subsequent sections explain each phase of proposed framework in detail.

3.2 Security Requirements Engineering Phase

This sub section of thesis will present a novel security requirements engineering framework, as a first contribution of the thesis. This phase consists of different activities:

(i) Security Requirements Elicitation

(ii) Security Requirements Prioritization and Specification

These activities are integrated into a single framework as shown in Figure 3.2.

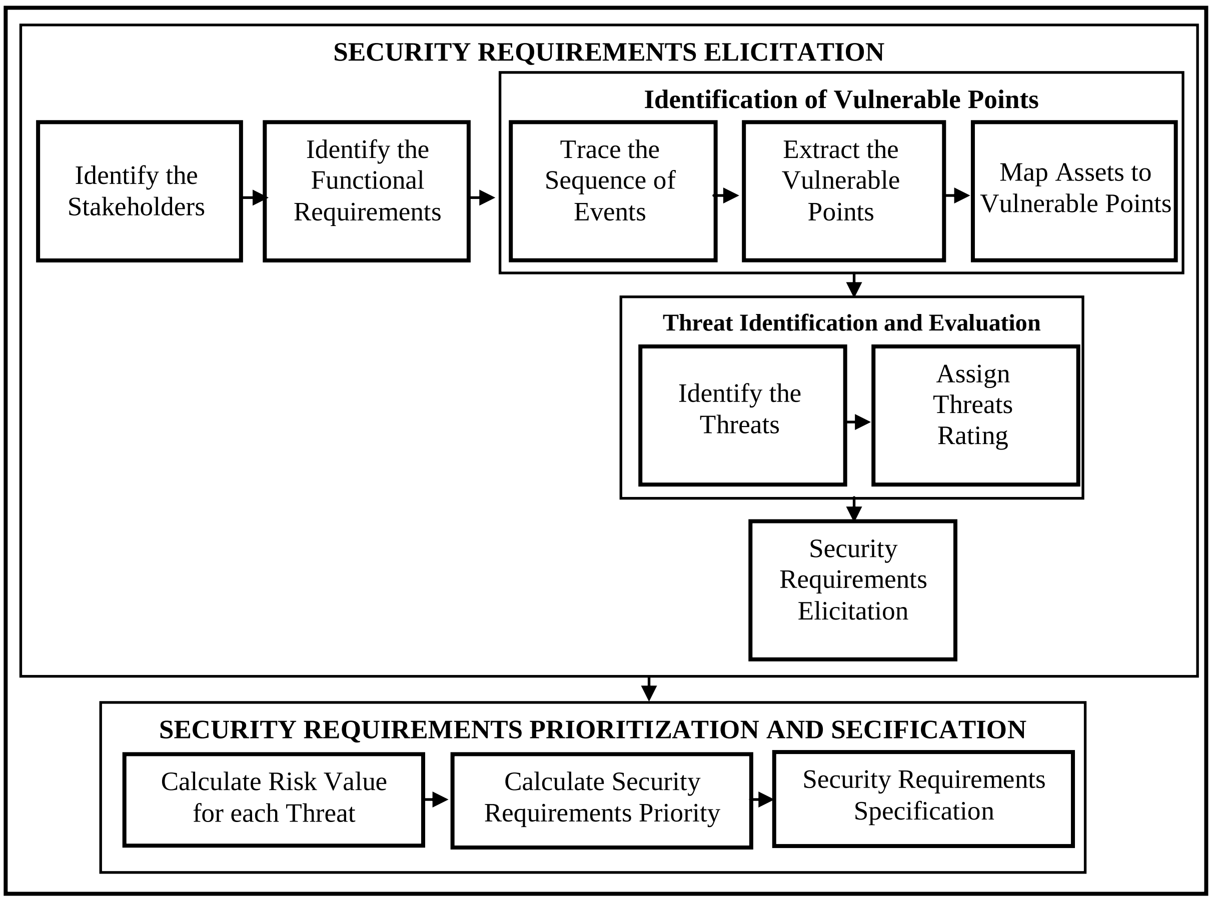

SECURITY REQUIREMENTS ELICITATION
Identification of Vulnerable Points
Identify the Stakeholders
Identify the Functional Requirements
Trace the Sequence of Events
Extract the Vulnerable Points
Map Assets to Vulnerable Points
Threat Identification and Evaluation
Identify the Threats
Assign Threats Rating
Security Requirements Elicitation
SECURITY REQUIREMENTS PRIORITIZATION AND SECIFICATION
Calculate Risk Value for each Threat
Calculate Security Requirements Priority
Security Requirements Specification

Figure 3.2 Security Requirements Engineering Process

As discussed in the previous chapter various methodologies like SQUARE (Mead, 2005), SREP (Mellado, Medina, & Piattini, 2007), Helay (Haley, Laney, Moffett, & Nuseibeh, 2008), Fabin (Fabian, Gurses, Heisel, Santen, & Schmidt, 2010) are present in the literature for performing different activities of security requirements engineering (requirements elicitation, analysis and prioritization). But none of them include all the activities of requirements engineering in a single framework. SQUARE proposes a process for elicitation and prioritization of security requirements. Here, security requirements are presented as a constraint to the system, also they are not suggesting any security algorithm/ mechanisms to implement the security threats. SREP covers elicitation, analysis, and prioritization of security requirements. Here, security requirements are expressed as architectural constraints. Also, they do not analyze various domain constraints. Helay et. al identifies the functional requirements, assets and corresponding security goals. They elicit the security requirements as constraints to security goals. They do not prioritize the security goals and no security algorithm is suggested/ chosen to implement the security requirements. In the proposal by Fabian, Security Requirements are elicited and analyzed both at the individual level and at the system level. The framework effectively deals with the conflict between the requirements arising due to different viewpoints of stakeholders. It identifies the vulnerable points, threats with security measures in an ad-hoc manner. No prioritization of security requirements is done, and security measures are defined without considering the environmental and device constraints.

In short all the present methodologies identify the security requirements in the form of constraint to functional requirements. Formal specification or categorization of security requirements is missing. A formal specification of security requirements as done by Firesmith (Firesmith, Engineering Security Requirements, 2003) is necessary to enhance the generic form of SRS document prescribed by IEEE (IEEE 830, 1998). This document would be augmented with additional section specifying the detailed security requirements.

Different activities of proposed Security Requirements Engineering framework which consists of two main phases: (i) Security Requirements Elicitation, (ii) Security Requirements Prioritization and Specification are discussed below in detail:

3.2.1 Security Requirements Elicitation

As shown in the Figure 3.2, activity of security requirements elicitation consists of different sub- activities:

(i) identify the Stakeholders

(ii) Identify the functional Requirements

(iii) Identification of Vulnerable Points

(iv) Threat Identification and Evaluation

(v) Security Requirements Elicitation.

All these sub- activities are now discussed below in detail:

- **Identify the Stakeholders:** Stakeholders are those who are directly or indirectly interacting with the system and using its services. Stakeholders of the system are identified using Viewpoint (VP) approach (Kotonya & Sommerville, 1996), here two class of actors are identified, direct and indirect actors. VP approach is chosen as it clearly distinguishes the direct and indirect actors to avoid conflict arising because of different viewpoints. Direct actors are directly interacting/ using the system services whereas indirect actors are involved in back-end operations, such as development and maintenance team members. Here, our attention is only on the direct actors because they are directly interacting with the system functionalities, so require security. For instance, in the Web-Based Banking system, direct actors are a customer, banker, and bank DBMS; indirect actors are administrator, vendor, etc.

- **Identify the Functional Requirements:** Functional requirement represents the true functionalities which stakeholder needs/ expect from the system. Therefore, functional requirements for all direct actors are identified. With the identification of functional requirements, different associated non-functional requirements and assets are also identified. For example, functional requirement of direct stakeholder for a customer in the Web-Based Banking system can be Registration, Login, Deposit Money, Withdraw Money. Non-functional requirements are reliability, performance and assets are customer login information, smart card information, account information.

- **Identification of Vulnerable Points:** Vulnerability is the weakness in the system environment, which may be exploited by an attacker to cause damage to system assets (Uzunov, Falkner, & Fernandez, 2015) (Mayer, Heymans, & Matulevicius, 2007) (Stoneburner, Alice, & Feringa, 2002). Functional requirements give rise to vulnerable points. Therefore, all the vulnerable points analogous to the functional requirements of the system are identified. Vulnerable point identification consists of following sub-activities:

(i) Trace the Sequence of Events. Trace the sequence of events which occurs in the execution of functionality, event trace is generated by drawing the sequence diagram for identified functionality. Sequence diagram depicts the interaction between the involved objects for a given functionality. A sample sequence diagram for 'Login' functionality is shown in Figure 3.3. A sequence diagram is enriched with additional information like assets used by the functionality and note of any private and secret data exchange, which are required for embedding security in the system.

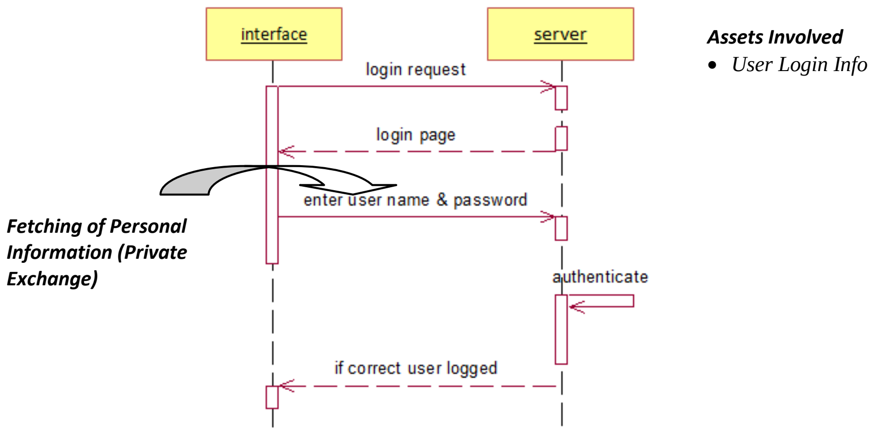


Figure 3.3 Scenario using Sequence Diagram for 'Login' Functionality

(ii) Extract the Vulnerable Points. Vulnerable points are extracted from the sequence trace generated in the previous activity. Points where input or output is provided and communication between objects occurs are considered to be vulnerable points for attack. Therefore, all such points are extracted for all functional requirements, and vulnerabilities are mapped to identified vulnerable points. Continuing our example, vulnerable point for above functionality 'Login' are: (i) while sending the request (ii) while entering the login details (iii) during the authentication. Mapped vulnerabilities to these points are AAA, Insecure Interface, Remote Access, Communication Encryption, Lack of weak encryption of archive and data in transit.

(iii) Map Assets to Vulnerable Points. An asset can be anything that has value to the organization; it may be tangible (infrastructure) or intangible (customer information, trust). Assets mapping is necessary because it is the target of attackers and needs protection. So, all the identified assets are mapped to different vulnerable points for further analysis.

- **Threats Identification and Evaluation:** Threats are circumstances that have potential to cause harm to system assets. Threat occurs at vulnerable points to cause damage to system assets. Threats identification and evaluation consists of following sub-activities:

(i) Identify the Threats. Repository of potential threats similar to common criteria (Common Criteria Implementation Board, 1999), CVE database (Ozkan) is created after analyzing existing software systems. Sample repository is shown in Table 3.1, column of table represents the vulnerabilities and row represents the threats possible in the system. 'X' in the table shows the occurrence of threat on given vulnerability. Threats to vulnerable points are extracted from a predefined repository based on following parameters: the actor's functionality, type of data (private and secret) involved in functionality, and type of functionality (read, write, read-write). The threat list can be updated further if some new threat is identified or reported.

Let’s take an example, as it can be seen form Table 3.1 on vulnerability ‘AAA’ following threats are possible: password cracking, impersonate, disclose data, repudiate, data theft, password reuse, insider, MITM, Operational Log Compromise, Security Log Compromise, Privilege Abuse, Data Leakage, Management Interface Compromise. So, here for threat extraction criteria is as follows:

- Functionality is ‘Login’
- Type of Data is ‘Secret’
- Type of Functionality is ‘read-write’

Based on these parameters threat extracted from repository is ‘Password Cracking.’

(ii) Assign Threats Rating. Threat rating depicts the occurrence probability of threat; it is the rough measure of how likely a threat would exploit the vulnerabilities of the system to gain access to system assets. Threat rating is calculated by checking number of vulnerabilities exploited by a threat in Vulnerability/ Threat mapping table shown in Table 3.1. For instance Threat Rating for threat ‘T.Password Cracking’ is ‘1’ as it exploits only vulnerability ‘V. AAA’.

Table 3.1 Vulnerabilities-Threats Mapping Table for web-based system

Vulnerability →		1	2	3	4	5	6	7	8	9	10	11
Threats ↓		V.AAA	V.User Provisioning	V.User De-Provisioning	V.Remote Access to Management Interface	V.Communication Encryption Vulnerabilities	V.Lack of Weak Encryption of Archive and Data in Transit	V.Impossibility of Processing Data in encrypted form	V.Poor Key Management Procedures	V.Misconfiguration	V.System/ OS Vulnerabilities	V.Lack of Poor and Untested Business Continuity and Disaster Recovery Plan
1	T.Password Cracking	X										
2	T.Impersonate	X						X				
3	T.Sniffing					X						
4	T.Social Engineer		X			X						
5	T.Disclose Data	X										
6	T.Malicious code				X	X	X					
7	T.Repudiate	X	X	X								
8	T.Change Data	X			X	X	X	X				
9	T.Data Theft	X			X	X	X	X				
10	T.Password Reuse	X										
11	T.Insider	X							X			
12	T.MITM	X			X	X	X					
13	T.Spoofing					X	X					
14	T.Network Issues					X	X			X	X	X
15	T.DoS				X					X	X	
16	T.DDoS				X					X	X	
17	T.Sabotage											
18	T.Operational Logs Compromise	X	X	X							X	
19	T.Security Log Compromise	X	X	X							X	
20	T.Data Deletion											
21	T.Priviledge Abuse	X	X	X						X		
22	T.Unauthorized Physical Access											
23	T.Natural Disaster											X
24	T.Data Leakage	X				X			X			
25	T.Loss of Encryption Keys							X	X			
26	T.Management Interface Copmpromise	X			X					X	X	

Table 3.1 Continued

Vulnerability →		12	13	14	15	16	17	18	19	20	21	Threat Rating
Threats ↓		V.Inadequate Resource Provisioning and Investments in Infrastructures	V.Storage of Data in Multiple Jurisdiction and Lack of Transparency	V.Lack of Completeness and Transparency in Terms of Use	V.Lack of Security Awareness	V.Unclear Roles and Responsibilities	V.Lack of Forensic Readines s	V.Need-To-Know Principle Not Applied	V.Inadequate Physical Security Procedures	V.Lack of Policy or Poor Procedures for Log Collection and Retention	V.Inadequate/ Misconfigured Filtering Resources	
1	T.Password Cracking											1
2	T.Impersonate											2
3	T.Sniffing				X							2
4	T.Social Engineer								X			3
5	T.Disclose Data											1
6	T.Malicious code											3
7	T.Repudiate		X	X		X						6
8	T.Change Data											5
9	T.Data Theft											5
10	T.Password Reuse											1
11	T.Insider			X		X						4
12	T.MITM				X							5
13	T.Spoofing	X										3
14	T.Network Issues	X										6
15	T.DoS				X	X		X			X	7
16	T.DDoS				X	X		X			X	7
17	T.Sabotage			X		X						2
18	T.Operational Logs Compromise						X			X		6
19	T.Security Log Compromise						X			X		6
20	T.Data Deletion			X								1
21	T.Priviledge Abuse							X				5
22	T.Unauthorized Physical Access								X			1
23	T.Natural Disaster								X			2
24	T.Data Leakage											3
25	T.Loss of Encryption Keys			X								3
26	T.Management Interface Copmpromise	X							X		X	7

- **Security Requirements Elicitation:** Security Requirements (Firesmith, Engineering Security Requirements, 2003) expresses the measures needed to protect the assets from threats. Threats are mapped to security requirements listed in section 1.6 on page number 12, through analysis and experience of requirement engineers. For example, Security requirements to mitigate threat 'Password Cracking' are 'Identification' and 'Authentication.'

3.2.2 Security Requirements Prioritization and Specification

This phase has two main activities (a) Analysis of Security Requirements and (b) Prioritization of Security Requirements.

Security requirements identified in the previous step are analyzed, prioritized and specified to get a concrete set of important security requirements. During analysis following sub activities are performed:

(i) All security requirements are checked for completeness and consistency.

(ii) Similar requirements are grouped together.

(iii) Conflicting requirements are dealt carefully, and if any conflict is found would be removed immediately.

Elicited security requirements are not of equal importance for an organization because of following reasons:

(a) Organization has time and budget constraint

(b) One algorithm is not sufficient to implement all the security requirements.

Therefore, prioritization of security requirements is done, and high priority security requirements are implemented/ handled first. Also, prioritization would help the developer/ user in knowing which security requirements are more important and need immediate focus. Following steps are followed in prioritization of security requirements:

- **Calculate Risk Value for each Threat.** Risk shows the system exposure to harm. The risk value is calculated for all identified threats using equation (3.1) provided by OWASP (OWASP, 2004).

$$\textbf{Risk = Threat Rating * Impact} \qquad \textbf{(3.1)}$$

In equation (3.1), Threat rating depicts the occurrence probability of threat whose value is taken directly from the previous step. The impact is the consequence of a successful exploit of the vulnerable point by threat. So, impact value is calculated by analyzing the assets affected by the occurrence of threat. Assets are the possession which needs to be protected from threats. We have associated a value with each assets showing its importance known as asset rating. Value for asset will vary from one application to another, values are assigned based on the importance of asset for the stakeholders by taking their view. Therefore, the impact would be the summation of rating of affected assets. Calculation of asset rating is explained below:

- **Calculation of Asset Rating.** Evaluation of asset is done to know its importance for the organization and protection of assets is the aim of our research. In most of the risk analysis methods such as CRAMM (CRAMM, 2005), CORAS (Braberl, Hogganvik, Lund, Stølen, & Vraalsen, 2007) assets are assigned value based on its importance for the organization. But, we conjecture that if asset values are allocated by taking the view of each concerned or involved stakeholders, it will be more accurate. Stakeholders may need different assets, and different stakeholder can use a single asset. Therefore, M: N mapping exist between the asset and stakeholder.

- **Rate the Assets.** Each direct stakeholder assigns a value on the scale of (0-10) to the involved assets. Values are assigned based on the asset importance and its criticality for the user. Scale to assign value is shown in Table 3.2. Hence, asset value assigned are similar to existing proposals but they are more accurate as they have been validated by multiple actors. For illustration consider an asset 'User Login Information' which is rated by the Stakeholder's Customer as (8), Bank Employee as (6) and Bank DBMS as (6).

Table 3.2 Criteria for Assigning Asset Rating

Criticality	Asset Rating
Critical	9-10
Very Important	6-8
Important	4-5
Normal	2-3
No Influence	0

- **Calculate Final Assets Rating.** Final asset value is calculated by analyzing the view of involved stakeholders. It would give more accurate values for assets as they have been validated by multiple actors. For example, asset 'User Login Information' value comes to be '6' (calculated by taking an average of three values specified in above step).

- **Calculate Security Requirements Priority.** The security requirements priority is computed by using the security requirements value. Security Requirements value is computed by adding the risk values of threats mitigated by the corresponding security requirements. Security requirements value is computed as:

Case1: Simple Value – If security requirement is mitigating the single threat. Then, its value is simply the risk value of threat mitigated.

Case2: Complex Value – If the security requirement is mitigating more than one threat then the security requirement value will be computed by adding the risk values of corresponding threats mitigated.

Then, based on security requirements value priority of security requirements is decided, higher the security requirement value higher is the priority. After this finalized security requirement are documented for further processing.

- **Security Requirements Specification.** After the elicitation, analysis, and prioritization of Security Requirements, specification of security requirements is done by incorporating one more section after functional requirements. Existing SRS document prescribed by IEEE (IEEE 830, 1998) includes the Security under Specific Requirements, where they are specifying the factors to protect the software from accidental or malicious access, use, modification, destruction, or disclosure. As per IEEE standard, the Specific requirements includes:

  a) Utilize certain cryptographical techniques;

  b) Keep specific log or history of data sets;

  c) Assign certain functions to different modules;

  d) Restrict communications between some areas of the program;

  e) Check data integrity for critical variables.

  As mentioned in point (a) above specifying cryptographical algorithm here in requirements engineering phase may constrain the system. As requirements engineers are good in specifying the requirements not the design algorithm. So, specifying the algorithm here is not the good choice. As selection of algorithm is done by design engineer based on the design constraints, algorithm specified here in this phase may not be the optimal algorithm for implementation.

Proposed SRS format is compared with existing IEEE SRS format in Table 3.3.

Table 3.3 Comparison of SRS

IEEE 830 SRS Format	Improved SRS Format
1. Introduction	1. Introduction
1.1 Purpose	1.1 Purpose
1.2 Scope	1.2 Scope
1.3 Definitions, acronyms, and abbreviations	1.3 Definitions, acronyms, and abbreviations
1.4 References	1.4 References
1.5 Overview	1.5 Overview
2. Overall description	2. Overall description
2.1 Product perspective	2.1 Product perspective
2.2 Product functions	2.2 Product functions
2.3 User characteristics	2.3 User characteristics
2.4 Constraints	2.4 Constraints
2.5 Assumptions and dependencies	2.5 Assumptions and dependencies
3. Specific requirements	3. Specific requirements
3.1 External interfaces	3.1 External interfaces
3.2 Functions	3.2 Functions
3.3 Performance requirements	3.3 Performance requirements
3.4 Logical database requirements	3.4 Logical database requirements
3.5 Design constraints	3.5 Design constraints
3.5.1 Standards compliance	3.5.1 Standards compliance
3.6 Software system attributes	3.6 Software system attributes
3.6.1 Reliability	3.6.1 Reliability
3.6.2 Availability	3.6.2 Availability
3.6.3 Security	3.6.3 Maintainability
3.6.4 Maintainability	3.6.4 Portability
3.6.5 Portability	**3.7 Security Requirements**
	3.7.1 Vulnerabilities
	3.7.2 Threats
	3.7.3 Assets
	3.7.4 Risk

3.3 Security Design Engineering

This phase deals with designing a software structure that realizes the specified security requirements. Security measures are chosen here, to implement the identified security requirements to meet the desired/ required security level in the system. Security mechanisms are the popular algorithm such as cryptography algorithms, physical security mechanisms, etc. which are required to implement security services. Bad decisions made during the design phase can lead to design flaws that can leave the system vulnerable to security threats. Analysis of security mechanisms which exist to implement the security services is done on various domain constraints such as computational and communicational constraints, and threats they mitigate. Based on the analysis, most suitable algorithms for implementing security in the system are identified. Activities of this phase are depicted in Figure 3.4 and explained as follows:

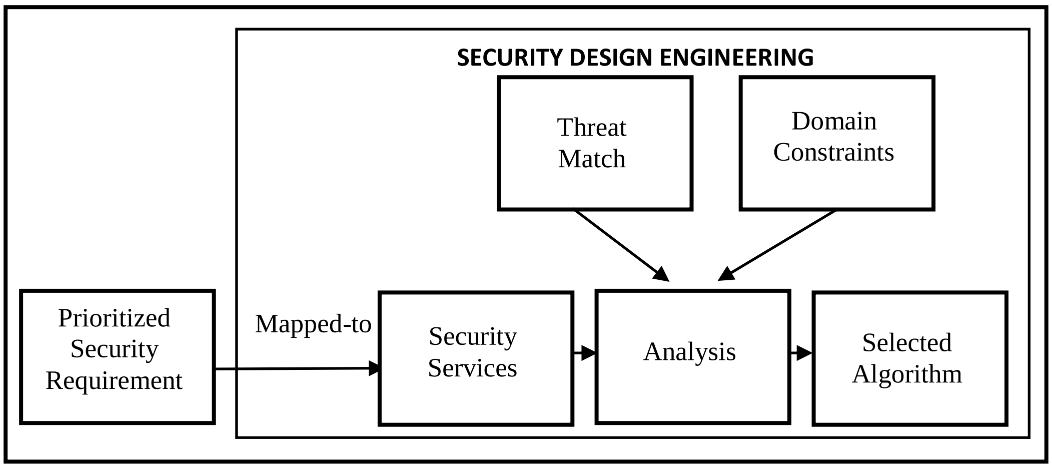


Figure 3.4. Security Design Engineering Process

- **Mapping of Security Requirements with Security Services.** Prioritized security requirements are mapped to the security services. Key security services are data confidentiality, data integrity, authentication, non-repudiation and access control (Forouzan, 2007). Mapping would help in the selection of suitable cryptographic technique pertaining to a particular class of security service.

- **Security Design Analysis.** Many techniques are available for the implementation of security services, so comprehensive evaluation of each is required for the selection of efficient algorithm. Analysis of algorithm depend on following:

  (i) Threat Match. Security algorithms are analyzed based on the threats they mitigate. To achieve this goal, a repository is created by studying and analyzing different cryptography algorithms (Forouzan, 2008) (Stallings, 2006). The repository will contain the detail of each algorithm regarding threat it mitigates and other threats. A sample repository for password based authentication in a wireless environment is shown in Table 3.4. So, from here algorithms with highest (maximum) mitigated threat match are selected.

Table 3.4 Repository of Algorithm

Threats	**Suitable Cryptography Algorithm**										
	Asymmetric Algorithm			**Symmetric Algorithm**			**Hashing Algorithm**		**Signature Algorithm**		
	RSA	**ECC**	**HECC**	**AES**	**DES**	**Triple DES**	**MD5**	**SHA1**	**RSA+ DSA**	**ECDSA**	**HECDSA**
DoS	Y	Y	Y	N	N	N	N	N	N	N	N
Impersonate	N	Y	Y	N	N	N	Y	Y	Y	N	N
Change Data	N	Y	Y	N	N	N	Y	Y	Y	Y	Y
Repudiate	N	N	N	N	N	N	N	N	Y	Y	Y
Password Cracking	N	Y	Y	N	N	N	Y	Y	Y	Y	Y
Spoofing	N	Y	Y	N	N	N	Y	Y	Y	N	N
Malicious Code	Y	Y	Y	N	N	N	Y	Y	N	N	N
Data Theft	N	Y	Y	N	N	N	Y	Y	Y	Y	Y
Disclose Data	N	Y	Y	N	N	N	Y	Y	Y	Y	Y
Data Deletion	Y	Y	Y	N	N	N	Y	Y	N	N	N
Sniffing	N	Y	Y	N	Y	Y	Y	Y	Y	Y	Y
Impact	**3**	**10**	**10**	**0**	**2**	**2**	**12**	**12**	**11**	**9**	**9**

Applicability of the threat/ attacks on an algorithm shows that whether the algorithm resists the attack or not. An entry ‘Y’ in Table 3.4 means that algorithm can resist the attack and ‘N’ entry shows algorithm is unable to resist the attack. Total Impact of each algorithm is calculated, which is the sum of ‘Y’ occurrences for each algorithm. The value of Total Impact would help in the identification of algorithms with highest threat match (high impact value). Similarly, we have calculated the impact of all the specified algorithms for authentication scheme.

- **Domain Constraints.** After threat analysis, algorithms with highest threat match are evaluated based on domain constraints. Domain constraints depend on implementation environment like wireless, mobile, mobile ad-hoc. Based on implementation environment two type of constraints are considered: (a) communicational constraints which are fundamental requirements for the system such as bandwidth, response time, throughput, power, etc. and (b) computational constraints which are related to resources involved in the computation such as memory, encryption speed, energy, etc. Both communicational and computational constraints are need to be considered while choosing the algorithm. Consideration of communication constraint in selection of algorithm and other efficiency parameters are supported in literature (Hankerson, Hernandez, & Meneze, 2000) (Maurice, Heemels, R. Teel, Wouw, & Dragan, 2010). For example, some algorithm need more power and bandwidth for its processing as compared to other, such as RSA require more power compared to ECC. Also change of environment causes change in communicational parameter values, such as sensor network has limited power and processing capability while the web-based system does not have any such constraint. Consideration of computational parameters is done to check the efficiency of selected algorithm which are impacted by communicational constraints. In short we can say that communicational parameters are the base of computational parameters. The computational and communication parameters are further categorized based on the type of device (low-end or high-end). Categorization of domain constraints is shown in Figure 3.5.

As these parameters are very critical for effective working of any security techniques. Every algorithm does not work in its optimal capacity under constrained environment. Such as Low-end devices have Power, Complexity, and Memory constraints whereas High-end devices do not have such constraint. Therefore, these parameters require careful consideration.

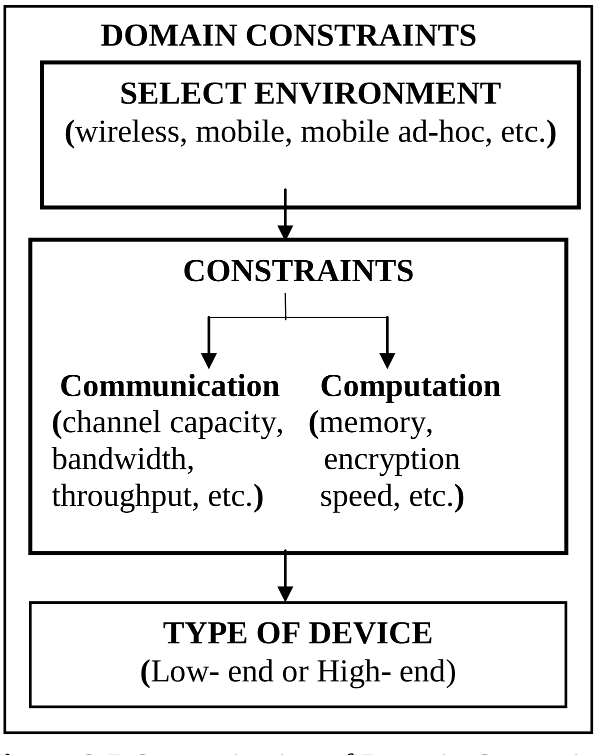

DOMAIN CONSTRAINTS
SELECT ENVIRONMENT
(wireless, mobile, mobile ad-hoc, etc.)
CONSTRAINTS
Communication
(channel capacity, bandwidth, throughput, etc.)
Computation
(memory, encryption speed, etc.)
TYPE OF DEVICE
(Low- end or High- end)

Figure 3.5 Categorization of Domain Constraints

- **Selection of Algorithm:** Based on the results from previous activities, suitable security algorithms are selected for implementation. As all threats cannot be mitigated by a single technique alone so, it needs to be used in conjunction with other techniques. After that, a design template is generated, the template would contain all the design phase related information like threats mitigated, constraints accounted for selection of algorithm, etc.

3.4 Security Testing

The process of security testing starts once the prioritized security requirements are implemented using suggested security algorithms. Security testing is the process to evaluate chosen security algorithms for implementing the prioritized security requirements. For the purpose of assessment, a metric (Security Index) is calculated which estimates the gap left in security. The metric value helps the software developer in deciding whether an enhancement or revision in the algorithm is required. Finally, a test report that contains all related information about security activities is generated. Proposed process of Security Testing is shown in Figure 3.6, and various activities of the process are as follows:

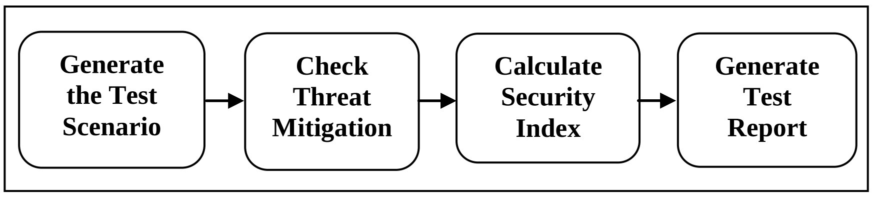


Figure 3.6 Process of Security Testing

- **Generate the Test Scenario.** Test scenarios are created from the sequence diagram drawn during the vulnerable point identification process. The test scenario is produced for all the functionality. It depicts the possible Threats on different

Vulnerable Points with assets affected and corresponding risk values. Continuing our example a sample test scenario for Login is shown in Figure 3.7.

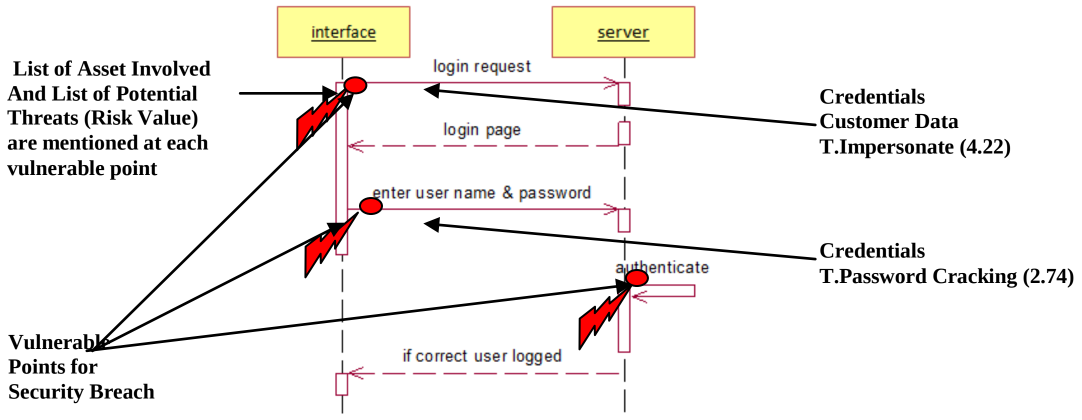

interface
server
List of Asset Involved And List of Potential Threats (Risk Value) are mentioned at each vulnerable point
login request
login page
Credentials
Customer Data
T.Impersonate (4.22)
enter user name & password
Credentials
T.Password Cracking (2.74)
authenticate
Vulnerable Points for Security Breach
if correct user logged

Figure 3.7 Test Scenario for 'Login' functionality

- **Check Threat Mitigation.** Threats identified for functionality is validated for its mitigation from the threat analysis of deployed cryptography algorithms. Here, all the remaining threats are considered as live/ active threats. Live threats analogous to functionality are assembled by analyzing the scenario diagram. A Vulnerability Metric (V_i) is calculated using equation (3.2) for each functionality.

$$V_i = \sum_{i=1}^{n} f(\textit{Risk Value of Live Threats for the Functionality } F_i) \qquad (3.2)$$

- **Calculate the Security Index.** Security index shows the gap in the security of the system. Gap in security is identified or calculated based on how many threats are mitigated and how many are live after the application of security algorithm. Security index is simply the ratio of summation of risk values of live threats to the total risk value of threats corresponding to functionality. Hence, mathematically equation to calculate Security Index is depicted by the proposed equation (3.3).

$$\boldsymbol{Security\ Index, SI} = \sum_{i=1}^{n} f\left(\frac{V_i}{R_i}\right) / n \qquad (3.3)$$

Where **V**$_i$ is the vulnerability metric of live threats for the functionality **F**$_i$

R$_i$ is the total risk value of functionality **F**$_i$

n is a number of Functionality considered.

The value of **security index will act as a quality parameter showing the threat proneness of the system.** Lower is the security index value higher will be the security and vice versa. Hence, the Security Index value indicates the effectiveness of chosen security algorithms. Next, the security index is compared with the Reference Value. The reference value is defined by the administrator based on the domain of application, level of CIA required, criticality of the system. Its value may change from one system to other based on its domain constraints.

If (SI ≤ Reference)

Then the system is in a **safe state.**

Otherwise, the system is in an **unsafe state**.

Unsafe state means design decision need to be modified, which can be handled by:

Case 1: Replace the existing algorithm with the new algorithm to mitigate the live threats.

Case 2: Choose a new algorithm and implement it along with the existing algorithm to handle the live threats.

SI value is said to be high or low based on the rating of threats which are alive inspite of security algorithm adopted. For example, (a) if threat ‘Password Cracking’ is alive and its risk value is ‘High’, then SI value is High. (b) if threat ‘Password Cracking’ is alive and its risk value is ‘Low’, then SI value is Low.

- **Generate Test Report.** The test report is generated for the system under test representing the summary of testing activities. The template will help the developer to decide further activities as it contains all the security related information. The template has fields like Name of Functionality under Test, Security Algorithms Applied, Threats Identified and a measure of Risk, Threats mitigated and Threats live, Result and Remarks. A sample test report for the system under test is shown in Figure 3.8.

System under Test	
Security Algorithms Applied	Efficient algorithms identified and chosen during security design engineering phase for implementation.
Threats Identified and value of Risk	List of threats at various vulnerable points with their calculated risk values.
Threats Mitigated	List of threats mitigated
Threats Live	List of threats left after application of security algorithms
Result	The value of Security Index
Remarks	Any suggestion or recommendation required for enhancing the level of security in the system

Figure 3.8 Sample Test Report

3.5 Case Study: Content Management System

In this section, the proposed framework is applied to a case study of **Content Management System** for a detailed explanation of our proposed framework. A computer application which provides the platform for the conception and adjustment of digital content is known as Content Management System (CMS). CMS specifies various features such as Web-based publishing, format management, history editing and version control, indexing, search, and retrieval. Various CMS are available such as Joomla, Drupal, Wordpress and others. Now each phase of our proposal is discussed in detail for CMS:

3.5.1 Security Requirements Engineering

- **Security Requirements Elicitation**
  - **Identify the Stakeholder.** Direct stakeholders for CMS are Vendor, System Developer, Author, Site User.
    - *Vendor.* Vendors are the CMS providers such as Wordpress, Joomla. They are providing the platform for development and receive payment for services provided.
    - *System Developer.* System developers are those who are making the website, blog, etc. using the services provided by vendor. System developers are paying to vendor for the services used.
    - *Author.* Authors are those who are creating, posting their blog on website and paying for it.
    - *Viewer/ Site User.* Site user is the one who is just using the provided features or reading, commenting, and liking the blog posts.

  - **Identify the Functional Requirements.** Different functional requirements, non-functional requirements, and associated assets are identified for all direct actors. Identified information related to actors are listed in Table 3.5.

Table 3.5 Functional and Non-Functional Requirements for CMS

Stakeholders	**Functional Requirements**	**Non-functional Requirements**	**Assets**
Vendor	1. Login/Logout 2. Update Profile 3. Change Account Password 4. Manage Developers 5. Manage Applications 6. Manage Content 7. Manage Content Category 8. Maintain Database 9. Transaction of Money	1. Reliability 2. Correctness 3. Robustness 4. Scalable 5. Integrity 6. Response time 7. Execution Time	• Login Details • Personal Information • Developers Information • Application Details • Content Details • Payment Information • IT Infrastructure
System Developer	1. Login/Logout 2. Update Profile 3. Change Account Password 4. Manage Comment 5. Manage Blog 6. Manage Web Page 7. Manage Authors 8. Manage Plugins 9. Transaction of money	1. Correctness 2. Response Time 3. Robustness 4. Scalable 5. Response Time	i. Login Details ii.Personal Information iii. Authors and User Information iv. Application Details v. Web Content vi. Payment Information vii. IT Infrastructure
Author	1. Login/Logout 2. Update Profile 3. Change Account Password 4. Create Content 5. Publish Content 6. Manage Comments 7. Transaction of Money	1. Reliability 2. Response Time 3. Integrity	i. Login Details ii. Personal Information iii. User Information iv. Payment Information v. IT Infrastructure
Viewer/Site User	1. Login/Logout 2. Update Profile 3. Change Account Password 4. View Content 5. Comment on Content 6. Like Content	1. Correctness 2. Response Time 3. Robustness 4. Response Time	1. Login Details 2. Viewer Details 3. Application Details 4. IT Infrastructure

- **Identification of Vulnerable Points:** Vulnerable points are identified for all functional requirements executed by direct actors specified in the previous activity. Identified vulnerabilities are depicted in Table 3.6.

Table 3.6 Vulnerabilities, Threats, Affected Assets and Security Requirements of CMS

SNo	Functionality	Sequence Trace	Vulnerabilities	Threats	Affected Assets	Security Requirement
			Vendor			
1	Login/ Logout	While Entering Details for Registration/ Login During Data Transmission While Updating Information While Verification While Retrieval	1. AAA 2. Insecure Interface 3. Remote Access 4. Communication Encryption 5. Lack of weak encryption of archive and data in transit	1. Password Cracking	Login Details (1)	Identification Authentication
2, 3	Update Profile, Change Account Password			1. Change Data 2. Data Theft 3. Impersonate 4. Deny Service 5. Malicious Code 6. Sniffing	Login Details (1,2,3,6) Personal Information (1,2,3,5,6) IT Infrastructure (3,4,5)	Authentication Authorization Integrity Immunity Intrusion Detection Privacy
4	Manage Developers	While Opening Dashboard While Updating/ Editing Information While Verification During Data Transmission While Retrieval	1. AAA 2. User Provisioning 3. User De-provisioning 4. Remote Access to management Interface 5. Communication Encryption Vulnerabilities 6. Lack of weak encryption of archive and data in transit 7. Lack of Security Awareness 8. Lack of Policy or Poor Procedures	1. Deny Service 2. Disclose Data 3. Repudiate 4. Data Theft 5. Social Engineer 6. Privilege Abuse 7. Management Interface Compromise 8. Sniffing 9. Change Data	Developer Information (2,3,4,5,6,8,9) IT Infrastructure (1,6,7)	Authentication Authorization Intrusion Detection Immunity Non-Repudiation Physical Protection Identification Privacy
5	Manage Applications			1. Deny Service 2. Disclose Data 3. Repudiate 4. Data Theft	Application Specific Data (1,2,3,4,5,6,7) IT Infrastructure (1,5)	Authentication Authorization Identification Physical

		During Information Processing While Storing Information While Serving to Enquiry	for Log Collection and Retention 9. Audit or Certification Not Available to Customer	5. Management Interface Compromise 6. Sniffing 7. Change Data		Protection Intrusion Detection Immunity Non-Repudiation
6, 7	Manage Content, Manage Content Category			1. Deny Service 2. Impersonate 3. Insider 4. Operational Log Compromise 5. Data Deletion 6. Disclose Data 7. Repudiate 8. Data Theft 9. Privilege Abuse 10. Sniffing 11. Malicious Code	Content (2,3,5,6,7,8,9,10,11) IT Infrastructure (1,4,9,11)	Authentication Authorization Privacy Intrusion Detection Immunity Non-Repudiation Integrity
8	Currency Transaction	While Entering Details for Registration/ Login During Data Transmission While Updating Information While Verification While Retrieval	1. AAA 2. Insecure Interface 3. Remote Access 4. Communication Encryption 5. Lack of weak encryption of archive and data in transit 6. Lack of Security Awareness 7. Lack of Policy or Poor Procedures for Log Collection and Retention	1. Password Cracking 2. Malicious Code 3. Disclose Data 4. Sniffing 5. Spoofing 6. Impersonate 7. Repudiate	Payment Details (2,3,4,5,6,7) Login Details (1,3,4,5,6) IT Infrastructure (2)	Identification Authentication Authorization Intrusion Detection Privacy Immunity Non-Repudiation
System Developer						
1	Login/ Logout	While Entering Details for Registration/ Login During Data Transmission While Updating	1. AAA 2. Insecure Interface 3. Remote Access 4. Communication Encryption 5. Lack of weak encryption of archive and data in transit 6. Lack of Security Awareness	1. Password Cracking	Login Details (1)	Identification Authentication
2, 3	Update Profile, Change Account Password			1. Change Data 2. Impersonate 3. Deny Service 4. Malicious Code 5. Sniffing 6. Data Theft	Login Details (1,2,5,6) Personal Information (1,2,5,6) IT Infrastructure (2,3,4)	Authentication Authorization Immunity Intrusion Detection Privacy

		Information While Verification While Retrieval	7. Lack of Policy or Poor Procedures for Log Collection and Retention			
4	Manage Authors	While Opening Dashboard While Updating/ Editing Information While Verification During Data Transmission While Retrieval During Information Processing While Storing Information While Serving to Enquiry	1. AAA 2. Remote Access to Management Interface 3. Communication Encryption Vulnerabilities 4. Lack of Standard Technologies and Solutions 5. Unclear Roles and Responsibility 6. Poor Enforcement of Role Definition	1. Deny Service 2. Disclose Data 3. Repudiate 4. Data Theft 5. Social Engineer 6. Privilege Abuse 7. Management Interface Compromise 8. Sniffing 9. Change Data	Developer Information (2,3,4,5,6,7,8,9) IT Infrastructure (1,4,6,7)	Authentication Authorization Identification Physical Protection Privacy Intrusion Detection Immunity Non-Repudiation
5,6,7,8	Manage Comment, Manage Blog, Manage Web Page, Manage Plugins			1. Impersonate 2. Repudiate 3. Data Theft 4. Data Deletion 5. Privilege Abuse 6. Malicious Code	IT Infrastructure (5,6) Content (3,4,5) Personal Information (6)	Authentication Non-Repudiation Authorization Integrity Intrusion Detection Immunity
9	Currency Transaction	While Entering Details for Registration/ Login During Data Transmission While Updating Information While Verification	1. AAA 2. Insecure Interface 3. Remote Access 4. Communication Encryption 5. Lack of weak encryption of archive and data in transit 6. Lack of Security Awareness 7. Lack of Policy or Poor Procedures for Log Collection and Retention	1. Password Cracking 2. Malicious Code 3. Disclose Data 4. Sniffing 5. Spoofing 6. Impersonate 7. Repudiate	Payment Details (2,3,4,5,6,7) Login Details (1,3,4,5,6)	Identification Authentication Authorization Intrusion Detection Immunity Privacy Non-Repudiation

		While Retrieval				
Author						
1	Login/ Logout	While Entering Details for Registration/ Login During Data Transmission While Updating Information While Verification While Retrieval	1. AAA 2. Insecure Interface 3. Remote Access 4. Communication Encryption 5. Lack of weak encryption of archive and data in transit 6. Lack of Security Awareness	1. Password Cracking	Login Details (1)	Identification Authentication
2, 3	Update Profile, Change Account Password			1. Impersonate 2. Deny Service 3. Malicious Code 4. Change Data 5. Sniffing	Login Details (1,2,5) Personal Information (1,3,4,5) IT Infrastructure (2)	Authentication Authorization Immunity Intrusion Detection Privacy
4,5, 6	Create Content, Publish Content, Manage Comments	While creation While communication While posting data on server Validation on server While updating While deletion	1. AAA 2. Inadequate Resource Provisioning and Investment in Infrastructure 3. Lack of Security Awareness 4. Inadequate Physical Security Procedures	1. Impersonate 2. Repudiate 3. Data Theft 4. Malicious Code 5. Change Data	IT Infrastructure (4) Content (1,2,3,5) Personal Information (1,2,3,5)	Authentication Authorization Non-Repudiation Immunity Intrusion Detection
7	Currency Transaction	While Entering Details for Registration/ Login During Data Transmission While Updating Information While	1. AAA 2. Insecure Interface 3. Remote Access 4. Communication Encryption 5. Lack of weak encryption of archive and data in transit 6. Lack of Security Awareness 7. Lack of Policy or Poor Procedures for Log Collection	1. Password Cracking 2. Malicious Code 3. Disclose Data 4. Sniffing 5. Spoofing 6. Impersonate 7. Repudiate	Payment Details (2,3,4,5,6,7) Login Details (1,3,5,6)	Identification Authentication Authorization Intrusion Detection Immunity Privacy Integrity Non-Repudiate

		Verification While Retrieval	and Retention			
Viewer/ Site User						
1	Login/ Logout	While Entering Details for Registration/ Login During Data Transmission While Updating Information While Verification While Retrieval	1. AAA 2. Insecure Interface 3. Remote Access 4. Communication Encryption 5. Lack of Security Awareness	1. Password Cracking	Login Details (1)	Identification Authentication
2, 3	Update Profile, Change Account Password			1. Impersonate 2. Deny Service 3. Malicious Code 4. Sniffing	Login Details (1,3) Personal Information (1,3) IT Infrastructure (2)	Authentication Authorization Immunity Intrusion Detection Privacy
4,5,	View Content, Like Content, Comment on Content	While opening the content While reading the content While submitting the like During transmission While entering the comment While transmission While saving	1. AAA 2. Lack of Security Awareness 3. Inadequate Physical Security Procedures 4. Remote Access to management Interface 5. Lack of Security Awareness 6. Communication Encryption Vulnerabilities	1. Deny Service 2. Repudiate 3. Impersonate 4. Sniffing 5. Data Theft 6. Malicious Code	IT Infrastructure (1,6) Content (2,3,4,5,6)	Authentication Authorization Non-Repudiation Privacy Intrusion Detection Immunity

- **Threats Identification and Evaluation.** Threats are identified for various vulnerable points, are shown in Table 3.6. After that, threats are evaluated, and Threat Rating is shown in Table 3.7. Threat Rating is calculated by analyzing the occurrence of a given threat at different vulnerable points.

- **Security Requirements Elicitation:** Security requirements are elicited to mitigate threats based on the knowledge of requirements engineer. Elicited security requirements for CMS are shown in Table 3.6.

Table 3.7 Threat and its Rating

Threats	**Threat Rating**
Deny Service	7
Password Cracking	1
Impersonate	2
Sniffing	2
Malicious Code	3
Disclose Data	1
Spoofing	3
Data Theft	5
Repudiate	6
Social Engineer	3
Privilege Abuse	5
Management Interface Compromise	7
Insider	4
Operational Log Compromise	6
Data Deletion	1
Change Data	5

3.5.2 Security Requirements Prioritization and Specification

- **Calculate Risk Value for each Threat.** Risk values for identified threats are calculated using equation (3.1), calculated risk value for threats are shown in Table 3.9.

- **Calculation of Asset Rating.** Calculation of assets rating require an evaluation of involved assets. Evaluation of threats are shown in Table 3.8.

Table 3.8 Assets Evaluation for Content Management System

Asset	**View of involved Stakeholders**				**Asset Value**
	Vendor	**System Developer**	**Author**	**Site User**	
Login Details	9	7	7	3	**7**
Personal Information	6	6	6	6	**6**
Payment Details	9	8	8	4	**8**
Developers Information	6	9	4	4	**6**
Application Details	7	8	8	5	**7**
Content Details	8	8	7	5	**7**
IT Infrastructure	7	6	6	5	**6**

- **Security Requirements Priority.** The priority value of each security requirement is shown in Table 3.9. First, the security requirements values are calculated which is the summation of risk value of threats mitigated by security requirements. Summation of security requirements value is done by considering the cases mentioned in section 3.2.2. Then, based on security requirements value, priority of security requirements are decided. Higher the security requirements value is higher is the priority.

  Prioritized security requirements are shown in Table 3.9.

Table 3.9 Risk Calculation and Security Requirements Prioritization

Security Requirements	**Threats**	**Threat Rating**	**Impact**	**Risk**	**Security Requirements Value**	**Priority**
Identification	Password Cracking	1	7	7	112	3
	Social Engineer	3	6	18		
	Management Interface Compromise	7	6	42		
	Spoofing	3	15	45		
Authentication	Disclose Data	1	15	15	199	1
	Impersonate	2	15	30		
	Deny Service	7	6	42		
	Password Cracking	1	7	7		
	Social Engineer	3	6	18		
	Spoofing	3	15	45		
	Management Interface Compromise	7	6	42		
Authorization	Deny Service	7	6	42	164	2
	Disclose Data	1	15	15		
	Data Theft	5	7	35		
	Data Deletion	1	7	7		
	Privilege Abuse	5	13	65		
Integrity	Data Deletion	1	7	7	7	8
Intrusion Detection	Sniffing	2	7	14	53	4
	Malicious Code	3	13	39		
Privacy	Sniffing	2	7	14	50	5
	Operational Log Compromise	6	6	36		
Immunity	Malicious Code	3	13	39	39	7
Physical Protection	Management Interface Compromise	7	6	42	42	6
Non Repudiation	Repudiate	6	7	42	42	6

3.5.3 Security Design Engineering

- **Mapping of Security Requirements with Security Services.** Mapping of security requirements to security services is shown in Table 3.10 to enable the start of security design engineering phase. Possible security algorithm to implement each security service is also mentioned in Table 3.10.

Table 3.10 Mapping of Security Requirements to Security Services

Security services	**Security requirement**	**Possible Security Algorithms**
Confidentiality	Privacy	Cryptography Techniques, Two- Factor Authentication, Multi- factor Authentication
	Immunity	
	Authentication	
	Identification	
Integrity	Integrity	Physical Protection Mechanism, Need-to-know Principle Enforcement, RnR Clarity
Non-Repudiation	Non-repudiation	Digital Signature, Notarization
Access Control	Intrusion Detection	Access Control Mechanism
	Authorization	

- **Security Design Analysis.** To achieve the goal of this activity Repository of threat for cryptographic algorithms is maintained. Threat analysis of algorithms for **password based authentication in a wireless environment** is shown in Table 3.11, from this repository algorithm with maximum threats match is selected for implementation.

Applicability of the attacks on an algorithm shows that whether the algorithm resists the attack or not. An entry ‘Y’ in Table 3.11 means that algorithm can resist the attack and ‘N’ entry shows algorithm is unable to resist the attack. Total Impact of each algorithm is calculated, which is the sum of ‘Y’ occurrences for each algorithm. The value of Total Impact would help in the identification of algorithms with highest threat match (high impact value). Similarly, we have calculated the impact of all the specified algorithms for authentication scheme.

Table 3.11 Threats Analysis Repository

Threats	**Suitable Cryptography Algorithm**										
	Asymmetric Algorithm			**Symmetric Algorithm**			**Hashing Algorithm**		**Signature Algorithm**		
	RSA	**ECC**	**HECC**	**AES**	**DES**	**Triple DES**	**MD5**	**SHA1**	**RSA+ DSA**	**ECDSA**	**HECDSA**
DoS	Y	Y	Y	N	N	N	N	N	N	N	N
Impersonate	N	Y	Y	N	N	N	Y	Y	Y	N	N
Change Data	N	Y	Y	N	N	N	Y	Y	Y	Y	Y
Repudiate	N	N	N	N	N	N	N	N	Y	Y	Y
Password Cracking	N	Y	Y	N	N	N	Y	Y	Y	Y	Y
Spoofing	N	Y	Y	N	N	N	Y	Y	Y	N	N
Malicious Code	Y	Y	Y	N	N	N	Y	Y	N	N	N
Data Theft	N	Y	Y	N	N	N	Y	Y	Y	Y	Y
Disclose Data	N	Y	Y	N	N	N	Y	Y	Y	Y	Y
Replay attacks	N	Y	Y	N	Y	Y	Y	Y	Y	Y	Y
Data Deletion	Y	Y	Y	N	N	N	Y	Y	N	N	N
Password Reuse	Y	Y	Y	N	N	N	Y	Y	Y	Y	Y
Sniffing	N	Y	Y	N	Y	Y	Y	Y	Y	Y	Y
Impact	**4**	**13**	**13**	**0**	**2**	**2**	**12**	**12**	**11**	**9**	**9**

- **Domain Constraints.** Evaluation of domain parameter is shown in Table 3.12. Here, the environment is considered to be wireless, and devices are high end.

- **Selection of Algorithm.** Based on above steps ECC is suggested for implementation among the available algorithms ECC, HECC, MD5, and SHA1. As MD5 and SHA1 are not appropriate for providing authentication. HECC is more complex and requires more computation time under given constraints as compared to ECC. Hence, ECC is suggested for Content Management System. Finally, the design template is generated, as shown in Table 3.13.

Table 3.12 Domain Constraint Analysis

Domain Attributes	**Priority (Low, Medium, High)**
Bandwidth	Medium
Response Time	Medium
Throughput	High
Power	Low
Memory	High
Encryption Speed	High
Energy	High
Power Consumption	Low
Network Availability	Medium
OS Independence	High
Compatibility	High

Table 3.13 Design Template

Design Template	
Security Mechanism	**Threats Mitigated**
ECC	Sniffing DoS Impersonate Change Data Password Cracking Spoofing Malicious Code Data Theft Disclose Data
Two Factor Authentication/ Multi-Factor Authentication	Unauthorized Access, Insider, Social Engineer
Physical Protection Mechanisms	Unauthorized Physical Access, Natural Disaster (to some extent)
Need-to-know Principle Enforcement	Insider, Social Engineer
Roles and Responsibility (RnR) Clarity	Privilege Abuse, Insider, Repudiate
Code Review Employ Secure Programming	Overflow Memory Corruption Bug Malicious Code

Analysis. Currently, the system is using password based authentication scheme which is vulnerable to password cracking and Change Data attacks. Here, our framework suggests public key cryptography scheme, Elliptic Curve

Cryptography (ECC) for implementation based on domain constraints. As content management system uses high-end devices where the security is the ultimate concern. In such cases, public key techniques are preferable for its small key size and high security. In this environment, ECC is one of the best public key technique for its small key size and high security for password based authentication.

3.5.4 Security Testing

- **Check Threat Mitigation.** Only threat 'Repudiate' is left unmitigated. Hence, the Vulnerability Metric (V_i) value is calculated using equation (3.2) specified in section 3.4. The value of V would be for live threat 'Repudiate':

$$V = 115$$

- **Calculate the Security Index.** SI value is calculated using equation (3.3) specified in Section 3.3.

$$SI = (115/ 1043) * 100 = 11.02$$

  As the value of SI is 11% approximately which is high because 'Repudiate' is a high-risk threat. So to remove repudiate threat system need to incorporate a Digital Signature algorithm in the system. Hence, with Asymmetric Algorithm, 'ECC' system need to implement the Signature algorithm for complete protection of system from potential threats.

- **Generate Test Report.** Test report for Content Management System is shown in Figure 3.9.

Content Management System	
Security Algorithms Applied	ECC
Threats Identified and measure of Risk	Sniffing, DoS, Impersonate, Change Data, Password Cracking, Spoofing, Malicious Code, Data Theft, Disclose Data, Repudiate
Threats Mitigated	Sniffing, DoS, Impersonate, Change Data, Password Cracking, Spoofing, Malicious Code, Data Theft, Disclose Data
Threats Live	Repudiate
Result	SI value is 11% (approx.)
Remarks	Need to implement the Signature algorithm for complete protection of system

Figure 3.9 Test Report for Content Management System

3.6 Case Study of Open Source Software: Wordpress

Various CMS are available in the market such as Joomla, Drupal, Wordpress and others. But here for further explanation we have chosen Wordpress because it is the most widely used CMS by websites as reported in (W3Techs). According to the sources approximately 28% of websites are developed using it (Bate, 2017).

Wordpress has evolved in the year 2003 with its initial version 0.70. Further, it has launched updated versions to incorporate new functionalities of the modern world along with the various patches for the reported security vulnerabilities. Since the inception of wordpress many vulnerabilities have been reported by the researchers working on it, which are then fixed by wordpress team. The count of vulnerabilities over the year for wordpress (CVE, 2004) is depicted in the Figure 3.10.

We have developed a website using wordpress version 2.1.5 for mobile repair shop. Some screenshots of website are shown in Figure 3.11- 3.17. Various software's are available of vulnerability scanning such as Nexpose, Acunetix, etc. but here we are using Acunetix. Scan result window is shown in Figure 3.18. Vulnerabilities found by scanning our developed website are shown in Table 3.14.

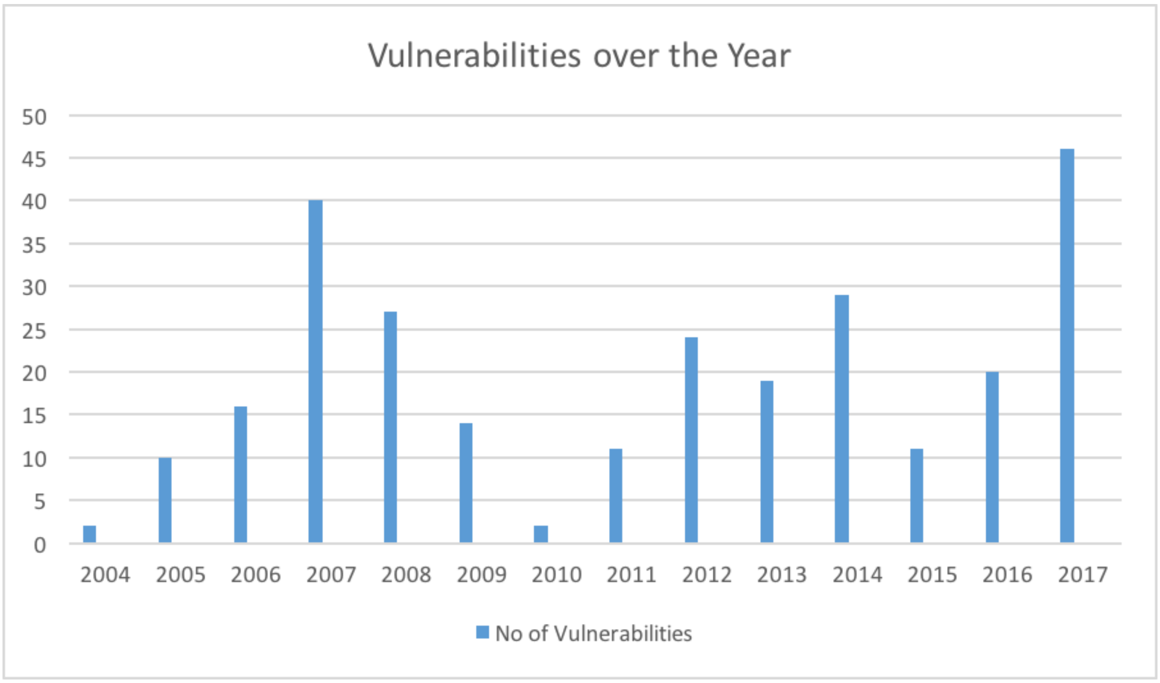
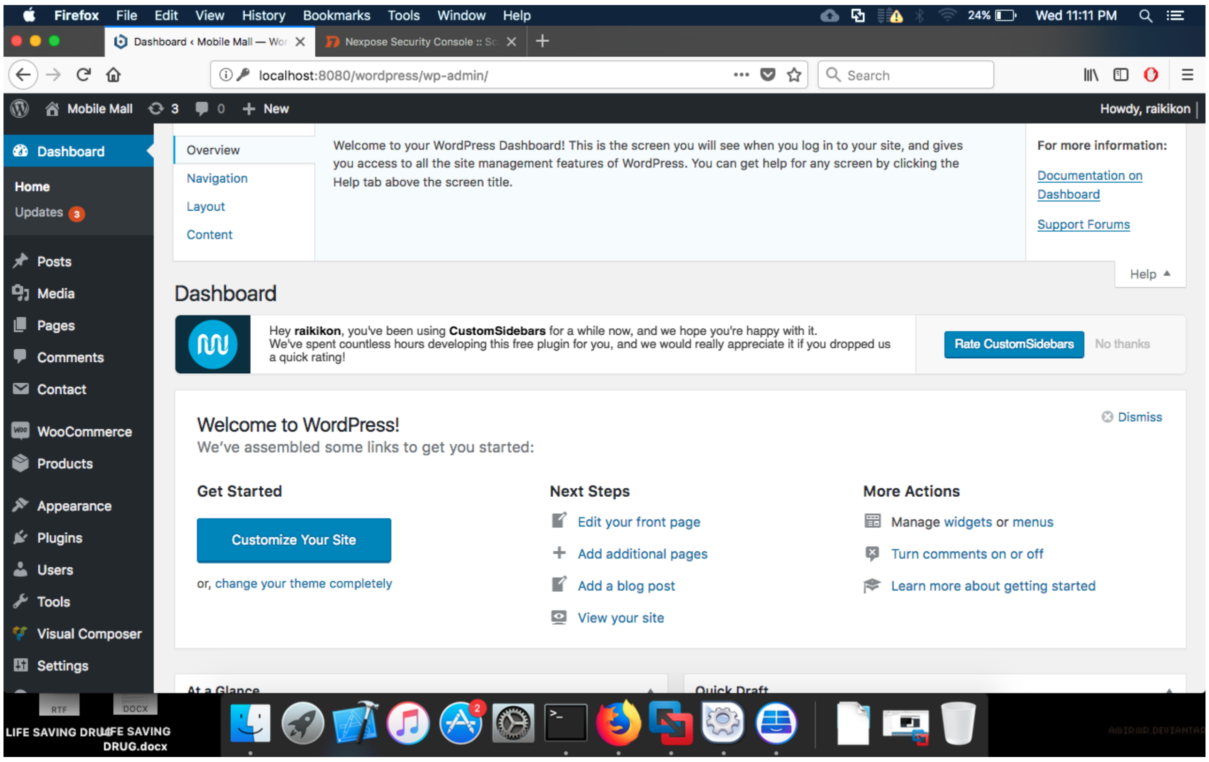


Figure 3.10 Vulnerabilities over the year for Wordpress



Welcome to your WordPress Dashboard! This is the screen you will see when you log in to your site, and gives you access to all the site management features of WordPress. You can get help for any screen by clicking the Help tab above the screen title.

Hey **raikikon**, you've been using **CustomSidebars** for a while now, and we hope you're happy with it. We've spent countless hours developing this free plugin for you, and we would really appreciate it if you dropped us a quick rating!

We've assembled some links to get you started:

Figure 3.11 Main Page of showing Dashboard

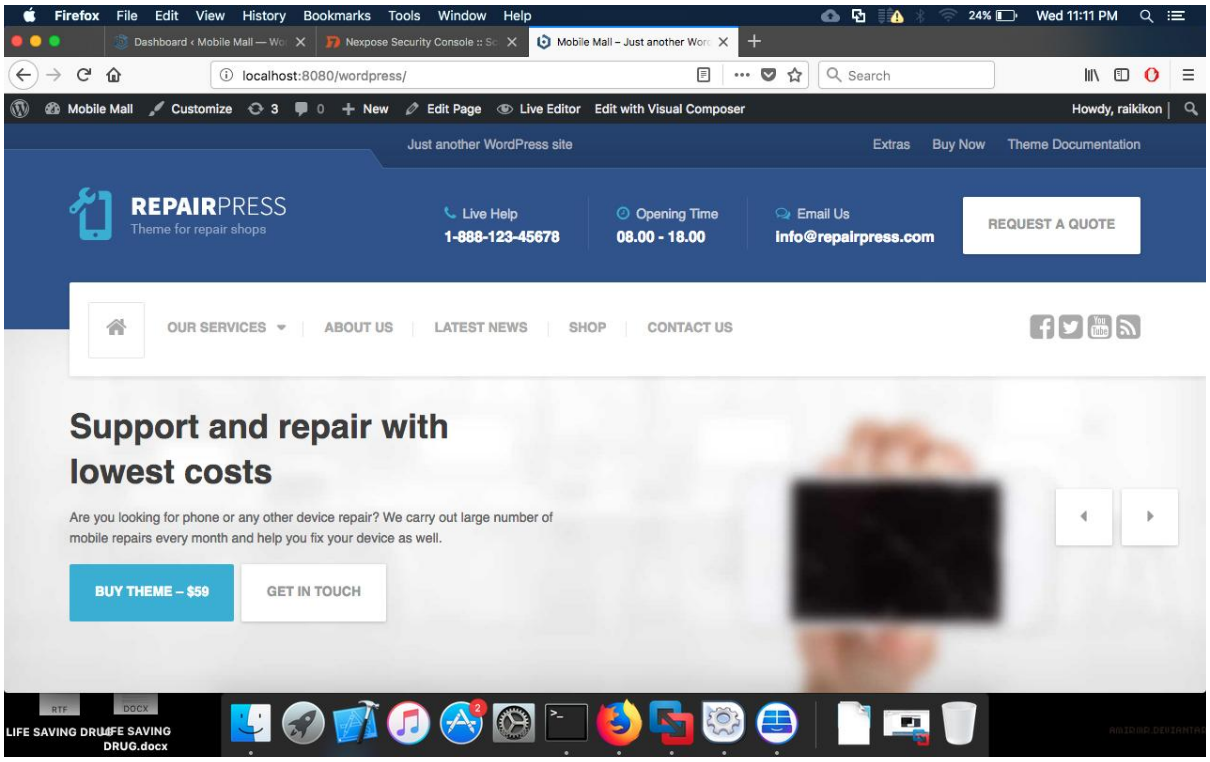
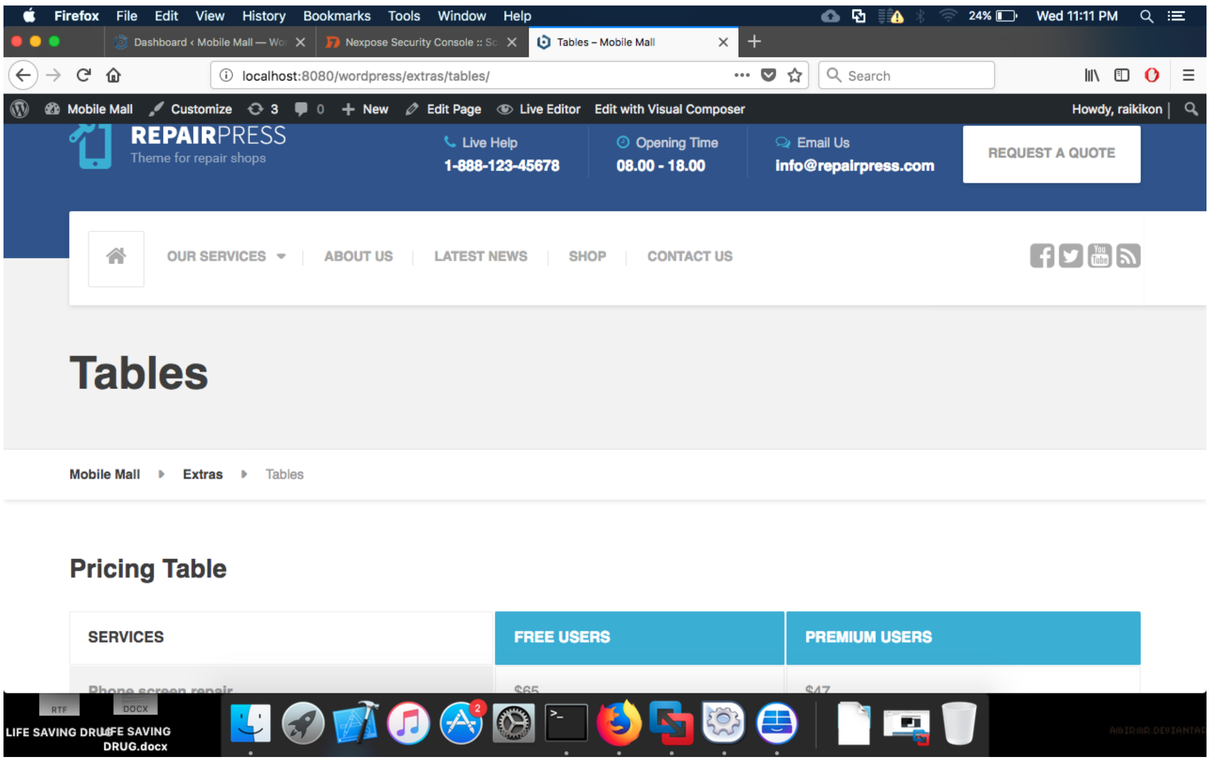

Firefox File Edit View History Bookmarks Tools Window Help
Wed 11:11 PM
Dashboard ‹ Mobile Mall — Wo
Nexpose Security Console :: Sc
Mobile Mall – Just another Wor
localhost:8080/wordpress/
Search
Mobile Mall Customize 3 0 New Edit Page Live Editor Edit with Visual Composer
Howdy, raikikon
Just another WordPress site
Extras Buy Now Theme Documentation
REPAIRPRESS
Theme for repair shops
Live Help 1-888-123-45678
Opening Time 08.00 - 18.00
Email Us info@repairpress.com
REQUEST A QUOTE
OUR SERVICES ABOUT US LATEST NEWS SHOP CONTACT US
Support and repair with lowest costs
Are you looking for phone or any other device repair? We carry out large number of mobile repairs every month and help you fix your device as well.
BUY THEME – $59
GET IN TOUCH

Figure 3.12 Theme Purchase for Repair on Website


Firefox File Edit View History Bookmarks Tools Window Help
Wed 11:11 PM
Dashboard ‹ Mobile Mall — Wo
Nexpose Security Console :: Sc
Tables – Mobile Mall
localhost:8080/wordpress/extras/tables/
Search
Mobile Mall Customize 3 0 New Edit Page Live Editor Edit with Visual Composer
Howdy, raikikon
REPAIRPRESS
Theme for repair shops
Live Help 1-888-123-45678
Opening Time 08.00 - 18.00
Email Us info@repairpress.com
REQUEST A QUOTE
OUR SERVICES ABOUT US LATEST NEWS SHOP CONTACT US
Tables
Mobile Mall Extras Tables
Pricing Table

SERVICES	FREE USERS	PREMIUM USERS
Phone screen repair	$65	$47

Figure 3.13 Showing the price information for different users

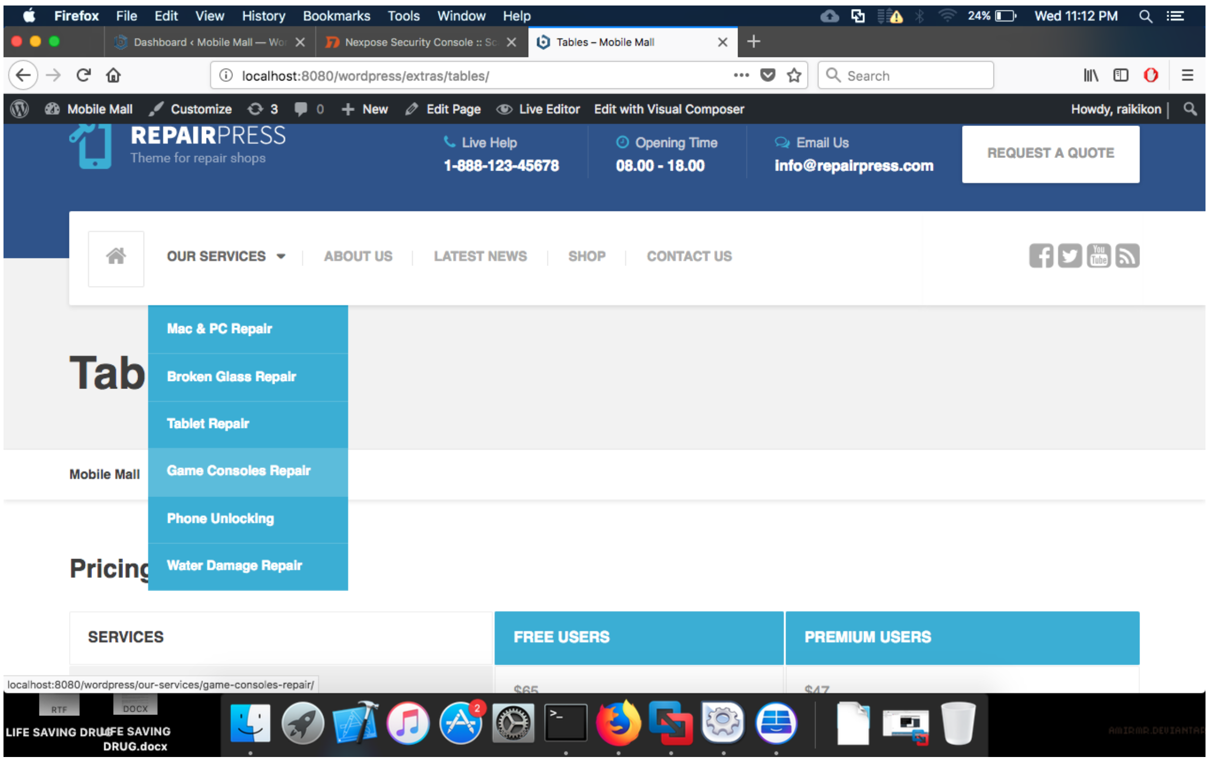


Figure 3.14 Showing Services provided by Website

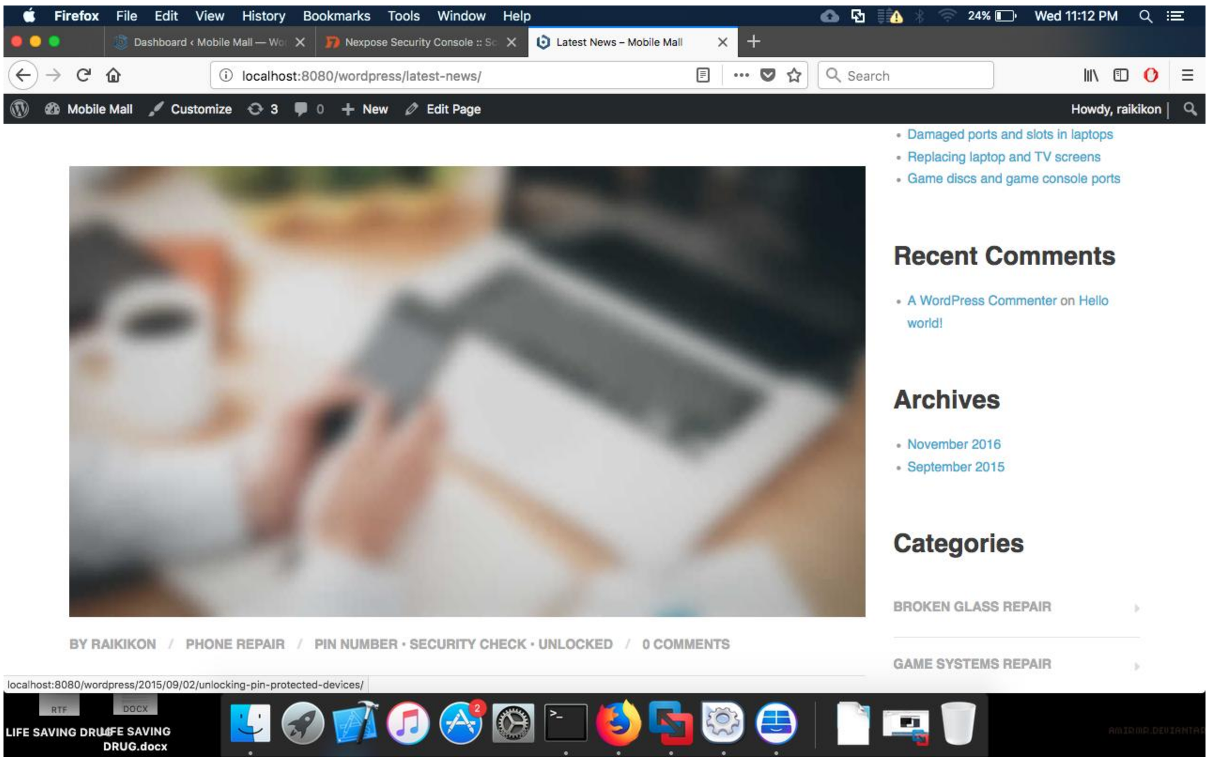

localhost:8080/wordpress/latest-news/
Damaged ports and slots in laptops
Replacing laptop and TV screens
Game discs and game console ports
Recent Comments
A WordPress Commenter on Hello world!
Archives
November 2016
September 2015
Categories
BROKEN GLASS REPAIR
GAME SYSTEMS REPAIR
BY RAIKIKON / PHONE REPAIR / PIN NUMBER • SECURITY CHECK • UNLOCKED / 0 COMMENTS
localhost:8080/wordpress/2015/09/02/unlocking-pin-protected-devices/

Figure 3.15 Another screen of website

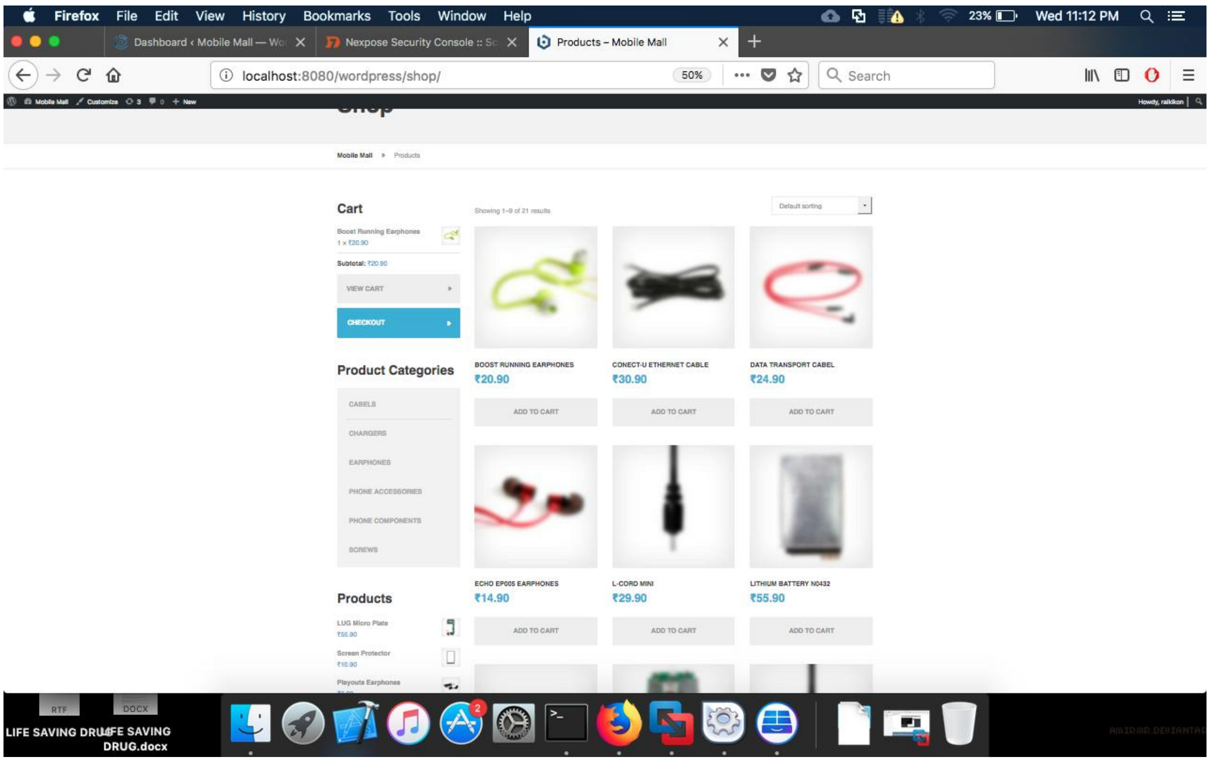


Figure 3.16 Screen Showing the Items available with its price

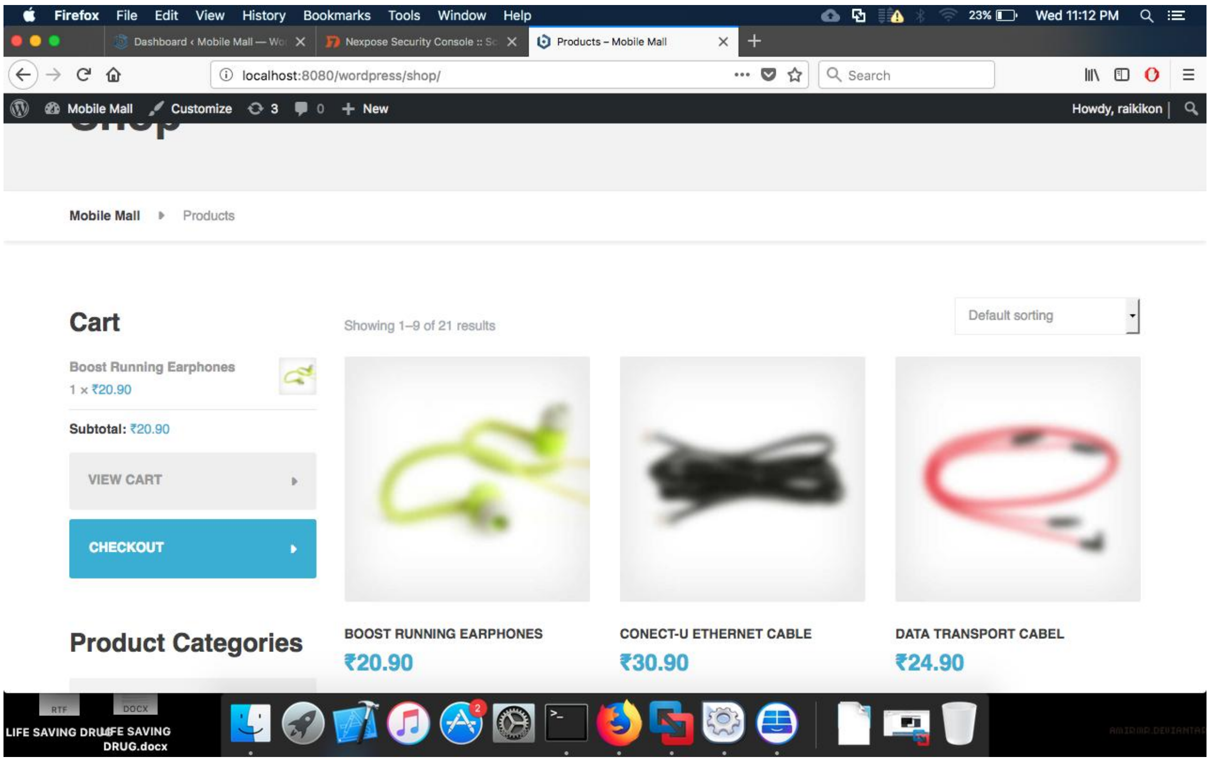
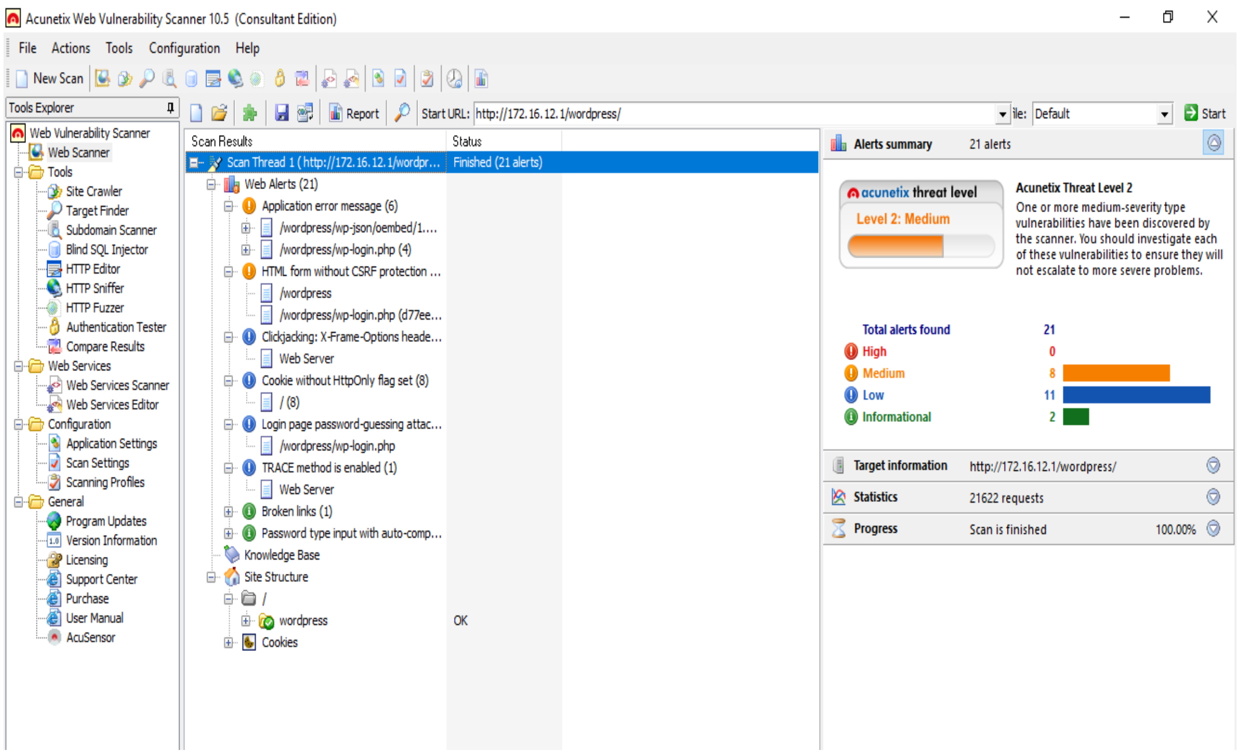


Figure 3.17 Checkout Screen



Start URL: http://172.16.12.1/wordpress/ Profile: Default

```
Scan Results                                         Status
Scan Thread 1 ( http://172.16.12.1/wordpr...         Finished (21 alerts)
  Web Alerts (21)
    Application error message (6)
      /wordpress/wp-json/oembed/1....
      /wordpress/wp-login.php (4)
    HTML form without CSRF protection ...
      /wordpress
      /wordpress/wp-login.php (d77ee...
    Clickjacking: X-Frame-Options heade...
      Web Server
    Cookie without HttpOnly flag set (8)
      / (8)
    Login page password-guessing attac...
      /wordpress/wp-login.php
    TRACE method is enabled (1)
      Web Server
    Broken links (1)
    Password type input with auto-comp...
  Knowledge Base
  Site Structure
    /
      wordpress                                      OK
    Cookies
```

Alerts summary 21 alerts

acunetix threat level — Level 2: Medium

Acunetix Threat Level 2
One or more medium-severity type vulnerabilities have been discovered by the scanner. You should investigate each of these vulnerabilities to ensure they will not escalate to more severe problems.

Total alerts found	21	
High	0	
Medium	8	🟧
Low	11	🟦
Informational	2	🟩

Target information	http://172.16.12.1/wordpress/	
Statistics	21622 requests	
Progress	Scan is finished	100.00%

Figure 3.18 Acunetix Scan Result Window

We have made a comparison of threats identified using our approach with the threats possible on previous versions of wordpress, threats reported by CVE for wordpress and threats present in our developed website using Acunetix, as shown in Table 3.14. it can be noted from table that, threats reported by CVE have already been accounted in our approach. In addition, we have identified more threats and suggested security mechanism to mitigate them.

Table 3.14 Threat Comparison

S.No	Possible Threats	Threats possible on previous versions of Wordpress	Threats reported by CVE for Wordpress	Threats identified by Acunetix on our website	Threats identified by Security Engineering Framework
1	Deny Service (DoS)	YES	YES	NO	YES
2	Password Cracking	YES	NO	YES	YES

3	Impersonate	**NO**	**NO**	**YES**	**YES**
4	Sniffing	**YES**	**YES**	**NO**	**YES**
5	Malicious Code*	**YES**	**YES**	**YES**	**YES**
6	Disclose Data	**NO**	**NO**	**NO**	**YES**
7	Spoofing	**NO**	**NO**	**NO**	**YES**
8	Data Theft	**YES**	**YES**	**YES**	**YES**
9	Repudiate	**YES**	**NO**	**NO**	**YES**
10	Social Engineer	**NO**	**YES**	**YES**	**YES**
11	Privilege Abuse	**YES**	**YES**	**NO**	**YES**
12	Management Interface Compromise	**NO**	**YES**	**NO**	**YES**
13	Insider	**NO**	**NO**	**NO**	**YES**
14	Operational Log Compromise	**NO**	**NO**	**NO**	**YES**
15	Data Deletion	**YES**	**NO**	**NO**	**YES**
16	Change Data	**YES**	**YES**	**NO**	**YES**
17	Password Reuse	**NO**	**NO**	**NO**	**NO**
18	MITM	**NO**	**NO**	**NO**	**NO**
19	Network Issues	**NO**	**NO**	**NO**	**NO**
20	DDoS	**NO**	**NO**	**NO**	**NO**
21	Sabotage	**NO**	**NO**	**NO**	**NO**
22	Security Log Compromise	**NO**	**NO**	**NO**	**NO**
23	Unauthorized Physical Access	**NO**	**NO**	**NO**	**NO**
24	Management Interface Compromise	**NO**	**NO**	**NO**	**NO**
25	Natural Disaster	**NO**	**NO**	**NO**	**NO**
26	Data Leakage	**NO**	**NO**	**NO**	**NO**
			Overflow** Memory Corruption**		

* Malicious Code: It refers to the change in the source code with an intention of security breach. It can be created by inserting SQL queries having untrusted data, or by exploiting an existing bug in the code, or by adding camouflaging XML scripts in dynamic web pages. Threats Code Execution, Sql Injection, XSS (Cross Site Scripting), Http Response Splitting, Cross Site Request Forgery (CSRF) are comes

under Malicious Code.

** Programming errors like Overflow and Memory Corruption can lead to abnormal working of the software systems. Presence of such errors can make system vulnerable to attack.

Threats on previous version are possible because initial version is not equipped with suitable/ optimal security mechanisms. For instance, password encryption is enabled in version 1.2. password protection of posts are also introduced in version 0.72, Unauthenticated Privilege Escalation is empowered in version 4.7.2, etc. **If they have used our approach these vulnerabilities have been avoided, as our approach suggests suitable security algorithms for implementation.**

Summary

- A process for security requirements engineering is presented where elicitation, analysis, prioritization and specification of Security Requirements is done along with the functional and non-functional requirements. Existing methodologies are not specifying the Security Recruitments explicitly. Our proposal of explicit specification of security requirements will improve the SRS document.

- Our proposal identifies threats and vulnerabilities in a structured manner which helps in uncovering the various security breaches possible in the software system. It can be seen from Table 3.14, threats identified by our approach covers almost all the threats listed by CVE, and also the threats identified by the scan report using tool Acunetix for our website developed using wordpress. Our proposal identifies almost all vulnerabilities beforehand which will save the resources required in release of new versions of software.

- As indicated in Figure 3.10, the number of vulnerabilities identified over the year for wordpress has increased whereas our proposal has identified all these vulnerabilities in the first phase of software development lifecycle. Hence, it can be said that if our proposal is used then all these efforts of incorporating security patches would have been saved.

- Based on prioritized and specified security requirements for given domain constraint near optimal security techniques are identified for deployment of security threats. The thesis also tests the deployed security mechanism for various threats and evaluate the Security Index (SI) showing the risk of live threats. This SI is checked with a reference value to test the suitability of the algorithm. If (SI ≥ Reference) then another acceptable algorithm can be chosen from the design phase.

- Hence, the given security engineering framework presents a methodology that can be embedded in the traditional software development proposals to embed security from requirements engineering phase. To cater the need of providing security we have developed a vulnerability threat mapping table. The mapping table will ease the requirements engineer in identifying the potential threats to the system.

- There is a software myth: “Software defects are high during the initial phases and further it will improve and later it become obsolete.” So if our framework is adopted in conjunction to phases of software development, high defect during the initial time span of software system will get reduced. This will ultimately improve the performance of the software system.

In this way, the first, second and third goals of thesis specified in Section 1.8 are successfully addressed. In later chapters’ thesis will try to show that the proposed framework can be applied to a cloud system, IoT systems, and big data databases.

Publication from this work:

(a) Jaiswal, S., and Gupta, D. (2016). Measuring Security: A step towards Enhancing Security of System. *International Journal of Information Systems in the Service Sector (IJISSS). (accepted)* **[ESCI, Scopus Indexed]**

(b) Gupta, D., Chatterjee, K., and Jaiswal, S. (2013). A Framework for Security Testing. *ICCSA-2013, published by Springer-Verlag in Lecture Notes for Computer Science.* **[Scopus Indexed]**

CHAPTER 4
SECURITY FRAMEWORK FOR CLOUD SYSTEM

With ever increasing demand for cloud computing services, the rate for the security threats has amplified drastically, and this cannot be overlooked. Cloud-based systems can be used for storing and processing highly confidential data. These threats create a chaotic situation which makes customer hesitant to develop the trust. The chapter starts with the brief overview of Cloud Computing Architecture to extract knowledge about different threats in the cloud system. It surfs the existing proposals to further establish the research gap in the cloud system. This knowledge is used to enhance generic security engineering framework to elicit, analyze, prioritize and specify the security requirements in the cloud system. Thereafter, the efficient algorithms are selected based on domain constraints and security index is calculated. The framework is illustrated for cloud-based storage systems.

4.1 Security Issues in Cloud

Providing security in cloud computing architecture is much more difficult and challenging as compared to a network system, because of underlying complex architecture that has a wider range of new concepts such as multi-location data placement, multi-tenancy, trust. Providing security in the cloud system requires effort from both cloud developer and cloud customer. Therefore, in this section, we see the generic architecture, and then the security issues present in cloud-based systems is discussed.

4.1.1 Cloud Architecture

Cloud architecture provided by NIST (Liu, et al., 2011) is depicted in Figure 4.1. Conceptual view of the architecture presents the involved actors their roles, how they communicate, and other necessary components of the cloud-based system.

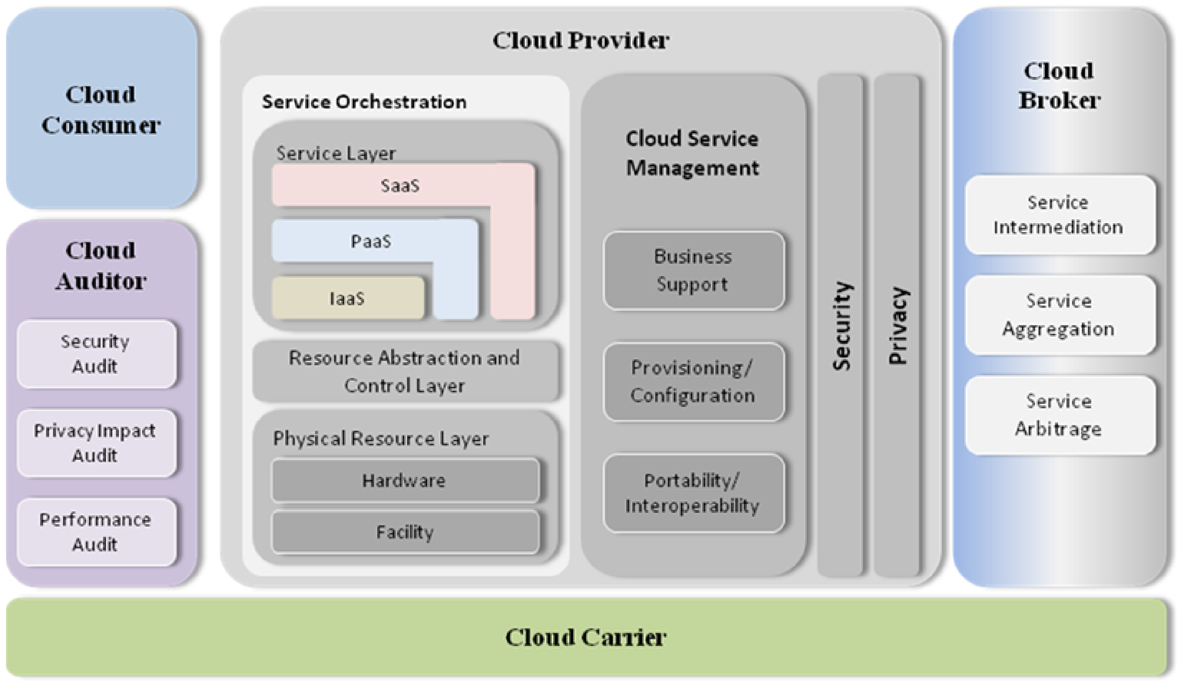


Figure 4.1 Cloud Reference Architecture (Liu, et al., 2011)

Involved Actors in Cloud

- *Cloud Customer:* Cloud customer is individual or organization who uses Cloud products and services and pay for it.
- *Cloud Provider:* Who owns, manage and operate the Cloud system to deliver services, and receive payment from Cloud customers for the services provided.
- *Cloud Broker:* Cloud Broker acts as the intermediary between customer and provider. It helps consumers through the complexity of cloud service offerings and may create value-added cloud services.

- *Cloud Auditor:* Cloud Auditor provides a valuable function to the government by conducting the independent performance and security monitoring of cloud services.
- *Cloud Carrier:* The Cloud Carrier is the organization which has the responsibility of transferring the data, somewhat akin to the power distributor for the electric grid.

Services Models of Cloud

- *Infrastructure as a Service (IaaS):* IaaS model provides the computing infrastructure like servers, network equipment, and software. The customer can install operating system images with applications to create their customized environment. The Cloud Service Provider (CSP) owns the hardware and is responsible for housing and maintaining them. Some known IaaS Clouds are Rackspace, Amazon EC2, Google Compute Engine, GoGrid.

- *Platform as a Service (PaaS):* PaaS provides the computing platform as on demand, which is used to develop and deploy applications. In addition to the computing platform, solution stack consisting of operating systems, programming language environment, databases and web servers is also provided. This model is suitable for developers and testers. The purpose of PaaS is to reduce the cost and complexity of buying and managing underlying hardware and software components. Clouds in this category are GAE, Force.com, Windows Azure Compute.

- *Software as a Service (SaaS):* SaaS provides access to application software to Cloud Service User (CSU) which is installed and maintained by CSP. Implementation and Deployment are hidden from the user, only limited set of configuration control is made available to the customer by the provider. Its principal advantage is the reduction in hardware cost and software development and maintenance cost. Major SaaS clouds are Microsoft Office 365, Quickbooks online, Salesforce.com

  Beside the above trivial services here one new service for storage is considered known as Storage as a Service (StaaS).

- *Storage as a Service (StaaS):* StaaS provides the storage requirements of various other service models. Here we are presenting it as a separate model because of its involvement in each basic service model. Moreover, its growing need gives rise to various security issues inherent in it. Security of storage is among the key challenges as mentioned in studies by various researchers (Grobauer, Walloschek, & Stocker, 2011) (Fernandes, Soares, Gomes, Freire, & In´acio, 2014) (Vu, Pham, Truong, Dustdar, & Asal, 2012) (Xiao & Xiao, 2013). Therefore, it must be considered separately then the trivial service models.

4.1.2 Security Issues

Cloud computing encompasses many technologies such as networks, operating systems, databases, resource scheduling, virtualization, transaction management, load balancing, concurrency control, and memory management. Security issues applicable to these existing technologies also applies to the cloud system.

Identification and handling of Security issues in cloud system have gained the attention of researchers because the security of customer's critical/ personal data is in the hands of someone else (providers, vendors, broker, etc.). Some important issues present in cloud system (Sen, 2013) (Jansen, 2011) (Pearson & Benameur, 2010) are as follows:

- **Shared Environment:** Shared environment of cloud computing is a potential point of attack; which may lead to Unauthorized Access to user's sensitive data and cause non-fulfillment of confidentiality and integrity goal.
- **Insecure Interfaces:** Interface of applications to other interacting application, software, and users are a vulnerable source of the attack, this may lead to limited monitoring capabilities, reusing credentials and password, improper authorization.
- **Multi-location Data Placement:** In a cloud system, client critical data is being stored at various sites across the globe. Multi-location data placement provides easy access and low-cost data recovery, but it may result in loss of data and integrity threats.
- **Scavenging:** Scavenging refers to the acquisition of leftover data from residue. It is a serious vulnerability that will give rise to password cracking threat.
- **Technological Obsolescence:** Use of antiquated and outdated technologies also leads to various threats.
- **Data Security:** In traditional system organizations are using on-premises application deployment model, where critical data is stored within the organizational boundary. However, in the cloud deployment, the organization data is stored with cloud providers at some remote location, which makes it a target for attack.

- **Privacy:** Privacy is another big concern, as the customer has to trust the cloud providers for providing security to their data.
- **Dependency and vendor lock-in:** Dependency on the provider is one of the major issues of cloud computing, if the customer wants to move from one provider to another, it could be cumbersome as there is a need to transfer complete data from one service provider to another.
- **Limited control and flexibility:** Most of the time users have limited control over functionality and execution of services since the application and services are provided by the third party and handled by them only.
- **Technical difficulties and downtime:** Since customers are linked to the cloud using the internet, so the problem in network connectivity will make the cloud set up useless, downtime is possible by even best cloud vendors.
- **Increased Vulnerability:** Nothing is secure on the internet; private data is the main target of intruders and hijackers. It increases the vulnerability by increasing the chance of data breach, data theft, data loss.
- **Network Security:** All the processing starting from registration to payment is all on the internet, hence need to secure this network from various attacks is necessary.
- **Trust:** Trust is a critical issue in cloud computing; in the present scenario, it depends on factors such as reputation, services, performance, security, and privacy. Providing trust in the system is a collective work of various involved actors in the cloud system. Study of trust in cloud depends on the study of links between the cloud users to cloud services (or providers) through those intermediary cloud entities (broker, administrator, auditors).

4.2 Existing Proposals for Security in Cloud-Based Systems

Various proposals (Stocker, Grobauer, & Walloschek, 2011) (Fernandes, Soares, Gomes, Freire, & In´acio, 2014) presents the survey on security issues presents in the cloud-based system. Also, (Fernandes, Soares, Gomes, Freire, & In´acio, 2014) have listed the available techniques to implement the security issues. In one of the early approach, Islam et al. (Islam, Mouratidis, & Edgar, 2011) has proposed a goal-oriented risk management technique to assess and manage risks which are barrier to the adaptation of cloud-based system in an organization. The process starts with identification of goals of the system. These goals are further refined and threat, vulnerabilities to goals are identified, and analyzed. Risk assessment is done to categorize the threats as Low, Medium, High, etc. Finally, treatment plans such as Encrypt the Confidential Data while storing, employ Strong Key Management policy, Check User Credentials, etc. are suggested without considering any domain constraints.

In proposal (Beckers, Cote, Fabbender, Heisel, & Hofbauer, 2013), the authors have developed information security management system that deals with security, privacy and legal compliance for cloud systems using ISO 27001 standard. The process starts with the identification of security goals to the assets of the system. Thereafter threats to assets are identified, and risk assessment is done. After risk assessment security policies of ISO 27001 are defined like assets management, human resources security, access control, physical and environmental security. They fail to explicitly specify the specific security algorithm used to implement the security policies.

A recent proposal by Naveed et al. (Naveed & Abbas, 2014) is using security framework by Haley et al. (Haley, Laney, Moffett, & Nuseibeh, 2008) to elicit and specify the security goals of the cloud system. Thereafter, security requirements are defined as a constraint to functionalities. The proposed framework helps the user in understanding the security need of the system; they are suggesting the control strategy as a constraint to system functionality. Similar to previous methods they have specified the control strategies without considering the domain constraints. In another proposal (Ficco, Palmieri, & Castiglione, 2015), authors have presented a framework for identification and specification of security requirements. After that, the broad measures are suggested for implementing security.

Conclusions drawn from existing proposals: In all the above proposals the security requirements are not specified explicitly. Further none of them considers the security issue multi-trust which is very important for the generation of trust among the cloud users. As it can be seen from cloud architecture threats are multifold compared to network systems. Therefore, it is hard to identify various assets, functionalities, vulnerabilities, and threats. If repositories for functionality-asset mapping and vulnerability-threat mapping can be developed, it would guide the user to identify the requirements efficiently and completely. In turn, security requirements can be elicited easily. None of them are suggesting the particular algorithms for implementation of the security requirements. Next section would present the novel framework for a cloud-based system.

4.3 Need for new Framework

As explained in the foregoing section, security issues in cloud based system are more complex compared to web based system. Hence, framework proposed in Chapter 3 is not directly applicable to cloud based system. It needs enhancement for the reasons mentioned below:

- In web- based systems stakeholders vary form one system too other and their functionalities vary from application to application. Whereas, in case of cloud-based system actors are broadly classified into five classes namely customer, user, auditor, provider, broker as defined by NIST (Liu, et al., 2011).
- The above mentioned stakeholders can execute or do some specific task on the assets of cloud system.
- Vulnerabilities in cloud- based system are more complicated and vast as compared to web- based system. Some vulnerabilities of cloud- based system are Hypervisor Vulnerability, Lack of Resource Isolation, Internal Cloud Network Probing, Synchronizing Responsibility or Contractual Obligation External to cloud, Cross- Cloud Application Creating Hidden Dependency, Certification Schemes Not Adapted to Cloud Infrastructure, Unclear Asset Ownership, Poor Provider Selection, etc.
- In addition to threats of web based system more threats are present in cloud system such as Conflict between Customer Provider Hardening Process in Cloud Environment, Lock-In, Loss of Governance, Compliance Challenges, Cloud Service Termination, Backup lost Stolen, etc.
- Our initial framework has limited set of threats and vulnerability. Hence, the database built for web- based system is not sufficient enough for cloud

environment. Also, to tackle the adoption need, new threats and vulnerabilities, a new security requirement Multi- Trust is proposed.

From above points we conclude that our initial framework needs modification to make it adaptable for cloud- based system. Modified framework to handle the security issues in cloud system is presented and discussed in next section.

4.4 Proposed Framework for Cloud-based System

As pointed out in the foregoing section, stakeholders are limited and they can execute some specific functionalities. Whereas, the list of assets, threats and vulnerabilities are exponential in number. Therefore, to elicit the security requirements in cloud system one need to explore the details of assets which gives rise to vulnerable points that is exploited by threats. Therefore, framework explained here for cloud based system will incorporate these features.

Architecture provided by NIST (Liu, et al., 2011), has identified five different actors namely cloud consumer, cloud provider, cloud broker, cloud auditor and cloud carrier. Here we are focusing only on the actor Customer, which can perform various tasks such as he can choose a provider and get registered for using services. Services span from availing software or hardware, storage space, to computing infrastructure.

The novel framework for handling security in a cloud system is depicted in Figure 4.2, which works in three different phases:

- Specification
- Prioritization

- Implementation and Validation

4.4.1 Specification

Similar to the activities of Security Requirements Engineering phase of framework presented in the previous chapter, here activities in this phase are executed along with the requirement engineering activities that builds on the understanding of the requirement engineers who are good at eliciting the functional requirements but are not well experienced when it comes to security requirements (Firesmith, Engineering Security Requirements, 2003). In this phase, security requirements are generated by following the activities shown in Figure 4.2.

- **Associate Assets with the Functionality:** Functionalities works on assets of the system, so all the assets required by functionality need to be identified. To achieve this, we have identified the possible functionalities and assets for cloud systems and developed a Functionality-Asset association table having a dimension (34X 22), shown in Table 4.1. Using the association table assets corresponding to functionalities are identified.

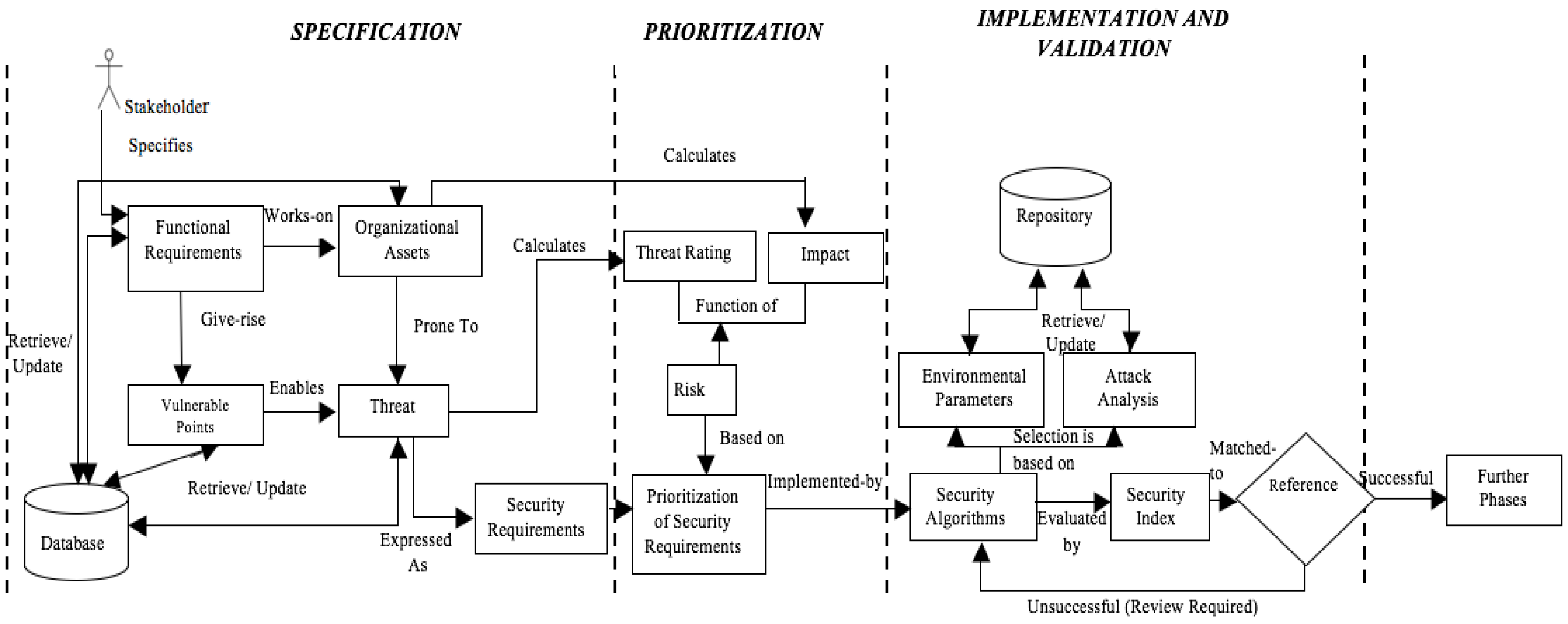


Figure 4.2 Proposed Framework for Cloud-Based Systems

Table 4.1 Functionality-Asset Association for Cloud-Based Systems

Assets →		1	2	3	4	5	6	7	8	9	10	11
Functionality ↓		Personal Sensitive Data	Personal Data	Personal Data Critical	HR Data	Service Delivery Real Time Services	Service Delivery	Access Control/ Authentication/ Authorization	Credentials	User Directory (Data)	Cloud Service Management Interface	Management Interface APIs
1	Registration	X	X									
2	Login/ Update Login	X	X									
3	Manage Encryption Key											
4	Store data	X	X		X	X	X		X	X		X
5	Download Data	X	X		X	X	X		X	X		X
6	Create Group of Users											
7	Share Data (with synchronization)	X	X		X	X	X		X	X		X
8	Manage Sharing											
9	Manage automatic backup								X			
10	Upgrade storage space								X			
11	Make/ Receive payment		X	X								
12	Log Collection											
13	Security Monitoring											
14	Maintenance and Management of Identity Management System	X	X		X							X
15	Maintenance and Management of Authentication platform (including enforcing password policy)	X	X		X							X
16	Maintenance and Management of Data	X	X		X							X
17	Data Traffic monitoring for security risk avoidance	X	X		X							X
18	Delete Data from Cloud	X	X	X	X				X			X
19	Migrate from one Cloud Provider to other	X	X	X	X				X			X
20	End of Subscription	X	X	X	X				X			X
21	Other Security Concerns (firewall rules, IDS/IPS tuning)	X	X		X	X	X		X	X		X
22	Manage Security Patch updates					X		X	X	X	X	
23	Upgrade Hardware/ Software need					X		X				
24	Multitenant Application Separation							X	X	X	X	
25	Physical Support Infrastructure, Security and Availability											X
26	Define Backup Strategy								X			
27	System Monitoring											X

Table 4.1 Continued

Assets → Functionality ↓		1	2	3	4	5	6	7	8	9	10	11
		Personal Sensitive Data	Personal Data	Personal Data Critical	HR Data	Service Delivery Real Time Services	Service Delivery	Access Control/ Authentication/ Authorization	Credentials	User Directory (Data)	Cloud Service Management Interface	Management Interface APIs
28	OS Patch management and Hardening Procedures											
29	Data Processing	X	X	X	X			X	X			
30	Manage Hardware and Software					X	X	X		X	X	
31	Manage Account	X	X	X								
32	Maintain SLA				X	X	X	X	X			
33	Join Group	X	X	X				X				
34	Un join the group	X	X	X				X				
	Assets Rating	10	7	8	8	10	6	8	8	10	5	8

Table 4.1 Continued

Assets → Functionality ↓		12	13	14	15	16	17	18	19	20	21	22
		Network (connections , etc.)	Physical Hardware	Buildings	CP Application (Source Code)	Operational Logs (Customer and Cloud Provider)	Security Logs	Backup and Archive Data	Company Reputation	Customer Trust	Employee Loyalty and Experience	Intellectual Property
1	Registration	X										
2	Login/ Update Login	X										
3	Manage Encryption Key	X										
4	Store data	X						X	X		X	
5	Download Data	X						X	X		X	
6	Create Group of Users	X							X		X	
7	Share Data (with synchronization)	X						X	X		X	
8	Manage Sharing											
9	Manage automatic backup							X				
10	Upgrade storage space							X				
11	Make/ Receive payment	X										
12	Log Collection											
13	Security Monitoring											

Table 4.1 Continued

14	Maintenance and Management of Identity Management System	X				X	X	X	X		X	
15	Maintenance and Management of Authentication platform (including enforcing password policy)	X				X	X	X	X		X	
16	Maintenance and Management of Data	X				X	X	X	X		X	
17	Data Traffic monitoring for security risk avoidance	X				X	X	X	X		X	
18	Delete Data from Cloud	X				X	X	X			X	
19	Migrate from one Cloud Provider to other	X				X	X	X			X	
20	End of Subscription	X				X	X	X			X	
21	Other Security Concerns (firewall rules, IDS/IPS tuning)	X	X			X	X	X	X		X	X
22	Manage Security Patch updates				X				X	X		
23	Upgrade Hardware/ Software need							X				
24	Multitenant Application Separation				X					X	X	
25	Physical Support Infrastructure, Security and Availability		X	X								
26	Define Backup Strategy					X	X	X				
27	System Monitoring		X	X					X	X	X	
28	OS Patch management and Hardening Procedures					X	X	X				
29	Data Processing	X				X	X	X				
30	Manage Hardware and Software											
31	Manage Account	X										
32	Maintain SLA									X	X	X
33	Join Group	X										
34	Un join the group	X										
	Assets Rating	10	5	6	8	6	6	10	10	9	8	8

- **Identify the Threats:** Threats are circumstances that have potential to cause harm to system assets. Threats occur at vulnerable points to cause damage to system assets. Threats to cloud systems are collected after doing the extensive literature review (Subashini & Kavitha, 2011) (ENISA, 2009) (Armbrust, et al., 2010) (CSA Cloud Security Alliance, 2010) and stored in the repository. The threat repository can be updated further if some new threat is identified. Threats corresponding to the vulnerable point are identified using, the developed Vulnerability-Threat mapping table of the dimension of (39 X 45) presented in Table 4.2. Mapping table as shown in Table 4.2 is constructed by reviewing various sources (ENISA, 2009) (CSA Cloud Security Alliance, 2010) (Cloud Security Alliance, CSA, 2013) and knowledge-based approach. The mapping table entry 'X' represents the threat occurrence at vulnerable points. The table contains all promising vulnerabilities that can be exploited by a threat. Therefore, each row of the table represents the list of all possible vulnerable points that a threat can exploit, and a column represents the threats that can occur at a vulnerable point. As various threats are present corresponding to the single vulnerable point, so threats applicable to the particular functionality is extracted based on the scenario of functionality created in the previous activity. Threats corresponding to vulnerability 'AAA' are Password Cracking, Impersonate, Password Reuse which is extracted from the Mapping Table 4.2.

Table 4.2 Vulnerabilities-Threats Mapping for Cloud Systems

Vulnerability →		1	2	3	4	5	6	7	8	9	10	11
Threats ↓		V.AAA	V.User Provisioning	V.User De-Provisioning	V.Remote Access to Management Interface	V.Hypervisor Vulnerabilities	V.Lack of Resource Isolation	V.Lack of Reputational Isolation	V.Communication Encryption Vulnerabilities	V.Lack of Weak Encryption of Archive and Data in Transit	V.Impossibility of Processing Data in encrypted form	V.Poor Key Management Procedures
1	T.Password Cracking	X										
2	T.Impersonate	X									X	
3	T.Sniffing								X			
4	T.Social Engineer		X				X		X			
5	T.Disclose Data	X										
6	T.Malicious code					X	X	X				
7	T.Repudiate	X										
8	T.Change Data								X	X		
9	T.Data Theft	X							X	X		
10	T.Password Reuse	X										
11	T.Insider	X										X
12	T.MITM	X							X	X		
13	T.Replay Attack								X	X		
14	T.Network Issues											
15	T.Resource Exhaustion											
16	T.DDoS											
17	T.Sabotage											
18	T.Conflict between Customer Provider Hardening Process and Cloud Environment											
19	T.LockIn											
20	T.Loss of Governance											
21	T.Operational Logs Compromise	X	X	X								
22	T.Security Log Compromise	X	X	X								
23	T.Data Deletion											
24	T.Supply Chain Failure											
25	T.Compliance Challenges											
26	T.Legal Issues						X					
27	T.Priviledge Abuse	X	X	X		X						
28	T.Backup Lost Stolen	X	X	X								
29	T.Unauthorized Physical Access											

Table 4.2 Continued

Threats ↓	Vulnerability →	1	2	3	4	5	6	7	8	9	10	11
		V.AAA	V.User Provisioning	V.User De-Provisioning	V.Remote Access to Management Interface	V.Hypervisor Vulnerabilities	V.Lack of Resource Isolation	V.Lack of Reputational Isolation	V.Communication Encryption Vulnerabilities	V.Lack of Weak Encryption of Archive and Data in Transit	V.Impossibility of Processing Data in encrypted form	V.Poor Key Management Procedures
30	T.Natural Disaster											
31	T.Malicious Probes Scans											
32	T.Data Leakage	X							X			X
33	T.Side Channel Attack											
34	T.Cloud Service Termination											
35	T.Loss of Encryption Keys										X	
36	T.Cloud Provider Acquisition											
37	T.Management Interface Coppmpromise	X			X							
38	T.Compromise Service Engine					X	X					
39	T.Modifying Network Traffic		X	X					X			

Table 4.2 Continued

Threats ↓	Vulnerability →	12	13	14	15	16	17	18	19	20	21	22	23
		V.Key Generation Low Entropy for Random Number Generation	V.Lack of Standard Technologies and Solutions	V.No Source Escrow Agreement	V.Inaccurate Modeling of Resource Usage	V.No Control of Vulnerability Assessment Process	V.Internal Cloud Network Probing	V.Co-Residence Checks	V.Lack of Forensic Readiness	V.Senstive Media Sanitization	V.Synchronizing Responsibility or Contractual Obligation External to Cloud	V.Cross-Cloud Applications Creating Hidden Dependency	V. SLA Clauses with Conflicting Promises to different Stakeholders
1	T.Password Cracking												
2	T.Impersonate												
3	T.Sniffing												
4	T.Social Engineer												
5	T.Disclose Data												
6	T.Malicious code												
7	T.Repudiate												
8	T.Change Data												
9	T.Data Theft												
10	T.Password Reuse												

Table 4.2 Continued

Vulnerability →		12	13	14	15	16	17	18	19	20	21	22	23
Threats ↓		V.Key Generation Low Entropy for Random Number Generation	V.Lack of Standard Technologies and Solutions	V.No Source Escrow Agreement	V.Inaccurate Modeling of Resource Usage	V.No Control of Vulnerability Assessment Process	V.Internal Cloud Network Probing	V.Co-Residence Checks	V.Lack of Forensic Readiness	V.Senstive Media Sanitization	V.Synchronizing Responsibility or Contractual Obligation External to Cloud	V.Cross-Cloud Applications Creating Hidden Dependency	V. SLA Clauses with Conflicting Promises to different Stakeholders
11	T.Insider												
12	T.MITM						X	X					
13	T.Replay Attack												
14	T.Network Issues										X		
15	T.Resource Exhaustion				X								
16	T.DDoS												
17	T.Sabotage												
18	T.Conflict between Customer Provider Hardening Process and Cloud Environment												X
19	T.LockIn		X										
20	T.Loss of Governance		X	X		X					X	X	X
21	T.Operational Logs Compromise								X				
22	T.Security Log Compromise								X				
23	T.Data Deletion									X			
24	T.Supply Chain Failure											X	
25	T.Compliance Challenges		X										
26	T.Legal Issues												
27	T.Priviledge Abuse												
28	T.Backup Lost Stolen												
29	T.Unauthorized Physical Access												
30	T.Natural Disaster												
31	T.Malicious Probes Scans						X	X					
32	T.Data Leakage						X	X					
33	T.Side Channel Attack						X	X					
34	T.Cloud Service Termination												
35	T.Loss of Encryption Keys	X											
36	T.Cloud Provider Acquisition												
37	T.Management Interface Copmpromise												
38	T.Compromise Service Engine												
39	T.Modifying Network Traffic					X							

Table 4.2 Continued

Vulnerability → / **Threats ↓**		24	25	26	27	28	29	30	31	32	33	34
		V.SLA Clauses Containing Excessive Business Risk	V.Audit or Certification not Available to Customer	V.Certification Schemes Not Adapted to Cloud Infrastructure	V.Inadequate Resource Provisioning and Investments in Infrastructures	V.No Policies for Resource Capping	V.Storage of Data in Multiple Jurisdiction and Lack of Transparency	V.Lack of information on Jurisdictions	V.Lack of Completeness and Transparency in Terms of Use	V.Lack of Security Awareness	V.Unclear Roles and Responsibilities	V.Poor Enforcement of Role Definitions
1	T.Password Cracking											
2	T.Impersonate											
3	T.Sniffing									X		
4	T.Social Engineer											
5	T.Disclose Data											
6	T.Malicious code											
7	T.Repudiate						X					
8	T.Change Data											
9	T.Data Theft											
10	T.Password Reuse											
11	T.Insider								X			
12	T.MITM											
13	T.Replay Attack				X							
14	T.Network Issues				X	X						
15	T.Resource Exhaustion					X						
16	T.DDoS					X						
17	T.Sabotage								X		X	
18	T.Conflict between Customer Provider Hardening Process and Cloud Environment								X			
19	T.LockIn		X	X			X	X	X		X	X
20	T.Loss of Governance											
21	T.Operational Logs Compromise											
22	T.Security Log Compromise											
23	T.Data Deletion								X			
24	T.Supply Chain Failure		X	X			X	X	X			
25	T.Compliance Challenges						X	X	X			
26	T.Legal Issues										X	X
27	T.Priviledge Abuse											
28	T.Backup Lost Stolen											
29	T.Unauthorized Physical Access											
30	T.Natural Disaster											

Table 4.2 Continued

Vulnerability → / **Threats ↓**		24	25	26	27	28	29	30	31	32	33	34
		V.SLA Clauses Containing Excessive Business Risk	V.Audit or Certification not Available to Customer	V.Certification Schemes Not Adapted to Cloud Infrastructure	V.Inadequate Resource Provisioning and Investments in Infrastructures	V.No Policies for Resource Capping	V.Storage of Data in Multiple Jurisdiction and Lack of Transparency	V.Lack of information on Jurisdictions	V.Lack of Completeness and Transparency in Terms of Use	V.Lack of Security Awareness	V.Unclear Roles and Responsibilities	V.Poor Enforcement of Role Definitions
31	T.Malicious Probes Scans											
32	T.Data Leakage											
33	T.Side Channel Attack								X			
34	T.Cloud Service Termination											
35	T.Loss of Encryption Keys								X			
36	T.Cloud Provider Acquisition											
37	T.Management Interface Copmpromise											
38	T.Compromise Service Engine											
39	T.Modifying Network Traffic											

Table 4.2 Continued

Vulnerability → / **Threats ↓**		35	36	37	38	39	40	41	42	43	44	45	
		V.Need-To-Know Principle Not Applied	V.Inadequate Physical Security Procedures	V.Misconfiguration	V.System/OS Vulnerabilities	V.Lack of Poor and Untested Business Continuity and Disaster Recovery Plan	V.Unclear Asset Ownership	V.Poor Provider Selection	V.Lack of Supplier Redundancy	V.Application Vulnerabilities or Poor Patch Management	V.Lack of Policy or Poor Procedures for Log Collection and Retention	V.Inadequate/ Misconfigured Filtering Resources	Threat Rating
1	T.Password Cracking												1
2	T.Impersonate												2
3	T.Sniffing												1
4	T.Social Engineer		X										5
5	T.Disclose Data												1
6	T.Malicious code												3
7	T.Repudiate												1

Table 4.2 Continued

Vulnerability → / Threats ↓		35	36	37	38	39	40	41	42	43	44	45	
		V.Need-To-Know Principle Not Applied	V.Inadequate Physical Security Procedures	V.Misconfiguration	V.System/OS Vulnerabilities	V.Lack of Poor and Untested Business Continuity and Disaster Recovery Plan	V.Unclear Asset Ownership	V.Poor Provider Selection	V.Lack of Supplier Redundancy	V.Application Vulnerabilities or Poor Patch Management	V.Lack of Policy or Poor Procedures for Log Collection and Retention	V.Inadequate/ Misconfigured Filtering Resources	Threat Rating
8	T.Change Data												3
9	T.Data Theft												3
10	T.Password Reuse												1
11	T.Insider												2
12	T.MITM												6
13	T.Replay Attack												2
14	T.Network Issues			X	X	X							5
15	T.Resource Exhaustion								X				4
16	T.DDoS			X	X							X	4
17	T.Sabotage												1
18	T.Conflict between Customer Provider Hardening Process and Cloud Environment												3
19	T.LockIn												2
20	T.Loss of Governance												13
21	T.Operational Logs Compromise				X						X		6
22	T.Security Log Compromise				X						X		6
23	T.Data Deletion												1
24	T.Supply Chain Failure							X	X				4
25	T.Compliance Challenges												6
26	T.Legal Issues												4
27	T.Priviledge Abuse	X		X									8
28	T.Backup Lost Stolen		X										4
29	T.Unauthorized Physical Access		X										1
30	T.Natural Disaster					X							1
31	T.Malicious Probes Scans												2
32	T.Data Leakage									X			6

Table 4.2 Continued

Vulnerability → / **Threats ↓**		35	36	37	38	39	40	41	42	43	44	45	
		V.Need-To-Know Principle Not Applied	V.Inadequate Physical Security Procedures	V.Misconfiguration	V.System/OS Vulnerabilities	V.Lack of Poor and Untested Business Continuity and Disaster Recovery Plan	V.Unclear Asset Ownership	V.Poor Provider Selection	V.Lack of Supplier Redundancy	V.Application Vulnerabilities or Poor Patch Management	V.Lack of Policy or Poor Procedures for Log Collection and Retention	V.Inadequate/ Misconfigured Filtering Resources	Threat Rating
33	T.Side Channel Attack												2
34	T.Cloud Service Termination							X	X				3
35	T.Loss of Encryption Keys												2
36	T.Cloud Provider Acquisition												1
37	T.Management Interface Copmpromise			X	X					X			5
38	T.Compromise Service Engine												2
39	T.Modifying Network Traffic												4

- **Security Requirements Specification:** According to Researcher Firesmith (Firesmith, Engineering Security Requirements, 2003), Security Requirements expresses what functionality is required to represent the threats of the system, twelve security requirements are defined by him for representing threats. Besides these twelve security requirements, one more security requirement “multi-trust” is added to the cluster for a cloud-based system. This security requirement would take care of any breach in SLA (Security Level Agreement) between the Cloud Provider and the Customer. Security Requirements are mapped to threats through analysis and experience. For instance, security requirements for threats ‘Password Cracking,’ ‘Impersonate,’ and ‘Password Reuse,’ are ‘Identification’ and ‘Authentication.’

4.4.2 Prioritization

Identified security requirements are prioritized using risk analysis method to get the ordered list of security requirements that are representing threats with high impact. One algorithm is not sufficient to implement all the security requirements. Therefore, prioritization of security requirements is done, and high priority security requirements are implemented/ handled first. Prioritization would help the developers and users in knowing which security requirement is more critical and need immediate focus. Activities of security requirements prioritization are shown in part two of Figure 4.2 and explained as follows:

- **Identify the Threat Rating.** Threat rating depicts the occurrence probability of the threat, or it is the rough measure of how likely a threat would exploit the vulnerabilities of the system to gain access to assets. Threat rating is calculated by checking number of vulnerabilities exploited by a threat in Vulnerability/ Threat mapping table shown in Table 4.2. Vulnerabilities exploited by a threat is represented by the presence of ‘X’ in a row of vulnerability/ threat mapping table, mathematically it is represented by equation (4.1). For instance, Threat Rating for threat ‘T.Password Cracking’ is ‘1’ as it exploits only vulnerability ‘V. AAA’.

Threat Rating = ∑ (number of occurrence of ‘X’ in a row of vulnerability-threat mapping table) (4.1)

- **Identify the Value of Impact.** Impact shows the consequence of a successful exploit of the vulnerable point. Impact value is calculated by analyzing the number of assets affected by the occurrence of a threat, as assets are the possession that needs to be protected from threats. Also, each asset has a value associated with it known as asset rating, showing its importance. Any exploit in the system would affect the assets so the impact would be the summation of assets rating represented by equation (4.2). Here assets rating is taken from the Table 4.1, the last row of the table depicts the asset rating. Assets rating is calculated by analyzing the use of the asset by functionalities (number of occurrence of ‘X’ in a column). Higher use of the asset by functional requirements depicts the higher rating of the asset.

  Impact = ∑ (Asset rating of affected assets by the Threat) (4.2)

- **Calculate Risk.** The risk is defined as the probability that a threat agent will exploit system vulnerability (weakness) and thereby create an effect detrimental to the system, or risk is an unwanted event that has some adverse consequences on the system. The value of risk is calculated for each threat using the equation (3.1) defined in Section 3.2.2 of Chapter 3.

The calculated risk values would act as threat value; more is the value of threat higher is its severity. Thereafter, the Threats are categorized based on their risk value; the category of threats is mentioned in Table 4.3. Categorization of threats is necessary to know more prevalent threats and handle them accordingly.

Table 4.3. Category of Threats Based On Risk Values

Category	**Risk Range**	**Need of Handling Threat**
Catastrophic	Risk $\geq$ 60	Threats are critical as they are impacting various (greater than three) high-value assets, so need urgent handling.
Important	60>Risk $\geq$ 20	These threats are important as they are impacting: • Various (greater than three) moderate assets • Either single high-value asset or two moderate assets So require careful consideration.
Tolerable	20> Risk $\geq$ 5	These threats are impacting single, or two average value assets so can be considered or ignored.
No Influence	Risk < 5	These threats are impacting single low-value asset, therefore ignore them.

- **Prioritize the Security Requirements.** Security requirements priority is calculated by adding together the risk values of threats represented by the security requirements under consideration, higher the security requirement value higher is the priority.

4.4.3 Implementation and Validation

In this phase, security algorithms are chosen to implement the security requirements based on the different domain constraints (communication, computational). Then the selected algorithm is validated by calculating the security index which shows the effectiveness of selected security algorithms. It consists of following steps as depicted in part three of Figure 4.2:

- **Mapping of Security Requirements with Security Services.** Key security services determined for cloud platforms are confidentiality, integrity, availability, non-repudiation, access control, auditability, and multi-trust. Auditability and multi-trust are new security services that are added to the cluster of existing security services defined in cryptography (Forouzan, 2007). Prioritized security requirements are mapped to one of the identified security services. Mapping would further help in the selection of security algorithm by specifying which cryptography techniques are more suitable for a particular scenario. Table 4.4 shows the mapping of security requirements to security services and possible security algorithms available to implement them.

Table 4.4. Mapping of Security Requirements with Security Services

Security services	**Security requirement**	**Possible Security Algorithms**
Confidentiality	Privacy	Cryptography Techniques, Two Factor Authentication
	Immunity	
	Authentication	
	Identification	
Integrity	Integrity	Physical Protection Mechanism, Need-to-know Principle Enforcement, RnR Clarity
Availability	Physical Protection	Vulnerability Assessment Tools, Physical Protection Mechanism, Key management protocol
	System Maintenance	
	Survivability	
Non-Repudiation	Non-repudiation	Digital Signature, Notarization
Access Control	Intrusion Detection	Access Control Mechanism
	Authorization	
Audit ability	Auditing	Auditing Mechanisms
Multi-Trust	Multi-Trust	RnR Clarity, SLA Strengthening, Data Portability

- **Security Design Analysis.** Many techniques are available for the implementation of security services, so comprehensive evaluation of each is required. Analysis of algorithm consists of following sub-activities:

    - **Threat Match.** Security algorithms are analyzed based on the threats they mitigate. To achieve this goal, a pre-defined repository is created by analyzing different algorithms. The repository contains the list of attacks mitigated and not- mitigated by algorithms. So, from here algorithms with highest threat match are selected.
    - **Domain Constraints.** After attack analysis, algorithms with highest threat match are evaluated on various domain parameters as mentioned in Section 3.3.

    Perceived value of domain constraints for different cloud-based service models for the wireless environment is shown in Table 4.5.

- **Selection of Algorithm:** Based on the results of previous activities, suitable security algorithm is selected for implementation. As all threats cannot be mitigated by a single technique alone so, it needs to be used in conjunction with other mitigation techniques and a design template is generated. The template would contain all the design phase related information like threats mitigated, constraints accounted for selection of algorithm, and others.

Table 4.5 Constraints for Cloud-Based System

Domain Attributes	**IaaS**		**PaaS**		**SaaS**			
					Cloud User		**Customer**	
	Complexity	Priority	Complexity	Priority	Complexity	Priority	Complexity	Priority
Runtime Performance	High	High	High	High	High	High	High	High
Low Memory Footprint	Medium	Medium	Medium	Medium	Medium	Medium	High	High
Power Consumption	Medium	Low	Medium	Low	Medium	Medium	High	High
Network Availability	High	High	High	High	High	High	High	Medium
Security Objectives	High	High	High	High	High	High	High	High
Scalable without affecting current functioning	High	High	High	High	High	High	High	Medium
OS Independence	Medium	High	High	High	High	Medium	Medium	High
Compatibility	High	High	High	High	High	Medium	High	Medium
Programmability	High	Low	High	High	Medium	Medium	Medium	Low
Cost of Chosen Solution	Medium	High	Medium	High	Medium	High	Medium	High

- **Validation.** Security algorithm chosen for implementation is validated to check if the potential threats are mitigated or not. For the purpose of validation, a Security Index value is calculated, which shows the gap in the security of the system. Security index is calculated using equation (3.2) and (3.3) mentioned in Section 3.4 of Chapter 3. Next, the SI value is compared with the reference value. The comparison is done in the same manner as mentioned in section 3.4 of Chapter 3.

After the selection of appropriate security algorithm, further, phases of software development are followed.

4.5 Application of Proposed Framework to Cloud Storage model

In this section methodology described in the previous section is applied and explained for a cloud-based storage system. Here an assumption is made that we are designing a new cloud-based storage system that provides the storage space to customers. Storage model is chosen for illustration of our proposal among the available service models because it is the most widely used service and requires high security. Everyone is using it knowingly or unknowingly, for instance, Google Drive is being used for information sharing and storage with Gmail. Here, functionalities are taken based on our study conducted for dropbox, google drive, mega, and one drive.

4.5.1 Security Requirements Specification

The security needs of the system are derived from functional requirements and represented in the form of security requirements. In a cloud-based storage system, different stakeholders are Cloud Customer, Cloud User, Cloud Service Provider. Functional requirements of all stakeholders are shown in Table 4.6. Only one actor that is Cloud Customer is considered for illustration of further activities, as cloud user functionalities are the subset of customer functionalities, the provider is serving to customer requests. Therefore, all security issues are dependent on the functionalities related to cloud customer. Functionality, asset, vulnerability, threats and security requirements for the system are shown in Table 4.7.

Table 4.6 Functional Requirements for different Cloud Actors

Actors	Functional Requirements	Non Functional Requirements
Cloud Customer	1. Registration & Login 2. Update Login Details 3. Store data into Cloud 4. Manage automatic backup 5. Manage Sharing with cloud user 6. Download data stored in the cloud 7. Select storage location 8. Make payment for services used 9. Maintenance of identity management system 10. Identity management system 11.Authentication platform management (including enforcing password policy) 12. Data and Traffic monitoring for security risk avoidance	1.Reliability 2. Less Response Time 3.Scalable 4.Correctness 5.Consistency 6. Recovery 7. Lawfulness of content 8. Compliance with data protection law 9. Personnel Security 10.Supply Chain Assurance
Cloud Users	1. Registration & Login 2.View shared data based on permission 3. Submit request to join the group 4.Unjoin a group	1. Reliability 2. Less Response Time 3. Scalable 4. Correctness 5. Consistency 6. Recovery
Cloud Service Provider	1. Manage Cloud Customer's Account 2. Manage Customer Data 3. Manage cloud hardware and software 4. Receive cloud usage payment 5. Maintain SLA 6. Data Processing 7. Physical support infrastructure, security and availability 8.OS patch management and hardening procedures 9.Security platform management and configuration (Firewall rules, IDS/IPS tuning, etc.) 10. Systems monitoring 11. Log Collection & security monitoring 12. Define Backup Strategy	1.Reliability 2. Integrity 3. Recovery 4.Performance 5. Data and Traffic monitoring for security risk avoidance 6.Personnel Security 7.Supply Chain Assurance 8.Scalability 9. Response Time 10. Restricted access to concerned cloud customer enterprise

Table 4.7 Vulnerabilities, Threats, Assets for Customers Functionalities

S.No	Functionality	Vulnerabilities Extracted	Threats	Affected Assets	Security Requirement
1.	Registration (new Customer)/ Login (Existing Customer) and Update Login Detail	AAA	1. Password Cracking 2. Impersonate 3. Password Reuse	Personal Data (2) Personal Sensitive Data (2) Credentials (1,3)	Identification Authentication
2.	Store data/ Download Data/ Share Data	Lack of Resource Isolation Lack of Reputational Isolation Communication Encryption Vulnerabilities Lack of or weak Encryption or Archives and Data in Transit Sensitive Media Sanitization Storage of Data in Multiple Jurisdictions and lack of Transparency about this Misconfiguration Lack of, or a poor and untested, Business Continuity and Disaster Recovery Plan	1. Social Engineer 2. Change Data 3. Sniffing 4. Data Theft 5. MITM 6. Replay Attack 7. Data Leakage 8. Modifying Network Traffic 9. Data Deletion 10. Loss of Governance 11. Compliance Challenge 12. Legal Issues 13. Network Issues 14. DDoS 15. Privilege Abuse 16. Natural Disaster 17. Loss of Encryption Keys	Company Reputation (2,10,11,12,16) Employee Loyalty and Experience (2,15) Personal Sensitive Data (3,4,5,7) Personal Data (3,4,5,7,17) HR Data (3,4,5) Service Delivery- real time services (14, 16) Service Delivery (14) Credentials (1,6) User Directory (data) (3,4,9,16) Cloud Service Management Interface (6,11) Network (connections, etc) (3,5,8,13,16) Backup or Archive Data (3,4,7,9,10,11,12,16,17)	Identification Authentication Immunity Integrity Intrusion Detection Privacy Survivability Multi-Trust System Maintenance Authorization Physical Protection
3.	Manage automatic backup	Sensitive Media Sanitization Untrusted Software	1. Data Deletion 2. Malicious Code	User Directory (data) (1,2) Backup Archive Data (1,2)	Integrity Immunity Intrusion Detection
4.	Upgrade storage space	No Policies for Resource Capping Resource Consumption Vulnerabilities	1. DDoS 2. Resource Exhaustion	User Directory (data) (1) Backup or Archive Data (1,2)	Immunity Survivability
5.	Make payment	AAA	1. Password Cracking 2. Impersonate 3. Password Reuse	Personal Data (2,4) Credentials (1,3)	Identification Authentication Authorization

			4. Insider		
6.	Maintenance and Management of Identity Management System/ Authentication platform (including enforcing password policy)/ Data and Traffic monitoring for security risk avoidance	AAA	1. Password Cracking 2. Impersonate 3. Disclose Data 4. Repudiate 5. Data Theft 6. Password Reuse 7. Insider 8. Operational Log Compromise 9. Security Log Compromise 10. Privilege Abuse 11. Data Leakage	Company Reputation (2,4) Employee Loyalty and Experience (3,4,7,10) Personal Sensitive Data (2,3,5,11) Personal Data (3,5,7,11) Credentials (1,2,4,6) User Directory (data) (5,11) Cloud Service Management Interface (14) Operational Logs (5,8) Security Logs (5,9) Backup or Archive Data (3,11)	Identification Authentication Privacy Immunity Non-Repudiation Authorization Security Auditing
7.	Delete Data from Cloud/ Migrate from one Cloud Provider to other/ End of Subscription	AAA User De-Provisioning	1. Repudiate 2. Data Theft 3. Password Reuse 4. Insider 5. Operational Log Compromise 6. Security Log Compromise 7. Backup Lost Stolen 8. Modifying Network Traffic	Employee Loyalty and Experience (1,4) Personal Sensitive Data (2,4,7) Personal Data (2,4,7) HR Data (2,7) Credentials (1,3) User Directory (data) (2,7) Operational Logs (5) Security Logs (6) Backup or Archive Data (2,7) Network (connections, etc) (8)	Non-Repudiation Privacy Authentication Authorization Security Auditing Integrity Physical Protection Intrusion Detection Survivability
8.	Other Security Concerns (not specific to functionality)	Impossibility of processing data in Encrypted format No Control on Vulnerability Assessment Process Cross Cloud Applications Creating Hidden Dependency SLA clauses with conflicting promises to different stakeholders SLA clauses containing excessive business Risk No policies for Resource Capping Lack of Information on Jurisdictions	1. Insider 2. Data Leakage 3. Loss of Governance 4. Modifying Network Traffic 5. Supply Chain Failure 6. Conflict between Customer Provider Hardening Process and Cloud Environment 7. Resource Exhaustion 8. DDoS 9. Sabotage 10. Compliance Challenges 11. Legal Issues 12. Privilege Abuse 13. Backup Lost Stolen 14. Unauthorized Physical Access 15. Network Issues 16. Operational Log	Company Reputation (3,5,7,10,11,18,19) Employee Loyalty and Experience (1,2,12) Personal Sensitive Data (1,13,18) Personal Data (1,2,13,18) HR Data (13,18) Service Delivery- real time services (5,6,8,19) Service Delivery (5,6,8) Credentials (19) User Directory (data)(9,13,18,20) Cloud Service Management Interface (10,19) Network (connections, etc) (4,15) Backup or Archive Data (10,11,13,18,20) Intellectual Property (11) Physical Hardware (7,9,14) Operational Logs	Authorization Immunity Multi Trust Intrusion Detection Survivability System Maintenance Integrity Physical Protection Security Auditing

		Unclear Roles and Responsibilities Poor Enforcement of Role Definitions Need- to- Know Principle Not Applied Inadequate Physical Security Procedures System or OS Vulnerabilities Lack of or incomplete or Inaccurate Asset Inventory Lack of or incomplete or Inaccurate Asset Classification Poor Identification of Project Requirements Application Vulnerabilities or Poor Patch Management Lack of Policy or poor Procedures for Logs Collection and Retention	Compromise 17.Security Log Compromise 18.Lock In 19.Management Interface Compromise 20.Side Channel Attack	(13,16) Security logs (13,17)	

4.5.2 Prioritization

Security requirements identified during the previous activity are now prioritized. Table 4.8 shows the priority value calculation for Identification Security requirement:

- Risk values for threats mitigated are calculated using equation (3.1) described in section 3.2.2, are shown in Table 4.8.

- All the risk values are summed together and assigned to Identification Security Requirement Value (SR Value) [10+60+50+64] = 184. Similarly, all other values are calculated, shown in Table 4.8; higher number represents higher priority and vice versa.

- Depending on SR Value Security Requirements Priority (SR Priority) is decided.

4.5.3 Implementation and Validation

Based on the identified and prioritized set of security requirements, most efficient algorithm to implement the security requirements is identified based on various domain constraints. It consists of following steps:

- **Mapping of Security Requirements with Security Services.** Security requirements are mapped to identified security services already shown in Table 4.4.

Table 4.8 Calculation of Security Requirement Priority

Security Requirements	Threats Mitigated	Threat Rating	Impact	Risk Value	SR Value	SR Priority
Identification	T.Password Cracking	1	10	10	184	7
	T.Impersonate	2	30	60		
	T.Social Engineer	5	10	50		
	T.Priviledge Abuse	8	8	64		
Authorization	T.Priviledge Abuse	8	8	64	119	5
	T.Insider	2	25	50		
	Unauthorized Physical Access	1	5	5		
Authentication	T.Password Cracking	1	10	10	60	2
	T.Password Reuse	1	10	10		
	T.Replay Attack	2	20	40		
Auditing	T.Operational Logs Compromise	6	6	36	72	3

	T.Security Log Compromise	6	6	36		
Integrity	T.Data Deletion	1	18	18	198	8
	Backup Lost Stolen	4	45	180		
Intrusion Detection	T.Malicious Code	3	18	54	303	11
	T.Sniffing	1	51	51		
	MITM	6	33	198		
Survivability	T.Loss of Encryption Keys	2	17	34	230	9
	Modifying Network Traffic	4	8	32		
	Supply Chain Failure	4	26	104		
	T.Resource Exhaustion	4	15	60		
Multi-Trust	T.Loss of Governance	13	20	260	622	13
	T.Compliance Challenges	6	30	180		
	T.Legal Issues	4	28	112		
	T.Lock In	2	35	70		
Immunity	T.DDoS	4	18	72	535	12
	T.Malicious Code	3	18	54		
	T.MITM	6	33	198		
	T.Data Leakage	6	27	162		
	T.Sabotage	1	13	13		
	T.Side Channel Attack	2	18	36		
Physical Protection	T.Backup Lost Stolen	4	45	180	231	10
	T.Natural Disaster	1	46	46		
	T.Unauthorized Physical Access	1	5	5		
System Maintenance	T.Conflict between Customer Provider Hardening Process and Cloud Environment	3	16	48	88	4
	T.Network Issues	5	8	40		
Non-Repudiation	T.Repudiate	1	28	28	28	1
Privacy	T.Data Theft	3	43	129	164	6
	T.Disclose Data	1	35	35		

- **Analysis and Selection**
    - **Threat Match.** Continuing our example based attack analysis result ECC/HECC algorithm will be chosen. Attack analysis for cloud-based system is shown in Table 4.9

Table 4.9 Attack Analysis Repository for Cloud Systems

Attacks	Suitable Cryptography Algorithm										
	Asymmetric Algorithm			Symmetric Algorithm			Hashing Algorithm		Signature Algorithm		
	RSA	ECC	HECC	AES	DES	Triple DES	MD5	SHA1	RSA + DSA	ECDSA	HECDSA
Data Leakage	Y	N	N	Y	Y	Y	N	N	Y	N	N
MITM	Y	N	N	Y	Y	Y	N	N	N	N	N
DDoS	N	N	N	Y	Y	Y	Y	Y	Y	Y	Y
Replay attacks	Y	N	N	Y	N	N	N	N	N	N	N
Side Channel attack	Y	N	N	Y	Y	Y	N	N	Y	Y	Y
Password Cracking	Y	N	N	Y	Y	Y	N	N	N	N	N
Impersonate	Y	N	N	Y	Y	Y	N	N	N	Y	Y
Password Reuse	Y	Y	Y	N	N	N	Y	Y	Y	Y	Y
Sniffing	Y	N	N	Y	Y	Y	N	N	N	N	N
Impact	**8**	**1**	**1**	**8**	**7**	**7**	**2**	**2**	**4**	**4**	**4**

- **Domain Constraints.** Different constraints pertaining to cloud systems are mentioned in Table 4.5.

- **Selection of Algorithm**
    - **Analysis of Algorithm.** Based on the domain attributes HECC algorithm is chosen for encryption among the ECC and HECC. As HECC is more efficient based on runtime constraints as compared to the algorithm ECC.

Moreover, various other algorithms are needed, and guidelines are made for providers and customers are listed in Table 4.10.

Table 4.10 Security Guidelines for Cloud-Based System

Security Algorithm	**Purpose**	**Threats Mitigated**	**Providers End**	**Customers End**
HECC	As a cryptography protocol to secure data from various threats	Data Leakage, MITM, DDoS, Replay attacks, Side Channel Attack, Password Cracking, Impersonate, Sniffing	The developer should implement the algorithm for protecting customer data.	Nothing to be done
Two Factor Authentication/ Multi-Factor Authentication	For authentication	Unauthorized Access, Insider, Social Engineer	To be implemented by providing another layer of security on authentication (mainly used when customer/ user is using different device other than the registered one for data access)	Nothing to be done
Diffie-hellman key exchange with Kerberos	For network authentication by allowing nodes to communicate over an insecure network to prove identity to each other in a secure manner.	Loss of Encryption Key	Must be implemented for session management and secure key exchange between the parties.	Nothing to be done
Physical Protection Mechanisms	For protecting physical resources from theft and physical assaults	Backup Lost Stolen, Unauthorized Physical Access, Natural Disaster (to some extent)	Must have physical protection algorithms such as retina scan, fingerprint scanning, and others to avoid theft and unauthorized physical access	The customer should also provide an algorithm at his end also to prevent any theft and loss because of physical assaults.
Need-to-know Principle Enforcement	For providing protection from malicious insider	Insider	What need to be done and how should be clearly defined.	Nothing to be done
Roles and Responsibility (RnR) Clarity	Roles and responsibilities should be clearly mentioned	Privilege Abuse, Insider, Repudiate	The developer should clearly define the role of each employee for proper working and mitigating threats like an insider, privilege abuse, unauthorized	The customer should also define the role of each user or employee as a client can be an organization.

			physical access and so on.	
Vulnerability Assessment Tools	For detection and protection from malware and cracking attempts	Password Cracking, Malicious Code, Password Reuse, Network Issues	Need to monitor any attempt to unauthorized access to customer data.	Customer must install some antivirus software to detect any malware attempt.
Auditing Algorithms (SSAE 16/ SAS-70)	For auditing purpose	Security Log and Operational Log Compromise	For Manual auditing	Nothing to be done
SLA Strengthening	Provide clauses for proper functioning	Loss of Governance, Compliance Challenges, Legal Issues, Lock In, Sniffing, Sabotage	Legal document and must be carefully designed by the cloud developer	must be carefully studied and understood by the cloud customer
Data Portability	Provided to mitigate LOCK-IN threat by using Standard set of APIs	Lock In	Should be provided using standard APIs and protocols for implementation, data storage, and replication	Nothing to be done
Quota	It is the restriction imposed by the government on goods that can be imported or exported during a particular period.	Resource Exhaustion	Developer should specify the limit that the customer cannot cross in the form of some terms and conditions	Nothing to be done

- **Validation.** Chosen security algorithms are mitigating all the threats.

$$\textbf{Security Index, SI} = \frac{0}{66} = 0 \qquad \textbf{(2)}$$

As the SI value is zero, need not to compare it with any reference value. If this is the case, the system is in a safe state.

4.6 Evaluation of the Existing Cloud Storage Models

We have studied existing cloud-based storage providers and evaluated them to know the level of security provided by them to users. Various security incidents and threats are reported in the past (Kuppuswamy & Al-Khalidi, 2014) (Dropbox: Yes, We Were

Hacked, 2012) (DropBox storage service, 2012) (Newton, 2011) on cloud storage systems.

Security algorithms implemented by different cloud storage providers are shown in Table 4.11. Here we are only considering the **Login and Store Data** functionalities. Now the security index value is calculated for each cloud storage specified in Table 4.11 (main focus is on the authentication and encryption part).

Table 4.11 Security Algorithm Employed by Cloud Storage Systems

Storage Service	**Personal Encryption (user manage their encryption key or password; provider is not able to access user data)**	**Transmission**	**Storage Encryption**	**2 Step Authentication**	**SI Value**	**Remarks**
DropBox (initial version)	No	AES-128, SSL/TLS	AES-256	No	71.5	SI very high
DropBox (later version)	No	AES-128, SSL/TLS	AES-256	Yes	18.7	SI is not acceptable
MEGA	Yes	AES-256 TLS and RSA	AES-128	No	0	SI is nil but if the user lost the encryption key he is not able to retrieve its data.
Google Drive	No	256-SSL/ TLS	AES-128	Yes	18.7	SI is not acceptable
iCloud	---	AES-128 SSL	AES-128	Yes	---	As all security algorithms are not known

a) Dropbox

Initial Version:

Dropbox provides TLS secure communication channel for both registration and login process. Customers are allowed to enter first and last name, email address and desired password during the registration process. Email address is used to login into Dropbox and password length should be of six characters. It shows the already registered e-mail address error warning to users during the registration process which in addition to weak password strength makes 'T.Password Cracking' threat easy and it would lead to 'T.Password Reuse' and 'T.Social Engineering' threats. It also does not send any activation emails after the registration to customers, resulting in 'T.Impersonate' threat possible on it.

It uses AES-128-bit encryption algorithm for the encryption of customer data stored on its servers, but only at server side using its encryption key to which the client is unaware. Hence, a 'T.Change Data' and 'T.Data Theft' threat applies on Dropbox.

$$\boldsymbol{SI} = \frac{\left(\frac{2.94 + 6.44 + 2.94}{2.94 + 6.44 + 2.94} + \frac{3.22 + 18.03 + 15.24}{3.22 + 18.03 + 15.24 + 13.9 + 11.4 + 6.01 + 5.12 + 11.4 + 4.34}\right)}{2}$$

SI = 0.715

The SI value is divided by two as we have considered two functionalities. Therefore, the security index comes to be a combination of two functionalities.

Therefore, SI is 0.715 that is 71.5% which is very high that shows system is not safe. As various security breaches are possible, so Dropbox has added new security feature in a later version.

Later Version:

$$\boldsymbol{SI} = \frac{\left(\frac{0}{2.94 + 6.44 + 2.94} + \frac{18.03 + 15.24}{3.22 + 18.03 + 15.24 + 13.9 + 11.4 + 6.01 + 5.12 + 11.4 + 4.34}\right)}{2}$$

SI = 0.187

Again the SI value is not low (18.7%), so we can say that if the control of encryption key is with provider system is susceptible to a security breach which is not acceptable.

b) MEGA

Security Index = 0/ 64.98 = 0

In the case of MEGA encryption control is with the user, so it is secure.

However, the problem with MEGA is that if the user forgets the key, he cannot make access to the storage system.

Hence from the Table 4.11 and calculated the value of security index, we can conclude that initial version of Dropbox lacks in security. Also, the later version is also not that much secure as it should be. As we can see from Table 4.11 that various other cloud providers are providing encryption to data at rest but are keeping the control of encryption key with them. That leads to severe threats and is not acceptable because we are storing our personal, private, financial data in these cloud storages.

4.7 Case Study of Open Source Software: ownCloud

ownCloud is a suite of client–server software for creating file hosting services and using them. Its functionality is very similar to the widely used Dropbox, with the primary functional difference being that the Server Edition of ownCloud is free and open-source, and thereby allowing anyone to install and operate it without charge on a private server. Various vulnerabilities are reported for ownCloud by CVE over the years (CVE Details, 2012). Frequency of vulnerability occurrence can be seen from the graph shown in Figure 4.3.

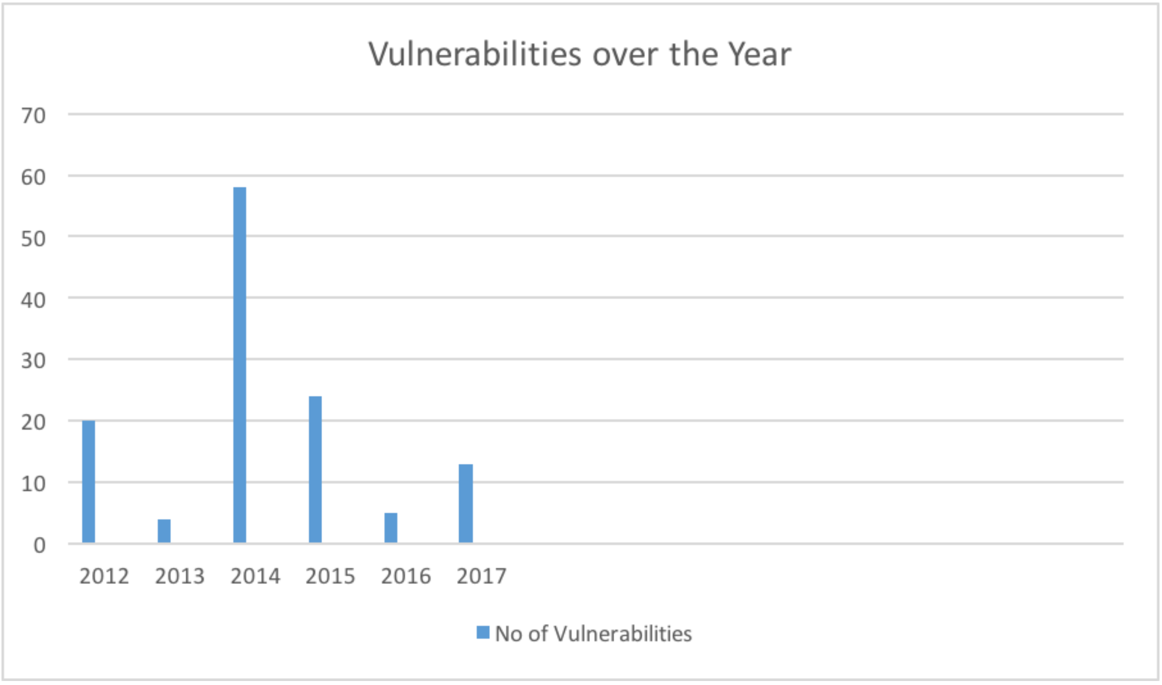


Figure 4.3 Vulnerability to ownCloud over the years

We have made a comparison of threats identified for our case study of cloud based storage system with the threats reported by CVE for ownCloud. Result of comparison is shown in Table 4.12.

Table 4.12 Threat Comparison

SNO.	Possible Threats	Threats reported by CVE for ownCloud	Threats identified Using Security Engineering Framework
1	Distributed Denial of Service (DDoS)	**NO**	**YES**
2	Denial of Service (DoS)	**YES**	**YES**
3	Password Cracking	**NO**	**YES**
4	Password Reuse	**NO**	**YES**
5	MITM	**NO**	**YES**
6	Replay Attack	**NO**	**YES**
7	Data Leakage	**NO**	**YES**
8	Impersonate	**NO**	**YES**
9	Lock In	**NO**	**YES**
10	Sniffing	**NO**	**YES**
11	Malicious Code*	**YES**	**YES**
12	Data Theft	**YES**	**YES**
13	Repudiate	**NO**	**YES**
14	Social Engineer	**YES**	**YES**
15	Privilege Abuse	**YES**	**YES**
16	Management Interface Compromise	**YES**	**YES**
17	Insider	**NO**	**YES**
18	Operational Log Compromise	**NO**	**YES**
19	Security Log Compromise	**NO**	**YES**
20	Data Deletion	**NO**	**YES**
21	Side Channel Attack	**NO**	**YES**
22	Change Data	**YES**	**YES**
23	Modifying Network Traffic	**NO**	**YES**
24	Loss of Governance	**NO**	**YES**
25	Compliance Challenges	**NO**	**YES**
26	Legal Issues	**NO**	**YES**
27	Network Issues	**NO**	**YES**
28	Natural Disaster	**NO**	**YES**
29	Loss of Encryption Keys	**NO**	**YES**
30	Resource Exhaustion	**NO**	**YES**
31	Backup Lost Stolen	**NO**	**YES**
32	Supply Chain Failure	**NO**	**YES**
33	Conflict between Customer Provider Hardening Process and	**NO**	**YES**

	Cloud Environment		
34	Sabotage	**NO**	**YES**
35	Unauthorized Physical Access	**NO**	**YES**
36	Disclose Data	**NO**	**NO**
37	Cloud Service Termination	**NO**	**NO**
38	Cloud Provider Acquisition	**NO**	**NO**
39	Compromise Service Engine	**NO**	**NO**

Malicious Code: It refers to the change in the source code with an intention of security breach. It can be created by inserting SQL queries having untrusted data, or by exploiting an existing bug in the code, or by adding camouflaging XML scripts in dynamic web pages. Threats Code Execution, Sql Injection, XSS (Cross Site Scripting), Http Response Splitting, Cross Site Request Forgery (CSRF) are comes under Malicious Code.

From the list of threats reported on ownCloud by CVE and the threats addressed by our framework. Threats identified by our approach is much more than the threats reported by CVE. In addition to this, proper security mechanisms are suggested to handle the reported/ identified threats. Hence, we can say that our framework aids the identification of threat and vulnerabilities at right time and try to handle them accordingly. It will also reduce the problems that may occur in future due to security negligence.

Summary

We have modified the generic framework of security engineering to enact novel security framework for cloud-based systems. The framework does its work in three phases that are specification, prioritization, and implementation & validation. In specification and prioritization phase security requirements are elicited, analyzed and prioritized. In implementation and validation phase optimal algorithm is selected based on domain constraints, and system security level is tested. We have illustrated our proposal for cloud storage model such as 'Dropbox,' 'MEGA,' and others.

Novel contributions of the Chapter:

The proposed framework differs from the framework proposed in foregoing chapter, as it has following distinguished features:

a) A **new security requirement named multi-trust is added** to the cluster of security requirements defined by researcher Firesmith.

b) During the requirements engineering phase to elicit the security requirements we have generated a **Functionality-Asset mapping table having a dimension (34X 22)** and a **vulnerability-threat mapping table of dimension of the (39 X 45)**. This would guide the user in handling a large number of assets, threats, vulnerabilities associated with actors.

c) Security algorithms are chosen considering various domain constraints (environmental consideration, communicational and computational parameters, and type of devices used) during the design engineering phase. For example, our proposal has recommended HECC algorithm for implementation based on domain constraints. Whereas currently available proposals like Dropbox has employed AES-128 bit encryption which is vulnerable to "T.Change Data" and "T.Data Theft" threats.

d) **An open source file hosting service provider ‘owncloud’ has been evaluated** for the purpose of validation of our proposal. And it is found that our proposal identifies more threats.

e) Also, **major cloud-based storage providers such as Dropbox, MEGA, Google Drive, iCloud were evaluated** for checking the security level by generating security metric. It was established from the case studies that our proposed framework can provide more security.

f) The proposed framework can be used for implementing security in the development of new cloud system.

Publications from the work

(a) **Shruti Jaiswal,** Daya Gupta, ***“Engineering and validating security to make cloud secure”,*** International Journal of System Assurance Engineering and Management, pp. 1-23, DOI: 10.1007/s13198-017-0612-x.]

(b) **Shruti Jaiswal**, Daya Gupta, ***“Security Requirements Engineering: A Challenge for Cloud System”***, in Proceedings of ‘Second International Conference on Emerging Research in Computing, Information, Communication and Applications’(ERCICA-14), published in **Elsevier** proceedings.

CHAPTER 5

SECURITY FRAMEWORK FOR IOT SYSTEMS

Internet of Things (IoT) is a network of physical objects embedded with sensors, software, and network connectivity, which enables them to collect and exchange data. IoT network allows sensing and controlling of objects remotely across existing network infrastructure. The Internet of Things (IoT) is an exhortation applied in various domains such as healthcare, education & research, home automation, manufacturing, and transportation, which contributes to our everyday life. Security is a serious concern in the IoT-based system. As a customer has to trust on the devices and a third party for management and protection of their confidential and private data from attacks. In this chapter, we first identify the assets at different layers of IoT system, then threats to assets and security requirements to mitigate the threats are identified. Based on domain constraints security algorithm is chosen and finally, system security is tested by generating a security index. The chapter starts with a brief discussion about IoT and security issues present in IoT-based systems. In the next part, novel security engineering framework is presented for IoT systems. Finally, the framework is explained for the IoT-based healthcare system.

5.1 Internet of Things

Internet of Things is a concept of future development of Internet intending to connect everyday objects to it. It is a proposed network of “things” or physical objects embedded with sensors, electronics, software and network connectivity enabling these devices to communicate. IoT enables objects/things/devices/sensors to be controlled and sensed remotely across the network, creating more opportunities for interaction between computer-based systems and physical world resulting in more business opportunities, economic benefit, and ease of living.

Kevin Ashton, a British entrepreneur, first used the term in 1999 while working at Auto-ID Labs (which was originally called Auto-ID centers) -referring to a network of connected objects globally, based on RFID (Radio-frequency identification). Reports published in (IDC, 2016, [online]) (O'Donnell, 2016, [online]) has predicted that IoT spending will reach $1.7 trillion by 2020. Gartner (Gartner, 2015) has forecasted that a hefty 21 billion IoT devices will be in place.

"Things" in Internet of Things, refers to a wide variety of sensors/devices like biochip transponders on farm animals, heart monitoring implants, automobiles with built-in sensors, electric clams in coastal waters, or field operation devices that help firefighters in rescue and search operations. These devices collect useful information with the help of several existing technologies. The collected data may flow autonomously between the devices. Present market examples include washer/dryers and smart thermostat systems having built-in sensors, actuators, internet connectivity and ability to exchange data using Wi-Fi or other technologies. Virtually all devices can be part of IoT which can connect to the internet but those that have sensing or actuating capabilities are preferred.

IoT will help organizations and industries in reducing cost by improving productivity, efficiency and resource utilization. By using IoT systems many tasks could be performed remotely without risk to human life, reduce travel time providing real-time insights, informed decisions can be taken. All these will help in making smarter decisions creating opportunities for people and industries.

As more and more things are connected to the Internet, it calls for a security check. Due to the constraints involved with IoT devices and the high stakes associated with the working of these devices. It can cause a lot of harm and financial loss if the security is not dealt early in the development cycle. Security is one of the key factors and concern area for the success of IoT (Miorandi, Sicari, Pellegrini, & Chlamtac, 2012) (Rose, Eldridge, & Chapin, 2015) (Trappe, Howard, & Moore, 2015).

5.1.1 IoT Architecture

Internet of things has a layered architecture shown in Figure 5.1, consisting of three layers namely user interface layer, network layer, and sensing layer.

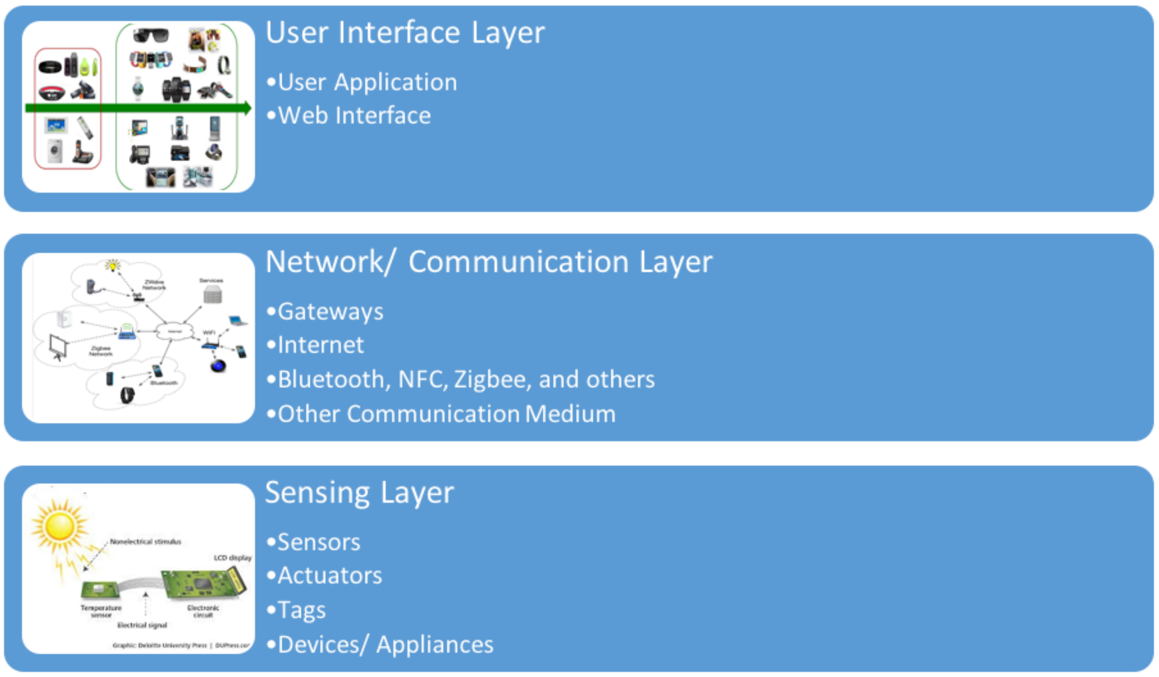


Figure 5.1. Architecture of IoT

Layer 1. Sensing Layer It consists of sensors, actuators, tags which are fitted in things to acquire data about the things characteristics and its environment. Sensors are the base of IoT systems as they collect the data and store it in the database for processing. Thereafter, processing of information is done as per the context of use, which varies from actor to actor. For instance, in a room, temperature sensors are fitted which senses the temperature every minute and store it in the local database. Upper layers can use the collected data in a different context such as (1) by the air conditioner to know when to increase/ decrease the temperature. (2) use to predict the how is the day (hot/ cold/ moderate) based on the average temperature of the day.

Layer 2. Network/ Communication Layer Network layer consists of various communication mediums to enable the sensors, devices, users, and others to communicate with each other. The communication medium can be a low power blue tooth for sensors, Zigbee for short distance, internet, WLAN, LAN and other technologies to enable communication and transfer of data/ information over the globe.

Layer 3. Application/ User Interface Layer Application layer provides interaction of the user with the system. The interface of the user to the system can be provided using mobile applications, web applications, through some device or using other mechanisms. Here, the user will vary depending on the domain such as in the case of the healthcare domain the users can be doctors, nurses, patients and in smart home users can be a house owner, guests, people living, and servants.

5.1.2 Difference between IoT Security and Network Security

At first glance, one can think of IoT is same as network system and security techniques applicable to existing network systems can be applied to IoT systems. But, IoT is a fusion of heterogeneous networks, which not only involves the same security problems as present in the existing network. However, more particular ones like privacy protection problem, heterogeneous network authentication and access control, information storage, and management, need to be handled. Our research shows that IoT security is different from Internet security (Islam, Kwak, Kabir, Hossain, & Kwak, 2015), it is far more complicated. Table 5.1 elaborates how IoT is more complex than network security, here for explaining the difference healthcare domain is considered.

Table 5.1 Comparison of IoT Security and Network Security

Design Parameters	**IoT Security**	**Network Security**
Memory constraints	The on-device memory of IoT devices is very low. They mainly use embedded operating system (OS), r the system software. Therefore, the system does not have enough memory to execute complicated security protocols.	No such memory constraint.
Speed of Computation and Resource constraints	Low-speed processors are available for IoT devices. The central processing unit (CPU) in such devices is not very fast. Therefore, finding a security solution that works on it, is a difficult task.	High-speed CPUs are available
Energy Limitations or Power consumption	An IoT network includes small health devices with limited battery power and has low CPU speed. They use the power-saving mode to conserve energy when sensors are idle. Therefore, the energy constraint makes finding security solution challenging task.	No battery problem. They are equipped with power backups
Scalability	There is a gradual increase in number of devices. Therefore, need to select scalable security algorithm becomes a challenging task.	They are connected by reliable wired links and have established wireless links also which are scalable.
Communications Channel	IoT devices are mainly connected to the network through wireless links such as Zigbee, Z-Wave, Bluetooth, Bluetooth Low Energy, WiFi, GSM, WiMax, and 3G/4G. Therefore, it is difficult to have a security protocol that works for wireless links and provides security comparable to wired links.	Less number of mobile devices
Security Updates	Need to keep security protocols up-to-date to mitigate potential vulnerabilities, Automatic updating of security protocol is difficult.	They have the established system for security.

5.2 Security Issues in IoT

IoT encompasses many technologies – networks, cloud system, operating systems, databases, resource scheduling, virtualization, transaction management, load balancing, concurrency control, and memory management. Security concerns pertaining to these technologies may apply to an IoT system. The network that connects the systems in the IoT has to be secure, and mapping of virtual machines to the physical machines needs to be carried out securely. Data security via encryption and appropriate policy enforcement for data sharing must be in place with secure resource allocation and memory management policies.

Some important issues present in IoT systems, extracted from different sources (Granjal, Monteiro, & Silva, 2015) (Roman, Najera, & Lopez, 2011) (Stankovic, 2014) (Sood, Yu, & Xiang, 2015) are as follows:

Identification / Authentication / Authorization: Authentication in IoT is very difficult as it involves authentication of a heterogeneous network. Identification and authentication of Things (sensors) must be done before they join the network. IoT requires a unique identification code or a global unique identifier (UID) for each entity in the network. Once the identification and authentication is done, authorization of user should be done i.e. set of rules can be provided which are permitted to him.

Confidentiality & Privacy: Need to ensure that the personal and sensitive information is not accessible to unauthorized users. In addition, confidential& private messages should not be revealed to eavesdroppers.

Resilience: In IoT, if some interconnected nodes are compromised, then the system should still protect the network/ device/information from attacks.

Fault Tolerance: The system should be able to function with relevant security services in case of a fault such as a device compromise or failure.

Self-Healing: If a sensor in an IoT network fails then, the other devices must be able to provide a minimum level of security.

Heterogeneity / Standardization / Interoperability: The devices used in IoT are mostly standalone, they are made for a specific purpose. Thousands of devices having different architectures and following different protocols, constitute an IoT network. Also, there is a lack of standardization between these devices. Interoperability between the devices is also a serious concern. This requires a proper security design process which needs to be followed to mitigate security problems arising from Heterogeneity, lack of Standardization and Interoperability.

Data Freshness: For any IoT network to work in an efficient manner nodes must have access to recent (fresh) messages or data. For example, to analyze the functionality of patient heart, the doctor needs the most recent ECG readings in a remote patient monitoring system.

Liability: In the case of any misuse, loss, theft or unusual event some liability or accountability should be provided.

Big Data: When the devices in IoT system communicate with themselves or with external entities a large amount of data is generated. Secure handling of this data needs to be ensured.

Constraints: IoT devices are constrained they have limited resources. Providing security with limited resources is a challenging task.

Trust: Trust should be present which determines a user's willingness to use the system. If a user is ensured that the system is not compromised, he is more willing to use the system.

Anonymity: Anonymity should be maintained. In some cases, the user does not want to disclose their identity. For example, in a Remote Patient Monitoring system, many medical patients do not want to disclose their identity or reports to anyone.

5.3 Existing Proposals in IoT for Security

Our dependence on IoT has created immense apprehension for study, analysis, and implementation of information security in IoT-based systems. As mentioned by Charles Renert (Websence Security Lab, 2015), vice president Websence Security Labs ***"The Internet of Things means consumer products from TVs to refrigerators are now digitally connected. While the enterprise need not fear the implications of an interconnected home appliance, every new employee's internet-connected device, application and upgrade is a potential threat vector."*** Iot has a layered structure and security threats are affecting the working of IoT at each layer. Researchers have identified various threats like man in middle attack, eavesdropping, malware attacks. It is gaining the attention of investigators to work on the different aspect of security to make IoT system free from various security lapses. A range of techniques is identified to prevent the security attacks. However, selecting a technique/ algorithm that best suits the given condition is a complex task.

Researchers are working on the security aspect of IoT and proposed techniques for identification of threats at different layers of IoT architecture (Li, Tryfonas, & Li, 2016) (Roman, Najera, & Lopez, 2011) (Jing, Vasilakos, Wan, Lu, & Qiu, 2014). In (Li, Tryfonas, & Li, 2016), researchers have identified potential threats at different layers of IoT architecture but left the solution aspect unexplored. In another research (Jing, Vasilakos, Wan, Lu, & Qiu, 2014), researchers have identified threats at each layer and focused on some cross-layer threats. Researchers have given some available solutions to handle threats such as Physical based schemes, password based schemes, permissions, frameworks have been proposed for privacy protection (Jing, Vasilakos, Wan, Lu, & Qui, 2014) (Valera, Zamora, & Skarmeta, 2010); RFID is the basic unit of IoT which can be easily forged, hence tag authentication is required, RFID tracking and inventorying are the two main privacy concerns (Jules, 2006). They have left the overall security architecture with the complete solution to an open issue for further research.

Conclusions are drawn from Existing Proposals

Classical ways of providing security are not sufficient in IoT environment. Traditional communication techniques TCP, WAN, IPSec, are used at transportation layer. The security issues in these techniques also pose a threat to IoT security. For example, (a) Denial of service attacks can lead to unavailability of the system, (b) Man-in-the-middle attack causes an information breach, (c) Network paralysis attacks can halt the ongoing traffic. Therefore, secure routing is needed. Data need to be secured when processing using different techniques.

Need physical mechanism to handle theft and misplacement of different sensor nodes (mobile phones, embedded chips, RFID tags) deployed. If theft occurs, then steps should be taken to immediately block the stolen device to avoid any unauthorized access to the sensitive data. As physical security is a great concern in IoT-based network. The interoperability among various networks (WSN, LAN, RFID, sensor) might pose some risk to the security, privacy, and trust. Therefore, there is a need to explore security solutions during interoperability.

All papers revolve around the identification of threats, and they have devised the solution approaches in an ad hoc way. There is no precise method available that identifies security requirements and specify them. Therefore, it calls for a process that identifies the security requirements efficiently by providing guidelines to handle them appropriately.

In short, security of IoT is very important as its roots are growing exponentially. For its positive growth, the shortcomings have to be removed to build and gain the confidence (trust) of the users. Every solution has its strengths and limitations. New techniques should be discovered which can give maximum throughput and have minimum limitations. The security solutions should be dynamic and adaptive. Security solutions for implementing security issues of IoT system are required which should be (a) Light weight (Consume less power/ energy, require less computation),

(d) Need less storage space, and (e) Need to work with things such as RFID tags, embedded systems, sensors etc. which has less computational. In the following section, we modify the Generic Security Engineering framework that will address new security issues in IoT and consider foregoing domain constraints.

5.4 Need for new Framework

Since the IOT systems are heterogeneous networks, its security issues are difficult to handle compared to any other system. Hence, framework proposed in previous chapters require modification. Some of the reasons for improvement are listed below:

- In web- based systems and cloud based system our process starts with the stakeholder's identification. But, in IOT based system we are focusing on assets because IOT system focuses on communication between physical devices which are known as *'things'*. Things are the asset of the system.
- Instead of identifying vulnerabilities for functional requirements, here vulnerabilities are identified for assets based on its role for the user.
- Some new vulnerabilities and threats are considered here such as Monitoring Absence, Untrained Users, Unsecured API Firmware, Obsolete system, etc. So threats and vulnerabilities of our previous frameworks have to be updated.
- Also, new security services are applicable for IOT based systems such as Data Freshness and Trust to tackle the adoption need and new threats and vulnerabilities.
- Some new hybrid algorithms are also available for IOT based systems. So, algorithm and threat repository should be changed.
- Also domain constraints are to be modified based on the layer at which security is to be considered.

From the above points, we conclude that our previous framework requires modification to make it adaptable for IOT- based system. Modified framework to handle the security issues of IOT system is presented and discussed in next section.

5.5 Security Engineering Framework for IoT-based Systems

The novel security engineering framework for IoT-based systems is shown in Figure 5.2. It works in two phases, in first phase security issues present in the system are identified and represented in the form of security requirements. In the next phase, different techniques to implement the identified security requirements are suggested or recommended based on domain constraints.

5.5.1 Identification and Specification. In IoT- based systems requirements engineer focuses on asset identification in contrast to identification of stakeholders as done in previous frameworks. Therefore, in the first phase generic assets of IoT-based systems are identified. IoT architecture has three layers and each layer is filled with assets (hardware and software's) which needs protection (see Table 5.2). Hence, focus should be on assets identification and then based on role of assets potential points of attacks (vulnerabilities) are identified. Once the vulnerabilities are known, potential threats to system assets are identified. Thereafter, identified threats are evaluated to deduce their severity by estimating the risk values. Finally, security requirements are elicited, prioritized and specified. Different activities of identification and specification phase are elaborated below:

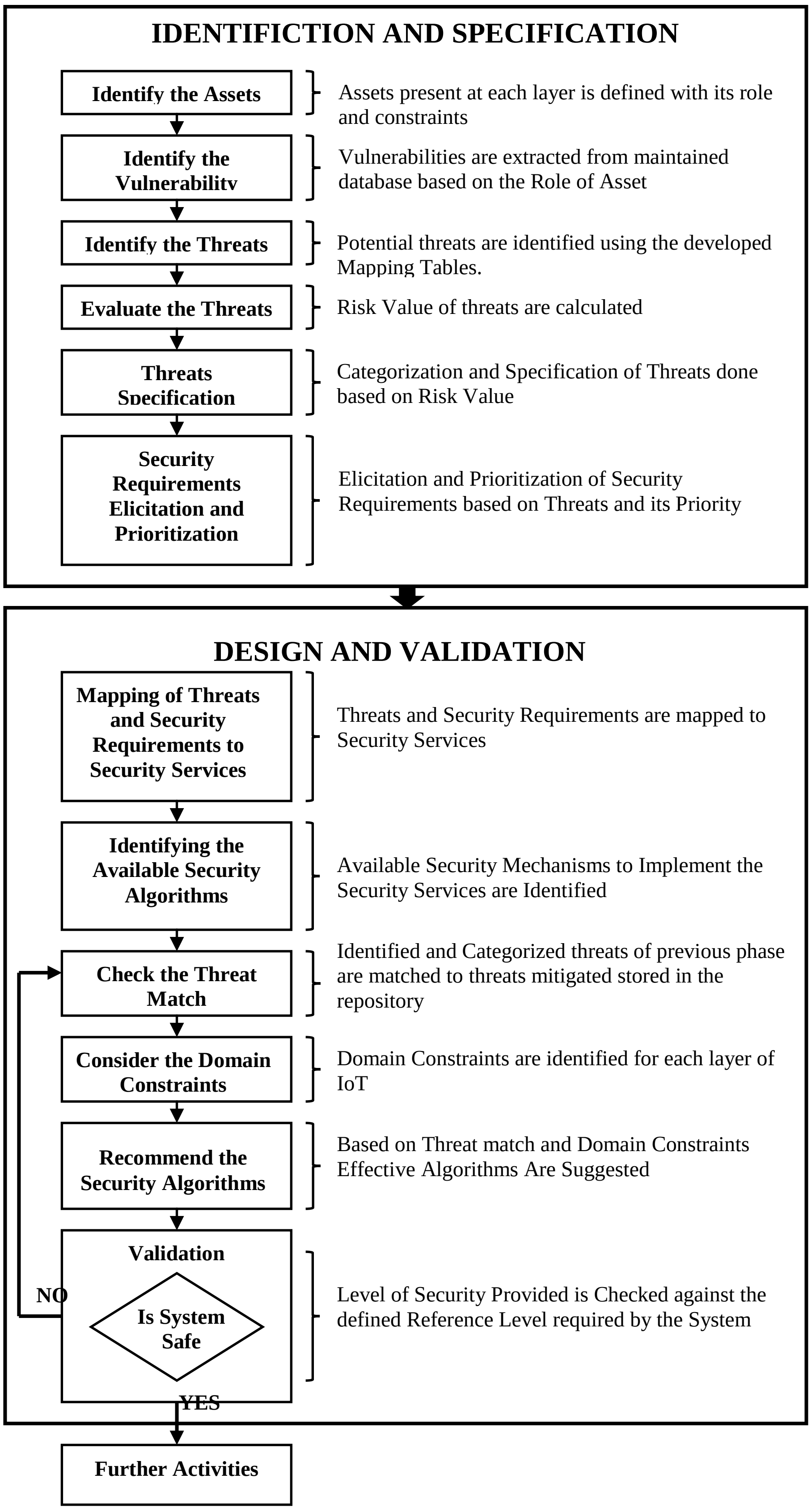

IDENTIFICTION AND SPECIFICATION
Identify the Assets
Assets present at each layer is defined with its role and constraints
Identify the Vulnerability
Vulnerabilities are extracted from maintained database based on the Role of Asset
Identify the Threats
Potential threats are identified using the developed Mapping Tables.
Evaluate the Threats
Risk Value of threats are calculated
Threats Specification
Categorization and Specification of Threats done based on Risk Value
Security Requirements Elicitation and Prioritization
Elicitation and Prioritization of Security Requirements based on Threats and its Priority
DESIGN AND VALIDATION
Mapping of Threats and Security Requirements to Security Services
Threats and Security Requirements are mapped to Security Services
Identifying the Available Security Algorithms
Available Security Mechanisms to Implement the Security Services are Identified
Check the Threat Match
Identified and Categorized threats of previous phase are matched to threats mitigated stored in the repository
Consider the Domain Constraints
Domain Constraints are identified for each layer of IoT
Recommend the Security Algorithms
Based on Threat match and Domain Constraints Effective Algorithms Are Suggested
Validation
Is System Safe
NO
YES
Level of Security Provided is Checked against the defined Reference Level required by the System
Further Activities

Figure 5.2 Proposed Framework for IoT-Based Systems

(i) Identify the Assets. Thesis aim is to secure assets of the system so that hassle-free services can be provided to users. An asset can be anything that has value to the organization it may be tangible (infrastructure) or intangible (customer information, trust). Functionalities work on the assets and are always the target of attackers. Therefore, generic assets associated with different layers of IoT architecture are identified with their roles. Assets are identified by analyzing various domains where IoT is implemented such as smart home, healthcare, vehicle tracking, and transportation. Further, assets are classified based on its types (if available). Also, possible constraints/limitation applicable to assets are identified. Identification of constraints on an asset is an important activity as it plays a vital role when it comes to suggesting efficient security solutions for implementing security requirements. A repository of assets present at different layers is built, and if some new asset is identified, it is added to the list. List of assets at various layers of IoT with suitable constraints and role are depicted in Table 5.2.

Table 5.2 Assets at various Layers of IoT

S.No	Layer	Assets	Further Sub-Categorization	Constraints	Role
1.	Sensing Layer	Sensors/ Actuators/ Controllers	▪ Environment Sensors (Light, Temperature, Humidity/ moisture) ▪ Body Sensors (ECG, Blood Pressure, etc.) ▪ Motion Sensors ▪ Microphone Sensors ▪ Gas/ Smoke Sensors ▪ Electrical Current/ ON-OFF Sensors ▪ Door (magnet) Sensors ▪ Physiological sensors	▪ Low power ▪ Low in Battery power ▪ Low memory ▪ Low computational speed ▪ Low communication bandwidth	Used for acquiring data through sensing

2.		Tags and Markers	▪ RFID ▪ NFC ▪ Security tokens ▪ Chip cards ▪ SIM	▪ Low power ▪ Low in Battery power ▪ Low memory ▪ Low computational speed ▪ Low communication bandwidth	For identification of devices
3.		Information Storage	▪ In-house server ▪ Cloud storage ▪ Removable resources	▪ Bulky/ Huge data ▪ No Fixed Structure (Structured, Unstructured) ▪ Confidential data ▪ Generated from different sources ▪ Different format ▪ Availability	For storing huge amount of data being generated and created
4.		Appliances	▪ Home (Refrigerator, Washing machine, and others) ▪ Hospital (Different Machines) ▪ Displays ▪ Speakers	▪ Availability ▪ Environment Constraints ▪ Battery/ Power/ Charging	Appliances that are dependent or attached to IoT network
5.	Communication Layer	Networking	▪ Internet connection (wired, wireless) ▪ Networking components (Routers, Bridge, Repeaters, Gateway, Firewall, Switch, and others)	▪ Limited bandwidth depending on the devices	For efficient communication between devices, data centers, and other involved equipment.
6.	User Interface layer	Human-machine interface device	▪ Specialized terminal ▪ Interface to gateway ▪ Remote control handset ▪ Smartphone ▪ Smart TV ▪ Tablet computer ▪ Desktop computer/PC	▪ Battery/ Power/charging ▪ Availability ▪ Environment Constraints	Mode of interaction

			▪ SOS/Emergency button ▪ Set-top-box user interface ▪ Calendar/Reminder device		
7.	Between interface of two Layers	Software	▪ Operating system(s) ▪ Device drivers ▪ Applications ▪ Firmware	▪ Auto update ▪ Security Patch Update ▪ Compatibility ▪ Battery/ Power/ Charging	For Data Processing
8.	At each layer	Information	▪ Access and payment credential for external accounts ▪ Smart (home, hospital, city) setup/ structure/ inventory information ▪ Status information ▪ Users preferences ▪ Value/IPRs ▪ Security • Passwords • User identification ▪ Privacy • User biometrics • Behavioral patterns and trends ▪ Resources • Music • A/V media • Pictures • Documents	▪ Distributed ▪ Bulky/ Huge ▪ Confidential ▪ Generated from different sources ▪ Different formats	Crucial and important data for processing
9.	Miscellaneous	Physical Resources	▪ Building ▪ Hardware (Air conditioners, Meters, Lighting and others)	▪ Physical constraints	Provides Infrastructure
10.	People/ Users	People/ User	▪ End users ▪ Providers ▪ Customers	▪ From different technical background ▪ May/may not have security knowledge	They will access and manage the system.

(ii) Identification of Vulnerabilities. Vulnerabilities present in the system give rise to threats, they are the weak points of the system which are exploited by the attackers to gain access to system resources (assets). Therefore, the presence of such points should be identified to protect assets from attackers. Vulnerabilities are collected and stored in the repository by doing extensive literature survey (Barnard-Wills, Marinos, & Portesi, 2014) (Mitrokotsa, Beye, & Peris-Lopez, 2010) (Jing, V. Vasilakos, Wan, Lu, & Qiu, 2014). Vulnerabilities corresponding to assets are extracted from the repository by considering their roles. Vulnerability for identified assets and are depicted in Table 5.3 which acts as a repository. For convenience and easy distinction, Vulnerabilities are prefixed with "V." Vulnerabilities are extracted from a repository based on the role (function) of assets for involved stakeholder, to do this a scenario diagram as explained in Section 3.2.1 of Chapter 3 is created which depicts when the assets are accessed and used. If a new vulnerability is reported, its details are accounted in the repository for further use.

(iii)Identify the Threats. To provide security to the system assets, one need to know the potential attacks to the system assets. Therefore, potential threats at different vulnerable points need to be identified, to do this a mapping table is proposed as shown in Table 5.4. Mapping table shows the probable threats at different vulnerable points. An "X" in the mapping table means that the threat can occur at given vulnerable point. Therefore, threats are extracted from the mapping table using the information (private exchange, assets involved and their role). For example, threat, T.Identity Fraud is possible due to vulnerabilities in column 1, 2, 9, 14, 21 corresponding to V.Weak Access Control, V.Inadequate Logging, V.Untrained User, V.Old Data, and V.Insufficient Security Configuration. For convenience and easy distinction, Threats are prefixed with “T.”.

Table 5.3 Identified Vulnerabilities for Assets

S.No	**Assets**	**Vulnerabilities**	**S.No**	**Assets**	**Vulnerabilities**
1	Sensors	V.Weak Access Control V.Unencrypted Data V.Physical Security V.Misconfiguration V.Insecure Interfaces V.Insufficient Security Configurability V.Remote Access V.System Misuse V.Monitoring Absence V.Inadequate Logging V.Lack of Standards	**6**	Tags and Markers	V.Weak Access Control V.Unencrypted Data V.Physical Security V.Misconfiguration V.Unsecured API Firmware V.Insufficient Security Configurability V.Remote Access V.Inadequate Logging V.Lack of Standards
2	Software	V.Inadequate Logging V.Misconfigurations V.Unsecured API Firmware V.Obsolete System V.Lack of Standards V.Intrusion Detection	**7**	Networking	V.Weak Access Control V.Unencrypted Data V.Breached Firewall V.Insecure Network services V.Insufficient Security Configurability
3	Human-machine interface device	V.Untrained Users V.Misconfigurations V.Unsecured Interface V.Obsolete System V.System Misuse	**8**	Information Storage	V.Weak Access Control V.Unencrypted Data V.Misconfigurations V.Insecure Interfaces V.Insufficient Security Configurability V.System Misuse V.Intrusion Detection
4	Information	V.Old Data V.Inadequate Logging V.Weak Access Control V.Lack of Standards V.Legal Audit	**9**	Appliances	V.Weak Access Control V.Monitoring Absence V.Physical Security V.Misconfigurations V.Obsolete System V.Lack of Standards V.Intrusion Detection
5	Physical Resources	V.Monitoring Absence V.Physical Security V.Misconfigurations V.Obsolete System V.Insufficient Security Configurability	**10**	People/ Users	V.Untrained Users V.System Misuse

Table 5.4 Vulnerability-Threat Mapping Table for IoT systems

Vulnerabilities→ Threats ↓		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
		V.Weak Access Control	V.Inadequate Logging	V.Breached Firewall	V.Unvalidated Input	V.Unsecured API Firmware	V.Obsolete System	V.Misconfig uration	V.Unencrypted Data	V.Untrained User	V.Monitoring Absence	V.Unsecured Network
1	T.Identity Fraud	X	X							X		
2	T.Infected e-mail			X		X						X
3	T.Denial of Service										X	
4	T.Information Leakage								X			X
5	T.Generation and use of Rouge Certificates	X	X									
6	T.Manipulation of Hardware and Software											
7	T.Manipulation of information	X	X						X	X	X	
8	T.Misuse of Audit Tools	X								X		
9	T.Falsification of Records	X			X	X					X	
10	T.Unauthorized use of Administration of devices and systems	X	X									
11	T.Unauthorized access to information system	X	X						X			
12	T.Unauthorized use of software	X	X			X						
13	T.Unauthorized installation of software	X		X		X		X				
14	T.Compromising Confidential Information									X		
15	T.Credential Theft	X									X	
16	T.Abuse of personal Data				X							
17	T.Malware			X	X	X					X	X
18	T.Communication Infiltration								X			X
19	T.Eavesdropping								X			X
20	T.Replay Message	X		X								X
21	T.Man in the Middle								X			X
22	T.Repudiation		X									

Table 5.4 Continued

	Vulnerabilities→ Threats ↓	1	2	3	4	5	6	7	8	9	10	11
		V.Weak Access Control	V.Inadequate Logging	V.Breached Firewall	V.Unvalidated Input	V.Unsecured API Firmware	V.Obsolete System	V.Misconfiguration	V.Unencrypted Data	V.Untrained User	V.Monitoring Absence	V.Unsecured Network
23	T.Hardware Failure						X					
24	T.Lack of Resources (water, electricity supply)							X				
25	T.Internet Outage							X				
26	T.Loss of Support Services							X				
27	T.Violation of Law or Regulations											
28	T.Physical Attacks										X	
29	T.Unintentional Damages									X		
30	T.Natural Disaster											
31	T.Environmental Disaster											
32	T.Faliure and Malfunctions							X			X	
33	T.Privacy Violated									X		
34	T.Insider	X	X		X	X		X	X		X	
35	T.Phishing	X								X	X	
36	T.Human Error				X					X		
37	T.Spoofing	X								X	X	
38	T.Node Capture											
39	T.Fake Node							X				X
40	T.Obsolete Data											

Table 5.4 Continued

	Vulnerabilities→ Threats ↓	12	13	14	15	16	17	18	19	20	21	22	
		V.Intrusion Detection	V.Physical Security	V.Old Data	V.System Misuse	V.Legal Audit Issues	V.Lack of Standards	V.Resource Isolation	V.Poor Key Management	V.Remote Access	V.Insufficient Security Configurability	V.Insecure Interfaces	Threat Rating
1	T.Identity Fraud			X							X		5
2	T.Infected e-mail	X											4
3	T.Denial of Service	X											2

Table 5.4 Continued

Vulnerabilities→		12	13	14	15	16	17	18	19	20	21	22	
Threats ↓		V.Intrusion Detection	V.Physical Security	V.Old Data	V.System Misuse	V.Legal Audit Issues	V.Lack of Standards	V.Resource Isolation	V.Poor Key Management	V.Remote Access	V.Insufficient Security Configurability	V.Insecure Interfaces	Threat Rating
4	T.Information Leakage								X		X		4
5	T.Generation and use of Rouge Certificates			X	X					X			5
6	T.Manipulation of Hardware and Software				X			X				X	3
7	T.Manipulation of information								X		X		7
8	T.Misuse of Audit Tools					X							3
9	T.Falsification of Records			X							X		6
10	T.Unauthorized use of Administration of devices and systems												2
11	T.Unauthorized access to information system			X			X						4
12	T.Unauthorized use of software						X						4
13	T.Unauthorized installation of software				X		X						6
14	T.Compromising Confidential Information										X	X	3
15	T.Credential Theft				X						X	X	5
16	T.Abuse of personal Data											X	2
17	T.Malware												5
18	T.Communication Infiltration								X		X		4
19	T.Eavesdropping	X							X		X		5
20	T.Replay Message	X			X				X		X	X	8
21	T.Man in the Middle	X							X		X		5
22	T.Repudiation	X							X		X	X	5
23	T.Hardware Failure							X					2
24	T.Lack of Resources (water, electricity supply)							X					2
25	T.Internet Outage							X					2
26	T.Loss of Support Services				X								2

Table 5.4 Continued

Vulnerabilities→ Threats ↓		12	13	14	15	16	17	18	19	20	21	22	
		V.Intrusion Detection	V.Physical Security	V.Old Data	V.System Misuse	V.Legal Audit Issues	V.Lack of Standards	V.Resource Isolation	V.Poor Key Management	V.Remote Access	V.Insufficient Security Configurability	V.Insecure Interfaces	Threat Rating
27	T.Violation of Law or Regulations					X	X					X	3
28	T.Physical Attacks		X										2
29	T.Unintentional Damages						X				X	X	4
30	T.Natural Disaster		X										1
31	T.Environmental Disaster		X										1
32	T.Faliure and Malfunctions							X			X		4
33	T.Privacy Violated				X						X	X	4
34	T.Insider		X		X			X					10
35	T.Phishing				X			X				X	6
36	T.Human Error				X	X	X						5
36	T.Spoofing				X			X			X	X	7
38	T.Node Capture		X					X					2
39	T.Fake Node		X					X					4
40	T.Obsolete Data			X									1

(iv) Evaluate the Threat. Evaluation of threats is a necessary activity as it helps us to know the severity (impact) of threat in the system. Here we are prioritizing the threats based on the associated risk values. Following sub-activities need to be followed for prioritization of threats as depicted in Figure 5.3 and explained further:

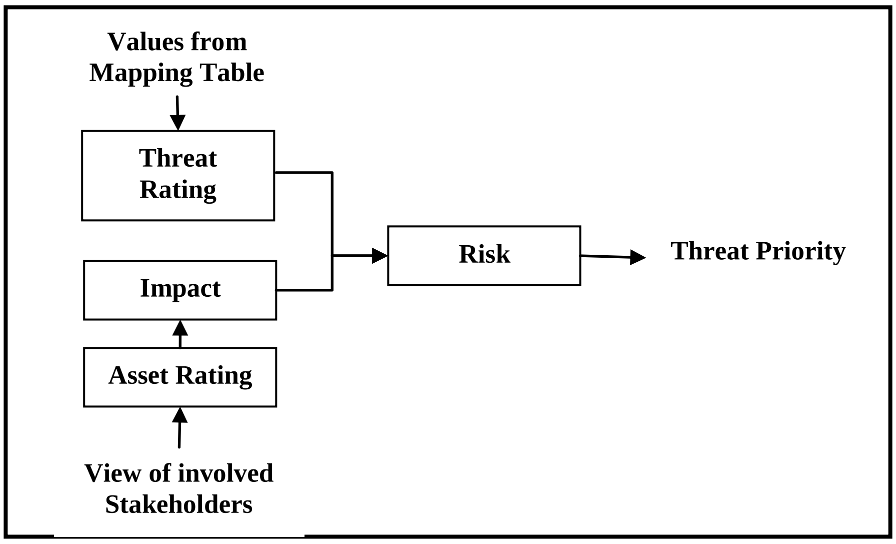


Figure 5.3 Process for Prioritization of Threats

(a) Identify the Threat Rating. Threat rating shows the occurrence frequency of a threat in the system. Threat rating is assigned by analyzing the incidence of 'X' in a row of vulnerability/ threat mapping table presented in Table 5.4. The presence of 'X' denotes, a threat can occur at given vulnerable point, represented by equation (4.1) in the Section 4.3.2 of Chapter 4. For instance, threat T.Indentity Fraud is occurring at weak points V.Weak Access Control, V.Inadequate Logging, V.Untrained User, V.Old Data, V.Insufficient Security Configuration so its threat rating would be '5' as 'X' occurs at five weak points.

(b) Identify the Impact. The impact of a threat on the system is estimated using the equation (4.1) presented in Section 4.3.2 of Chapter 4. The process for impact calculation is same as mentioned in previous chapters. To calculate the impact value, a repository of assets affected by potential threats are maintained which is shown in Table 5.5.

Table 5.5 Threats and Affected Assets

Threats	**Affected Assets**	**Threats**	**Affected Assets**
T.Identity Fraud	Sensors Software Human-machine interface device Information Physical Resources Tags and Markers Networking Information Storage Appliances	T.Infected e-mail	Software Tags and Markers Networking Information Storage Appliances
T.Manipulation of Hardware and Software	Sensors Software Human-machine interface device Tags and Markers Information Storage	T.Information Leakage	Sensors Physical Resources Tags and Markers Networking Information Storage
T.Generation and use of Rouge Certificates	Sensors Software Human-machine interface device Information Tags and Markers Networking Information Storage Appliances	T.Manipulation of information	Sensors Software Information Physical Resources Tags and Markers Networking Information Storage Appliances
T.Denial of Service	Sensors Software Physical Resources Tags and Markers Information Storage Appliances	T.Misuse of Audit Tools	Sensors Human-machine interface device Information Tags and Markers Networking Information Storage Appliances
T.Falsification of Records	Sensors Software Information Physical Resources Tags and Markers Networking Information Storage Appliances	T.Unauthorized access to information system	Sensors Software Information Tags and Markers Networking Information Storage Appliances

T.Unauthorized use of Administration of devices and systems	Sensors Software Information Tags and Markers Networking Information Storage Appliances	T.Unauthorized use of software	Sensors Software Information Tags and Markers Networking Information Storage Appliances
T.Unauthorized installation of software	Sensors Software Human-machine interface device Information Physical Resources Tags and Markers Networking Information Storage Appliances	T.Credential Theft	Sensors Software Human-machine interface device Information Physical Resources Tags and Markers Networking Information Storage Appliances
T.Compromising Confidential Information	Sensors Software Human-machine interface device Physical Resources Tags and Markers Networking Information Storage	T.Malware	Sensors Software Physical Resources Tags and Markers Networking Information Storage Appliances
T.Abuse of personal Data	Software Human-machine interface device Tags and Markers Information Storage	T.Communication Infiltration	Sensors Physical Resources Tags and Markers Networking Information Storage
T.Eavesdropping	Sensors Software Physical Resources Tags and Markers Networking Information Storage Appliances	T.Man in the Middle	Sensors Software Physical Resources Tags and Markers Networking Information Storage Appliances
T.Replay Message	Sensors Software Human-machine interface device Information Physical Resources Tags and Markers Networking Information Storage Appliances	T.Repudiation	Sensors Software Human-machine interface device Information Physical Resources Tags and Markers Networking Information Storage Appliances
T.Hardware Failure	Software Human-machine interface device Physical Resources Networking Appliances	T.Internet Outage	Sensors Software Human-machine interface device Physical Resources Tags and Markers Information Storage Appliances

T.Lack of Resources (water, electricity supply, etc)	Sensors Software Human-machine interface device Physical Resources Tags and Markers Information Storage Appliances	T.Loss of Support Services	Sensors Software Human-machine interface device Physical Resources Tags and Markers Information Storage Appliances
T.Violation of Law or Regulations	Sensors Software Human-machine interface device Information Tags and Markers Appliances	T.Natural Disaster	Sensors Physical Resources Tags and Markers Appliances
T.Physical Attacks	Sensors Physical Resources Tags and Markers Appliances	T.Environmental Disaster	Sensors Physical Resources Tags and Markers Appliances
T.Unintentional Damages	Sensors Software Human-machine interface device Information Physical Resources Tags and Markers Networking Information Storage Appliances	T.Faliure and Malfunctions	Sensors Software Human-machine interface device Physical Resources Tags and Markers Networking Information Storage Appliances
T.Privacy Violated	Sensors Software Human-machine interface device Physical Resources Tags and Markers Networking Information Storage	T.Insider	Sensors Software Human-machine interface device Information Physical Resources Tags and Markers Networking Information Storage Appliances
T.Phishing	Sensors Software Human-machine interface device Information Physical Resources Tags and Markers Networking Information Storage Appliances	T.Human Error	Sensors Software Human-machine interface device Information Tags and Markers Appliances
T.Spoofing	Sensors Software Human-machine interface device Information Physical Resources Tags and Markers Networking Information Storage	T.Fake Node	Sensors Software Human-machine interface device Physical Resources Tags and Markers Appliances

	Appliances		
T.Node Capture	Sensors Physical Resources Tags and Markers Appliances	T.Obsolete Data	Information

(c) Calculate the Risk. Risk shows the damage caused to the system assets by the occurrence of threats. The risk is the function of threat rating and impact as given by OWASP (OWASP, 2004) represented by equation (3.1) in section 3.2.2 of Chapter 3. For instance, Risk value for threat T.Identity Fraud is 70 (5*14); where '5' is threat rating of threat T.Identity Fraud and '14' is its impact (calculated in the previous step). So, the risk value of all potential threats to the system is calculated.

(d) Threat Specification. Threat specification means a clear and precise representation of threats. After the calculation of risk values, categorization and prioritization of threats are done based on the identified risk values. Then the prioritized and categorized threats are stored to take further necessary action during design and validation phase. A criterion for categorization of threats based on risk values is already discussed in Table 4.3. Based on the category, essential and important threats are handled first, and then tolerable threats may be considered based on the time and budget available for the system. Threats in no influence category can be ignored for further handling.

e) Security Requirements Identification and Prioritization. Security Requirements are elicited to represent the threats. After the elicitation, security requirements are prioritized based on threats priority. Finally, the prioritized security requirements are specified for further activities.

5.5.2 Design and Validation. After the security requirements are prioritized and specified, we proceed to the next phase of the security engineering process. During the design and validation phase, appropriate security algorithms for the implementation of threats are identified. Selection of security algorithm depends on (i) Number of threats mitigated and (ii) appropriateness of algorithm under given domain constraints. Moreover, after selection of all required security algorithms validation is done, to check if all potential threats are mitigated. Different activities of this phase are elaborated below:

(i) **Mapping of Threats and Security Requirements to Security Services.** Threats and security requirements are mapped to various security services (Forouzan, 2007) such as Confidentiality, Integrity, Availability, Non-Repudiation and Access Control. This would eventually help in the later stages of design and validation process, by specifying the suitable security mechanism for implementation. However, here besides the basic security services two more security services Trust and Data Freshness are considered for IoT systems. Trust security service would take care of issues pertaining to building the confidence of customers/ users in IoT-enabled systems. Data Freshness will take care of access to the latest and fresh data for use. Mapping of threats to security services is shown in Table 5.6.

(ii) **Identifying the available Security Algorithms.** Various security algorithms available to implement the security mechanisms/ services are identified as shown in Table 5.7. Design and security team members will analyze the algorithms based on domain constraints, and most appropriate technique is chosen for implementation.

Table 5.6 Mapping of Threats to Security Services and Mechanism

Security Services	**Security Requirements**	**Threats**	**Security Mechanism Available**
Data Confidentiality	Privacy Immunity Authentication Identification	T.Unauthorized use of software T.Unauthorized Installation of Software T.Compromise of Confidential Information T.Communication Infiltration T.Evesdropping T.Man in the Middle T.Privacy Violated T.Malicious Insider T.Identity Fraud	Encryption, Routing Control
Data Integrity	Integrity	T.Infected e-mail T.Information Leakage T.Manipulation of Hardware and Software T.Manipulation of Information T.Falsification of Records T.Credential Theft T.Abuse of Personal Data	Encryption, Digital Signature, Data Integrity
Availability	System Maintenance Survivability	T.Replay Messages T.Denial of Service T.Malware T.Hardware Failure T.Lack of Resources T.Internet Outage T.Unintentioanl Damages T.Human Error T.Failure and Malfunction	Encryption, Digital Signature, Authentication Exchanges
Non-Repudiation	Non-Repudiation	T.Generation and use of Rouge Certificates T.Repudiation	Digital Signature, Data Integrity, Notarization
Access Control	Intrusion Detection Authorization	T.Unauthorized use of Administration of Devices and /systems T.Unauthorized Access to Information system T.Misuse of Audit Tools T.Node Capture T.Fake Node	Access Control Mechanism
Data Freshness	**---**	**T.Obsolete Data** **T.Modified Data** **It is part of availability, but here it is considered separately as it is always required to access to latest data when predicting something about the system behavior.**	**Associate some Counter Mechanism**
Trust	---	T.Phishing T.Spoofing **T.Violation of Law and Regulation** **T.Lost of Support Services**	**By Implementing all other Security Services, Arranging training sessions for users to make them understand the system and its benefits.**
Physical Security	Physical Protection	T.Physical Attacks T.Natural Disaster T.Environmenatal Disaster T.Theft T.Node Failure	Using physical protection mechanisms

Table 5.7 Available Security Mechanisms

Security Mechanism	**Possible Techniques**	**Algorithm Characteristic**
Encryption	**Asymmetric** • **AES** • **DES** • **Triple DES** **Symmetric** • **RSA** • **Rabin's Scheme** • **ECC** • **HECC**	**Asymmetric** • Every node has its own set of keys (no complicated key management protocol required) • Good scalability • Takes more power because of computational complexity **Symmetric** • Simple calculation so takes less power • Require complex key management protocol • Confidentiality of key Authentication takes more power
Routing Control	Ad-hoc on demand Multipath Distance Vector routing protocol for IoT (AOMDV-IoT) Secure Multihop Routing Protocol (SMRP) Energy-aware Ant Routing Algorithm (EARA) Routing protocol over low power and lossy networks(RPL) Multiparent routing in RPL PAIR (Pruned Adaptive IoT Routing) REL (Routing protocol based on Energy and Link Quality)	• Prevent routing attacks such as spoofing, sink hole and selective forwarding.
Digital Signature	**RSA + DSA** **ECDSA** **HECDSA**	• Authentication takes more power because extra space is required to transmit the digest.
Data Integrity	**Hashing Algorithms** • **MD5** • **SHA1**	• Power requirement
Authentication Exchanges	**2 Step Authentication** **Multi-Step Authentication**	• Complexity
Notarization	**Build a Notary Server**	• Complexity
Access Control Mechanism	**Discretionary Access Control (DAC)** **Mandatory Access Control (MAC)** **Role-Based Access Control (RBAC)**	• Scalability • Manageability • Effectiveness • Resource constrained devices
Physical Protection Mechanisms	**Hardware security primitive** **Anti-theft policies**	• Cost

(iii)Check the Threat Match. Attack analysis repository of various security algorithms is maintained. So specified threats are checked in the repository and algorithm mitigating maximum threats are selected. A sample repository for confidentiality is shown in Table 5.8. A 'Y' in Table 5.8 denotes the security technique is able to handle the corresponding threat. For example, Technique AES under Asymmetric category is mitigating the T.Unauthorized use of software, T.Unauthorized Installation of Software, T.Communication Infiltration, T.Man in-the-Middle, T.Privacy Violated Threats, T.Compromising Confidential Information, T.Credential Theft, and T.Information Leakage. Last row of Table 5.8 contains values showing the total impact means number of threats corresponding techniques is mitigating.

Table 5.8 Threats to Mechanism Mapping

	Asymmetric			**Symmetric**			**Hybrid**
Techniques→ **Threats ↓**	**AES**	**DES**	**Triple-DES**	**RSA**	**ECC**	**HECC**	**ECIES**
T.Unauthorized use of software	Y	Y	Y	Y	Y	Y	Y
T.Unauthorized Installation of Software	Y	Y	Y	Y	Y	Y	Y
T.Compromise of Confidential Information	N	N	N	N	N	N	Y
T.Communication Infiltrations	Y	N	N	N	Y	Y	Y
T.Evesdropping	N	N	N	N	N	N	Y
T.Man in the Middle	Y	N	N	N	Y	Y	Y
T.Privacy Violated	Y	Y	Y	Y	Y	Y	Y
T.Malicious Insider	N	N	N	N	N	N	Y
T.Identity Fraud	N	N	N	Y	Y	N	Y
T.Compromising Confidential Information	Y	N	N	N	Y	Y	Y
T.Credential Theft	Y	N	N	N	Y	Y	Y
T.Information Leakage	Y	N	N	N	Y	Y	Y
TOTAL IMPACT	8	3	3	4	9	8	12

(iv)Consider the Domain Constraints. As algorithms are selected based on the number of threat matches, so further analysis of algorithm on constraints imposed by domain is required to be done. As all algorithms cannot be applied in every scenario, selected algorithms should be evaluated based on the domain constraints such as light-weight (power/ energy requirements, the speed of computation), memory needed, etc. Constraints pertaining to different layers of IoT system is shown in Table 5.9.

Table 5.9 Constraints in Different Layers of IoT

Constraints	**Sensing Layer**	**Communication Layer**	**User Interface Layer**
Performance Parameters			
Memory	Low	Medium	High
Speed of Computation	Low	Medium	High
Energy / Power	Low	Medium-High	High
Run Time performance	Medium	High	High
Other Parameters			
Security Objectives	High	High	High
Mobility Compatibility	High	High	Medium-High
Scalability	High	High	High
Cost of chosen solution	Low	Low	Low
Portability	High	High	High

(v) Recommend the Security Algorithm. Based on the above two steps of threat match and domain constraints, efficient security algorithms are identified and recommended for implementation.

(vi) Validation. Now validation of selected algorithm is done to check if potential threats to the system are mitigated or not. For the purpose of validation, Security Index value is calculated which shows the security gap left in the system. Security index is the ratio of live threats to a total number of threats identified (potential threats) in the system. SI is calculated using equation (3.2) and (3.3) given in Section 3.4 of Chapter 3.

If the value of SI is low (tending towards 0) means the system is safe and if it is high (tending towards 100) system is unsafe and requires modification in chosen security algorithm. For performing any modification in the systems security decision, the developer needs to go back to starting of design and validation phase to choose another algorithm for implementation.

5.6 CASE STUDY: Patient Monitoring System

Remote Patient Monitoring System shown in Figure 5.4 is a part of the healthcare system has following components:

(i) **Wireless body Area Network.** It is the patient body having wearable sensors capable of storing small information or sending it to a remote location.

(ii) **E-Health Gateway.** It would forward the packets from Wireless Body Network to remote servers and data centers over the Internet.

(iii) **Internet.** A communication network that would carry information.

(iv) **Healthcare Data Centre.** Store all the information that sensors in body generate. Data generated would be voluminous and need proper handling.

(v) **Medical Service.** Medical Facility such as doctor consultation, insurance service and medicines provided to the patient.

In the remote patient monitoring system, users/ actors are Patient, Doctor, and Insurance Service Providers. Therefore, asset role will vary from one user to other; here we are considering the abstract of all roles for further explanation.

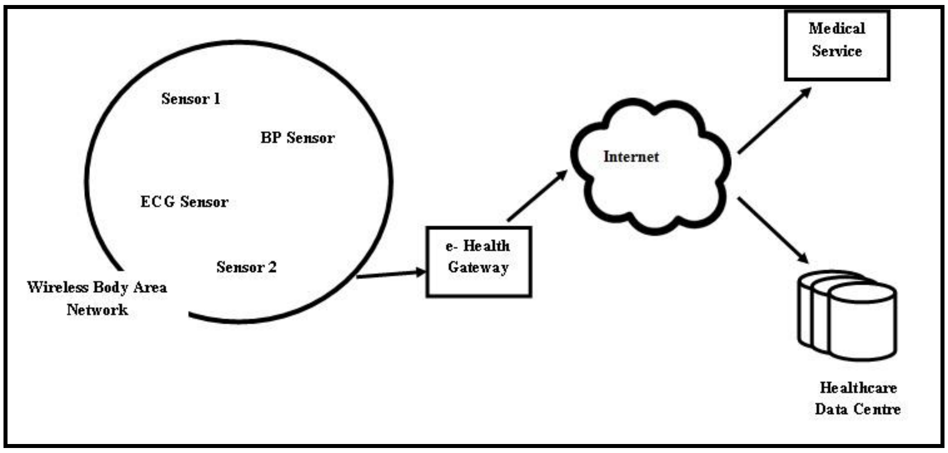


Figure 5.4 Remote Patient Monitoring System

5.6.1 Identification and Specification

(i) Identify the Assets. Assets involved in remote patient monitoring system are Body Sensors, Information storage, network and connections, Human machine interface devices (Smartphone, tablet, etc.), information (Patient, Doctor, etc.). Involved assets in the system with its role is depicted in Table 5.10.

Table 5.10 Involved Assets with their roles

S.No.	Asset	Role
1	Body Sensors	Fitted on Patient Body for sensing body parameters and sending it to either remote storage or remote diagnostic and treatment machines for processing.
2	Information Storage	Store collected data in a cloud-based storage
3	Network and connections	For communication between nodes
4	Human machine interface devices (Smartphone, tablet, etc.)	For user interaction
5	Information (Patient, Doctor, etc.).	Patient personal/ health information which is processed by the system.

(ii) Identification of vulnerabilities. Vulnerabilities extracted for the system from developed repository shown in Table 5.3 are listed in Table 5.11.

Table 5.11 Vulnerabilities and Threats for Assets

Assets	**Vulnerability**	**Threats**
Body Sensors	V.Weak Access Control V.Unencrypted Data V.Physical Security V.Misconfiguration V.Insecure Interfaces V.Insufficient Security Configurability V.Remote Access	T.Manipulation of Information T.Falsification of Records T.Infomation Leakage T.Physical Attacks T.Failure and Malfunctions T.Compromise of Confidential Information T.Abuse of Personal Data T.Repudiation T.Unintentional Damages
Information Storage	V.Weak Access Control V.Unencrypted Data V.Misconfigurations V.Insecure Interfaces V.Insufficient Security Configurability V.System Misuse V.Intrusion Detection	T.Manipulation of Information T.Misuse of Audit Tools T.Falsification of Records T.Unauthorized access to information system T.Unauthorized use of software T.Credential Theft T.Malicious Insider T.Phishing T.Spoofing T.Infomation Leakage T.Physical Attacks T.Natural Disaster T.Environmental Disaster T.Node Capture T.Fake Node T.Failure and Malfunctions T.Compromise of Confidential Information T.Abuse of Personal Data T.Violation of Law and Regulations T.Unintentional Damages T.Privacy Violated T.Denial of Service
Network and connections	V.Weak Access Control V.Unencrypted Data V.Breached Firewall V.Insecure Network services V.Insufficient Security Configurability	T.Identity Fraud T.Generation and use of Rouge Certificates T.Manipulation of Information T.Falsification of Records T.Unauthorized use of Administration of devices and systems T.Replay Message T.Phishing T.Spoofing T.Infomation Leakage T.Communication Infiltration T.Eavesdropping T.Man in the Middle T.Infected email

		T.Malware T.Fake Node T.Compromise of confidential Information T.Repudiation T.Unintentional Damages T.Failure and Malfunctions
Human machine interface devices (Smartphone, tablet, etc.)	V.Untrained Users V.Misconfigurations V.Unsecured Interface V.Obsolete System V.System Misuse	T.Identity Fraud T.Manipulation of Information T.Unintentional Damages T.Privacy Violated T.Human Error T.Unauthorized installation of software T.Loss of Support Services T.Failure and Malfunctions T.Compromise of Confidential Information T.Credential Theft T.Abuse of Personal Data T.Replay Messages T.Repudiation T.Violation of law and Regulations T.Hardware Failure T.Generation and use of Rouge Certificates
Information (Patient, Doctor, etc.).	V.Old Data V.Inadequate Logging V.Weak Access Control V.Lack of Standards V.Legal Audit	T.Identity Fraud T.Generation and use of Rouge Certificates T.Falsification of Records T.Manipulation of Information T.Credential Theft T.Repudiation T.Malicious Insider T.Phishing T.Spoofing T.Violation of law and Regulations T.Unintentional Damages T.Obsolete Data

(iii) Identify the Threats. Threats identified for Remote Patient Monitoring system using threat/ vulnerability mapping table established in Table 5.4 are shown in Table 5.11.

(iv) Evaluate the Threats. As threats are prioritized based on its impact on assets of the system. Sub-activities of prioritization are:

a) **Identify the Threats Rating.** Threat rating of all the identified threats is taken from vulnerability/ threat mapping table shown in Table 5.4. Threat ratings are depicted in Table 5.13.

b) **Identify the Impact.** To calculate the impact of threat on the system, we first need to calculate the asset values. Therefore, evaluation of identified system assets by involved stakeholders is shown in Table 5.12. Using these asset ratings, the impact is calculated as shown in Table 5.13.

c) **Calculate the Risk.** A risk value of identified threats is shown in Table 5.13.

Table 5.12 Evaluation of Assets

Asset	**View of involved Stakeholders**			**Asset Rating**
	Patient	**Doctor**	**Insurance Provider**	
Body Sensors	**8**	**9**	**7**	**8**
Interface Device (Smart Phone with application)	**8**	**8**	**8**	**8**
Patient Information	**9**	**8**	**7**	**8**
Network and Connections	**7**	**7**	**8**	**7**
Information Storage	**8**	**9**	**9**	**9**

Table 5.13 Calculation of Risk Value for Potential Threats

Threats	**Affected Assets**	**Threat Rating**	**Impact**	**Risk**
T.Identity Fraud	Sensors Interface device Information Networking Information Storage	5	40	200
T.Denial of Service	Sensors Information Storage	2	17	34
T.Generation and use of Rouge Certificates	Sensors Interface device Information Networking Information Storage	5	40	200

T.Falsification of Records	Sensors Information Networking Information Storage	6	32	192
T.Unauthorized use of Administration of devices and systems	Sensors Information Networking Information Storage	2	32	64
T.Unauthorized installation of software	Sensors Interface device Information Networking Information Storage	6	40	240
T.Compromising Confidential Information	Sensors Interface device Networking Information Storage	3	32	96
T.Credential Theft	Sensors Interface device Information Networking Information Storage	5	40	200
T.Eavesdropping	Sensors Networking Information Storage	5	24	120
T.Replay Message	Sensors Interface device Information Networking Information Storage	8	40	320
T.Hardware Failure	Interface device Networking	2	15	30
T.Violation of Law or Regulations	Sensors Interface device Information	3	24	72
T.Physical Attacks	Sensors	2	8	16
T.Unintentional Damages	Sensors Interface device Information Networking Information Storage	4	40	160
T.Privacy Violated	Sensors Interface device Networking Information Storage	4	32	128
T.Phishing	Sensors Interface device Information Networking Information Storage	6	40	240
T.Spoofing	Sensors Interface device	7	40	280

	Information Networking Information Storage			
T.Fake Node	Sensors Interface device	4	16	64
T.Manipulation of Information	Sensors Information Networking Information Storage	7	32	224
T.Information Leakage	Sensors Networking Information Storage	4	24	96
T.Failure and Malfunctions	Sensors Interface device Networking Information Storage	4	32	128
T.Abuse of Personal Data	Interface device Information Storage	2	17	34
T.Repudiation	Sensors Interface device Information Networking Information Storage	5	40	200
T.Misuse of Audit Tools	Sensors Interface device Information Networking Information Storage	3	40	120
T.Unauthorized Access to Information System	Sensors Information Networking Information Storage	4	32	128
T.Unauthorized use of software	Sensors Information Networking Information Storage	4	32	128
T.Malicious Insider	Sensors Interface device Information	10	24	240
T.Natural Disaster	Sensors	1	8	8
T.Environmenetal Disaster	Sensors	1	8	8
T.Node Capture	Sensors	2	8	16
T.Communication Infiltration	Sensors Networking Information Storage	4	24	96
T.Man in the Middle	Sensors Networking Information Storage	5	24	120
T.Infected Email	Networking Information Storage	4	16	64
T.Malware	Sensors	5	24	120

	Networking Information Storage			
T.Human Error	Sensors Interface device Information Networking Information Storage	5	40	200
T.Loss of Support Services	Sensors Interface device Information Storage	2	25	50
T.Obsolete Data	Information	1	8	8

d) **Threat Specification.** Threats are categorized based on the calculated risk values. Categorized threats are shown in Table 5.14.

Table 5.14 Categorized Threats

S.No	**Category**	**Threats**
1.	Catastrophic	T.Identity Frauds, T.Generation and use of Rogue Certificates, T.Falsification of Records, T.Unauthorized use of Administration of devices and systems, T.Unauthorized installation of software, T.Compromising Confidential Information, T.Credential Theft, T.Eavesdropping, T.Replay Message, T.Violation of Law or Regulations, T.Unintentional Damages, T.Privacy Violated, T.Phishing, T.Spoofing, T.Fake Node, T.Manipulation of Information, T.Information Leakage, T.Failure and Malfunctions, T.Repudiation, T.Misuse of Audit Tools, T.Unauthorized Access to Information System, T.Unauthorized use of software, T.Malicious Insider, T.Communication Infiltration, T.Man in the Middle, T.Infected Email, T.Malware, T.Human Error
2.	Important	T.Denial of Service, T.Hardware Failure, T.Abuse of Personal Data, T.Loss of Support Services
3.	Tolerable	T.Physical Attacks, T.Natural Disaster, T.Environmenetal Disaster, T.Node Capture,
4.	No influence (negligible)	Nil

e) **Security Requirements Identification and Prioritization.** Elicited security requirements are shown in Table 5.15 with threats they represent and their priority. A higher value represents higher priority.

Table 5.15 Elicited and Prioritized Security Requirements

Security Requirements	**Threats Handled**	**Risk**	**Security Requirements Priority**
Identification	T.Communication Infiltration T.Evesdropping T.Man in the Middle T.Malicious Insider T.Identity Fraud	96 120 120 240 200	776
Authentication	T.Identity Fraud T.Man in the Middle T.Unauthorized use of software T.Unauthorized Installation of Software	200 120 128 240	688
Privacy	T.Privacy Violated T.Compromise of Confidential Information T.Communication Infiltration	128 96 96	320
Immunity	T.Malicious Insider T.Communication Infiltration	240 96	336
Integrity	T.Infected e-mail T.Information Leakage T.Manipulation of Information T.Falsification of Records T.Credential Theft T.Abuse of Personal Data	64 96 224 192 200 34	810
System Maintenance	T.Hardware Failure T.Unintentioanl Damages T.Human Error T.Failure and Malfunction	30 160 200 128	518
Survivability	T.Replay Messages T.Denial of Service T.Malware	320 34 120	474
Non-Repudiation	T.Generation and use of Rouge Certificates T.Repudiation	200 200	400
Intrusion Detection	T.Unauthorized use of Administration of Devices and /systems T.Unauthorized Access to Information system T.Misuse of Audit Tools T.Node Capture	64 128 120 16	392

	T.Fake Node	64	
Authorization	T.Unauthorized use of Administration of Devices and /systems T.Unauthorized Access to Information system T.Fake Node	64 128 64	256
Data Freshness	T.Obsolete Data	8	8
Trust	T.Phishing T.Spoofing **T.Violation of Law and Regulation** **T.Lost of Support Services**	240 280 72 50	642
Physical Protection	T.Physical Attacks T.Natural Disaster T.Environmenatal Disaster	16 8 8	32

5.6.2 Design and Validation

(i) Mapping of threats to Security Requirements and Security Services. Mapping of threats to security requirements and services are already mentioned in Table 5.6.

(ii) Identifying the available Security Mechanisms. Available security mechanisms are already mentioned in Table 5.7.

(iii) Recommend Security Algorithm. Based on the specified match and constraints algorithms are suggested for implementation as shown in Table 5.16.

Table 5.16 Recommended Security Techniques

Security Mechanism	**Techniques Recommended**
Encryption	**Asymmetric** • **AES** **Symmetric** • **ECC** **Hybrid** • **ECIES**
Routing Control	**Energy-aware Ant Routing Algorithm (EARA)**
Digital Signature	**ECDSA**
Data Integrity	**MD5**
Authentication Exchanges	**2 Step Authentication**
Notarization	**Build a Notary Server**
Access Control Mechanism	**Role-Based Access Control (RBAC)**
Physical Protection Mechanisms	**Lock, Guarding**

(vi) Validation. For ensuring Confidentiality

Threats to confidentiality are:

- T.Identity Frauds
- T.Falsification of Records
- T.Unauthorized installation of software
- T.Compromising Confidential Information
- T.Credential Theft
- T.Eavesdropping
- T.Privacy Violated
- T.Manipulation of Information
- T.Information Leakage
- T.Unauthorized use of software
- T.Communication Infiltration
- T.Man in the Middle

Asymmetric Technique: AES is chosen

SI = 5/ 12 * 100 = 41.67 %

Symmetric Technique: ECC is chosen

SI = 4/ 12 * 100 = 33.33%

But both of the algorithms is not sufficient to provide protection. Hence hybrid techniques are required. A **hybrid technique ECIES** is chosen.

SI = 3/ 12 * 100 = 25%

A hybrid algorithm is much better than the existing algorithms. But, more algorithms are required for effective working.

Summary

Security Engineering in IoT is in infancy stage. Hence, no open source software or tool is available for the purpose of validation. But, we conjecture that our proposal identifies maximum threats that are present in current scenario.

- Elicitation, analysis, and prioritization of Security Threats for IoT systems is presented. Here threats to assets are identified instead of functional requirements. After specification of Security Threats, they are mapped to security requirements. Security algorithms are chosen based on various domain constraints such as Light weight (Consume less power, require less computation time), Need less storage space, etc. Finally, a metric is generated showing system security level. The process is explained for IoT-enabled Healthcare domain for remote patient monitoring.

To achieve the above-said goals following activities are being done:

- Various <u>**assets at different layers of IoT is identified with probable vulnerabilities and threats.**</u>
- Various <u>**threats that can affect assets of the system are identified and mapping table of dimension 39 X 30 (approx.)**</u>, is generated which helps in risk assessment.
- Security mechanism is suggested based on domain constraints.

- It can be noted that the result of Security Index is very high, which suggests that existing security algorithms are not suitable. It requires the development of new algorithms which are hybrid of existing elliptic curve based algorithms.

Novel contributions of the Chapter

a) The framework elicits, analyzes, prioritizes, and specify the threats to assets. Security Requirements are also specified to represent the threats.

b) <u>**Two new security services namely data freshness and trust are added**</u> to the cluster of existing security services.

c) During the requirements engineering phase, identification and specification of threats to assets are done. To elicit the security threats, we have generated a repository of:

   1) **<u>Assets at each layer of IoT architecture</u>**
   2) **<u>A vulnerability threat mapping table is constructed of dimension 39 X 22</u>**
   3) **<u>Threats affecting assets of the system whose dimension is approximately 39 X 30.</u>**

   This would help in handling various assets, functionality, threats, vulnerabilities.

d) Security algorithms are chosen considering various domain constraints pertaining to different layers of IoT during the design and validation phase.

e) Finally, a security metric is generated showing system security level. In case of remote health monitoring system, new hybrid algorithm is required for implementation of security requirements security metric value is very high.

CHAPTER 6

SECURITY ENGINEERING FRAMEWORK FOR BIG DATA DATABASES

In the current scenario, big data databases have gained a lot of attention due to its nature of providing highly scalable storage space to the organization for data storage and use. These databases are mainly known as NoSQL databases which are required to fulfill the three V's that are volume, velocity, and variety need of Big Data environment. The most important challenge of these databases is the security issues that are inherent in it because of schema-less design which makes it different from traditional databases. Security challenges make client data at risk; these issues provide intruders a chance to attack for stealing client's personal and confidential information. Therefore, security concerns should be addressed to protect the confidential data stored in databases. Here a security methodology for handling security issues of NoSQL databases, a big data store. Therefore, the chapter starts with the discussion on Big Data with its security issues. The discussion is followed by the framework for security engineering for big data system, and further, a case study of MongoDB is presented.

6.1 Security Issues in Big Data Databases

Providing security in big data databases is very difficult and challenging because of various constraints like they do not has defined schema, the amount of data is very huge, data consists of a variety of media, etc. Therefore, in this section, we present the overview of big data databases and then the security issues present in it is discussed.

6.1.1 Overview

Big data environment is characterized by 3 V's (Volume, Variety, Velocity) of data. It is changing the society by its tremendous use in various sectors such as healthcare, finance, and social networking. Data generated from these sources are in hundreds of terabyte (TB) to petabytes (PB) with a high rate of data generation. In addition to this, the data generated from these sources are mainly unstructured or semi-structured. Traditional RDBMS systems can not handle these issues. Therefore, to handle all these issues NoSQL databases have evolved.

As the database contains the valuable asset 'data' of the organization, access to which by an intruder is always a major security threat. Also, Integrity, Confidentiality, and Availability of data are need to be ensured. Ensuring CIA is a difficult task in Big Data environment, because of inherent vulnerabilities like distributed nature, data fragmentation, inadequate logging facility, etc. These vulnerabilities are cause for various attacks such as impersonation, repudiation, DoS, communication interception. Therefore, a security methodology is required which deals with such vulnerabilities and threats.

"Big Data is data whose scale, diversity and complexity requires new architecture, techniques, algorithms, and analytics to manage it and extract value and hidden knowledge from it" (Finkelstein & Fuks, 1989) (T & Kumar, 2013). Data produced by social networking or e-commerce organizations including Facebook, Google, LinkedIn, Twitter or Amazon are analyzing user status and search terms to trigger targeted advertising on user pages. Some examples of Big databases are MongoDB, Cassandra, CouchDB, and Redis. All these databases are NoSQL because they do not have defined structure as RDBMS.

Previously our needs were being fulfilled by RDBMS systems, but now they are not able to handle the growing needs. Hence, a shift is made to handle the need as discussed above. The RDBMS systems are following ACID properties as they can fulfill security needs.

During the initial years of NoSQL development, it is designed to cater the need of organizations neglecting the security aspect. But, with the growing demand for Big Data need to handle security in NoSQL databases are arises. Big data storage techniques have presented a breakthrough in achieving scalability, cost reduction, performance and flexibility in the management of the tremendous amount of data. NoSQL data stores are vulnerable to the same security risks as traditional RDBMS data stores, so the usual best practices for storing sensitive data should be applied when developing a NoSQL-based application. Besides the security issues related to RDBMS other constraints specific to big data environment are no defined schema, a huge amount of data, data consists of a variety of media, etc. are need to be considered.

The privacy of Big Data is a growing concern. The customer or user information is collected and used for value added services without user awareness. Integration of large data sets involving personal information may lead to the inference of new facts about the person which may be confidential. These facts may be secretive, and the person might want them to be hidden from other users or organizations. Some Big Data application like Healthcare has strict laws governing the privacy of data like the HIPAA. 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 the privacy of sensitive data.

Papers are found in the literature focusing on access control, integrity and other security aspects in traditional databases (Bertino & Sandhu, 2005) (Ambhore, Waghmare, & Meshram, 2007) (Pan, 2009). Our research shows that security aspect of NoSQL databases is left unexplored in detail. Security issues present in NoSQL databases are found in some papers (Vormetric, 2012) (Katal, Wazid, & Goudar, 2013). However, these papers only identify the security issues present in NoSQL databases; they do not focus on how to handle them. In the literature, no papers were found that proposes a process/ method/ methodology/ framework to handle security issues in NoSQL databases.

6.1.2 Security Issues

The amount, diversity, and rate of data being generated for processing and storage results in sheer masses of data that need to be secured. Data generated in Big Data environment are in the hands of organizations are highly valuable, and are subject to privacy laws and compliance regulations, hence need to be protected. The following are some security issues of Big Data environment.

i) **Distributed nature:** Nodes within the Big Data environment are distributed which makes it an easy target for attack affecting the data stored in nodes.

ii) **Integrity of data:** The protection of integrity is much harder in Big Data environments because of its heterogeneous nature, the absence of central control and its schema-less nature.

iii) **Communication between nodes:** Interactions between distributed nodes rely on RPC over TCP/IP, which makes RPC ports vulnerable. Security concerns emanate as nodes interact through message passing.

iv) **Fragmentation and sharing of data:** NoSQL databases horizontally fragments the data known as shards and share them across multiple servers. These shards are replicated across the nodes. Maintenance of replicated shards of data that includes passwords is computationally expensive, prone to error and increases the risk of theft. As the model is not centralized, securing data is difficult because of replication of data.

v) **Lack of central management security:** Clients accessing NoSQL databases are in contact with resource managers and nodes directly. In situations where malicious data get propagated from a single compromised location, the entire system is compromised. Protecting nodes, name servers, and clients become difficult, especially when there is no central management security point.

vi) **Encryption of data:** Some NoSQL databases keeps the data in unencrypted form, in such type of databases some applications are requested to encrypt private data explicitly before storing in a database. Example: Cassandra database.

vii) **Enforcing access control:** The NoSQL databases have schema-less structure makes Role-based access control difficult to enforce. Because different types of data are stored at one huge database.

viii) **Authenticating Clients:** Kerberos can be used to authenticate clients, Data Node, and Name Node in the Big Data environment. Malicious Clients or Nodes can gain unauthorized access to the Big Data environments upon stealing or duplicating the Kerberos ticket.

ix) **Auditing and logging:** Audits are performed, and logs are created to aid the discovery of malicious activities in the database system. However, without actually looking at the data timely and developing policies to detect malicious activities, logging is not useful. Also, the frequency at which Audits are carried out can impact on system effectiveness.

x) **Monitoring, input validation, and blocking:** Big Data collect data from different sources. Existing Big Data monitoring tools lacks the capability of identifying malicious queries, misuse activities and blocking. Monitoring undertaken by several tools in the Big Data environment, mostly perform their task at the API.

xi) **API security:** APIs may be subjected to several attacks such as Code injection, buffer overflows, command injection as they access the NoSQL databases. The APIs for big data clusters need to be protected from code and command injection, buffer overflow attacks, and other web service attacks.

xii) **Inference problem:** Big data management usually involves applying data mining and analytics. It brings many security concerns related to sharing of big data analytics as there is a risk of loss of privacy and confidentiality of data. If there is 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.

6.2 Need for New Framework

Big data comprises of schema less and scalable data storage spaces, therefore data is of utmost concern. Since big data concentrates on large volume of data with high rate of generation and variety. Initial framework should be enhanced so that is can cope up with the characteristic of big data. Reasons in support of this modification are mentioned below:

- Need to handle security issues such as Fragmentation and sharing, Encryption of data, Enforcing access control, etc.

- Limited stakeholders with limited functionality has to deal with large amount of data.
- In contrast to variety of assets in previous domains, here data is only asset in NoSQL databases.
- Domain constraints vary from previous considered frameworks. The domains constraint for big data are source of data and its complexity, type of information, structured or unstructured data, etc.

Form above points we conclude that our initial framework needs modification to make it adaptable for big data domain. Modified framework to handle the issues listed above is presented in the next section.

6.3 Proposed Security Engineering Framework for Big Data Databases

As NoSQL databases has limited set of functionalities that can be performed on the only asset 'Data' available. In big data databases domain constraints (source of data generation, type of data, complexity of data, etc.) are very much different from the previous ones (bandwidth, memory, energy, speed, etc.). Hence, for incorporating above features a modified framework for Security Engineering for Big Data Databases or NoSQL databases is depicted in Figure 6.1, it consists of two phases that are identification and design. Each phase is described here in detail.

6.3.1 Identification Phase. Security requirements to mitigate the threats present in the system are identified, analyzed, and prioritized. Various activities involved in the identification of Security Requirements depicted in Figure 6.1 are discussed below:

- **Security Requirements Elicitation:** Security requirements are important for implementing security in the system so its elicitation is necessary. Therefore, the process for elicitation of security requirements consists of following activities:

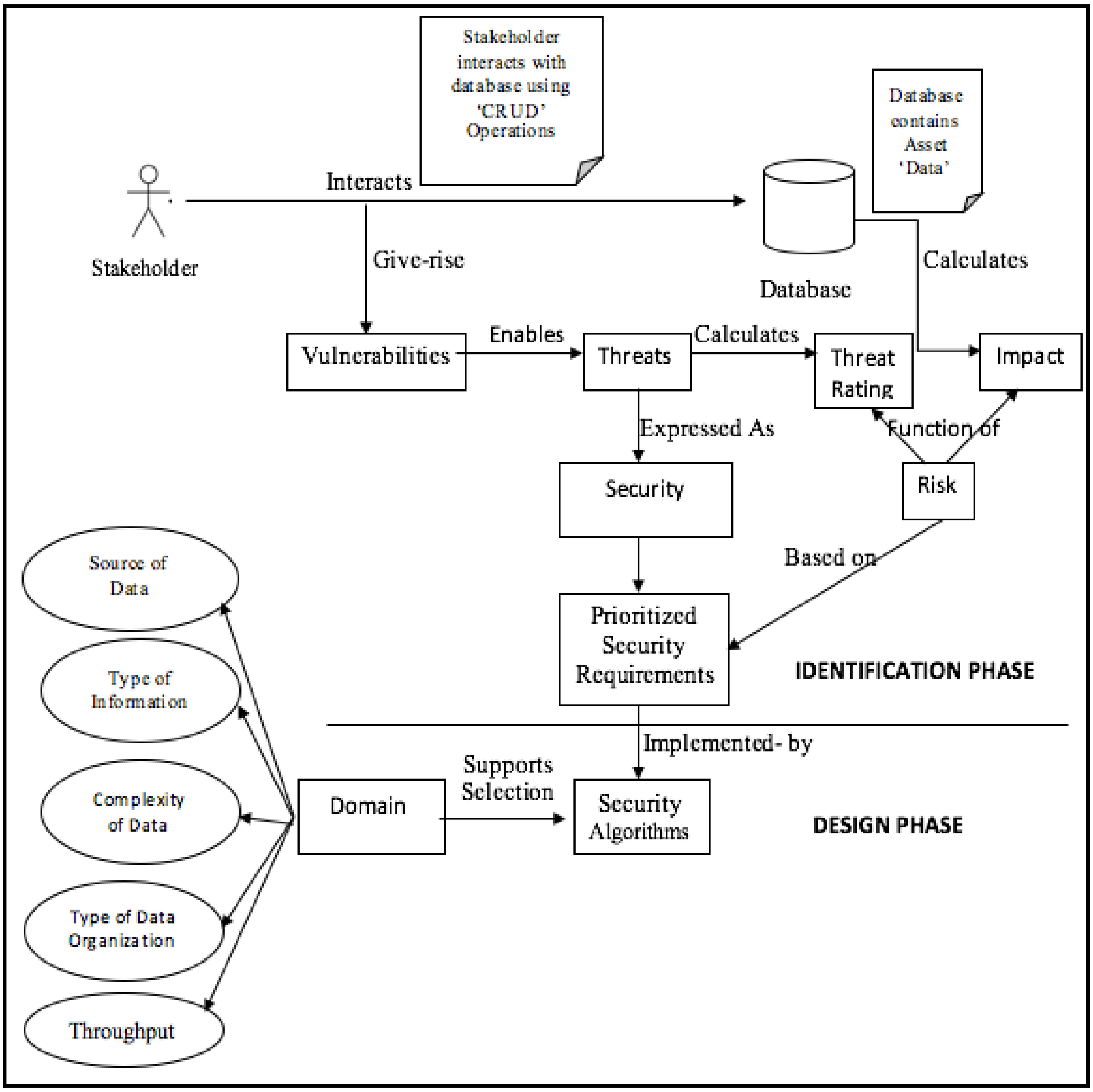


Figure 6.1 Security Engineering Framework for Big Data Databases

- **Stakeholder Identification:** Stakeholders are identified using view-point approach (Kotonya & Sommerville, 1996). Direct stakeholders for big data databases is user which is further classified as Human, Cooperative and Autonomous. Details of each are listed in Table 6.1.

Table 6.1 List of Direct Stakeholders

Stakeholder	**Description**
Human User	Human actors are active entities that interact directly with the information system.
Cooperative User	DMBS must cooperate with the primary actor, such as if the data is being generated or transferred from some other data source.
Autonomous User	Act independently of the information system but have connections to it. Data generated from some devices such as medical sensors, monitoring devices, and others.

- **Operation Identification:** Operations are performed by all the specified direct actors for interacting with databases. CRUD (Create, Read, Update and Delete) operations are used by stakeholders for manipulating data in databases. Access to the database is provided by these generic operations. Details of specific operation are defined in Table 6.2.

Table 6.2 Operations for Stakeholders

Stakeholder	**Generic Operations**	**Description**
For all Stakeholders	Create/ Insert/ Put/ Post	Adding new entries to the database
	Read/ Retrieve/ Select/ Get	Retrieve, search, or view existing entries without changing the data
	Update/ Modify/ Update/ Put/ Patch	Edit or modify existing entries (changes the data values by insertion, deletion or update)
	Delete/ Destroy	Remove or deactivate existing entries

- **Vulnerability identification:** Vulnerability is a flaw or weakness in the system environment (Stoneburner, Alice, & Feringa, 2002), that a malicious attacker could exploit to cause damage to the system. It is the vulnerability that enables a threat to be exercised within the system. In Big Data environment, vulnerabilities arise due to the complexity brought in by the type and distributed nature of data involved. Vulnerabilities are preceded with a prefix V. for a clear distinction. Vulnerabilities are identified by analyzing different system and available literature. Possible vulnerabilities extracted for CRUD operation of NoSQL environments are shown in Table 6.3.

Table 6.3 Vulnerabilities for CRUD Operations

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 ->Database	Read Request	1. V.Weak_Access_Control 2. V.Untrained_Users 3. V.Monitoring_Absence 4. V.Network_Partition 5. V.Inadequate_Logging
	Database->Database	Search	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	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	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	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	1. V.Network_Partition
DELETE	User ->Database	Delete Request	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	1. V.Monitoring_Absence 2. V.Physical_Security 3. V.Misconfigurations 4. V.Inadequate_Logging
	Database-> User	Return Confirmation	1. V.Network_Partition

- **Threats Identification:** Vulnerability may lead to the occurrence of threats, mapping of vulnerability to threats is required to be done for detailed risk analysis of the system. Prefix T. is used with threat name to make a distinguishable and threat-vulnerability database is maintained from where the mapping of various threats to vulnerabilities is done. Using Table 6.4 threats are mapped to identifed vulnerabilities.

- **Security Requirements Identification:** After the identification of vulnerabilities and threats, security requirements are elicited to mitigate the threats and protect the asset. Security requirements are identified according to mapping criteria specified in Table 6.5.

Table 6.4 Vulnerabilities-Threats Mapping Table for Big Data Databases

Vulnerability →	1	2	3	4	5	6	7	8	9	10	11	12	13	**Threat Rating**
Threats ↓	V.Weak Access Control	V.Inadequate Logging	V.Breached Firewall	V.Unvalidated Input	V.Unsecured API	V.Obsolete System	V.Misconfiguration	V.Unencrypted Data	V.Untrained User	V.Monitoring Absence	V.Unsecured Network	V.Network Partition	V.Physical Security	
T.Change Data	X	X	X					X	X		X			6
T.Data Theft	X		X					X						3
T.Impersonate	X													1
T.Social Engineer	X								X					2
T.Fraud	X	X												2
T.Privacy Violated	X							X						2
T.Repudiation Receive		X												1
T.Repudiate Send		X												1
T.Credential Theft	X									X				2
T.Phishing	X								X	X				3
T.Insider	X	X		X	X		X	X		X			X	8
T.Spoofing	X								X	X				3
T.Human Error				X					X					2
T.Disclose Data								X	X					2
T.DDoS										X		X		2
T.Misuse of System Resources									X					1
T.Injection Attack				X	X									2
T.Malware			X	X	X					X	X			5
T.Communication Interception								X			X			2
T.Communication Infiltration								X			X			2
T.Eavesdropping								X			X			2
T.Technical Failure						X	X					X		3
T.Power Failure							X							1
T.Network Infrastructure Failure							X							1
T.Hardware Failure						X						X	X	3
T.Unavailability												X		1
T.Vandalism													X	1
T.Operational Issues						X								1

Table 6.5 Security Requirements mapping to Threats for Big Data Database Systems

Security Requirements → Threats ↓	Identification	Authentication	Authorization	Immunity	Integrity	Intrusion Detection	Non-Repudiation	Privacy	Security Auditing	Survivability	Physical Protection	System Maintenance
T.Change Data	X	X	X									
T.Data Theft	X	X	X									
T.Impersonate	X						X					
T.Social Engineer	X	X										
T.Fraud		X										
T.Privacy Violated								X				
T.Repudiation Receive							X					
T.Repudiate Send							X					
T.Credential Theft		X										
T.Phishing		X	X									
T.Insider	X	X	X		X		X					
T.Spoofing	X	X	X				X					
T.Human Error					X							
T.Disclose Data			X				X					
T.DDoS		X			X					X		
T.Misuse of System Resources		X										
T.Injection Attack			X	X		X						
T.Malware				X		X						
T.Communication Interception					X							
T.Communication Infiltration					X							
T.Eavesdropping					X							
T.Technical Failure										X		
T.Power Failure										X		
T.Network Infrastructure Failure										X		
T.Hardware Failure										X		
T.Unavailability										X		
T.Vandalism											X	
T.Operational Issues										X		

- **Security Requirements Analysis and Prioritization**

  After elicitation of security requirements analysis and prioritization is done. During analysis elicited security requirements are analyzed for Completeness and Consistency, and if any conflict occurs it would be removed immediately. Also, the security requirements with similar characteristics are grouped together.

  As all requirements cannot be implemented so, prioritization is done. Elicited security requirements are prioritized, so that depending on available resources; high priority requirements are implemented first. Following activities are followed for prioritization:

  - **Likelihood Estimation:** Likelihood shows the rough measure of how likely a threat would occur in the system. Likelihood ratings are estimated for each threat based on the degree of satisfaction of vulnerability. Therefore, the value of threat rating is taken as the total number of vulnerabilities exploited by particular threat in vulnerability-threat mapping table shown in Table 6.4.

  - **Impact Estimation.** The impact is usually calculated using the number of assets affected. As here our asset is only data. Therefore, impact value would be taken same for all the threats. Here Impact is taken as unity.

  - **Security Requirements Prioritization.** Risk is calculated using equation (3.1) defined in Section 3.2.2 of chapter 3.

Here, Impact is taken as unity because only asset 'data' is considered whose value would be same. Therefore, the risk is equal to likelihood value.

After the calculation of risk value, security requirements priority is calculated by summing up the risk value of all the threats mitigated by security requirements under consideration. For instance, security requirement 'Identification' mitigates threats 'Change Data' and 'Data Theft' whose threat rating is '6' and '3' respectively. So, its priority value is 9 (6+ 3) which is the summation of all risk values.

6.3.2 Design Phase: Security mechanism to implement the security requirements are identified in this phase. The security mechanism is the popular algorithms such as cryptography algorithms, physical security mechanisms, etc. which are required to implement the security requirements. The security algorithms are identified, and detailed analysis is done on domain parameters. It consists of following activities:

- **Mapping of Security Requirements with security services.** Prioritized security requirements are mapped to security services provided by cryptography. Also, possible security mechanisms that exist for the implementation of Security services are listed as depicted in Table 6.6.

- **Identify and Analyze Domain Constraints.** Various techniques are available for the implementation of security requirements, so comprehensive evaluation of algorithms under given domain constraints is done for choosing best algorithm for security requirements implementation. The outcome of this step would show a detailed analysis of different constraints.

Table 6.6 Mapping of Security Services with Security Requirements

Security Services	**Security Requirements**	**Security Mechanisms**
Confidentiality	Privacy	Encryption, Digital Signature
Integrity	Integrity Intrusion Detection Immunity	Encryption, Digital Signature, Data Integrity
Authentication	Authentication	Authentication Exchanges, Two Factor Authentication
Non repudiation	Identification Non Repudiation	Digital Signature, Data Integrity, Notarization
Access Control	Authorization	Need-to-know Principle Enforcement, RnR Clarity
Others (Required for smooth running of system)	Auditing Survivability Physical Protection System Maintenance	Physical Security, Auditing, and related certifications

In the case of NoSQL databases, constraints are based on the environment of big data such as:

- From where the data to be stored is generated (social networking site, sensors, satellites, or other sources)
- Type of data generated (structured, semi-structured and unstructured).
- Type of Information (Personal, Financial, and others)
- Complexity of Data
- Performance needed from database (throughput)

The Table 6.7 shows the predictive value for each criterion based on analysis of the various system and Table 6.8 shows analysis of various data origination source.

Table 6.7 Metric showing the Constraints and Possible Value

Criteria	**Category**	**Predictive Value**	**Reason**
Source of Data Generation	Social Network Medical Sensors Financial Transactions Satellite Data Mobile Data Organization Data Website content		
Type of Information	Personal (data related to user personal information),	Medium to High	Data is confidential/ private to user
	Critical (financial data),	High	Contains transfer of money, smart card details and other sensitive information
	Normal (other than personal and critical)	Low to Medium	These are a normal website, sensor (used at some entrance, or in the device) data.
Complexity of Data	Data generated media type • Text only • Image only • Video only • Audio only • Mixture of two or above-mentioned categories	Simple Composite	Consists of single Media Type Consists of hybrid Media Type
Type of Data Organization	Structured	Low	As they follow ACID property of RDBMS
	Semi-Structured	Medium	Has some structure such as tag field based on some attribute
	Unstructured	High	No defined structure
Throughput	What efficiency is expected for operations (read, write, update, delete)	Low Medium High	

Table 6.8 Source of Origin of Data Analysis

Source of Data	Description	Type of Data	Possible Constraints
Social Networking Site	This data is generated from the social media platforms such as YouTube, Facebook, Twitter, LinkedIn, and Flickr	Semi-structured (Human Generated)	Contains Personal Information
Medical Sensor Data	There is a huge explosion in the number of sensors producing streams of data all around us.	Semi-Structured	Contains personal Information related to Medical Problems
Financial Transactions (payment)	Payment made to various e-commerce and social sites	Semi-Structured	Contains credit card and other account details
Satellite images	It includes weather data or the data that the government captures in its satellite surveillance imagery. Just think how about Google Earth haves instant access to locations.	Semi-structured (machine generated)	Contains Sensitive Data
Organizational Information	Think of all the text within documents, logs, survey results, and e-mails. Enterprise information represents a large percent of the text information in the world today.	Unstructured (human generated)	Contains company confidential Information
Mobile Data	It includes data such as text messages and location information.	Unstructured (human generated)	Contains personal Information
Website content	It comes from any site delivering unstructured content, like YouTube, Flickr, or Instagram.	Unstructured (human generated)	Contains general Information such as news, educational materials, movies, and others

Therefore, to protect personal, confidential and critical data, it is required to focus on every aspect of security.

- **Security Design Decision.** Based on above two steps proper security algorithms are suggested for implementation. A decision template and guideline is generated in this step. A decision template is generated as shown in Table 6.9, which contains all security related algorithms for implementation.

Table 6.9 Decision Template

	Mechanism Selected	**Reasoning**
Encryption	AES-256	For data at rest, the algorithm can encrypt all type of data at rest
	SSL/ TLS	For network
Digital Signature	RSA+DSA	For Confidentiality of Information
Data Integrity	Tokenization	It is appended to the end of message to ensure data integrity
Authentication Exchanges	Password protection 2 step Authentication Kerberos	Enforce strong policy for password management such as regular password change/ update. Where access to sensitive data is occurring 2-step authentication should be used. Here some secret code is sent to user’s mobile or on email.
Notarization	Delegation of task (for key distribution, identification, and authentication of user, employee)	For management of encryption key using some third party system as KDC (key distribution center) used in Kerberos.
Access Control Mechanisms	Define Roles (Role-based Access Control) 2 step Authentication	Assign strict roles to everyone Some authorization mechanism should be implemented such as • 2 step authentication • OTP system can be implemented
Physical Security	Guarding, Locking, Fencing, and others	Deploy necessary physical security in the premises
Auditing	Implement some technique for it	It looks at the log and identifies any security issues related to authentication failure, authorization failure, denial of service and others. After identification, it must be reported to the administrator for proper handling.
Certifications	PCI-DSS, SSAE 16, ISAE3402, ISO 270001:2013	
Others	R n R Clarity need to know principal enforcement Vulnerability assessment tools	Roles and responsibility must be defined to each stakeholder. Sessions must be organized for stakeholders such that they would know what they require from the system Some vulnerability scanning tool must be provided for continuous monitoring of system for identification of any vulnerable points in the database.

- **Generate Security Guidelines:** Here guideline for security is generated it would help all the stakeholders in understanding the security need of the system. As providing security in NoSQL databases is the responsibility of all involved stakeholders. Table 6.10 shows a brief guideline template.

Table 6.10 Guidelines for Stakeholders to handle Security Properly

Stakeholder	**Expected Security Action**
Common for All Stakeholders	• Do not share password • Access only data for which authorization is provided • Get educated about basic security standards • Log of all interaction should be maintained
Human User	• Keep password strong and change it after certain time interval • Get educated about security aspects and implement it
Cooperative User	• It must be authenticated first and then only allowed to interact with and user or database
Autonomous User	• All devices, sensors, nodes should be authenticated before interaction with the database.

6.4 Security Analysis of MongoDB

MongoDB is a highly flexible, scalable, schema-less, a 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" (Okman, Gal-Oz, Gonen, Gudes, & Abramov, 2011). MongoDB claims to put together the features of the RDBMS, document databases, key-value stores and object databases.

MongoDB is chosen here because (1) It is one of the most widely used NoSQL databases it is being used by a large spectrum of organizations such as SourceForge, Bitly, Foursquare, GitHub, Shutterfly, Evite, The New York Times, Etsy, and much more. (2) It has very clear and organized documentation available freely on the internet which explains everything (MongoDB, 2014) (MongoDB). Table 6.10 shows the security mechanism employed by MongoDB in the current version as well as the deficiencies in previous versions.

Loopholes in Mongo DB are identified by researching the various available versions. Our identification is shown in Table 6.11. As seen from the Table 6.11, initial versions of MongoDB lacked a lot of basic security features required by the system and caused various security breaches in the past. These security breaches caused information leaks, loss of reputation, and other major issues. Whereas, if our framework would be adopted and followed all security related issues are handled during the initial development. As we have considered maximum viewpoint and identified and analyzed all security related issues in detail. Also, all the available security mechanisms are analyzed and evaluated based on the environment constraints of application and chosen the best algorithms for implementation that fits the identified set of constraints. Also, it can be noted as whatever algorithms are identified using our approach is almost same as used in the current version of MongoDB. It proves that our framework is effective as it can identify all security algorithms required by a NoSQL database.

Table 6.11 Security Features in MongoDB

Security Features	**Security Features in Current Version**	**Security Loopholes in the Previous Version**
Encryption	• Encrypt Connections to the database (**SSL/ TLS)** • Support FIPS 140-2 • Encrypt data at rest (**AES 256 bit in CBC and GCM mode)** • Sign and rotate encryption keys	• No Encryption at rest • No support for SSL/ TLS.
Authentication	• Create Unique Security Credential for developers, admin, DBA • Nodes are also authenticated • Enforce Password Policies • **Kerberos** • **LDAP** • **X.509** • **PKI Certificates** • **SCRAM-SHA-1**	• No support for x.509-based authentication • No support to SCRAM-SHA-1 challenge-response user authentication mechanism.
Access Control (Authorization)	• Grant minimal access to entities • Group common access privileges into roles • Control which action an entity can perform • Control access to sensitive data • **Role-based Access Control** • **Field Level Redaction**	• No support for role-based access control system
Auditing	• Track changes to DB configuration • Track changes to data (not keeping track of all read/ write as they are huge in number)	
Environment and Process Control	• Installation of **Firewall or ACL Routers** • Network Configurations • Defining File System Permissions • Creation of Physical Access Control to IT environment • **Binding of IP Addresses** • **Running in VPN's is limited** • **Dedicated OS user account** • **File system permissions**	
Others	• DBA and Developer Training • Database Monitoring and Backup (**MONGO STAT and MONGOTOP)** • Database Maintenance	

The Figure 6.2 list the vulnerabilities over the year for MongoDB reported by CVE (CVE, 2013). As evident from the Figure 6.2, we can say that if our framework has been followed then all these vulnerabilities would have been handled during the initial versions. Table 6.12 shows the comparison of threats identified using our approach and threats listed by CVE.

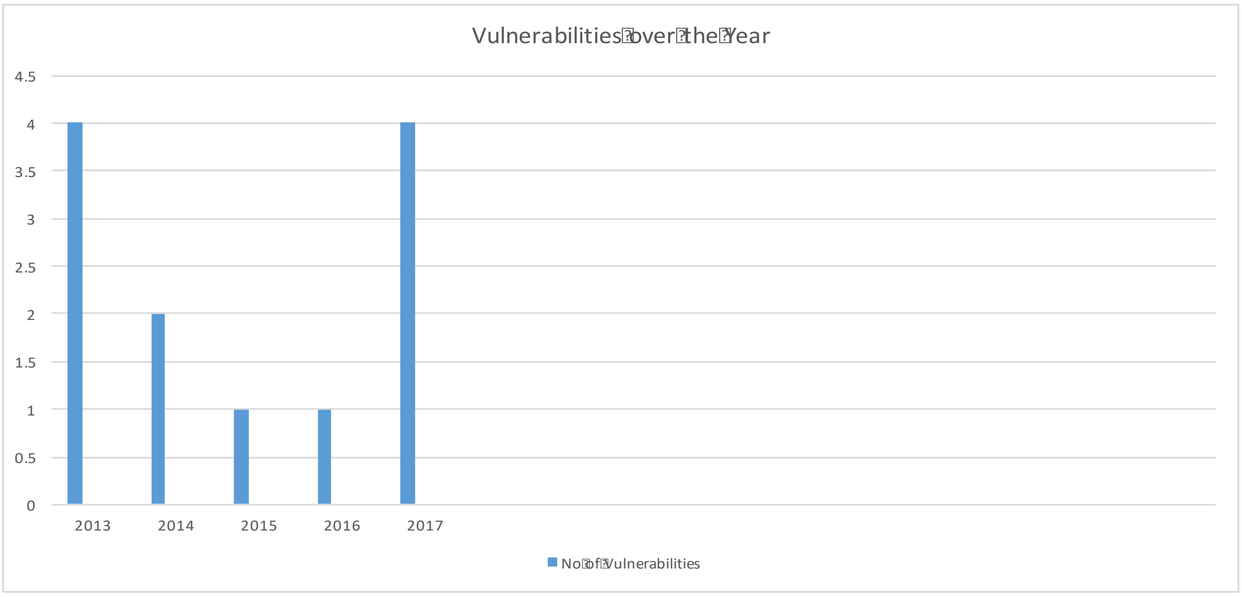


Figure 6.2 Vulnerabilities over the year for MongoDB

Our research shows that there are no recent research publications related to the handling of the security issues in a detailed manner. Previous research mainly focused on the security issues that exist in the NoSQL environment. No one has proposed a detailed framework that deals with security issues in a structured manner.

Table 6.12 Comparison of Threats

S.No	Threats Identified Using our Approach	Threats reported by CVE on MongoDB	Threats Identified using Security Engineering Framework
1	T.Change Data	**NO**	**YES**
2	T.Data Theft	**NO**	**YES**

3	T.Impersonate	**NO**	**YES**
4	T.Social Engineer	**NO**	**YES**
5	T.Fraud	**NO**	**YES**
6	T.Privacy Violated	**NO**	**YES**
7	T.Repudiation Receive	**NO**	**YES**
8	T.Repudiate Send	**NO**	**YES**
9	T.Credential Theft	**NO**	**YES**
10	T.Phishing	**NO**	**YES**
11	T.Insider	**NO**	**YES**
12	T.Spoofing	**NO**	**YES**
13	T.Human Error	**NO**	**YES**
14	T.Disclose Data	**NO**	**YES**
15	T.DDoS	**YES**	**YES**
16	T.Misuse of System Resources	**YES**	**YES**
17	T.Injection Attack	**YES**	**YES**
18	T.Malware	**YES**	**YES**
19	T.Technical Failure	**YES**	**YES**
20	T.Power Failure	**NO**	**YES**
21	T.Hardware Failure	**NO**	**YES**
22	Privacy Violated	**NO**	**NO**
23	Communication Interception	**NO**	**NO**
24	Communication Infiltration	**NO**	**NO**
25	Eavesdropping	**NO**	**NO**
26	Network Infrastructure Failure	**NO**	**NO**
27	Unavailability	**NO**	**NO**
28	Vandalism	**NO**	**NO**
29	Operational Issues	**NO**	**NO**

Summary

- It identifies, analyzes, and prioritize the security requirements for NoSQL databases along with the possible operations.

- During the requirements phase, to automate the process of identification of security requirements we have generated:
    - A vulnerability-threat mapping table of dimension (28 X 13) for identification of vulnerabilities and threats.
    - A threat-security requirements mapping table of dimension (28 X 12) is built to enable easy identification of security requirements.

Novel Contribution of the Chapter

- **Various domain constraints pertaining to big data are identified such as source of data generation, type of data, complexity of data, etc.** Their predictive values are also identified during the design phase. Based on these constraints best-suited security algorithms are chosen and suggested for implementation.

- Also, the proposed framework provides the security guidelines to stakeholders for understanding the security need.

- Finally, security analysis of a commonly used NoSQL database 'MongoDB' was presented. And it is found that threats reported by CVE for it are all covered by our approach. Also, threats identified by our approach are large in number, and proper security algorithms are also suggested to handle them.

CHAPTER 7

IMPLEMENTATION

Chapter 3 of the thesis presented a generic security engineering framework for information system development. Further, the framework is adapted for various emerging domains namely Cloud Systems, Internet of Things (IoT) and Big Data Databases as explained in Chapters 4, 5, and 6 respectively. This chapter discusses the implementation of the tool to assist the users in identifying the security related details for different domains. The chapter starts with the overview of the tool and further detailed discussion on tools phases is presented. Also, different screen shots of the tool are presented for clear understanding.

7.1 Overview

As proposed in chapter 3, our framework starts with the identification of actors, functionality, and assets. Then vulnerability and threats to functional requirements are identified. Threats are then evaluated based on risk measure. After that, security requirements are elicited to represent the threats. Further, these requirements are analyzed and prioritized. Then, in the next phase, security algorithm is chosen to implement the prioritized security requirements based on domain constraints.

Hence, to automate the whole process, a tool is developed named Security Engineering Tool (SET). The tool work in two phases which corresponds to two phases of our generic framework. The first phase of the tool is Security Requirements Elicitation Prioritization Specification (SREPS), and the second phase is Security

Design Engineering (SDE). In next forthcoming sections, the architecture of both the phases of our tool is presented.

7.2 Security Requirements Elicitation Prioritization Specification (SREPS)

The architecture of SREPS is shown in Figure 7.1, which consists of two parts:

- **Front end** part that provides, a user interface for selecting the details for elicitation, prioritization and specification of security requirements.
- **Back end** part of tool helps in extraction of information from maintained repositories based on details provided by the user.

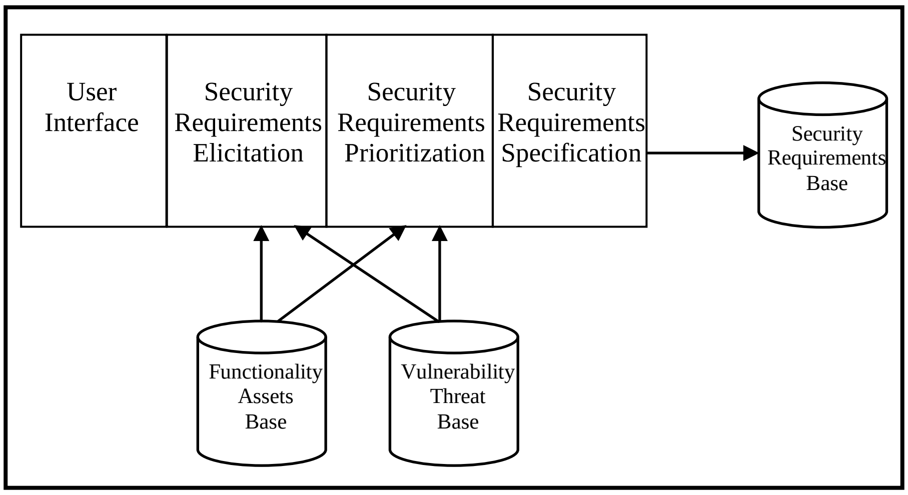

User Interface
Security Requirements Elicitation
Security Requirements Prioritization
Security Requirements Specification
Security Requirements Base
Functionality Assets Base
Vulnerability Threat Base

Figure 7.1 Architecture of SREPS

Functionality, Assets base shown in Figure 7.1, contains the functional requirements and assets of the system. This information is presented to the users for selection of functionality and needed assets by him, to start the process of elicitation of security requirements. To start with the process user first need to do the login, login window is shown in Figure 7.2. After verification of the login details, authenticated user can get access to the functions of the tool for elicitation, analysis, prioritization, and specification.

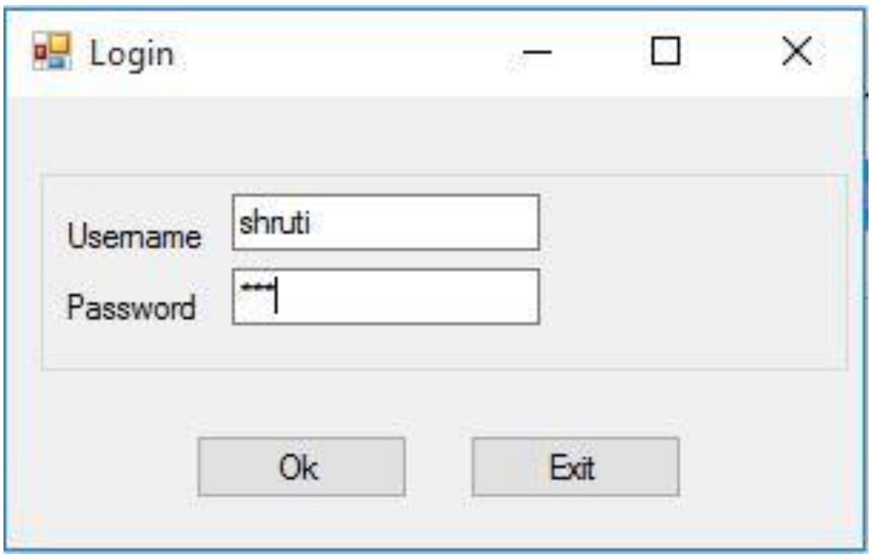
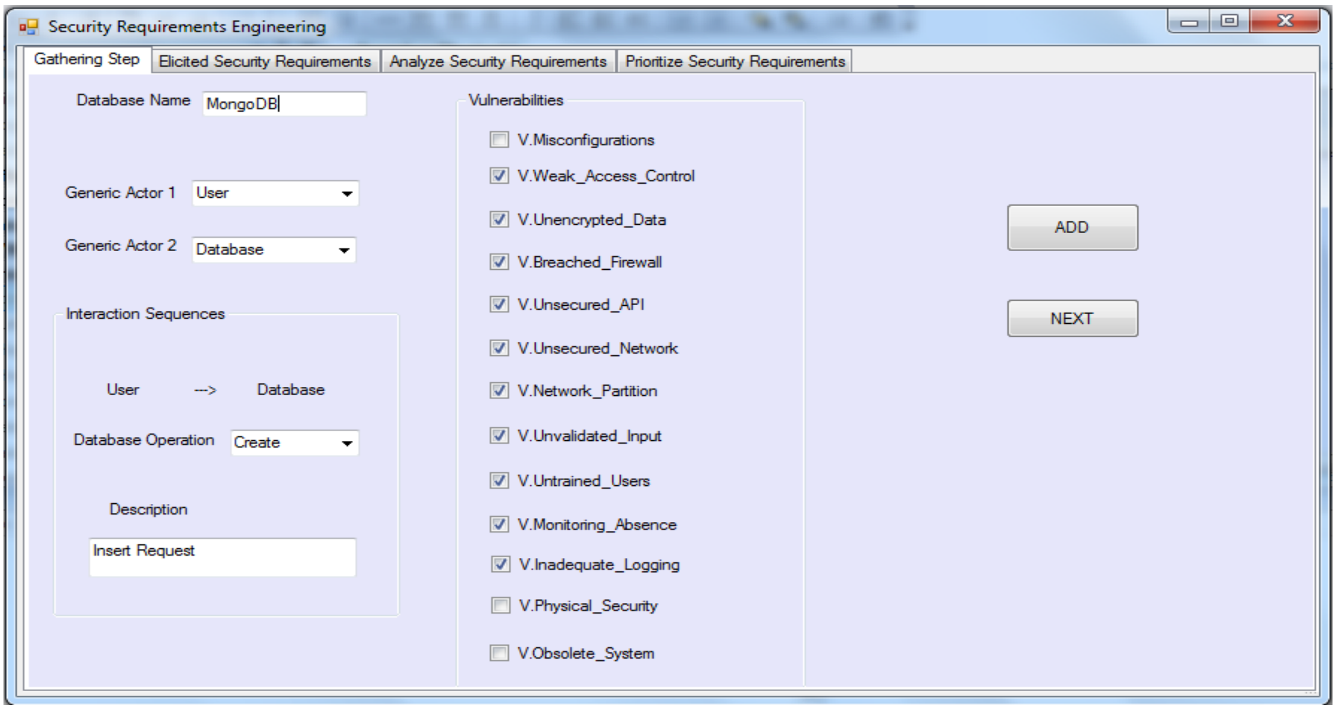


Figure 7.2 Login Page of Tool

Vulnerability, Threat base contains the potential vulnerabilities and threats of the system. After Login in the system user now select the actors, functionality/ operation and other related details. Based on the selected details vulnerabilities and threats are identified from maintained vulnerability, threats base. Further, to mitigate the threats security requirements are elicited.

Figure 7.3 shows the selection of different fields to start with the elicitation process. Based on the selection parameters Figure 7.4 shows the elicited security requirements for the operation ‘Create.’


Security Requirements Engineering

Gathering Step | Elicited Security Requirements | Analyze Security Requirements | Prioritize Security Requirements

Database Name: MongoDB

Generic Actor 1: User

Generic Actor 2: Database

Interaction Sequences

User --> Database

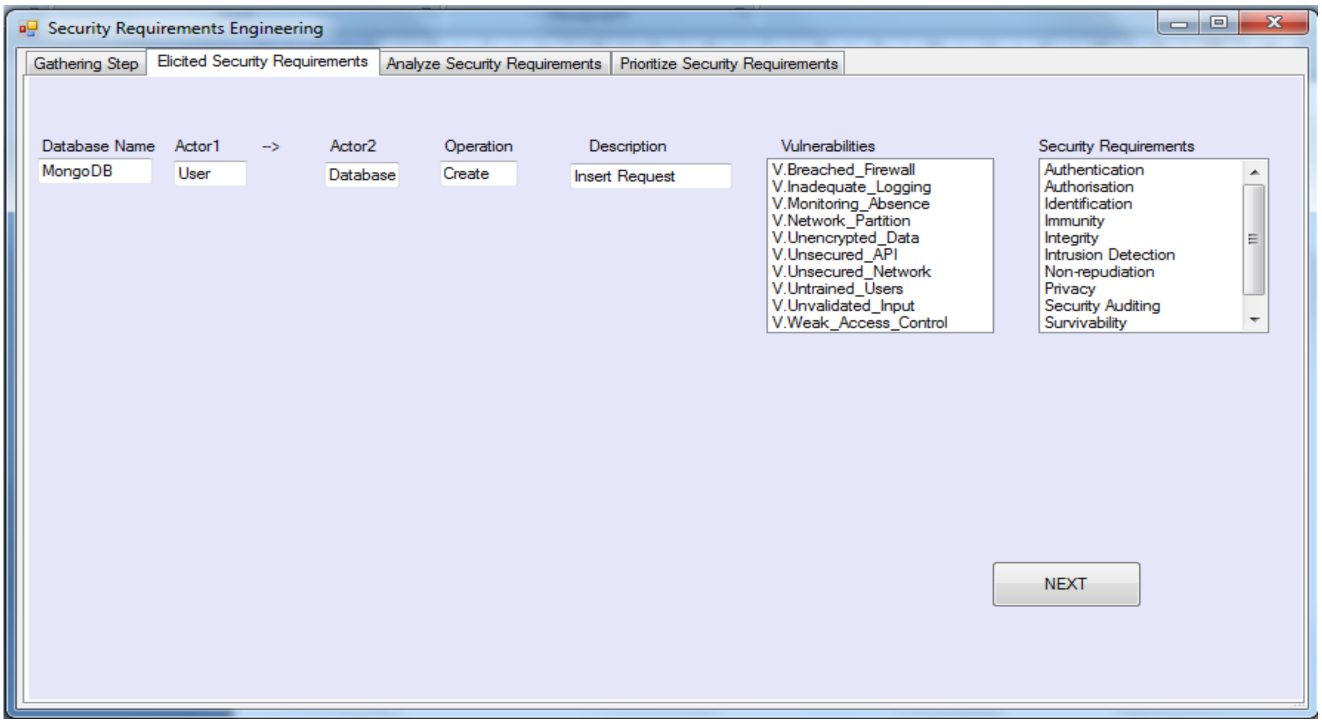
Database Operation: Create

Description: Insert Request

Vulnerabilities

- [ ] V.Misconfigurations
- [x] V.Weak_Access_Control
- [x] V.Unencrypted_Data
- [x] V.Breached_Firewall
- [x] V.Unsecured_API
- [x] V.Unsecured_Network
- [x] V.Network_Partition
- [x] V.Unvalidated_Input
- [x] V.Untrained_Users
- [x] V.Monitoring_Absence
- [x] V.Inadequate_Logging
- [ ] V.Physical_Security
- [ ] V.Obsolete_System

Figure 7.3 Window showing fields required for elicitation process

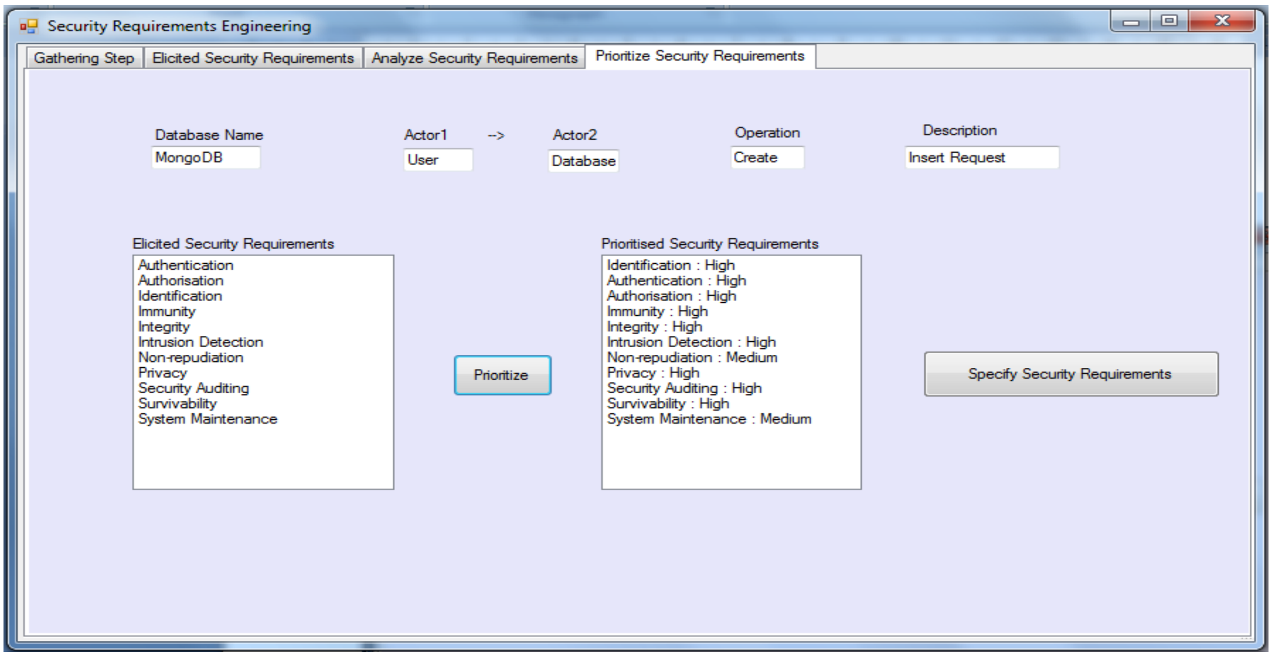

Security Requirements Engineering

Gathering Step | Elicited Security Requirements | Analyze Security Requirements | Prioritize Security Requirements

Database Name	Actor1	-->	Actor2	Operation	Description	Vulnerabilities	Security Requirements
MongoDB	User		Database	Create	Insert Request	V.Breached_Firewall V.Inadequate_Logging V.Monitoring_Absence V.Network_Partition V.Unencrypted_Data V.Unsecured_API V.Unsecured_Network V.Untrained_Users V.Unvalidated_Input V.Weak_Access_Control	Authentication Authorisation Identification Immunity Integrity Intrusion Detection Non-repudiation Privacy Security Auditing Survivability

Figure 7.4 Elicited Security Requirements

Further, the elicited security requirements are prioritized. Prioritization window is shown in Figure 7.5.

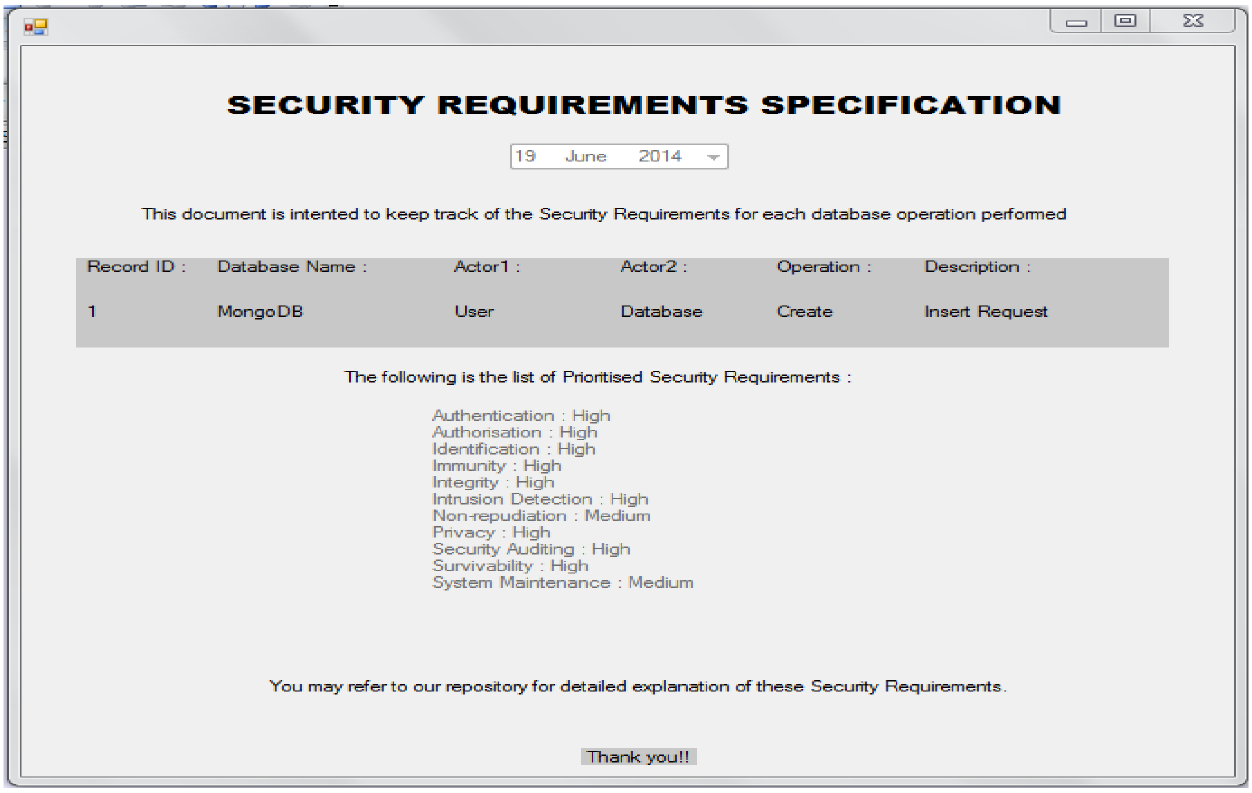

Security Requirements Engineering
Gathering Step
Elicited Security Requirements
Analyze Security Requirements
Prioritize Security Requirements
Prioritize
Specify Security Requirements

Database Name	Actor1	-->	Actor2	Operation	Description
MongoDB	User		Database	Create	Insert Request

Elicited Security Requirements	Prioritised Security Requirements
Authentication	Identification : High
Authorisation	Authentication : High
Identification	Authorisation : High
Immunity	Immunity : High
Integrity	Integrity : High
Intrusion Detection	Intrusion Detection : High
Non-repudiation	Non-repudiation : Medium
Privacy	Privacy : High
Security Auditing	Security Auditing : High
Survivability	Survivability : High
System Maintenance	System Maintenance : Medium

Figure 7.5 Prioritized Security Requirements

After the elicitation and prioritization of security requirements, these requirements are specified. Specified security requirements are shown in Figure 7.6. The prioritize and specified security requirements are stored in the Security Requirements base.



SECURITY REQUIREMENTS SPECIFICATION

19 June 2014

This document is intented to keep track of the Security Requirements for each database operation performed

Record ID :	Database Name :	Actor1 :	Actor2 :	Operation :	Description :
1	MongoDB	User	Database	Create	Insert Request

The following is the list of Prioritised Security Requirements :

Authentication : High
Authorisation : High
Identification : High
Immunity : High
Integrity : High
Intrusion Detection : High
Non-repudiation : Medium
Privacy : High
Security Auditing : High
Survivability : High
System Maintenance : Medium

You may refer to our repository for detailed explanation of these Security Requirements.

Thank you!!

Figure 7.6 Specified Security Requirements

Information Extractor. Different information is extracted from the repositories created. Back-end architecture of SREPS is shown in Figure 7.7 (a) and Figure 7.7 (b).

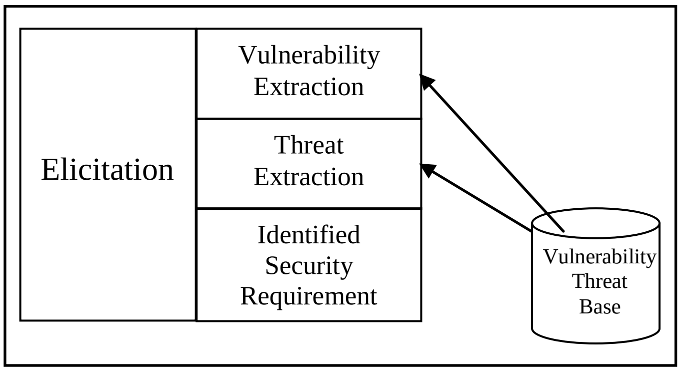


Figure 7.7 (a) Back End Architecture of SREPS

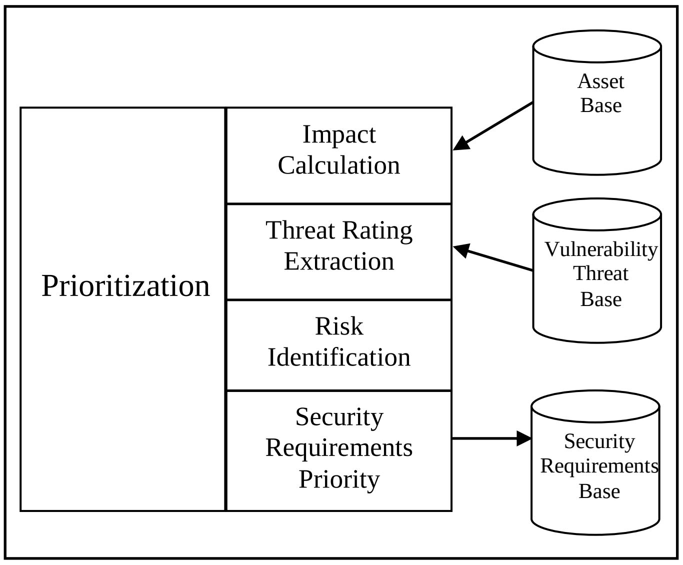
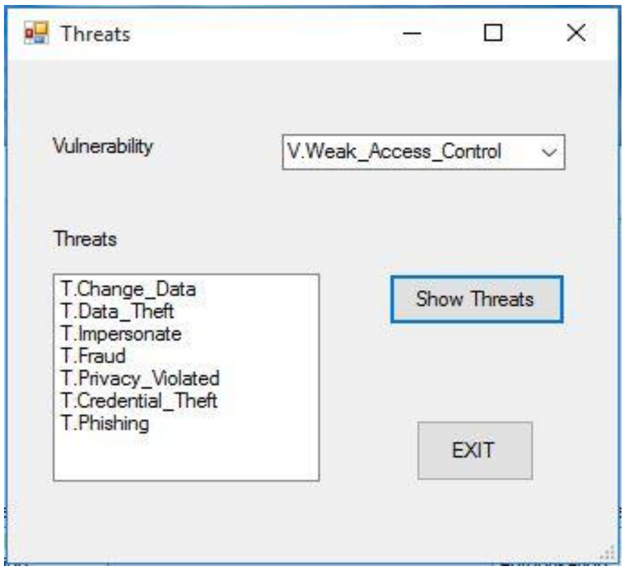


Figure 7.7 (b) Back End Architecture of SREPS

Threats corresponding to vulnerabilities are extracted from the database based on the criteria specified in the form of a matching table as presented in Chapters 4, 5 and 6. Threats Corresponding to vulnerability 'V.Weak Access Contol' is shown in Figure 7.8 for Big Data DB environment.

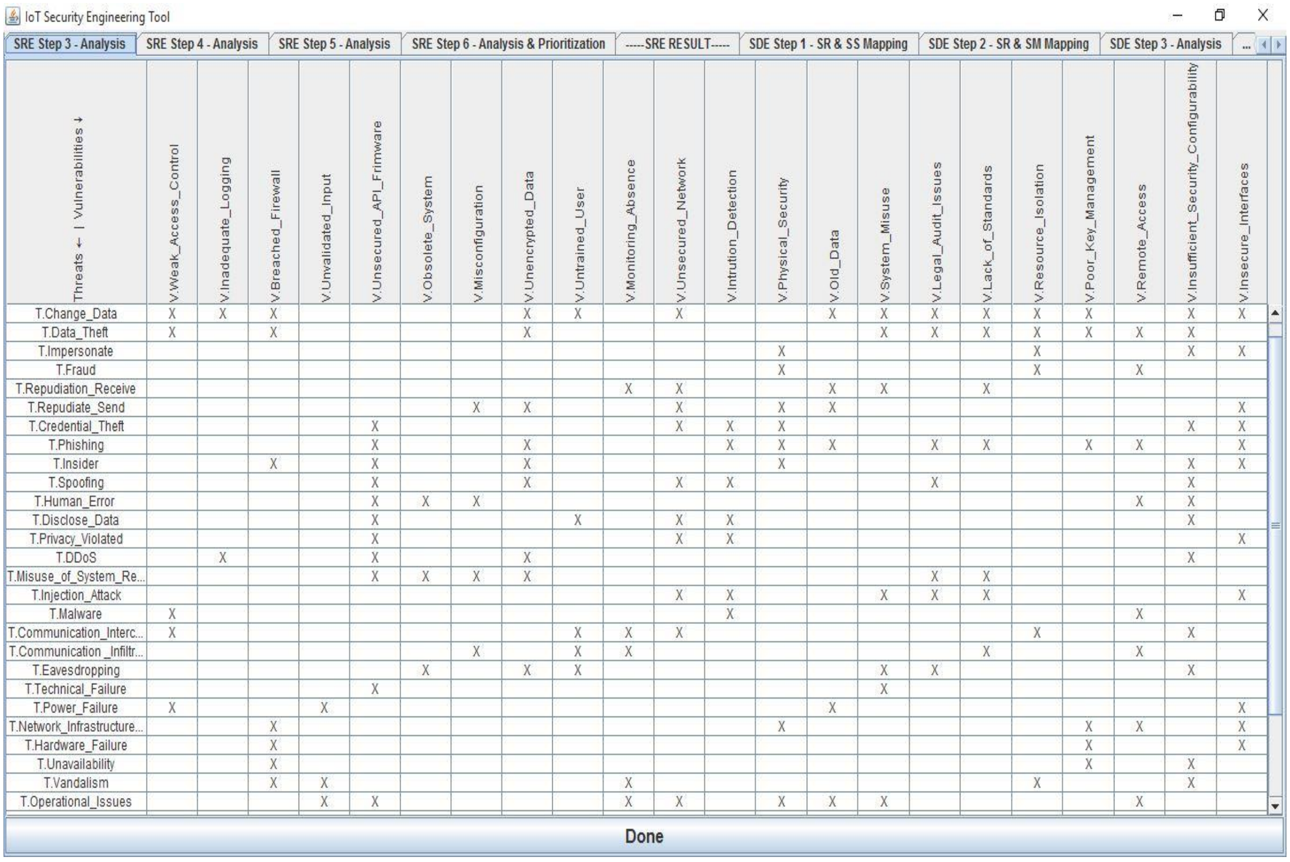


V.Weak_Access_Control

T.Change_Data
T.Data_Theft
T.Impersonate
T.Fraud
T.Privacy_Violated
T.Credential_Theft
T.Phishing

Figure 7.8 Threats Corresponding to Selected Vulnerability

Repository of vulnerability-threat mapping for IoT system is shown in Figure 7.9. For the purpose of prioritization of security requirements impact value is calculated. The impact is calculated using analyzing the information of assets affected by threats. Figure 7.10 shows the mapping of affected assets by threats for IoT based-systems.

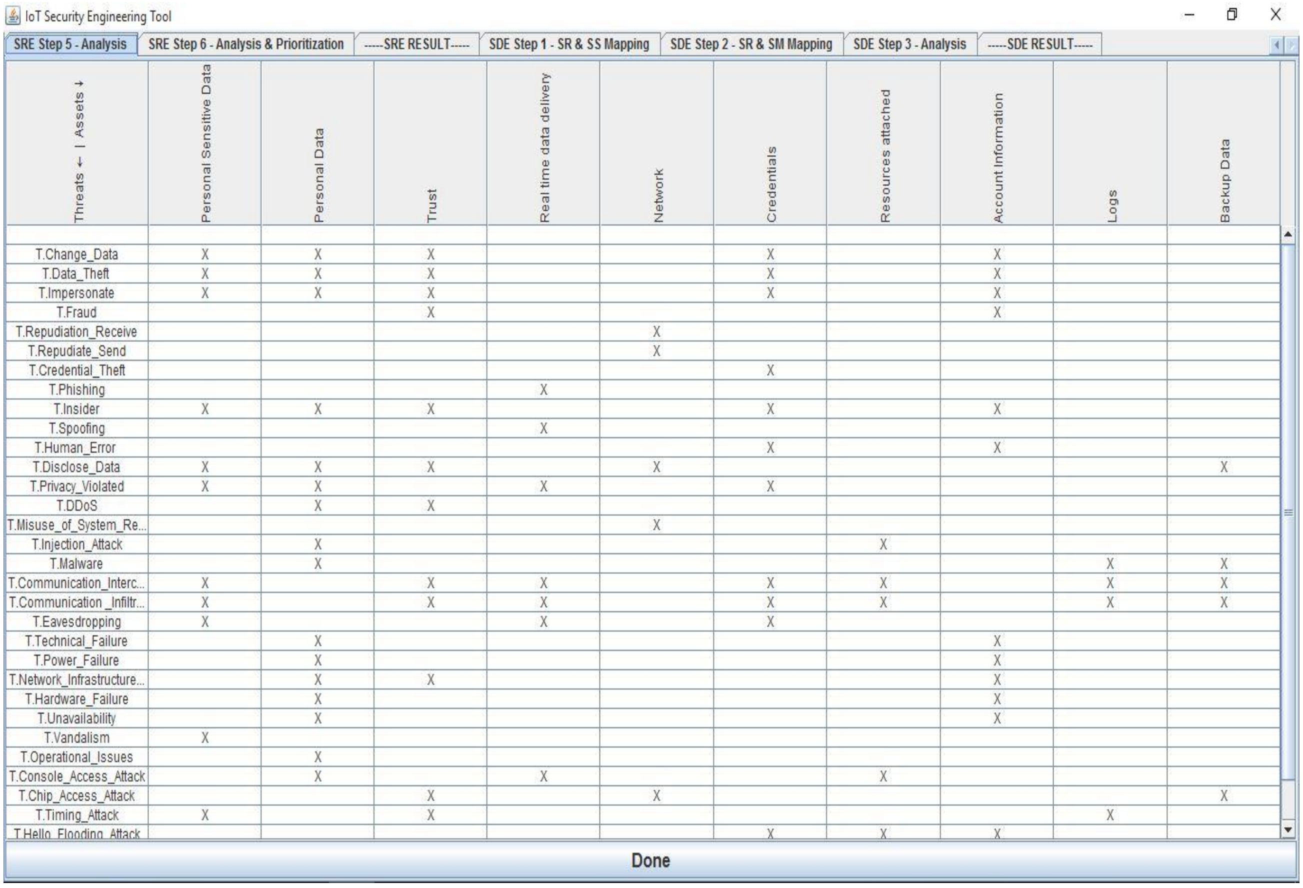

IoT Security Engineering Tool

SRE Step 3 - Analysis | SRE Step 4 - Analysis | SRE Step 5 - Analysis | SRE Step 6 - Analysis & Prioritization | -----SRE RESULT----- | SDE Step 1 - SR & SS Mapping | SDE Step 2 - SR & SM Mapping | SDE Step 3 - Analysis | ...

Threats ← | Vulnerabilities →	V.Weak_Access_Control	V.Inadequate_Logging	V.Breached_Firewall	V.Unvalidated_Input	V.Unsecured_API_Firmware	V.Obsolete_System	V.Misconfiguration	V.Unencrypted_Data	V.Untrained_User	V.Monitoring_Absence	V.Unsecured_Network	V.Intrution_Detection	V.Physical_Security	V.Old_Data	V.System_Misuse	V.Legal_Audit_Issues	V.Lack_of_Standards	V.Resource_Isolation	V.Poor_Key_Management	V.Remote_Access	V.Insufficient_Security_Configurability	V.Insecure_Interfaces
T.Change_Data	X	X	X					X	X		X			X	X	X	X	X	X		X	X
T.Data_Theft	X		X					X							X	X	X	X	X	X	X	
T.Impersonate													X					X			X	X
T.Fraud													X					X		X		
T.Repudiation_Receive										X	X			X	X		X					
T.Repudiate_Send							X	X			X		X	X								X
T.Credential_Theft					X						X	X	X								X	X
T.Phishing					X			X				X	X	X		X	X		X	X		X
T.Insider			X		X			X					X								X	X
T.Spoofing					X			X			X	X				X					X	
T.Human_Error					X	X	X													X	X	
T.Disclose_Data					X				X		X	X									X	
T.Privacy_Violated					X						X	X										X
T.DDoS		X			X			X													X	
T.Misuse_of_System_Re...					X	X	X	X								X	X					
T.Injection_Attack											X	X			X	X	X					X
T.Malware	X											X								X		
T.Communication_Interc...	X								X	X	X							X			X	
T.Communication _Infiltr...							X		X	X							X			X		
T.Eavesdropping						X		X	X						X	X					X	
T.Technical_Failure					X										X							
T.Power_Failure	X			X										X								X
T.Network_Infrastructure...			X										X						X	X		X
T.Hardware_Failure			X																X			X
T.Unavailability			X																X		X	
T.Vandalism			X	X						X								X			X	
T.Operational_Issues				X	X					X	X		X	X	X					X		

Done

Figure 7.9 Vulnerability-Threat Mapping Repository


IoT Security Engineering Tool

SRE Step 5 - Analysis | SRE Step 6 - Analysis & Prioritization | -----SRE RESULT----- | SDE Step 1 - SR & SS Mapping | SDE Step 2 - SR & SM Mapping | SDE Step 3 - Analysis | -----SDE RESULT-----

Threats ← | Assets →	Personal Sensitive Data	Personal Data	Trust	Real time data delivery	Network	Credentials	Resources attached	Account Information	Logs	Backup Data
T.Change_Data	X	X	X			X		X		
T.Data_Theft	X	X	X			X		X		
T.Impersonate	X	X	X			X		X		
T.Fraud			X					X		
T.Repudiation_Receive					X					
T.Repudiate_Send					X					
T.Credential_Theft						X				
T.Phishing				X						
T.Insider	X	X	X			X		X		
T.Spoofing				X						
T.Human_Error						X		X		
T.Disclose_Data	X	X	X		X					X
T.Privacy_Violated	X	X		X		X				
T.DDoS		X	X							
T.Misuse_of_System_Re...					X					
T.Injection_Attack		X					X			
T.Malware		X							X	X
T.Communication_Interc...	X		X	X		X	X		X	X
T.Communication _Infiltr...	X		X	X		X	X		X	X
T.Eavesdropping	X			X		X				
T.Technical_Failure		X						X		
T.Power_Failure		X						X		
T.Network_Infrastructure...		X	X					X		
T.Hardware_Failure		X						X		
T.Unavailability		X						X		
T.Vandalism	X									
T.Operational_Issues		X								
T.Console_Access_Attack		X		X			X			
T.Chip_Access_Attack			X		X					X
T.Timing_Attack	X		X						X	
T.Hello_Flooding_Attack						X	X	X		

Done

Figure 7.10 Repository of Threats affecting Assets

7.3 Security Design Engineering (SDE)

The architecture of SDE is shown in Figure 7.11 (a) and Figure 7.11 (b).

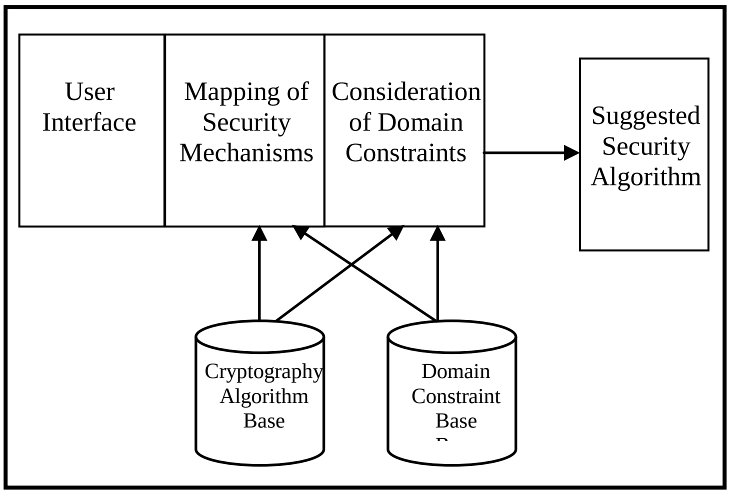

User Interface
Mapping of Security Mechanisms
Consideration of Domain Constraints
Suggested Security Algorithm
Cryptography Algorithm Base
Domain Constraint Base

Figure 7.11 (a). Architecture of SDE

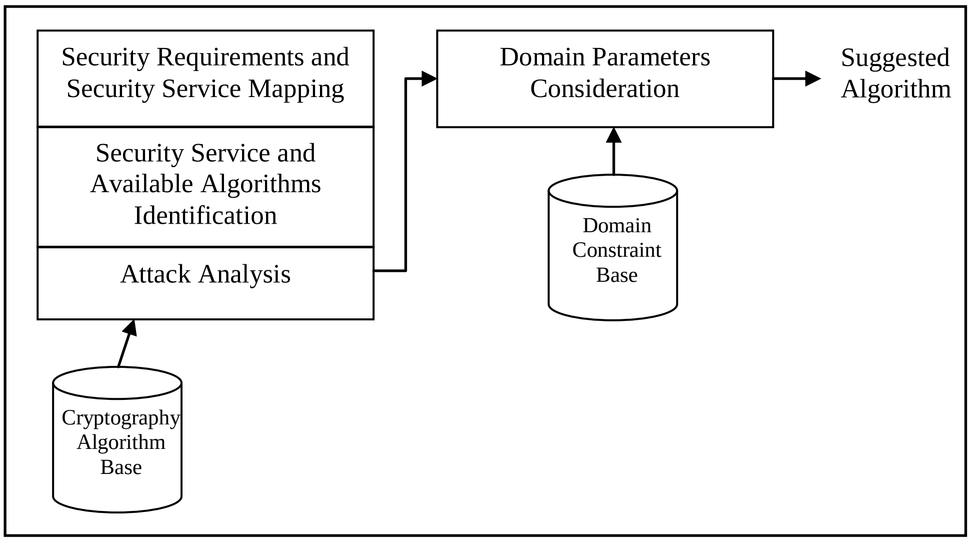
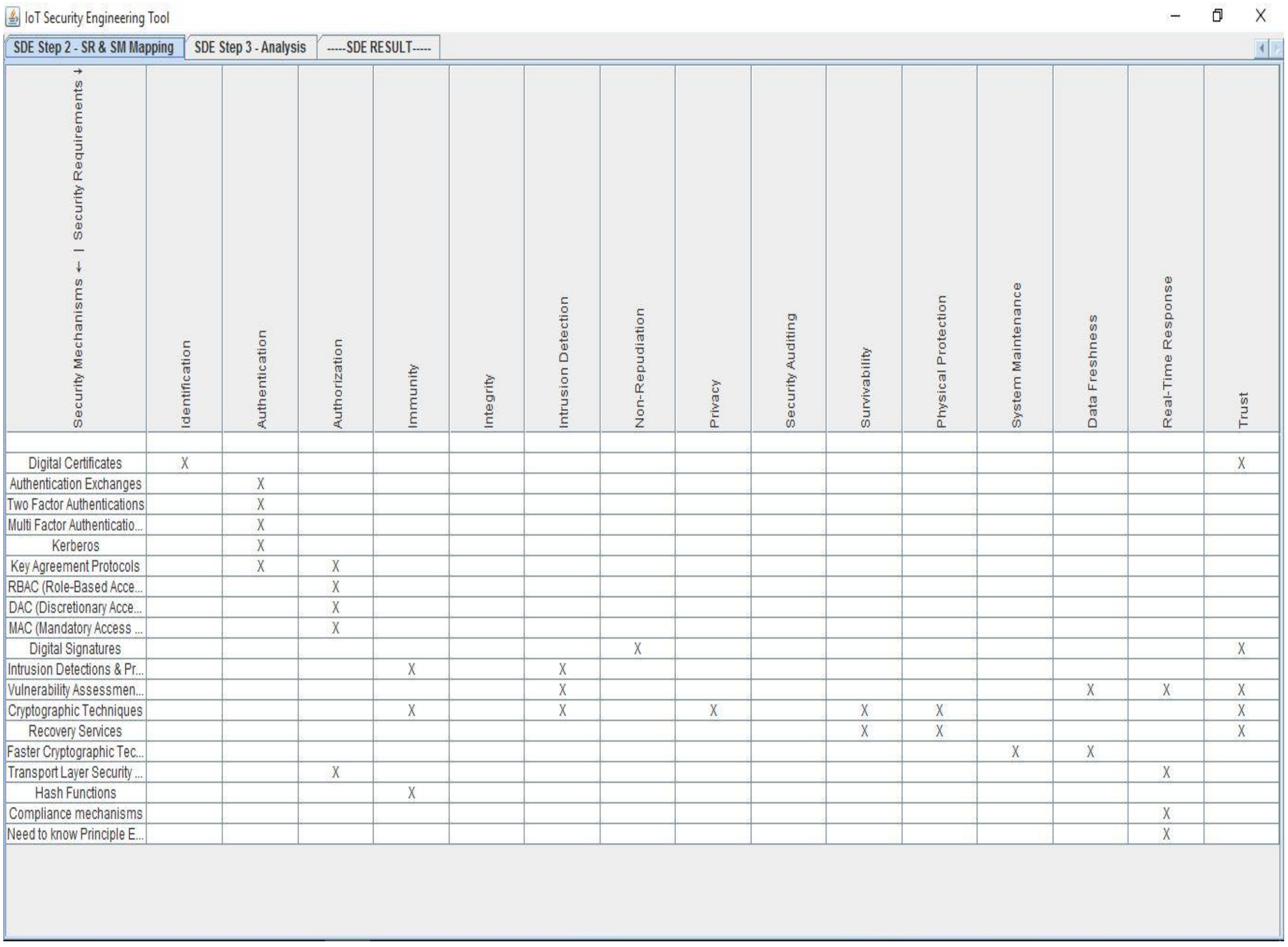

Security Requirements and Security Service Mapping
Security Service and Available Algorithms Identification
Attack Analysis
Domain Parameters Consideration
Suggested Algorithm
Domain Constraint Base
Cryptography Algorithm Base

Figure 7.11 (b). Architecture of SDE

Cryptography Algorithm base contains the information about attacks mitigated by various available cryptography algorithms. This would help in the identification security algorithm for implementation. To achieve this task mapping of security requirements to security mechanisms is done as shown in Figure 7.12. for IoT-based systems. The result of this step would be the algorithm with the highest attack mitigation rate.

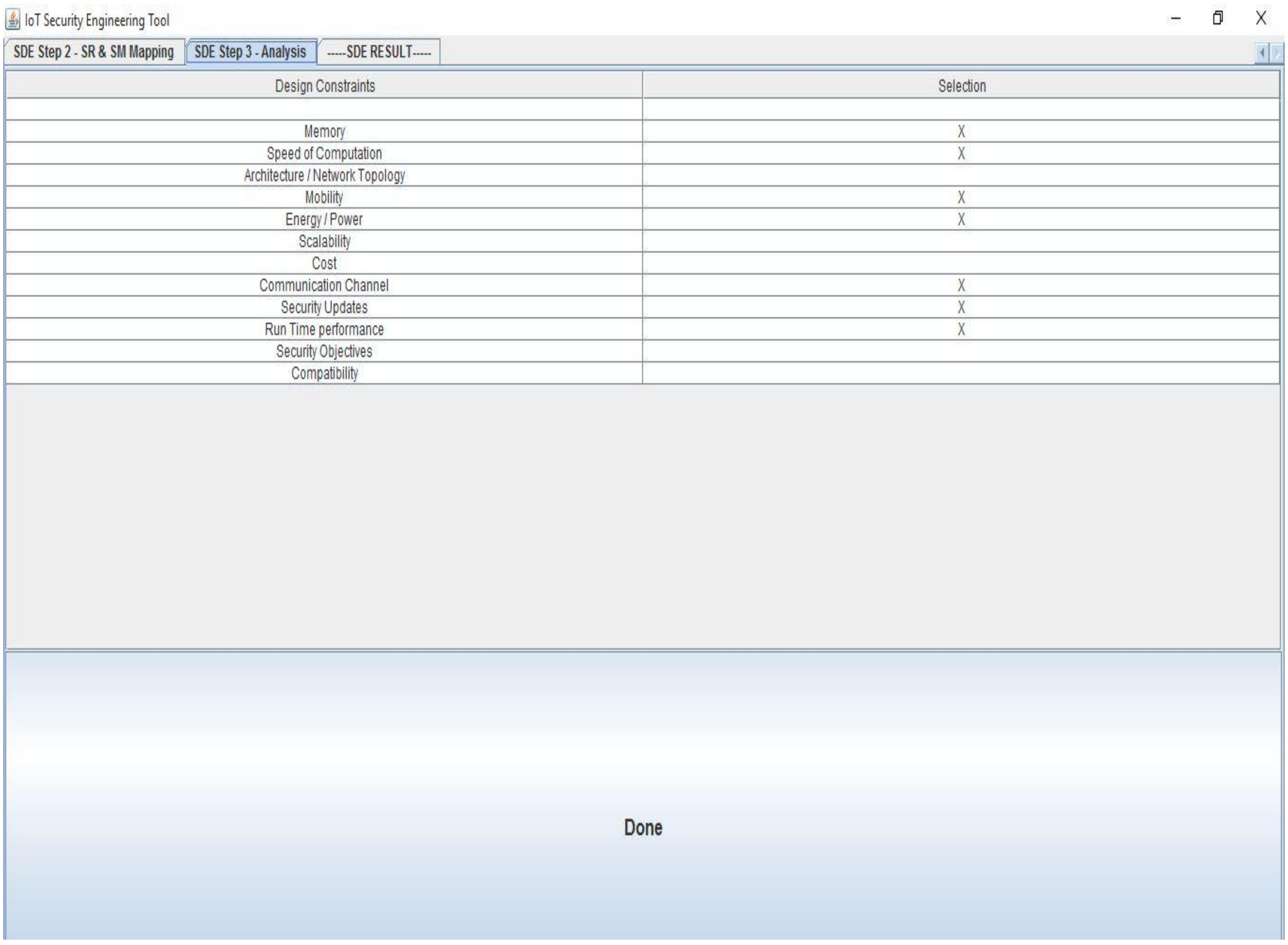


Security Mechanisms ← | Security Requirements →	Identification	Authentication	Authorization	Immunity	Integrity	Intrusion Detection	Non-Repudiation	Privacy	Security Auditing	Survivability	Physical Protection	System Maintenance	Data Freshness	Real-Time Response	Trust
Digital Certificates	X														X
Authentication Exchanges		X													
Two Factor Authentications		X													
Multi Factor Authenticatio...		X													
Kerberos		X													
Key Agreement Protocols		X	X												
RBAC (Role-Based Acce...			X												
DAC (Discretionary Acce...			X												
MAC (Mandatory Access ...			X												
Digital Signatures							X								X
Intrusion Detections & Pr...				X		X									
Vulnerability Assessmen...						X							X	X	X
Cryptographic Techniques				X		X		X		X	X				X
Recovery Services										X	X				X
Faster Cryptographic Tec...												X	X		
Transport Layer Security ...			X											X	
Hash Functions				X											
Compliance mechanisms														X	
Need to know Principle E...														X	

Figure 7.12 Repository of Security Mechanisms and Security Requirements Mapping

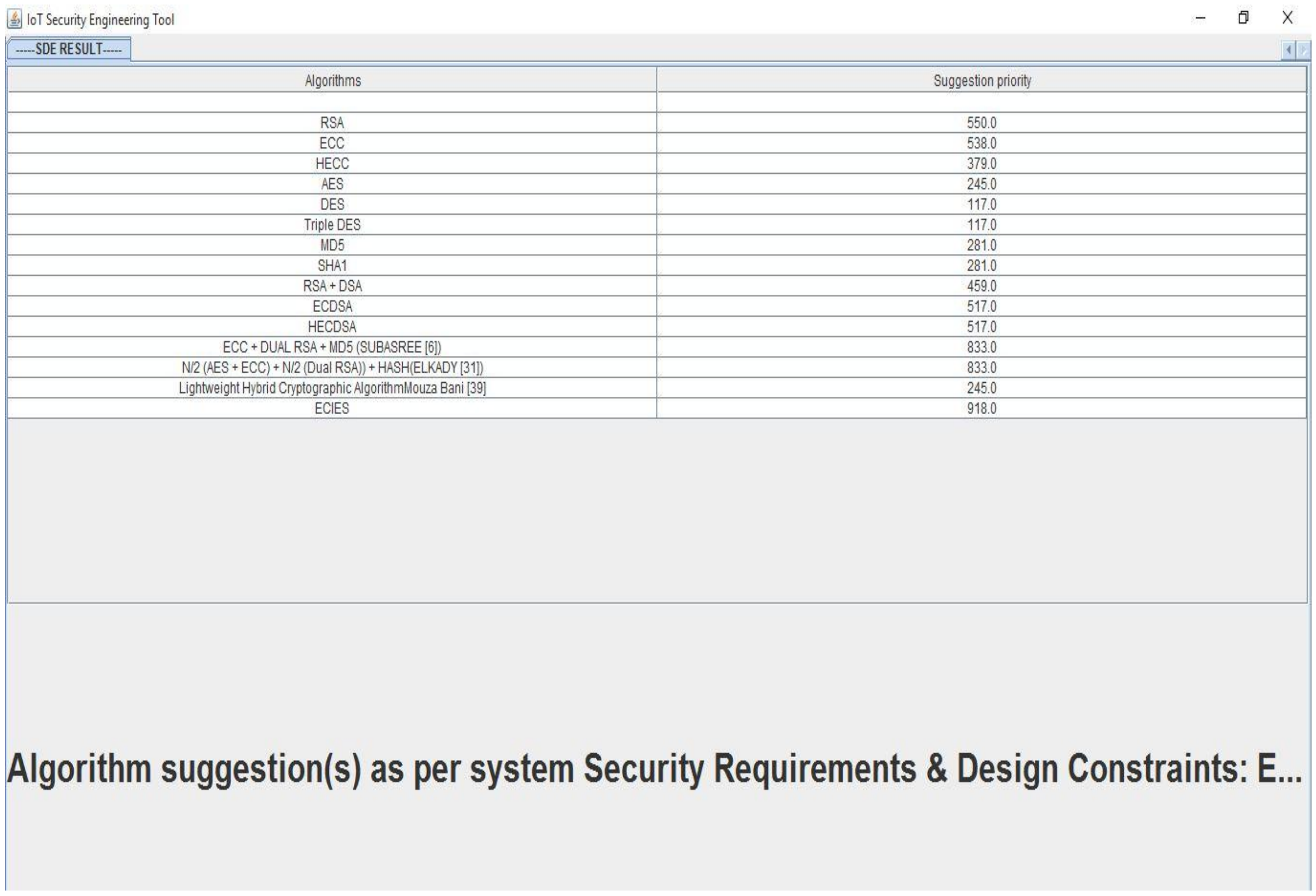
Next, the Domain Constraint base contains the applicable list of domain constraints for the system. Different applicable domain constraints are chosen for the system under consideration. Figure 7.13 depicts the choosing of domain constraints for IoT-based systems. This step would help in choosing efficient security algorithm for implementation.

Based on the application of two repositories ‘ECIES’ is suggested for encryption, as it mitigates the maximum number of threats and works efficiently under given domain constraints among the available algorithms. Figure 7.14 depicts the algorithm with their priority value, an algorithm with highest priority value will be chosen for implementation. Hence in case of IoT system algorithm, ‘ECIES’ is selected for implementation.



Design Constraints	Selection
Memory	X
Speed of Computation	X
Architecture / Network Topology	
Mobility	X
Energy / Power	X
Scalability	
Cost	
Communication Channel	X
Security Updates	X
Run Time performance	X
Security Objectives	
Compatibility	

Figure 7.13 Selection of Domain Constraints



Algorithms	Suggestion priority
RSA	550.0
ECC	538.0
HECC	379.0
AES	245.0
DES	117.0
Triple DES	117.0
MD5	281.0
SHA1	281.0
RSA + DSA	459.0
ECDSA	517.0
HECDSA	517.0
ECC + DUAL RSA + MD5 (SUBASREE [6])	833.0
N/2 (AES + ECC) + N/2 (Dual RSA)) + HASH(ELKADY [31])	833.0
Lightweight Hybrid Cryptographic AlgorithmMouza Bani [39]	245.0
ECIES	918.0

Algorithm suggestion(s) as per system Security Requirements & Design Constraints: E...

Figure 7.14 Suggested Security Algorithm for Implementation

Summary. This chapter of the thesis discussed the architecture and working of tool SET: (1) SREPS and (2) SDE. Sample repositories and screen shots of tool are shown and discussed.

CHAPTER 8

CONCLUSIONS, CONTRIBUTIONS AND FUTURE SCOPE

From the literature review, this research found that existing methodologies focus on identification of security goals, threats, attack, vulnerability to the asset. They specify security requirements in the form of protection measures to meet the security goals. These measures are nothing but the architectural constraints. Thus the present methodologies do not distinguish between architectural constraints and security requirements. Developing a secure software is a complex task in the web-based systems. More challenges are faced in emerging fields of cloud computing, IoT, and big data. Therefore, an engineering approach to secure software development requires a methodology which should first elicit, analyze, prioritize, and specify the security requirements during requirements engineering phase. Then, it should deploy the security algorithms based on different constraints (computational, communicational, device) during design engineering phase.

The finding of the thesis has established, that current research has following gaps:

- They elicit, analyze, prioritize the security goals to functional requirements/ assets, but specify the security requirements in the form of architectural constraints.
- When a security measure is specified as an architectural constraint, it does not consider the domain constraints such as memory, power, computation speed, etc. This may result in unnecessary constraints which make the system slow.

- None of the approaches validates deployed security algorithms for identified threats or attacks. Analogous to traditional software engineering the system must validate the embedded security.
- Most of the methodologies focus on web-based systems; they do not consider the emerging domains like cloud computing, Internet of Things (IoT).

While addressing these challenges, the thesis suggests *a novel security engineering framework to handle security issues which are inherent in the system in a structured manner. The proposed framework can become an integral part of current software development processes.*

- The generic framework presented in Figure 8.1 is divided into three phases namely security requirements engineering, security design engineering, and security testing. In security requirements engineering phase, security requirements are identified, analyzed, prioritized and finally specified. After that in security design engineering phase, an efficient algorithm is selected based on different constraints. Finally, during the Security Testing phase, a metric is generated showing the system security level after deployment of selected security algorithms. This metric is then compared with the pre-defined threshold value to check whether the security provided by selected algorithms is sufficient or not. For the purpose of validation open source software ‘wordpress’ is taken, and found that our proposal is effectively identifying and recommending suitable security algorithm for implementation.

After that, the proposed framework is applied to various new emerging domains like cloud computing, IoT, big data. These new emerging fields are chosen because they have a complex architecture and embedding security is difficult due to the presence of numerous functionalities and assets.

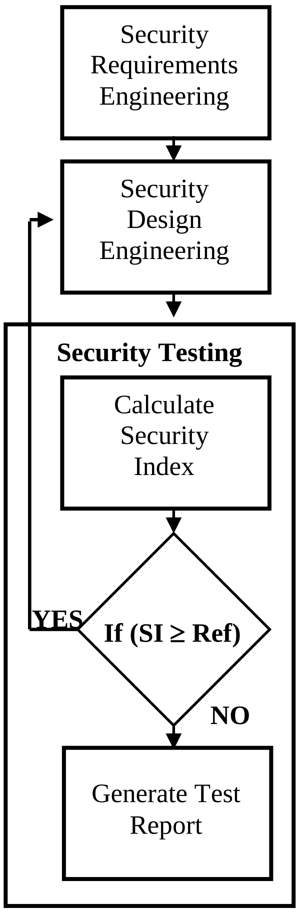


Figure 8.1 Security Engineering Framework

a) **CLOUD COMPUTING**

- It elicits, analyzes, prioritizes, and specify the Security Requirements along with the functional and non- functional requirements.
- To cater the need of handling a large set of functionalities and assets, a <u>Functionality-Asset mapping table having a dimension (34X 22)</u> has been developed. The table acts as a repository from where functionalities are chosen and corresponding assets are listed automatically.

- Another table for <u>vulnerability-threat mapping of the dimension of (39 X 45)</u> has been developed. This table would help in the identification of potential threats at different vulnerable points.
- A new security requirement named multi-trust was added to the cluster of security requirements defined by researcher Firesmith.
- Security algorithms are chosen considering various domain constraints (environmental consideration, communicational and computational parameters, and type of devices used) during the design engineering phase.
- Finally, a security metric is generated showing system security level.
- Guidelines are generated for implementing security in the development new cloud system.
- For the purpose of validation open source software 'ownCloud' is taken, and found that our proposal is efficient in identifying and recommending suitable security algorithm for implementation.

b) **IOT**

- As done in the cloud computing, different assets involved at various layers of IoT are identified and then to cater the identification of vulnerabilities, threats, security requirements following repositories are built:
    - A vulnerability threat mapping table was constructed of a dimension <u>39 X 22</u>
    - Various threats that can affect assets of the system are identified whose dimension is approximate <u>39 X 30</u>.
- New security services: Data Freshness and Liability, are added to the pre-defined cluster of security services.

- Security algorithms are chosen considering various domain constraints pertaining to different layers of IoT during the design and validation phase.
- Finally, a security metric is generated showing system security level. Also, a case study of remote health monitoring system scenario is considered for explanation of our novel proposal.

c) **BIG DATA DATABASES**

- Possible Security Requirements are identified, analyzed, prioritized, and specified for NoSQL databases.
- During the requirements engineering phase to automate the process of identification of security requirements we have generated
    - A vulnerability-threat mapping table for identification of vulnerabilities and threats.
    - A threat-security requirements mapping table is built to enable easy identification of security requirements.
- Based on various constraints (source of data generated, type of data generated, etc.) best-suited security algorithms are chosen for implementation. Also, security guidelines for involved stakeholders are generated.
- We have applied and analyzed our proposal on open source software ‘MongoDB’, a commonly used NoSQL database and it is found that our proposal is effectively identifying and recommending suitable security algorithm for implementation.

Contributions of the thesis

A summary of the contributions made in the thesis is as follows:

1. A generic framework for security engineering has been proposed for solving the security issues. The phases of proposed framework work together to achieve the main goal of the system, which is its security.
2. A Security Requirements Engineering process is established to do the identification, analysis, prioritization, and specification of security requirements.
3. Similar to functional and non-functional requirements, security requirements are explicitly embedded in SRS document as a separate section to enable starting of security consideration from starting phases of SDLC.
4. A process for security design engineering is presented, to identify efficient algorithms for implementation of security requirements in a structured way based on domain parameters (computational constraints such as bandwidth, energy, etc. and device constraints such as memory, power, etc.).
5. The Testing process is developed to measure the effectiveness of selected security algorithms in terms of a metric denoted by Security Index. Metric is used as performance measure that will show the threat proneness of the system. Also, security index value will show the appropriateness of selected security algorithms based on the number of threats they mitigate.
6. A tool has been developed to assist the user in elicitation of security requirements. Various repositories mentioned in previous chapters are maintained, from where with the help of queries data is fetched to assist the user.
7. The proposed process is adapted for various emerging domains namely cloud systems, IoT, and big data databases.
8. Open source software’s namely wordpress, ownCloud, MongoDB are taken for the purpose of validation.

a) Cloud-based systems:

- New security requirements Multi-Trust is identified.
- Generated functionality-asset mapping and vulnerability- threat mapping tables, which guides the user to identify the assets, functionality, vulnerabilities, threats efficiently. Further, it helps in identification of security requirements.
- Also, it selects the efficient algorithm for implementation based on domain constraints.
- Further various cloud storage models are evaluated.

b) IoT systems:

- New security requirements Data Freshness and Trust are identified.
- Generated vulnerability-threat mapping table and database of threats affecting assets. These repositories would help in identification of assets, vulnerabilities, threats. This will in turn, help in the identification of security requirements.
- Constraints for IoT layers are identified such as (a) Light weight, (b) Consume less power, (c) Require less computation, (d) Less storage space, and (e) Need to work on things such as RFID tags, embedded systems, sensors, etc.
- Based on foregoing constraints algorithm is suggested for Remote Health Monitoring system. But, the suggested algorithm is not efficient, so need to develop a new hybrid algorithm arises.

c) Big Data Databases:

- Generated vulnerability-threat mapping and threat- security requirements mapping tables. These tables would help in identification and prioritization of security requirements.
- Constraints for Big Data environments are identified such as Source of Data Generation, Type of Information, Complexity of Data, Type of Data Organization, and Throughput.
- Existing big data database MongoDB is evaluated.

Future work and open problems

The thesis has given a number of directions of future work as follows:

1. During the design phase, optimal algorithms are selected from the set of standard available algorithms. The standard algorithms can implement specific security services only. Hence, new hybrid algorithms are required that can implement more than one security service based on the system's domain constraints.
2. During the elicitation process, this thesis has considered N: M mapping of threats and vulnerabilities. But it may be possible that combination of vulnerabilities and threats may lead to new security issue. Hence, combination of threats and vulnerabilities need to be considered, as sometime lower order threats may get combined and create a serious problem.
3. Automation of our process of vulnerability and threat identification, security requirements elicitation, and appropriate algorithm suggestion can be done by applying machine learning algorithms. Hence, this area needs to be explored.

REFERENCES

1. Alexander, I. (2003). Misuse Cases: Use Cases with Hostile Intent. *IEEE Software, 20*(1), 58-66.
2. Ambhore, P., Waghmare, V., & Meshram, B. (2007). A Implementation of Object Oriented Database Security. *International Conference on Software Engineering Research, Management and Applications*, (pp. 359-365).
3. Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., . . . Zaharia, M. (2010, April). A View of Cloud Computing. *Communications of the ACM, 53*(4), 50-58.
4. Barnard-Wills, D., Marinos, L., & Portesi, S. (2014). *Threat Landscape and Good Practice Guide for Smart Home and Converged Media.* European Union Agency for Network and Information Security (ENISA). ENISA.
5. Bate, B. (2017, June 22). *wordpress-now-powers-28-websites*. Retrieved from envato.com: https://envato.com/blog/wordpress-now-powers-28-websites/
6. Beckers, K., Cote, I., Fabbender, S., Heisel, M., & Hofbauer, S. (2013). A pattern-based method for establishing a cloud- specific information security management system. *Requirements Engineering, 18*, 343-395.
7. Bertino, E., & Sandhu, R. (2005, January-March). Database Security—Concepts, Approaches, and Challenges. *IEEE Transactions on Dependable and Secure Computing, 2*(1), 2-19.
8. Boehm, B., & Papaccio, P. (1988). Understanding and Controlling Software Costs. *IEEE Transactions on Software Engineering, 14*(10), 1462-1477.
9. Braberl, F., Hogganvik, I., Lund, M., Stølen, K., & Vraalsen, F. (2007). Model-based security analysis in seven steps—a guided tour to the CORAS method. *BT Technology Journal, 25*(1), 101–117.

10. Chatterjee, K., Gupta, D., & De, A. (2013, June). A framework for development of secure software. *CSI Transactions on ICT, 1*(1), 143-157.

11. Cloud Security Alliance, CSA. (2013). *The Notorious Nine Cloud Computing Top Threats in 2013.* CSA. CSA.

12. Coman, A., & Ronen, B. (2010). Icarus predicament: Managing the pathologies of overspecification and overdesign. *International Journal of Project Management, 28*(3), 237-244.

13. Common Criteria Implementation Board. (1999). *Common Criteria for Information Technology Security Evaluation, Technical Report, CCIMB 99–031.* Technical.

14. CRAMM. (2005). *United Kingdom Central Computer and Telecom- munication Agency (CCTA) Risk analysis and management method, CRAMM user guide, Issue 5.1.* . United Kingdom.

15. CSA Cloud Security Alliance. (2010). *Top Threats to Cloud Computing V1.0.* Cloud Security Alliance.

16. CVE. (2004). *Wordpress : Vulnerability Statistics*. Retrieved from cvedetails: https://www.cvedetails.com/product/4096/Wordpress-Wordpress.html?vendor_id=2337

17. CVE Details. (2012). *Owncloud: Vulnerability Statistics*. Retrieved from www.cvedetails.com: https://www.cvedetails.com/product/22262/Owncloud-Owncloud.html?vendor_id=11929

18. CVE. (2013). *Mongodb: Vulnerability Statistics*. Retrieved from www.cvedetails.com: https://www.cvedetails.com/product/25450/Mongodb-Mongodb.html?vendor_id=12752

19. Department for Business Innovation & Skills. (2014). *2013 Information Security Breaches Survey Technical Report.* Retrieved from http://www.pwc.co.uk: http://www.pwc.co.uk/assets/pdf/cyber-security-2014-technical-report.pdf

20. *DropBox storage service*. (2012). Retrieved march 2016, from https://www.dropbox.com/

21. *Dropbox: Yes, We Were Hacked.* (2012, August). Retrieved december 2015, from http://gigaom.com/cloud/dropbox-yes-we-were-hacked/

22. IEEE 830. (1998). *IEEE Recommended Practice for Software Requirements Specifications.* USA: IEEE.

23. IEEE Standard, 610.12. (1990). *IEEE Standard Glossary of Software Engineering Terminology*. IEEE.

24. ENISA. (2009, November). *Cloud Computing Benefits, risks and recommendations for information security.* Retrieved from The European Network and Information Security Agency: http://www.enisa.europa.eu

25. Ellison, R. J. (2005). *Attack Trees.* Software Engineering Institute. Carnegie Mellon University.

26. Fabian, B., Gurses, S., Heisel, M., Santen, T., & Schmidt, H. (2010, March). A comparison of Security Requirements engineering methods. *Requirement Engineering Journal, 15*(1), 7-40.

27. Fernandes, D., Soares, L., Gomes, J., Freire, M., & In´acio, P. (2014). Security Issues in Cloud Environments — A Survey. *International Journal of Information Security (IJIS), 13*(2), 113-170.

28. Ficco, M., Palmieri, F., & Castiglione, A. (2015, April 22). Modeling security requirements for cloud-based system development’. *Concurrency and Computation: Practice and Experience, 27*(8), 2107-2124.

29. Finkelstein, A., & Fuks, H. (1989, May). Multiparty Specification. *ACM SIGSOFT Software Engineering Notes, 14*(3), 185-195.

30. Firesmith, D. G. (2003). Security Use Cases. *Journal of Object Technology, 2*(3), 53-64.

31. Firesmith, D. G. (2003). Engineering Security Requirements. *Journal of Object Technology, 2*(1), 53-68.

32. Flyvbjerg, F., & Budzierkdsjfk, A. (2011). Why Your IT Project Might Be Riskier Than You Think. *Harvard Business Review, 89*(9), 23-25.

33. Forouzan, B. (2007). *Cryptography & Network Security.* New York, USA: McGraw-Hill.

34. Gartner. (2015). *Gartner: 21 Billion IoT Devices To Invade By 2020*. Retrieved from Gartner: 21 Billion IoT Devices To Invade By 2020: http://www.informationweek.com/mobile/mobile-devices/gartner-21-billion-iot-devices-to-invade-by-2020/d/d-id/1323081

35. Goguen, J., & Linde, C. (1993). Techniques for requirements elicitation. *Proceedings of IEEE International Symposium on Requirements Engineering* (pp. 152-164). San Diego, CA, USA, USA: IEEE.

36. Granjal, J., Monteiro, E., & Silva, J. S. (2015). Security for the Internet of Things: A Survey of Existing Protocols and Open Research Issues. *IEEE COMMUNICATION SURVEYS & TUTORIALS, 17*(3), 1294-1312.

37. Grobauer, B., Walloschek, T., & Stocker, E. (2011). Understanding cloud computing vulnerabilities. *Security Privacy, 9*(2), 50-57.

38. Haley, C., Laney, R., Moffett, J., & Nuseibeh, B. (2008). Security Requirements Engineering: A Framework for Representation and Analysis. *IEEE Transactions on Software Engineering, 34*(1), 133-153.

39. Hankerson, D., Hernandez, J. L., & Meneze, A. (2000). Software Implementation of Elliptic Curve Cryptography over Binary Fields. *Koç Ç.K., Paar C. (eds) Cryptographic Hardware and Embedded Systems — CHES 2000. CHES 2000. Lecture Notes in Computer Science. 1965*, pp. 1-24. Berlin: Springer.

40. Hickey, A., & Davis, A. (2003). Barriers to Transferring Requirements Elicitation Techniques to Practice. *Business Information Systems Conf.* IEEE.

41. IDC. (2016, [online]). *Explosive Internet of Things Spending to Reach \$1.7 Trillion in 2020*. Retrieved from Available: http://www.idc.com/getdoc.jsp?containerId=prUS25658015.

42. *Internet_of_things*. (n.d.). Retrieved 2016, from Wikipedia: IoT Wikipedia, https://en.wikipedia.org/wiki/Internet_of_things

43. Islam, S., Mouratidis, H., & Edgar, R. W. (2011). A Goal- Driven Risk Management Approach to Support Security and Privacy Analysis of Cloud-Based System. In E. F.-M. Mario Piattini, *Security Engineering for Cloud Computing.* IGI Global.

44. Islam, S., Kwak, D., Kabir, M., Hossain, M., & Kwak, K.-S. (2015). The Internet of Things for Health Care: A Comprehensive Survey. *IEEE Access, 3*, 678-708.

45. Jansen, W. (2011). Cloud Hooks: Security and Privacy Issues in Cloud Computing. *Proceedings of the 44th Hawaii International Conference on System Sciences*, (pp. 1-10).

46. Jing, Q., Vasilakos, A., Wan, J., Lu, J., & Qui, D. (2014, November). Security of the Internet of Things: perspectives and challenges. *Wierless Networks, 20*(8), 2481- 2501.

47. Jules, A. (2006, 6 06). RFID security and privacy: a research survey. *EEE Journal On Selected Areas In Communications*, *24*(2), 381-394.

48. Katal, A., Wazid, M., & Goudar, R. (2013). Big data: Issues, challenges, tools and Good practices. *Sixth International Conference on Contemporary Computing (IC3)* (pp. 404-409). IEEE.

49. Kotonya, G., & Sommerville, I. (1996, January). Requirements engineering with viewpoints. *Software Engineering Journal, 11*(1), 5-18.

50. Kuppuswamy, P., & Al-Khalidi, S. (2014). Analysis of Security Threats and Prevention in Cloud Storage: Review Report. *International Journal of Advanced Research in Engineering and Applied Sciences, Vol. 3, No. 1.*

51. Lamsweerde, A. V. (2004). Elaborating Security Requirements by Construction of Intentional Anti-Models. (pp. 148-157). Washington, DC USA: 26th International Conference on software engineering (ICSE'04).

52. Lapouchnian, A. (2005). *Goal-Oriented Requirements Engineering: An Overview of the Current Research.* University Of Toronto, Department of Computer Science .

53. Liu, F., Tong, J., Mao, J., Bohn, R., Messina, J., Badger, L., & Leaf, D. (2011). *NIST Cloud Computing Reference Architecture: Recommendations of the National Institute of Standards and Technology.* Gaithersburg: NIST Special Publication 500-292.

54. Maurice, W., Heemels, H., R. Teel, A., Wouw, N. v., & Dragan, N. (2010, August 8). Networked Control Systems With Communication Constraints: Tradeoffs Between Transmission Intervals, Delays and Performance. *IEEE TRANSACTIONS ON AUTOMATIC CONTROL, 55*(8), 1781-1796.

55. McDermott, J., & Fox, C. (1999). Using Abuse Case Models for Security Requirements Analysis. *ACSAC '99 Proceedings of the 15th Annual Computer Security Applications Conference.* Washington, DC, USA: IEEE.

56. Mead, N. R. (2005). Security Quality Requirements Engineering (SQUARE) Methodology. *Software Engineering for Secure Systems (SESS05) proceedings of International Workshop on Requirements for High Assurance Systems.* St. Louis.

57. Mellado, D., Medina, E. F., & Piattini, M. (2007, February). A common criteria based Security Requirements engineering process for the development of secure information system. *Computer Standards & Interfaces, 29*(2), 244-253.

58. Miorandi, D., Sicari, S., Pellegrini, F., & Chlamtac, I. (2012). Internet of Things: Vision, applications and research challenges. Ad Hoc Networks, 10, 1496-1516.

59. Mitrokotsa, A., Beye, M., & Peris-Lopez, P. (2010). Classification of RFID Threats based on Security Principles. 1-27.

60. MongoDB. (2014). *MongoDB CRUD Operations Introduction Release 2.2.7.* MongoDB. MongoDB.

61. MongoDB. (2013). *MongoDB security features.* Retrieved Feburary 16, 2014, from https://docs.mongodb.org/maual/release-notes

62. Mouratidis H., G. P. (2002). A Natural Extension of Tropos Methodology for Modeling Security. *Agent-Oriented Methodologies Workshop, Annual ACM Conference on Object Oriented Programming, Systems, Languages (OOPSLA).* Seattle- USA.

63. Mouratidis, H., & Giorgini, P. (2007). Security Attack Testing (SAT)—testing the security of information systems at design time. *Information Systems, 32*, 1166–1183.

64. Newton, D. (2011). *Dropbox authentication: insecure by design.* Retrieved from dereknewton.com: http://dereknewton.com/2011/04/dropbox-authentication-static-host-ids/

65. Naveed, R., & Abbas, H. (2014). Security Requirements Specification Framework for Cloud Users . In *Future Information Technology* (Vol. 276, pp. 297-305). Lecture Notes in Electrical Engineering.

66. O'Donnell, L. (2016, [online]). *IOT Predictions for 2016*. Retrieved from IOT Predictions for 2016: http://www.crn.com/slide-shows/networking/300079629/10-iot-predictions-for-2016.htm?itc=refresh

67. Okman, L., Gal-Oz, N., Gonen, Y., Gudes, E., & Abramov, J. (2011). Security Issues in NoSQL Databases. *10th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)* (pp. 341-347). Changsha, China: IEEE.

68. OWASP. (2004). *OWASP_Risk_Rating_Methodology*. Retrieved Feburary 2014, from OWASP: https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology

69. Ozkan, S. (n.d.). *CVE Details*. Retrieved from CVE Details: https://www.cvedetails.com/

70. Pan, L. (2009). A Unified Network Security and Fine-Grained Database Access Control Model. *Second International Symposium on Electronic Commerce and Security, ISECS '09.* , (pp. 265-269). Nanchang City, China,.

71. Pearson, S., & Benameur, A. (2010). Privacy, Security and Trust Issues Arising from Cloud Computing. *2nd IEEE International Conference on Cloud Computing Technology and Science* (pp. 693- 702). IEEE.

72. Roman, R., Najera, P., & Lopez, J. (2011, September 12). Securing the Internet of Things. *Computer, 44*(9), 51-58.

73. Rose, K., Eldridge, S., & Chapin, L. (2015). The Internet of Things: An Overview Understanding the Issues and Challenges of a More Connected World. The Internet Society. The Internet Society.

74. Sen, J. (2013). Security and Privacy Issues in Cloud Computing. In P. G.-L. Ruiz-Martinez, *Architectures and Protocols for Secure Information Technology.* USA: IGI- Global.

75. Sindre, G., & Opdahl, A. L. (2005, January). Eliciting security requirements with misuse cases. *Requirements Engineering, 10*(1), 34-44.

76. Sommerville, I. (2004). *Software Engineering (7th Edition).* Pearson Addison Wesley.

77. Sood, K., Yu, S., & Xiang, Y. (2015, September 28). Software Defined Wireless Networking Opportunities and Challenges for Internet of Things: A Review. *IEEE Internet of Things Journal, 3*(4), 453-463.

78. Suchman, L., & Jordan, B. (1990). Interactional troubles in face-to-face survey interviews. *Journal of the American Statistical Association, 85*(409), 232-241.

79. Stallings, W. (2006). *Cryptography And Network Security: Principles And Practices, 4th Edition.* Pearson.

80. Stankovic, J. (2014, Feburary). Research Directions for the Internet of Things. *IEEE Internet of Things Journal, 1*(1), 3-9.

81. Stocker, E., Grobauer, B., & Walloschek, T. (2011, March/ April). Understanding cloud computing vulnerabilities. *IEEE Security & Privacy, 9*(2), 50-57.

82. Stoneburner, G., Alice, G., & Feringa, A. (2002). *Risk Management Guide for Information Technology Systems: Recommendations of the National Institute of Standards and Technology*. NIST Special Publication 800-30. Gaithersburg: NIST.

83. Subashini, S., & Kavitha, V. (2011). A survey on security issues in service delivery models of cloud computing. *Journal of Network and ComputerApplications, 34*, 1-11.

84. Symantec. (2013). *Internet Security Threat Report.*

85. Trappe, W., Howard, R., & Moore, R. (2015). Low-energy security: Limits and opportunities in the internet of things. EEE Security & Privacy, 13(1), 14-21.

86. T, S., & Kumar, V. (2013, July). Application of Big Data in Data Mining. *International Journal of Emerging Technology and Advanced Engineering, 3*(7), 390-393.

87. Uzunov, A., Falkner, K., & Fernandez, E. (2015). A comprehensive pattern-oriented approach to engineering security methodologies. *Information and Software Technology, 57*, 217-247.

88. Valera, A., Zamora, M., & Skarmeta, A. (2010). An Architecture Based on Internet of Things to Support Mobility and Security in Medical Environments. *Consumer Communications and Networking Conference (CCNC), 2010 7th IEEE*. IEEE.

89. Verizon. (2014). *2014 DATA BREACH INVESTIGATIONS REPORT.*

90. Vormetric. (2012). *Securing Big Data: Security Recommendations for Hadoop and NoSQL Environments.* Vormetric.

91. Vu, Q. H., Pham, T.-V., Truong, H.-L., Dustdar, S., & Asal, R. (2012). DEMODS: A Description Model for Data-as-a-Service. *26th IEEE International Conference on Advanced Information Networking and Applications* (pp. 605-612). IEEE.

92. W3Techs. (n.d.). *World Wide Web Technology Surveys* . Retrieved from https://w3techs.com: https://w3techs.com

93. Websence Security Lab. (2015). *2015 SECURITY PREDICTIONS.* Retrieved from websense.com/securitylabs: http://www.portantier.com/files/websense-report-2015-security-predictions-en.pdf

94. Ware, M. S., Bowles, J. B., & Eastman, C. M. (2006). Using the Common Criteria to Elicit Security Requirements with Use Cases. *SoutheastCon* (pp. 273-278). IEEE.

95. Xiao, Z., & Xiao, Y. (2013). Security and Privacy in Cloud Computing. *IEEE Communications Surveys & Tutorials, 15*(2), 843-859.

AUTHOR'S BRIEF BIOGRAPHY

Shruti Jaiswal is a Research Scholar in the Computer Engineering Department of Delhi Technological University (formerly, Delhi College of Engineering), New Delhi India. She received her ME from the Delhi College of Engineering in Computer Technology and Application in 2009. Her field of interests is information security, software engineering, and requirements engineering.

She has full-time teaching experience of 2+ years where she has taught subjects to undergraduate students. Also, as a research scholar, she has taught many courses to undergraduate and postgraduate students. These include Operating System, Computer Graphics, Software Project Management, Computer Organization. She is currently working as an Assistant Professor in Inderprastha Engineering College, Ghaziabad.