A TRUST BASED REPUTATION MODEL FOR SECURE MULTI AGENT COMMUNICATION IN OPEN ENVIRONMENT

BY

PATRICK KINYUA GIKUNDA

A RESEARCH PROJECT REPORT SUBMITTED IN PARTIAL FULFILLMENT FOR AWARD OF MASTERS OF SCIENCE IN COMPUTER SCIENCE, FROM UNIVERSITY OF NAIROBI

August 2015

DECLARATION

This project report is my original work and has not been presented for any award in any other university or any other institution of higher learning for examination.

…………………………… **Date** …………………….

Patrick Kinyua Gikunda
Reg. No: P58/76336/2012

This research report has been submitted for examination with my approval as the University Supervisor

…………………………………… **Date** ……………………….

Dr. Elisha Opiyo

School of Computing and Informatics
Nairobi University

Table of Contents

DECLARATION .... **I**

LIST OF FIGURES .... **IV**

LIST OF TABLES .... **V**

LIST OF ABBREVIATIONS .... **VI**

DEDICATION .... **VII**

ACKNOWLEDGEMENT .... **VIII**

ABSTRACT .... **IX**

1 CHAPTER ONE:NTRODUCTION **1**

- **1.1 INTRODUCTION** 1
- **1.2 DEFINITIONS OF TERMS** 1
- **1.3 BACKGROUND INFORMATION** 2
- **1.4 PROBLEM STATEMENT** 3
- **1.5 THE GOAL** 4
- **1.6 SPECIFIC OBJECTIVES** 4
- **1.7 JUSTIFICATION AND SIGNIFICANCE OF THE PROJECT** 4
- **1.8 ASSUMPTIONS AND LIMITATIONS OF THE RESEARCH PROJECT** 4

2 CHAPTER TWO: LITERATURE REVIEW **5**

- **2.1 INTRODUCTION** 5
- **2.2 AGENT** 5
- **2.3 MULTI AGENTS SYSTEM** 6
- **2.4 BAYESIAN NETWORK-BASED TRUST MODEL** 7
  - ***2.4.1 Trust Formation*** *7*
  - ***2.4.2 Reputation Estimation*** *7*
  - ***2.4.3 Risk-related decision making*** *7*
- **2.3 EIGENTRUST** 7
- **2.4 REGRET** 8
- **2.5 FCTRUST** 8
- **2.6 GLOBAL TRUST MODEL** 8
- **2.7 SIMULATION** 9
- **2.8 PROPOSED CONCEPTUAL ARCHITECTURE** 9

3 CHAPTER THEREE: METHODOLOGY **11**

- 3.1 INTRODUCTION 11
- 3.2 DATA COLLECTION TECHNIQUE 11
  - ***3.2.1 Literature survey*** *11*
  - ***3.2.2 Expert Consultations and review*** *11*
- **3.3 ANALYSIS AND DESIGN** 11

4 CHAPTER FOUR: SYSTEM ANALYSIS AND DESIGN **13**

- **3.4 ANALYSIS AND DESIGN: PROMETHEUS METHODOLOGY** 13
  - ***3.4.1 System Implementation: Simulation*** *13*
- 3.5 SIMULATION PHASES 15

3.6 SIMULATION ENVIRONMENT 16
4.3 ANALYSIS OF THE DIFFERENT FACTORS RELATED TO EVALUATING THE TRUST 16
4.3.1 Satisfaction 16
4.3.2 Similarity 17
4.3.3 Feedback Credibility 17
4.3.4 Direct Trust 17
4.3.5 Indirect Trust 17
4.3.6 Recent Trust 17
4.3.7 Historical Trust 17
4.3.8 Expected Trust 18
4.3.9 Decay model 18
4.3.10 Deviation Reliability 18
4.3.11 Overall Trust Metric 18
4.4 OUR SECURITY MODEL 20
4.4.1 Authentication 20
4.4.2 Key Management 20
4.4.3 Denial of Service Protection 20
4.5 RSA ALGORITHM 20

5 CHAPTER FIVE: MODEL IMPLEMENTATION CONCEPTUAL MODELING 22

5.1 SYSTEM REQUIREMENTS 22
5.2 MODULES 22
5.3 PROCESS FLOW DIAGRAM 23

6. CHAPTER SIX: EXPERIMENTATION, RESULTS & DISCUSSION 26

6.1 AGENT'S BEHAVIORAL PATTERN 26
6.2 TRANSACTION SETTING 26
6.3 PERFORMANCE EVALUATION INDEX 26
6.4 EXPERIMENT SIMULATION 26
6.5 EXPERIMENT TESTING 27
6.6 SNIFFING 28
6.6.1 HOW IS SNIFFING DONE? 29
6.7 EXPERIMENT RESULTS 30
6.8 Discovered Vulnerabilities 31

7. CHAPTER SEVEN: CONCLUSION & FUTURE WORKS 32

7.1 SUGGESTED FURTHER WORK 32

8. REFERENCES 33

9. APPEDIX A: SCREEN SHOT OF VULNERABILITY AUDIT REPORT 35

10. APPENDIX B: SAMPLE CODE 35

List of Figures

Figure 1: Model Architecture, Source Author 10
Figure 2: Summary of pro & cons of simulation........ 14
Figure 3: Simulation Phases 15
Figure 4: Trust Computation Flow Chart. Source: Atul Saurabh and Swapnil Parikh 19
Figure 5: Process Flow, Source Author 23
Figure 6: OSI Layers 29
Figure 7: Sniffing........ 30

LIST OF TABLES

Table 1: STR over localhost 27
Table 2: STR over Internet 27
Table 3: Discovered Vulnerabilities 31
Table 4: Non Secure Message 31

LIST OF ABBREVIATIONS

VPN	Virtual Private Network
P2P	Peer to Peer
MAS	Multi Agent System
MANET	Mobile Ad Hoc Network
CA	Certification Authorities

DEDICATION

To my family, who have supported me every step of the way, through the good and the tough times.

ACKNOWLEDGEMENT

I wish to appreciate and thank the Board of Post Graduate Studies of the University of Nairobi for giving me an opportunity to pursue this course. I sincerely wish to express my appreciation to my supervisor Dr. Elisha T. O. Opiyo for his immense support and guidance throughout, he has been a tremendous mentor to me, without whose support this report would not have been complete. I also wish to appreciate the support given to me by Lecturers who taught me throughout the entire course.

Thank you all.

ABSTRACT

Security and privacy issues have become critically important with the fast expansion of multi-agent systems. Most network applications such as pervasive computing, grid computing and P2P networks can be viewed as multi-agent systems which are open, anonymous and dynamic in nature. Such characteristics of multi-agent systems introduce vulnerabilities and threats to providing secured communication. One feasible way to minimize the threats is to evaluate the trust and reputation of the interacting agents. Many trust/reputation models have done so, but they fail to properly evaluate trust when malicious agents start to behave in an unpredictable way. Most trust/reputation models have not yet addressed this issue. So, to cope with the strategically altering behavior of malicious agents; we present a trust based reputation model for secure multi agent communication in open environment. First analyze the different factors related to evaluating the trust of an agent and then propose a model for effective secure agent's communication. Simulation results indicate that our model compared to other existing models can effectively cope with strategic behavioral change of malicious agents.

1 CHAPTER ONE:NTRODUCTION

1.1 Introduction

In a multi-agent system, agents interact with each other to achieve a definite goal that they cannot achieve alone and such systems include P2P, grid computing, the semantic web, pervasive computing and MANETs. Multi-agent Systems (MASs) are increasingly becoming popular in carrying valuable and secured data over the network. Nevertheless, the open and dynamic nature of MAS has made it a challenge to operate MAS in a secured environment for information transaction. Malicious agents are always seeking ways of exploiting any existing weakness in the network. This is where trust and reputation play a critical role in ensuring effective interactions among the participating agents. Researchers have long been utilizing trust theory from social network to construct trust models for effectively suppressing malicious behaviors of participating agents. Trust issues have become more and more popular since traditional network security approaches such as the use of fire-wall, access control and authorized certification cannot predict agent behavior from a ‘trust’ viewpoint.

This research project involves theory review, evaluation, model development and simulation. This study involves methodical use of a conceptual framework towards solving a clearly defined problem.

1.2 Definitions of Terms

An agent is a computer program that performs various actions continuously and autonomously on behalf of an individual. For example, an agent may archive various computer files or retrieve electronic messages on a regular schedule.

An intelligent agent possess, to varying degrees, autonomy, mobility, a symbolic model of reality, a capacity to learn from experience, and an ability to cooperate with other agents and systems.

Multi Agent Systems is a computerized system composed of multiple interacting intelligent agents within an environment.

Secure Communication ensures integrity and confidentiality of the message being passed across.

simjava is a process based discrete event simulation package for Java, similar to Jade's Sim++, with animation facilities.

Grid computing is the collection of computer resources from multiple locations to reach a common goal. The grid can be thought of as a distributed system with non-interactive workloads that involve a large number of files

Semantic Web is an extension of the Web through standards by the World Wide Web Consortium (W3C). The standards promote common data formats and exchange protocols on the Web

Pervasive computing (also called ubiquitous computing) is the growing trend towards embedding microprocessors in everyday objects so they can communicate information. The words pervasive and ubiquitous mean "existing everywhere." Pervasive computing devices are completely connected and constantly available.

Simulation is the imitation of the operation of a real-world process or system over time.

1.3 Background information

As explained by Steinmetz & Wehrle (2005) Multi-Agent Systems (MASs) are increasingly becoming popular in carrying valuable and secured data over the network. Nevertheless, the open and dynamic nature of MAS as described by Reza, Hakimi and Zahra (2012) has made it a challenge to operate MAS in a secured environment for information transaction. Malicious agents are always seeking ways of exploiting any existing weakness in the network. More than seventy six million households risk of losing their money through vulnerable networks as indicated by Jessica, Matthew and Nicole (2014). As MAS application evolve more confidential and secure data is put at risk as reported by Evan (2015) where more than 4.5 million records were stolen from health facility. Attackers use sophisticated malware to issue attacks and this calls for more comprehensive, flexible and robust secure measures.

This is where trust and reputation play a critical role in ensuring effective interactions among the participating agents. MAS is a system composed of multiple interacting intelligent agents within an environment. Multi-agent systems can be used to solve problems that are difficult or impossible for an individual agent or a monolithic system to solve.

Various techniques have been proposed to secure P2P networks over the last decade. Abdul and Hailes (2000) captured the most important characteristics of trust and reputation and proposed the general structure for developing trust and reputation in a distributed system. Most of the later works in this area followed their ideas, but in different application domain.

Network security consists of the provisions and policies adopted by a network administrator to prevent and monitor unauthorized access. Network security involves the authorization of access to data in a network, which is controlled by the network administrator. Security policy should keep the malicious users out and also exert control over potential risky users within the organization. In addition, the security policy should dictate a hierarchy of access permissions, grant users access only to what is necessary for the completion of their work. A network security system usually consists of many components. Ideally, all components work together, which minimizes maintenance and improves security.

Once authenticated, a firewall enforces access policies such as what services are allowed to be accessed by the network users. Though effective to prevent unauthorized access, this component may fail to check potentially harmful content such as worms or Trojans being transmitted over the network. The content or behavior and other anomalies to protect resources, e.g. from denial of service attacks or an employee accessing files at strange times. Individual events occurring on the network may be logged for audit purposes and for later high-level analysis. Network security components often include: Anti-virus and anti-spyware, Firewall to block unauthorized access to network or Virtual Private Networks (VPNs) to provide secure remote access.

1.4 Problem Statement

As discussed by Amirali and White (2013) it is a commonly held position that the main utility of trust and reputation models is minimizing the risk of interacting with others by avoiding interacting with malicious agents. With this view in mind, the principal objective of such models is the detection of untrustworthy agents. One feasible way to minimize the threats from malicious agents is to evaluate their trust and reputation during interactions. As Sarvapali, Ramchurn, Dong and Nicholas(2004) argues, agents make use of trust and reputation models in deciding how, when and who to interact with in a specific context. As seen from the literature review section, many multi agents interaction use trust/reputation models for secure communication, but they fail to properly evaluate trust when malicious agents start to behave in an unpredictable way. The problem thus reduces to authentication the reliable identification of agents' true identity.

1.5 The Goal

The main objective of this project is to formulate model for secure agent communication even in the presence of highly unpredictable and varying malicious behavior in the network by use of delegation of permissions and credibility.

1.6 Specific Objectives

1. Analysis of different parametres used to compute the trust of communicating agents
2. Designing and developing a java based prototype to simulate agents communication.
3. Deploy key pair and Certification Authority to encrypt/decrypt electronic data or transaction, or sign/authenticate the sender and the recipient.
4. Testing the model to evaluate the security and interpreting the simulation results.

1.7 Justification and significance of the Project

The goal of this project is to find methods that allow to building complex secure systems composed of autonomous and heterogeneous agents whose, while operating on local knowledge and possessing only limited abilities, are none the less capable of communicating securely even under unpredictable malicious agents and enacting the desired global behaviors.

1.8 Assumptions and limitations of the research project

We assume an open environment in which agents must interact with other agents with which they are not familiar. In particular, an agent will receive requests and assertions from other agents and must decide how to act on the requests and assess the credibility of the assertions.

In a closed environment, agents have well known and familiar transaction partners whose rights and credibility is known. The problem thus reduces to authentication, the reliable identification of agents' or their true identity. We apply the Multi-Agent System (MAS) concepts to facilitate the authentication and the authorization process in order to work with multi-clients more dynamically and efficiently.

2 CHAPTER TWO: LITERATURE REVIEW

2.1 Introduction

We view Trust Management in Multi agents systems from two perspectives: a “strong and crisp” approach, where decisions are founded on logical rules and verifiable properties encoded in digital credentials, and a “soft and social” approach, based on reputation measures gathered and shared by a distributed community. There exist currently two different major approaches for managing trust: policy-based and reputation-based trust management. The two approaches have been developed within the context of different environments and targeting different requirements. On the one hand, policy-based trust relies on objective “strong security” mechanisms such as signed certificates and trusted certification authorities (CA hereafter) in order to regulate the access of users to services. Moreover, the access decision is usually based on mechanisms with well-defined semantics (e.g., logic programming) providing strong verification and analysis support. The result of such a policy-based trust management approach usually consists of a binary decision according to which the requester is trusted or not, and thus the service (or resource) is allowed or denied. On the other hand, reputation-based trust relies on a “soft computational” approach to the problem of trust. In this case, trust is typically computed from local experiences together with the feedback given by other entities in the network (e.g., users who have used services of that provider).

2.2 Agent

An agent, also called softbot (“software robot”), is a computer program that performs various actions continuously and autonomously on behalf of an individual or an organization. For example, an agent may archive various computer files or retrieve electronic messages on a regular schedule. Such simple tasks barely begin to tap the potential uses of agents, however. This is because an intelligent agent can observe the behavior patterns of its users and learn to anticipate their needs, or at least their repetitive actions. Such intelligent agents frequently rely on techniques from other fields of artificial intelligence, such as expert systems and neural networks. Intelligent agents possess, to varying degrees, autonomy, mobility, a symbolic model of reality, a capacity to learn from experience, and an ability to cooperate with other agents and systems. An intelligent agent is most frequently classified by the role that it performs. For example, interface agents such as Microsoft’s Office Assistant monitor the user’s “desktop” actions and offer advice. Thus far, however, the most useful agents have been developed for Internet assistance. For example, Brewster Kahle, the inventor of the Wide Area Information Server (WAIS) for indexing Web sites, created Alexa, an Internet agent that monitors a user's pattern of Web "surfing" and suggests other sites of possible interest. Chatterbots, another type of Internet agent, provide assistance to Web site visitors by conducting a dialogue with them to determine their needs and to service their more routine requests. Mobile agents are expected to become particularly useful in gathering information—from Internet articles and academic research papers to electronic newspapers, magazines, and books—to match a user's interests. Simple agents have also been used to facilitate trading on eBay, an electronic auction site, as well as on various electronic exchanges. Elaborate multi-agent systems, or communities, are being constructed in which agents meet and represent the interests of their principals in negotiations or collaborations. In addition to agent-only electronic marketplaces, collaborative projects, in which each agent provides some portion of the necessary information, are under development.

2.3 Multi Agents System

A multi-agent system (M.A.S.) is a computerized system composed of multiple interacting intelligent agents within an environment. Multi-agent systems can be used to solve problems that are difficult or impossible for an individual agent or a monolithic system to solve. Intelligence may include some methodic, functional, procedural approach, algorithmic search or reinforcement learning. In artificial intelligence research, agent-based systems technology has been hailed as a new paradigm for conceptualizing, designing, and implementing software systems. Agents are sophisticated computer programs that act autonomously on behalf of their users, across open and distributed environments, to solve a growing number of complex problems. Increasingly, however, applications require multiple agents that can work together. A multi-agent system (MAS) is a loosely coupled network of software agents that interact to solve problems that are beyond the individual capacities or knowledge of each problem solver. In recent years, multi-agent systems (MASs) have received increasing attention in the artificial intelligence community. Research in multi-agent systems involves the investigation of autonomous, rational and flexible behavior of entities such as software programs or robots, and their interaction and coordination in such diverse areas as robotics Kitano (1997), information retrieval and management, and simulation. When designing agent systems, it is impossible to foresee all the potential situations an agent may encounter and specify an agent behavior optimally in advance. Agents therefore have to learn from, and adapt to, their environment, especially in a multi-agent setting.

2.4 Bayesian network-based trust model

In Bayesian network based trust model Wang & Vassieleva (2003) believes that trust is multi-dimensional and agents need to evaluate trust from different aspects of an agent's capability. This makes it easy both to extend the model to involve more dimensions of trust and to combine Bayesian networks to form an opinion about the overall trustworthiness of an entity. Each entity can evaluate his peers according to his own criteria.

The dynamic characteristics of criteria and of peer behavior can be captured by updating Bayesian networks. In our model, we use Bayesian networks for both estimating reputation values of other entities and filtering out unfair raters. Simulations show that the estimation follows the decision maker's subjective, dynamic criteria very well. Bayesian network consists of three components: Trust formation, Reputation estimation and Risk-related decision making.

2.4.1 Trust Formation

Beta probability density functions are used to represent the distribution of trust values according to interaction. We make more detailed analysis of interactions and extend to multi-dimension application specific outcomes. The granularity is determined by the complexity of applications and the requirement of users.

2.4.2 Reputation Estimation

A Bayesian network is constructed to perform the estimation for each dimension of trust. In the Bayesian network, only the nodes corresponding to reliable agents are set to specific states according to the ratings, the other nodes are set to an unknown state.

2.4.3 Risk-related decision making

Agents can select a utility function according to their attitude to risk. An agent with risk-tolerant behavior can choose an exponential function for the utility, a risk neutral agent can choose a linear one and a risk-averse agent can choose a logarithmic function.

2.3 EigenTrust

In EigenTrust Kamvar, Schlosser & Garcia (2003) aggregates the local trust values of all agents to calculate the unique global trust value of a given agent. An agent depends on some pre-trusted agents for trust evaluation in absence of trustworthy recommenders. Even though EigenTrust may work well in social network infrastructure where pre-trusted neighbors (agents) are likely to be trustworthy, but in case of other multi-agent systems like P2P, EigenTrust poses a few problems. Firstly, In P2P network such pre-determined trustworthy agents are not readily available. Secondly, depending on these pretrusted agents create a vulnerability in the sense that if some of these pre-trusted agents get compromised then it will be much easier to launch a large scale malicious attack.

2.4 ReGreT

Jordi and Sierra (2001) explains a reputation model that computes reputation from three dimensions: individual dimension, social dimension and ontological dimension. The individual dimension reflects an agent’s own observation whereas the social dimension considers the opinion of other agents in the community. Finally, the ontological dimension considers reputation as a multi-facet concept and computes reputation from different aspect. While ReGreT considers different dimensions in computing reputation, it does not address the collusion problem associated with computing global reputation.

2.5 FCTrust

HU, Wu and Zhou (2008) uses transaction density and similarity measure to define the credibility of any recommender providing feedback as opposed to which use global trust to weigh the quality of feedbacks. In other words, FCTrust differentiates the role of providing feedbacks from that of providing services. However, FCTrust’s main drawback is that in computing direct trust it retrieves all the transactions performed within a time frame. This imposes storage overhead. SFTrust computes service trust as a weighted average of local trust and recommendation trust, but the weight itself is static and as a result it cannot properly accommodate the experience gained by the evaluating agent over time.

2.6 Global Trust Model

The trust model proposed by Wen, Huaimin, Yan & Peng (2004) is similar to EigenTrust, but it does not consider the use of pre-trusted agents in the calculation of trust. The global trust model we consider is based on binary trust, an agent is either trustworthy or not. In order to disseminate information about transactions agents can forward it to other agents. Since we assume that usually trust exists and malicious behavior is the exception, we just consider information on dishonest interactions as relevant. A social mechanism to detect dishonest behavior will not work for private interactions. Whereas it is straightforward for an agent to collect all information about its own interactions with other agents, it is very difficult for it to obtain all the complaints about any other specific agent. The global trust model we consider is based on binary trust, an agent is either trustworthy or not. In order to disseminate information about transactions agents can forward it to other agents. Since we assume that usually trust exists and malicious behavior is the exception, we just consider information on dishonest interactions as relevant. A social mechanism to detect dishonest behavior will not work for private interactions. Whereas it is straightforward for an agent to collect all information about its own interactions with other agents, it is very difficult for it to obtain all the complaints about any other specific agent.

2.7 Simulation

According to Shannon (1975), digital computer simulation is the process of designing a model of a real system and conducting experiments with this model on a digital computer for a specific purpose of experimentation. Based on the taxonomy given by Nance (1993), digital computer simulation may be divided into three categories: (1) Monte Carlo, (2) continuous, and (3) discrete event. Monte Carlo simulation is a method by which an inherently non-probabilistic problem is solved by a stochastic process; the explicit representation of time is not required. In a continuous simulation, the variables within the simulation are continuous functions, e.g. a system of differential equations. If value changes to program variables occur at precise points in simulation time (i.e. the variables are “piecewise linear”), the simulation is discrete event. Nance (1993) notes that three related forms of simulation are commonly used. A combined simulation refers generally to a simulation that has both discrete event and continuous components. Hybrid simulation refers to the use of an analytical sub model within a discrete event model. Finally, gaming can have discrete event, continuous, and/or Monte Carlo modeling components. The focus of this research is limited to discrete event simulation

2.8 Proposed Conceptual Architecture

We present an approach to the security problems in multi agent systems based on distributed trust and the delegation of permissions, and credibility. We assume an open environment in which agents must interact with other agents with which they are not familiar. In particular, an agent will receive requests and assertions from other agents and must decide how to act on the requests and assess the credibility of the assertions. In a closed environment, agents have well known and familiar transaction partners whose rights and credibility is known. The problem thus reduces to authentication the reliable identification of agents' true identity. We apply the Multi-Agent System (MAS) concepts to facilitate the authentication and the authorization process in order to work with multi-clients more dynamically and efficiently. We are using key pair and Certification Authority to encrypt/decrypt electronic data or transaction, or sign/authenticate the sender and the recipient.

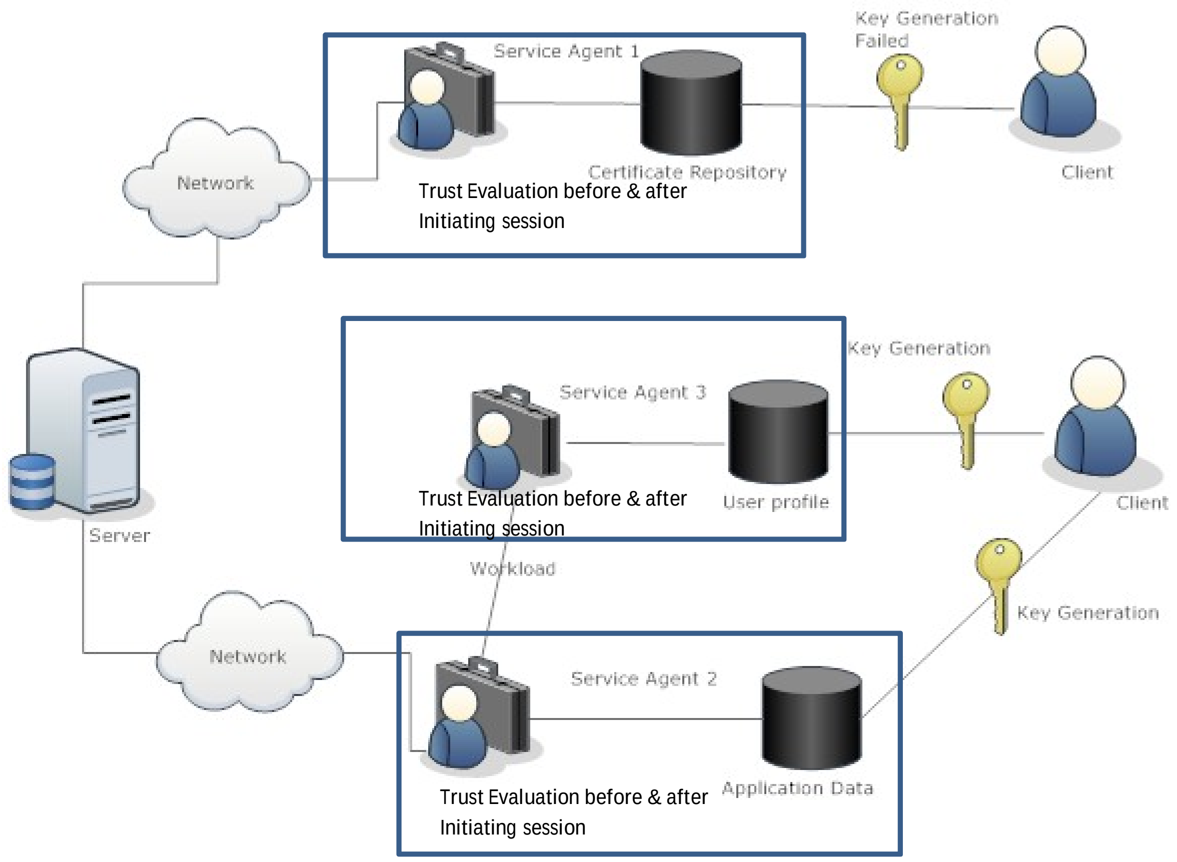


Figure 1: Model Architecture, Source Author

3 CHAPTER THEREE: METHODOLOGY

3.1 Introduction

The main objective of this project is to provide a secure MAS communication model for effective and secure sharing of data among agents even in the presence of highly oscillating malicious behavior.

In order to realize the success of the study, the following steps were taken into consideration;

I. Requirement Gathering
II. Analysis and design of MAS model.
III. Implementation
IV. Results, Testing and Evaluation
V. Documentation and submission of final report

3.2 Data Collection Technique

3.2.1 Literature survey

Relevant literature was reviewed to gain an understanding of the security problems in multi agent systems based on distributed trust and the delegation of permissions, and credibility. Sources of the literature included the internet, journals, newspaper articles, and industry publications.

3.2.2 Expert Consultations and review

Experts in the IT security Management were consulted. This helped in giving deeper insight into the secure information sharing, and specifically MAS communication. MAS Systems Analysts and Security Management were extensively consulted during the problem identification phase, and subsequent phases.

3.3 Analysis and Design

Prometheus methodology was used in the system specification, system design and implementation. The Prometheus methodology is a detailed process for specifying, designing and implementing agent systems.

This methodology was used in the study because it distinguishes itself from other methodologies by supporting the development of intelligent agents, providing start to end support, having evolved out of practical industrial and pedagogical experience, having been both used in the industry and academia. Various activities listed below were undertaken;

System specification consists of the following activities:

- Identify system goals and sub-goals
- Develop use case scenarios
- Identify the agent system's interface to the environment in terms of actions, percepts, and external data
- Identify functionalities
- Identify data read and written by functionalities
- Prepare functionality schemas (name, description, actions, percepts, data used/produced, interaction (with other functionalities), and goals)

Architectural Design consists of the following activities:

- Group functionalities to determine agent types using data coupling and agent acquaintance diagrams to assess alternative groupings
- Define agent types (also define the number and life-cycle of the agent types) and develop agent descriptors
- Produce a system level overview diagram describing the overall structure of the system
- Develop interaction protocols from use case scenarios (via interaction diagrams)

Detailed Design consists of the following activities:

- Develop process diagrams
- Produce agent overview diagrams showing the internal workings of agents in terms of capabilities, events, data and plans
- Refine capability internals (add included capabilities and interactions)
- Introduce plans to handle events
- Define details of events (external, between agents, between capabilities and within agents)
- Define details of plans (relevance, context, subgoals)
- Define details of beliefs/data

4 CHAPTER FOUR: SYSTEM ANALYSIS AND DESIGN

3.4 Analysis and Design: Prometheus Methodology

Prometheus methodology was used in the system specification, system design and implementation. The Prometheus methodology is a detailed process for specifying, designing and implementing agent systems.

This methodology was used in the study because it distinguishes itself from other methodologies by supporting the development of intelligent agents, providing start to end support, having evolved out of practical industrial and pedagogical experience, having been both used in the industry and academia.

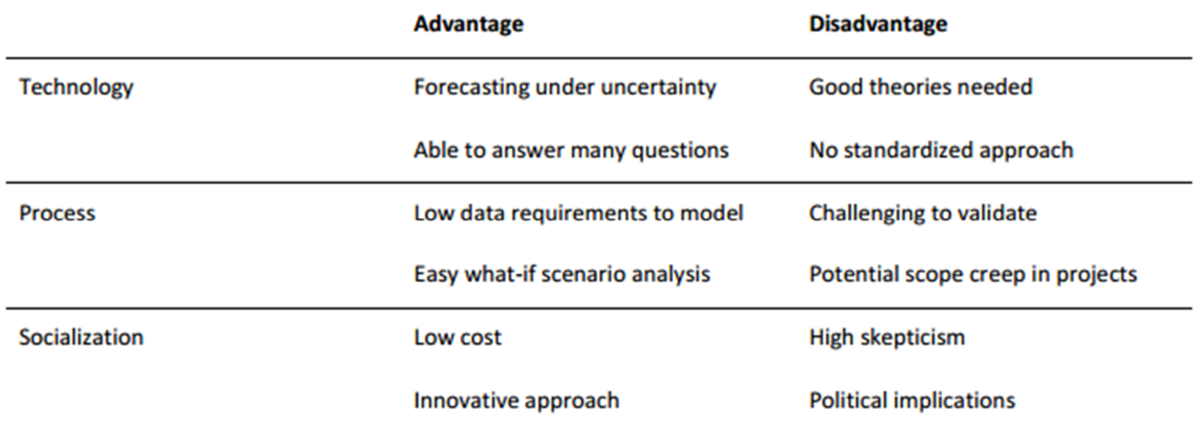
It has detailed and elaborate process for system specification, implementation, testing and debugging. It is also supported by Prometheus Design Tool that allows the user to design overview diagrams. Further it has examples which help to better understand what is required in each stage of development.

The methodology consist of three phases: System Specification, Architectural design and detailed design. An overview of the methodology, including its phases, deliverables and intermediate products is depicted below. Although the phases are described in sequential fashion it is acknowledged that Evolutionary prototyping methodology was used during the implementation phase.

3.4.1 System Implementation: Simulation

During the Detailed Design phase agent overview diagram was produced and the agent capabilities and interactions were also produced. Simulation methods of analysis, because of powerful and user-friendly software tools. Simulation method used especially in Computer Science because it offers the possibility to investigate systems or regimes that are outside of the experimental domain or the systems that is under invention or construction. Normally complex phenomena that cannot be implemented in laboratories evolution of the universe. Some domains that adopt computer simulation methodologies are sciences such as astronomy, physics or economics; other areas more specialized such as the study of non-linear systems, virtual reality or artificial life also exploit these methodologies. A lot of projects can use the simulation methods, like the study of a new developed network protocol. To test our model we have to build a huge network with a lot of expensive network tools, but this network can't be easily achieved. For this reason we can use the simulation method.

Summarized are the pro and cons of using simulation methodology;



	Advantage	**Disadvantage**
Technology	Forecasting under uncertainty	Good theories needed
	Able to answer many questions	No standardized approach
Process	Low data requirements to model	Challenging to validate
	Easy what-if scenario analysis	Potential scope creep in projects
Socialization	Low cost	High skepticism
	Innovative approach	Political implications

Figure 2: Summary of pro & cons of simulation

3.5 Simulation Phases

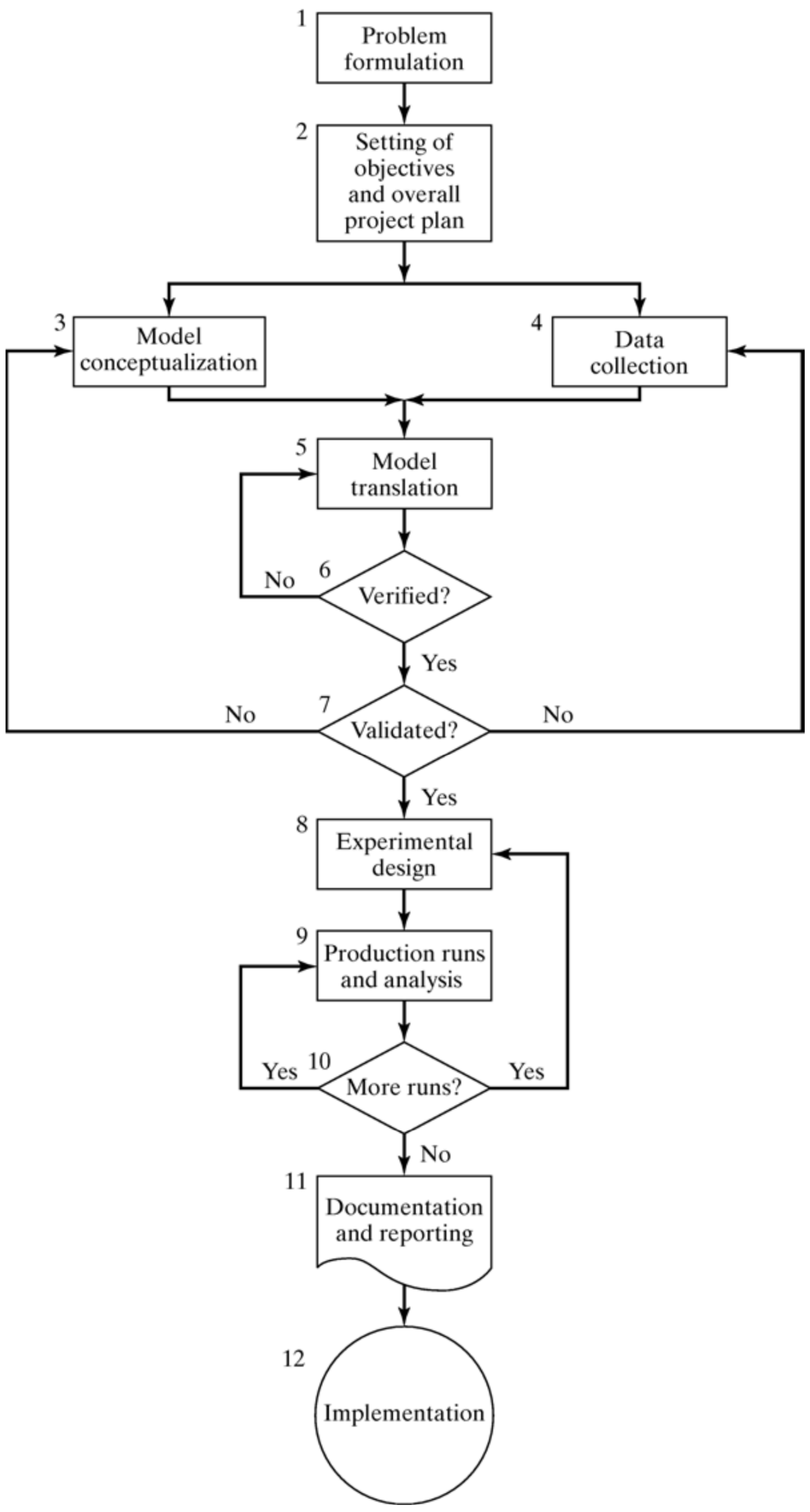


Figure 3: Simulation Phases

3.6 Simulation Environment

We will develop a simulation in Java using Netbeans and the discrete event simulation toolkit SimJava. In simulation environment we will use N agents.

The agents are of mainly two types- good and malicious. Good agents cooperate in providing both good service and honest feedback. In contrast, malicious agents are opportunistic in the sense that they cheat whenever it is advantageous for them. Malicious agents provide both ineffective service and false feedback.

4.3 Analysis of the different factors related to evaluating the trust

We have explained below a number of parameters used for computing the trust of an agent, but none of these models using these parameters can fully cope with the strategic adaptations made by malicious agents. The mathematical and logical definitions used for these parameters also cannot reflect the true scenarios faced in real life. Today MAS security is becoming a more complex problem which requires intensive and continuous research of solutions to ever emerging security lapses.

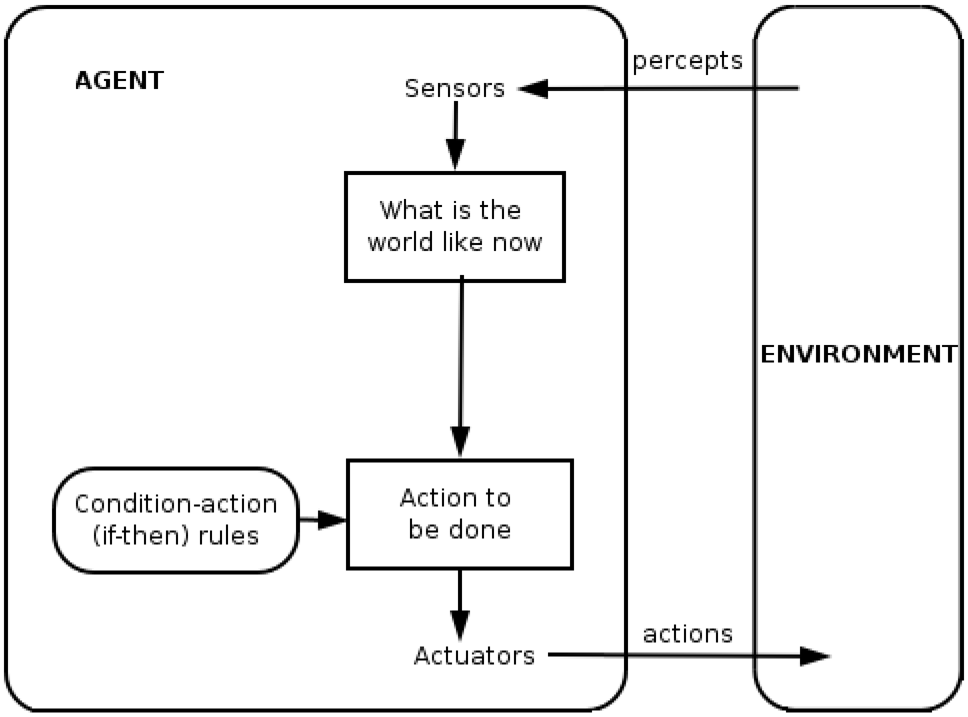


Figure 4: Agent Environment

4.3.1 Satisfaction

Satisfaction function measures the degree of satisfaction an agent has about a given service provider. In other words, it keeps record of the satisfaction level of all the transactions an agent makes with another agent.

4.3.2 Similarity

Similarity metric defines to what extent two agents are alike.

4.3.3 Feedback Credibility

Feedback credibility is used to measure the degree of accuracy of the feedback information that the recommending agent provides to the evaluator. Normally it is assumed that good agents always provide true feedback and malicious agents provide false feedback. However, this is not always the real scenario as good agents might provide false feedbacks to their competitors and malicious agents might occasionally provide true feedbacks to hide their real nature. So feedback credibility is needed to determine the reliability of the feedback. During trust evaluation, feedbacks provided by agents with higher credibility are trust worthier, and are therefore weighted more than those from agents with lower credibility

4.3.4 Direct Trust

Direct trust also known as local trust represents the portion of trust that an agent computes from its own experience about the target agent.

4.3.5 Indirect Trust

Indirect trust also referred as recommendation is computed from the experience of other agents. An agent utilizes the experience gained by other agents in the system to make effective transaction decisions especially when it has no or very little experience with the given target agent. To do so, an agent requests other agents to provide recommendation about the target agent. The evaluating agent then aggregates recommendation from other agents along with the feedback credibility of the recommenders.

4.3.6 Recent Trust

Recent trust reflects only the recent behaviors.

4.3.7 Historical Trust

Historical trust is built from past experience and it reflects long term behavioral pattern.

4.3.8 Expected Trust

Expected trust reflects expected performance of the target agent and it is deduced from both recent and historical trust.

4.3.9 Decay model

Due to the highly dynamic nature of agents, trust should attenuate with the elapse of time in absence of interaction. If an agent remains idle for a long time i.e., if it does not interact with the network for a long period, the evaluation of its trust should degrade gradually.

4.3.10 Deviation Reliability

Deviation reliability is a measure of how much deviation we are willing to tolerate. Malicious agents sometimes strategically oscillate between raising and milking their reputation which seriously affects the performance of the network. So, some form of measurement is required to handle such scenario. Deviation reliability handles such trust fluctuation.

4.3.11 Overall Trust Metric

This is the actual trust value used in prioritizing all agents. It is computed from expected trust and deviation reliability. Reputation based trust models are basically divided into two category based on the way information is aggregated from an evaluator’s perspective. They are ”Direct / Local experience model” and ”Indirect / Global reputation model ” where direct experience is derived from direct encounters or observations and indirect reputation is derived from inferences based on information gathered indirectly.

For the following sections we assume that agent p (called evaluator) needs to calculate the trustworthiness of agent q (called the target agent) to n transactions in the t-th time interval.

$$\textbf{Trust}\ {}^{t}_{n}(p, q) = \textbf{ET (Expected Trust)}\ {}^{t}_{n}(p, q) \times \textbf{DR (Deviation Reliablity)}\ {}^{t}_{n}(p, q)$$

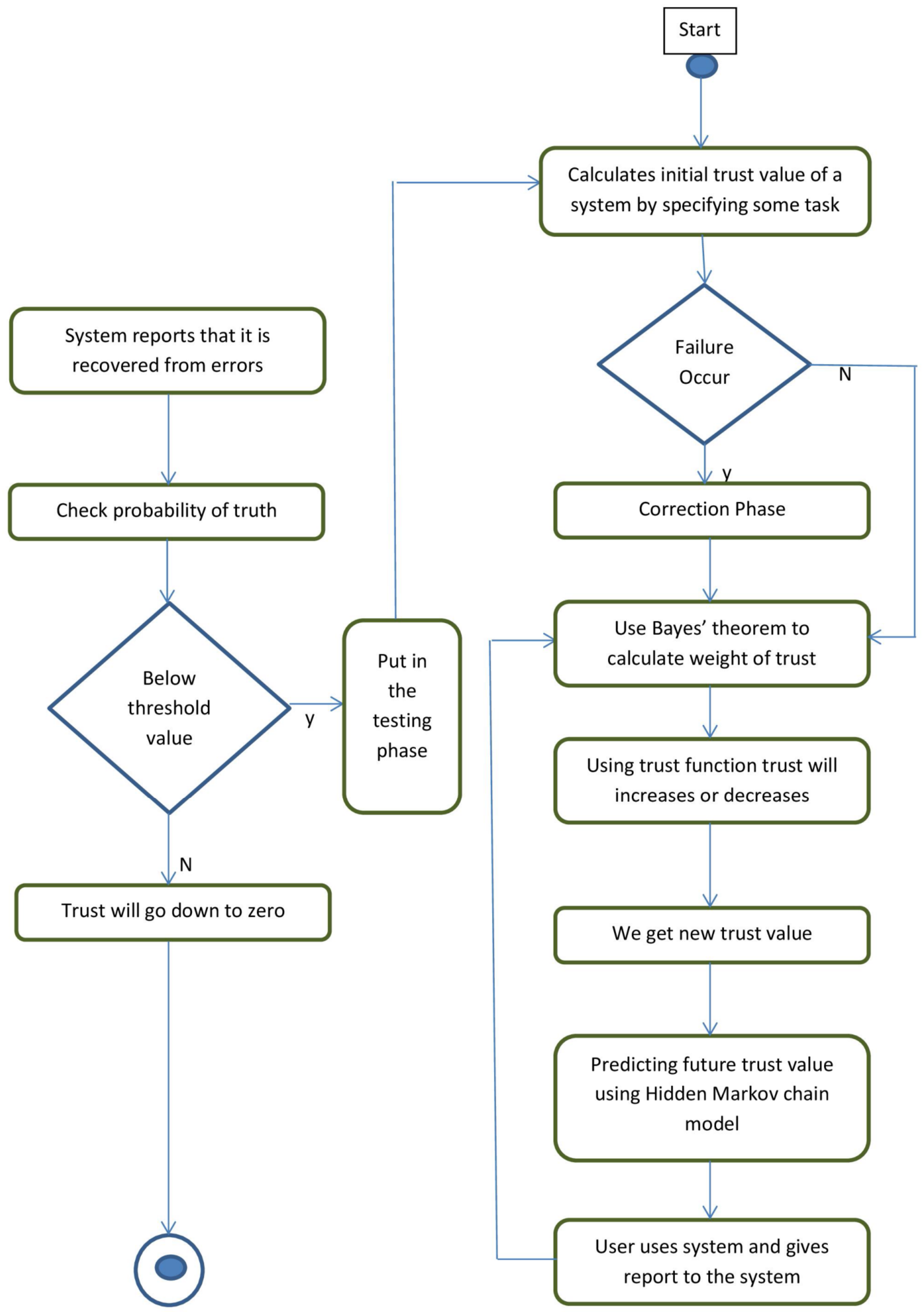

Start
Calculates initial trust value of a system by specifying some task
Failure Occur
N
y
Correction Phase
Use Bayes' theorem to calculate weight of trust
Using trust function trust will increases or decreases
We get new trust value
Predicting future trust value using Hidden Markov chain model
User uses system and gives report to the system
System reports that it is recovered from errors
Check probability of truth
Below threshold value
y
Put in the testing phase
N
Trust will go down to zero

Figure 4: Trust Computation Flow Chart. Source: Atul Saurabh and Swapnil Parikh

4.4 Our Security Model

4.4.1 Authentication

In order to be a reliable source of information, the responses to be used in trust evaluation must be authenticated. In our model, this is obtained by signing the hash of the responses provided. For protection against replay attacks, the signed hash covers, among other fields, the ID of the querying and responding peers, a query ID number, and the file hash being offered**.**

4.4.2 Key Management

Our Prototype makes use of digital signatures for authentication of critical messages. The core trust issue in public key systems is to ascertain that a public key received on-line indeed belongs to the claimed party. The classical solution to this problem is by trusted certification authorities, but this may not be an option in a P2P system due to its decentralized nature.

4.4.3 Denial of Service Protection

The requirement of responding to every relevant query with a digital signature is likely to be an excessive burden on the peers. Moreover, it can easily be exploited for denial of service attacks by attackers continually issuing many high match queries. In the initial response, the file hash is sent without any signature. Then the querying peer decides on which file versions he is genuinely interested in and solves the puzzles of a limited number of the respondents for each version.

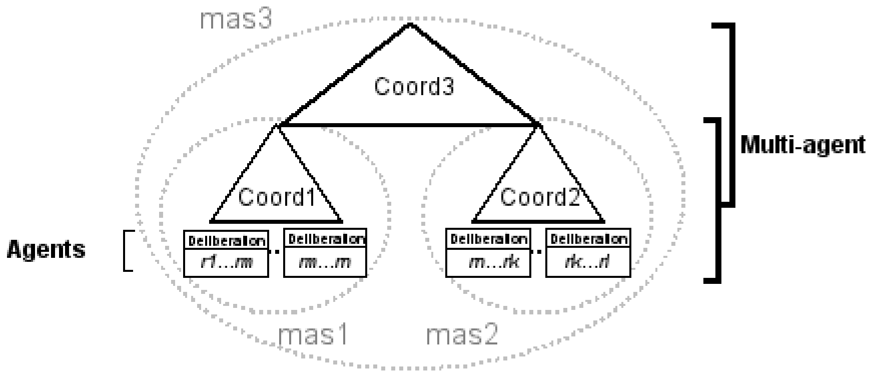


Figure 5: Agent Coordination

4.5 RSA Algorithm

We have used RSA algorithm for public-key cryptography that is based on the presumed difficulty of factoring large integers. It is the most widely used public key encryption algorithm. The basis of the security of the RSA algorithm is that it is mathematically infeasible to factor sufficiently large integers.

1. Choose two distinct prime numbers p and q.
   - For security purposes, the integers p and q should be chosen at random, and should be of similar bit-length. Prime integers can be efficiently found using a primality test.
2. Compute n = pq.
   - n is used as the modulus for both the public and private keys
3. Compute φ(n) = (p – 1)(q – 1), where φ is Euler's totient function.
   - Choose an integer e such that 1 < e < φ(n) and greatest common divisor of (e, φ(n)) = 1
   - e is released as the public key exponent.
   - e having a short
4. Determine d as:

$$d \equiv e^{-1} \pmod{\varphi(n)}$$

d is the multiplicative inverse of e mod φ(n).

- This is more clearly stated as solve for d given (de) = 1 mod φ(n)
- This is often computed using the extended Euclidean algorithm.
- d is kept as the private key exponent.

5 CHAPTER FIVE: MODEL IMPLEMENTATION CONCEPTUAL MODELING

In this chapter we present a java application prototype that model an approach to security problems in multi agent systems based on distributed trust and the delegation of permissions, and credibility.

5.1 System requirements

Software Requirements

• OS	:	Windows 8.1
• Language	:	Java
• IDE	:	NetBeans 6.9.1

Hardware Requirements

• System	:	Pentium IV2.4GHz.
• Hard Disk	:	250 GB.
• Monitor	:	15 VGA Color
• Ram	:	1GB.

5.2 Modules

- ➢ Server Configuration
- ➢ Client Registration
- ➢ Key Pair Generation
- ➢ Agent Trust Evolution
- ➢ Client Agent Validation
- ➢ File Transfer

5.3 Process Flow Diagram

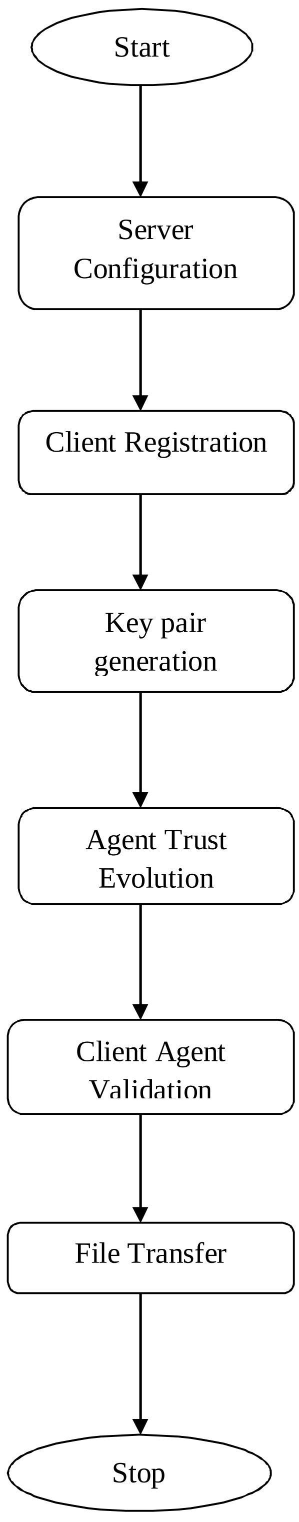


Figure 5: Process Flow, Source Author

Module Description

1. Server Configuration

A server is a physical computer dedicated to running one or more services to serve the needs of the users of other computers on the network. Depending on the computing service that it offers it could be a database server, file server, mail server, web server, or some other kind of server. In the context of client-server architecture, a server is a computer program running to serve the requests of other programs, the "clients". Thus, the "server" performs some computational task on behalf of "clients". The clients either run on the same computer or connect through the network. Every agent has different needs, and the different servers all represent different sets of trust.

2. Client Registration

Different agents have different requirements for registered agents. Typically, the agent must be a legal resident of the state that allows entities to serve as registered agents. Client Application Services is the name of the client and services framework. In order to support large sites with many clients provides a batch client registration feature that allows you to define multiple clients by way of a single “clients definition” file. In order to import multiple clients, you must:

1. Key Pair Generation

The Key Pair Generator class is used to generate pairs of agents. All key pair generators share the concepts of a key size and a source of randomness. There is an - method in this Key Pair Generator class that takes these two universally shared types of arguments. There is also one that takes just a key size argument, and uses the Secure Random implementation of the highest-priority installed provider as the source of randomness.

4. Agent Trust Evolution

Agents generally interact by making commitments to one another to carry out particular tasks. In most realistic environments there is no guarantee that a contracted agent will actually enact its commitments. It could interact with all agents and then derive trust measures from the history of interactions. Current mechanisms for evaluating the trustworthiness of an agent within an electronic marketplace depend either on using a history of interactions or one commendations from other agents. The calculation and measurement of trust in unsupervised virtual communities like multi agent environments involves complex aspects such as credibility rating for opinions delivered by peer agents.

5. Client Agent Validation

The agents in the Multi-Agent System are able to gather data by generation of logs as well as provide run-time validation and verification support by watch agents and also agents to check any violation of invariants at run-time. The validation is an essential parts of the model development process if agent to be accepted and used to support decision making. Validation is an essential part to increase the confidence of agent making it more valuable.

The service agent expects all the credentials necessarily at the time of request. In order to use its services, a requesting agent must send all required credentials along with the request for service. The service agent will check its knowledge base, and question other agents about their beliefs in order to verify the credentials.

6. File Transfer

File transfer is a generic term for the act of transmitting files over a computer network like the Internet. There are numerous ways and protocols to transfer files over a network. File transfer is the movement of one or more files from one location to another. The File Transfer is a common way to transfer a single file or a relatively small number of files from client to sever.

6. CHAPTER SIX: EXPERIMENTATION, RESULTS & DISCUSSION

6.1 Agent’s Behavioral Pattern

The behavioral pattern of good agents is quite easy to simulate as they provide good service and honest feedback. However, it is challenging to simulate an agent’s malicious behavior realistically. We mainly study three behavioral patterns namely- non-collusive, collusive and strategically altering.

In **non-collusive** setting malicious agents cheat during transaction and give false feedback to other agents i.e., they rate good agents poorly while rating malicious agents highly.

The **collusive** setting is similar to the non-collusive setting with one additional feature that malicious agents form a collusive group and deterministically help each other by performing numerous fake transactions to boost their own rating while disparaging other good agents.

In the **strategically altering** setting a malicious agent may occasionally decide to cooperate in order to confuse the system. In this case, other agents are commonly fooled into thinking that the malicious agent is actually a good agent.

6.2 Transaction Setting

There are three types of transaction setting simulated namely, random setting, trust prioritized setting and load balanced setting. In the random setting, agents randomly interact with each other. In the trust prioritized setting an agent first initiates a transaction request. Against each request certain percentage of agents respond. The response percentage is controlled by response rate parameter. The initiating agent then sorts the responders based on their trust value and selects the agent with the highest trust value to perform the desired transaction. Finally, in the load balancing scheme a service provider with least amount of workload is selected.

6.3 Performance evaluation index

To compare the performance of our model with other existing trust models we use an evaluation index named, successful transaction rate (STR). STR is described as the ratio of the number of successful transactions to the total number of transactions.

6.4 Experiment Simulation

In our experiment we have executed 10 runs as below;

1. Message sharing among clients and agents on the localhost computer
2. Clients and agents connected over the internet.

A table below was used for recording the results over local host setting;

Number of Clients	Number of Agents	Secure	STR(%)
5	5	Yes	100
3	6	No	100
1	10	Yes	100
5	7	Yes	100
1	10	No	100

Table 1: STR over localhost

A table below was used for recording the results over internet setting;

Number of Clients	Number of Agents	Secure	STR(%)
5	5	Yes	100
3	6	No	78
1	10	Yes	92
5	7	No	85
1	10	Yes	98

Table 2: STR over Internet

Secure STR Average on Local = 300/3= 100%

Non-Secure STR Average on Local **=** 300/3= 100%

Secure STR Average on Internet = 290/3 = 96.6667%

Non-Secure STR Average on Internet = 163/2 = 81.5

From Above result it is clear that messages among agents over internet which are secured have the highest probability of being delivered to the intended destinations. Likewise messages among agents on the local environment also have high probability of being delivered to destination successfully due to the fact of no malicious threat. `

6.5 Experiment Testing

We used Security Penetration Testing open software **nexpose** to test for vulnerabilities. Developed by Rapid7 and used by every pentester and ethical hacker in the world. The Nexpose Project is a security project which delivers information about security vulnerabilities and helps penetration testing and Intrusion detection. The open source project – known as the Metasploit Framework, is used by security professionals to execute exploit code against a remote target machine – for penetration testing of course!

6.6 Sniffing

On the penetration testing sniffing was used. Sniffing involves capturing, decoding, inspecting and interpreting the information inside a network packet on a TCP/IP network. The purpose is to steal information, usually user IDs, passwords, network details, credit card numbers, etc. Sniffing is generally referred to as a “passive” type of attack, wherein the attackers can be silent/invisible on the network. This makes it difficult to detect, and hence it is a dangerous type of attack.

The TCP/IP packet contains vital information required for two network interfaces to communicate with each other. It contains fields such as source and destination IP addresses, ports, sequence numbers and the protocol type. Each of these fields is crucial for various network layers to function, and especially for the Layer 7 application that makes use of the received data. By its very nature, the TCP/IP protocol is only meant for ensuring that a packet is constructed, mounted on an Ethernet packet frame, and reliably delivered from the sender to the receiver across networks. However, it does not by default have mechanisms to ensure data security. Thus, it becomes the responsibility of the upper network layers to ensure that information in the packet is not tampered with. To understand why hackers sniff, we need to know what they can get from the network. Below shows the OSI layers and the information a hacker can steal at each layer.

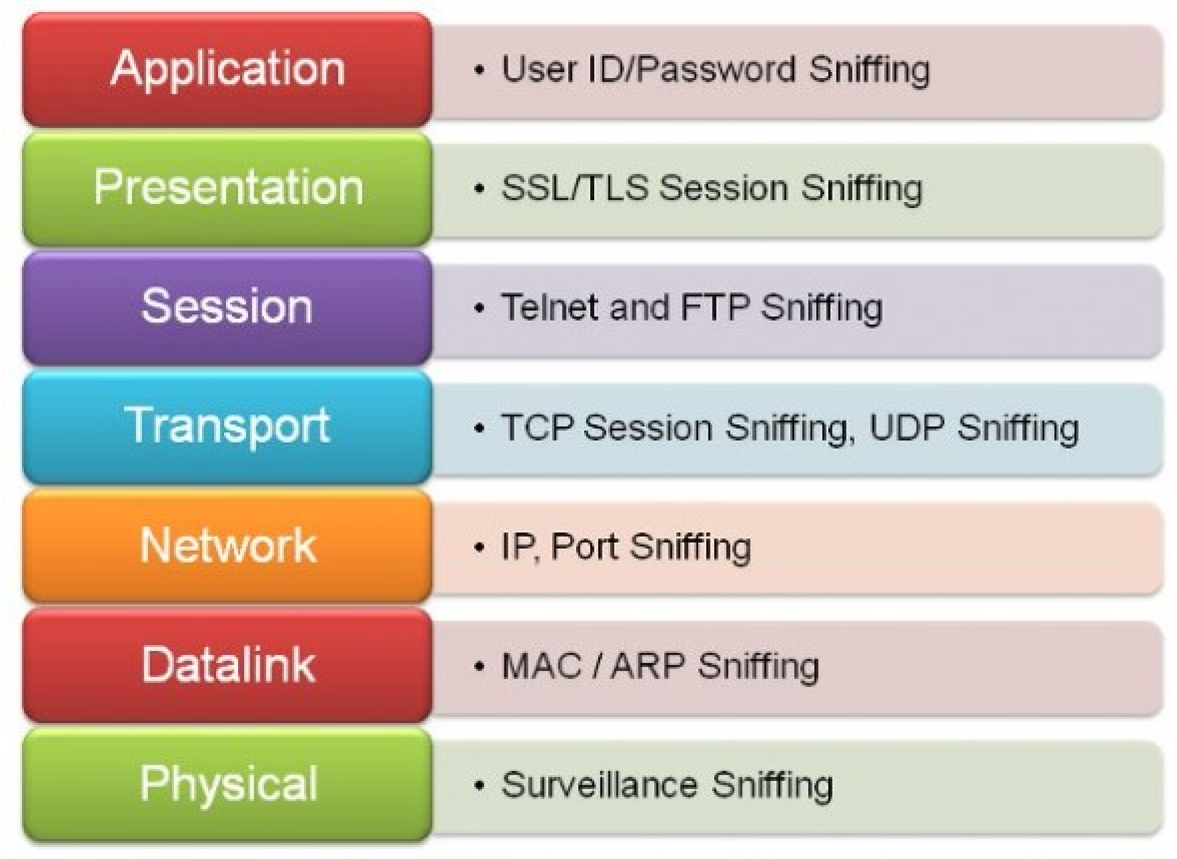


Figure 6: OSI Layers

The sniffing process is used by hackers either to get information directly or to map the technical details of the network in order to create a further attack. Hackers are always in favor of sniffing, because it can be done for a longer time without getting caught.

6.6.1 How is sniffing done?

Network sniffing uses sniffer software, either open source or commercial. Broadly, there are three ways to sniff a network, as shown below.

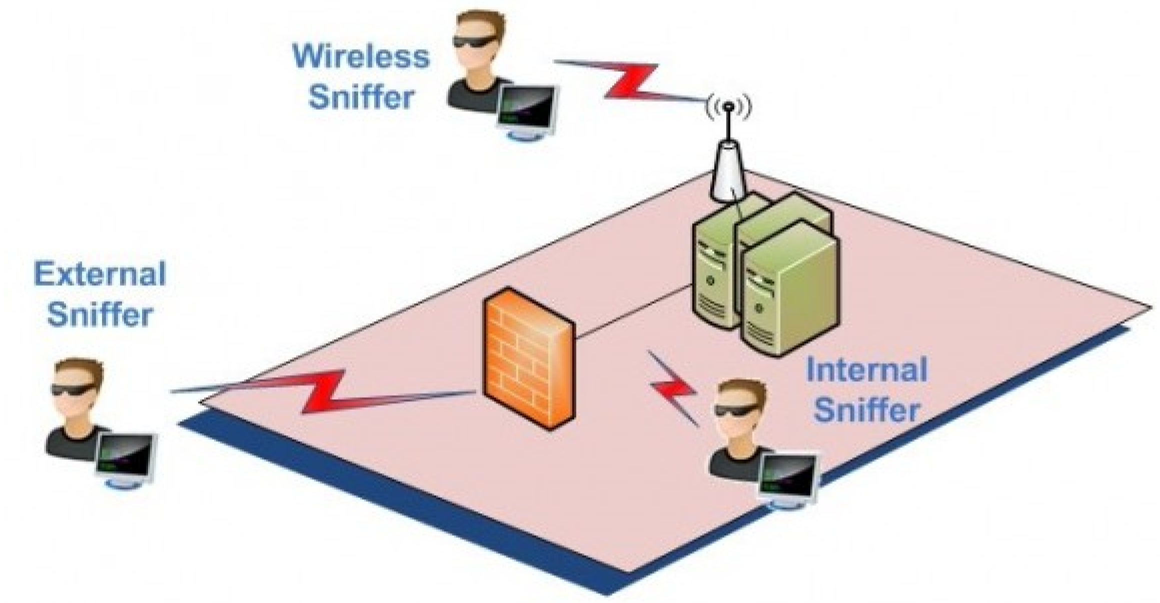
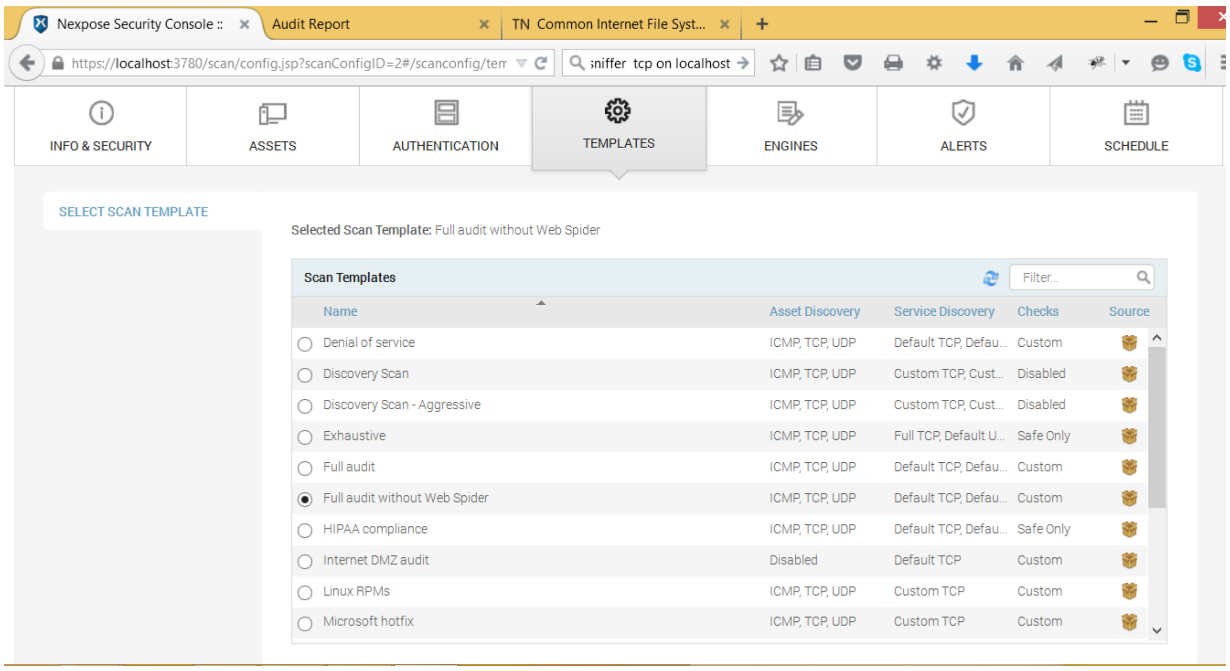


Figure 7: Sniffing

6.7 Experiment Results

Out of the 10 runs the data was successfully shared with intended agents in different clients as intended. No vulnerability was recorded on the secured message. On our vulnerability scan, we have used the TCP template for conclusive test.



Selected Scan Template: Full audit without Web Spider

Scan Templates

	Name	Asset Discovery	Service Discovery	Checks	Source
○	Denial of service	ICMP, TCP, UDP	Default TCP, Defau...	Custom	
○	Discovery Scan	ICMP, TCP, UDP	Custom TCP, Cust...	Disabled	
○	Discovery Scan - Aggressive	ICMP, TCP, UDP	Custom TCP, Cust...	Disabled	
○	Exhaustive	ICMP, TCP, UDP	Full TCP, Default U...	Safe Only	
○	Full audit	ICMP, TCP, UDP	Default TCP, Defau...	Custom	
●	Full audit without Web Spider	ICMP, TCP, UDP	Default TCP, Defau...	Custom	
○	HIPAA compliance	ICMP, TCP, UDP	Default TCP, Defau...	Safe Only	
○	Internet DMZ audit	Disabled	Default TCP	Custom	
○	Linux RPMs	ICMP, TCP, UDP	Custom TCP	Custom	
○	Microsoft hotfix	ICMP, TCP, UDP	Custom TCP	Custom	

6.8 Discovered Vulnerabilities

Discovered vulnerabilities when client 1 port was opened for sending message.

Device	**Protocol**	**Port**	**Vulnerabilities**	**Additional Information**
10.2.6.120 – Client 1	tcp	139	1	• Windows 8.1 Pro 6.3
10.2.6.120 – Client 2	tcp	445	0	• Windows 8.1 Pro 6.3

Table 3: Discovered Vulnerabilities

Our approach to the security problems in multi agent systems based on distributed trust and the delegation of permissions, and credibility. On this setting the biggest problem is the authentication and reliable identification of agents' true identity. The key pair and Certification Authority are deployed to encrypt/decrypt electronic data or transaction, or sign/authenticate the sender and the recipient. Our approach has proved to be secure even on highly malicious agents infected network like internet.

From the below table extracted from Nexpose vulnerability testing tool, it is clear that the tool can detect unsecure/vulnerable messages with less effort

packets	**File Name**	**Message type**	**Additional Information**
27	test.txt	Non Encrypted	10 bytes… src 10.2.6.102.139 dest 10.2.6.102.139………………………….. ……………. …………..

Table 4: Non Secure Message

Encrypting a message using our model, Nexpose cannot detect any vulnerabilities.

packets	**File Name**	**Message type**	**Additional Information**
27	test.txt	Encrypted	No info

Table 5: Encrypted Message

Using extracts from the Nexpose Vulnerability Audit report we can conclude that our model is more robust in tackling the security problem faced by muilt agent systems in the dynamic and open environment.

7. CHAPTER SEVEN: CONCLUSION & FUTURE WORKS

In this report we have described a generic trust computation model for multi-agent systems. Our model can be tuned to the meet the requirements of a specific application. For example our model can be used in electronic markets and e-commerce environments (like Amazon Auctions, eBay, OnSale Exchange) where buyers rate sellers regarding their purchase after they make a transaction. Sellers might attempt to raise their trust value (hence reputation) by creating fake buyers and fake transactions. So, in such scenarios our model can be applied to filter out potential bad sellers.

Like any other reputation model our model assists agents to choose reputed agents while avoiding untrustworthy ones. However, reputation-based trust mechanism also introduces

Another challenging threat that most trust models fail to handle is the dynamic personality of malicious agents. By cleverly alternating between good and malicious nature they try to remain undetected while causing damage.

If a malicious agent can easily switch its identity then the trust system may suffer as malicious agents can easily dispatch their bad history. The defense against such attacks should not depend on the trust model but rather on the authentication and access control system.

7.1 Suggested further work

A pilot can be conducted to allow extensive tests of the model. This will help in evaluating the model in terms of data transfer speeds, stability, network load and robustness.

8. References

Abdul-Rahman, S. Hailes, “Supporting trust in virtual communities”, In Proceedings of the Hawai’i International Conference on System Sciences, Maui, Hawaii, Jan 4-7 2000.

Evan Perez 2014, “Hospital network hacked, 4.5 million records stolen”, CNN Money (New York) August 18. Available from:<http://money.cnn.com/2014/08/18/>.[18 September 2014].

Hu. J, Wu. Q, and Zhou. B, “FC Trust: A robust and efficient feedback credibility-based distributed P2P trust model,” in Proceedings of IEEE 9th International Conference for Young Computer Scientists (ICYCS), 2008, pp. 1963–1968.

Jessica Silver-Greenberg, Matthew Goldstein and Nicole Perlroth 2014, “JPMorgan Chase Hacking Affects 76 Million Households”, The New York Times 16 February. Available from: <http://dealbook.nytimes.com/2014/10/02>. [2 October 2014].

Jordi Sabater and Carles Sierra, “Social regret, a reputation model based on social relations,” ACM SIGecom Exchanges - Chains of commitment, vol. 3, pp. 44–56, December 2001.

Kamvar S. D, Schlosser M. T, and Garcia H, “The EigenTrust algorithm for reputation management in P2P networks,” in Proceedings of the 12th ACM international World Wide Web conference (WWW), 2003, pp. 640–651.

Kitano, H, Kuniyoshi, Y, Noda, I, Asada, M, Matsubara, H and Osawa, E, 1997, “Robocup: a challenge problem for AI” AI Magazine, 18 73–85.

Nance, R.E. “A History of Discrete Event Simulation Programming Languages,” In: proceedings of the Second ACM SIGPLAN History of Programming Languages Conference, Cambridge, MA, April 20-23, Reprinted in ACM SIGPLAN Notices, 28(3), pp.149- 175.1993

N. R. Jennings, T. D. Huynh, and N. R. Shadbolt, “FIRE: An integrated trust and reputation model for open multi-agent systems,” in Proceedings of the 16th European Conference on Artificial Intelligence (ECAI), 2004, pp. 18–22.

Sarvapali D. Ramchurn, Dong Huynh, and Nicholas R. Jennings. Trust in multi-AgentSystems. Knowl. Eng. Rev., 19(1):1–25, 2004.

Steinmetz. R and Wehrle. K, Peer-to-Peer Systems and Applications. Springer-Verlag New York, Inc., 2005.

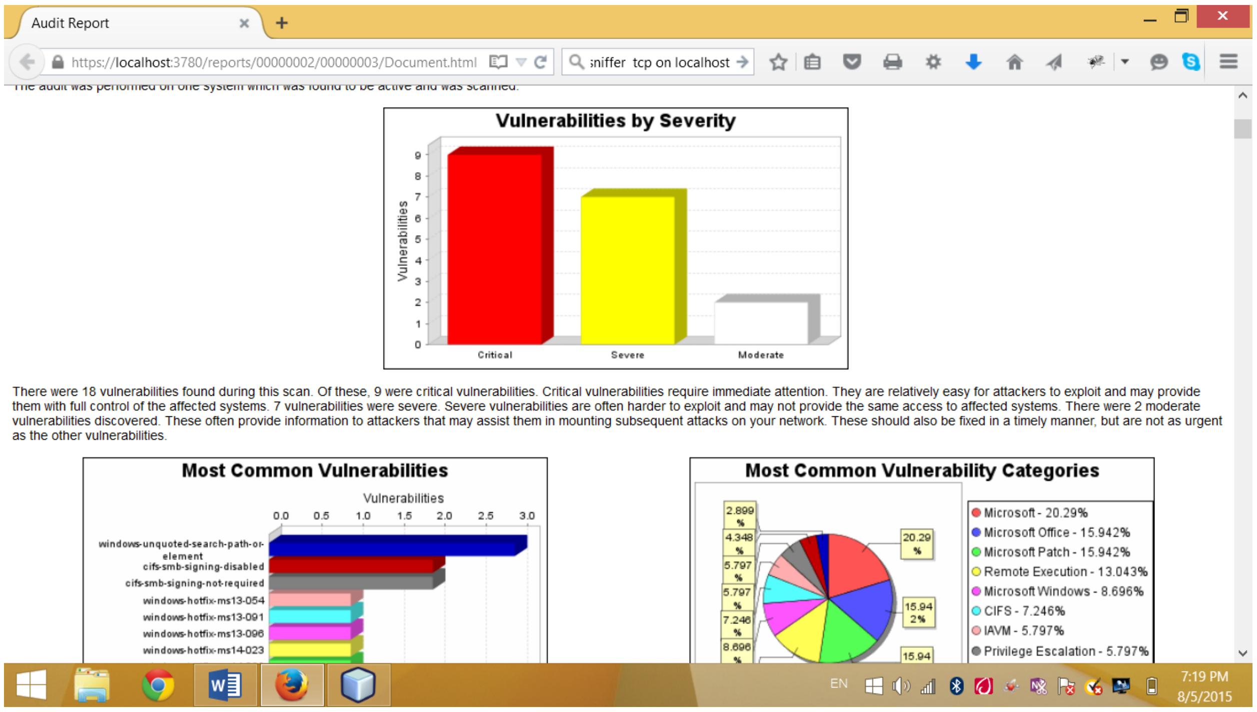
Wang. Y and Vassileva. J, “Bayesian network-based trust model,”in *Proceedings of IEEE/WIC International Conference on Web Intelligence (WI)*, Halifax, Canada, October 2003, pp. 372–378.

Wen. D, Huaimin.W , Yan. J, and Peng. Z, “A recommendationbased peer-to-peer trust model,” *Journal of Software*, vol. 15, no. 4, pp. 571–583, 2004.

Xiong.L and L. Li, “Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities,” *IEEE Transactions onKnowledge and Data Engineering*, vol. 16, no. 7, pp. 843–857, 2004.

Zhang. Y, Chen. S, and Yang. G, “SFTrust: A double trust metric based trust model in unstructured P2P systems,” in *Proceedings of IEEE International Symposium on Parallel and Distributed Process-ing (ISPDP)*, 2009, pp. 1–7.

9. APPEDIX A: Screen Shot of Vulnerability Audit Report

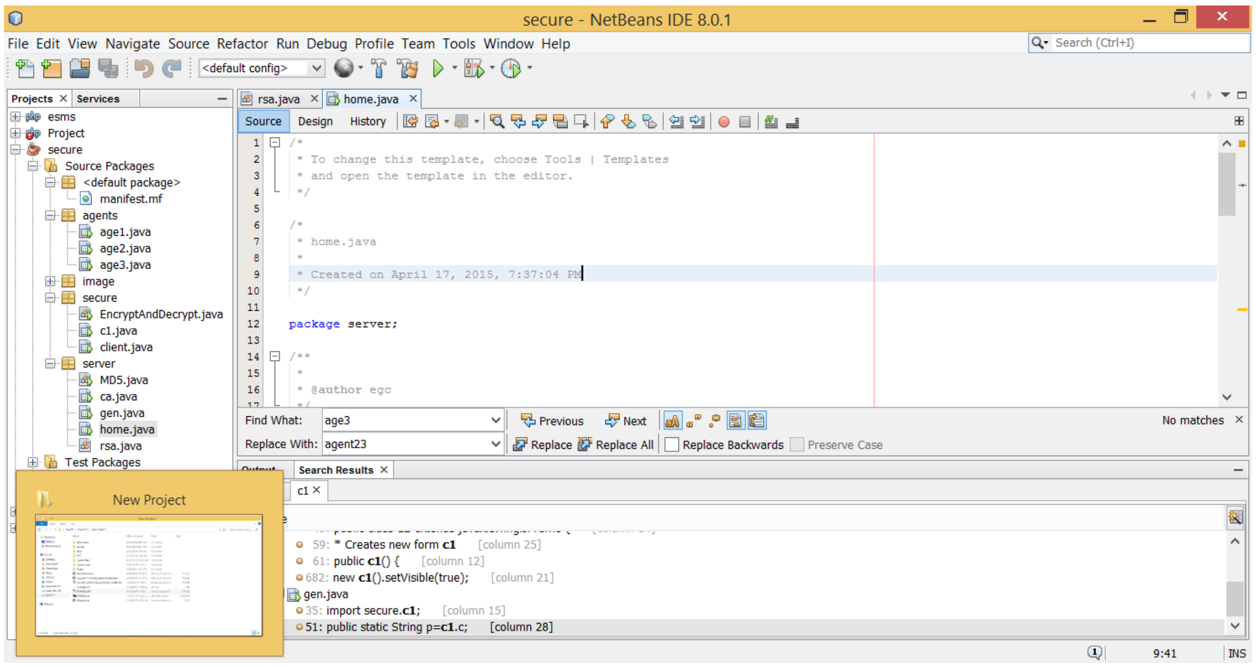


Audit Report
https://localhost:3780/reports/00000002/00000003/Document.html
sniffer tcp on localhost
Vulnerabilities by Severity
Vulnerabilities
Critical
Severe
Moderate
Most Common Vulnerabilities
windows-unquoted-search-path-or-element
cifs-smb-signing-disabled
cifs-smb-signing-not-required
windows-hotfix-ms13-054
windows-hotfix-ms13-091
windows-hotfix-ms13-096
windows-hotfix-ms14-023
Most Common Vulnerability Categories
Microsoft - 20.29%
Microsoft Office - 15.942%
Microsoft Patch - 15.942%
Remote Execution - 13.043%
Microsoft Windows - 8.696%
CIFS - 7.246%
IAVM - 5.797%
Privilege Escalation - 5.797%
7:19 PM
8/5/2015

The audit was performed on one system which was found to be active and was scanned.

There were 18 vulnerabilities found during this scan. Of these, 9 were critical vulnerabilities. Critical vulnerabilities require immediate attention. They are relatively easy for attackers to exploit and may provide them with full control of the affected systems. 7 vulnerabilities were severe. Severe vulnerabilities are often harder to exploit and may not provide the same access to affected systems. There were 2 moderate vulnerabilities discovered. These often provide information to attackers that may assist them in mounting subsequent attacks on your network. These should also be fixed in a timely manner, but are not as urgent as the other vulnerabilities.

10. APPENDIX B: SAMPLE CODE



```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/*
 * home.java
 *
 * Created on April 17, 2015, 7:37:04 PM
 */

package server;

/**
 *
 * @author egc
```

Find What: age3 | Replace With: agent23 | No matches

```
59: * Creates new form c1      [column 25]
61: public c1() {      [column 12]
682: new c1().setVisible(true);      [column 21]
gen.java
35: import secure.c1;      [column 15]
51: public static String p=c1.c;      [column 28]
```

Agent Code

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/*
 * age2.java
 *
 * Created on June 28, 2015, 9:26:31 AM
 */

package agents;
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.PrintStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Random;

import javax.crypto.Cipher;

import javax.crypto.SecretKey;

import javax.crypto.SecretKeyFactory;

import javax.crypto.spec.DESedeKeySpec;

import javax.crypto.spec.PBEKeySpec;

import javax.crypto.spec.PBEParameterSpec;

import javax.swing.JOptionPane;

/**

 *

 * @author Admin

 */

public class age2 extends javax.swing.JFrame {

public static String original=" ";

 File f;

    private static FileInputStream inFile;

    private static FileOutputStream outFile;

    private static String filename;

     public static File d=server.gen.f;

      public static String stat="null";

    /** Creates new form age2 */

    public age2() {

        initComponents();

    }


    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();
        jTabbedPane1 = new javax.swing.JTabbedPane();
        jPanel2 = new javax.swing.JPanel();
        jLabel2 = new javax.swing.JLabel();
        jTextField1 = new javax.swing.JTextField();
        jLabel3 = new javax.swing.JLabel();
        jTextField2 = new javax.swing.JTextField();
        jButton1 = new javax.swing.JButton();
        jPanel3 = new javax.swing.JPanel();
        jTextField3 = new javax.swing.JTextField();
        jButton2 = new javax.swing.JButton();
        jPanel4 = new javax.swing.JPanel();
        jScrollPane1 = new javax.swing.JScrollPane();
        jTextArea1 = new javax.swing.JTextArea();
        jButton3 = new javax.swing.JButton();
        jButton4 = new javax.swing.JButton();
        jScrollPane2 = new javax.swing.JScrollPane();
        jTextArea2 = new javax.swing.JTextArea();
```

```
        jButton5 = new javax.swing.JButton();
        jLabel4 = new javax.swing.JLabel();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setMinimumSize(new java.awt.Dimension(778, 516));
        getContentPane().setLayout(null);

        jLabel1.setFont(new java.awt.Font("Times New Roman", 3, 36));
        jLabel1.setText("Agent 2");
        getContentPane().add(jLabel1);
        jLabel1.setBounds(330, 10, 115, 42);

        jTabbedPane1.setBorder(javax.swing.BorderFactory.createMatteBorder(1, 1, 1, 1, new
java.awt.Color(255, 0, 0)));

        jLabel2.setText("Authenticated code");

        jTextField1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jTextField1ActionPerformed(evt);
            }
        });

        jLabel3.setText("Client Name");

        jButton1.setText("Key Authorized");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
```

```
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton1ActionPerformed(evt);
        }
    });

    javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
    jPanel2.setLayout(jPanel2Layout);
    jPanel2Layout.setHorizontalGroup(
        jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel2Layout.createSequentialGroup()
            .addGap(88, 88, 88)
            .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, 
false)
                .addComponent(jLabel3,                    javax.swing.GroupLayout.DEFAULT_SIZE, 
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(jLabel2, javax.swing.GroupLayout.DEFAULT_SIZE, 140, Short.MAX_VALUE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                .addGroup(jPanel2Layout.createSequentialGroup()
                    .addGap(124, 124, 124)
                    .addComponent(jTextField2,        javax.swing.GroupLayout.DEFAULT_SIZE,        136, 
Short.MAX_VALUE))
                .addComponent(jTextField1,        javax.swing.GroupLayout.PREFERRED_SIZE,        136, 
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(99, 99, 99))
        .addGroup(jPanel2Layout.createSequentialGroup()
            .addGap(194, 194, 194)
            .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 161, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addContainerGap(232, Short.MAX_VALUE))
    );
    jPanel2Layout.setVerticalGroup(
        jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel2Layout.createSequentialGroup()
            .addGap(81, 81, 81)
            .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 38, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jTextField1, javax.swing.GroupLayout.DEFAULT_SIZE, 36, Short.MAX_VALUE))
            .addGap(69, 69, 69)
            .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE, 28, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jTextField2, javax.swing.GroupLayout.DEFAULT_SIZE, 42, Short.MAX_VALUE))
            .addGap(40, 40, 40)
            .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 52, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(73, 73, 73))
    );


    jTabbedPane1.addTab("Code", jPanel2);


    jTextField3.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jTextField3ActionPerformed(evt);
        }
    });

    jButton2.setText("Get_Key");
    jButton2.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton2ActionPerformed(evt);
        }
    });

    javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3);
    jPanel3.setLayout(jPanel3Layout);
    jPanel3Layout.setHorizontalGroup(
        jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel3Layout.createSequentialGroup()
            .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(jPanel3Layout.createSequentialGroup()
                    .addGap(217, 217, 217)
                    .addComponent(jButton2,              javax.swing.GroupLayout.PREFERRED_SIZE,              138,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGroup(jPanel3Layout.createSequentialGroup()
                    .addGap(88, 88, 88)
                    .addComponent(jTextField3,           javax.swing.GroupLayout.PREFERRED_SIZE,            414,
javax.swing.GroupLayout.PREFERRED_SIZE)))
            .addContainerGap(85, Short.MAX_VALUE))
    );
```

```
        jPanel3Layout.setVerticalGroup(
            jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel3Layout.createSequentialGroup()
                .addGap(128, 128, 128)
                .addComponent(jTextField3,                javax.swing.GroupLayout.PREFERRED_SIZE,                45,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(76, 76, 76)
                .addComponent(jButton2,                javax.swing.GroupLayout.PREFERRED_SIZE,                39,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(75, Short.MAX_VALUE))
        );

        jTabbedPane1.addTab("Authorzied_key", jPanel3);


        jTextArea1.setColumns(20);
        jTextArea1.setRows(5);
        jTextArea1.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                jTextArea1MouseClicked(evt);
            }
        });
        jScrollPane1.setViewportView(jTextArea1);

        jButton3.setText("Encrypt");
        jButton3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton3ActionPerformed(evt);
```

```
    }
});

jButton4.setText("Send");
jButton4.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton4ActionPerformed(evt);
    }
});

jTextArea2.setColumns(20);
jTextArea2.setRows(5);
jScrollPane2.setViewportView(jTextArea2);

jButton5.setText("Re-Encrypt");
jButton5.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton5ActionPerformed(evt);
    }
});

javax.swing.GroupLayout jPanel4Layout = new javax.swing.GroupLayout(jPanel4);
jPanel4.setLayout(jPanel4Layout);
jPanel4Layout.setHorizontalGroup(
    jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel4Layout.createSequentialGroup()
            .addGap(42, 42, 42)
            .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 158, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(54, 54, 54)
            .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED_SIZE, 149, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(40, 40, 40)
            .addComponent(jButton4, javax.swing.GroupLayout.PREFERRED_SIZE, 103, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(41, 41, 41))
          .addGroup(jPanel4Layout.createSequentialGroup()
            .addGap(74, 74, 74)
            .addComponent(jButton3, javax.swing.GroupLayout.DEFAULT_SIZE, 86, Short.MAX_VALUE)
            .addGap(118, 118, 118)
            .addComponent(jButton5, javax.swing.GroupLayout.PREFERRED_SIZE, 107, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(202, 202, 202))
        );
        jPanel4Layout.setVerticalGroup(
          jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
          .addGroup(jPanel4Layout.createSequentialGroup()
            .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
              .addGroup(jPanel4Layout.createSequentialGroup()
                .addGap(142, 142, 142)
                .addComponent(jButton4, javax.swing.GroupLayout.PREFERRED_SIZE, 41, javax.swing.GroupLayout.PREFERRED_SIZE))
              .addGroup(jPanel4Layout.createSequentialGroup()
                .addGap(23, 23, 23)
```

```
.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
                    .addComponent(jScrollPane2, javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jScrollPane1, javax.swing.GroupLayout.Alignment.LEADING, 
javax.swing.GroupLayout.DEFAULT_SIZE, 251, Short.MAX_VALUE))
                .addGap(18, 18, 18)

.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                    .addComponent(jButton5, javax.swing.GroupLayout.DEFAULT_SIZE, 
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(jButton3, javax.swing.GroupLayout.DEFAULT_SIZE, 39, 
Short.MAX_VALUE))))
            .addContainerGap(32, Short.MAX_VALUE))
        );

        jTabbedPane1.addTab("Data", jPanel4);

        getContentPane().add(jTabbedPane1);
        jTabbedPane1.setBounds(90, 60, 594, 390);

        jLabel4.setIcon(new javax.swing.ImageIcon(getClass().getResource("/image/bl.jpg"))); // NOI18N
        getContentPane().add(jLabel4);
        jLabel4.setBounds(0, 0, 780, 516);

        pack();
    }// </editor-fold>

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
```

```
    try
    {
        String m;
        Socket s1=new Socket("localhost", 168);
        BufferedReader br = new BufferedReader(new InputStreamReader(s1.getInputStream()));
        m =br.readLine();
        s1.close();
        s1=null;
        jTextField3.setText(m);

    }
    catch(Exception e)
    {
        System.out.println(e);
    }
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
     try
    {
        String s=jTextField1.getText();
        String s1=jTextField2.getText();
        if(s1.equals("client1"))
        {
        ServerSocket so1=new ServerSocket(145);
        Socket ss=so1.accept();
        PrintStream ps=new PrintStream(ss.getOutputStream());
        ps.println(s);
        ps.println(s1);
        ss.close();
        so1.close();
        ss=null;
        so1=null;
        ps=null;
        }
        if(s1.equals("client2"))
        {
        ServerSocket so1=new ServerSocket(146);
        Socket ss=so1.accept();
        PrintStream ps=new PrintStream(ss.getOutputStream());
        ps.println(s);
        ps.println(s1);
        ss.close();
        so1.close();
        ss=null;
        so1=null;
        ps=null;
        }
         if(s1.equals("client1") || s1.equals("client2"))
    {


    }
```

```
else

{



        Thread.sleep(1000);

        JOptionPane.showMessageDialog(this, "no");

}

    }

    catch(Exception e)

    {

      System.out.println(e);

    }

  }


  private void jTextField3ActionPerformed(java.awt.event.ActionEvent evt) {

    // TODO add your handling code here:

  }


  private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {

    try {

      filename ="./data.txt";

      String password = "super_secret";

      inFile = new FileInputStream(filename);

      FileWriter fw = new FileWriter("./enc.txt");

      outFile = new FileOutputStream("./enc.txt");

      PBEKeySpec keySpec = new PBEKeySpec(password.toCharArray());

      SecretKeyFactory keyFactory = SecretKeyFactory.getInstance("PBEWithMD5AndDES");
```

```
SecretKey passwordKey = keyFactory.generateSecret(keySpec);
byte[] salt = new byte[8];
Random rnd = new Random();
rnd.nextBytes(salt);
int iterations = 100;
PBEParameterSpec parameterSpec = new PBEParameterSpec(salt, iterations);
Cipher cipher = Cipher.getInstance("PBEWithMD5AndDES");
cipher.init(Cipher.ENCRYPT_MODE, passwordKey, parameterSpec);
outFile.write(salt);
byte[] input = new byte[64];
int bytesRead;
while ((bytesRead = inFile.read(input)) != -1) {
    byte[] output = cipher.update(input, 0, bytesRead);
    if (output != null) {
        outFile.write(output);
    }
}
byte[] output = cipher.doFinal();
if (output != null) {
    outFile.write(output);
}
inFile.close();
outFile.flush();
outFile.close();
FileInputStream fs;
fs = new FileInputStream("./enc.txt");
```

```
    DataInputStream in = new DataInputStream(fs);
    BufferedReader br = new BufferedReader(new InputStreamReader(in));
    String get;
    while((get=br.readLine())!=null)
    {
        jTextArea1.append("\n"+get);
    }
        JOptionPane.showMessageDialog(null, "encrypted");
    } catch (Exception e) {
        System.out.println(e);
    }
}

private void jTextField1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
     try {
        ServerSocket ser = new ServerSocket(140);
        Socket soc = ser.accept();
        BufferedOutputStream out = new BufferedOutputStream(soc.getOutputStream());
        InputStream in = new BufferedInputStream(new FileInputStream("./enc.txt"));
        byte[] buf = new byte[100];
        int length;
        while ((length = in.read(buf)) > 0) {
                out.write(buf, 0, length);
            }
            out.close();
            soc.close();
        } catch (Exception e) {
            System.out.println(e);

        }
    }

    private void jTextArea1MouseClicked(java.awt.event.MouseEvent evt) {
 try {
            jTextArea1.setText(" ");
            String get;
            //          ServerSocket ss=new ServerSocket(1000);
            Socket s1 = new Socket("localhost", 1000);
            BufferedReader br = new BufferedReader(new InputStreamReader(s1.getInputStream()));

            while ((get = br.readLine()) != null) {
                System.out.println(stat);
                if (!stat.equals("full")) {
                    jTextArea1.append("\n" + get);
//                  JOptionPane.showMessageDialog(this, "already exists");
                } else {
                    try {
                        ServerSocket ser = new ServerSocket(1002);
                        Socket soc = ser.accept();

                        BufferedOutputStream out = new BufferedOutputStream(soc.getOutputStream());

                        PrintStream ps=new PrintStream(soc.getOutputStream());

                        ps.println(get);

                        ps.flush();

                        JOptionPane.showMessageDialog(this, "transfer to next agent");

                    } catch (Exception e) {

                        System.out.println(e);


                    }

                }

            }

            stat = "full";

            s1.close();

            //          JOptionPane.showMessageDialog(null, "encrypted");

        } catch (Exception e) {

            System.err.println(e);

        }

        try

        {

//          ServerSocket ss=new ServerSocket(1000);

            Socket s1=new Socket("localhost",1001);

            BufferedReader br = new BufferedReader(new InputStreamReader(s1.getInputStream()));

            original=br.readLine();

            jTextArea1.setText(original);

        }
```

```
    catch(Exception e)
    {
      System.err.println(e);
    }
  }

  private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
      try {
        filename = "./data.txt";
        String password = "super_secret";
        inFile = new FileInputStream(filename);
        FileWriter fw = new FileWriter("./enc.txt");
        outFile = new FileOutputStream("./enc.txt");
        PBEKeySpec keySpec = new PBEKeySpec(password.toCharArray());
        SecretKeyFactory keyFactory = SecretKeyFactory.getInstance("PBEWithMD5AndDES");
        SecretKey passwordKey = keyFactory.generateSecret(keySpec);
        byte[] salt = new byte[8];
        Random rnd = new Random();
        rnd.nextBytes(salt);
        int iterations = 100;
        PBEParameterSpec parameterSpec = new PBEParameterSpec(salt, iterations);
        Cipher cipher = Cipher.getInstance("PBEWithMD5AndDES");
        cipher.init(Cipher.ENCRYPT_MODE, passwordKey, parameterSpec);
        outFile.write(salt);
        byte[] input = new byte[64];
        int bytesRead;
    while ((bytesRead = inFile.read(input)) != -1) {
        byte[] output = cipher.update(input, 0, bytesRead);
        if (output != null) {
            outFile.write(output);
        }
    }
    byte[] output = cipher.doFinal();
    if (output != null) {
        outFile.write(output);
    }
    inFile.close();
    outFile.flush();
    outFile.close();
     FileInputStream fs;
fs = new FileInputStream("./enc.txt");
DataInputStream in = new DataInputStream(fs);
BufferedReader br = new BufferedReader(new InputStreamReader(in));
String get;
while((get=br.readLine())!=null)
{
    jTextArea2.append("\n"+get);
}

    JOptionPane.showMessageDialog(null, "Re-encrypted");

} catch (Exception e) {
        System.out.println(e);
    }
}

/**
* @param args the command line arguments
*/
public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new age2().setVisible(true);
        }
    });
}

// Variables declaration - do not modify
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JButton jButton3;
private javax.swing.JButton jButton4;
private javax.swing.JButton jButton5;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JPanel jPanel2;
```

```
    private javax.swing.JPanel jPanel3;

    private javax.swing.JPanel jPanel4;

    private javax.swing.JScrollPane jScrollPane1;

    private javax.swing.JScrollPane jScrollPane2;

    private javax.swing.JTabbedPane jTabbedPane1;

    private javax.swing.JTextArea jTextArea1;

    private javax.swing.JTextArea jTextArea2;

    private javax.swing.JTextField jTextField1;

    private javax.swing.JTextField jTextField2;

    private javax.swing.JTextField jTextField3;

    // End of variables declaration


}
```