INTRODUCTION

1.1 Project Objective

In the course of doing business, sometimes sensitive data must be handed over to supposedly trusted third parties. For example, a hospital may give patient records to researchers who will devise new treatments. Similarly, a company may have partnerships with other companies that require sharing customer data. Another enterprise may outsource its data processing, so data must be given to various other companies. We call the owner of the data the *distributor* and the supposedly trusted third parties the *agents*. Our goal is to *detect* when the distributor's sensitive data has been *leaked* by agents, and if possible to identify the agent that leaked the data.

We consider applications where the original sensitive data cannot be perturbed. Perturbation is a very useful technique where the data is modified and made "less sensitive" before being handed to agents. For example, one can add random noise to certain attributes, or one can replace exact values by ranges.

However, in some cases it is important not to alter the original distributor's data. For example, if an outsourcer is doing our payroll, he must have the exact salary and customer bank account numbers. If medical researchers will be treating patients (as opposed to simply computing statistics), they may need accurate data for the patients. Traditionally, leakage detection is handled by watermarking, e.g., a unique code is embedded in each distributed copy. If that copy is later discovered in the hands of an unauthorized party, the leaker can be identified. Watermarks can be very useful in some cases, but again, involve some modification of the original data.

Furthermore, watermarks can sometimes be destroyed if the data recipient is malicious. In this project we study unobtrusive techniques for detecting leakage of a set of objects or records. Specifically, we study the following scenario: After giving a set of objects to agents, the distributor discovers some of those same objects in an unauthorized place. (For example, the data may be found on a web site, or may be obtained through a legal discovery process.) At this point the distributor can assess the likelihood that the leaked data came from one or more agents, as opposed to having been independently gathered by other means. Using an analogy with cookies stolen from a cookie jar, if we catch Freddie with a single cookie, he can argue that a friend gave him the cookie. But if we catch Freddie with 5 cookies, it will be much harder for him to argue that his hands were not in the cookie jar. If the distributor sees "enough evidence" that an agent leaked data, he may stop doing business with him, or may initiate legal proceedings.

In this project we develop a model for assessing the "guilt" of agents. We also present algorithms for distributing objects to agents, in a way that improves our chances of identifying a leaker. Finally, we also consider the option of adding "fake" objects to the distributed set. Such objects do not correspond to real entities but appear realistic to the agents. In a sense, the fake objects acts as a type of watermark for the entire set, without modifying any individual members. If it turns out an agent was given one or more fake objects that were leaked, then the distributor can be more confident that agent was guilty.

1.2 Scope of the Project

The software, Site Explorer is designed for management of web sites from a remote location. This Document plays a vital role in the development life cycle (SDLC) and it describes the complete requirement of the system. It is meant for use by the developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process.

1.3 Motivation

1.3.1 Definitions

Data Leakage

A data breach is the unintentional release of secure information to an untrusted environment.

Data Privacy

Information privacy, or data privacy is the relationship between collection and dissemination of data, technology, the public exception of privacy, and the legal and political issues surrounding them.Privacy concerns exist wherever personally identifiable information is collected and stored - in digital form or otherwise. Improper or non-existent disclosure control can be the root cause for privacy issues.

Fake Records

Records which are false or containing misleading appearance.

Unobstrusive Techniques

Unobtrusive technique is a technique of data collection. They describe methodologies which

do not involve direct elicitation of data from the research subjects. The unobtrusive approach often seeks unusual data sources.

Fake

One of the circles or windings of a cable or hawser, as it lies in a coil; a single turn or coil. To coil (a rope, line, or hawser), by winding alternately in opposite directions, in layers usually of zigzag or figure of eight form,, to prevent twisting when running out.

Guilt

Guilt is the state of being responsible for the commission of an offense. It is also a cognitive or an emotional experience that occurs when a person realizes or believes accurately or not that he or she has violated a moral standard, and bears significant responsibility for that violation. It is closely related to the concept of remorse.

1.3.2 Abbreviations

ACRONYM	ABBREVATION
DLD	Data Leakage Detection
GA	Guilty Agent
AGM	Agent Guilt Model
DA	Data Allocation
GMA	Guilt Model Analysis
FO	Fake Objects
FT	Fake Tuple
DR	Data Request
EF	Explicit Fake Object
SF	Simple Fake Object
SR	Simple Request
ER	Explicit Request
CFO	Create Fake Object

IGP	Internet Grouping Protocol
EGP	Exterior Gateway Protocol
IGRP	Interior Gateway Routing Protocol
VLSM	Variable Length Subnet Masking
PL	Physical Layer
LL	Link Layer
NL	Network Layer
TL	Transport Layer
SL	Session Layer
PL	Presentation Layer
AL	Application Layer
NIDS	Network Instruction Detection System
SSL	Secure Socket Layer
HTTP	Hyper Text Transfer Protocol
FTP	File Transfer Protocol
DMZ	Demilitarized Zone
SMTP	Simple Mail Transfer Protocol
POP3	Post Office Protocol version 3
PP	Pre-Pishing
SQL	Structured Query Language
EGP	Exterior Gateway Protocol
IGMP	Internet Group Management Protocol
MBGP	Multiprotocol Extension for BGP
RIP	Routing Information Protocol
MTU	Maximum Transmission Unit
RFC	Request For Comment

MLD	Multicast Listener Discovery
PIM	Protocol Independent Multicast
IETF	Internet Engineering Task Force
ICMP	Internet Control Message Protocol
EIGRP	Enhanced Interior Gateway Routing Protocol
OMT	Object Modelling Technique
NSFNet	National Science Foundation Network
ARPANet	Advanced Research Project Agency Network

1.3.3 Model Diagrams

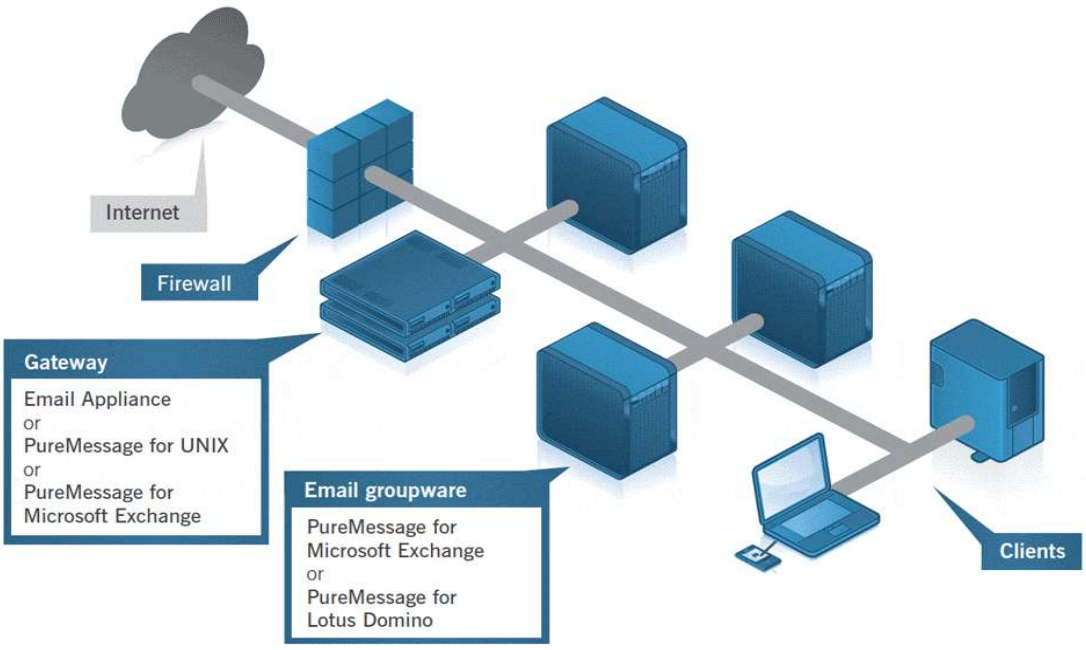


Figure 1.1 Model Diagram for Email Security Control

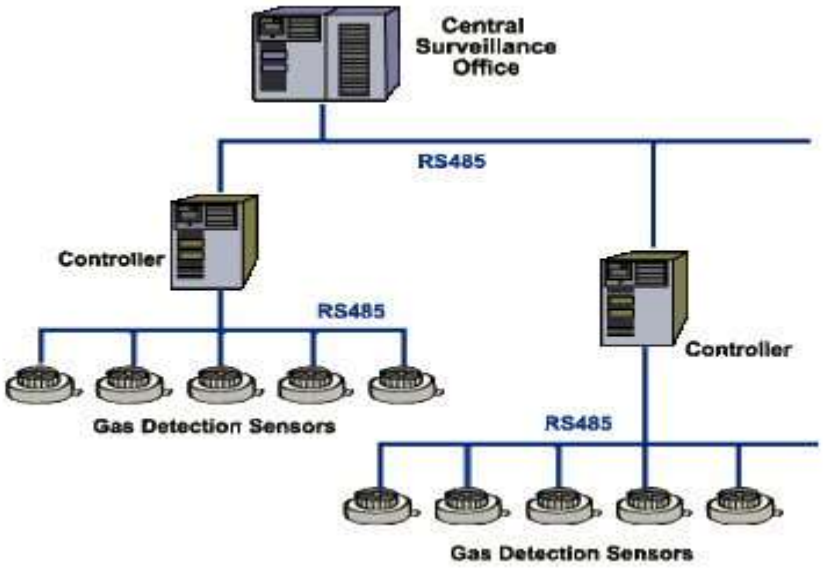


Figure 1.2 Model Diagram for Gas Leakage

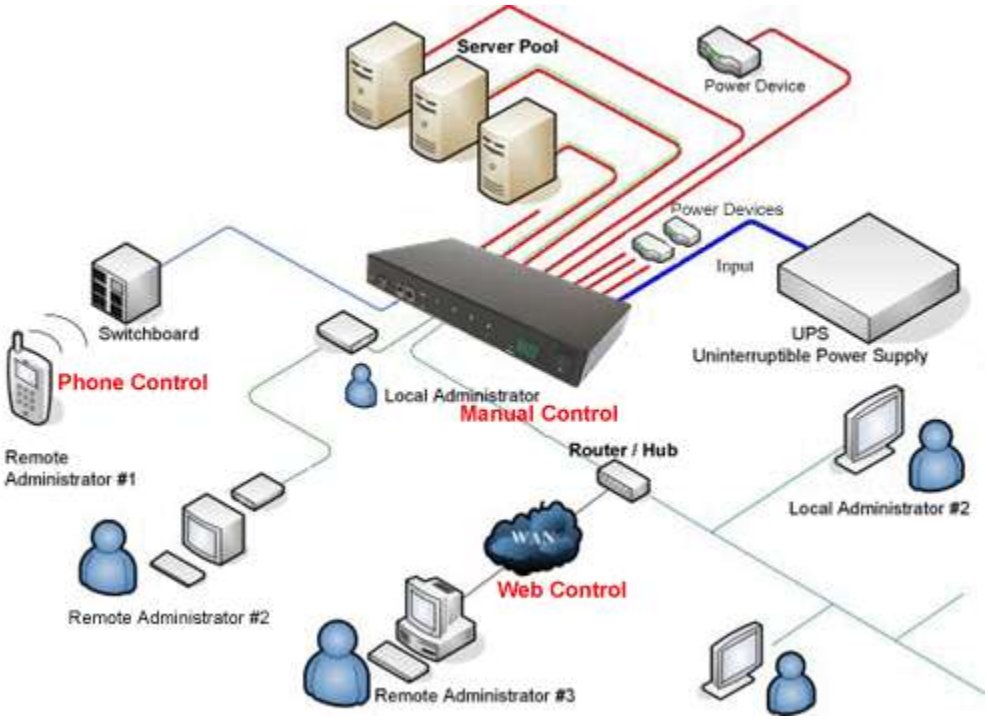


Figure 1.3 Model Diagram for IP-P3-Demo Leakage

1.4 Over view of the Project

Traditionally, leakage detection is handled by *watermarking*, e.g., a unique code is embedded in each distributed copy. If that copy is later discovered in the hands of an unauthorized party, the leaker can be identified. Watermarks can be very useful in some cases, but again, involve some modification of the original data. Furthermore, watermarks can sometimes be destroyed if the data recipient is malicious.

In this project we study *unobtrusive* techniques for detecting leakage of a *set* of objects or records. Specifically we study the following scenario: After giving a set of objects to agents, the distributor discovers some of those same objects in an unauthorized place. (For example, the data may be found on a web site, or may be obtained through a legal discovery process.) At this point the distributor can assess the likelihood that the leaked data came from one or more agents, as opposed to having been independently gathered by other means. Using an analogy with cookies stolen from a cookie jar, if we catch Freddie with a single cookie, he can argue that a friend gave him the cookie.

In this project we develop a model for assessing the "guilt" of agents. We also present algorithms for distributing objects to agents, in a way that improves our chances of identifying a leaker. Finally, we also consider the option of adding "fake" objects to the distributed set. Such objects do not correspond to real entities but appear realistic to the agents. In a sense, the fake objects acts as a type of watermark for the entire set, without modifying any individual members. If it turns out an agent was given one or more fake objects that were leaked, then the distributor can be more confident that agent was guilty

The developer is responsible for:

- Developing the system, which meets the SRS and solving all the requirements of the system?
- Demonstrating the system and installing the system at client's location after the acceptance testing is successful.
- Submitting the required user manual describing the system interfaces to work on it and also the documents of the system.
- Conducting any user training that might be needed for using the system. Maintaining the system for a period of one year after installation.

LITERATURE SUREVEY

2.1 Introduction

Literature survey is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy n company strength. Once these things r satisfied, ten next steps are to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from book from websites. Before building the system the above consideration r taken into account for developing the proposed system.

We have to analysis the "**Monitoring for Detection & Prevention of Fake Agents** (Data leakage Detection)

To compute this *Pr{Gi|S}*, we need an estimate for the probability that values in *S* can be "guessed" by the target. For instance, say some of the objects in *S* are emails of individuals. We can conduct an experiment and ask a person with approximately the expertise and resources of the target to find the email of say 100 individuals. If this person can find say 90 emails, then we can reasonably guess that the probability of finding one email is 0.9. On the other hand, if the objects in question are bank account numbers, the person may only discover say 20, leading to an estimate of 0.2. We call this estimate *pt*, the probability that object *t* can be guessed by the target.

Probability *pt* is analogous to the probabilities used in designing fault-tolerant systems. That is, to estimate how likely it is that a system will be operational throughout a given period, we need the probabilities that individual components will or will not fail. A component failure in our case is the event that the target guesses an object of *S*.

The component failure is used to compute the overall system reliability, while we use the probability of guessing to identify agents that have leaked information. The component failure probabilities are estimated based on experiments, just as we propose to estimate the *pt*'s. Similarly, the component probabilities are usually conservative estimates, rather than exact numbers.

For example, say we use a component failure probability that is higher than the actual probability, and we design our system to provide a desired high level of reliability. Then we will know that the actual system will have at least that level of reliability, but possibly higher. In the same way, if we use *pt*'s that are higher than the true values, we will know that the agents will be guilty with at least the computed probabilities. To simplify the formulas that we present in the rest

of the project, we assume that all *T* objects have the same *pt*, which we call *p*. Our equations can be easily generalized to diverse *pt*'s though they become cumbersome to display.

Next, we make two assumptions regarding the relationship among the various leakage events. The first assumption simply states that an agent's decision to leak an object is not related to other objects. We study a scenario where the actions for different objects are related, and we study how our results are impacted by the different independence assumptions.

2.2 Internal Threats – Intentional or Inadvertent?

According to data compiled from EPIC.org and PerkinsCoie.com, 52% of Data Security breaches are from internal sources compared to the remaining 48% by external hackers . The internal noteworthy breaches aspect are of these examined, the figures is percentage that, due to

when the malicious intent is remarkably low, at less than 1%. The corollary of this is that the level of inadvertent data breach is significant (96%). This is further deconstructed to 46% being due to employee oversight, and 50% due to poor business process .

2.2.1 Intentional Internal Data Leakage or sabotage

Whilst the data presented suggests the main threat to internal data leakage is from inadvertent actions, organizations are nevertheless still at risk of intentional unauthorized release of data information by internal users. The methods by which insiders leak data could be one or many, but could include mediums such as Remote Access , Instant Messaging, email, Web Mail, Peer-to-peer, and even File Transfer Protocol. Use of removable media, hard copy, etc is also possible.

Motivations are varied, but include reason such as corporate espionage, financial reward, or a grievance with their employer. The latter appears to be the most likely. According to a study conducted by The US Secret Service and CERT, 92% of insider related offences was following a " negative work-related event". Of these, the offenders were predominantly male (96%) and the these attacks related not just to data, of the attacks studied, 49% included the objective of "sabotaging information and/or data". An example of such an attack is described in the USSS/CERT study as follows, note how the characteristics match the finding above (highlighted in bold):

2.2.2 Unintentional internal Data Leakage

As discussed earlier in this section, a significant amount of data security breaches are due to either employee oversight or poor business process. This presents a challenge for business as the solution to these problem will be fat greater than simply deploying a secure content management

system. Business processes will need to be retrained, and probably re-engineered; personnel will need to be retrained, and a cultural change may be required with in the organization. These alone are significant challenges for business. A recent example of what is probably unintentional featured an Australian employment agency's web site publishing **"confidential data including names, email address and passwords of clients"** from its database on the public web site. An additional embarrassing aspect of this story was the fact that some of the agency's staff made comments regarding individuals, which were also included.

2.3 Internal Data Leakage Vectors

2.3.1 Instant Messaging / Peer – to – Peer

Many organizations allow employees to access Instant Messaging from their workstations or laptops, with 2005 estimate suggesting 80% of large companies in the US having some form of Instant Messaging. This includes product such as MSN Messenger, Skype, AOL, Google Talk, ICQ, and numerous others. Many of the clients available (and all of those mentioned here) are capable of file transfer. It would be a simple process for an individual to send a confidential document (such as an Excel file containing sensitive pricing or financial data) to a third party. Equally a user could divulge confidential information in an Instant Messaging chat Session.

Instant Messaging is also increasingly becoming a vector for Malware. For example the highly popular Skype has been targeted in recent times. Recent examples of malware targeting Skype include W32/Pykse.worm.b, W32/Skipi.A and W32.Pykpa.D.

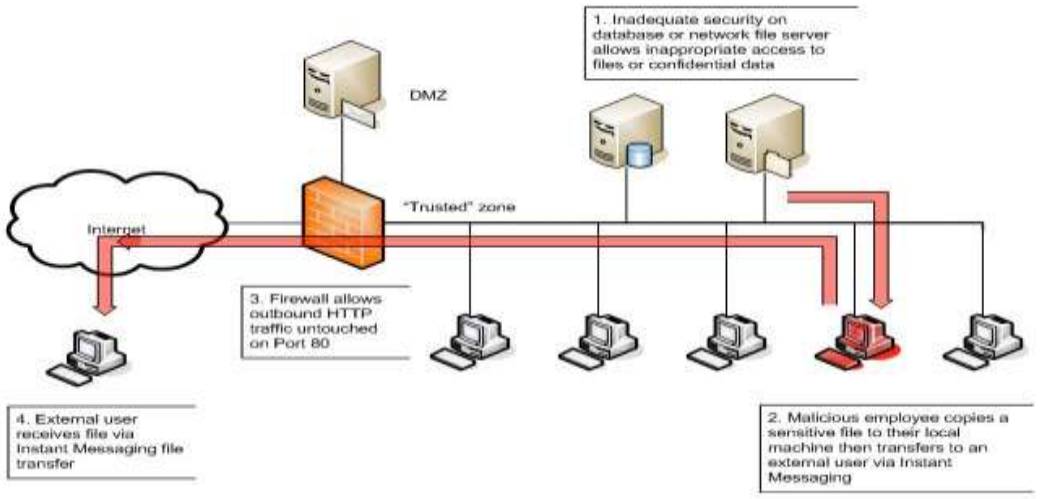


Figure 2.3.1 Instant Messaging Data Leakage Vector

Peer – to – Peer (P2P) also presents presents a significant threat to data confidentiality. Popular P2P clients include eDonkey and Bit Torrent with the latter appearing to have between 50 and 75% share of global P2P traffic. It has recently been described as "new national security risk" by Retired General Wesley K. Clark, who is a board member with an organization that scans through peer-to-peer networks for confidential or sensitive data. He commented "We found more than 200 classified government documents in a few hours search over P2P networks" and "We found everything from Pentagon network server secrets to other sensitive information on P2P networks that hackers dream about".

A few moments consideration regarding the implications of these findings will yield the issue of potential widespread distribution and availability of the data. The number of potential users on P2P networks that could access the confidential or sensitive data is enormous.

2.3.2 Email

Traditional email clients, such as Microsoft Outlook, Lotus Notes, Eudora, etc. are ubiquitous within organizations. An internal user with the motivation could email a confidential document to an unauthorized individual as an attachment. They may also choose to compress and / or encrypt the file, or embed it within other files in order to disguise its presence. Steganography may also be utilized for this purpose. Alternatively, instead of attaching a document, text could be copied into the email message body.

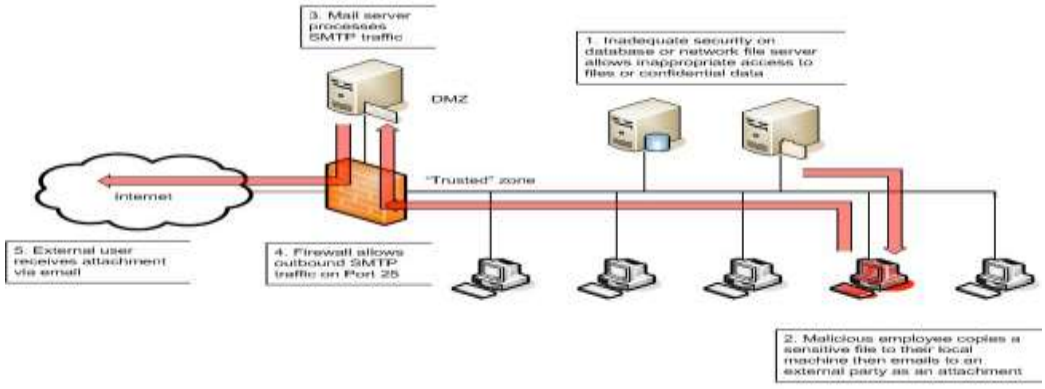


Figure 2.3.2 Email Data Leakage Vector

Email also represents a vector for inadvertent disclosure due to employee oversight or poor business process. An employee could attach the wrong file inadvertently, select the wrong recipient in the email, or even be tricked into sending a document through social engineering.

2.3.3 Web Mail

Web Mail is well entrenched with users. Gmail, Yahoo, and Hotmail are popular examples. It represents another way for an individual to leak confidential data, either as an attachment or in the message body. Because Web Mail runs over HTTP/S firewall may allow it through un-inspected as port 80 or 443 will in most organizations be allowed, and the connection is initiated from an internal IP address. HTTPS represents a more complex challenge due to the encryption of the traffic.

2.3.4 Web Logs / Wikis

Web Logs (Blogs) are web sites where people can write their thoughts, comments, opinions on a particular subject. The blog site may be their own, or a public site, which could include the input from thousands of individuals. Blogs could be used by someone to release confidential information, simply through entering the information in their blog. However, they would most likely be able to be tracked, so this is perhaps a less likely medium. A wiki site is "a collaborative website which can be directly edited by anyone with access to it", such as wikipedia.org. These sites are often available to most internet users around the world, and contain the possibility that confidential information may be added to a wiki page.

2.3.5 Malicious Web Pages

Web sites that are either compromised or are deliberately malicious, present the risk of a user's computer being infected with malware, simply by visiting a web page containing malicious code with an OS/Browser that contains vulnerability. The malware could be in the form of a key logger, Trojan, etc. *With a Key logger the risk of data theft is introduced.* A recent example was the Miami Dolphins (host to the NFL super Bowl XLI) web site being compromised. Users with Vulnerabilities MS06-014 and MS07-004 would download a key logger/backdoor, "providing the attacker with full access to the compromised computer".

2.3.6 Hiding in SSL

In order to obfuscate data, a user may attempt to utilize a public proxy service via an SSL connection (often called Proxy Avoidance).

They access the proxy service via a browser, type in the URL of the site they wish to visit, and their entire session is then encrypted. A Stateful Packet Inspection firewall will not be able to examine the data as it will be encrypted. Consequently sensitive information may be leaked through this medium without detection. For example the Mega proxy SSL VPN provides this capability.

2.3.7 File Transfer Protocol (FTP)

FTP is included in this discussion as it represents another (perhaps less likely) method for an individual to release information. It is straightforward to install and configure a basic FTP server external to the organization (or it may be a special folder on a competitor's FTP server). The individual then merely has to install a publicly available FTP client and upload the file or files to the server. This method could even utilize a "dead drop" public FTP site hosted off-shore, where the third party also has access. As FTP is a popular protocol there is the likelihood it will be allowed through the firewall. FTP is probably more likely to be used in intentional leakage than unintentional leakage, due to the fact that uploading a file to an FTP server is generally not something an average user performs on a daily basis, nor would do inadvertently, as compared to attaching a file to an email.

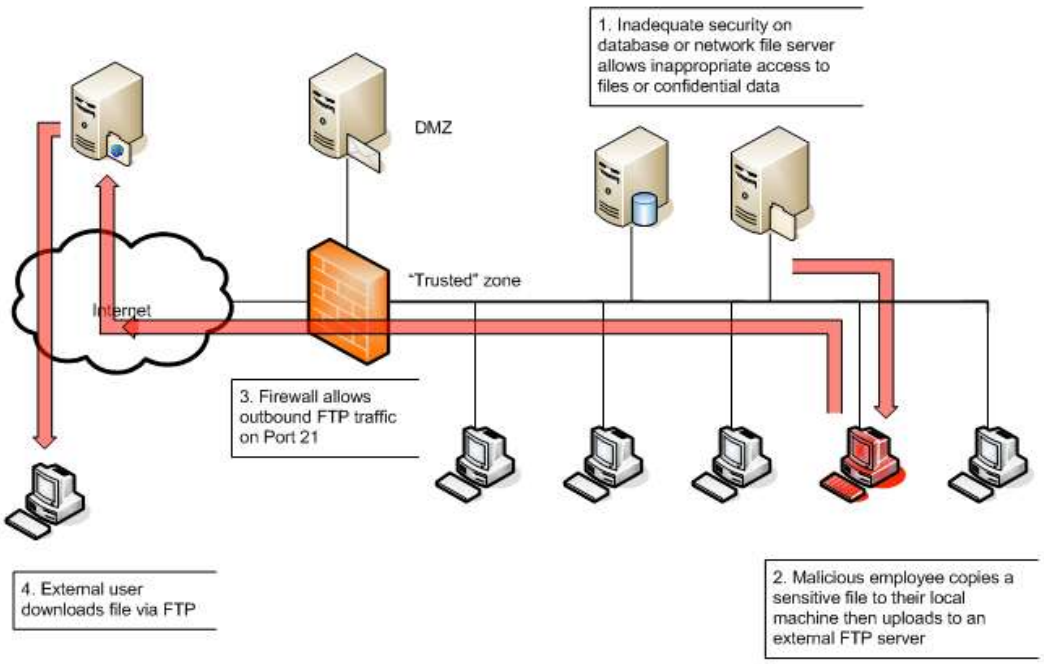


Figure 2.3.3. FTP Data Leakage Vector

2.3.8 Removable Media / Storage

Symantec reported in March 2007 that "Theft or loss of a computer or data storage medium, such as a USB memory key, made up 54 percent of all identity theft-related data breaches".

This is very cheap removable storage. Copying a large spread sheet or document (say 500MB) onto a USB key is effortless. The user merely needs to insert the device, open Windows Explorer, and drag and drop the target files to the device.19 The key is then removed, placed in the employees pocket and walked out of the building. Alternatively, if the user has a CD or DVD burner on their laptop or desktop, they can copy the information that way .

2.3.9 Security Classification errors

Security models such as Biba and Bell LaPadula21 are intended to provide a framework for organizations to avoid classified and / or sensitive information being sent to individuals (internally and externally) without the appropriate security clearance level. It is conceivable that an individual with Top Secret clearance may either intentionally or inadvertently send a Top Secret document to another individual with only "Classified" clearance .

2.3.10 Hard copy

If an individual wishes to provide a competitor with sensitive material, and the victim organization has already implemented electronic countermeasures, it is still possible for the individual to print out the data and walk out of the office with it in their briefcase. Or, they simply place it in an envelope and mail it, postage happily paid by the victim organization!

2.3.11 Cameras

Again, if an organization has implemented a range of protective measures, the prevention of the escape of information is still not guaranteed. A determined individual may choose to take digital photos (or non-digital for that matter) of their screens. A camera is not even needed nowadays. Cellular telephones today are likely to have a camera built in, perhaps with up to 2 mega pixels or more. The photo could then be sent by email or Mobile Messaging directly from the telephone.

2.3.12 Inadequate folder and file protection

If folders and files lack appropriate protection (via user/group privileges etc) then it becomes easy for a user to copy data from a network drive (for example) to their local system. The user could then copy that file to removable media, or send it out externally by methods discussed above.

2.3.13 Inadequate database security

Poor SQL programming can leave an organization exposed to SQL injection attacks, or allow inappropriate information to be retrieved in legitimate database queries. Additionally, organizations

should not implement broad database privileges22 (i.e. one-size-fits- all) as this can lead to users accessing confidential information (either intentionally or inadvertently).

2.4 External Threats

According to the Privacy Rights clearinghouse, in 2005 US companies exposed the personal information of over in 53 million People.

2.4.1 Data theft by intruders

An ever-popular topic in the media is the electronic break-in to an organization by intruders including the theft of sensitive information. There have been numerous stories in the press of the theft of credit card information by intruders (note that the press often refer to intruders as hackers). In 2005 it was estimated that as many as 40 Million credit card numbers were stolen by intruders from Master Card, VISA, American Express, and other credit card brands.

More recently, Monster.com lost hundreds of thousands (potentially as many as 1.3 million) of job site users' IDs to intruders "...hackers grabbed resumes and used information on those documents to craft personalized "phishing" e-mails to job seekers."

This particular event holds significant concern, because resumes contain a significant amount of information about an individual, including their full name, address, phone number(s), employment history, interests, and possibly contact details of third parties, such as referees. This allows for particularly targeted, and if crafted well, believable phishing attacks or perhaps even more audacious social engineering attacks such as phone calls.

2.4.2 SQL Injection

Web sites that use an SQL server as the back end database may be vulnerable to SQL Injection attacks, if they fail to correctly parse user input. This is usually a direct result of poor coding. SQL Injection attacks can result in content within the database being stolen.

For example, a site that does not correctly sanitize user input may cause a server error to occur. For example: The initial action of the attack could be to enter a single quote within the input data in a POST element on a website, which may generate an SQL statement as follows:

SELECT info

FROM table

WHERE search = 'mysearch''

Note the additional quote mark. Should the application not sanitize the user input correctly a server error may occur. This indicates to the attacker that the user input is not being sanitized and that the site is vulnerable to further exploitation. Further trial and error by the attacker could eventually reveal table names, field names, and other information, that, once obtained, will allow them to construct an SQL query within the POST element that yields sensitive data.

2.4.3 Malware

In recent years, the sircam worm would, after infecting a computer, Scan through the My documents folder and send a file at random out via email to the user's email contacts. If malware is classified as a zero day threat, and there is no signature yet available, there is a higher likelihood that the malware will evade inbound gateway protection measures and desktop anti-virus. Once this malware infects a PC, it may then initiate outbound communications, potentially sending out files which may contain sensitive data. One aspect to be mindful of is that to a firewall, the traffic is from an internal source. This is an important point, because most firewalls will not restrict traffic that is initiated internally via an acceptable protocol.

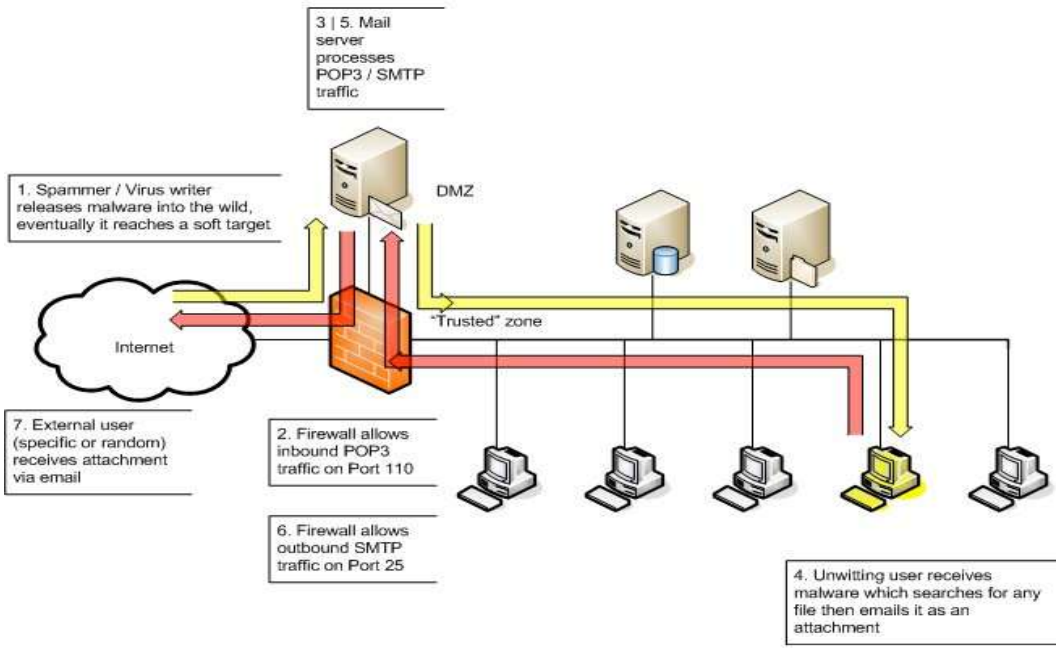


Figure 24.1 Malware Data Leakage Vector

As discussed key loggers present a threat as they capture potentially sensitive information, such as login credentials , personal information, leading to the risk of identity theft.

2.4.4 Dumpster Diving

Organizations that do not take appropriate care with the destruction of hard copy information run the risk of confidential information falling into unauthorized hands. Instead of having such information destroyed securely, businesses may simply throw their confidential information (perhaps unwittingly) into the rubbish. An attacker may decide to raid the company's dumpster and discover this information. This extends to information stored on media such as CDs and DVDs, as well as printed material. s

2.4.5 Phishing and pre-phising

Phishing sites, and the spam email that solicits visits to them, pose a threat to organizations, and not just individuals. Phishing spam may be received at peoples' work email address. Should they be fooled into visiting the phishing site, then they may lose personal information and or financial information. It is also possible that a key logger (as previously download the spam received directs them to a site hosting malware, which could discussed). Phishers have recently been using the lure of tax returns from various taxation offices as a means to fool people. For example in Australia, the Australian Tax Office has been targeted by phishers.29 Phishing is of course a form of social engineering (which will be discussed shortly).

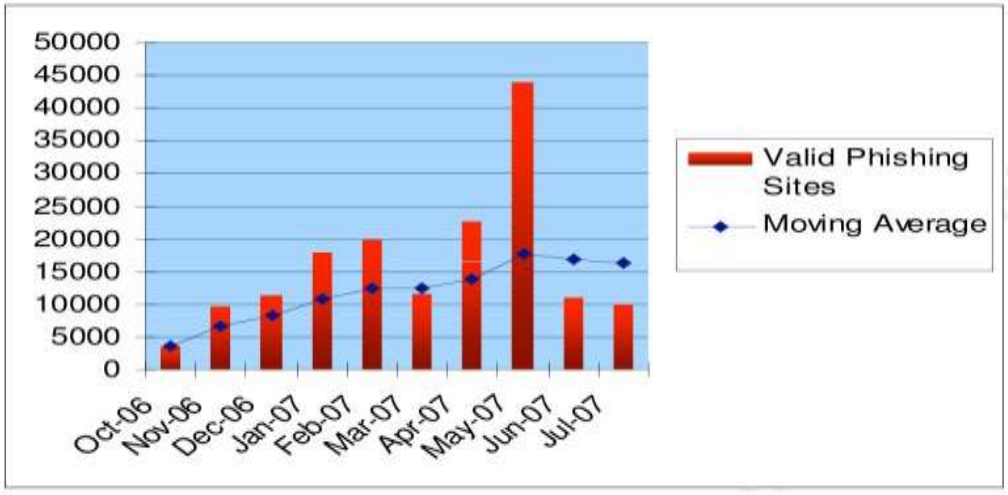


Figure 2.4.2 Phishing site activity

Pre-Phishing

Pre-phishing is emerging as a new method used by phisher, initially as a reconnaissance attack. Instead of attempting to directly obtain credentials for a financial site, social networking and email sites are targeted. The attack seeks to obtain username and password combinations, on

the (likely) assumption that in many cases, users will use the same or similar combinations on other web sites. The second part of the attack is to conduct a CSS History Hack, where the phishers can determine whether the user has visited specified sites. The CSS History Hack uses the 'a:visited' component in CSS which alters the behavior of links that have been visited. Banking sites visited by users may be obtained, and the phishers can then visit these and attempt to gain access using the compromised credential combinations.

2.4.6 Social Engineering

Without going into excessive detail about Social Engineering, some of the common scenarios and risks include:

→ Phone calls to Help Desk from a social engineer claiming to be an employee in another office, desperate for a password reset.

→ Phone calls to unsuspecting employees from social engineer tricking them into sending out sensitive information. Individuals that would not recognize the fact that the information is sensitive are prime targets.

2.4.7 Physical Theft

Physical theft of computer systems, laptops, backup tapes, and other media also presents a data leakage risk to organizations. This may be due to poor physical security at an organization's premises or poor security practice by individuals. For instance, a laptop may be left unattended in the back seat of a car whilst the owner pays for petrol, allowing an opportunistic theft to occur. Also possible is the mass theft of laptops from within an organizations premises after hours, should the business fail to secure the laptops overnight.

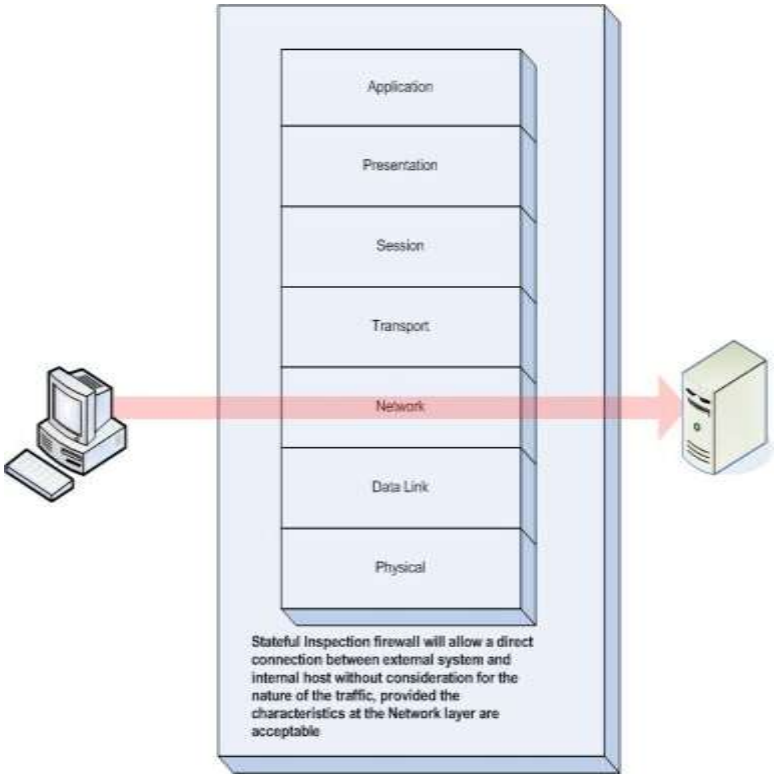


Figure 2.4.3 Stateful Inspection Firewall conceptual diagram

Architecture

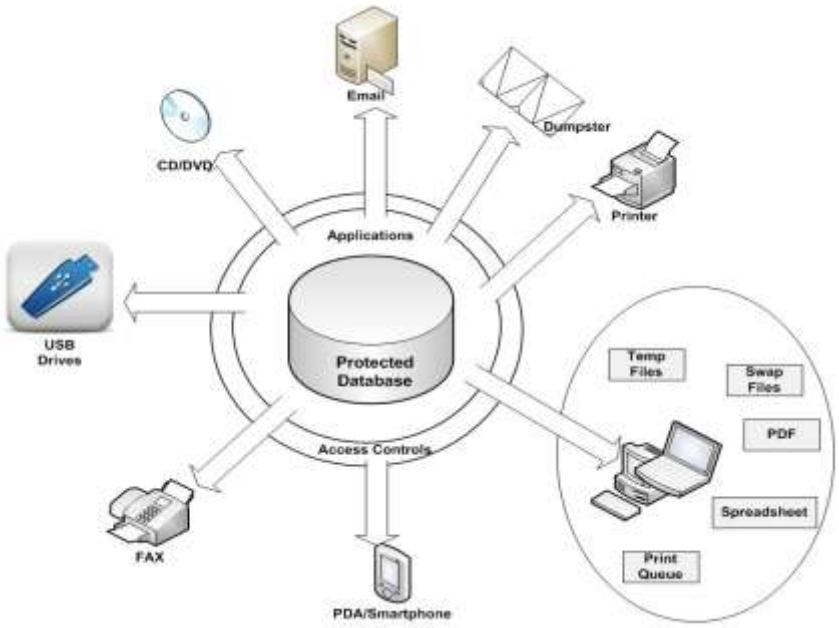


Figure 2.4.4 SSL Proxy conceptual diagram

SSL Tunnelling Mitigation

In order to obfuscate the sending of data, a more technically savvy individual may choose to create an SSL tunnel in which to send their data. As SSL data is normalized, it is very difficult for many firewalls and security appliances to detect the nature of the data in the message. There are a small number of products that can inspect SSL traffic. This is achieved by a device acting as an SSL proxy. Please refer to the diagram below during the explanation of this concept. The client system initiates an SSL handshake with the Proxy with a GET request for a secure web page. The proxy then initiates a secure session with the host. The host and the proxy perform a key exchange and the host issues a certificate to the proxy . The proxy checks the certificate against Certificate Revocation Lists. It then relays the GET request for the page. The secure server then delivers the page to the proxy. The proxy decrypts this traffic so then has the clear text of the communication, and this can be inspected according to defined policies for malware, confidential information, etc.

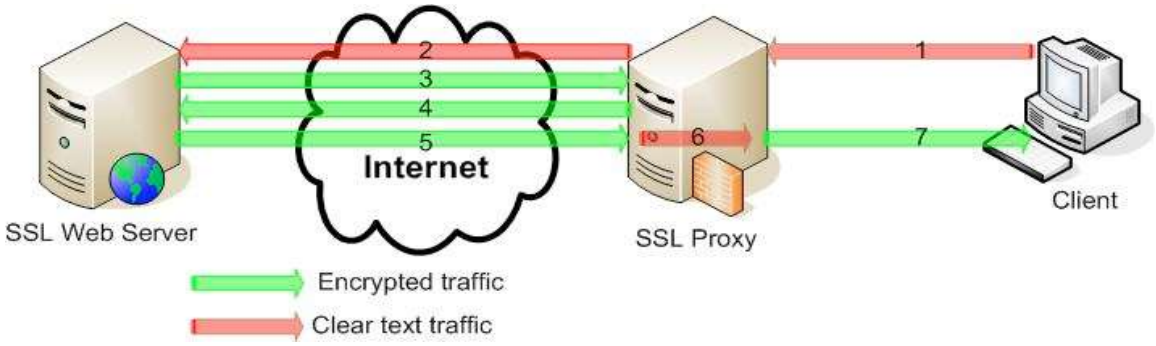


Figure 2.4.5 SSL Proxy conceptual diagram

Alternatively, an organization may consider blocking SSL traffic on port 443 completely, or via web filtering (see below) as a means to prevent this. However this will obviously prevent users from acceptable usage, such as online banking, etc, so may not be practical.

Advantages: Will detect encrypted traffic that users are utilizing to bypass other security measures.

Disadvantages: Limited vendors providing this type of solution will involve additional cost.

2.5 Performance Requirements

Performance is measured in terms of the efficiency of the signing and verification modules. Digital Signature project uses various singing algorithms, which are very fast and reliable. Since it provides GUI it should generate and perform all the events as specified

The specific requirement gives the expected behaviour of the System Following are the requirements of the System:

1. The system will repair any damages occurred in the images by using the concept of segmentation or sector division.

2. The system can also use for making the blurred images to somehow better visual enhance to make the appearance called the clarity of the image by neural network concepts.

3. It also performs the edge detection concept for identification of the edges of the image for removal of blurring.

2.6 Technology Used

2.6.1 Introduction To .Net Framework

The **Microsoft .NET Framework** is a software technology that is available with several Microsoft Windows operating systems. It includes a large library of pre-coded solutions to common programming problems and a virtual machine that manages the execution of programs written specifically for the framework. The .NET Framework is a key Microsoft offering and is intended to be used by most new applications created for the Windows platform.

Programs written for the .NET Framework execute in a software environment that manages the program's runtime requirements. Also part of the .NET Framework, this runtime environment is known as the Common Language Runtime (CLR). The CLR provides the appearance of an application virtual machine so that programmers need not consider the capabilities of the specific CPU that will execute the program. The CLR also provides other important services such as security, memory management, and exception handling. The class library and the CLR together compose the .NET Framework.

Principal design features

Interoperability

Because interaction between new and older applications is commonly required, the .NET Framework provides means to access functionality that is implemented in programs that execute outside the .NET environment. Access to COM components is provided in the System.Runtime.InteropServices and System.EnterpriseServices namespaces of the framework; access to other functionality is provided using the P/rInvoke feature.

Common Runtime Engine

The Common Language Runtime (CLR) is the virtual machine component of the .NET framework. All .NET programs execute under the supervision of the CLR, guaranteeing certain properties and behaviours in the areas of memory management, security, and exception handling.

Base Class Library

The Base Class Library (BCL), part of the Framework Class Library (FCL), is a library of functionality available to all languages using the .NET Framework. The BCL provides classes which encapsulate a number of common functions, including file reading and writing, graphic rendering, database interaction and XML document manipulation.

Simplified Deployment

Installation of computer software must be carefully managed to ensure that it does not interfere with previously installed software, and that it conforms to security requirements. The .NET framework includes design features and tools that help address these requirements.

Security

The design is meant to address some of the vulnerabilities, such as buffer overflows, that have been exploited by malicious software. Additionally, .NET provides a common security model for all applications.

Portability

The design of the .NET Framework allows it to theoretically be platform agnostic, and thus cross-platform compatible. That is, a program written to use the framework should run without change on any type of system for which the framework is implemented. Microsoft's commercial implementations of the framework cover Windows, Windows CE, and the Xbox 360. In addition, Microsoft submits the specifications for the Common Language Infrastructure (which includes the core class libraries, Common Type System, and the Common Intermediate Language), the C# language, and the C++/CLI language to both ECMA and the ISO, making them available as open standards. This makes it possible for third parties to create compatible implementations of the framework and its languages on other platforms.

Architecture

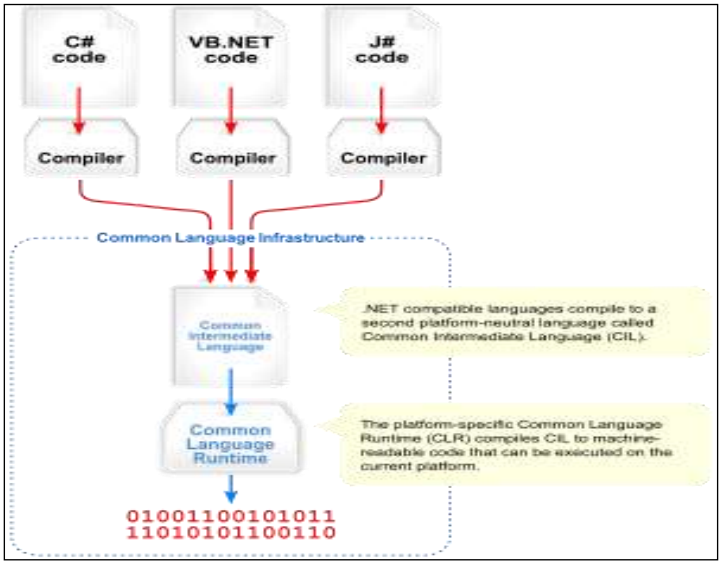


Figure 2.6.1 Visual overview of the Common Language Infrastructure (CLI)

Common Language Infrastructure

The core aspects of the **.NET framework** lie within the Common Language Infrastructure, or **CLI**. The purpose of the CLI is to provide a language-neutral platform for application development and execution, including functions for exception handling, garbage collection, security, and interoperability. Microsoft's implementation of the CLI is called the **Common Language Runtime** or **CLR**.

Assemblies

The intermediate CIL code is housed in .NET assemblies. As mandated by specification, assemblies are stored in the Portable Executable (PE) format, common on the Windows platform for all DLL and EXE files. The assembly consists of one or more files, one of which must contain the manifest, which has the metadata for the assembly. The complete name of an assembly (not to be confused with the filename on disk) contains its simple text name, version number, culture, and public key token. The public key token is a unique hash generated when the assembly is compiled, thus two assemblies with the same public key token are guaranteed to be identical from the point of view of the framework. A private key can also be specified known only to the creator of the assembly and can be used for strong naming and to guarantee that the assembly is from the same author when a new version of the assembly is compiled (required to add an assembly to the Global Assembly Cache).

Metadata

All CLI is self-describing through .NET metadata. The CLR checks the metadata to ensure that the correct method is called. Metadata is usually generated by language compilers but developers can create their own metadata through custom attributes. Metadata contains information about the assembly, and is also used to implement the reflective programming capabilities of .NET Framework.

Class library

Namespaces in the BCL

System

System. CodeDom

System. Collections

System. Diagnostics

System. Globalization

System. IO

System. Resources

System. Text

System.Text.RegularExpressions

Microsoft **.NET Framework** includes a set of standard **class libraries**. The class library is organized in a hierarchy of namespaces. Most of the built in APIs are part of either System.* or Microsoft.* namespaces. It encapsulates a large number of common functions, such as file reading and writing, graphic rendering, database interaction, and XML document manipulation, among others. The .NET class libraries are available to all .NET languages. The .NET Framework class library is divided into two parts: the **Base Class Library** and the **Framework Class Library**.

The **Base Class Library** (BCL) includes a small subset of the entire class library and is the core set of classes that serve as the basic API of the Common Language Runtime. The classes in

mscorlib.dll and some of the classes in System.dll and System.core.dll are considered to be a part of the BCL. The BCL classes are available in both .NET Framework as well as its alternative implementations including .NET Compact Framework, Microsoft Silver light and Mono.

The **Framework Class Library** (FCL) is a superset of the BCL classes and refers to the entire class library that ships with .NET Framework. It includes an expanded set of libraries, including Win Forms, ADO.NET, ASP.NET, Language Integrated Query, Windows Presentation Foundation, Windows Communication Foundation among others. The FCL is much larger in scope than standard libraries for languages like C++, and comparable in scope to the standard libraries of Java.

Versions

Microsoft started development on the .NET Framework in the late 1990s originally under the name of Next Generation Windows Services (NGWS). By late 2000 the first beta versions of .NET 1.0 were released.

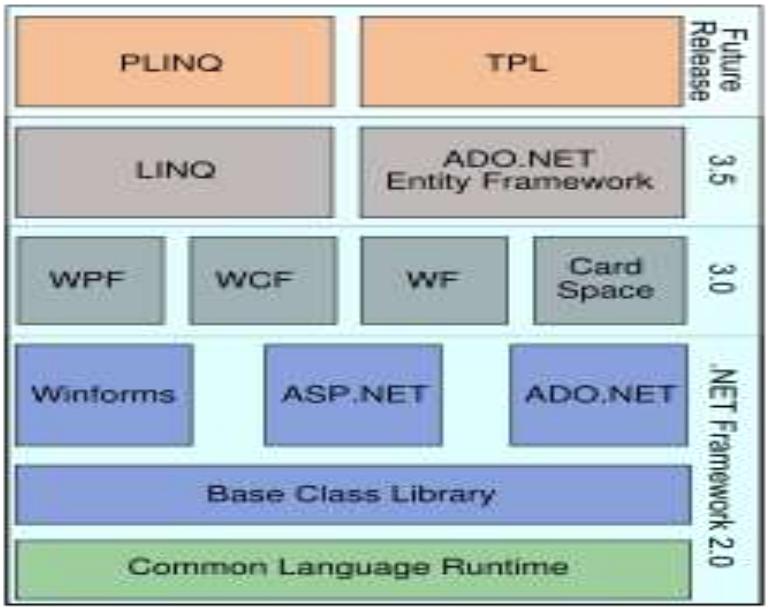


Figure 2.6.2 .NET Framework stack

Version	Version Number	Release Date
1.0	1.0.3705.0	2002-01-05
1.1	1.1.4322.573	2003-04-01
2.0	2.0.50727.42	2005-11-07
3.0	3.0.4506.30	2006-11-06
3.5	3.5.21022.8	2007-11-09

Table 2.6.1 .NET Version Table

2.6.2 ASP.NET

Server Application Development

Server-side applications in the managed world are implemented through runtime hosts. Unmanaged applications host the common language runtime, which allows your custom managed code to control the behavior of the server. This model provides you with all the features of the common language runtime and class library while gaining the performance and scalability of the host server.

The following illustration shows a basic network schema with managed code running in different server environments. Servers such as IIS and SQL Server can perform standard operations while your application logic executes through the managed code.

Server-Side Managed Code

ASP.NET is the hosting environment that enables developers to use the .NET Framework to target Web-based applications. However, ASP.NET is more than just a runtime host; it is a complete architecture for developing Web sites and Internet-distributed objects using managed code. Both Web Forms and XML Web services use IIS and ASP.NET as the publishing mechanism for applications, and both have a collection of supporting classes in the .NET Framework.

Active Server Pages.Net

ASP.NET is a programming framework built on the common language runtime that can be used on a server to build powerful Web applications. ASP.NET offers several important advantages over previous Web development models:

- **Enhanced Performance.** ASP.NET is compiled common language runtime code running on the server. Unlike its interpreted predecessors, ASP.NET can take advantage of early binding, just-in-time compilation, native optimization, and caching services right out of the box. This amounts to dramatically better performance before you ever write a line of code.

- **World-Class Tool Support.** The ASP.NET framework is complemented by a rich toolbox and designer in the Visual Studio integrated development environment. WYSIWYG editing, drag-and-drop server controls, and automatic deployment are just a few of the features this powerful tool provides.

- **Power and Flexibility.** Because ASP.NET is based on the common language runtime, the power and flexibility of that entire platform is available to Web application developers. The .NET Framework class library, Messaging, and Data Access solutions are all seamlessly accessible from the Web. ASP.NET is also language-independent, so you can choose the language that best applies to your application or partition your application across many languages. Further, common language runtime interoperability guarantees that your existing investment in COM-based development is preserved when migrating to ASP.NET.

- **Simplicity.** ASP.NET makes it easy to perform common tasks, from simple form submission and client authentication to deployment and site configuration. For example, the ASP.NET page framework allows you to build user interfaces that cleanly separate application logic from presentation code and to handle events in a simple, Visual Basic - like forms processing model. Additionally, the common language runtime simplifies development, with managed code services such as automatic reference counting and garbage collection.

- **Manageability.** ASP.NET employs a text-based, hierarchical configuration system, which simplifies applying settings to your server environment and Web applications. Because configuration information is stored as plain text, new settings may be applied without the aid of local administration tools. This "zero local administration" philosophy extends to deploying ASP.NET Framework applications as well. An ASP.NET Framework application is deployed to a server simply by copying the necessary files to the server. No server restart is required, even to deploy or replace running compiled code.

- **Scalability and Availability.** ASP.NET has been designed with scalability in mind, with

features specifically tailored to improve performance in clustered and multiprocessor environments. Further, processes are closely monitored and managed by the ASP.NET runtime, so that if one misbehaves (leaks, deadlocks), a new process can be created in its place, which helps keep your application constantly available to handle requests.

• **Customizability and Extensibility.** ASP.NET delivers a well-factored architecture that allows developers to "plug-in" their code at the appropriate level. In fact, it is possible to extend or replace any subcomponent of the ASP.NET runtime with your own custom-written component. Implementing custom authentication or state services has never been easier.

• **Security.** With built in Windows authentication and per-application configuration, you can be assured that your applications are secure.

Language Support

The Microsoft .NET Platform currently offers built-in support for three languages: C#, Visual Basic, and Java Script.

What Is Asp.Net Web Forms?

The ASP.NET Web Forms page framework is a scalable common language runtime programming model that can be used on the server to dynamically generate Web pages.

Intended as a logical evolution of ASP (ASP.NET provides syntax compatibility with existing pages), the ASP.NET Web Forms framework has been specifically designed to address a number of key deficiencies in the previous model. In particular, it provides:

• The ability to create and use reusable UI controls that can encapsulate common functionality and thus reduce the amount of code that a page developer has to write.

• The ability for developers to cleanly structure their page logic in an orderly fashion (not "spaghetti code").

• The ability for development tools to provide strong WYSIWYG design support for pages (existing ASP code is opaque to tools).

Introduction To Asp.Net Server Controls

In addition to (or instead of) using <% %> code blocks to program dynamic content, ASP.NET page developers can use ASP.NET server controls to program Web pages. Server controls are declared within an .aspx file using custom tags or intrinsic HTML tags that contain a **runat="server"** attributes value. Intrinsic HTML tags are handled by one of the controls in the

System.Web.UI.HtmlControls namespace. Any tag that doesn't explicitly map to one of the controls is assigned the type of **System.Web.UI.HtmlControls.HtmlGenericControl**.

Server controls automatically maintain any client-entered values between round trips to the server. This control state is not stored on the server (it is instead stored within an **<input type="hidden">** form field that is round-tripped between requests). Note also that no client-side script is required.

In addition to supporting standard HTML input controls, ASP.NET enables developers to utilize richer custom controls on their pages. For example, the following sample demonstrates how the **<asp:adrotator>** control can be used to dynamically display rotating ads on a page.

1. ASP.NET Web Forms provide an easy and powerful way to build dynamic Web UI.
2. ASP.NET Web Forms pages can target any browser client (there are no script library or cookie requirements).
3. ASP.NET Web Forms pages provide syntax compatibility with existing ASP pages.
4. ASP.NET server controls provide an easy way to encapsulate common functionality.
5. ASP.NET ships with 45 built-in server controls. Developers can also use controls built by third parties.
6. ASP.NET server controls can automatically project both uplevel and downlevel HTML.
7. ASP.NET templates provide an easy way to customize the look and feel of list server controls.
8. ASP.NET validation controls provide an easy way to do declarative client or server data validation.

2.6.3 C#.NET

Ado.Net Overview

ADO.NET is an evolution of the ADO data access model that directly addresses user requirements for developing scalable applications. It was designed specifically for the web with scalability, statelessness, and XML in mind.

ADO.NET uses some ADO objects, such as the **Connection** and **Command** objects, and also introduces new objects. Key new ADO.NET objects include the **Dataset**, **Data Reader**, and **Data Adapter**.

The important distinction between this evolved stage of ADO.NET and previous data architectures is that there exists an object -- the **DataSet** -- that is separate and distinct from any data stores. Because of that, the **DataSet** functions as a standalone entity. You can think of the DataSet as an always disconnected recordset that knows nothing about the source or destination of the data it contains. Inside a **DataSet**, much like in a database, there are tables, columns, relationships, constraints, views, and so forth.

The following sections will introduce you to some objects that have evolved, and some that are new. These objects are:

- **Connections**. For connection to and managing transactions against a database.
- **Commands**. For issuing SQL commands against a database.
- **DataReaders**. For reading a forward-only stream of data records from a SQL Server data source.
- **DataSet**. For storing, Remoting and programming against flat data, XML data and relational data.
- **DataAdapters**. For pushing data into a **DataSet**, and reconciling data against a database.

Connections

Connections are used to 'talk to' databases, and are represented by provider-specific classes such as **SqlConnection**. Commands travel over connections and resultsets are returned in the form of streams which can be read by a **DataReader** object, or pushed into a **DataSet** object.

Commands

Commands contain the information that is submitted to a database, and are represented by provider-specific classes such as **SqlCommand**. A command can be a stored procedure call, an UPDATE statement, or a statement that returns results. You can also use input and output parameters, and return values as part of your command syntax. The example below shows how to issue an INSERT statement against the **Northwind** database.

DataReaders

The **Data Reader** object is somewhat synonymous with a read-only/forward-only cursor over data. The **DataReader** API supports flat as well as hierarchical data. A **DataReader** object is returned after executing a command against a database. The format of the returned **DataReader** object is different from a recordset. For example, you might use the **DataReader** to show the results

of a search list in a web page.

Datasets and dataadapters

DataSets

The **Dataset** object is similar to the ADO **Recordset** object, but more powerful, and with one other important distinction: the **DataSet** is always disconnected. The **DataSet** object represents a cache of data, with database-like structures such as tables, columns, relationships, and constraints. However, though a **DataSet** can and does behave much like a database, it is important to remember that **DataSet** objects do not interact directly with databases, or other source data. This allows the developer to work with a programming model that is always consistent, regardless of where the source data resides. Data coming from a database, an XML file, from code, or user input can all be placed into **DataSet** objects. Then, as changes are made to the **DataSet** they can be tracked and verified before updating the source data. The **GetChanges** method of the **DataSet** object actually creates a second **DatSet** that contains only the changes to the data. This **DataSet** is then used by a **DataAdapter** (or other objects) to update the original data source.

Dataadapters (Oledb/Sql)

The **DataAdapter** object works as a bridge between the **DataSet** and the source data. Using the provider-specific **SqlDataAdapter** (along with its associated **SqlCommand** and **SqlConnection**) can increase overall performance when working with a Microsoft SQL Server databases. For other OLE DB-supported databases, you would use the **OleDbDataAdapter** object and its associated **OleDbCommand** and **OleDbConnection** objects.

1. ADO.NET is the next evolution of ADO for the .Net Framework.
2. ADO.NET was created with n-Tier, statelessness and XML in the forefront. Two new objects, the **DataSet** and **DataAdapter**, are provided for these scenarios.
3. ADO.NET can be used to get data from a stream, or to store data in a cache for updates.
4. There is a lot more information about ADO.NET in the documentation.
5. Remember, you can execute a command directly against the database in order to do inserts, updates, and deletes. You don't need to first put data into a **DataSet** in order to insert, update, or delete it.

2.6.4 Sql Server -2008

A database management, or DBMS, gives the user access to their data and helps them transform the data into information. Such database management systems include dBase, paradox, IMS, SQL Server and SQL Server. These systems allow users to create, update and extract information from their database.

A database is a structured collection of data. Data refers to the characteristics of people, things and events. SQL Server stores each data item in its own fields. In SQL Server, the fields relating to a particular person, thing or event are bundled together to form a single complete unit of data, called a record (it can also be referred to as raw or an occurrence). Each record is made up of a number of fields. No two fields in a record can have the same field name.

During an SQL Server Database design project, the analysis of your business needs identifies all the fields or attributes of interest. If your business needs change over time, you define any additional fields or change the definition of existing fields.

Sql Server Tables

SQL Server stores records relating to each other in a table. Different tables are created for the various groups of information. Related tables are grouped together to form a database.

Primary Key

Every table in SQL Server has a field or a combination of fields that uniquely identifies each record in the table. The Unique identifier is called the Primary Key, or simply the Key. The primary key provides the means to distinguish one record from all other in a table. It allows the user and the database system to identify, locate and refer to one particular record in the database.

Relational Database

Sometimes all the information of interest to a business operation can be stored in one table. SQL Server makes it very easy to link the data in multiple tables. Matching an employee to the department in which they work is one example. This is what makes SQL Server a relational database management system, or RDBMS. It stores data in two or more tables and enables you to define relationships between the table and enables you to define relationships between the tables.

Foreign Key

When a field is one table matches the primary key of another field is referred to as a foreign key. A foreign key is a field or a group of fields in one table whose values match those of the primary key of another table.

Referential Integrity

Not only does SQL Server allow you to link multiple tables, it also maintains consistency between them. Ensuring that the data among related tables is correctly matched is referred to as maintaining referential integrity.

Data Abstraction

A major purpose of a database system is to provide users with an abstract view of the data. This system hides certain details of how the data is stored and maintained. Data abstraction is divided into three levels.

Physical level: This is the lowest level of abstraction at which one describes how the data are actually stored.

Conceptual Level: At this level of database abstraction all the attributed and what data are actually stored is described and entries and relationship among them.

View level: This is the highest level of abstraction at which one describes only part of the database.

Advantages Of RDBMS

- Redundancy can be avoided
- Inconsistency can be eliminated
- Data can be Shared
- Standards can be enforced
- Security restrictions ca be applied
- Integrity can be maintained
- Conflicting requirements can be balanced
- Data independence can be achieved.

Disadvantages Of DBMS

A significant disadvantage of the DBMS system is cost. In addition to the cost of purchasing of developing the software, the hardware has to be upgraded to allow for the extensive programs and the workspace required for their execution and storage. While centralization reduces duplication, the lack of duplication requires that the database be adequately backed up so that in case of failure the data can be recovered.

Features Of SQL Server (RDBMS)

SQL SERVER is one of the leading database management systems (DBMS) because it is the only Database that meets the uncompromising requirements of today's most demanding information systems. From complex decision support systems (DSS) to the most rigorous online transaction processing (OLTP) application, even application that require simultaneous DSS and OLTP access to the same critical data, SQL Server leads the industry in both performance and capability.

SQL SERVER with transactions processing option offers two features which contribute to very high level of transaction processing throughput, which are

- The row level lock manager

Enterprise Wide Data Sharing

The unrivalled portability and connectivity of the SQL SERVER DBMS enables all the systems in the organization to be linked into a singular, integrated computing resource.

Portability

SQL SERVER is fully portable to more than 80 distinct hardware and operating systems platforms, including UNIX, MSDOS, OS/2, Macintosh and dozens of proprietary platforms. This portability gives complete freedom to choose the database server platform that meets the system requirements.

Open Systems

SQL SERVER offers a leading implementation of industry –standard SQL. SQL Server's open architecture integrates SQL SERVER and non –SQL SERVER DBMS with industry's most comprehensive collection of tools, application, and third party software products SQL Server's Open architecture provides transparent access to data from other relational database and even non-relational database.

Distributed Data Sharing

SQL Server's networking and distributed database capabilities to access data stored on remote server with the same ease as if the information was stored on a single local computer. A single SQL statement can access data at multiple sites. You can store data where system requirements such as performance, security or availability dictate.

Unmatched Performance

The most advanced architecture in the industry allows the SQL SERVER DBMS to deliver unmatched performance.

Sophisticated Concurrency Control

Real World applications demand access to critical data. With most database Systems application becomes "contention bound" – which performance is limited not by the CPU power or by disk I/O, but user waiting on one another for data access. SQL Server employs full, unrestricted row-level locking and contention free queries to minimize and in many cases entirely eliminates contention wait times.

No I/O Bottlenecks

SQL Server's fast commit groups commit and deferred write technologies dramatically reduce disk I/O bottlenecks. While some database write whole data block to disk at commit time, SQL Server commits transactions with at most sequential log file on disk at commit time, On high throughput systems, one sequential writes typically group commit multiple transactions.

SYSTEM ANALYSIS

3.1 Existing System

We consider applications where the original sensitive data cannot be perturbed. Perturbation is a very useful technique where the data is modified and made "less sensitive" before being handed to agents. However, in some cases it is important not to alter the original distributor's data.

Traditionally, leakage detection is handled by watermarking, e.g., a unique code is embedded in each distributed copy. If that copy is later discover ed in the hands of an unauthorized party, the leaker can be identified. Watermarks can be very useful in some cases, but again, involve some modification of the original data. Furthermore, watermarks can sometimes be destroyed if the data recipient is malicious.

3.2 Problem Statement

A *distributor* owns a set $T=\{t_1,\ldots,t_m\}$of valuable data objects. The distributor wants to share some of the objects with a set of agents $U_1,U_2,\ldots U_n$, but does not wish the objects be *leaked* to other third parties. The objects in T could be of any type and size, e.g., they could be tuples in a relation, or relations in a database. An agent Ui receives a subset of objects, determined either by a sample request or an explicit request:

1. Sample request
2. Explicit request

An agent U_i recieve a subset of objects $R_i \subseteq T$, determined either by a sample request or an explict request:

--> Sample request R_i= SAMPLE*(T,m_i):* Any subset of m_i records from T can be given to U_i

--> Explicit request R_i= EXPLICT(T,*$cond_i$):* Agent U_i receives *all T* objects that satisfy $cond_i$

Although we do not discuss it here, our model can be easily extended to requests for a sample of objects that satisfy a condition (e.g., an agent wants any 100 california customer records). Also note that we do not concern ourselves with the randomness of a sample. (We assume that if a random sample is required, there are enough T records so that the to-be-presented object selection schemes can pick random records T).

Assumption 1. *For all t, t' ∈ S such that t ≠ t', the provenance of t is independent of the provenance of t'.*

The term provenance in this assumption statement refers to the source of a value *t* that appears in the leaked set. The source can be any of the agents who have *t* in their sets or the target itself (guessing).

To simplify our formulas, the following assumption states that joint events have a negligible probability. As we argue in the example below, this assumption gives us more conservative estimates for the guilt of agents, which is consistent with our goals.

Assumption 2. *An object $t \in S$ can only be obtained by the target in one of two ways:*

• *A single agent U_i leaked t from its own R_i set;*

• *The target* guessed *(or obtained through other means) t without the help of any of the n agents In other words, for all $t \in S$, the event that the target guesses t and the events that agent U_i (i = 1, . . . , n) leaks object t are disjoint.*

Before we present the general formula for computing the probability *Pr{Gi|S}* that an agent U_i is guilty, we provide a simple example. Assume that the distributor set *T*, the agent sets *R*'s and the target set *S* are:

T= {t_1, t_2,t_3}, R_1={t_1, t_2}, R_2={t_1,t_3}, S= {t_1, t_2,t_3}.

In this case, all three of the distributor's objects have been leaked and appear in *S*. Let us first consider how the target may have obtained object t_1, which was given to both agents. From Assumption 2, the target either guessed *t*1 or one of U_1 or U_2 leaked it. We know that the probability of the former event is *p*, so assuming that probability that each of the two agents leaked *t*1 is the same we have the following cases:

- The target guessed *t*1 with probability *p*;
- Agent U_1 leaked t_1 to *S* with probability (1 − *p*)/2;
- Agent U_2 leaked t_1 to *S* with probability (1 − *p*)/2

Similarly, we find that agent U_1 leaked t_2 to *S* with probability 1 − *p* since he is the only agent that has *t*2. Given these values, the probability that agent U_1 is not guilty, namely that U_1 did not leak either object is:

$$Pr\{G'_1 \mid S\} - (1 - (1 - p) / 2) \times (1 - (1 - p)) \quad \text{---------- (1)}$$

and the probability that *U*1 is guilty is: $Pr\{G_1 \mid S\} = 1 - Pr\{G_1\}$

Note that if Assumption 2 did not hold, our analysis would be more complex because we would need to consider joint events, e.g., the target guesses t_1 and at the same time one or two agents leak the value. In our simplified analysis we say that an agent is not guilty when the object can be guessed, regardless of whether the agent leaked the value. Since we are "not counting" instances when an agent leaks information, the simplified analysis yields conservative values (smaller probabilities).

In the general case (with our assumptions), to find the probability that an agent *Ui* is guilty given a set *S*, first we compute the probability that he leaks a single object *t* to *S*. To compute this we define the set of agents $Vt = \{U_i | t \in R_i\}$ that have *t* in their data sets. Then using Assumption 2 and known probability *p*, we have:

Pr{some agent leaked t to S} = 1- p ------- (3)

Assuming that all agents that belong to *Vt* can leak *t* to *S* with equal probability and using

Assumption 2 we obtain:

$$Pr\{U_i \text{ leaked } t \text{ to } S\} = \begin{cases} \frac{1-p}{|V_t|}, & \text{if } U_i \in V_t \\ 0, & \text{otherwise} \end{cases} \qquad (4)$$

Given that agent U_i is guilty if he leaks at least one value to *S*, with Assumption 1 and Equation 4 we can compute the probability $Pr\{G_i|S\}$ that agent U_i is guilty:

$$Pr\{G_i|S\} = 1 - \prod_{t \in S \cap R_i} \left(1 - \frac{1-p}{|V_t|}\right) \qquad (5)$$

3.3 Proposed System

After giving a set of objects to agents, the distributor discovers some of those same objects in an unauthorized place. At this point the distributor can assess the likelihood that the leaked data came from one or more agents, as opposed to having been independently gathered by other means. If the distributor sees "enough evidence" that an agent leaked data, he may stop doing business with him, or may initiate legal proceedings. In this project we develop a model for assessing the "guilt" of agents. We also present algorithms for distributing objects to agents, in a way that improves our chances of identifying a leaker. Finally, we also consider the option of adding "fake" objects to the distributed set. Such objects do not correspond to real entities but appear. If it turns out an agent

was given one or more fake objects that were leaked, then the distributor can be more confident that agent was guilty.

3.3.1 Advantages

➔ As web threats get ever more sophisticated businesses need to take a more proactive stance if they are to successfully defend against them.

➔ M86 Security offers a range of web security solutions and its latest Secure Web Gateways have a number of unique capabilities designed to target and eliminate the most devious of attacks.

➔ The appliance defaults to an explicit proxy so you can change your client browser proxy settings using group policies or PAC scripts.

➔ The SWG 5000 can also operate transparently but it will still be necessary to redirect traffic to the appliance for scanning.

3.4 Feasibility Study

Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

- Technical Feasibility
- Operational Feasibility
- Economical Feasibility

3.4.1 Technical Feasibility

The technical issue usually raised during the feasibility stage of the investigation includes the following:

- Does the necessary technology exist to do what is suggested?
- Do the proposed equipment have the technical capacity to hold the data required to use the new system?

- Will the proposed system provide adequate response to inquiries, regardless of the number or location of users?
- Can the system be upgraded if developed?
- Are there technical guarantees of accuracy, reliability, ease of access and data security?

Earlier no system existed to cater to the needs of 'Secure Infrastructure Implementation System'. The current system developed is technically feasible. It is a web based user interface for audit workflow at NIC-CSD. Thus it provides an easy access to the users. The database's purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified.

Therefore, it provides the technical guarantee of accuracy, reliability and security. The software and hard requirements for the development of this project are not many and are already available in-house at NIC or are available as free as open source. The work for the project is done with the current equipment and existing software technology. Necessary bandwidth exists for providing a fast feedback to the users irrespective of the number of users using the system.

3.3.2 Operational Feasibility

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization's operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. Some of the important issues raised are to test the operational feasibility of a project includes the following: -

- Is there sufficient support for the management from the users?
- Will the system be used and work properly if it is being developed and implemented?
- Will there be any resistance from the user that will undermine the possible application benefits?

This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits.

The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

3.4.3. Economical Feasibility

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs.

The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, There is nominal expenditure and economical feasibility for certain.

3.5 Algorithm

1. Evaluation of Explicit Data Request Algorithms

In the first place, the goal of these experiments was to see whether fake objects in the distributed data sets yield significant improvement in our chances of detecting a guilty agent. In the second place, we wanted to evaluate our e-optimal algorithm relative to a random allocation.

2. Evaluation of Sample Data Request Algorithms

With sample data requests agents are not interested in particular objects. Hence, object sharing is not explicitly defined by their requests. The distributor is "forced" to allocate certain objects to multiple agents only if the number of requested objects exceeds the number of objects in set T. The more data objects the agents request in total, the more recipients on average an object has; and the more objects are shared among different agents, the more difficult it is to detect a guilty agent.

➔ **Finding the probablity of the agent that has leaked the data**

T – {t¬1, t2, t3,¬¬ t4,¬ ……t¬n¬ }. ------- Total no of Objects.

U – {u1, u2, u3, u4……u1}. ------- Total no of Agents.

Ri – {r1, r2, r3,…….rn} --------- Total no of Objects with agents.

Ri ¬C T. i.e., R is subset of T and is always less than T.

S – { s1, s2, s3,…….sn } ---------- Total no of leaked bjects.

Gi – ------- Guilty Agent.

The probability to find the guitly agent is as followed.

Case Study:

Consider the following, in the developed web application if the employees download the articles.

Total no of articles ---------- 12;

Articles downloaded by Emp1 are ---------- 3 and their id's are a1, a2,a3;

Articles downloaded by Emp 2 are ---------- 3 and their id's are a1, a3 ,a5.

Articles donwloaded by Emp3 are ---------- 1 and is id is a4.

So, we have

T = 12;

U = 3;

R = 7;

S = total no of leaked articles which is the input.

Now if administrator finds article with id a4 is leaked then the probability to identify the guilty agent as Emp3 is 100%. And in the second case if the leaked article is a1 the ad-min can suspect the employees, emp1 and emp2. By using the above formula we can find the probability for any number of cases and with better accuracy.

Input: Here input is the object which is to be detected.

Output: The output is probability of the agent who has leaked the data.

Data Allocation Problems.

In this we deal with the data allocation problems. We discuss about the techniques how intelligent they can be distributed to the agents so that the probability of finiding the guilty agent is maximum.

Fake Objects

Understanding the concept of fake objects. Consider the following two scenarios to understand the concept of fake objects. The scenarios are as which are discussed in the base project.

In most cases, individual objects are perturbed, e.g., by adding random noise to sensitive salaries, or adding a watermark to an image. In our case, we are perturbing the set of distributor objects by adding fake elements.

The distributed data objects are medical records and the agents are hospitals. In this case, even small modifications to the records of actual patients may be undesirable. However, the addition of some fake medical records may be acceptable, since no patient matches these records, and hence no one will ever be treated based on fake records.

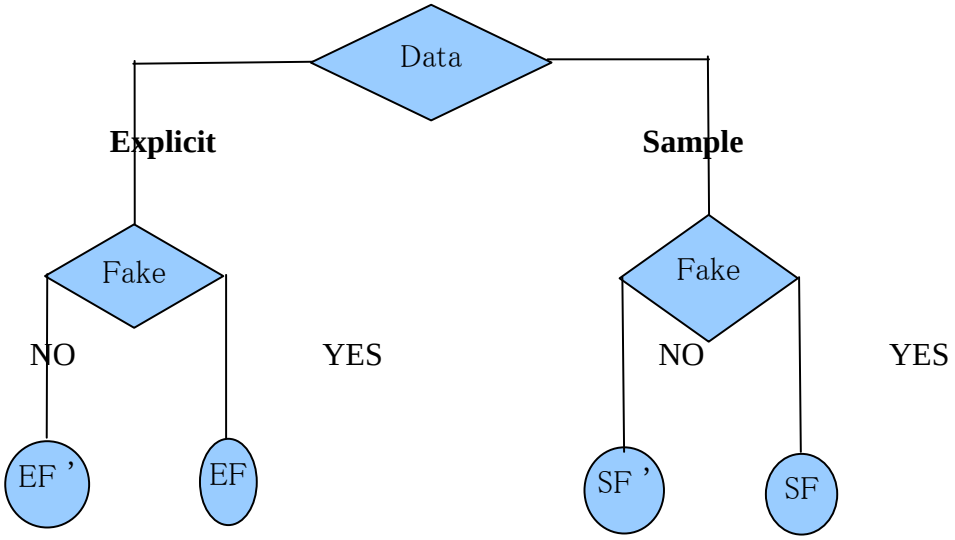


Figure 3.5.1 Sample flow chart of Data Leakage

Here, we model the creation of a fake object for agent Ui as a black-box function CREATEFAKEOBJECT(Ri; Fi; condi) that takes as input the set of all objects Ri, the subset of fake objects Fi that Ui has received so far and condi, and returns a new fake object. This function needs condi to produce a valid object that satisfies Ui's condition. Set Ri is needed as input so that the created fake object is not only valid but also indistinguishable from other real objects.

For example, the creation function of a fake payroll record that includes an employee rank and a salary attribute may take into account the distribution of employee ranks, the distribution of salaries as well as the correlation between the two attributes. Ensuring that key statistics do not change by the introduction of fake objects is important if the agents will be using such statistics in their work. Finally, function CREATEFAKEOBJECT() has to be aware of the fake objects Fi added so far, again to ensure proper statistics. The distributor can also use function CREATEFAKEOBJECT() when it wants to send the same fake object to a set of agents.

In this case, the function arguments are the union of the Ri and Fi tables respectively, and the intersection of the conditions $cond_i$'s. Although we do not deal with the implementation of CREATEFAKEOBJECT() we note that there are two main design options. The function can either produce a fake object on demand every time it is called, or it can return an appropriate object from a pool of objects created in advance.

As a conclusion it is made clear that fake objects can be anything depending on the distributor and these project doesnot deal with creation of fake objects, but CREATEFAKEOBJECT() method is defined in order to distribute the objects to the agents with fake object or without fake object depending on the request.

To allocate the data to the users the following four conditions are defined.

(a) EF (b) EF (c) SF (d)SF

Where

E – Explicit Request S – Sample Request

F – Fake Object F – Without Fake Object

Types Of Leakages :

Type of information leaked	Percentage
Confidential information	15%
Intellectual property	4 %
Health records	8%
Customer data	73%

System Requirements Specification

4.1 Introduction

In the course of doing business, sometimes sensitive data must be handed over to supposedly trusted third parties. For example, a hospital may give patient records to researchers who will devise new treatments. Similarly, a company may have partnerships with other companies that require sharing customer data. Another enterprise may outsource its data processing, so data must be given to various other companies. We call the owner of the data the distributor and the supposedly trusted third parties the agents. Our goal is to detect when the distributor's sensitive data has been leaked by agents, and if possible to identify the agent that leaked the data. We consider applications where the original sensitive data cannot be perturbed. Perturbation is a very useful technique where the data is modified and made "less sensitive" before being handed to agents. For example, one can add random noise to certain attributes, or one can replace exact values by ranges.

However, in some cases it is important not to alter the original distributor's data. For example, if an outsourcer is doing our payroll, he must have the exact salary and customer bank account numbers. If medical researchers will be treating patients (as opposed to simply computing statistics), they may need accurate data for the patients. Traditionally, leakage detection is handled by watermarking, e.g., a unique code is embedded in each distributed copy. If that copy is later discovered in the hands of an unauthorized party, the leaker can be identified. Watermarks can be very useful in some cases, but again, involve some modification of the original data.

4.2 Purpose

Furthermore, watermarks can sometimes be destroyed if the data recipient is malicious. In this project we study unobtrusive techniques for detecting leakage of a set of objects or records. Specifically, we study the following scenario: After giving a set of objects to agents, the distributor discovers some of those same objects in an unauthorized place. (For example, the data may be found on a web site, or may be obtained through a legal discovery process.) At this point the distributor can assess the likelihood that the leaked data came from one or more agents, as opposed to having been independently gathered by other means. Using an analogy with cookies stolen from a cookie jar, if we catch Freddie with a single cookie, he can argue that a friend gave him the cookie. But if we catch Freddie with 5 cookies, it will be much harder for him to argue that his hands were not in the cookie jar. If the distributor sees "enough evidence" that an agent leaked data, he may stop doing business with him, or may initiate legal proceedings.

4.3 Functional Requirements

Functional Requirements refer to very important system requirements in a software engineering process (or at micro level, a sub part of requirement engineering) such as technical specifications, system design parameters and guidelines, data manipulation, data processing and calculation modules etc.

Functional Requirements are in contrast to other software design requirements referred to as Non-Functional Requirements which are primarily based on parameters of system performance, software quality attributes, reliability and security, cost, constraints in design/implementation etc. The key goal of determining "functional requirements" in a software product design and implementation is to capture the required behaviour of a software system in terms of functionality and the technology implementation of the business processes.

The *Functional Requirement* document (also called Functional Specifications or Functional Requirement Specifications), defines the capabilities and functions that a System must be able to perform successfully.

Functional Requirements should include:

1. Descriptions of data to be entered into the system
2. Descriptions of operations performed by each screen
3. Descriptions of work-flows performed by the system
4. Descriptions of system reports or other outputs
5. Who can enter the data into the system?
6. How the system meets applicable regulatory requirements

The functional specification is designed to be read by a general audience. Readers should understand the system, but no particular technical knowledge should be required to understand the document.

Examples of Functional Requirements

Functional requirements should include functions performed by specific screens, outlines of work-flows performed by the system and other business or compliance requirements the system must meet.

Interface requirements

➔ Field accepts numeric data entry

➔ Field only accepts dates before the current date

➔ Screen can print on-screen data to the printer

Business Requirements

➔ Data must be entered before a request can approved

➔ Clicking the Approve Button moves the request to the Approval Workflow

➔ All personnel using the system will be trained according to internal training strategies

Regulatory/Compliance Requirements

➔ The database will have a functional audit trail

➔ The system will limit access to authorized users

➔ The spread sheet can secure data with electronic signatures

Security Requirements

➔ Member of the Data Entry group can enter requests but not approve or delete requests .

➔ Members of the Managers group can enter or approve a request, but not delete requests .

➔ Members of the Administrators group cannot enter or approve requests, but can delete requests

The functional specification describes what the system must do; how the system does it is described in the Design Specification. If a User Requirement Specification was written, all requirements outlined in the user requirement specification should be addressed in the functional requirements.

4.3 Non Functional Requirements

All the other requirements which do not form a part of the above specification are categorized as Non-Functional Requirements.

A system may be required to present the user with a display of the number of records in a database. This is a functional requirement.

How up-to-date this number needs to be is a non-functional requirement. If the number

needs to be updated in real time, the system architects must ensure that the system is capable of updating the displayed record count within an acceptably short interval of the number of records changing.

Sufficient network bandwidth may also be a non-functional requirement of a system.

Accessibility is a general term used to describe the degree to which a product, device, service, or environment is accessible by as many people as possible. Accessibility can be viewed as the "ability to access" and possible benefit of some system or entity. Accessibility is often used to focus on people with disabilities and their right of access to the system.

Availability is the degree to which a system, subsystem, or equipment is operable and in a committable state at the start of a mission, when the mission is called for at an unknown, *i.e.*, a random, time. Simply put, availability is the proportion of time a system is in a functioning condition.

Expressed mathematically, **availability** is 1 minus the unavailability.

A **backup** or the process of **backing up** refers to making copies of data so that these additional copies may be used to *restore* the original after a data loss event. These additional copies are typically called "backups."

Certification refers to the confirmation of certain characteristics of an object, system, or organization. This confirmation is often, but not always, provided by some form of external review, education, or assessment

Compliance is the act of adhering to, and demonstrating adherence to, a standard or regulation.

Configuration management (CM) is a field that focuses on establishing and maintaining consistency of a system's or product's performance and its functional and physical attributes with its requirements, design, and operational information throughout its life.

Documentation may refer to the process of providing evidence ("to document something") or to the communicable material used to provide such documentation (i.e. a document). Documentation may also (seldom) refer to tools aiming at identifying documents or to the field of study devoted to the study of documents and bibliographies

Disaster recovery is the process, policies and procedures related to preparing for recovery or continuation of technology infrastructure critical to an organization after a natural or human-induced disaster.

Extensibility (sometimes confused with forward compatibility) is a system design principle where the implementation takes into consideration future growth. It is a systemic measure of the ability to extend a system and the level of effort required to implement the extension. Extensions can be through the addition of new functionality or through modification of existing functionality. The central theme is to provide for change while minimizing impact to existing system functions.

Interoperability is a property referring to the ability of diverse systems and organizations to work together (inter-operate). The term is often used in a technical systems engineering sense, or alternatively in a broad sense, taking into account social, political, and organizational factors that impact system to system performance.

Maintenance is the ease with which a software product can be modified in order to:

- correct defects
- meet new requirements
- make future maintenance easier, or cope with a changed environment;

Open source describes practices in production and development that promote access to the end product's source materials—typically, their source code

Operability is the ability to keep equipment, a system or a whole industrial installation in a safe and reliable functioning condition, according to pre-defined operational requirements.

In a computing systems environment with multiple systems this includes the ability of products, systems and business processes to work together to accomplish a common task.

Computer performance is characterized by the amount of useful work accomplished by a computer system compared to the time and resources used.

Depending on the context, good computer performance may involve one or more of the following:

- Short response time for a given piece of work

- High throughput (rate of processing work)
- Low utilization of computing resource(s)
- High availability of the computing system or application
- Fast (or highly compact) data compression and decompression
- High bandwidth / short data transmission time

Price in economics and business is the result of an exchange and from that trade we assign a numerical monetary value to a good, service or asset

Portability is one of the key concepts of high-level programming. Portability is the software-code base feature to be able to reuse the existing code instead of creating new code when moving software from an environment to another. When one is targeting several platforms with the same application, portability is the key issue for development cost reduction.

Quality: The common element of the business definitions is that the quality of a product or service refers to the perception of the degree to which the product or service meets the customer's expectations. Quality has no specific meaning unless related to a specific function and/or object. Quality is a perceptual, conditional and somewhat subjective attribute.

Reliability may be defined in several ways:

- The idea that something is fit for purpose with respect to time;
- The capacity of a device or system to perform as designed;
- The resistance to failure of a device or system;
- The ability of a device or system to perform a required function under stated conditions for a specified period of time;
- The probability that a functional unit will perform its required function for a specified interval under stated conditions.
- The ability of something to "fail well" (fail without catastrophic consequences

Resilience is the ability to provide and maintain an acceptable level of service in the face of faults and challenges to normal operation.

These services include:

- supporting distributed processing

- supporting networked storage
- maintaining service of communication services such as
  - video conferencing
  - instant messaging
  - online collaboration
- access to applications and data as needed

Response time perceived by the end user is the interval between

(a) The instant at which an operator at a terminal enters a request for a response from a computer and

(b) The instant at which the first character of the response is received at a terminal.

In a data system, the system response time is the interval between the receipt of the end of transmission of an inquiry message and the beginning of the transmission of a response message to the station originating the inquiry.

Robustness is the quality of being able to withstand stresses, pressures, or changes in procedure or circumstance. A system or design may be said to be "robust" if it is capable of coping well with variations (sometimes unpredictable variations) in its operating environment with minimal damage, alteration or loss of functionality.

The concept of **scalability** applies to technology and business settings. Regardless of the setting, the base concept is consistent - The ability for a business or technology to accept increased volume without impacting the system.In telecommunications and software engineering, **scalability** is a desirable property of a system, a network, or a process, which indicates its ability to either handle growing amounts of work in a graceful manner or to be readily enlarged.

Security is the degree of protection against danger, loss, and criminals.

Security has to be compared and contrasted with other related concepts: Safety, continuity, reliability. The key difference between security and reliability is that security must take into account the actions of people attempting to cause destruction.

Security as a state or condition is *resistance to harm.* From an objective perspective, it is a structure's actual (conceptual and never fully knowable) degree of resistance to harm.

Stability - it means much of the objects will be stable over time and will not need changes.

Safety is the state of being "safe", the condition of being protected against physical, social, spiritual, financial, political, emotional, occupational, psychological, educational or other types or consequences of failure, damage, error, accidents, harm or any other event which could be considered non-desirable. This can take the form of being protected from the event or from exposure to something that causes health or economical losses. It can include protection of people or of possessions

Supportability (also known as **serviceability**) is one of the aspects of RASU (Reliability, Availability, Serviceability, and Usability)). It refers to the ability of technical support personnel to install, configure, and monitor products, identify exceptions or faults, debug or isolate faults to root cause analysis, and provide hardware or software maintenance in pursuit of solving a problem and restoring the product into service. Incorporating serviceability facilitating features typically results in more efficient product maintenance and reduces operational costs and maintains business continuity.

Testability, a property applying to an empirical hypothesis, involves two components: (1) the logical property that is variously described as contingency, defeasibility, which means that counter examples to the hypothesis are logically possible, and (2) the practical feasibility of observing a reproducible series of such counter examples if they do exist. In short it refers to the capability of an equipment or system to be tested

Usability is a term used to denote the ease with which people can employ a particular tool or other human-made object in order to achieve a particular goal. In human-computer interaction and computer science, usability often refers to the elegance and clarity with which the interaction with a computer program or a web site is designed.

4.5 Hardware Requirements

PROCESSOR : PENTIUM IV 2.4 GHz

RAM : 1 GB

MONITOR : 15”

HARD DISK : 160 GB

CDDRIVE : 52X

KEYBOARD : STANDARD 102 KEYS

MOUSE : 3 BUTTONS

4.6 Software Requirements

IDE : VISUAL STUDIO 2008

DATABAS : SQL SERVER2005/2008

CODING LANGUAGE : C#.NET

FRONT END : VB.NET, ASP.NET

OPERATING SYSTEM : WINDOWS XP

System Design

5.1 System Specification

The purpose of the design phase is to plan a solution of the problem specified by the requirement document. This phase is the first step in moving from the problem domain to the solution domain. In other words, starting with what is needed, design takes us toward how to satisfy the needs. The design of a system is perhaps the most critical factor affection the quality of the software; the output of this phase is the design document.

System Design also called top-level design aims to identify the modules that should be in the system, the specifications of these modules, and how they interact with each other to produce the desired results. At the end of the system design all the major data structures, file formats, output formats, and the major modules in the system and their specifications are decided.

5.2 System Components

The set of primary components that are identified by the ERD are

- ➔ Data object
- ➔ Relationships
- ➔ Attributes
- ➔ Various types of indicators.

The primary purpose of the ERD is to represent data objects and their relationships.

5.3 UML DIAGRAMS

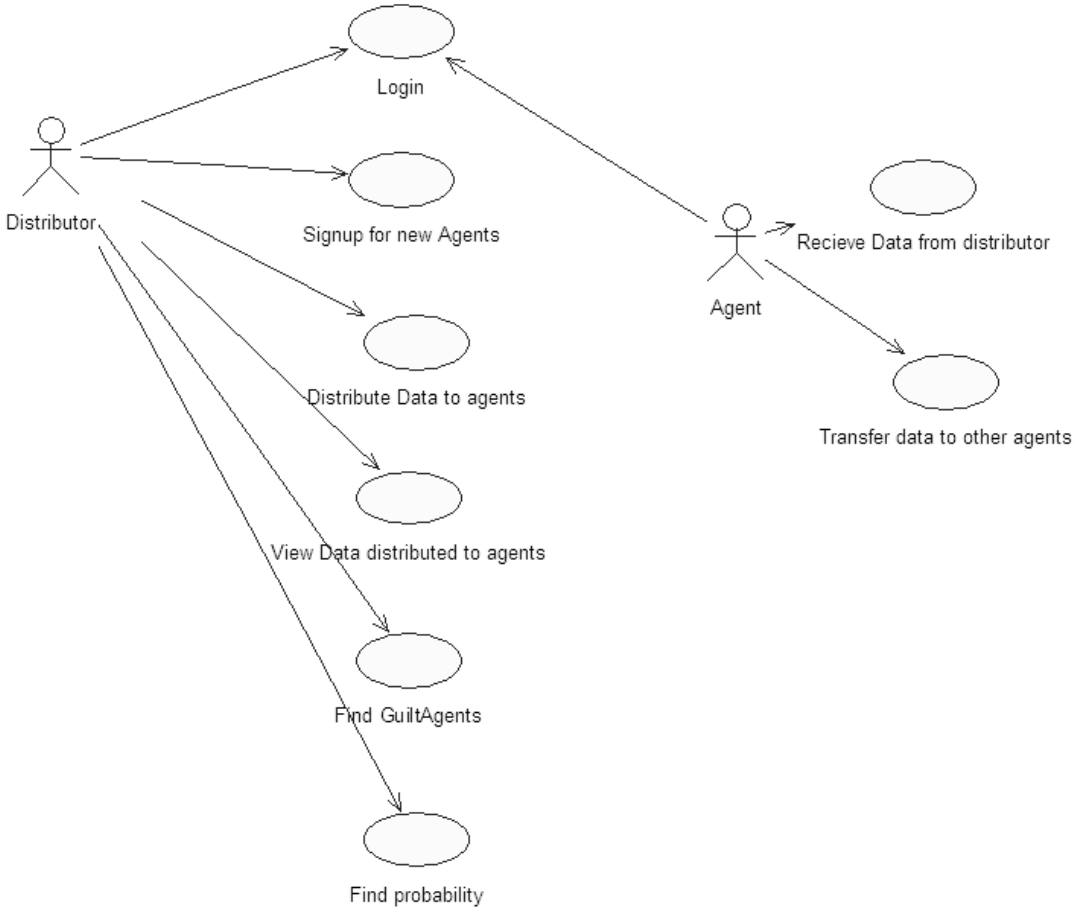


Figure 5.3.1 Overall Use Diagram for MFDPFA

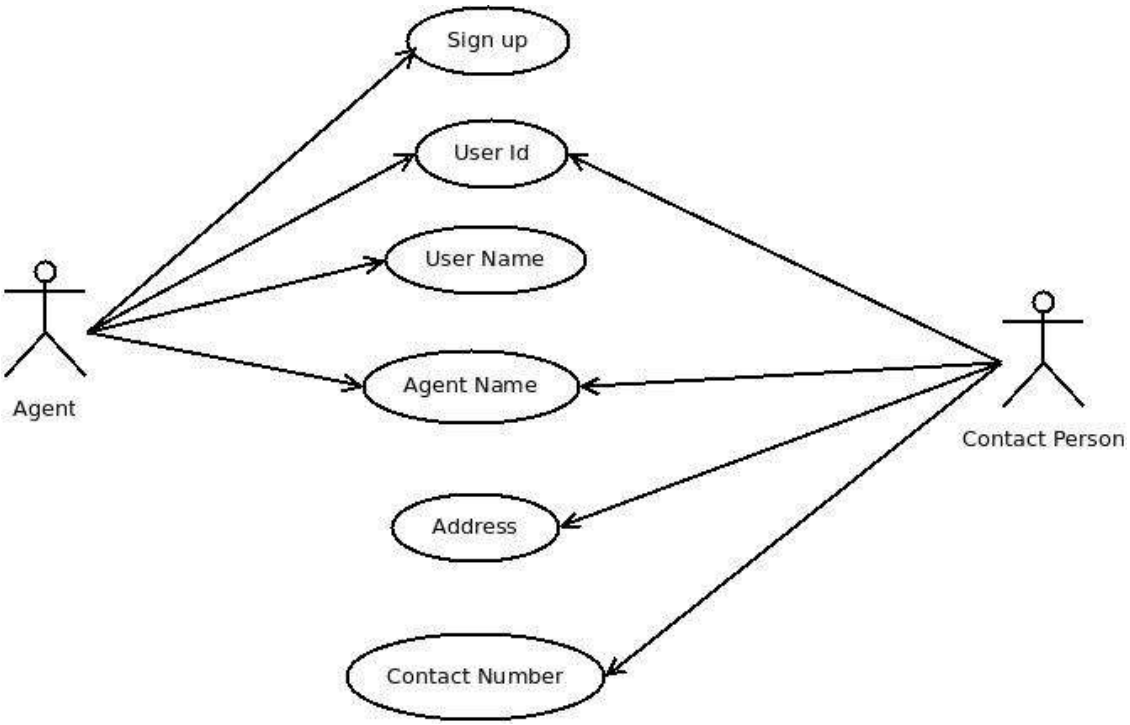


Figure 5.3.2 Use Diagram for Agent login

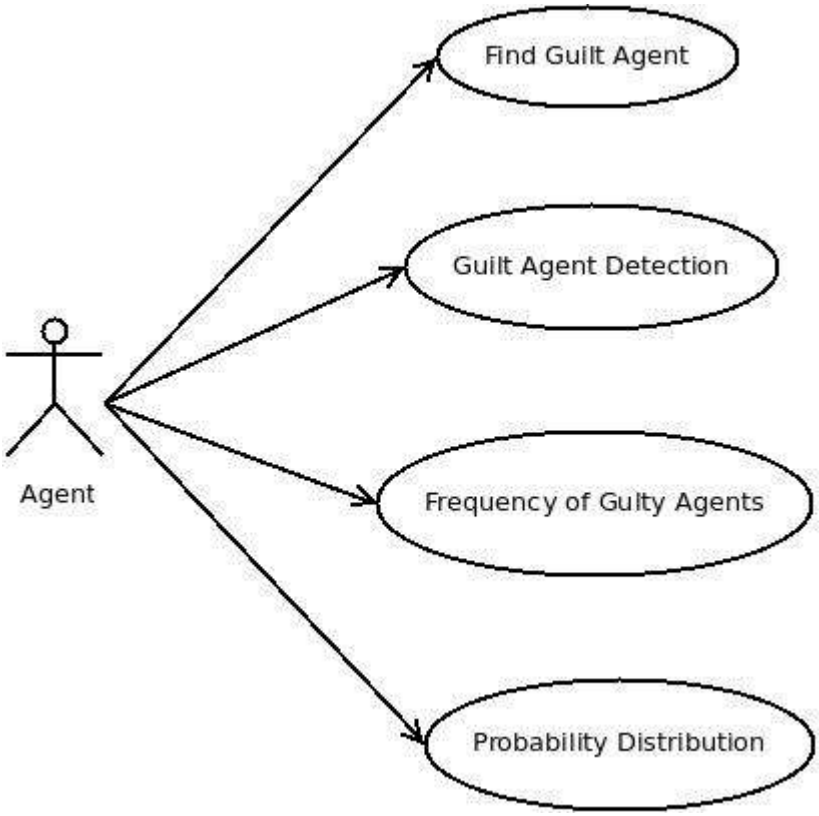


Figure 5.3.3 Use Diagram for Agent

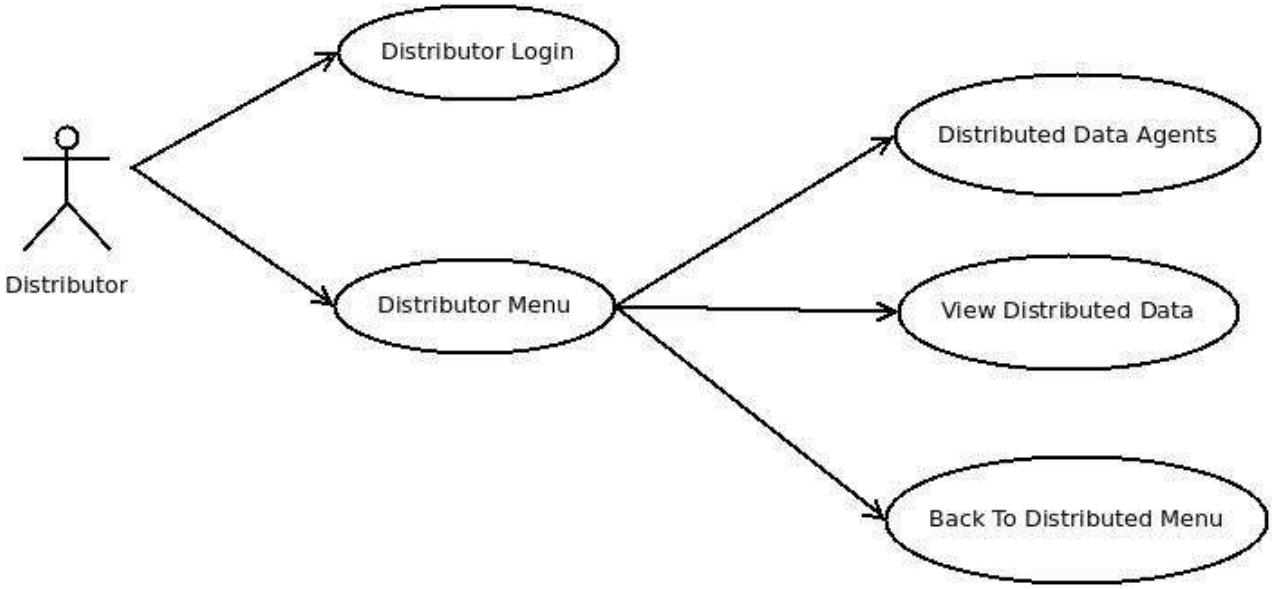


Figure 5.3.4 Use Diagram for Distributor

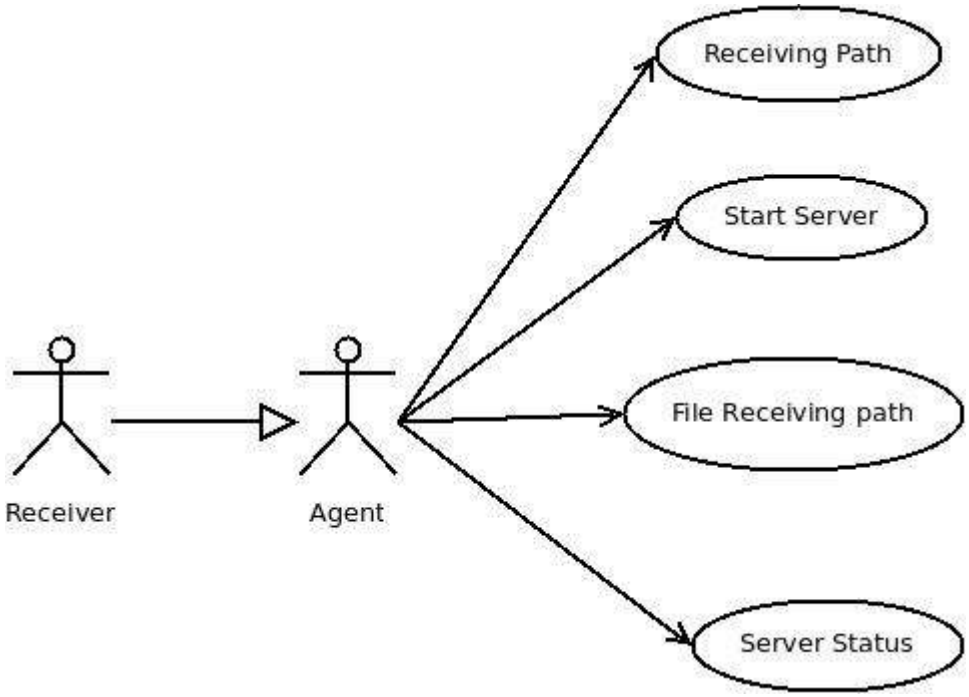


Figure 5.3.5 Use Diagram for Receiver through Agent

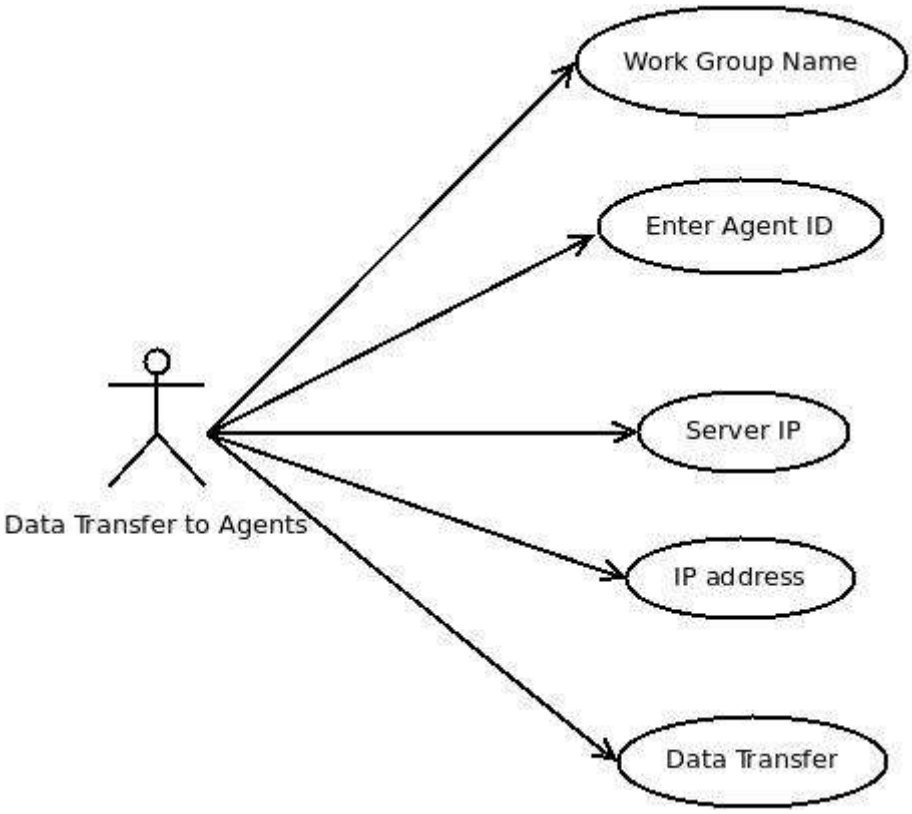


Figure 5.3.6 Use Diagram for Data Transfer to Agents

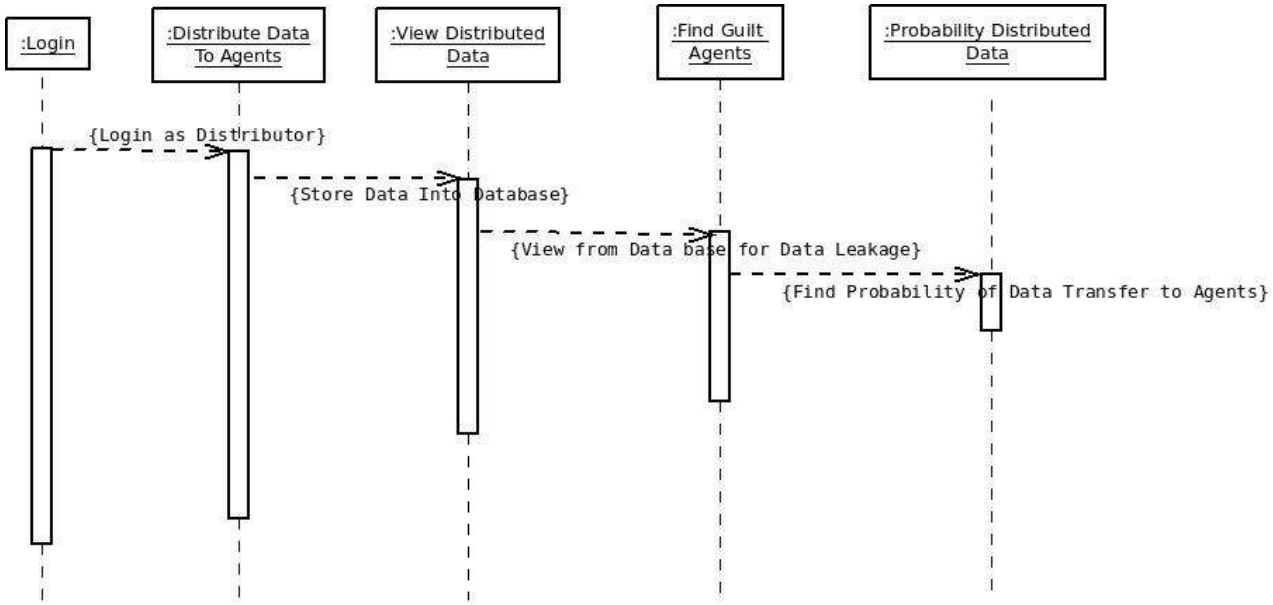


Figure 5.3.7 Sequence Diagram for Data Transfer from Distributor to Agents

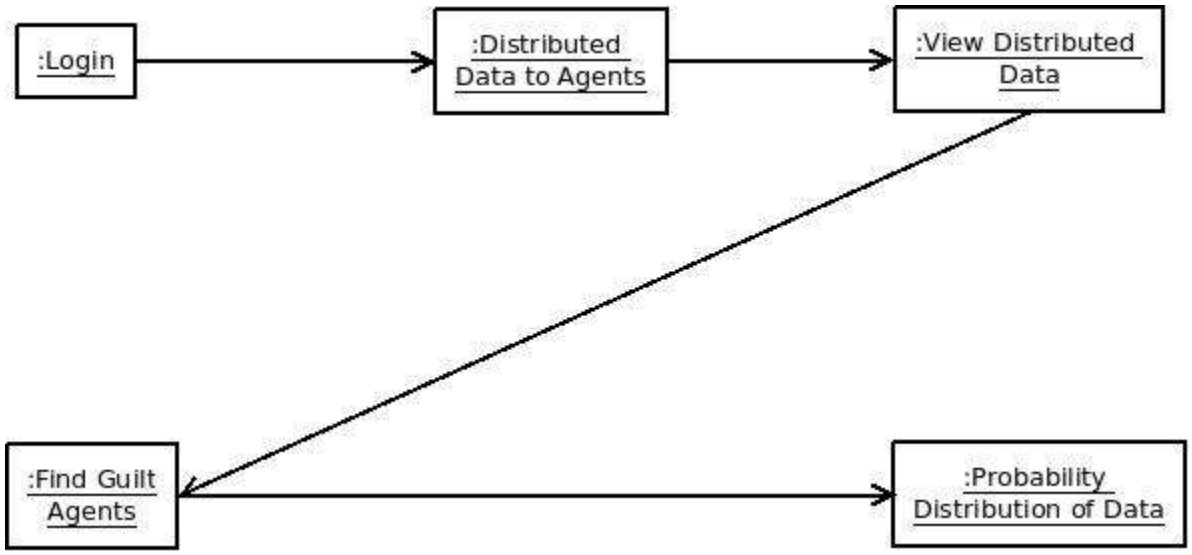


Figure 5.3.8 Collaboration Diagram for Data Transfer from Distributor to Agents

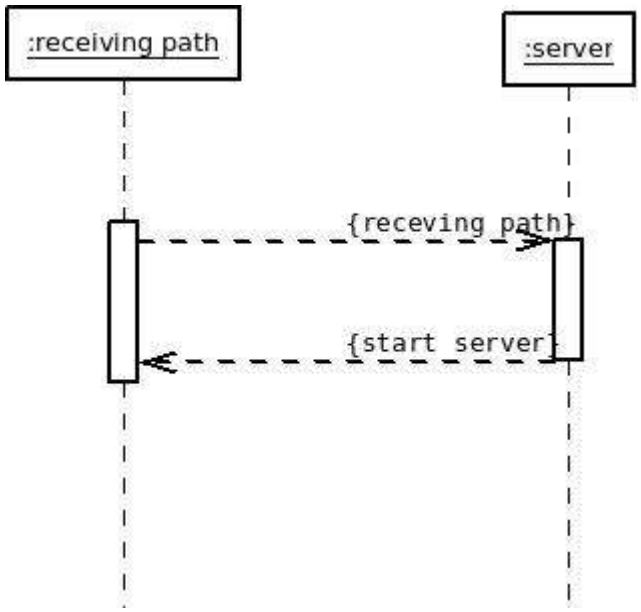


Figure 5.3.9 Sequence Diagram for Receiver path

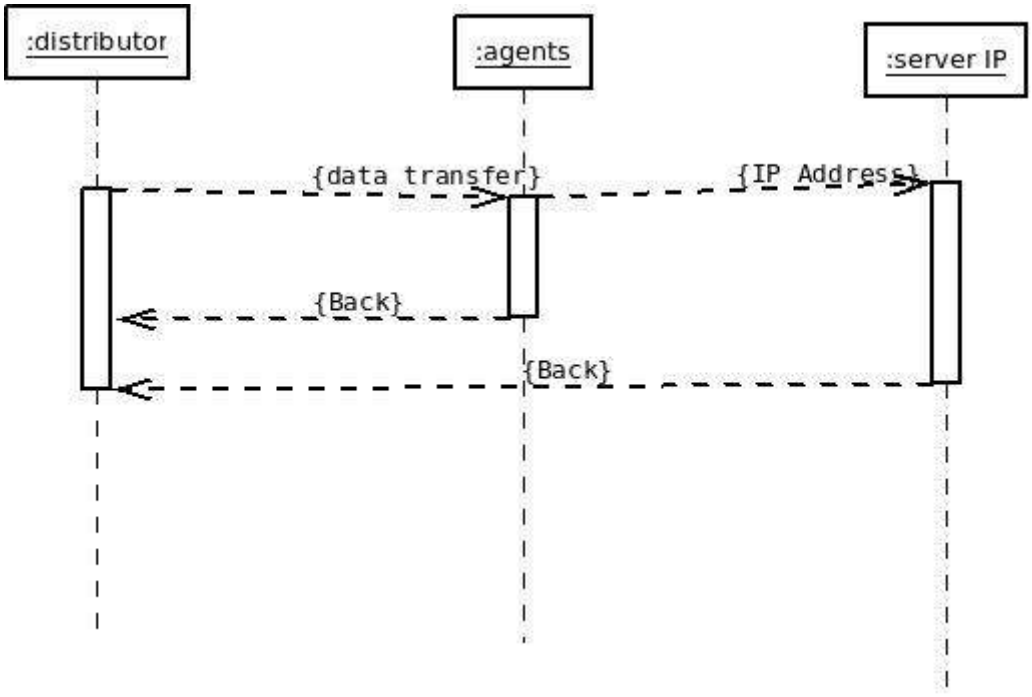


Figure 5.3.10 Sequence Diagram for Data Transfer to Agents

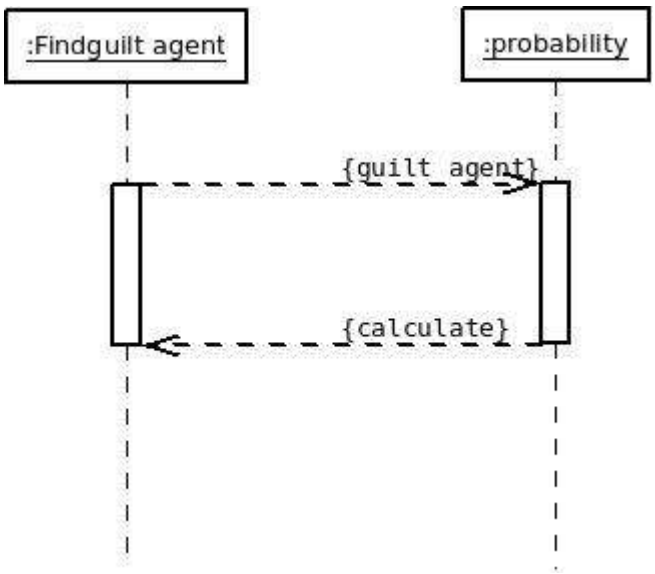


Figure 5.3.11 Sequence Diagram for Agents Guilt Model

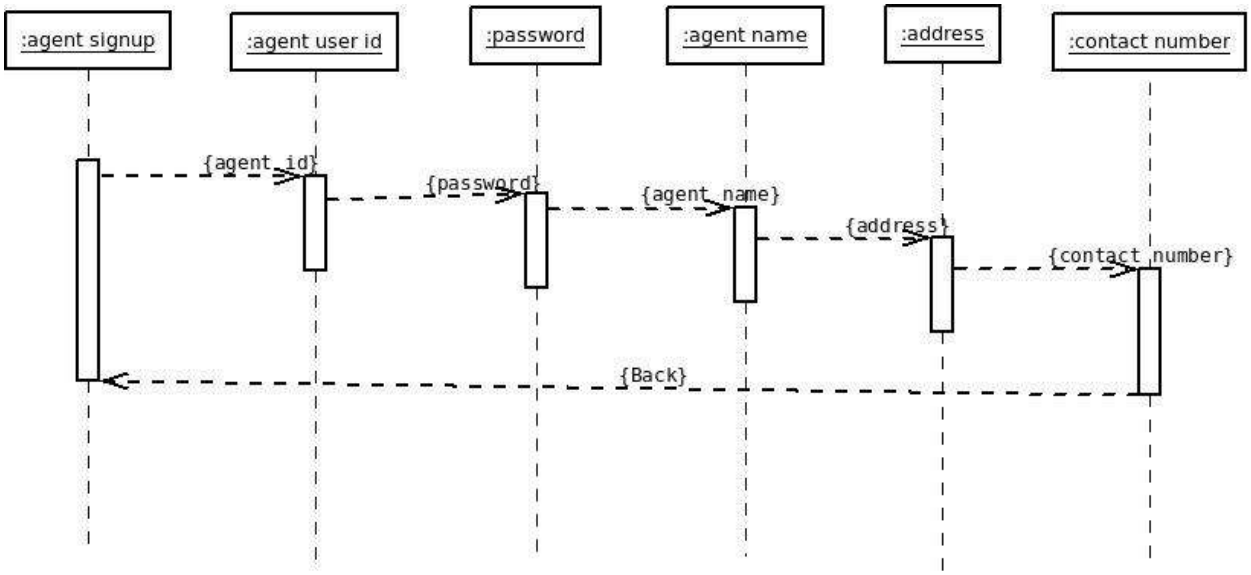


Figure 5.3.12 Sequence Diagram for Agent SignUp

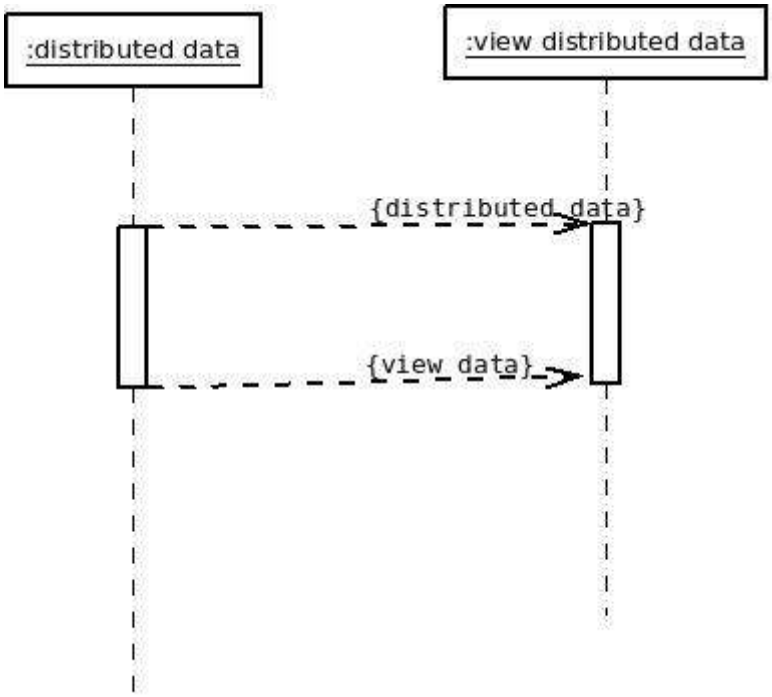


Figure 5.3.13 Sequence Diagram for Distributed to data & view data

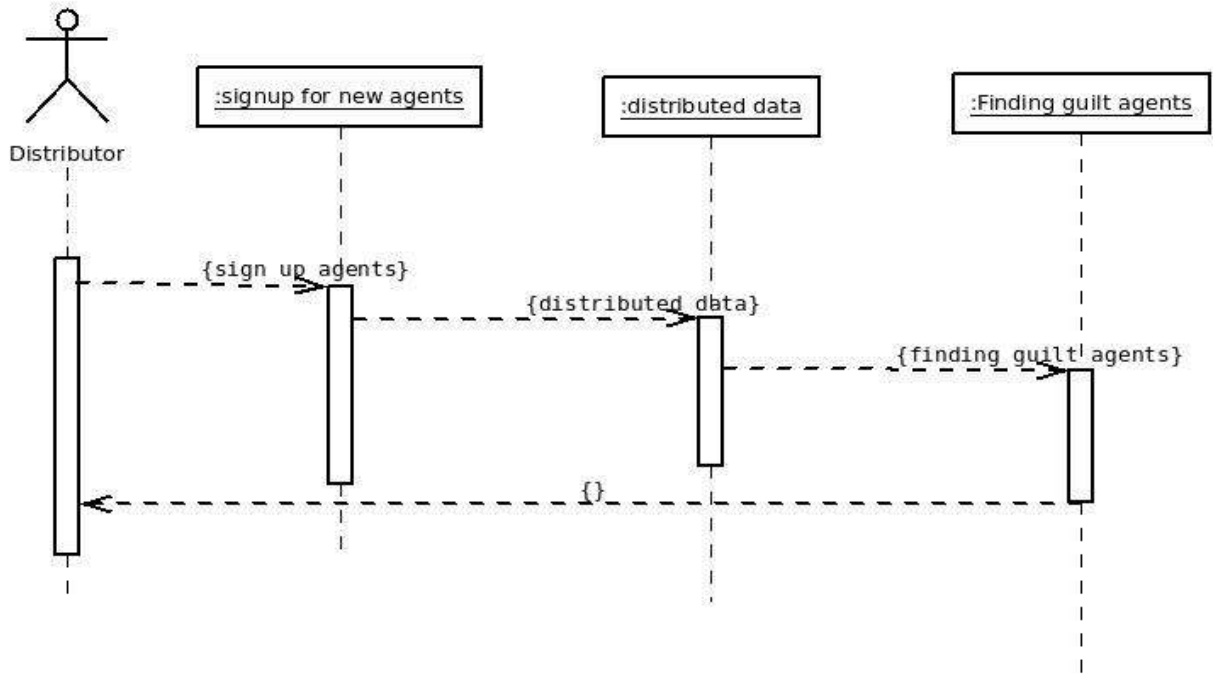
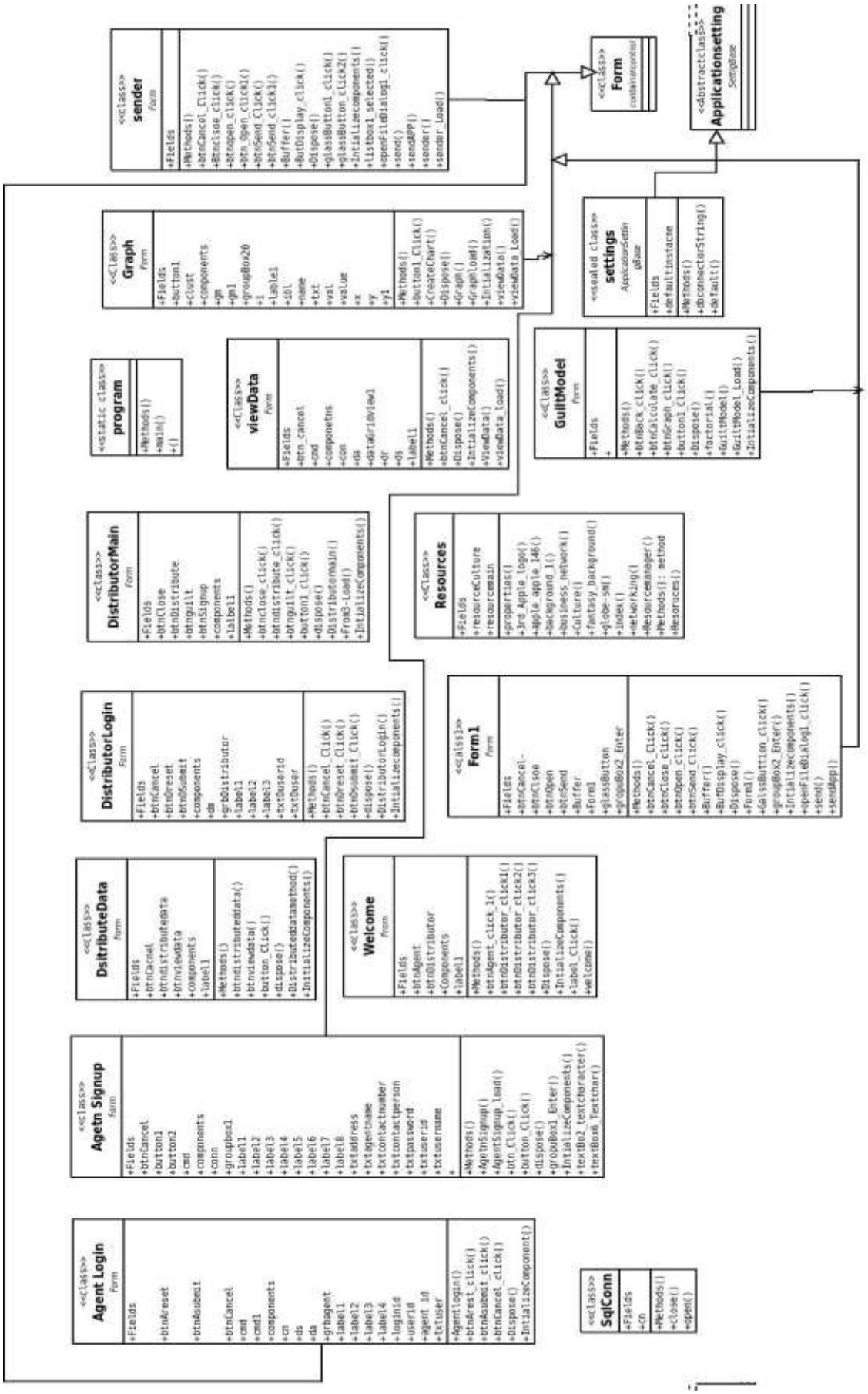


Figure 5.3.14 Sequence Diagram for Distributor Functions


sender
Graph
program
viewData
DistributorMain
DistributorLogin
DsitributeData
Welcome
Agetn Signup
Agent Login
SqlConn
Resources
GuiltModel
settings
Form
Applicationsetting
Form1

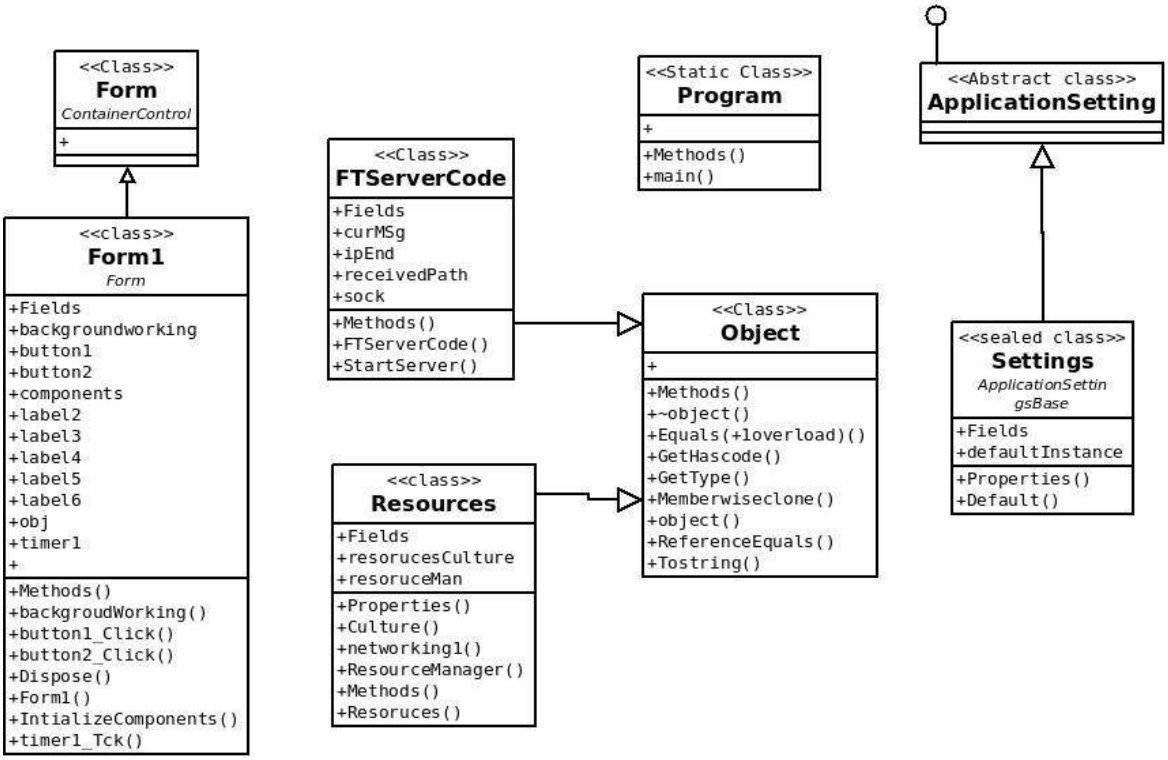


Figure 5.3.16 Class diagram for Receiver through agent

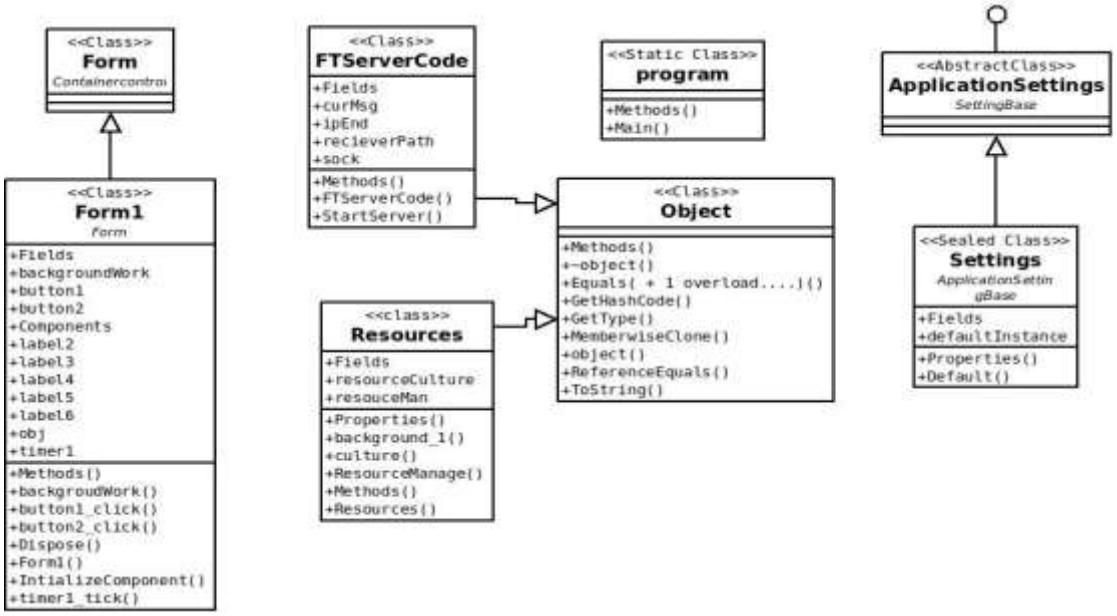


Figure 5.3.17 Class diagram for Receiver through Distributor

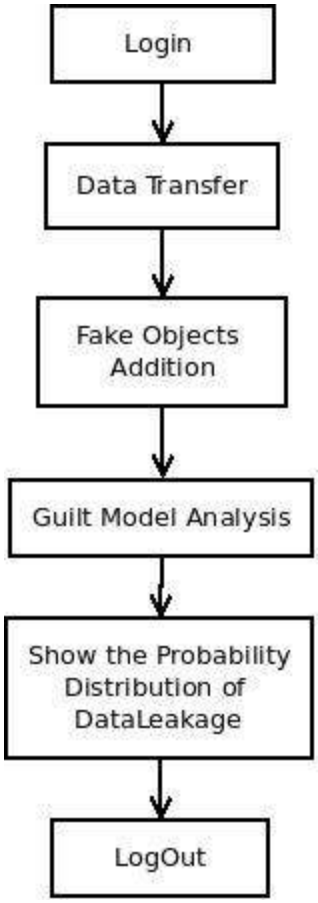


Figure 5.3.18 Flowchart for Data Transfer from Distributor to Agents

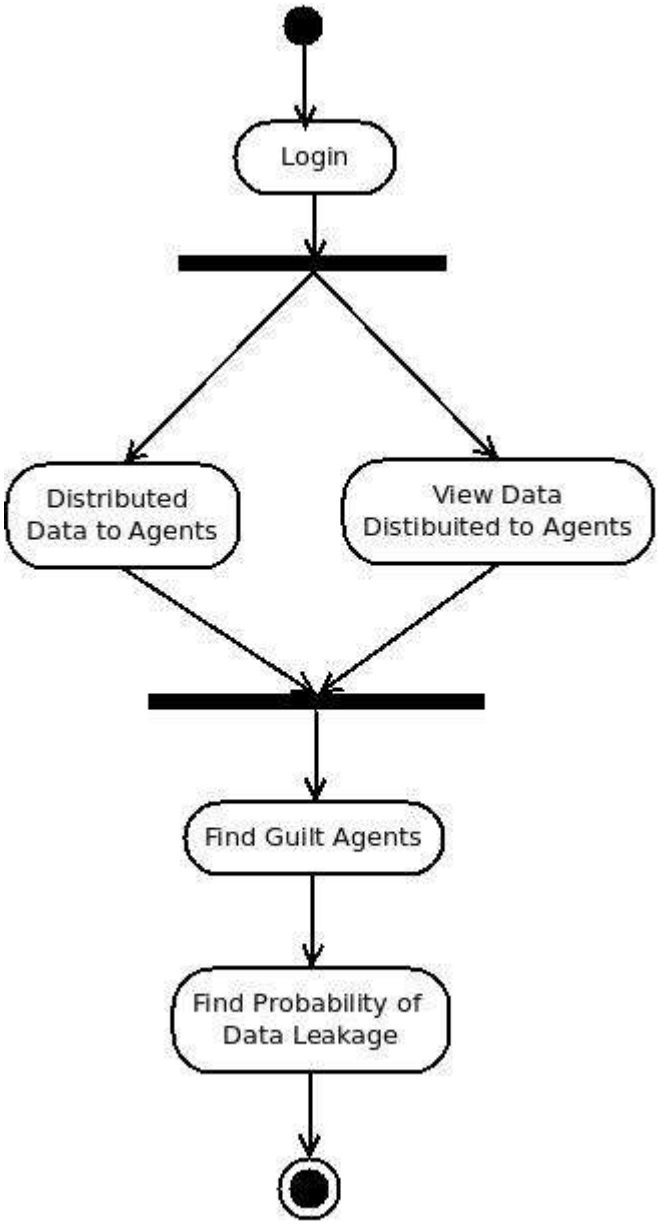


Figure 5.3.19 Activity Diagram for Data Transfer from Distributor to Agents

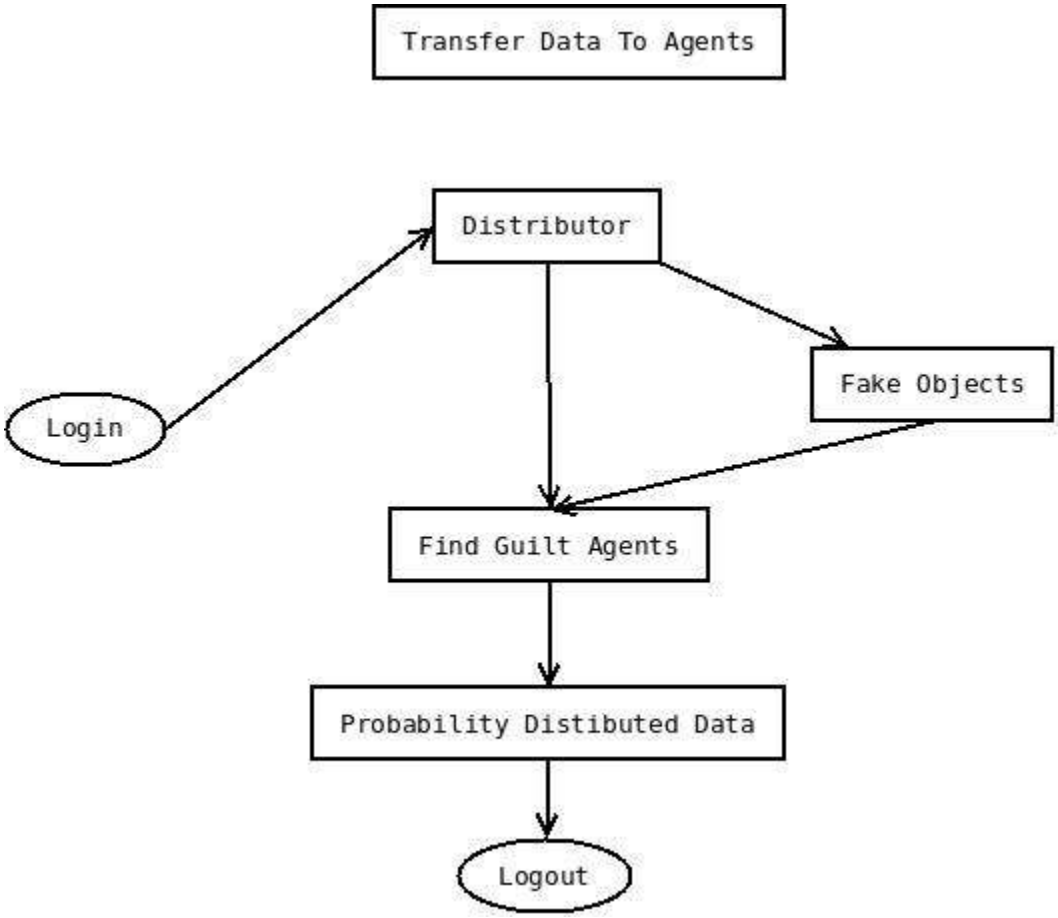


Figure 5.3.20 Flow Chart for Data Transfer from Distributor to Agents

GuiltProb			
	Column Name	Datatype	Allow Nulls
	auserid	varchar(10)	Yes
	Account	int	yes

Table 5.3.1 Guitlt Probability

Guiltagetns			
	Column Name	Datatype	Allow Nulls
	Aid	varchar(10)	Yes
	Adatapath	varchar	Yes
	Cnt	int	Yes

Table 5.3.2 Table for Guilt Agent

Asignup			
	Column Name	Datatype	Allow Nulls
Primary	auserid	varchar(10)	No
	ausername	varchar(10)	No
	apwd	varchar(10)	No
	aname	char(20)	No
	aadress	varchar(10)	No
	acontactno	varchar(10)	No
	acontactperson	char(20)	No

Table 5.3.3 Table for Agent Signup

Agentrecord			
	Column Name	Datatype	Allow Nulls
	aid	varchar(10)	Yes
	Aip	varchar(20)	Yes
	adatapath	varchar(100)	Yes

Table 5.3.4 Table for Agent record

Implementation

6.1 Sample code

Code for Welcome

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Configuration;
namespace dataleakage
{
   public partial class Welcome : Form
   {
      public Welcome()
      {
         InitializeComponent();
      }

      private void btnDistributor_Click(object sender, EventArgs e)
      {

         //this.Hide();
      }

      private void btnAgent_Click(object sender, EventArgs e)
      {
```

```
        }


        private void btnDistributor_Click_1(object sender, EventArgs e)
        {
            DistributorLogin dl = new DistributorLogin();
            dl.Show();
            this.Hide();
        }

        private void label1_Click(object sender, EventArgs e)
        {

        }

        private void btnAgent_Click_1(object sender, EventArgs e)
        {
            AgentLogin al = new AgentLogin();
            al.Show();
            this.Hide();
        }
    }
}
```

Distributor code

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
//using System.Linq;
```

```
using System.Text;
using System.Windows.Forms;
using System.Configuration;
namespace dataleakage
{
   public partial class DistributorMain : Form
   {
            public DistributorMain()
      {
         InitializeComponent();
      }

      private void Form3_Load(object sender, EventArgs e)
      {
         DistributorLogin f1 = new DistributorLogin();
         f1.Close();
      }

      private void button1_Click(object sender, EventArgs e)
      {
         AgentSignup asign=new AgentSignup();
         asign.Show();
         this.Hide();
      }
       private void btnclose_Click(object sender, EventArgs e)
      {
         Welcome w = new Welcome();
         w.Show();
         this.Hide();
      }
```

```
        private void btndistribute_Click(object sender, EventArgs e)
        {
            DistributeDataMenu dm = new DistributeDataMenu();
            dm.Show();
            this.Hide();
        }

        private void btnguilt_Click(object sender, EventArgs e)
        {
            GuiltModel gm = new GuiltModel();
            gm.Show();
            this.Hide();
        }
    }
}
```

SQL Connection Code

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Data.SqlClient;
using System.Configuration;

namespace dataleakage
{
    class SqlConn
    {
        public SqlConnection cn;
        public void open()
```

```
        {
                cn = new
SqlConnection(ConfigurationSettings.AppSettings["dataleakage"].ToString());
            cn.Open();
        }
        public void close()
        {
            cn.Close();
        }
    }
}
```

ASP Code for DLD

```
<?xml version="1.0" encoding="utf-8"?>
<Project DefaultTargets="Build" xmlns="http://schemas.microsoft.com/developer/msbuild/2003"
ToolsVersion="4.0">
  <PropertyGroup>
    <Configuration Condition=" '$(Configuration)' == " ">Debug</Configuration>
    <Platform Condition=" '$(Platform)' == " ">AnyCPU</Platform>
    <ProductVersion>9.0.30729</ProductVersion>
    <SchemaVersion>2.0</SchemaVersion>
    <ProjectGuid>{BA69E414-EAF7-4D89-BB21-227E52FCC348}</ProjectGuid>
    <OutputType>WinExe</OutputType>
    <AppDesignerFolder>Properties</AppDesignerFolder>
    <RootNamespace>Trust</RootNamespace>
    <AssemblyName>Trust</AssemblyName>
    <FileUpgradeFlags>
    </FileUpgradeFlags>
    <UpgradeBackupLocation>
    </UpgradeBackupLocation>
    <OldToolsVersion>3.5</OldToolsVersion>
```

```
  <IsWebBootstrapper>true</IsWebBootstrapper>
  <TargetFrameworkVersion>v2.0</TargetFrameworkVersion>
  <PublishUrl>http://localhost/Trust/</PublishUrl>
  <Install>true</Install>
  <InstallFrom>Web</InstallFrom>
  <UpdateEnabled>true</UpdateEnabled>
  <UpdateMode>Foreground</UpdateMode>
  <UpdateInterval>7</UpdateInterval>
  <UpdateIntervalUnits>Days</UpdateIntervalUnits>
  <UpdatePeriodically>false</UpdatePeriodically>
  <UpdateRequired>false</UpdateRequired>
  <MapFileExtensions>true</MapFileExtensions>
  <ApplicationRevision>0</ApplicationRevision>
  <ApplicationVersion>1.0.0.%2a</ApplicationVersion>
  <UseApplicationTrust>false</UseApplicationTrust>
  <BootstrapperEnabled>true</BootstrapperEnabled>
</PropertyGroup>
<PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|AnyCPU' ">
  <DebugSymbols>true</DebugSymbols>
  <DebugType>full</DebugType>
  <Optimize>false</Optimize>
  <OutputPath>bin\Debug\</OutputPath>
  <DefineConstants>DEBUG;TRACE</DefineConstants>
  <ErrorReport>prompt</ErrorReport>
  <WarningLevel>4</WarningLevel>
  <CodeAnalysisRuleSet>AllRules.ruleset</CodeAnalysisRuleSet>
</PropertyGroup>
<PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Release|AnyCPU' ">
  <DebugType>pdbonly</DebugType>
  <Optimize>true</Optimize>
  <OutputPath>bin\Release\</OutputPath>
```

```
    <DefineConstants>TRACE</DefineConstants>
    <ErrorReport>prompt</ErrorReport>
    <WarningLevel>4</WarningLevel>
    <CodeAnalysisRuleSet>AllRules.ruleset</CodeAnalysisRuleSet>
  </PropertyGroup>
  <ItemGroup>
    <Reference Include="GlassButton, Version=1.3.2.29932, Culture=neutral,
PublicKeyToken=2e983e6e44d23a4f, processorArchitecture=MSIL">
      <SpecificVersion>False</SpecificVersion>
      <HintPath>E:\ITDNW01\glassbutton\GlassButton (demo)\GlassButton.dll</HintPath>
    </Reference>
    <Reference Include="System" />
    <Reference Include="System.Data" />
    <Reference Include="System.Deployment" />
    <Reference Include="System.DirectoryServices" />
    <Reference Include="System.Drawing" />
    <Reference Include="System.Windows.Forms" />
    <Reference Include="System.Xml" />
    <Reference Include="ZedGraph, Version=5.1.5.28844, Culture=neutral,
PublicKeyToken=02a83cbd123fcd60, processorArchitecture=MSIL">
      <SpecificVersion>False</SpecificVersion>
      <HintPath>..\zedgraph_dll_v515\zedgraph_dll_v5.1.5\ZedGraph.dll</HintPath>
    </Reference>
  </ItemGroup>
  <ItemGroup>
    <Compile Include="AgentLogin.cs">
      <SubType>Form</SubType>
    </Compile>
    <Compile Include="AgentLogin.designer.cs">
      <DependentUpon>AgentLogin.cs</DependentUpon>
    </Compile>
```

```
<Compile Include="AgentSignup.cs">
  <SubType>Form</SubType>
</Compile>
<Compile Include="AgentSignup.designer.cs">
  <DependentUpon>AgentSignup.cs</DependentUpon>
</Compile>
<Compile Include="DistributeDataMenu.cs">
  <SubType>Form</SubType>
</Compile>
<Compile Include="DistributeDataMenu.designer.cs">
  <DependentUpon>DistributeDataMenu.cs</DependentUpon>
</Compile>
<Compile Include="DistributorLogin.cs">
  <SubType>Form</SubType>
</Compile>
<Compile Include="DistributorLogin.designer.cs">
  <DependentUpon>DistributorLogin.cs</DependentUpon>
</Compile>
<Compile Include="DistributorMain.cs">
  <SubType>Form</SubType>
</Compile>
<Compile Include="DistributorMain.designer.cs">
  <DependentUpon>DistributorMain.cs</DependentUpon>
</Compile>
<Compile Include="Graph.cs">
  <SubType>Form</SubType>
</Compile>
<Compile Include="Graph.Designer.cs">
  <DependentUpon>Graph.cs</DependentUpon>
</Compile>
<Compile Include="GuiltModel.cs">
```

```
  <SubType>Form</SubType>
</Compile>
<Compile Include="GuiltModel.Designer.cs">
  <DependentUpon>GuiltModel.cs</DependentUpon>
</Compile>
<Compile Include="GuiltRole.cs">
  <SubType>Form</SubType>
</Compile>
<Compile Include="GuiltRole.Designer.cs">
  <DependentUpon>GuiltRole.cs</DependentUpon>
</Compile>
<Compile Include="sender.cs">
  <SubType>Form</SubType>
</Compile>
<Compile Include="sender.Designer.cs">
  <DependentUpon>sender.cs</DependentUpon>
</Compile>
<Compile Include="Program.cs" />
<Compile Include="Properties\AssemblyInfo.cs" />
<EmbeddedResource Include="AgentLogin.resx">
  <DependentUpon>AgentLogin.cs</DependentUpon>
  <SubType>Designer</SubType>
</EmbeddedResource>
<EmbeddedResource Include="AgentSignup.resx">
  <DependentUpon>AgentSignup.cs</DependentUpon>
  <SubType>Designer</SubType>
</EmbeddedResource>
<EmbeddedResource Include="DistributeDataMenu.resx">
  <DependentUpon>DistributeDataMenu.cs</DependentUpon>
  <SubType>Designer</SubType>
</EmbeddedResource>
```

```
<EmbeddedResource Include="DistributorLogin.resx">
  <DependentUpon>DistributorLogin.cs</DependentUpon>
  <SubType>Designer</SubType>
</EmbeddedResource>
<EmbeddedResource Include="DistributorMain.resx">
  <DependentUpon>DistributorMain.cs</DependentUpon>
  <SubType>Designer</SubType>
</EmbeddedResource>
<EmbeddedResource Include="Graph.resx">
  <SubType>Designer</SubType>
  <DependentUpon>Graph.cs</DependentUpon>
</EmbeddedResource>
<EmbeddedResource Include="GuiltModel.resx">
  <SubType>Designer</SubType>
  <DependentUpon>GuiltModel.cs</DependentUpon>
</EmbeddedResource>
<EmbeddedResource Include="GuiltRole.resx">
  <SubType>Designer</SubType>
  <DependentUpon>GuiltRole.cs</DependentUpon>
</EmbeddedResource>
<EmbeddedResource Include="sender.resx">
  <SubType>Designer</SubType>
  <DependentUpon>sender.cs</DependentUpon>
</EmbeddedResource>
<EmbeddedResource Include="Properties\Resources.resx">
  <Generator>ResXFileCodeGenerator</Generator>
  <LastGenOutput>Resources.Designer.cs</LastGenOutput>
  <SubType>Designer</SubType>
</EmbeddedResource>
<EmbeddedResource Include="ViewData.resx">
  <SubType>Designer</SubType>
```

```
  <DependentUpon>ViewData.cs</DependentUpon>
</EmbeddedResource>
<EmbeddedResource Include="Welcome.resx">
  <DependentUpon>Welcome.cs</DependentUpon>
</EmbeddedResource>
<Compile Include="Properties\Resources.Designer.cs">
  <AutoGen>True</AutoGen>
  <DependentUpon>Resources.resx</DependentUpon>
  <DesignTime>True</DesignTime>
</Compile>
<None Include="app.config">
  <SubType>Designer</SubType>
</None>
<None Include="Properties\Settings.settings">
  <Generator>SettingsSingleFileGenerator</Generator>
  <LastGenOutput>Settings.Designer.cs</LastGenOutput>
</None>
<Compile Include="Properties\Settings.Designer.cs">
  <AutoGen>True</AutoGen>
  <DependentUpon>Settings.settings</DependentUpon>
  <DesignTimeSharedInput>True</DesignTimeSharedInput>
</Compile>
<Compile Include="SqlConn.cs" />
<Compile Include="ViewData.cs">
  <SubType>Form</SubType>
</Compile>
<Compile Include="ViewData.Designer.cs">
  <DependentUpon>ViewData.cs</DependentUpon>
</Compile>
<Compile Include="Welcome.cs">
  <SubType>Form</SubType>
```

```
  </Compile>
  <Compile Include="Welcome.Designer.cs">
    <DependentUpon>Welcome.cs</DependentUpon>
  </Compile>
</ItemGroup>
<ItemGroup>
  <BootstrapperPackage Include="Microsoft.Net.Client.3.5">
    <Visible>False</Visible>
    <ProductName>.NET Framework 3.5 SP1 Client Profile</ProductName>
    <Install>false</Install>
  </BootstrapperPackage>
  <BootstrapperPackage Include="Microsoft.Net.Framework.2.0">
    <Visible>False</Visible>
    <ProductName>.NET Framework 2.0 %28x86%29</ProductName>
    <Install>true</Install>
  </BootstrapperPackage>
  <BootstrapperPackage Include="Microsoft.Net.Framework.3.0">
    <Visible>False</Visible>
    <ProductName>.NET Framework 3.0 %28x86%29</ProductName>
    <Install>false</Install>
  </BootstrapperPackage>
  <BootstrapperPackage Include="Microsoft.Net.Framework.3.5">
    <Visible>False</Visible>
    <ProductName>.NET Framework 3.5</ProductName>
    <Install>false</Install>
  </BootstrapperPackage>
  <BootstrapperPackage Include="Microsoft.Net.Framework.3.5.SP1">
    <Visible>False</Visible>
    <ProductName>.NET Framework 3.5 SP1</ProductName>
    <Install>false</Install>
  </BootstrapperPackage>
```

```
  </ItemGroup>
  <ItemGroup>
    <None Include="Resources\apple-apple-1465730-1280-1024.jpg" />
  </ItemGroup>
  <ItemGroup>
    <None Include="Resources\3d_Apple_Logo_102_2.jpg" />
  </ItemGroup>
  <ItemGroup>
    <None Include="Resources\business_networking.jpg" />
  </ItemGroup>
  <ItemGroup>
    <None Include="Resources\networking.jpg" />
  </ItemGroup>
  <ItemGroup>
    <None Include="Resources\fantasy-background.jpg" />
  </ItemGroup>
  <ItemGroup>
    <None Include="Resources\background-1.jpg" />
  </ItemGroup>
  <ItemGroup>
    <None Include="Resources\globe_sm.gif" />
  </ItemGroup>
  <ItemGroup>
    <None Include="Resources\index.jpg" />
  </ItemGroup>
  <Import Project="$(MSBuildBinPath)\Microsoft.CSharp.targets" />
  <!-- To modify your build process, add your task inside one of the targets below and uncomment
it.
       Other similar extension points exist, see Microsoft.Common.targets.
  <Target Name="BeforeBuild">
  </Target>
```

```
  <Target Name="AfterBuild">
  </Target>
  -->
</Project>
```

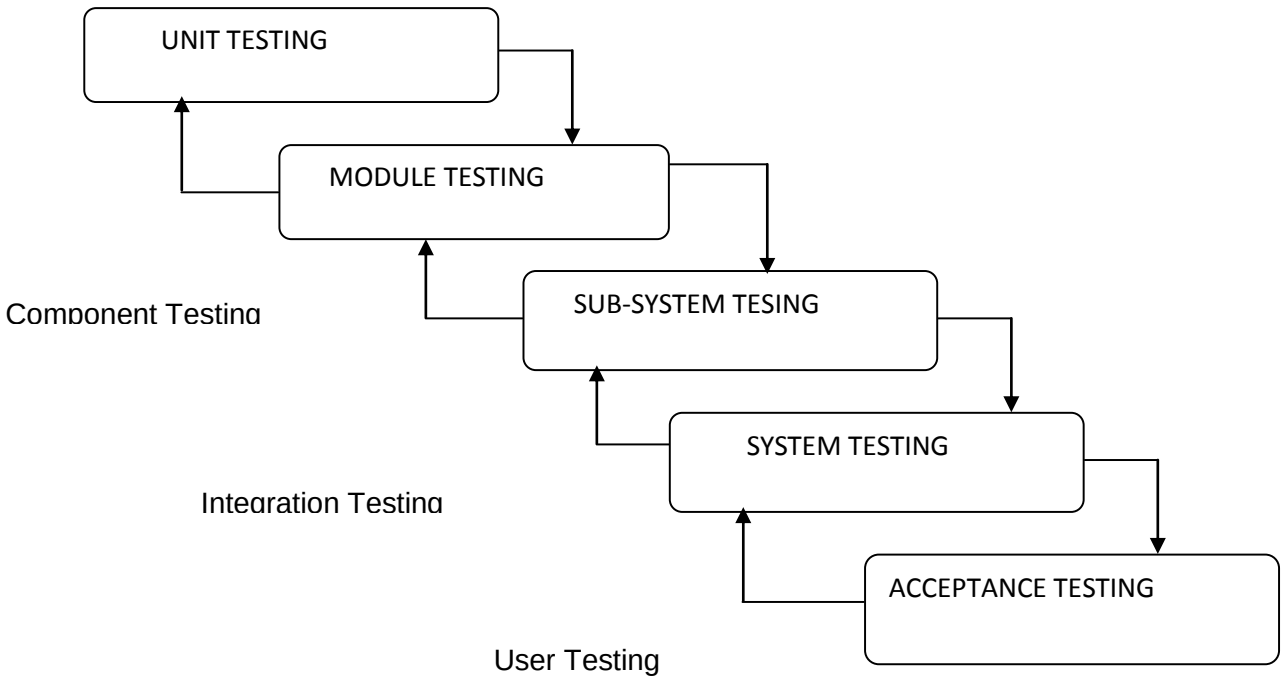
System Testing

7.1 Testing Methodologies

The software engineering process can be viewed as a spiral. Initially system engineering defines the role of software and leads to software requirement analysis where the information domain, functions, behavior, performance, constraints and validation criteria for software are established. Moving inward along the spiral, we come to design and finally to coding. To develop computer software we spiral in along streamlines that decrease the level of abstraction on each turn.

A strategy for software testing may also be viewed in the context of the spiral. Unit testing begins at the vertex of the spiral and concentrates on each unit of the software as implemented in source code. Testing progress by moving outward along the spiral to integration testing, where the focus is on the design and the construction of the software architecture. Talking another turn on outward on the spiral we encounter validation testing where requirements established as part of software requirements analysis are validated against the software that has been constructed. Finally we arrive at system testing, where the software and other system elements are tested as a whole.

WATER FALL MODEL



Unit Testing

Unit testing focuses verification effort on the smallest unit of software design, the module. The unit testing we have is white box oriented and some modules the steps are conducted in parallel.

White Box Testing

This type of testing ensures that

- All independent paths have been exercised at least once
- All logical decisions have been exercised on their true and false sides
- All loops are executed at their boundaries and within their operational bounds
- All internal data structures have been exercised to assure their validity.

To follow the concept of white box testing we have tested each form .we have created independently to verify that Data flow is correct, All conditions are exercised to check their validity, All loops are executed on their boundaries.

Basic Path Testing

Established technique of flow graph with Cyclomatic complexity was used to derive test cases for all the functions. The main steps in deriving test cases were:

Use the design of the code and draw correspondent flow graph.

Determine the Cyclomatic complexity of resultant flow graph, using formula:

V(G)=E-N+2 or

V(G)=P+1 or

V(G)=Number Of Regions

Where V(G) is Cyclomatic complexity,

E is the number of edges,

N is the number of flow graph nodes,

P is the number of predicate nodes.

Determine the basis of set of linearly independent paths.

Conditional Testing

In this part of the testing each of the conditions were tested to both true and false aspects. And all the resulting paths were tested. So that each path that may be generate on particular condition is traced to uncover any possible errors.

Data Flow Testing

This type of testing selects the path of the program according to the location of definition and use of variables. This kind of testing was used only when some local variable were declared. The *definition-use chain* method was used in this type of testing. These were particularly useful in nested statements.

Loop Testing

In this type of testing all the loops are tested to all the limits possible. The following exercise was adopted for all loops:

➔ All the loops were tested at their limits, just above them and just below them.

➔ All the loops were skipped at least once.

➔ For nested loops test the inner most loop first and then work outwards.

For concatenated loops the values of dependent loops were set with the help of connected loop.

Unstructured loops were resolved into nested loops or concatenated loops and tested as above. Each unit has been separately tested by the development team itself and all the input have been validated.

Integration Testing

Testing is done for each module. After testing all the modules, the modules are integrated and testing of the final system is done with the test data, specially designed to show that the system will operate successfully in all its aspects conditions. Thus the system testing is a confirmation that all is correct and an opportunity to show the user that the system works.

The purpose of integration testing is to verify functional, performance and reliability requirements placed on major design items. These “design items”, i.e. Assemblages (or groups of

units), are exercised through theor interfaces using black box testing, sucess and error cases being simulated via appropriate parameter and data inputs. Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface.

Test cases are constructed to test that all components with in assemblges interect correctly, for example across procedure calls or process activations, and this is done after testing individual modules, i.e. Unit testing.

The overall idea is a "building block" approach, in which verified assemblages are added to a verified base which is then used to support the integration testing of further assemblages.

7.2 Test Cases

Test cases for Distributor Login

Test case	Input being checked	Expected output
If password is not matched with confirm password	Password=confirm password	Mismatched password
1.password is empty	Password length>6	Password should have at least 6 characters
User id contains spaces	User id should be alphanumeric	Blank space are not allowed

Test case for Agent Signup

Test case	Input being checked	Expected output
1.UserId	User id should be alphanumeric	Valid / Invalid
2. User Name	User name should be alphabetic	Valid / Invalid user name
3. AgenetPwd	Passwordlenght>4	Password should have at least 4 charcter
4.Address	address should be alphanumeric	Blank spaces also eligible
5.Contact Number	Contact number should be Number	Contact Number have at least 10 Character
6. Contact Person	Person name should be character	Contact person name should be characters

Test case for Agent Login

Test case	Input being checked	Expected output
1. Agent Id	Agent id should be alphanumeric	Agent id not selected. Selected at least one id
2. UserName	User Name should be character	Valid / Invalid user name
3. AgenetPwd	Passwordlenght>4	Password should have at least 4 charcter

Test case for Data Transfer To Agents

Test case	Input being checked	Expected output
1. Agent Id	Agent id should be alphanumeric	Agent id not selected. Selected at least one id
2.ServerIP	Server IP should be Alphanumeric	Valid / Invalid IP
3. Work Group Name	Group name Alphanumeric	It's valid work group name or not

OUTPUT SCREENS

MONITORING FOR DETECTION & PREVENTION OF FAKE AGENTS

Figure 8.1.1 Output Screen for Welcome to

Monitoring For Detection & Prevention Of Fake Agents

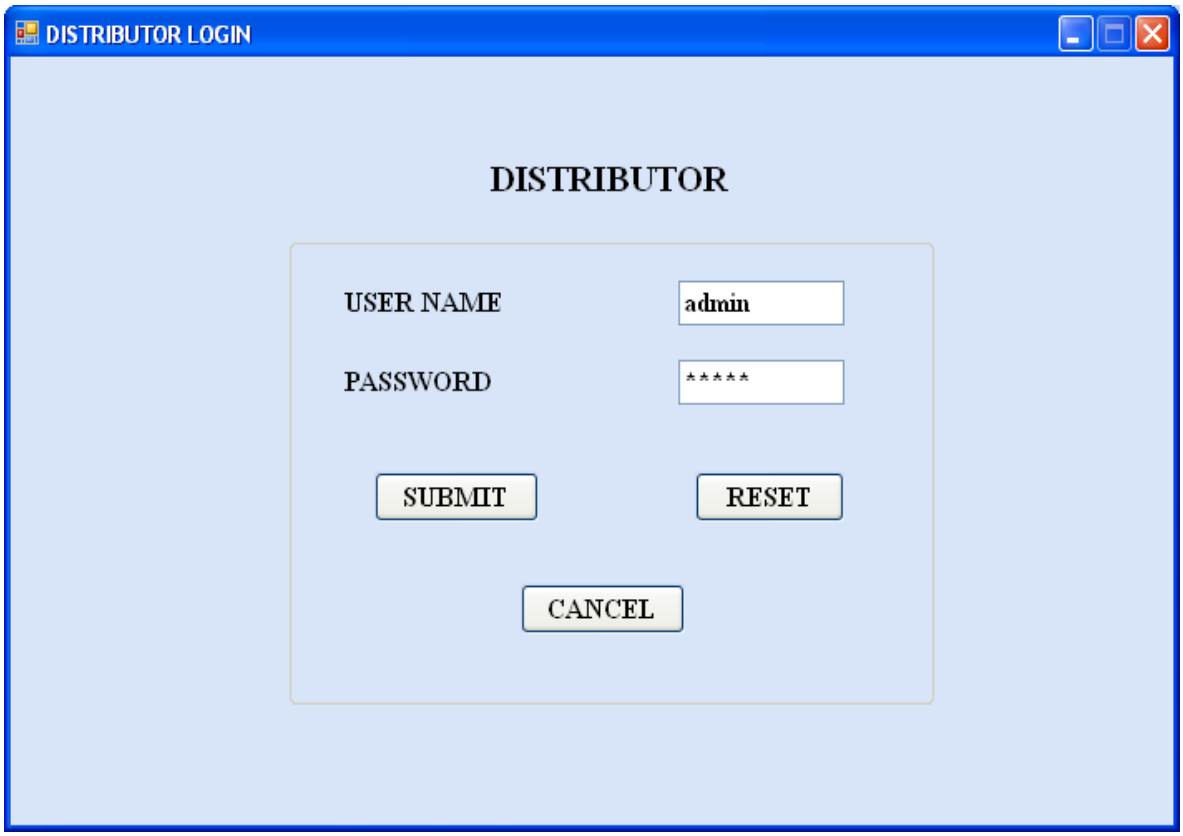


Figure 8.1.2 Output Screen for Distributor Login

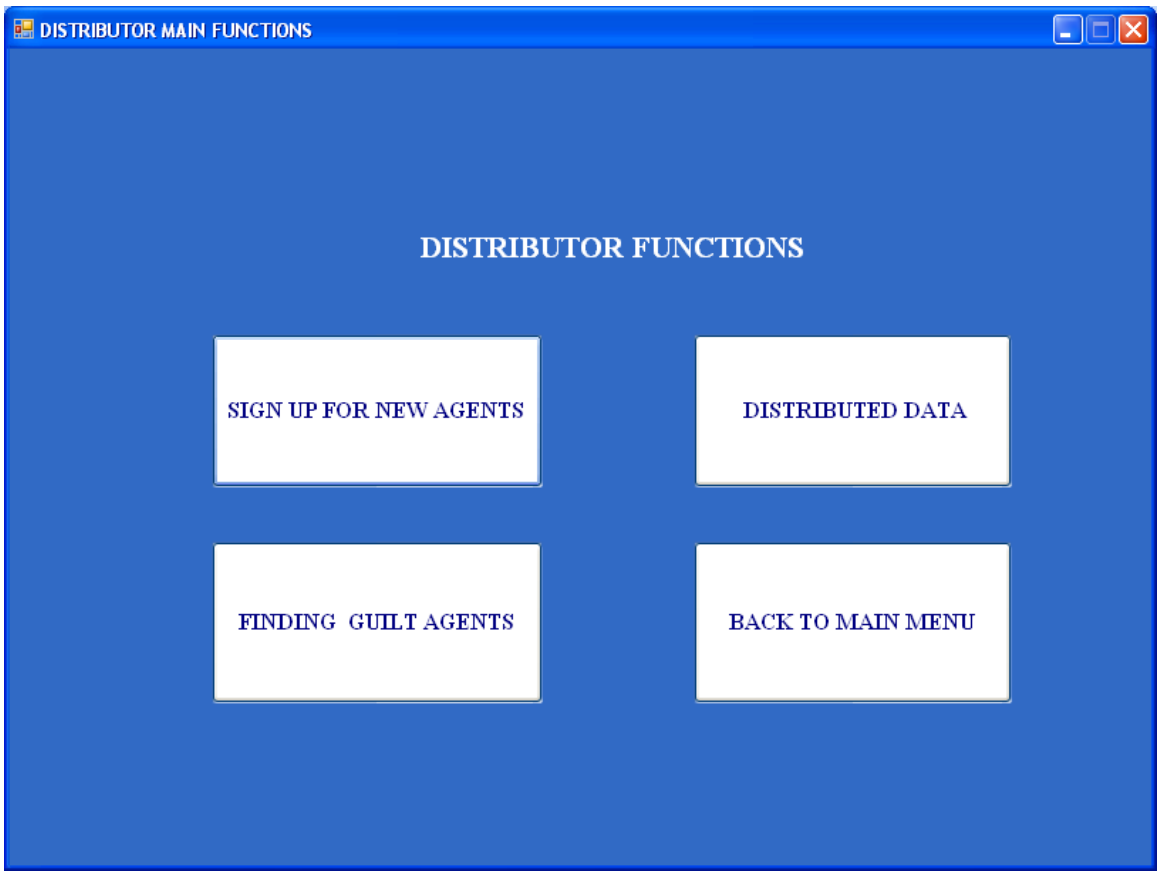


Figure 8.1.3 Output Screen for Distributor Functionalities

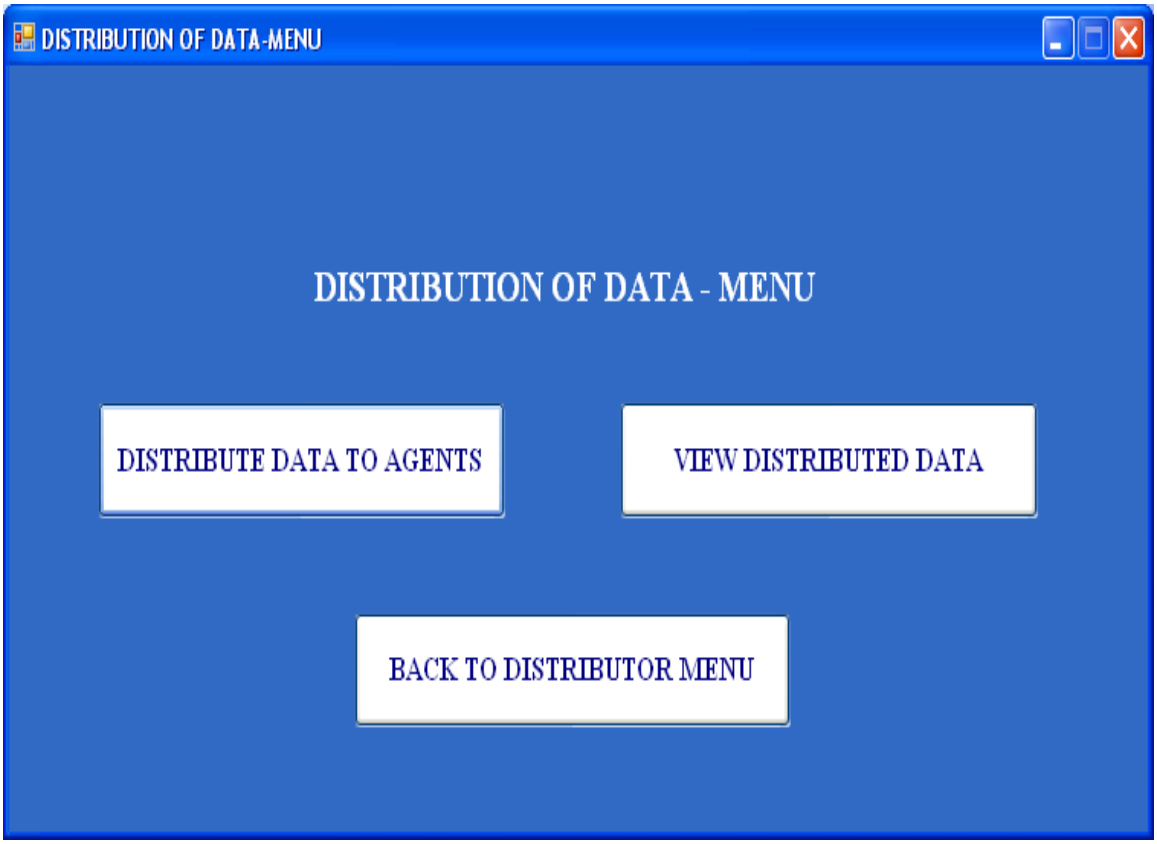


Figure 8.1.4 Output Screen for Distribution Of Data – Menu

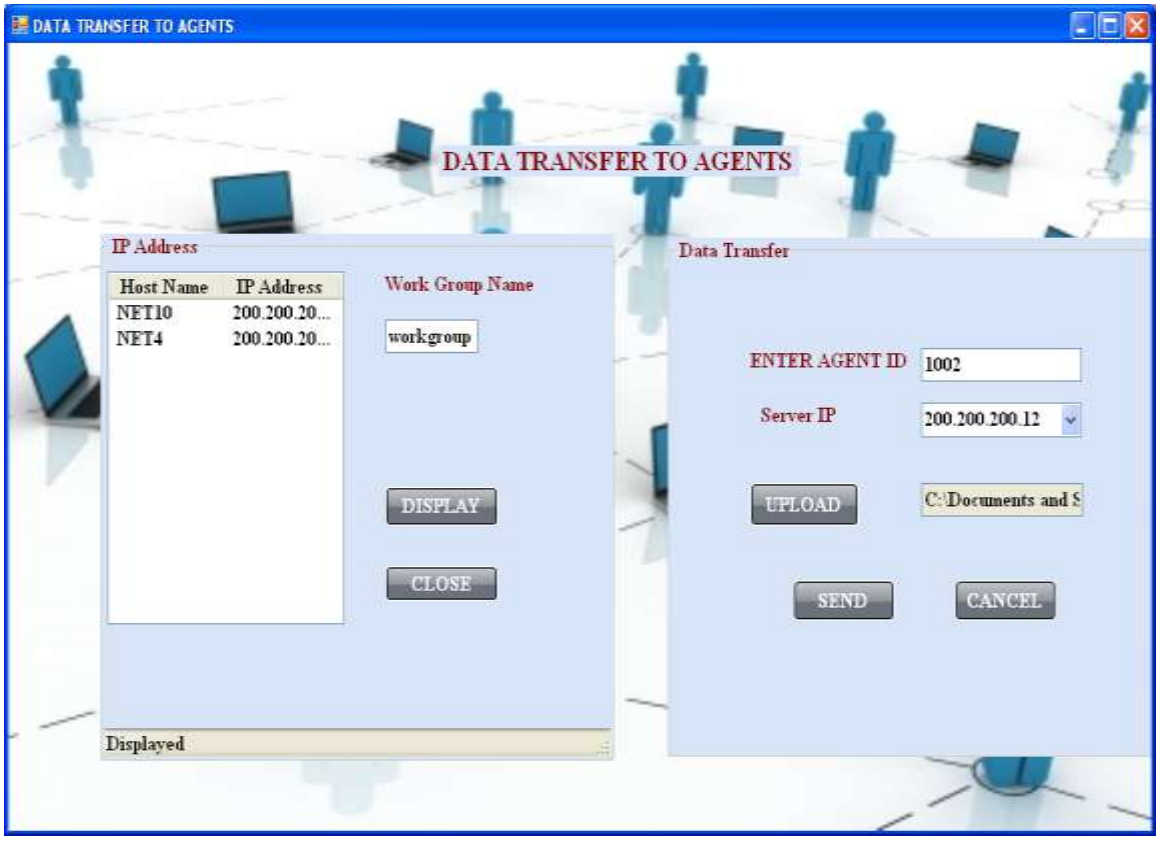


Figure 8.1.5 Output Screen for Distribute Data To Agent

Figure 8.1.6 Output Screen for Sign Up For New Agent

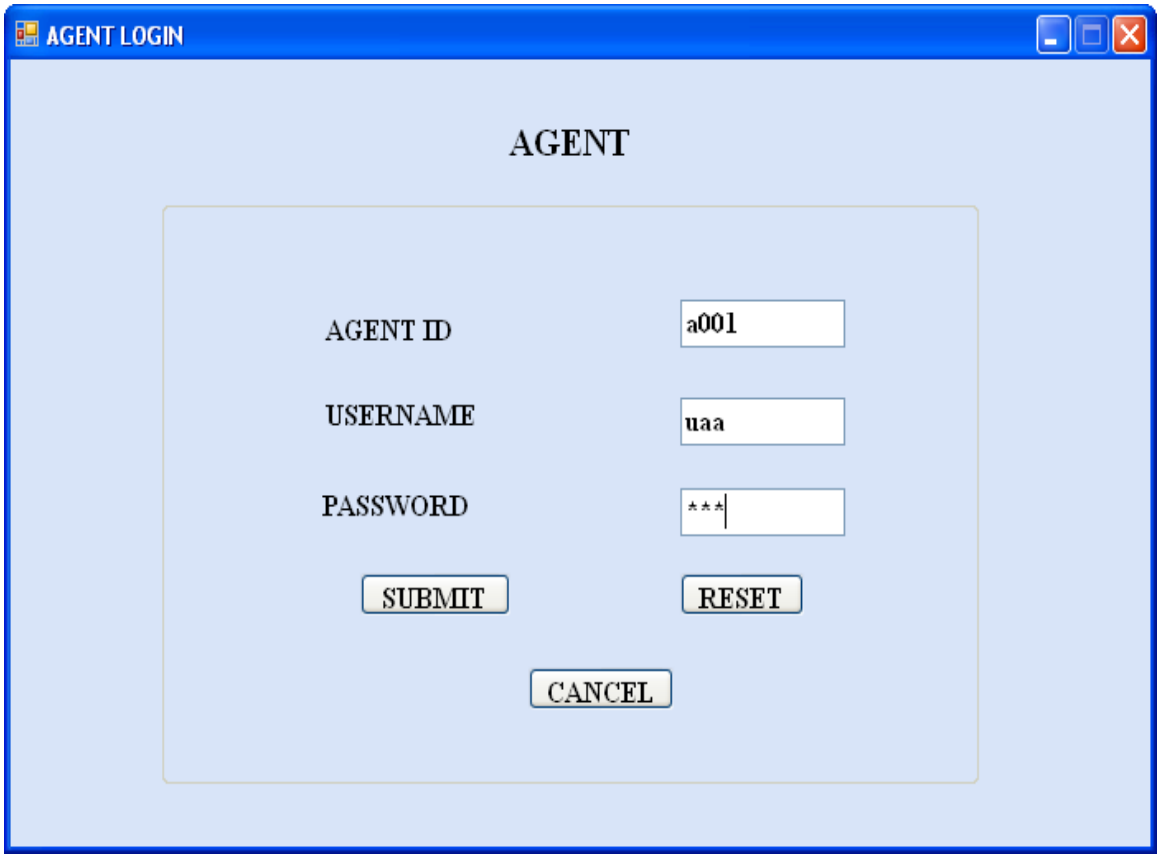


Figure 8.1.7 Output Screen for Agent Login

8.2 Reports

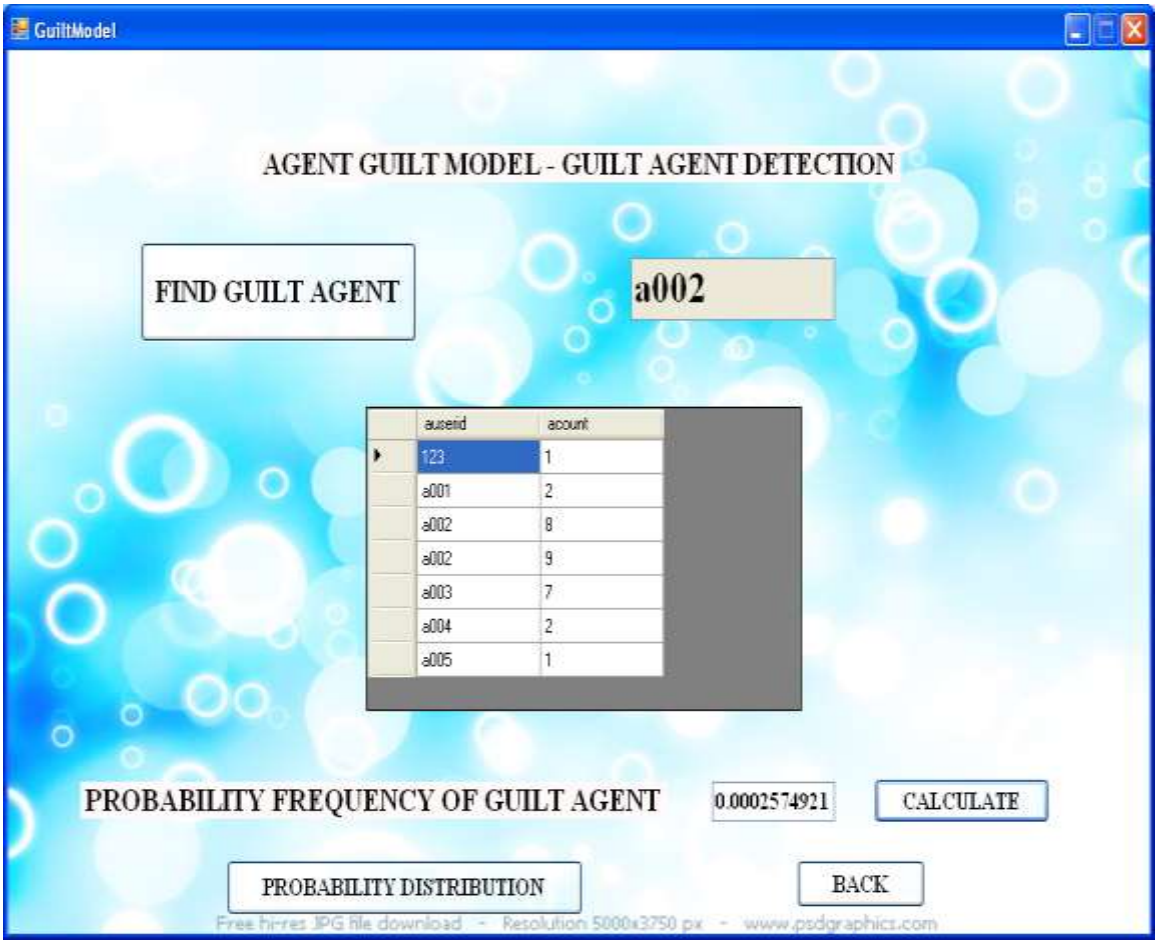


Figure 8.2.1 Report Screen For Finding Guilt Agent

Figure 8.2.2 Report Screen For Receiver Through Agent

Figure 8.2.3 Report Screen For Recieverthrudistributor

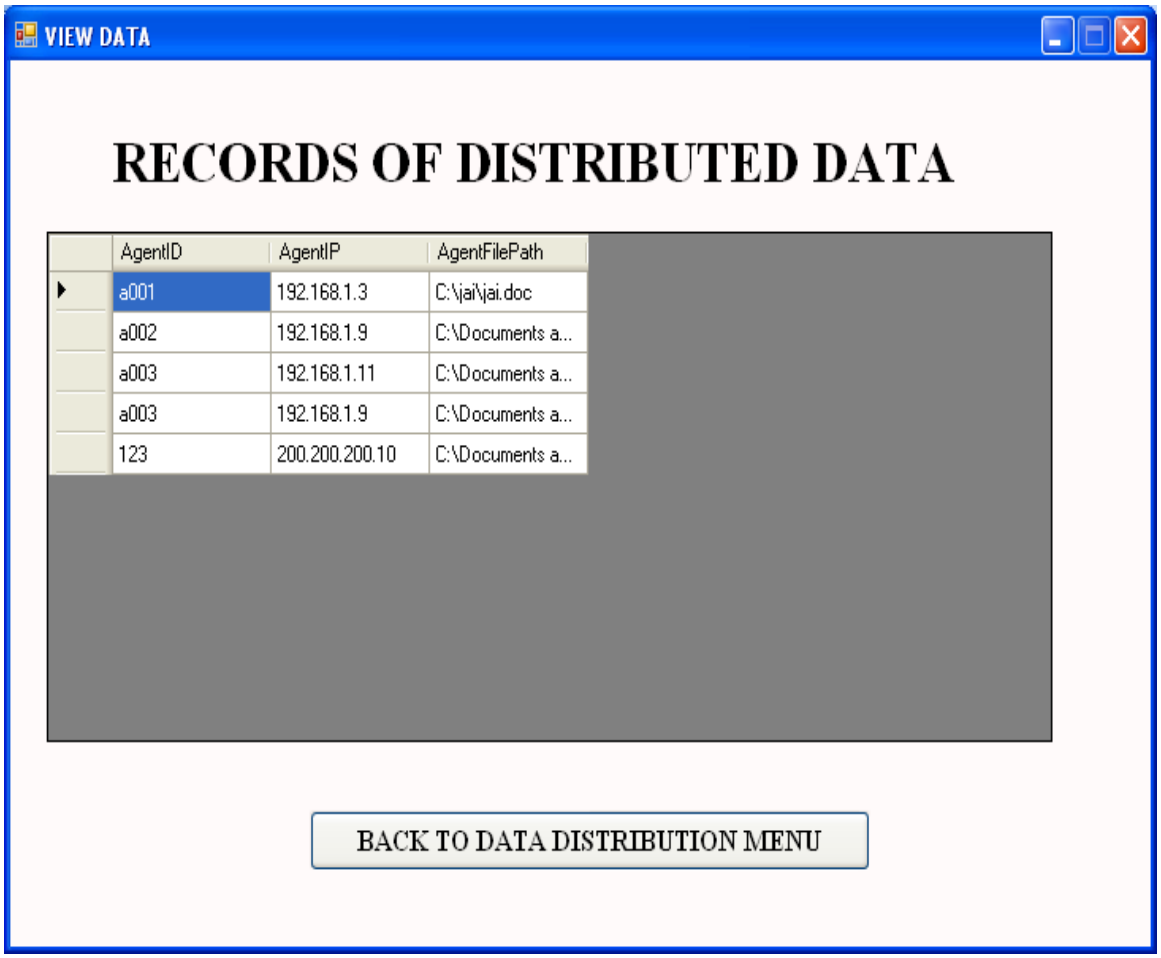


Figure 8.2.4 Report Screen For View Distributed Data

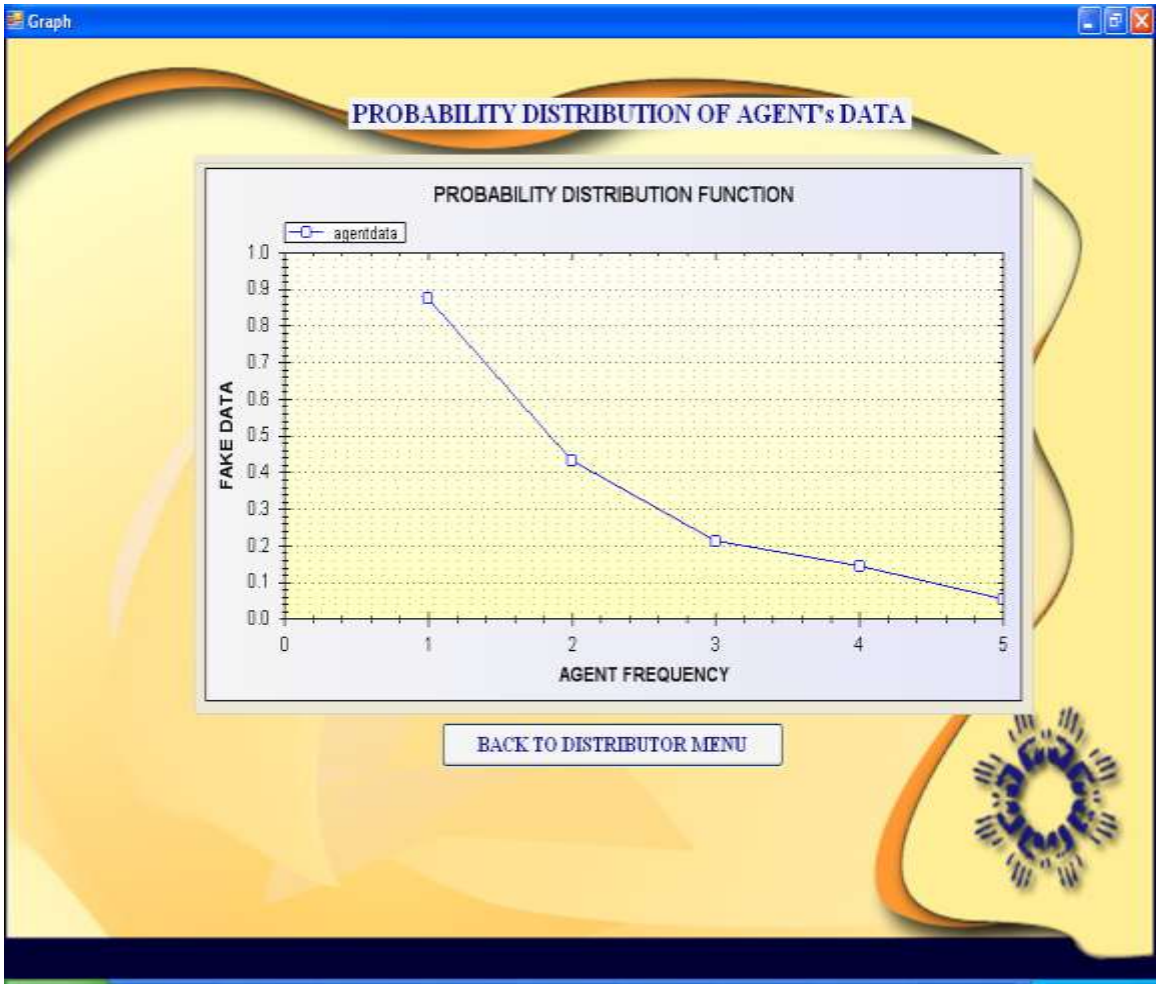


Figure 8.2.5 Report Screen For Probability Distribution

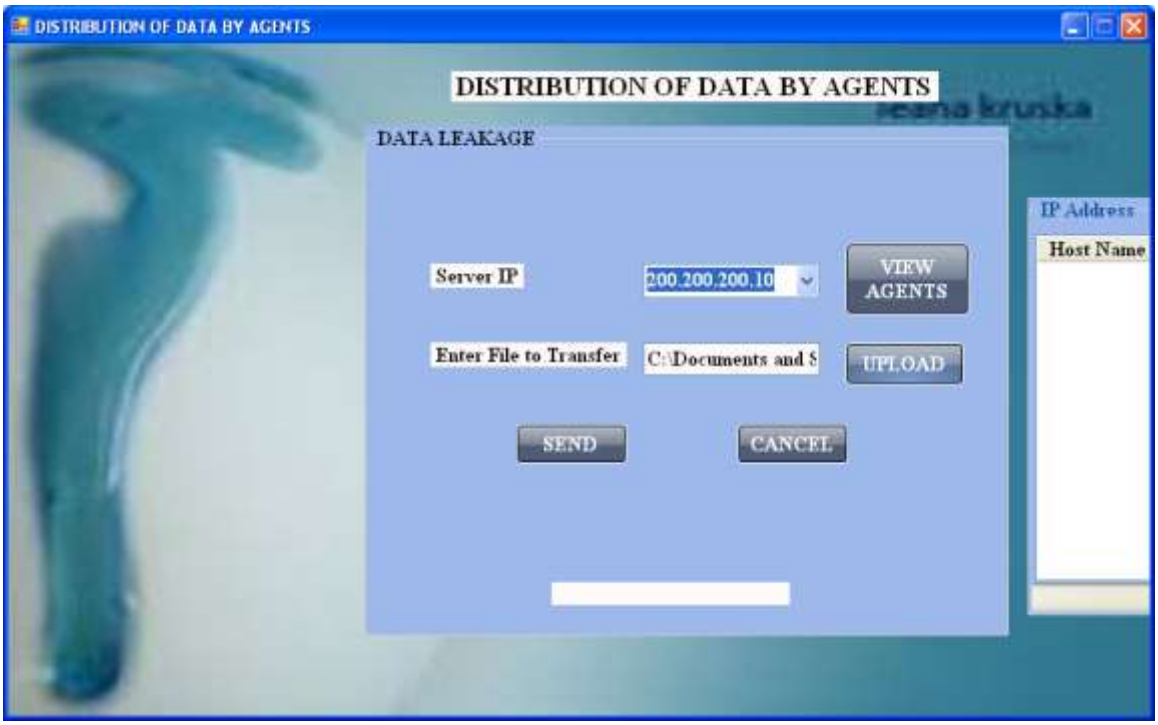


Figure 8.26 Output & Report Screen For Distribution Of Data By Agent

Figure 8.2.7 Successful Of The Distribution

Conclusion & Future Enhancement

9.1 Conclusion

In a perfect world, there would be no need to hand over sensitive data to agents that may unknowingly or maliciously leak it. And even if we had to hand over sensitive data, in a perfect world, we could watermark each object so that we could trace its origins with absolute certainty. However, in many cases, we must indeed work with agents that may not be 100 percent trusted, and we may not be certain if a leaked object came from an agent or from some other source, since certain data cannot admit watermarks. In spite of these difficulties, we have shown that it is possible to assess the likelihood that an agent is responsible for a leak, based on the overlap of his data with the leaked data and the data of other agents, and based on the probability that objects can be "guessed" by other means.

9.2 Future Enhancement

Our model is relatively simple, but we believe that it captures the essential trade-offs. The algorithms we have presented implement a variety of data distribution strategies that can improve the distributor's chances of identifying a leaker. We have shown that distributing objects judiciously can make a significant difference in identifying guilty agents, especially in cases where there is large overlap in the data that agents must receive.

Our future work includes the investigation of agent guilt models that capture leakage scenarios that are not studied in this project. For example, what is the appropriate model for cases where agents can collude and identify fake tuples? Another open problem is the extension of our allocation strategies so that they can handle agent requests in an online fashion (the presented strategies assume that there is a fixed set of agents with requests known in advance).

Glossary

ACRONYM	ABBREVATION
DLD	Data Leakage Detection
GA	Guilty Agent
AGM	Agent Guilt Model
DA	Data Allocation
GMA	Guilt Model Analysis
FO	Fake Objects
FT	Fake Tuple
DR	Data Request
EF	Explicit Fake Object
SF	Simple Fake Object
SR	Simple Request
ER	Explicit Request
CFO	Create Fake Object
IGP	Internet Grouping Protocol
EGP	Exterior Gateway Protocol
IGRP	Interior Gateway Routing Protocol
VLSM	Variable Length Subnet Masking
PL	Physical Layer
LL	Link Layer
NL	Network Layer
TL	Transport Layer

SL	Session Layer
PL	Presentation Layer
AL	Application Layer
NIDS	Network Instruction Detection System
SSL	Secure Socket Layer
HTTP	Hyper Text Transfer Protocol
FTP	File Transfer Protocol
DMZ	Demilitarized Zone
SMTP	Simple Mail Transfer Protocol
POP3	Post Office Protocol version 3
PP	Pre-Pishing
SQL	Structured Query Language
EGP	Exterior Gateway Protocol
IGMP	Internet Group Management Protocol
MBGP	Multiprotocol Extension for BGP
RIP	Routing Information Protocol
MTU	Maximum Transmission Unit
RFC	Request For Comment
MLD	Multicast Listener Discovery
PIM	Protocol Independent Multicast
IETF	Internet Engineering Task Force
ICMP	Internet Control Message Protocol
EIGRP	Enhanced Interior Gateway Routing Protocol
OMT	Object Modelling Technique

NSFNet	National Science Foundation Network
ARPANet	Advanced Research Project Agency Network

References

[1]. SANS Institute InfoSec Reading Room This Project is from the SANS Institute Reading Room site. Reposting is not permitted without express written permission. Data Leakage - Threats and Mitigation

[2]. Medical data mining: insights from winning two competitions by Saharon Rosset · Claudia Perlich · Grzergorz Swirszcz · Prem Melville · Yan Liu

[3]. Transient Analysis and Leakage Detection Algorithm using GA and HS algorithm for a Pipeline System by sang – Hyun Kim* Environmental Engineering, Pusan National University, Geumjeng-gu, busann 609-735, korea. "It's a Journal of Mechanical Science and Technology (KSME Int. J), Vol.20, No. 3 PP 426-434, 2006"

[4]. A Literature Review of Domain Adaptation with Unlabeled Data by Anna Margolis amargoli@u.washington.edu March 23, 2011

[5]. R. Agrawal and J. Kiernan, "Watermarking Relational Databases," Proc. 28th Int'l Conf. Very Large Data Bases (VLDB '02), VLDB Endowment, pp. 155-166, 2002.

[6]. P. Bonatti, S.D.C. di Vimercati, and P. Samarati, "An Algebra for Composing Access Control Policies," ACM Trans. Information and System Security, vol. 5, no. 1, pp. 1-35, 2002.

[7]. P. Buneman, S. Khanna, and W.C. Tan, "Why and Where: A Characterization of Data Provenance," Proc. Eighth Int'l Conf. Database Theory (ICDT '01), J.V. den Bussche and V. Vianu, eds., pp. 316-330, Jan. 2001.

[8]. P. Buneman and W.-C. Tan, "Provenance in Databases," Proc. ACM SIGMOD, pp. 1171-1173, 2007.

[9]. Y. Cui and J. Widom, "Lineage Tracing for General Data Warehouse Transformations," The VLDB J., vol. 12, pp. 41-58, 2003.

[10]. S . Czerwinski, R. Fromm, and T. Hodes, "Digital Music Distributionand Audio Watermarking," http://www.scientificcommons.org/43025658, 2007.

[11]. F. Guo, J. Wang, Z. Zhang, X. Ye, and D. Li, "An Improved Algorithm to Watermark Numeric Relational Data," Information Security Applications, pp. 138-149, Springer, 2006.

[12]. F. Hartung and B. Girod, "Watermarking of Uncompressed and Compressed Video," Signal Processing, vol. 66, no. 3, pp. 283-301, 1998.

[13]. S. Jajodia, P. Samarati, M.L. Sapino, and V.S. Subrahmanian, "Flexible Support for Multiple Access Control Policies," ACM Trans. Database Systems, vol. 26, no. 2, pp. 214-260, 2001.

[14]. Y. Li, V. Swarup, and S. Jajodia, "Fingerprinting Relational Databases: Schemes and Specialties," IEEE Trans. Dependable and Secure Computing, vol. 2, no. 1, pp. 34-45, Jan.-Mar. 2005.

[15]. B. Mungamuru and H. Garcia-Molina, "Privacy, Preservation and Performance: The 3 P's of Distributed Data Management," technical report, Stanford Univ., 2008.

[16]. V.N. Murty, "Counting the Integer Solutions of a Linear Equation with Unit Coefficients," Math. Magazine, vol. 54, no. 2, pp. 79-81, 1981.

[17]. S.U. Nabar, B. Marthi, K. Kenthapadi, N. Mishra, and R. Motwani, "Towards Robustness in Query Auditing," Proc. 32nd Int'l Conf. Very Large Data Bases (VLDB '06), VLDB Endowment, pp. 151-162, 2006.

[18]. P. Papadimitriou and H. Garcia-Molina, "Data Leakage Detec- tion," technical report, Stanford Univ., 2008.

[19]. P.M. Pardalos and S.A. Vavasis, "Quadratic Programming with One Negative Eigenvalue Is NP-Hard," J. Global Optimization, vol. 1, no. 1, pp. 15-22, 1991.

[20] . J.J.K.O. Ruanaidh, W.J. Dowling, and F.M. Boland, "Watermark- ing Digital Images for Copyright Protection," IEE Proc. Vision, Signal and Image Processing, vol. 143, no. 4, pp. 250-256, 1996.

[21]. R. Sion, M. Atallah, and S. Prabhakar, "Rights Protection for Relational Data," Proc. ACM SIGMOD, pp. 98-109, 2003.

[22]. L. Sweeney, "Achieving K-Anonymity Privacy Protection Using Generalization and Suppression," http://en.scientificcommons. org/43196131, 2002.

FOR .NET INSTALLATION

[1]. www.support.mircosoft.com

FOR DEPLOYMENT AND PACKING ON SERVER

[2]. www.developer.com

[3]. www.15seconds.com

FOR SQL

[4]. www.msdn.microsoft.com

FOR ASP.NET

Asp.Net 3.5 Unleashed

[5]. www.msdn.microsoft.com/net/quickstart/aspplus/default.com

[6]. www.asp.net

[7]. www.fmexpense.com/quickstart/aspplus/default.com

[8]. www.asptoday.com

[9]. www.aspfree.com

[10]. www.4guysfromrolla.com/index.aspx

→ Software Engineering (Roger's Pressman)

Index of List Of Figures

Tables Index

PUBLICATIONS

[1] Monitor For Detection & Prevention of Fake Agents has been published by Intenational Journal Of Computer Science Trends and Technology (IJCTT) in July – august 2012, by K.SUDHEER KUAMR[1], CH.S.V.V.S.N MURTY[2]

[2] Published "Releaving of Leakage Information" at International Conference On Recent Advaces In Computer Science 2012 (ICRACS2K12) at Godavari Institute Of Engineering & Technology during 30th-31st March -2012.

[1]PG Student, Department of CSE, Sri Sai Aditya Institute Of Science & Technology, Surampalem, Andhra Pradesh, India. sudheerkumarkotha@gmail.com, chsatyamurthy@gmail.com.