2011

E-mail behavior profiling: Based on attachment and language text

Onur Polatcan

E-mail Behavior Profiling:

Based on attachment and language text

by

Onur Polatcan

Committee Members

Dr. Yin Pan

Dr. Sumita Mishra

Dr. Charlie Border

Thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science in

Networking and Systems Administration

Rochester Institute of Technology

B. Thomas Golisano College

of

Computing and Information Sciences

02/01/2011

Abstract

Employees are stealing confidential information from its company via e-mail without detection. Invisible Witness tool can provide new assistance for an organization. Its techniques will automatically detect certain patterns across user accounts that indicate covert or malicious activities. Furthermore, this application assists the network administrator with targeted investigations. Many applications look for specific text or attachments, but Invisible Witness is the only application that is capable of creating user profiling. This application works with over 95 percent accuracy. I am sure that the Invisible Witness tool will minimize business risk to help protect valuable company information.

Acknowledgments

I would like to thank the following people whose support helped to make this thesis possible:

Dr. Yin Pan, for her leadership, support, attention to each and every detail and her invaluable suggestions made this work successful. She has been everything that a student could ever want from an advisor.

Dr. Sumita Mishra, for her support and for teaching some of the most interesting classes offered at RIT.

Dr. Charlie Border, for keeping me motivated and giving me key ideas to raise the overall quality for this project.

Professor Ed Wolf, for his inspiration and reading my work over and over again.

My parents, to whom I am deeply and forever indebted to for their support, love, and motivation.

My Wife, Julia, for her love and patience during this project.

Contents

List of Figures

1. Introduction

Background

We are using the e-mail method of communication more than using regular mail or phone calls. E-mail is a part of our daily activity. As I come to work each day, the first thing that I do is to check my e-mail and make sure there are not any urgent messages that need to be prioritized and processed before doing anything else. Since we are composing and reading e-mails every day, the mundane task facilitates a quick response without much thought about the wording, the context, the content, the punctuation, the spelling, and the grammar. With the various features that e-mail communication applications have to offer, such as Microsoft Outlook's auto-complete "TO" field, one little mistake with the use of that auto-complete function could cause public embarrassment, loss of reputation, loss of job, or even civil or criminal liability both for the sender and for the company. Solving Internet and email security greatly assists surveillance intelligence activities. For instance, the discovery of an employee account and detection of the behavior can be directed to uncover of certain classes of covert, clandestine or espionage behavior performed with Internet resources. Dr. Moore worked as a federal prosecutor in Manhattan, where she focused on investigating and prosecuting "White-Collar" criminal cases. She has also written many articles about e-mail communication. Dr. Moore pointed out that e-mail has become the "smoking gun" of choice in today's white collar prosecutions [16].

Many levels of data sensitivity exist. As an employee of a company, one should know that any confidential data owned by the company should not travel outside the organization without authorization. For instance, in my case, I work for a pump company

that is very successful in pump manufacturing. They have some sensitive information, such as customer lists, pump drawings, budget tactics, marketing strategies, and so forth. As a network administrator for my company, some of my duties are to make sure that no sensitive data are accessible to unauthorized parties. Also, those who are authorized should always be monitored since they handle sensitive data. Other sensitive areas in any organization include the use of USB flash drives, CDRW, and so forth. At my work all USB ports are disabled and cannot be used by anyone except IT staff, when needed. In almost every organization, employees can be bilingual or trilingual; and this also can be a leakage issue. For the reasons above, it is important to prevent leakage of confidential data by e-mail with content filtering that include text and attachments. When used appropriately, an e-mail behavioral profile can dramatically increase the odds of highlighting possible malicious activity. It can provide the network administrator with a specific time frame that identifies a sender's activity, and the application promotes a developmental plan that tracks activity and defines the next critical step. Ultimately, it can help organizations to pin-point their most critical security needs and help IT and upper management better allocate their protected resources.

2. Related Work

2.1 Overview

Many of us wear a mask to disguise the "dark side" of our personalities. In the natural one person can be disguised by having a beard one day, and with another day and another mood, clean shaven is the prescription. On one day a person may use a British accent for a specific purpose, and just as quickly, change the accent to a normal accent. Consider the role of a salesperson, and how he or she may dress to impress a potential client. The salesperson may use verbal and non-verbal communication techniques to reduce any resistance a potential client may have in the persuasive role of salesmanship.

My motivation and inspiration for this research originated from works of Hadjidj, Debbabi, Lounis, Iqbal, Szporer, and Benredjem (2009). Their work mainly focuses on keyword searching to authorship attribution of anonymous e-mails; however, it didn't include a notification tool that would alarm a network administrator when adverse circumstances may occur in an organizational workplace. I decided to investigate the possibilities of integrating a specialized tool into an e-mail client that would flag e-mails that include inflammatory language, internal and external espionage, and attachment violations in multiple ethnic languages. Because most companies have bilingual or trilingual employees, my motivation is to provide an internal tool that will monitor language modifications from English to other languages. For instance, an angry employee that is fluent in French could easily send protected company data using French or any other language to reduce detection within the system. Nearly 90 percent of the data loss commonly happens by the email method of communication. My tool will

be extremely helpful to protect any organization from internal and external threats where internet access is concerned.

2.2 New Methods

How do you read someone? Reading a newspaper and obtaining knowledge is relatively simple, but to really know someone takes time, communication cues – both verbal and non-verbal, intuition, and relationship. E-mail profiling has similar characteristics of relationships, and the techniques required will involve give and take, experience, and intently involved practice.

Many know that research on e-mail filtering is very limited; however, studies on spam filtering are very mature. One must know that spam filtering and e-mail filtering is not the same thing. Professionals may need e-mail filtering for their specific needs [24].

Making the transition from relationships to employee behavior includes several correlations. What happens when employees are laid off, become dissatisfied with their jobs, look for new opportunities, or are terminated from their position. "Nearly 60 percent of employees who quit a job or are asked to leave are stealing company data, according to a report by the Ponemon Institute, a Tucson based research group." [13]. In Krebs' paper he gives a broad view of how and why employees inadvertently send out e-mails containing important company data. Apparently, many companies are still naive or careless about securing their work environment to remain competitive in the business world.

The toolkit named, Integrated E-Mail Forensic Analysis Framework, also known as IEFAF[7], uses statistical distributions to give an overview of the entire email communication investigation. This research used a data mining software called, WEKA. All of the classification was created and converted into a feature and was transferred to

a WEKA compatible format to get a valuable data set. My research differs in the language set. Where their research focused on English by using a stream of characters and word sequences, my research will identify when English is not being used. The limitations in previous research include only one single language. However, IEFAF stipulates word sequences that treat a group of words as one word. For example, the words, United States of America, could be just one word, such as, USA.

2.3 Work Presented

This thesis will investigate how to detect abnormal e-mail activity based on the language of an e-mail and the use of attachments within an e-mail. This tool will detect all mime-types of an e-mail and analyze the body and its attachment parts. If the user does not use the English language in the message part, the network administrator will be notified if the limit is reached by the user. There is a user allowance percentage limit in the config.properties file to specify how much is allowed for typos, technical words, and so forth. This setting is set for a 25 percent limit, but it can be changed at any time by the network administrator. The application also creates a user profile for e-mail attachments and monitors dates for profiling, which can be specified by the user. In the experimental results of this research, I used three days for a specific user. If the Invisible Witness tool does not find any attachment data within that three day period, then it extends its search for a seven day period. If there is no data within seven days, then the application looks for the last attachment size regardless of the time limit. If there is no detection, then it displays a zero. The network administrator, however, will get an e-mail alert if an over-size attachment is sent. This message includes the sender, receiver, subject of that e-mail, date, attachment name, current sent attachment size,

user oversized average attachment size, and user up-to-date attachment size. This information would give enough sources to determine if there is a security risk involved or not. This application will eliminate the fear of losing intellectual property that is critical for an organization. After researching a considerable number of technical research papers [4,6,7,14,17,18,21,24], I have found very few applications that support an e-mail standard to restrict improper use of this medium.

3. Methodology

The thesis can monitor abnormal activities based on user profile, in order to statistically measure the effectiveness of this concept, I developed an application that captures attachment size that user sends within the given period, it also identifies the language of the text. To simulate a real-time scenario, I decided to use G-mail servers because it would give anyone an opportunity to replicate the same environment without environment setup overhead. For instance, if the application is running and a user sends an email with an attachment size greater than the average user profile allowed limit, application would do both notifying the network administrator and profiling the message information such as sender, receivers, attachments size, attachment name, and sent date.

3.1 UML Domain Model

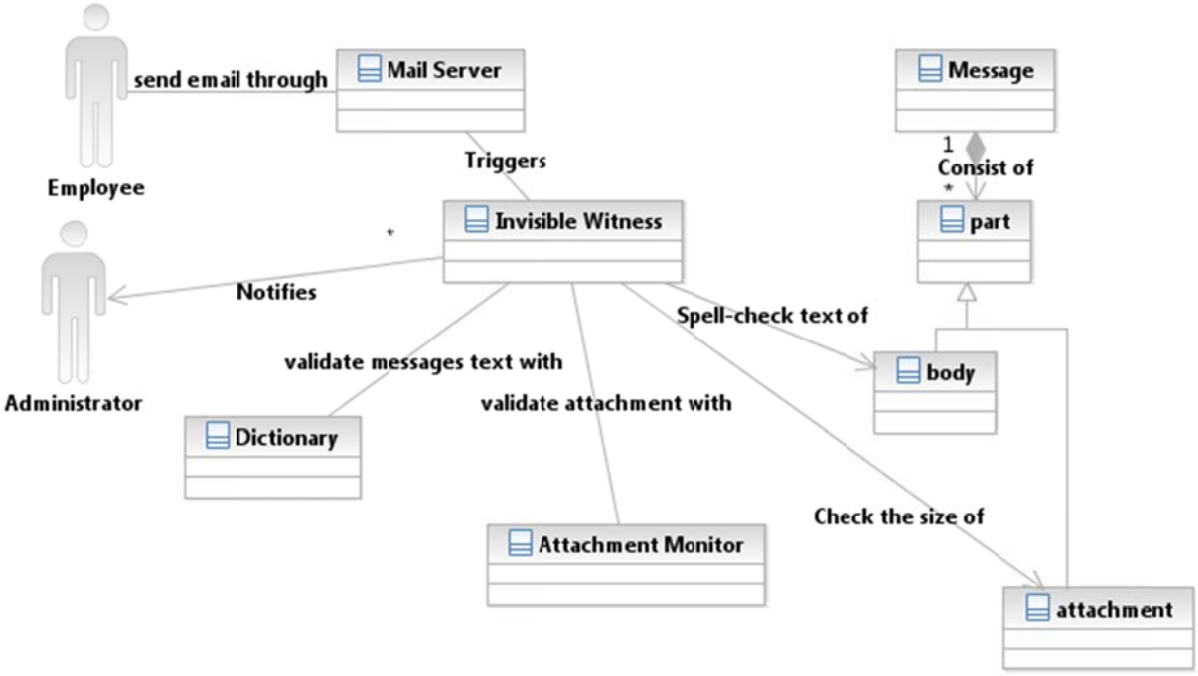


Figure 1, UML Domain Model

This diagram depicts the problem domain. In the model above, an employee sends an email using an e-mail server such as G-mail, Exchange, Lotus Domino and so forth. The tool (Invisible Witness) runs in the background and monitors the e-mail server for any activity. As soon as the Invisible Witness tool gets the notification from an e-mail server that e-mail has been sent, it checks the size of its attachment and spell checks the body for English language. After taking the necessary information, the tool then stores them into a database.

Concepts:

Messages: consists of at least one part. Each part might have a different mime-type. These parts might be Attachment, Plain body, inline attachment, HTML body, or any other type that is specified in the header.

Attachment Monitor: Whenever any message containing attachment is sent to the Invisible Witness, it would pass that message to the attachment monitor. The attachment monitor checks each attachment whether inline or plain attachment and validate each attachment size by calculating the previous user profile. Depending on the average attachment size, the Attachment Monitor would decide if administrator notification has to be sent and logs this activity into the database. If the attachment size is less than the average attachment, Attachment Monitor would log the activity but not notify the Network Administrator.

Dictionary: The dictionary is used by the Invisible Witness to spell check the message body content. This is important for the Invisible Witness to decide whether the message should be flagged or not. The dictionary is an external file containing only words without definition to minimize the file size and improve the system efficiency. Invisible Witness would accept any dictionary containing only a single word per-line.

Administrator: An administrator is also monitored by the Invisible Witness. For example, if upper management tried to send a huge attachment, the Invisible Witness would log that activity because the upper management is also considered employees.

Employee: An employee is any person working for the company, including upper management and administrators.

Mail Server: The mail server is a server that is used to send, receive, and store emails using different protocols. G-mail, Exchange, Lotus Domino are all considered mail servers.

3.2 Use Case Diagram

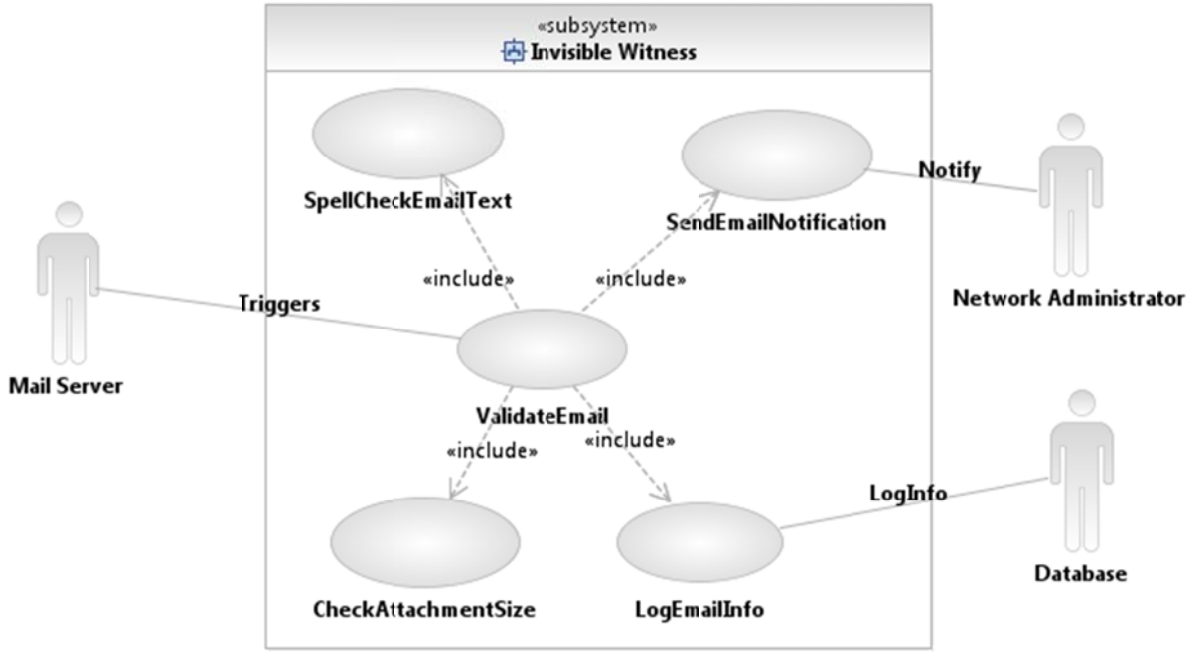


Figure 2, Use Case Diagram

Use Case ID:	IWS01		
Use Case Name:	Invisible Witness		
Created By:	Onur Polatcan	Last Updated By:	Onur Polatcan
Date Created:	11/01/2010	Date Last Updated:	01/08/2011
Actors:	Mail Server, System, Administrator, Database		
Description:	System is triggered when a new email has been sent in order to check that email.		
Trigger:	Mail Server received a message		
Normal Flow:	1. Mail Server triggers the system when a new email has been sent 2. System receives an email from the email server to analyze it 3. System gets a word and pass it to a dictionary 4. Dictionary return true for correct words 5. Flow repeated from 3 for each ne message sent 6. Percentage of misspelled word is under 25% 7. Message does not have any attachment 8. Exit the system		
Alternative Flows:	**6.1 percentage of misspelled words are greater than 25%** 6.1.1 system connect to mail server 6.1.2 mail server acknowledge connection 6.1.3 system prepare and send a message to the administrator **6.2 message does not have a body** 6.2.1 Flow continue from Normal Flow step 7 **7.1 message has attachment** 7.1.1 system request user profile from database 7.1.2 database calculates and returns the average 7.1.3 system compares the attachment size with the average returned from the database 7.1.4 attachment size is less than the average 7.1.5 Flow continue from Normal Flow step 7 **7.1.4.1 attachment size is greater than the average** 7.1.4.1.1 system prepares and send notification to the admin		
Exceptions:	1 Mail Server is down: Display error message 2 Database is down: Display error message		

3.3 Class Diagram

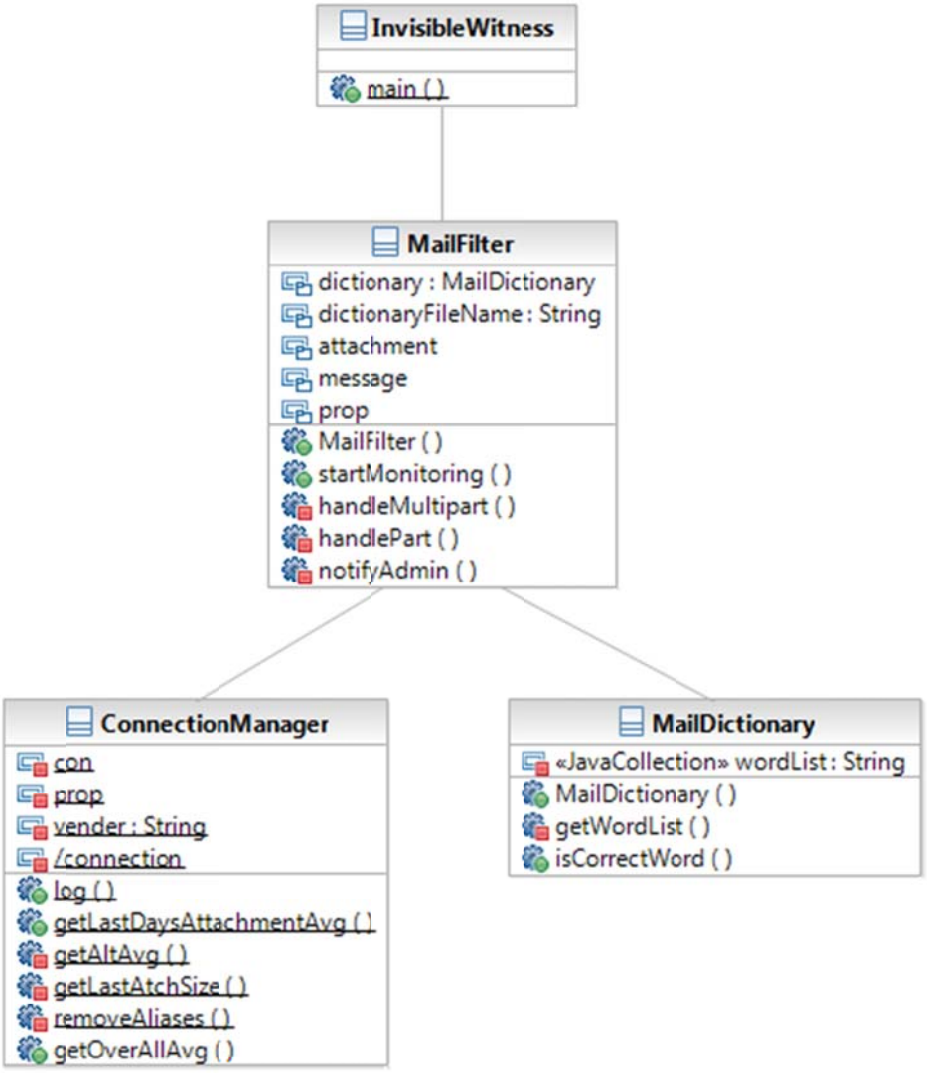


Figure 3, Class Diagram

Methods

MailFilter

This is the main class from which an application is managed. It creates an object from both MailDictionary and ConnectionManager to perform the required tasks. Following is a brief description of its methods:

startMonitoring(): it will connect to the mail server to retrieve the messages that the system has never yet read. Then, it decompose the message to its original parts to call one of the methods handleMultipart() or handlePart().

handleMultipart(): it will be called in case the message has more than one part to be analyzed. It would iterate through all the parts and call handlePart() for each part.

handlePart(): the method will search for the parts with plain/text (message body) and gets the content of that part to be validated. It will use isCorrectWord() from the Dictionary class to validate and count misspelled words. The method will also check whether the part is an attachment. If it is, it will get the average attachment size to compare the current attachment size and log the message if the current attachment size is larger than the average.

notifyAdmin(): The method will notify the administrator if the message exceeds the percentage of misspelled words or if its attachment size exceeds the average.

ConnectionManager:

It contains all the functionalities to communicate with the database. It consists of the following methods:

Log(): When an attachment exceeds the average, the Invisible Witness would use this message to log the attachment information to the database. The method accepts the message itself and the attachment to get enough information to be logged.

For instance, the message is needed to get general information such as sender, receiver, and sent date. The attachment is needed to get more detailed information about the attachment itself, such as the attachment size, name, and type.

getLastDaysAttachmentAvg(): Invisible Witness would get the average of the last logged activity by the user who sent a message containing at least one attachment to validate the attachment. This method will accept two parameters: sender name and time period. Sender name is to retrieve information about the user. The method will use time period in order to calculate the average of the attachment size that are logged during this period. In case the method did not find any activity during the specified period, it would call getAltAvg().

getAltAvg(): This will calculate the average attachment size using the other period specified in the configuration file, which is usually more than the first period. If no activities were found during this period, it would call getLastAtchSize().

getLstAtchSize(): This string would retrieve the size of the last attachment that the specified user sent. In case no profile was found, the method would return 0.

getOverAllAvg(): This method will get the average of all the attachments that the specified user has sent regardless of the attachment size restriction.

removeAliases(): This string removes the aliases from the email addresses. Unlike email addresses, aliases might be changed by the user at any time. So, it is important to remove theses aliases to improve the accuracy of the program.

MailDictionary

Contains all the dictionary related concepts, such as word validation, dictionary loading, and file specification. It consists of the following methods:

MailDictionary(): accepts the dictionary file name to make it easier to change the diction the application uses to validate the message content.

getWordList(): loads all the dictionary words to make them ready. Loading the words in the data structure is much more efficient than iterating a file each time a word is validated.

isCorrectWord: validates whether the passed word already exists as a parameter in the dictionary. If it found the word in the dictionary, it would return true. It would return false otherwise. The Invisible Witness will basically use the isCorrectWord() to validate each word in the body and calculate the average of the misspelled words.

4. Experimental Evaluation

In this section, I present the results of my experimental evaluation of the automatic filtering application, also known as Invisible Witness. Originally, I checked the accuracy of the application each week, for a total of five weeks; however, to make it as easy as possible for the reader, I decided to use only the last two weeks of data. In this case I could easily show and share the data with anyone. The main idea of the experimental evaluation is to measure the accuracy of Invisible Witness. The application should not be judged by the execution time or limited number of sent messages since they are not real-time scenarios. The execution time is not a real-time scenario since the application analyzes any sent message as soon as It Is sent. For this reason the application would not have to analyze a huge number of messages at once.

4.1 Datasets

Originally, I was going to use Enron e-mail corpus, made available by MIT; however, after looking at the given datasets, I learned that none of them have any attachments, and some of the e-mails have no e-mail sender values; therefore, I decided to create my own datasets.

4.2 Test Environment

- Windows 7 Professional 64 bit
- Microsoft Access 2010
- MySQL 5.5
- G-Mail server
- Java Mail Library

4.3 Settings

It must be noted that when setting up a database, you must ensure that enough characters be given for each column, and the administrator should always read the logs if any error message is printed.

4.4 Results

Originally, the Invisible Witness tool was only capturing the attachments that are over-size the average user profile limit and stores them in the database. However, after a debate with the committee, I made a change, and now the application stores all the attachments to a database regardless of the attachment size. I also created another column in the database called "oversize." If the attachment is oversize its limit, the

database display a “one,” but if the attachment is within the parameter, the database will display a “zero.” Each time the network administrator receives an e-mail, there is an informational message line that displays "User up-to-date attachment size," which means that the average attachment size is displayed for the specified user. 4.4.1 Result Section 1 does not include the changes; however, 4.5.1 does.

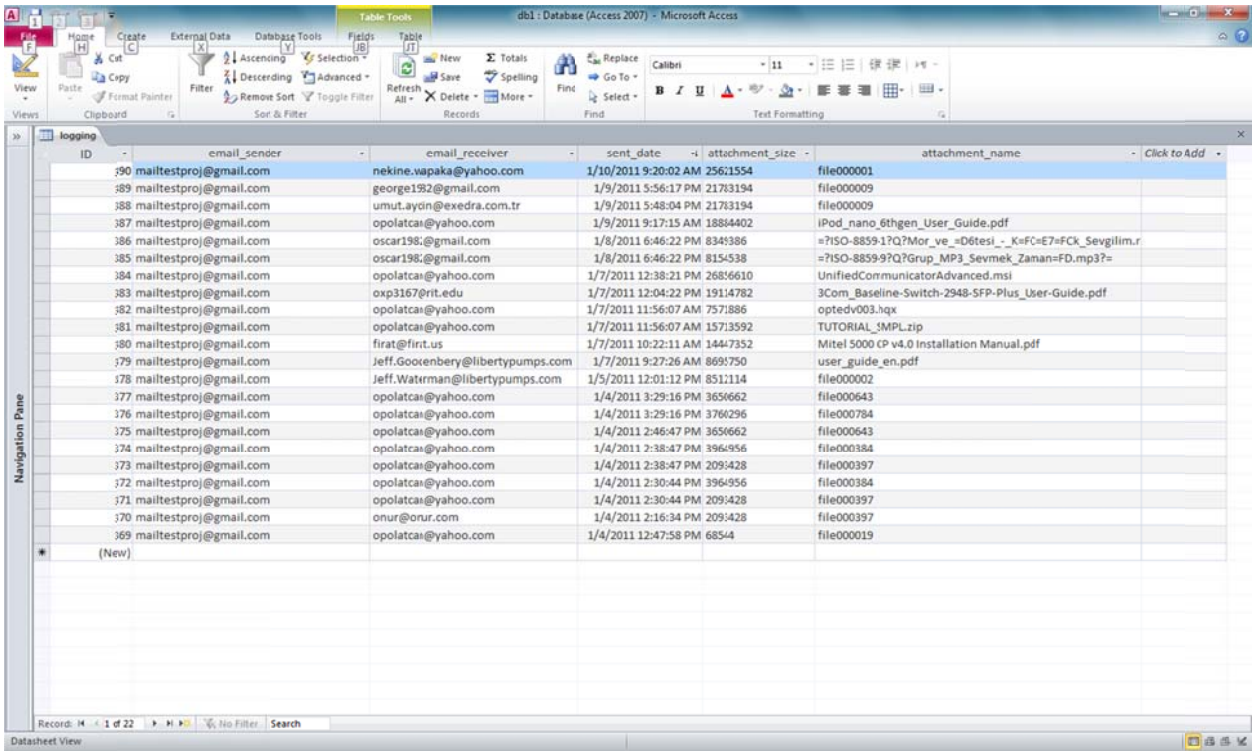
4.4.1 Results Section 1

Table 1, Invisible Witness Results

Date	Normal Traffic	FALSE	Turkish Content	Chinese Content	Russian	Over-size Notifications Sent	Total Attachment	Total Emails
1/4/2011	6	0	2	0	0	9	9	14
1/5/2011	1	0	0	1	0	1	17	3
1/6/2011	0	2	0	0	2	0	0	4
1/7/2011	3	0	2	0	0	6	39	9
1/8/2011	0	0	5	0	0	2	3	6
1/9/2011	3	0	2	0	0	3	8	8
1/10/2011	7	1	1	0	0	1	180	10
Total	20	3	12	1	2	22	256	54

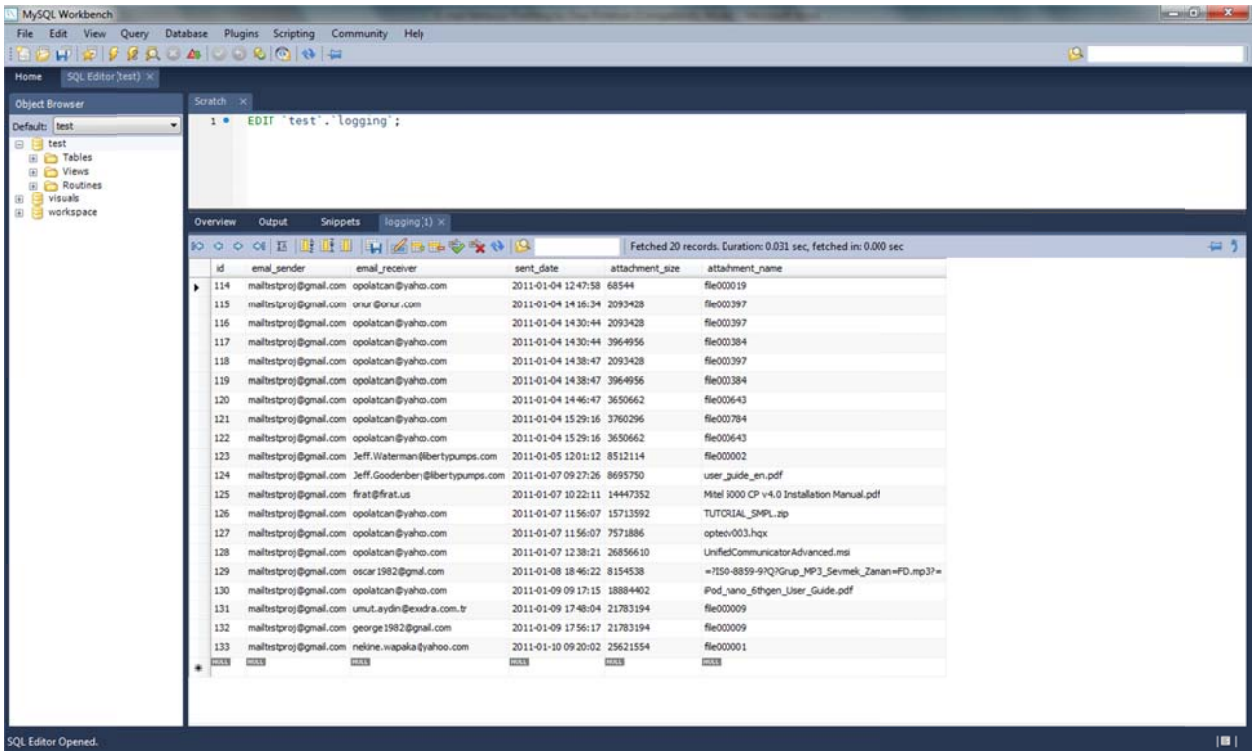
Starting from January 4, 2011, to January 10, 2011, a total of seven days of data has been recorded. The user sent a total of 54 e-mails. It appears that 20 of these e-mails have not been flagged: 3 of them are false positives; 12 of them have Turkish words; 1 email contains Chinese words; 2 e-mails contain pure Russian; and 22 notifications are sent because of a profile attachment size and is logged on the database. Out of 54 emails, a total of 22 e-mails had attachments on them totaling 256 attachments, but only 22 of them were over the profile size.

Table 2, Access Database Screenshot



This is the access database screenshot with the actual logs.

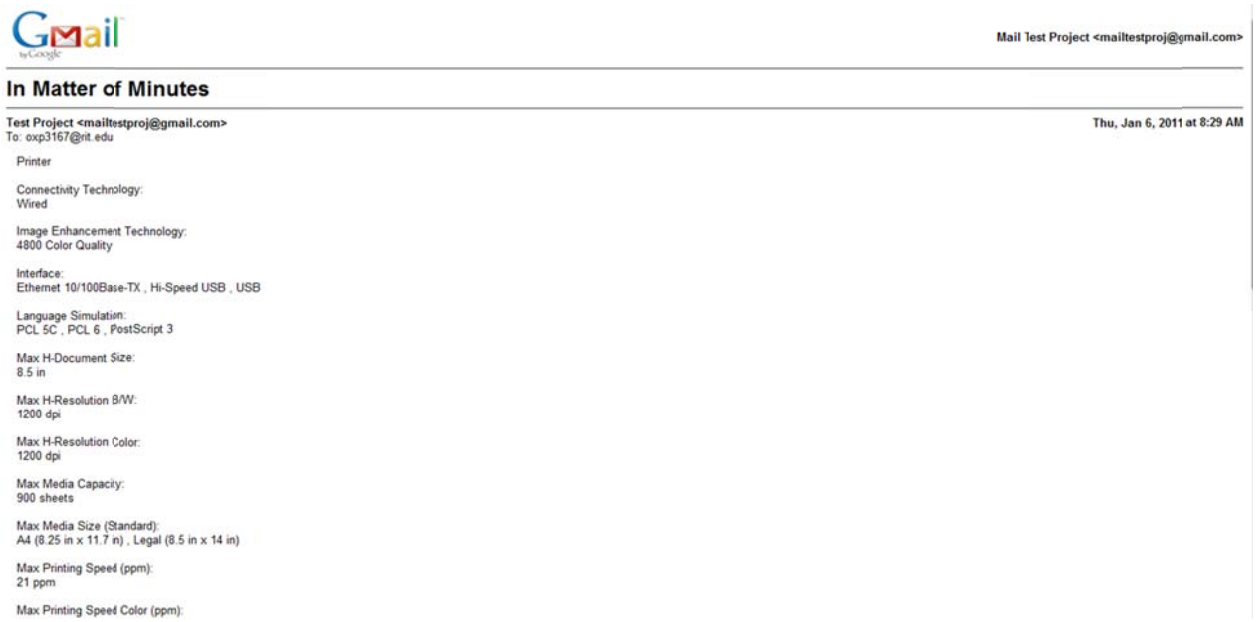
Table 3, MySQL Database Screenshot



This is the screenshot from MySQL server 5.5. If you look closely at the screenshot, there are only 20 logs; where as, Microsoft Access database has 22 logs recorded. This happened because the column accepts only 45 characters, and one of the attachment name has more than 45 characters. Since it cannot store the name of the attachment in that column, it just dumps it, but the administrator would be able to see the error message in the logs.

4.4.2 False Positive

Out of 54 e-mails, only 3 of the emails have false positives, which is less than 6%.



Gmail by Google — Mail Test Project <mailtestproj@gmail.com>

In Matter of Minutes

Test Project <mailtestproj@gmail.com> — **Thu, Jan 6, 2011 at 8:29 AM**
To: oxp3167@rit.edu

Printer

Connectivity Technology:
Wired

Image Enhancement Technology:
4800 Color Quality

Interface:
Ethernet 10/100Base-TX , Hi-Speed USB , USB

Language Simulation:
PCL 5C , PCL 6 , PostScript 3

Max H-Document Size:
8.5 in

Max H-Resolution B/W:
1200 dpi

Max H-Resolution Color:
1200 dpi

Max Media Capacity:
900 sheets

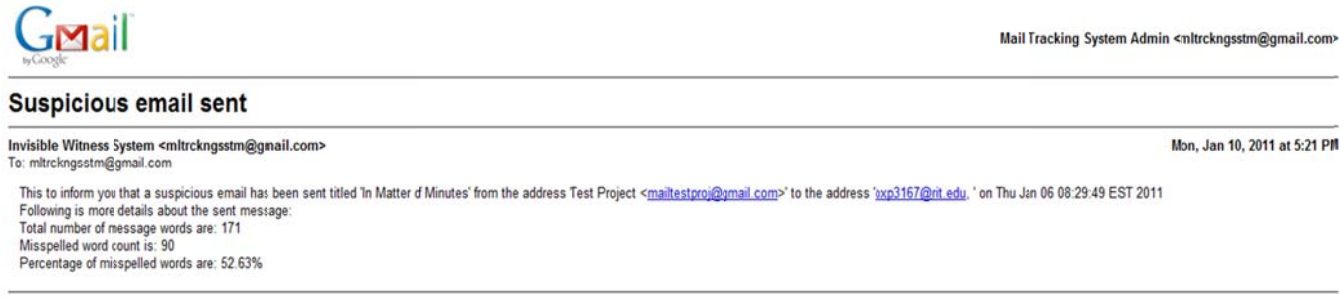
Max Media Size (Standard):
A4 (8.25 in x 11.7 in) , Legal (8.5 in x 14 in)

Max Printing Speed (ppm):
21 ppm

Max Printing Speed Color (ppm):

False Positive 1

This e-mail could not be validated because it contains many technical words. The network administrator should add these technical words if this particular user is using these words daily. Below is what Invisible Witness sent to acknowledge the network administrator.



Gmail by Google — Mail Tracking System Admin <mltrckngsstm@gmail.com>

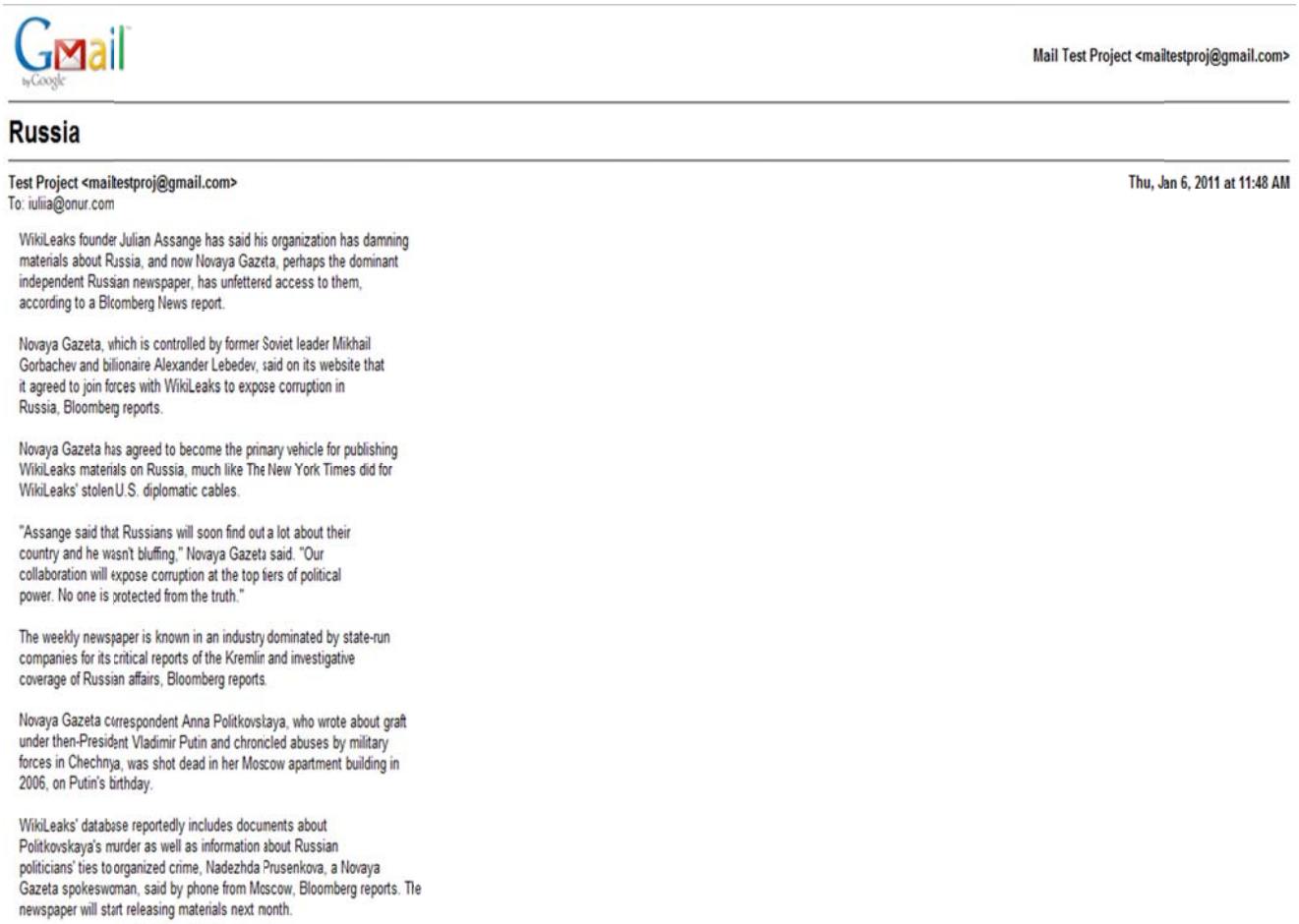
Suspicious email sent

Invisible Witness System <mltrckngsstm@gmail.com> — **Mon, Jan 10, 2011 at 5:21 PM**
To: mltrckngsstm@gmail.com

This to inform you that a suspicious email has been sent titled 'In Matter d Minutes' from the address Test Project <mailtestproj@gmail.com>' to the address 'oxp3167@rit.edu. ' on Thu Jan 06 08:29:49 EST 2011
Following is more details about the sent message:
Total number of message words are: 171
Misspelled word count is: 90
Percentage of misspelled words are: 52.63%

Notification E-Mail 1

Out of 171 words, 90 of them were not included in the dictionary, which gives more than 52%, so the network administrator was informed.



Gmail by Google

Mail Test Project <mailtestproj@gmail.com>

Russia

Test Project <mailtestproj@gmail.com>
To: iuliia@onur.com

Thu, Jan 6, 2011 at 11:48 AM

WikiLeaks founder Julian Assange has said his organization has damning materials about Russia, and now Novaya Gazeta, perhaps the dominant independent Russian newspaper, has unfettered access to them, according to a Bloomberg News report.

Novaya Gazeta, which is controlled by former Soviet leader Mikhail Gorbachev and billionaire Alexander Lebedev, said on its website that it agreed to join forces with WikiLeaks to expose corruption in Russia, Bloomberg reports.

Novaya Gazeta has agreed to become the primary vehicle for publishing WikiLeaks materials on Russia, much like The New York Times did for WikiLeaks' stolen U.S. diplomatic cables.

"Assange said that Russians will soon find out a lot about their country and he wasn't bluffing," Novaya Gazeta said. "Our collaboration will expose corruption at the top tiers of political power. No one is protected from the truth."

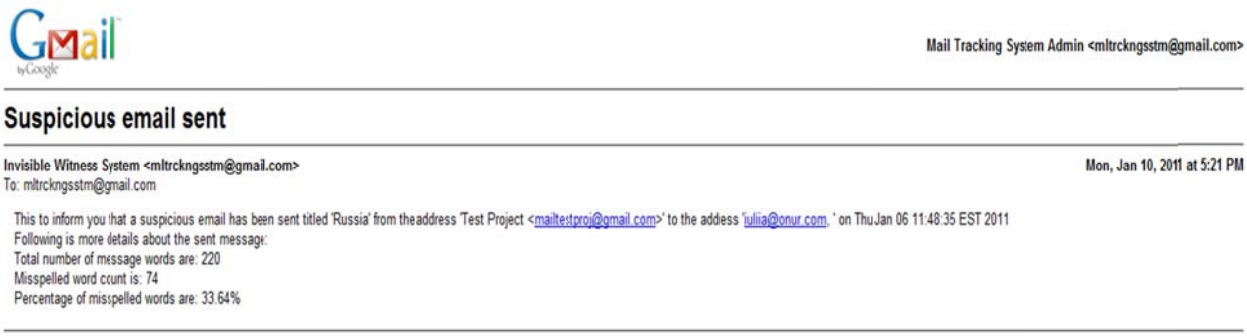
The weekly newspaper is known in an industry dominated by state-run companies for its critical reports of the Kremlin and investigative coverage of Russian affairs, Bloomberg reports.

Novaya Gazeta correspondent Anna Politkovskaya, who wrote about graft under then-President Vladimir Putin and chronicled abuses by military forces in Chechnya, was shot dead in her Moscow apartment building in 2006, on Putin's birthday.

WikiLeaks' database reportedly includes documents about Politkovskaya's murder as well as information about Russian politicians' ties to organized crime, Nadezhda Prusenkova, a Novaya Gazeta spokeswoman, said by phone from Moscow, Bloomberg reports. The newspaper will start releasing materials next month.

False Positive 2

This e-mail is pure English but has many names that are not included in the dictionary. It talks about a Russian newspaper and a Soviet leader. The network administrator was notified by Invisible Witness and a screenshot of this e-mail is below.



Mail Tracking System Admin <mltrckngsstm@gmail.com>

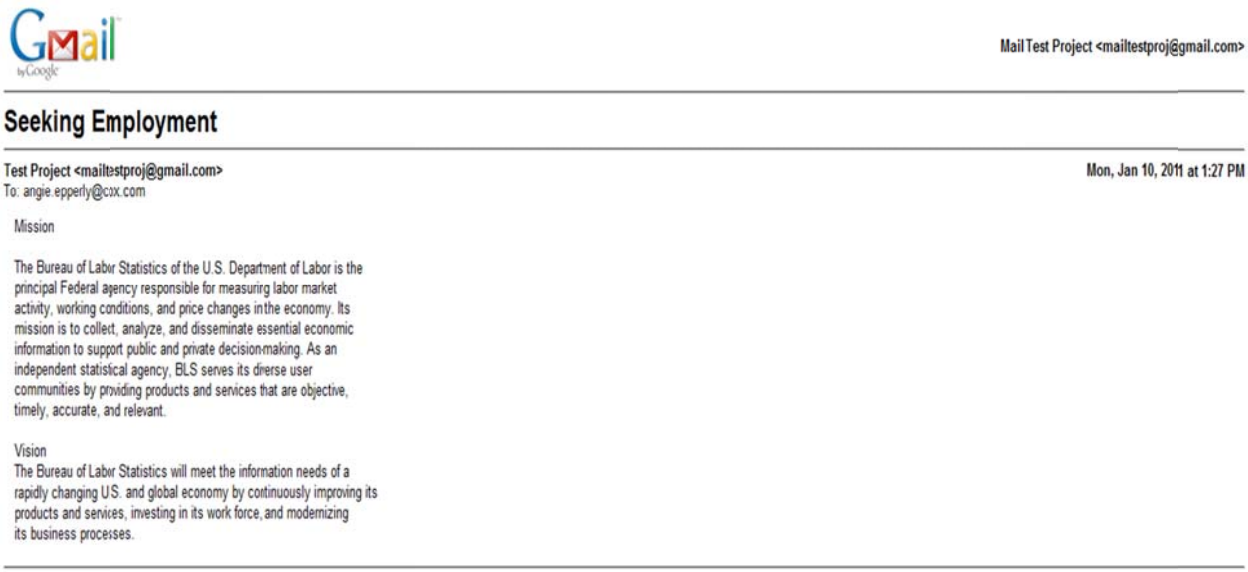
Suspicious email sent

Invisible Witness System <mltrckngsstm@gmail.com> **Mon, Jan 10, 2011 at 5:21 PM**
To: mltrckngsstm@gmail.com

This to inform you that a suspicious email has been sent titled 'Russia' from theaddress 'Test Project <mailtestproj@gmail.com>' to the address 'iuliia@onur.com, ' on ThuJan 06 11:48:35 EST 2011
Following is more details about the sent message:
Total number of message words are: 220
Misspelled word count is: 74
Percentage of misspelled words are: 33.64%

Notification E-Mail 2

This proves that more than 33% of the words are not included in the dictionary.



Mail Test Project <mailtestproj@gmail.com>

Seeking Employment

Test Project <mailtestproj@gmail.com> **Mon, Jan 10, 2011 at 1:27 PM**
To: angie.epperly@cox.com

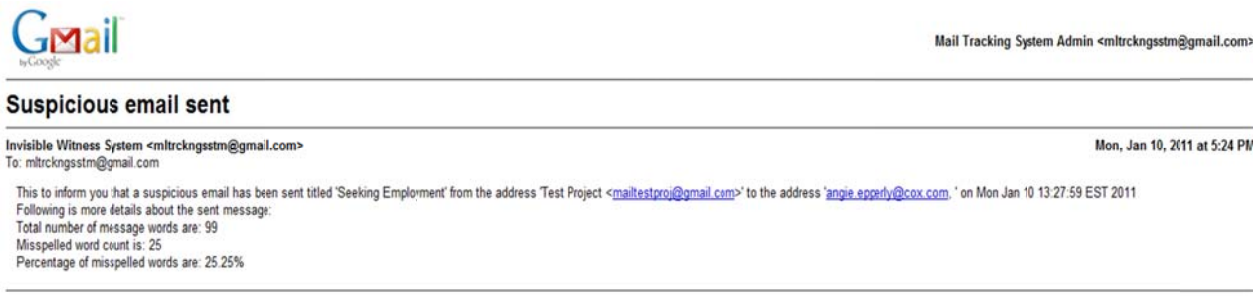
Mission

The Bureau of Labor Statistics of the U.S. Department of Labor is the principal Federal agency responsible for measuring labor market activity, working conditions, and price changes in the economy. Its mission is to collect, analyze, and disseminate essential economic information to support public and private decision-making. As an independent statistical agency, BLS serves its diverse user communities by providing products and services that are objective, timely, accurate, and relevant.

Vision
The Bureau of Labor Statistics will meet the information needs of a rapidly changing U.S. and global economy by continuously improving its products and services, investing in its work force, and modernizing its business processes.

False Positive 3

This e-mail contains some abbreviations and words that are not included in the dictionary. The network administrator was alerted anyway. More information about how to avoid false positive is given in the “limitation” section of this paper.



Mail Tracking System Admin <mltrckngsstm@gmail.com>

Suspicious email sent

Invisible Witness System <mltrckngsstm@gmail.com> Mon, Jan 10, 2011 at 5:24 PM
To: mltrckngsstm@gmail.com

This to inform you that a suspicious email has been sent titled 'Seeking Employment' from the address 'Test Project <mailtestproj@gmail.com>' to the address 'angie.epperly@cox.com. ' on Mon Jan 10 13:27:59 EST 2011
Following is more details about the sent message:
Total number of message words are: 99
Misspelled word count is: 25
Percentage of misspelled words are: 25.25%

Notification E-Mail 3

Out of 99 words, 25 words were not included in the dictionary; therefore, the network administrator received this e-mail above.

4.5.1 Results 2

Here is another set of data after the code modification, which will prove the accuracy of the application.

Table 4, Invisible Witness Results

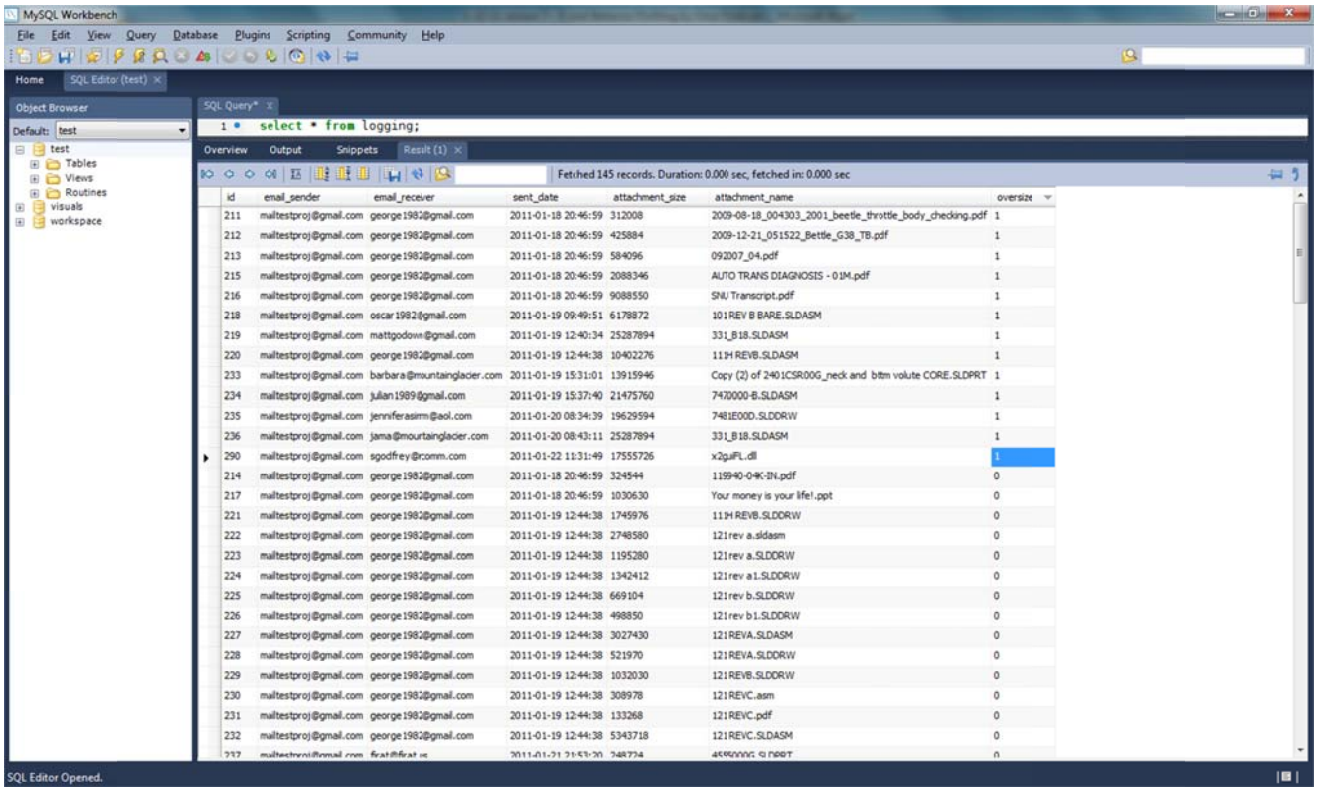
	A	B	C	D	E	F	G	H	I	J	K
1	Date	Normal Traffic	FALSE	Turkish Content	Spanish	Portuguese	Chinese Content	Russian	Over-size Notifications Sent	Total Attachment	Total Emails
2	1/18/2011	0	0	0	0	0	0	0	5	7	1
3	1/19/2011	0	0	1	0	0	0	0	5	17	6
4	1/20/2011	2	0	0	0	0	0	0	2	2	4
5	1/21/2011	1	0	1	0	0	2	2	0	37	6
6	2/22/2011	5	0	2	1	1	1	1	1	34	11
7	1/23/2011	5	1	3	1	0	0	0	0	48	10
8	Total	13	1	7	2	1	3	3	13	145	38

Starting from January 18, 2011, to January 23, 2011, a total of six days of data has been recorded. The user sent a total of 38 e-mails. It appears that 13 of these e-mails have not been flagged: 1 is a false positive; 7 of them have Turkish words; 3 e-mails contain Chinese words; 3 e-mails contain pure Russian; 1 email is pure Portuguese; 2 e-mails are Spanish; and 13 notifications are sent because of a profile attachment size and are logged on the database, and it flagged 1 on the “oversize” column. Out of 38 emails, a total of 21 e-mails had attachments on them totaling 145

attachments, but only 13 of them were over the profile size. When examining the table above, keep in mind that each e-mail can have multiple parts.

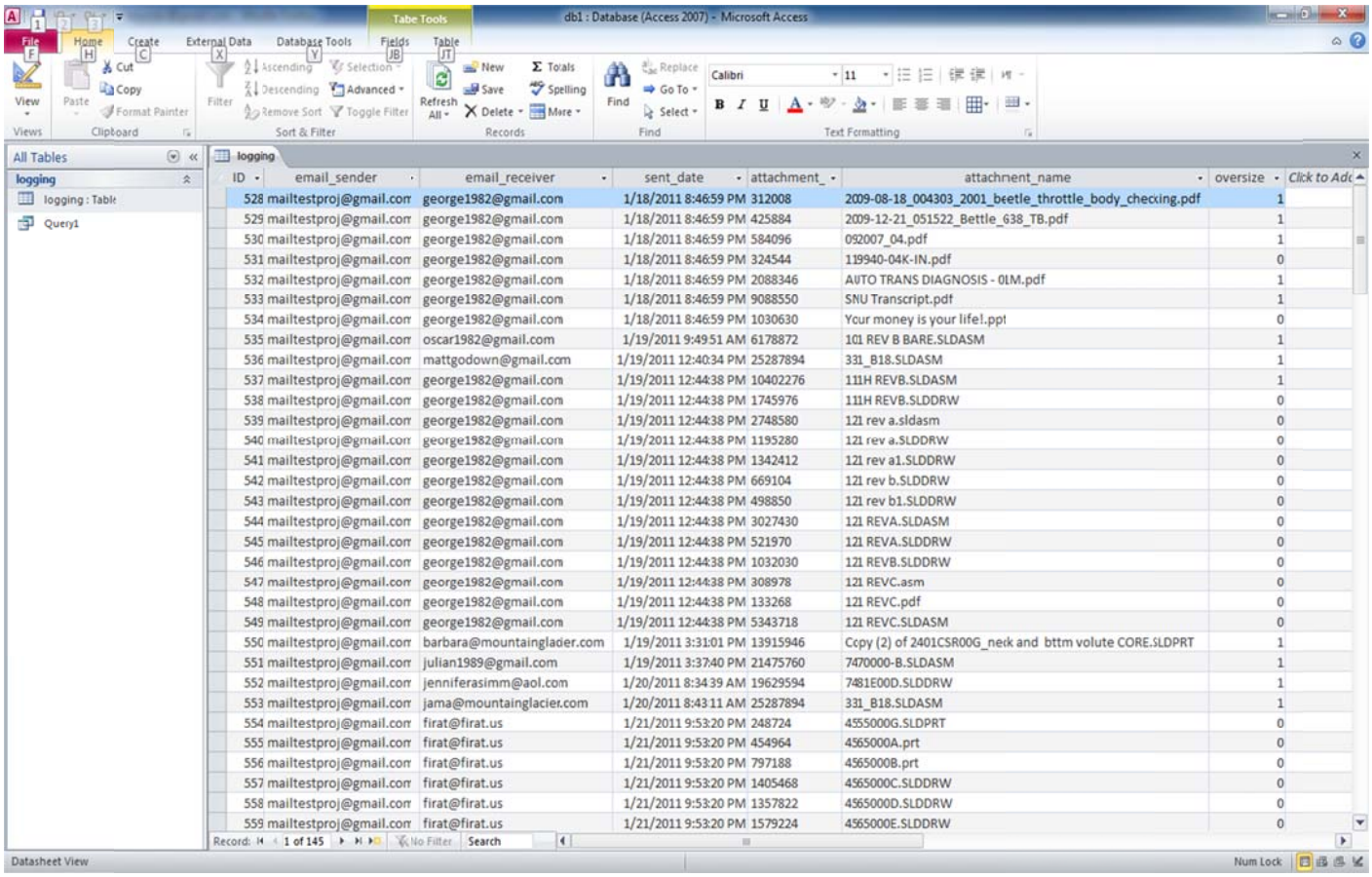
MySQL Database

Table 5, MySQL Database Screenshot



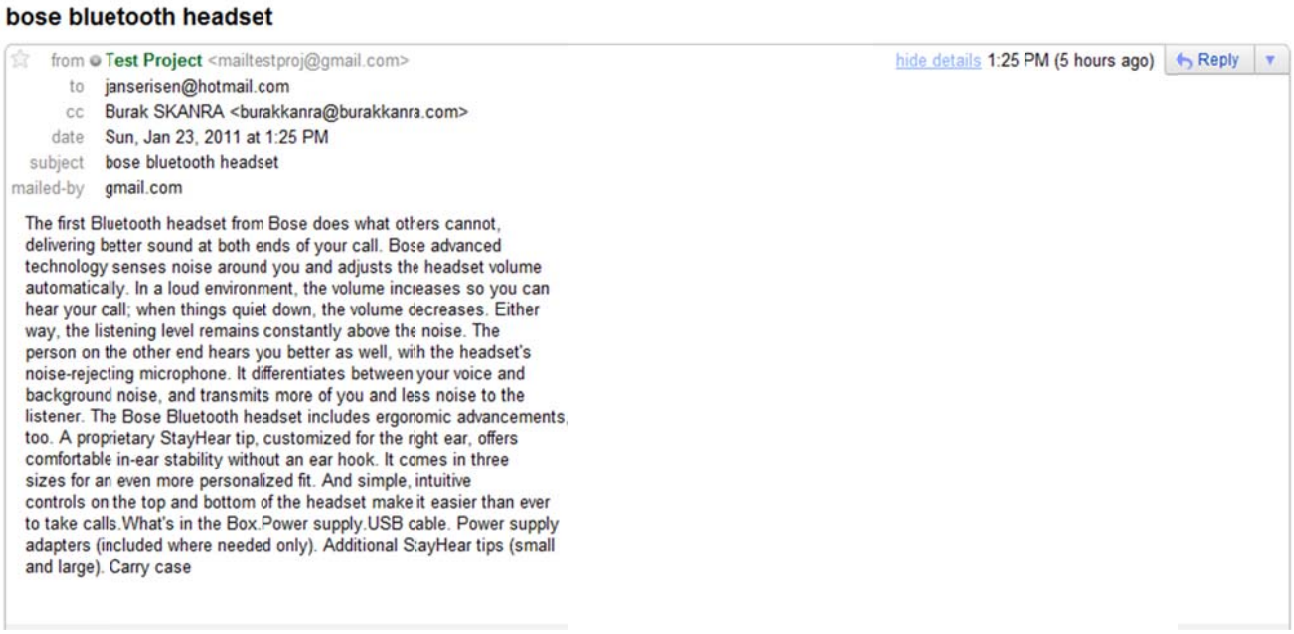
There are total of 13 profile allowed oversized attachments sent by the user. This behavior reported to Network Administrator via e-mail instantly. According to screenshot above 145 total attachments recorded.

Table 6, Access Database Screenshot



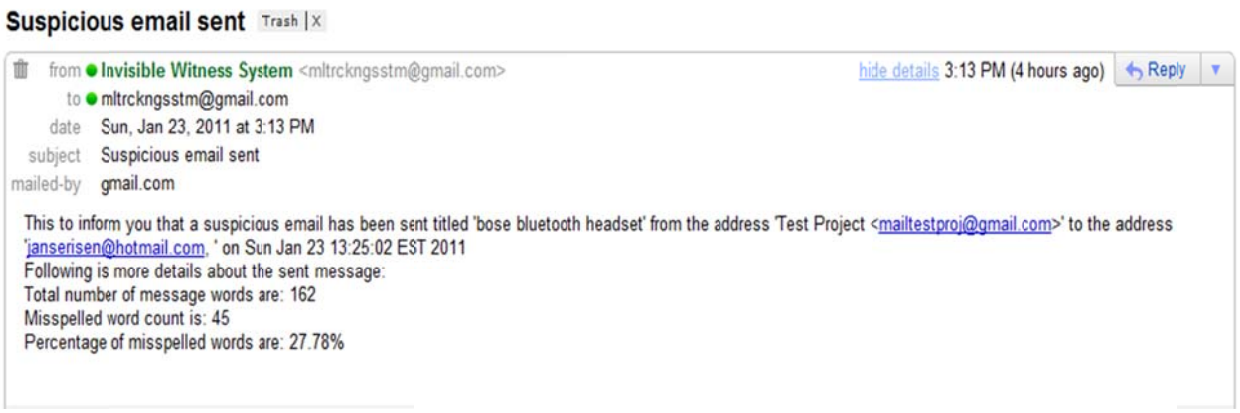
The above screenshot represents the same data results as MySQL. This application is capable of storing the data in either Microsoft Access or MySQL.

4.5.2 False Positive



False Positive 4

Out of 38 emails, this was the only false positive captured in this data set, which is less than 2 percent. There were three words that weren't included in the dictionary, which made this data set a false positive. The network administrator should closely observe the dictionary for undefined words to avoid future problems.



Notification E-Mail 4

The above e-mail had only 162 words, but 27 percent of the words were not included in the dictionary; therefore, Invisible Witness reported this incident to the network administrator.

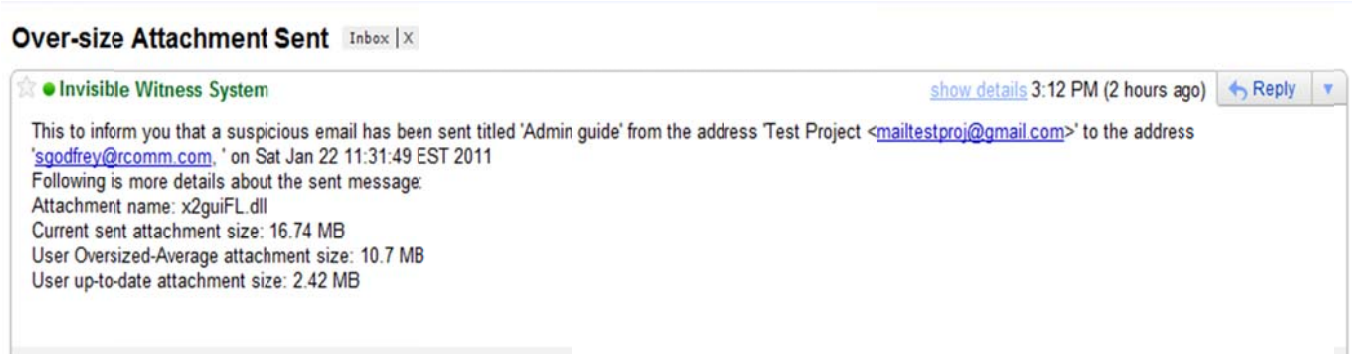


Figure 4, Over size attachment

This is the last alert received from Invisible Witness in this time period for the over-size attachment. According to the above screenshot, the current attachment size exceeds the oversized-average attachment size; therefore, the network administrator was notified. When one closely looks at the attachment carefully, it displays the current sent attachment size, the user oversized-average size, and the user up-to-date attachment size.

5. Conclusion

I present a robust and scalable application for filtering confidential data. This application can capture the differing distrubitions of e-mails in users' sent boxes. This application can be used for one or more user, if needed.

Over the years, most companies have conducted and will continue to conduct a significant portion of their business through e-mail. Some of these information transmitted by e-mail include confidential data that, if not properly secured, could damage the company's security severely. The significant risks that all companies face are when employees within the organization send confidential information through unprotected e-mails. I hope that organizations will be able to use the "Invisible Witness" tool to minimize its business risk to help protect valuable company information.

6. Future Work

This thesis is prove that a concept is needed to form the basis for possible testing in the near future. I believe this application can be improved by including functionality, performance, effectiveness, as well. The sections below will describe some of the improvements that can be done in near future.

6.1 Limitation

One of the biggest challenging parts was Gmail's IMAP. Gmail doesn't handle all of the standard IMAP flags, such as "\Recent". For this reason I had to use a work-around. It is normal behavior when an employee types an email, but it is automatically flagged as "read" by the Gmail server. Since the recent feature didn't work, I used a "seen" flag to get the data and marked the read message as "unseen" to check only the new messages that the system had not yet read.

Even though five weeks of data has been used with over 95 percent accuracy, further testing should be done to improve effectiveness. The dictionary that is used in this test bed has over 237,000 words in it. When an employee uses a word that is not in the dictionary, the system flags it, and for this reason, when the word is not included, the administrator should add the new words to get better results.

The connection to Gmail server will be closed after sending a specific number of messages. The total number of messages I could get per connection was 179 messages. The error message I received from Gmail was "550 5.4.5 Daily sending quota exceeded." They do not allow sending more than 500 messages per day, according to their gudelines, but I sent 800+ messages before I got this message. This is automatically done by Google to protect users from spammers. I also tried to make several connections from work, home, and the RIT campus, but I was refused since Gmail currently has a limit of 10 simultaneous IMAP connections per account. When I was using port 25 for an SMTP connection using a TLS at work, it was throwing connection refused errors, then I changed to port 587, and the application started to behave normally. This happened because of Cisco's ASA 5510 firewall.

6.2 Functionality

I believe this application can be advanced and used as commercial software sometime in the near future. There are four modifications that should be applied to advance it:

1. The mail server should push messages rather than the system pulls messages.

I would make the application to be triggered by the mail server whenever a message has been sent. Currently, the application pulls messages from the mail server. The computer resources usage will be limited to when the application is triggered.

2. Data representation

The system could be expanded to display graphically the users' profiles as a pie or bar chart to depict what is already logged. Currently, the system uses only the logged data to calculate the average profile of users, but these data could be used for many more different presentations.

3. Language add-on

The system could be expanded to add more language dictionaries so that it would identify the text based on the available language that is loaded on the system. For instance, Spanish, French, Chinese and so forth.

4. Attachment Analyzer

System could be expanded to analyze the attachments and determine what type of extensions they have or could be iterated through the hexadecimal object file format.

Appendix

A. Source Code – ConnectionManager.java

```
import java.io.File;
import java.io.FileInputStream;
import java.net.URL;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Timestamp;
import java.util.Properties;

import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Part;

public class ConnectionManager {
     private static Connection con = null;// holds the
connection object
     private static Properties prop = null; // properties for
the databases
     private static String vender = null; // venders for the
databases

     /**
      * store the message information in the database
      *
      * @param message
      *            to get information from
      * @param attachment
      *            to get the attachment information
      * @throws MessagingException
      */
     public static void log(Message message, Part attachment,
boolean isOverSize)
               throws MessagingException {
          // get the message sender
          String sender = message.getFrom()[0].toString();
          // remove the alias associated with the sender
          sender = removeAliases(sender);

          // get the receiver
          String receiver =
message.getRecipients(Message.RecipientType.TO)[0]
                    .toString();
          // remove the alias associated with the receiver
```

```
            receiver = removeAliases(receiver);
            // get the date when the message was sent
            Timestamp timeStamp = new
Timestamp(message.getSentDate().getTime());
            // get the attachment size and name
            double attachmentSize = attachment.getSize();
            String attachmentName = attachment.getFileName();

            try {
                  // get a connection object
                  con = getConnection();
                  // prepare a statement to insert the information
in the database
                  PreparedStatement insertStmt = null;
                  String insertQuery = "INSERT INTO logging (" +
"email_sender, "
                              + "email_receiver, " + "sent_date, " +
"attachment_size, "
                              + "attachment_name, oversize)" +
"VALUES(? , ?,?,?,?,?)";
                  // set the missing values to fill the prepared
statement
                  insertStmt = con.prepareStatement(insertQuery);
                  insertStmt.setString(1, sender);
                  insertStmt.setString(2, receiver);
                  insertStmt.setTimestamp(3, timeStamp);
                  insertStmt.setDouble(4, attachmentSize);
                  insertStmt.setString(5, attachmentName);
                  if (isOverSize) {
                        insertStmt.setInt(6, 1);
                  } else {
                        insertStmt.setInt(6, 0);
                  }
                  // execute the query
                  insertStmt.executeUpdate();

                  // close the statement and the connection
                  insertStmt.close();
                  con.close();

            } catch (SQLException e) {
                  e.printStackTrace();
            }

      }

      /**
       * calculate the average of the period specified with the
number of days
```

```
	 * specified in the property file
	 *
	 * @param sender
	 *            the email address of the sender
	 * @param numberOfDays
	 *            specify the number of days to get the average
from
	 * @param alternativeNumberOfDays
	 *            alternative number of days in case no records
were found
	 * @return the average of the attachment size within the
specified period
	 */
	public static double getLastDaysAttachmentAvg(String
sender,
			int numberOfDays, int alternativeNumberOfDays) {
		double avg = 0;
		try {
			// get the sender's email address
			sender = removeAliases(sender);
			// get a connection object
			con = getConnection();

			String dateAdd = "";
			if (vender.equals("access")) {
				dateAdd = "DATEADD('d', ?, NOW())";
			} else if (vender.equals("mysql")) {
				dateAdd = "DATE_ADD(NOW(), INTERVAL ? DAY)";
			}

			// prepare a select statement
			PreparedStatement selectStmt = null;
			ResultSet rs;
			/*
			 * multiply the number of specified date by '-1'
in order to
			 * calculate the starting date of the period. For
instance, if the
			 * number of days specified in the property file
were 7, it would be
			 * converted to '-7' which is 7 days ago
			 */
			numberOfDays = numberOfDays * -1;
			String selectQuery = "SELECT AVG(attachment_size)
as average "
					+ "FROM logging " + "WHERE sent_date >
" + dateAdd
					+ "AND email_sender = ? AND
oversize=?";
```

```
                selectStmt = con.prepareStatement(selectQuery);
                selectStmt.setInt(1, numberOfDays);
                selectStmt.setString(2, sender);
                selectStmt.setInt(3, 1); // 1 indicates oversize
attachment only

                // execute the query
                rs = selectStmt.executeQuery();
                avg = 0;
                if (rs.next()) {
                     avg = rs.getDouble("average");
                }

                if (avg == 0) {// there was no record found
                                    // try the alternative 7 day
period
                     avg = getAltAvg(sender,
alternativeNumberOfDays);
                     System.out.println("ALT AVG is: " + avg);
                }

                selectStmt.close();
                con.close();

           } catch (SQLException e) {
                e.printStackTrace();
           }

           return avg;
     }

     /**
      * called by 'getLastDaysAttachmentAvg()' to get the
average of the
      * alternative period specified in the configuration file
      *
      * @param sender
      *            the email from which the message was sent
      * @param altNumOfDays
      *            the alternative period
      * @return the average of the alternative period
      */
     private static double getAltAvg(String sender, int
altNumOfDays) {
           double avg = 0;
           try {

                con = getConnection();
```

```
                PreparedStatement selectStmt = null;
                ResultSet rs;
                altNumOfDays = altNumOfDays * -1;

                String dateAdd = "";
                if (vender.equals("access")) {
                     dateAdd = "DATEADD('d', ?, NOW())";
                } else if (vender.equals("mysql")) {
                     dateAdd = "DATE_ADD(NOW(), INTERVAL ? DAY)";
                }

                String selectQuery = "SELECT AVG(attachment_size)
as average "
                          + "FROM logging " + "WHERE sent_date >
" + dateAdd
                          + "AND email_sender = ? AND
oversize=?";

                selectStmt = con.prepareStatement(selectQuery);
                selectStmt.setInt(1, altNumOfDays);
                selectStmt.setString(2, sender);
                selectStmt.setInt(3, 1); // 1 indicates oversize
attachment only
                rs = selectStmt.executeQuery();
                avg = 0;
                if (rs.next()) {
                     // get the alternative 7 day average from
the database
                     avg = rs.getDouble("average");
                }

                if (avg == 0) {// there was no record found
within the alternative
                                    // 7 day period
                                    // get the size of the last
attachment
                     avg = getLastAtchSize(sender);
                }

                // close the statement and connection
                selectStmt.close();
                con.close();

           } catch (SQLException e) {
                e.printStackTrace();
           }

           return avg;
     }
```

```
     /**
      * retrieve the last attachment that the specified sender
sent
      *
      * @param sender
      *            the user email
      * @return the size of the last attachment
      */
     private static double getLastAtchSize(String sender) {
          double lastAttachmentSize = 0;
          try {
               // get a connection
               con = getConnection();
               PreparedStatement selectStmt = null;
               ResultSet rs;

               String selectQuery = "";
               if (vender.equals("access")) {

                    selectQuery = "SELECT top 1 id,
attachment_size , MAX(sent_date), MAX(id) "
                              + "FROM logging "
                              + "Where email_sender =? AND
oversize=? "
                              + "GROUP BY attachment_size,
logging.id "
                              + "ORDER BY logging.id desc;";
               } else if (vender.equals("mysql")) {
                    selectQuery = "SELECT id, attachment_size ,
MAX(sent_date), MAX(id) "
                              + "FROM logging "
                              + "Where email_sender =? AND
oversize=? "
                              + "GROUP BY attachment_size,
logging.id "
                              + "ORDER BY logging.id desc limit
1;";
               }

               // prepare a statement
               selectStmt = con.prepareStatement(selectQuery);
               selectStmt.setString(1, sender);
               selectStmt.setInt(2, 1); // 1 indicates oversize
attachments only
               rs = selectStmt.executeQuery();

               if (rs.next()) {
```

```
                    // get the size of the last attachment from
the database
                    lastAttachmentSize =
rs.getDouble("attachment_size");
               }
               // close the connection and the statement
               selectStmt.close();
               con.close();

          } catch (SQLException e) {
               e.printStackTrace();
          }

          return lastAttachmentSize;

     }

     /**
      * lazy initialize a connection if it is not already
Initialized or closed.
      *
      * @return a connection object
      * @throws SQLException
      */
     private static Connection getConnection() throws
SQLException {
          try {

               if (con == null || con.isClosed()) {
                    // load the property file to know which
database to connect to
                    prop = new Properties();
                    prop.load(new
FileInputStream("dbconfig.properties"));

                    vender = prop.getProperty("vender");

                    if (vender.equals("access")) {
                         // get the application path for
databases
                         ConnectionManager tst = new
ConnectionManager();
                         URL u =
tst.getClass().getProtectionDomain()

     .getCodeSource().getLocation();
                         File f = new File(u.toURI());
```

```
                        String dbFileName =
prop.getProperty("fileName");

     Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
                        String url =
"jdbc:odbc:Driver={Microsoft Access Driver "
                                  + "(*.mdb, *.accdb)};DBQ=" +
f.getParent() + "\\"
                                  + dbFileName;
                        con = DriverManager.getConnection(url);
                   }// end access connection
                   else if (vender.equals("mysql")) {
                        String userName =
prop.getProperty("userName");
                        String password =
prop.getProperty("password");
                        String host = prop.getProperty("host");
                        String schema =
prop.getProperty("schema");
                        String url = "jdbc:mysql://" + host +
"/" + schema;

     Class.forName("com.mysql.jdbc.Driver").newInstance();
                        con = DriverManager.getConnection(url,
userName, password);
                   }

              }
         } catch (Exception exp) {
              exp.printStackTrace();
         }
         return con;
     }

     /**
      * remove the aliases from email addresses. Aliases comes
before the actual
      * email addresses. The actual email address is inclosed by
'<>' signs
      *
      * @param address
      *            with alias to be removed
      * @return the actual email address
      */
     private static String removeAliases(String address) {
          int index = address.indexOf('<');// find the beginning
of the email
                                                              //
address
```

```
          if (index != -1) {// the address includes aliases
                                   // remove the aliases
               address = address.substring(address.indexOf('<')
+ 1,
                         address.lastIndexOf('>'));
          }
          return address;
     }

     /**
      * calculate the overall average of the attachment size
that user has ever
      * sent
      *
      * @param sender
      *            the email address of the sender
      * @return the overall average of the attachment size
      */
     public static double getOverAllAvg(String sender) {
          double avg = 0;
          try {
               // get the sender's email address
               sender = removeAliases(sender);
               // get a connection object
               con = getConnection();

               // prepare a select statement
               PreparedStatement selectStmt = null;
               ResultSet rs;

               String selectQuery = "SELECT AVG(attachment_size)
as average "
                         + "FROM logging " + "WHERE email_sender
=?";

               selectStmt = con.prepareStatement(selectQuery);
               selectStmt.setString(1, sender);

               // execute the query
               rs = selectStmt.executeQuery();
               avg = 0;
               if (rs.next()) {
                    avg = rs.getDouble("average");
               }
               System.out.println("up-to-date AVG is: " + avg);

               selectStmt.close();
               con.close();
```

```
            } catch (SQLException e) {
                e.printStackTrace();
            }

            return avg;
        }

}
```

B. Source Code – InvisibleWitness.Java

```
/**
 * This project is to monitor email activities happening
periodically
 *
 * @author polatcan
 *
 */
public class InvisibleWitness {

    /**
     * Just starts the program
     *
     * @param args
     */
    public static void main(String[] args) {
        try {
            MailFilter filter = new
MailFilter("dictionary.txt");
            filter.startMonitorint();

        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }

}
```

C. Source Code – MailDictionary.java

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;

/**
 * This class is used to retrieve the dictionary information
from a dictionary.
 * It is also used to look-up words.
 *
 * @author polatcan
 *
 */
public class MailDictionary {
     private ArrayList<String> wordList;// holds all the
dictionary words

     public MailDictionary(String fileName) throws IOException {
          wordList = new ArrayList<String>();
          getWordList(fileName);// load the array
     }

     /**
      * gets a list containing all the words from the given
dictionary
      *
      * @return a list of all the dictionary words
      * @throws IOException
      */
     private void getWordList(String fileName) throws
IOException {
          String word;
          File file = new File(fileName);
          BufferedReader bfreader = new BufferedReader(new
InputStreamReader(
                    new FileInputStream(file)));
          /*
           * iterate through all words in the file and add each
word to the array
           */
          while ((word = bfreader.readLine()) != null) {
               wordList.add(word.toLowerCase());
          }

     }
```

```
     /**
      * check whether the word already exists in the dictionary
      *
      * @param word
      *            to search for
      * @return true:if the word was found in the dictionary.
false:otherwise
      */
     public boolean isCorrectWord(String word) {

          word = word.toLowerCase();
          word = word.trim();
          if (word.equalsIgnoreCase("")) {
               return true;
          } else {
               return wordList.contains(word);
          }
     }
}
```

D. Source Code – MailFilter.java

```
import java.io.FileInputStream;
import java.io.IOException;
import java.text.DecimalFormat;
import java.util.Properties;
import java.util.StringTokenizer;

import javax.mail.Address;
import javax.mail.BodyPart;
import javax.mail.Flags;
import javax.mail.Flags.Flag;
import javax.mail.Folder;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Multipart;
import javax.mail.Part;
import javax.mail.Session;
import javax.mail.Store;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeBodyPart;
import javax.mail.internet.MimeMessage;
import javax.mail.internet.MimeMultipart;
import javax.mail.search.FlagTerm;
```

```java
/**
 * Handles all the interaction with the mail server and has
methods to handle
 * multiple and single part messages.
 *
 * @author polatcan
 *
 */
public class MailFilter {

	MailDictionary dictionary;
	String dictionaryFileName;
	Part attachment;
	Message message;
	Properties prop;

	/**
	 * construct the object and initializes the dictionary file
name from which
	 * the words will be compared
	 *
	 * @param dictionaryFileName
	 * @param maxAttachmentSize
	 *            maximum allowed attachment size in MB
	 */
	public MailFilter(String dictionaryFileName) {
		this.dictionaryFileName = dictionaryFileName;
		this.attachment = null;
		prop = null;

	}

	/**
	 * handle all the interaction with the mail server
	 *
	 * @throws Exception
	 */
	public void startMonitorint() throws Exception {
		// create an object of the dictionary class to
		// spell check the message body
		dictionary = new MailDictionary(dictionaryFileName);

		// load the property file with whatever in the
configuration
		// file 'config.properties'
		prop = new Properties();
		prop.load(new FileInputStream("config.properties"));
```

```
            // get some property values in order to be used for
authentication
            String host = prop.getProperty("host");
            String username = prop.getProperty("userName");
            String password = prop.getProperty("password");
            String protocol = prop.getProperty("protocol");
            String folderName = prop.getProperty("folder");
            int port = -1;

            // Get session
            Session session = Session.getInstance(prop, null);

            // Get the store
            Store store = session.getStore(protocol);
            store.connect(host, port, username, password);

            // Get folder
            Folder folder = store.getFolder(folderName);
            folder.open(Folder.READ_WRITE);

            /*
             * search in the folder for any 'seen' message because
all messages in
             * the 'sent items' folder are 'seen' by default
             */
            Message messages[] = folder.search(new FlagTerm(new
Flags(
                        Flags.Flag.SEEN), true));

            System.out.println("new message count is " +
messages.length);
            /*
             * iterate through and all the seen messages (messages
that are not
             * checked by the system yet)
             */
            for (int i = 0, n = messages.length; i < n; i++) {
                // get the current message to check and store it
in 'message'
                // temporary object
                message = messages[i];

     System.out.println("=====================================
= "+message.getSubject() + " SENT "+message.getSentDate() +"
======" );
                // get the content of the message
                Object content = messages[i].getContent();
```

```
                if (content instanceof Multipart) {// if the
message contains
                                                                //
multiple parts
                     handleMultipart((Multipart) content);
                } else {// if the message contains only a single
part
                     handlePart(messages[i]);
                }

                /*
                 * mark the read message as unseen This is a work
around to check
                 * only the new messages that the system has not
read yet
                 */
                messages[i].setFlag(Flag.SEEN, false);
           }

           // Close colder and store
           folder.close(false);
           store.close();
      }

      /**
       * iterates through all the parts and calls handlePart
method to handle each
       * part
       *
       * @param multipart
       *            message to be analyzed
       * @throws MessagingException
       * @throws IOException
       */
      private void handleMultipart(Multipart multipart)
                throws MessagingException, IOException {
           /*
            * for each single part call a method to handle that
part
            */
           for (int i = 0, n = multipart.getCount(); i < n; i++)
{
                handlePart(multipart.getBodyPart(i));
           }
      }

      /**
       * analyze single part by spell checking and attachment
file size
```

```
     *
     * @param part
     *            a single part to be analyzed
     * @throws MessagingException
     * @throws IOException
     */
    private void handlePart(Part part) throws
MessagingException, IOException {
        // used to identify the type of the part (i.e. body,
attachment,.. )
        String disposition = part.getDisposition();
        String contentType = part.getContentType();

        if (disposition == null) { // When just body
            System.out.println("Null: " + contentType);
            // Check if plain
            boolean isPlain =
contentType.toLowerCase().substring(0, 10)
                    .equals("text/plain");
            if ((contentType.length() >= 10) && isPlain) {
                // get the message body
                String messageBody =
part.getContent().toString();

                // scan the message body to get each word in
the message
                // Scanner scan = new Scanner(messageBody);
                StringTokenizer tok = new
StringTokenizer(messageBody,
                        "\t .!&%?@:;[]<>,{}0987654321");
                int wordCount = 0;// holds the total word
count
                int misspelledWordCount = 0;// holds the
misspelled word count
                String word = "";// holds a single word from
the message body

                /*
                 * iterate through each word in the message
body each word is
                 * spell-checked
                 */
                while (tok.hasMoreTokens()) {
                    // get a single word
                    word = tok.nextToken();
                    wordCount++;// increment the word count
                    if (!dictionary.isCorrectWord(word))
{// if misspelled word
```

```
          // was found
                              System.out.println(word + " -->
wrong");
                              misspllledWordCount++;
                         }

                    }
                    // calculate the misspelled word average
                    double misspelledWordPercentact =
(misspllledWordCount * 1.0 / wordCount) * 100;

                    // get the limit of the misspelled word
percentage from the
                    // config file
                    int limit = Integer.parseInt(prop

     .getProperty("misspelledLimitPer"));
                    /*
                     * if the percentage of misspelled words is
greater than 25% and
                     * the message body is not empty, notify the
admin
                     */
                    if (wordCount > 0 &&
misspelledWordPercentact > limit) {
                         DecimalFormat twoDForm = new
DecimalFormat("#.##");
                         misspelledWordPercentact =
Double.valueOf(twoDForm

     .format(misspelledWordPercentact));

                         // build the message
                         String message2Send = "Total number of
message words are: "
                                   + wordCount;
                         message2Send += "\nMisspelled word
count is: "
                                   + misspllledWordCount;
                         message2Send += "\nPercentage of
misspelled words are: "
                                   + misspelledWordPercentact +
"%";
                         // notify the admin
                         notifyAdmin(message2Send, "Suspicious
email sent");
                    }
```

```
                } else { // if other than plain text ignores
                     System.out.println("OTHER BODY: " +
contentType);

                }
           }

           else if (disposition.equalsIgnoreCase(Part.ATTACHMENT)
                     ||
disposition.equalsIgnoreCase(Part.INLINE)) {// if this part

                // is an

                // attachment or

                // inline

                // attachment
                System.out.println("Attachment: " +
part.getFileName() + " SIZE: "
                          + part.getSize());

                attachment = part;
                String sender = message.getFrom()[0].toString();

                /*
                 * get the period where the average to compare
with the current
                 * attachment size will be calculated
                 */
                int numberOfDays = Integer.parseInt(prop
                          .getProperty("numberOfDays"));
                int altNumberOfDays = Integer.parseInt(prop
                          .getProperty("alternativeDays"));

                // get the average from the database
                double avg =
ConnectionManager.getLastDaysAttachmentAvg(sender,
                          numberOfDays, altNumberOfDays);

                System.out.println("overavg AVG attachment size
is " + avg);
                if (part.getSize() > avg) {// if the size exceeds
the average size
                                                    // in the last
period

                     // log this record to the database
```

```
                    ConnectionManager.log(message, attachment,
true);

                    DecimalFormat twoDForm = new
DecimalFormat("#.##");
                    double currentAttachmentSize =
(part.getSize() / (1024.0 * 1024.0));
                    currentAttachmentSize =
Double.valueOf(twoDForm
                              .format(currentAttachmentSize));

                    double avgAttachmentSize = (avg / (1024.0 *
1024.0));
                    avgAttachmentSize = Double.valueOf(twoDForm
                              .format(avgAttachmentSize));
                    double overAllAVG =
(ConnectionManager.getOverAllAvg(sender) / (1024.0 * 1024.0));
                    overAllAVG =
Double.valueOf(twoDForm.format(overAllAVG));

                    // build a message
                    String temp = "Attachment name: " +
part.getFileName();
                    temp += "\nCurrent sent attachment size: "
                              + currentAttachmentSize + " MB";
                    temp += "\nUser Oversized-Average attachment
size: " + avgAttachmentSize
                              + " MB";
                    temp += "\nUser up-to-date attachment size:
" + overAllAVG
                              + " MB";

                    // notify the admin
                    notifyAdmin(temp, "Over-size Attachment
Sent");

               } else {
                    ConnectionManager.log(message, attachment,
false);
               }
          }
     }

     /**
      * sends a message to the administrator whenever a
suspicious behavior
      * happened
      *
      * @param MessageBody
```

```
     *            the content of the message
     * @param subject
     *            of the message
     */
    private void notifyAdmin(String MessageBody, String
subject) {
          // get the values from the property file
          String from_email = prop.getProperty("systemEmail");
          String password = prop.getProperty("sysPW");
          String from_name = prop.getProperty("sysAlias");
          String to_email = prop.getProperty("adminEmail");

          try {
               String sender = message.getFrom()[0].toString();

               // store all the recipients in the array
               Address[] addresses = message

    .getRecipients(Message.RecipientType.TO);
               String receivers = "";
               // prepare the recipient list to be written in
the email body
               for (int i = 0; i < addresses.length; i++) {
                    receivers += addresses[i] + ", ";
               }

               String messageSubject = message.getSubject();
               /*
                * start building the message by creating the
general information
                * first then append the detail of the suspicious
email
                */
               String body = "This to inform you that a
suspicious email "
                         + "has been sent titled '" +
messageSubject
                         + "' from the address '" + sender + "'"
                         + " to the address '" + receivers + "'
on "
                         + message.getSentDate().toString()
                         + "\nFollowing is more details about
the sent message: \n";

               body += MessageBody;

               // put more properties in the property object
               prop.put("mail.smtp.port", "587");
```

```
                prop.put("mail.smtp.socketFactory.fallback",
"false");
                prop.put("mail.smtp.quitwait", "false");
                prop.put("mail.smtp.host", "smtp.gmail.com");
                prop.put("mail.smtp.auth", "true");
                prop.put("mail.smtp.starttls.enable", "true");

                // get a session to send the email
                Session session =
Session.getDefaultInstance(prop);

                // create a message object
                Message msg = new MimeMessage(session);
                // set the subject of the method
                msg.setSubject(subject);

                // create email addresses to set the 'from' and
'to' data
                InternetAddress from = new
InternetAddress(from_email, from_name);
                InternetAddress to = new
InternetAddress(to_email);
                msg.addRecipient(Message.RecipientType.TO, to);
                msg.setFrom(from);

                /*
                 * create and build the message parts
                 */
                Multipart multipart = new
MimeMultipart("related");
                BodyPart bodyPart = new MimeBodyPart();
                // set the content to plain text
                bodyPart.setContent(body, "text/plain");
                // add the created part to the message body
                multipart.addBodyPart(bodyPart);
                msg.setContent(multipart);

                // start the connection to the server to send the
email
                Transport transport =
session.getTransport("smtp");
                transport.connect(from_email, password);
                // send the message
                transport.sendMessage(msg,
msg.getAllRecipients());
                // close the connection
                transport.close();
```

```
                System.out.println("[MailTool] mail sent
successfully ...");

            } catch (Exception e) {
                System.err.println("[MailTool] send() : " +
e.getMessage());
                e.printStackTrace();
            }
        }
}
```

E. Source Code – dbconfig.properties

```
#database vender (access or mysql)
vender=access
#------------ MS Access database ------------
#file name that is placed in the same directory of the .jar file
fileName=db1.accdb
#------------ MySQL -------------------------
#user name
userName=root
#password
password=admin
#database host
host=localhost
#database schema name
schema=test
```

F. Source Code – config.properties

```
#Sun Dec 26 02:53:39 EST 2010
userName=mailtestproj@gmail.com
password=XXXX
host=imap.gmail.com
protocol=imaps
folder=[Gmail]/Sent Mail
#used to specify the period to get the average of the attachment
numberOfDays=3
#alternatively used to specify the period
alternativeDays=7
#the system email from where notification will be sent to the
admin
systemEmail=mltrckngsstm@gmail.com
#password of the system email
```

```
sysPW=XXXX
#system email alias that will appear as a name to the admin
sysAlias=Invisible Witness System
#the admin email where the notifications will be sent
adminEmail=mltrckngsstm@gmail.com
#the accepted percent of misspelled words.
misspelledLimitPer=25
```

Works Cited

[1] Java 2 Platform Packages. (2010, 12 26). Retrieved 11 10, 2010, from JavaTM 2 Platform Standard Edition 5.0 API Specification: http://download.oracle.com/javase/1.5.0/docs/api/

[2] Mapping Data Types Between Relational Data Models and UML Class Diagrams. (2010, 03 12). Retrieved 06 06, 2010, from IBM: http://www-01.ibm.com/support/docview.wss?uid=swg21325957

[3] Gmail: Email from Google. (2011, 01 08). Retrieved 01 08, 2011, from http://mail.google.com/

[4] Ayers, D. (2009). A second generation computer forensic analysis system. *Digital Investigation 6*, S34-S87.

[5] Forsloff, C. (2010, 10 15). *Behavioral Profiling: Learn to Tell the Truth From Lies*. Retrieved 08 12, 2010, from ArticlesBase: http://www.articlesbase.com/psychology-articles/behavioral-profiling-learn-to-tell-the-truth-from-lies-569677.html

[6] Garfinkel, S., Farrell, P., Roussev, V., & Dinolt, G. (2009). Bringing science to digital forensics with standardized forensic corpora. *Digital Investigation*, S2-S11.

[7] Hadjidj, R., Debbabi, M., Lounis, H., Iqbal, F., Szporer, A., & Benredjem, D. (2009). Towards an integrated e-mail forensic analysis framework. *Digital Investigation*, 124-137.

[8] Harrington, J. L. (2009). *Relational Database Design.* Oxford: Elsevier.

[9] Hildreth, S. (2006, 07 10). E-MAIL MANAGEMENT:Controlling Content Chaos. *ComputerWorld*.

[10] Kaufmann, M. (1999). *Database design for smarties: Using UML for Data Modeling.* Chicago.

[11] Khurum Nazir Junejo, A. K. (2007). *Automatic Personalized Spam Filtering through Significant Word Modeling.* Pakistan: IEEE.

[12] Knoernschild, K. (2002). *Java Design: Object, UML, and Process.* Indinapolis: Pearson Education.

[13] Krebs, B. (2009, February 26). *Data Theft Common By Departing Employees*. Retrieved April 09, 2010, from The Washington Post: http://www.washingtonpost.com/wp-dyn/content/article/2009/02/26/AR2009022601821.html

[14] Marie-Francine Moens, J. D. (2010). *Identifying and Resolving Hidden Text Salting.* Belgium: IEEE.

[15] Melissa, W. (2007). A Draft of an Information Systems. *Journal of Information Systems*, 123-148.

[16] Moore, J. M. (2009). Your E-mail Trail: Where Ethics Meets Forensics. *Business and Society Review*, 274-293.

[17] Naksomboon, S., Charnsripinyo, C., & Wattanapongsakorn, N. (2008). *Considering Behavior of Sender in Spam Mail.*

[18] Olsson, J., & Boldt, M. (2009). Computer Forensic timeline visualization tool. *Digital Investigation*, 78-87.

[19] Peizhou He, X. W. (2009). *A simple Method for Filtering Image Spam.* Beijing, China: IEEE.

[20] Rogers, M. (2003). The role of criminal profiling in the computer forensics process. *CERIAS*, 292-298.

[21] Salvatore J. Stolfo, S. H.-W. (2005). Behavior Profiling of Email. *Email Mining Toolkit*.

[22] Vorobyov, G. D. (2005). Computer and Internet Use in the Work. *THE PSYCHOLOGIST-MANAGER JOURNAL*, 177-187.

[23] Wilfried N. Gansterer, A. G. (2007). *A reliable Component-Based Architecture for E-Mail Filtering.* Vienna: IEEE.

[24] Xiao Li, J. L. (2010). *E-Mail Filtering Based on Analysis of Structural Features and Text Classification.* Zhengzhou, China: IEEE.