# Investigating and Implementing an Email Forensic Readiness Architecture

by

**F.R. Van Staden**

Submitted in fulfilment of the requirements for the degree

**Magister Scientiae (Computer Science)**

in

**Faculty of Engineering, Built-Environment and Information Technology**

at the

**University of Pretoria**

August 2017

# Investigating and Implementing an Email Forensic Readiness Architecture

**by**

**F.R. Van Staden**

supervised by

**Prof H.S. Venter**

**Department of Computer Science**

## Abstract

Email forensic investigations rely on the collection and analysis of digital forensic evidence collected from email systems. Problems arise when the digital forensic evidence needed for the email forensic investigation is no longer available or there is a huge amount of email data that can be collected which take time to sift through to find the digital forensic evidence that is actually needed. The email digital forensic readiness (eDFR) architecture, as proposed in this dissertation, endeavours to address these problems. The eDFR architecture is based on the digital forensic readiness process described in ISO 27043.

To ensure that the collected email data can be used as digital forensic evidence a process to validate the collected email data was created. The validation process uses data collected from the email IP headers to validate the data in the SMTP headers ensuring that the SMTP header data was not spoofed or in any way changed. The IP header data is stored in an audit database together with the email data so that the validation process can be executed at any time. An audit database is used to store the collected data to ensure that once the data is stored it cannot be tampered with.

The digital forensic evidence collected using the eDFR architecture was found to be useable as part of an email forensic investigation. It was also found to be useful for other processes such as creating a graph representation of email sent and received by an email system or a group of email systems. It was shown that implementing the eDFR architecture could be achieved in an economical way that has almost no impact on current email systems.

**Table of Contents**

**List of Figures**

**List of Tables**

## Part I: Introduction

**Part I** consists of Chapter 1, the introduction of the dissertation. **Chapter 1** includes an overview of the **problem** that the dissertation **addresses**, the **environment** in which the problem **presents** itself and the **motivation** for addressing this specific problem.

# Chapter 1 : Introduction

## 1.1 Introduction

Electronic mail, normally called email, is probably the most common, efficient and cost effective electronic communication medium in today's interconnected world. Every person with access to an internet connection is expected to have an email account[1, 2]. Due to emails' popularity and ease of use it is prone to be abused. Unsolicited email communication, called spam, is one example of how email is abused. When email abuse is detected or reported, some form of mitigation action should be performed to either find the source of the abuse or to stop the abuse from happening again. A digital forensic investigation could be instigated as part of the incident response process.

Digital forensic investigations are dependent on the digital evidence that can be collected. The speed and effectiveness of a digital forensic investigation depends on the quality of digital evidence collected and the time it takes to collect the digital evidence. The time it takes to collect digital evidence is determined by the procedure that is used to collect this evidence. Email forensics in particular currently has an extensive procedure for collecting and analysing digital evidence from email.

The procedure is to collect email from the various email sources that are used to store email, such as the user's email account. The account resides on some form of post office or an email client is used to download email to a user's personal computer or mobile device. Once the email is collected it is deconstructed into its constituent components namely the header and the email body[3]. The investigator then analyse the email header and the email body to find digital forensic evidence that can be used as part of a digital forensic investigation. This process of analysing email data can be a long and cumbersome process, especially if there are a large number of emails to work through. Therefore, an architecture to facilitate the process of collection and analysis of email data to find digital forensic evidence is necessary.

The main idea for this research is to collect digital evidence from email systems by using digital forensic readiness techniques in order to affect consequences for perpetrators like spammers, phishers, cyber bullies and any other email abusers. During the process of collecting and analysing email data, it was found that potentially large amounts of digital evidence could be collected from email to support some types of email forensic

investigations. A determination was made that an automated process to collect email evidence was needed.

The remainder of this chapter is structured as follows: Section 1.2 discusses the terms that are frequently used throughout the dissertation. Section 1.3 discusses the problem and sub problems that are addressed by this research. The goal of the research is discussed in Section 1.4 and the scope of the research is discussed in Section 1.5. Section 1.6 gives an overview of the dissertation structure.

## 1.2 Terms

Digital forensics

- According to the Oxford Dictionary [4], digital forensic science is the systematic gathering of information about electronic devices that can be used in a court of law. Digital forensic science is more popularly called digital forensics and sometimes also called computer forensics.
- Palmer [5] defines digital forensics as " the use of scientifically derived proven methods towards the preservation, collection, validation, identification, analysis, interpretation, documentation and presentation of digital evidence derived from digital sources, for the purpose of facilitation or furthering the reconstruction of events".
- The Digital Forensic Process Moddel (DFPM) of Kohn [6] captures the definition of digital forensic science and states that any digital forensic process must have an outcome that is acceptable by law.

Digital forensic readiness

- Rowlingson [7] defines digital forensic readiness as consisting of two objectives. The first objective is to maximise the environment's capability of collecting digital forensic information and the second objective is to minimise the cost of a forensic investigation.
- Tan [8] also defines forensic readiness as consisting of two objectives, namely maximising digital data collection while minimising cost, but states that the usefullness of the digital data collected should be maximised and not the amount of data.

Email Forensics

- Email forensics is defined as the study of the source and content of an email to identify the author or sender, the recipient, the date and time and the origin of an email message [3].
- Email forensics can also be defined as the process of collecting and analysing email data such as archives and server logs, to establish a picture of email communication events [9].
- The author defines Email Forensic Readiness architecture as a set of digital forensic readiness processes that could be implemented together to achieve the outcome of collecting digital evidence from email systems.
- The author defines Digital Forensic Readiness process as a set of steps that are implemented to achieve a specific Digital Forensic Readiness outcome for example filtering the collected email evidence into spam and non-spam.

Email system

- According to Wood [10], Email systems today are mostly based on internet email systems that are based on a set of protocols, server software loaded on clustered email servers and accessed by email users using a mail user agent.

Performance monitoring tool

- According to Joyce [11], monitoring tools are used to support the debugging tests and performance of a software system.
- Performance monitoring tools are used to collect and store data on the performance of a software system by collecting data on the system hardware and software and converting the data collected into performance reports that are displayed to system administrators [12].

Spam, Spammer and Spoofing

- Lueg defines spam as unsolisited commercial email [13] and O'Brien defines spam as unsolisited bulk email [14].
- Anti-spam is defined as an application or device used by an email system administrator or email user, to reduce the amount of spam the user recieves [15].

- Email spoofing is the act of editing or falsifying the SMTP header information to hide the true origin or root of an email[16]. Spoofing is also defined as the act of adding fake validity to the content of an email by using a well-known and trusted domain as the originating domain in order to perpetrate a phishing attack [17].

The following section describes the problem statements that this dissertation addresses.

## 1.3 Problem statement

Huge amounts of data are generated by email systems every day. As soon as a digital forensic investigation is required, digital forensic evidence is manually collected from the email system, or the user machines or archive servers. To the researchers knowledge there exists no automated process currently, to assist digital forensic investigators to sift through the huge amounts of data. This issue is exacerbated by the manual process of collecting and analysis of digital forensic information. The net result is that investigators often miss the required digital evidence from email systems due to the excessive time the investigation might take. What is more, by the time that a digital forensic investigation is called for, the emails and/or logs might be missing due to issues such as limited storage capabilities, deletion of emails by users, volatility of network communications etc. In order to address these issues, organisation of email data needs to be proposed in the form of an email DFR process. Such organisation should enable one to automate the collection and analysis processes for the purpose of email forensics.

Another major problem with email systems is that the email headers, which are used for routing purposes, are not encrypted. Therefore, attackers can use a technique to change this data (like the origination address, destination address etc.). This technique is known as email spoofing. In order to determine if the email header data is authentic, a process is needed to validate SMTP header data.

The focus of this research is, therefore, to define an email forensic readiness architecture that could be used to collect digital evidence from email systems, organise the collected email data and validate the SMTP headers. The following questions need to be answered in order to dissect and address the problem.

**Q1:** Is there a way to automate the digital forensic data collection process for email?

The amount of data recorded to trace the origin of spam could potentially become very large. Currently, email forensic investigators collect digital evidence from email systems by accessing email accounts and extracting the digital evidence by copying the digital evidence. Collecting the digital evidence in this manual way could be a very time-consuming and tedious process. The problem is that, by the time the digital forensic data is collected, the spammer could have changed the originating spoofed email address yet again. Another possible problem could be that the digital evidence is lost if a user deletes the email containing the evidence from their mailbox.

**Q2:** What digital forensic data can be collected from email?

Sources of digital evidence in email systems, that contains the data needed to support email forensic investigations such as tracing the origin of email, should be identified. The digital evidence might already exist or might need to be added to email systems.

**Q3:** Can collected SMTP data be validated and verified for digital forensic purposes?

The SMTP header data collected must be validated to ensure that the SMTP header data was not tampered with and can be used as part of an email forensic investigation. It would be an advantage if the process of validation could be automated so that validation does not add to the time it takes to perform an email forensic investigation. The source of the SMTP header should also be recorded so that the collected data can be verified.

**Q4:** Can the collected digital forensic data be used for an email forensic investigation?

The data collected from the different sources identified will need to be critically analysed to ensure that the data can in fact be used as part of an email forensic investigation. Knowing what types of email forensic investigations can be performed with the collected digital forensic evidence, is crucial. The goals of the research are discussed in the following section.

A use case that is used to test the effectiveness of the proposed email forensic readiness architecture is to use the collected digital evidence to trace the origin of spam. There are several reasons why it is difficult to trace the origin of spam. Spammers replace the originating address of the spam message with another address to hide the origin of the spam message, called spoofing, which makes it difficult to trace the true origin of the spam message. Spoofed email, by its nature, have been tampered with, like the originating address,

therefore the evidence in a spoofed email could not be reliably used as digital forensic evidence, since the evidence cannot be verified. A digital forensic readiness process needs to be implemented to detect spoofed email as part of the data analysis process.

## 1.4 Goals

From the questions stated for the problem statement, the following goals were identified and are addressed in this research:

- Identify the data that needs to be collected in order to conduct email forensic investigations.
- Ensure that the data collected can be classified as digital forensic evidence by validating and verifying the collected data.
- Automate the data collection, data validation and data verification processes using digital forensic techniques.
- Develop a process that analyses the collected digital forensic evidence so that the evidence can be used as part of an email forensic investigation, for example, tracing the origin of email.

The following section discusses the scope of the research.

## 1.5 Scope

The main focus of the research is to create a digital forensic readiness process that collects digital forensic evidence from email systems that supports different forms of email forensic investigations for example, confirm the author of an email, to find the origin of an email, to determine the flow of information between suspects, called discussion threads, and/or to find proof of a cybercrime. Digital forensics is focussed on collecting and analysing digital evidence that could be used in a court of law. How the evidence is used in a court of law is not within the scope of this dissertation. Legal definitions are, however, used to define terms such as unsolicited electronic communication.

The research makes use of the term electronic communication but only focuses on email and does not expand into other forms of electronic communications like SMS, instant messaging or electronic voice communication. Unsolicited electronic communication is another general term that is used but the research focuses only on email spam and not on other types of spam for example instant messaging spam, newsgroup spam or mobile phone spam.

The research discusses some analysis processes performed on the data. These discussions only endeavour to show the data collected is useful for email forensic investigation. The analysis of the data in email forensic tools – once it is collected – is not within the scope of the research, since the main focus of the research is on collecting the data. The following section describes the layout of the dissertation.

## 1.6 Dissertation Structure

The layout of the dissertation is depicted in Figure 1.1. Chapters in the dissertation can be broken into five logical parts. **Part I** comprises Chapter 1, the introduction of the dissertation. **Chapter 1** includes an overview of the problem that the dissertation addresses, the environment in which the problem presents itself and the motivation for addressing this specific problem.

**Part II** of the dissertation contains the background chapters namely, Chapter 2 to Chapter 4. **Chapter 2** discusses the development of SMTP from its inception to the SMTP specification we use today and the devices that are used to implement an email system. Understanding what an email system consists of helps in understanding what data to collect from email systems and the particular places within the system the data can be collected from. **Chapter 3** discusses the history and the current **state of spam and anti-spam** technologies. Showing the co-evolutionary development life cycle of spam and anti-spam underpins the statement of the author that anti-spam is but a symptomatic cure to the spam problem. **Digital forensics, digital forensic readiness and email forensics** are discussed in **Chapter 4**. Understanding how to collect data and what type of data to collect is important during a digital forensic investigation. Once the process for collecting and analysing digital data is understood, a process for collecting the data could be automated.

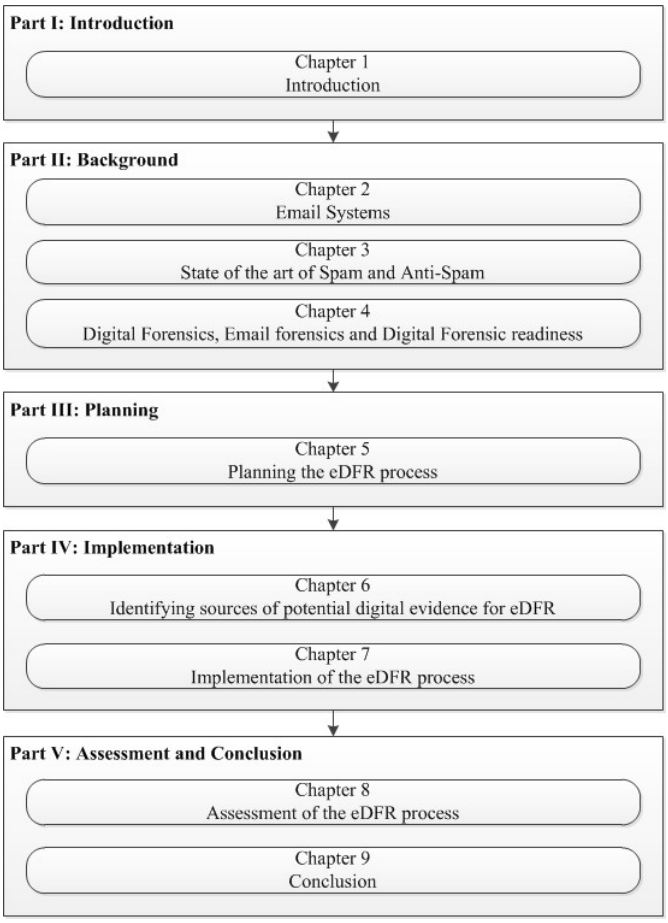


**Figure 1.1: Layout of the dissertation**

The remaining structure of the dissertation is based on the Digital Forensic Readiness design process that is described in ISO 27043 [18]. The Digital Forensic Readiness design process is broken into three sub-processes namely Planning, Implementation and Assessment which are represented by Part III, Part IV and Part V in the dissertation.

**Part III** of the dissertation contains **Chapter 5** which discusses the design of the **email Digital Forensic Readiness (eDFR)** architecture that is based on the readiness design process described in ISO 27043 [18]. The eDFR architecture was designed to capture digital forensic data from email systems. Digital forensic data is needed to trace the origin of spam. The rest of the research discussed in this dissertation supports the design of the eDFR architecture model.

**Part IV** of the dissertation contains Chapter 6 and Chapter 7. **Chapter 6** discusses the research that was conducted to **find digital evidence** in email systems. Defining the digital

forensic data that could be collected from email systems is part of the **implementation process** group of the ISO 27043 Readiness design process. **Chapter 7** gives and in depth discussion on the **implementation of the eDFR architecture** according to the planning done in Chapter 5 and the research conducted as discussed in Chapter 6. The implementation discussion shows how the data is **collected, handled, stored and analysed** using the eDFR architecture.

**Part V** of the dissertation contains Chapter 8 and Chapter 9. **Chapter 8** critically **assesses** the design of the **eDFR architecture** and proposes revisions that could be made to the eDFR architecture. **Chapter 9** gives the final **conclusion** of the dissertation and reviews the research that was conducted. The outcome of the research is discussed by analysing the research based on the problem statement that was made in Chapter 1.

The following is Part II with the first background chapter that discusses the components, protocols and processes that form part of a basic email system.

## Part II: Background

**Part II** of the dissertation contains the background chapters namely, Chapter 2 to Chapter 4. **Chapter 2** discusses the development of SMTP from its inception to the SMTP specification we use today and the devices that are used to implement an email system. Understanding what an email system consists of, helps in understanding what data to collect from email systems and the particular places within the system the data can be collected from. **Chapter 3** discusses the history and the current **state of spam and anti-spam** technologies. Showing the co-evolutionary development life cycle of spam and anti-spam underpins the statement of the author that anti-spam is but a symptomatic cure to the spam problem. **Digital forensics, digital forensic readiness and email forensics** are discussed in **Chapter 4**. Understanding how to collect data and what type of data to collect is important during a digital forensic investigation. Once the process for collecting and analysing digital data is understood, a process for collecting the data could be automated.

# Chapter 2 : Email systems

## 2.1 Introduction

Understanding email systems will help one to understand how certain fundamental aspects of email systems could be used to abuse email by the sending of spam. Understanding how email systems function will also help to identify where digital evidence can be collected from within these systems. The discussion of email systems also discusses components of email systems that should be implemented to safeguard the email system from abuse.

The first fundamental aspect of email systems that needs to be understood is the protocols that these systems are based on. Email systems make use of the Simple Message Transfer Protocol (SMTP) and the Internet Protocol (IP) to send and receive emails. Section 2.2 discusses the history and development of the SMTP that is implemented today, to show why SMTP is so easy to abuse. Section 2.3 gives an overview of IP, that email systems are also dependent on, to show possible sources of digital evidence. Security Architecture for IP is discussed in Section 2.4.

Email systems are created using a set of devices that implement SMTP and make use of IP to send and receive email. These email devices and the interaction between them, could possibly be used as sources of digital evidence. The devices used in an email system and how they work together are discussed in Section 2.5

Information security also has an impact on email since email is used to send and receive sensitive information. The information contained in an email should be protected in the same way as information in a database or a document. These protections influence the implementation of an email system. Section 2.6 gives a short overview of the impact that information security has on the implementation of email systems.

Another influence on the implementation of email systems are legal implications. Since an email is seen as a legal document there are also legislative aspects that influence the implementation and use of email systems. These legal aspects mainly focus on how email systems should be implemented to conform to legal standards. A short overview of the legislative impact is given in Section 2.7. The chapter summary is given in Section 2.8.

## 2.2 Simple Message Transfer Protocol

The history of the Simple Message Transfer Protocol (SMTP) is captured in a set of Request for Comments (RFC) publications. Request for comments publications are released by the Internet Engineering Task Force (IETF) and the Internet Society, to propose technical developments and standards with regard to the internet [19].

SMTP was first proposed in the RFC821 [20] in August of 1982. RFC821 describes a clear text protocol based on the "snail mail" architecture where electronic mail is sent from one "post office" to the next, until the mail is delivered to the intended "mailbox". SMTP is the protocol that is analogous to the "envelope" and not the content of the email. The basic rule of SMTP is: as soon as an SMTP host acknowledges delivery of the SMTP envelope, it is that SMTP host's responsibility to deliver the envelope to the correct email box. A mechanism to extend SMTP with extra services was proposed in February 1993 in RFC1425 [21]. The proposal included a way for calling SMTP clients to ask an SMTP server what services the SMTP server provides in addition to the services described in the SMTP standard. When a service extension becomes popular it is added to the SMTP specification. It was in this way that RFC821 was made obsolete in April 2001 by RFC2821 [22].The new specification included some service extensions and updates that were in popular use at the time. SMTP kept evolving to the latest RFC that describes SMTP, which is RFC5321 proposed October 2008 [23]. RFC5321 contains all the popular service extensions and updates of the services that were already defined for SMTP. SMTP services that were no longer used were removed. This process shows how the SMTP specification has evolved over time to stay current and usable. The process of proposing new SMTP service extensions is open so that anybody can propose new service extensions. A number of service extensions have been proposed to specifically combat spam, for example Purported Responsible Address (PRA) defined by RFC4407 [24], Sender Policy Framework (SPF) defined by RFC4408 [25] and Domain Keys Identified Mail (DKIM) defined in RFC 4871[26]. PRA, SPF and DKIM are discussed in more detail in Chapter 3.

The process of proposing new SMTP service extensions was the basis of the work conducted as discussed in Chapter 6. The idea described in Chapter 6 is to make use of a hash value to

safeguard information in the SMTP header, specifically the SMTP trace header. As stated before, information in the SMTP headers are stored in clear text. Therefore the information can easily be intercepted and edited (spoofed) by spammers. The possibility that the mail headers could have been edited makes the information in the headers suspect. Editing of the mail headers is done to hide information, like the origin of the email, from the receiving email box. The action of editing the mail headers to hide information is called spoofing, as stated in Section 1.2. The SMTP trace header is the most common point for spoofing since it is in human readable text and relatively easy to edit.

The Trace header, as defined in RFC5322 [27] ,consists of two sub headers, a return-path and received headers. The return path header is used to store the address where error reports, such as “specified email address does not exist” or “email was blocked by the receiver”, should be sent. The received header stores the delivery path with a date stamp for each delivery entry. The usage of the trace header is defined in RFC5321 [23] for delivering error reports to the sender, as well as to create delivery reports that can be used as input information when doing trouble shooting. Klensin [23] proposes that the trace header should be made compulsory for all SMTP servers that implement the RFC5321 standard so that error reporting can be better implemented for email services for better trouble shooting capabilities.

The augmentation, discussed later in Chapter 6, is defined for the received header, therefore only the received rule for the received header is given. The received rule and the received-token rule are shown in Figure 2.1.

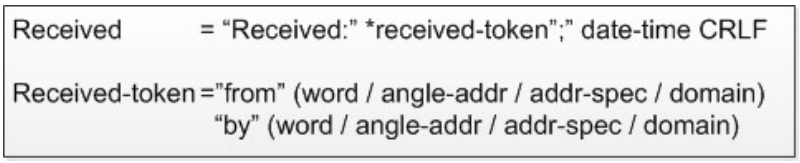


**Figure 2.1: Received rule and received-token rule**

The received rule in Figure 2.1 indicates that the received header must start with the word “Received” followed by a possible empty list of received-tokens. The list of received-tokens is followed by a date-time stamp and a Carriage Return Line Feed (CRLF) character that indicates the end of the received header entry.

The received-token rule indicates that the received-token must start with the word “from”, followed by the address of the sending host. The received-token ends with the word “by”,

followed by the receiving host's address. The following section gives a short overview of the Internet Protocol (IP) and the Security Architecture for the Internet Protocol (IPSec).

## 2.3 Internet Protocol

Chapter 6 discusses augmenting the Internet Protocol (IP) to add a hash value to validate the data in the IP headers. Chapter 7 discusses ways of collecting IP header data. Understanding how IP works and what it is used for is important background to the work discussed in the aforementioned chapters. Internet Protocol (IP) version 4 (IPv4) was first published in RFC791 [28]. This protocol described a way of routing network packets by assigning each network interface an IP address. IP also defines a way for the network packets to be fragmented when needed and for the reconstruction of the packets.

During the network transportation process the email is broken into packet size blocks that will fit into the data area of the TCP packet. These TCP packets containing the different data blocks are then transported using IP packets which encapsulate the TCP packets[29]. When the packets are delivered to the receiving network interface the packets are reconstructed and the TCP/IP packet data is sent to the application layer the data is intended for. The normal reconstruction process makes use of buffers in the network interface to store the TCP/IP packets while the reconstruction process is taking place but after the reconstruction is complete the TCP/IP packets are removed from the buffer.

RFC2460 [30], published in December 1998, described the next generation Internet Protocol, IP version 6. IPv6 was proposed to increase the address space of IP since the address space was getting too small for the number of network devices connecting to the internet. Gartner [31] expected about 4.9 Billion devices connected to the internet needing IP addresses in 2015. IPv4 has an address space of $2^{32}$ which is enough for 4.2 Billion devices. IPv6 has a theoretical address space of $2^{128}$ or enough for 340 undecillion devices. IPv4 is still widely in use, although some network devices now support IPv4 and IPv6 and the ability to convert between the two protocol versions. The specification for IPv4 and IPv6 describes an optional header, called the Routing header, which has an entry called the Record route option.

Every router that the datagram passes through updates the record route option for each datagram. The length of the Routing Header is set by the origin of the packet before the sending of the packet starts. When the maximum length is reached no more addresses are added to the Routing Header but an Internet Control Message Protocol message is sent to the

origin of the packet, stating that the Routing Header size is too small. The original sender of the packet then needs to increase the size of the routing header and send the packet again. This means that the packets should be delivered with the full route the packet took in the routing header. The following section discusses the Internet Protocol security extension that can be implemented to secure IP packets during transmission.

## 2.4 Security Architecture for Internet Protocol

The Security Architecture for the Internet Protocol (IPsec) provides interoperable, high quality, cryptographic-based security for IPv4 and IPv6 [32]. IPsec was published in August 1995 in RFC1825 and describes the security services offered and how these services can be employed in the IP environment. The security services offered by IPsec include access control, connectionless integrity, data origin authentication, detection and rejection of relays (A form of partial sequence integrity), confidentiality (via encryption) and limited traffic flow integrity.

Access control is defined as a method of protecting access to a resource by implementing a process for authenticating resource requestors before a resource can be accessed [33]. With regard to IPsec this access control is implemented for transportation security. Connectionless integrity is a service that detects modification of an individual IP datagram, without regard to the ordering of the datagram in a stream of traffic [34]. Data origin authentication is a service that verifies the identity of a purported source of data [34].

The specification discusses the use of traffic security protocols such as IP Authentication header (AH), as defined in RFC1826 [35] and the IP Encapsulating Security Payload (ESP), as described in RFC1827 [36]. The AH adds cryptographic authentication to the IP header which is normally inserted after the normal IP header but before the packet payload. ESP is a mechanism that provides integrity and confidentiality to IP datagrams. This means that IP datagrams can be secured from tampering. IPsec was upgraded in November 1998, with the publication of RFC2401 [30].

The latest specification for IPsec is RFC 4301 which contains enhanced specifications for the security services to be implemented for the IP layer. A correctly implemented IPsec deployment should introduce no adverse effects for users, hosts or other internet components that do not implement IPsec. The IPsec implementation specification is cryptographic algorithm independent, which means any one or more cryptographic algorithms can be used

to implement IPsec. Different cryptographic algorithms can therefore be selected for different implementation scenarios without affecting other implementations.

The following section discusses the devices used as part of an email system.

## 2.5 An overview of email system components

Email is one of the most widely used internet services and is based on the SMTP protocol, as discussed in Section 2.2, which is one of the oldest internet service protocols still in service today. Although many improvements have been made to the basic protocol, at its base, it is still the same SMTP that was first implemented as an internet service. Email systems still have the same basic architecture, with a few optional extras like anti-spam servers or bulk mailers, which could be added if the need exists. This section discusses the most common of these optional devices that are becoming less optional due to organizational needs and legislative pressures to secure email systems and the information they may contain. Section 2.5.1 discusses the basic building blocks of an email system by looking at the most common email system setup available today. The optional extras like the anti-spam server, the bulk mail server and the archive server are discussed in Section 2.5.2, 2.5.3 and 2.5.4 respectively. Although the term “optional” is used when discussing anti-spam, bulk email and email archive servers, best practise dictates that these servers form part of any corporate email system. The following section discusses the basic components of an email system.

### 2.5.1 Basic email systems

The initial SMTP standard, discussed in Section 2.2, describes email as being sent from one SMTP server to the next until it reaches its intended email recipient. Looking at the basic email system, it should be noted that email is no longer sent from one SMTP server to the next until the email is finally delivered. Instead, a Domain Name System (DNS) lookup is done for the receiving SMTP server’s IP address which is used to transport the email from the sending server to the receiving server. The research described in later chapters is based on the basic email system implementations discussed in this section.

Most email systems in use today are based on Internet mail architecture (IMA) described in RFC5598 [37]. The architecture is based on three main specifications namely SMTP, as described in Section 2.2, Internet Message Format (IMF) defined by RFC5322 [38] and Multipurpose Internet Mail Extensions (MIME).

IMF describes the format of internet text messages that are sent between different computer users. Where SMTP is the envelope of the message, IMF describes the text content of the message. IMF does not make any provision for content other than text. Any content other than text is defined by MIME.

MIME is defined by a number of standards since each internet content type has a MIME standard that describes it [39]. Content types described by MIME include rich content for example pictures, audio and video objects. The MIME for audio, for example, stipulates that the audio object should be described in the MIME headers so that the audio object can be correctly identified as audio and the type of audio object it is e.g. MP3, which MIME is described in RFC3003 [40]. MIME is not just used by email systems but also by HTTP to transfer content from the web server to the browser and other contexts.

The IMA is logically split into two distinct worlds namely the user world, in the form of Message User Agents (MUA) and the transfer world, in the form of Message Handling Service (MHS). The MHS accepts a message from one user (MUA) and delivers it to another user (MUA) creating a virtual MUA-to-MUA exchange environment[37]. The MUA and MHS are implemented as a number of programs also called email agents. A number of email agent classifications exist, such as Mail Submission Agent (MSA), Mail Access Agent (MAA), Mail Transfer Agent (MTA), Mail Delivery Agent (MDA) and Mail Retrieval Agent (MRA). Each of the email agents is responsible for its specific function in an email system, as indicated by the names. The email agents collectively make up the email system software.

Each email agent acts as an actor, with a specific role, within the email system. The different actor roles fall into three basic types namely User Actors, MHS Actors and Administrative Management Domain (ADMD) Actors. User Actors are responsible for the sending and receiving of messages. There are four user Actors namely Authors, Recipients, Return Handlers and Mediators. MHS Actors are responsible for creating a single MUA-to-MUA or end-to-end transfer from the Author Actor to the Recipient Actor. The MHS has four distinct Actors namely Originator, Relay, Gateway and Receiver.

ADMD Actors are implemented by an organization to enforce the operating policies and trust-based decision making of the organization. Each organization might also have different ADMD implementations for different types of traffic for which different operational policies must be applied. An example would be that a separate ADMD would be implemented for

internal, external and bulk email traffic. Figure 2.2 shows the way that all of these components work together to transfer a message from the Author/Sender of the email to the Recipient.

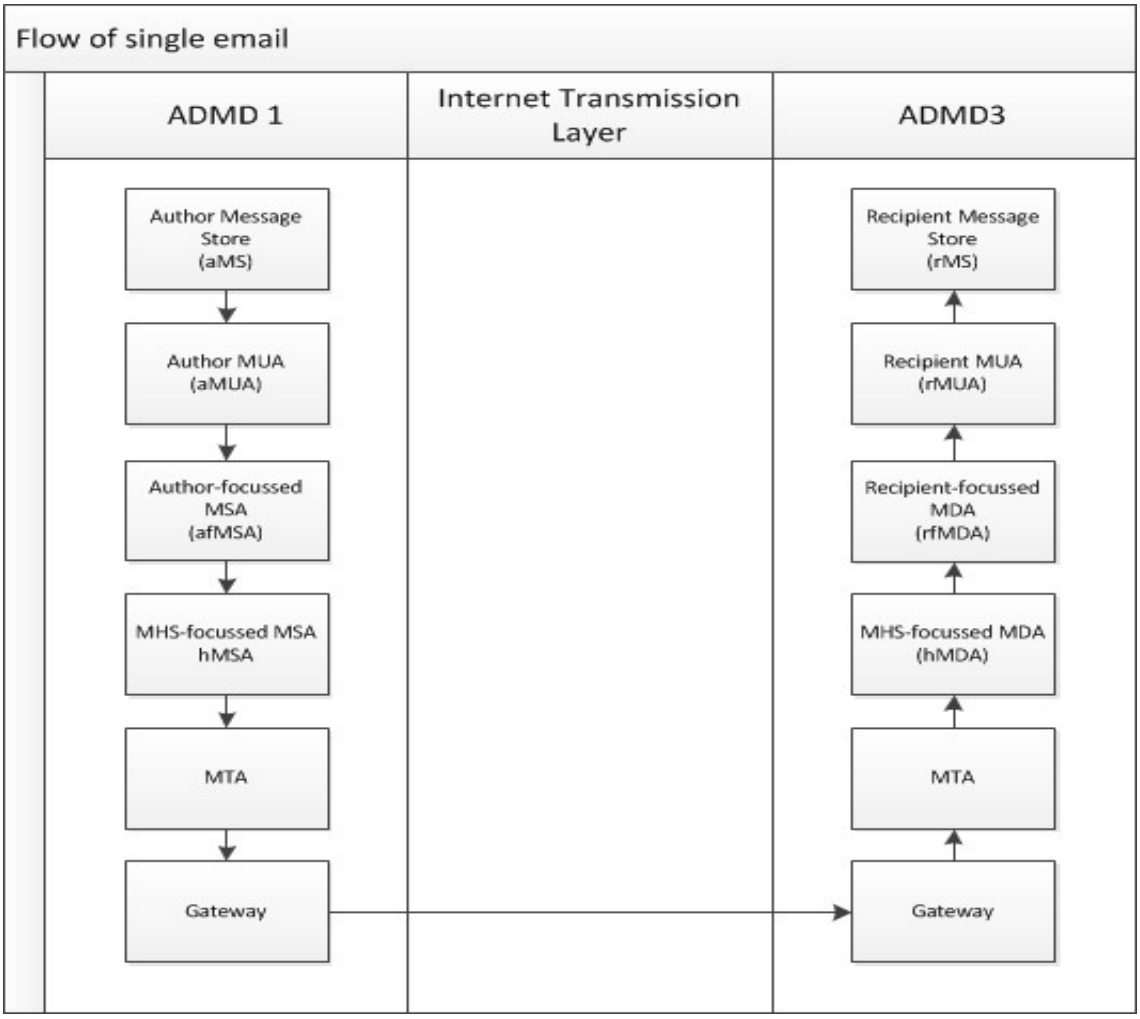


**Figure 2.2: The flow of an email through the different email components**

ADMD1 in Figure 2.2 is an ADMD implementation for the external email of an organization. ADMD1 enforces the policies of the organization on the transfer of the email message to another organization, through the internet. The message originates from the Author Message Store (aMS) when the email user sends the message. The message is then transferred from the aMS to Author-focussed MSA by the Author MUA. The Author MUA performs the function of the originator. The Author-focussed MSA hands off the message to the MHS-focussed MSA so that the message can be transferred to the recipient. The MTA handles the actual transfer of the message from the internal email system to the email gateway.

The email gateway is a hybrid between a user and a relay in that, like a relay, it performs MHS-level transfer-service and routing. The gateway can also operate as a user if it needs to change the message content. Message content is changed to force a message to adhere to the operation policies of another ADMD. The changes that the gateway makes is to allow the message to be delivered, even if the two ADMDs are not implemented with the same operational policies[37]. Email gateways are normally implemented on an SMTP gateway. The SMTP gateway is defined as an intermediate host that acts as either an SMTP relay or as an email gateway into some other transmission environment[22]. The SMTP gateway is used as the main routing point for all incoming and outgoing email. The SMTP gateway maintains a list of known organizational mailboxes.

A mailbox is an identity used by IMS to uniquely identify a mail sender or receiver. A mailbox is specified as an Internet Mail Address with two distinct parts separated by a '@' sign. The right side of the '@' sign is the domain name associated to the ADMD. The left side of the '@' sign is interpreted only by the entity specified by the ADMD policy. Other unique identifiers used by the IMS are; the domain name, the message-ID and the envelope identifier (ENVID). The domain name is a global reference to an internet resource such as a host, a service or a network [37, 41]. Two standardised tags exist to identify a message namely message-ID and ENVID. Message-ID pertains to the content of the message and the ENVID pertains to the transfer of the message. The message-ID is created by a MUA and the ENVID is created by MHS.

As can be seen from the description of the IMA the focus is still to deliver email to the intended recipient but the process of delivery has been implemented to be an internet service. Making use of IP to move data from a sender to a receiver is the base communication protocol for all internet services. Becoming an internet service means that SMTP servers no longer have to be concerned with how an email is transported and can focus on managing the SMTP envelope. Unfortunately, part of SMTP communication involves spam. An added function of SMTP becoming an IP service is that SMTP communication can be encrypted using an SSL which is called secure email [42].

### 2.5.2 Anti-Spam servers

Section 1.2 gives a definition for an anti-spam server. An anti-spam server is an SMTP gateway, as described in section 2.5.1, with anti-spam service implemented. The anti-spam

server scans incoming email before the email is routed by the SMTP gateway. Outgoing email should also be scanned by the anti-spam server before it is routed by the SMTP gateway. The SMTP gateway only accepts outgoing email that originates from the outgoing anti-spam server. Placing the SMTP gateway between the incoming anti-spam server and outgoing anti-spam server protects the SMTP gateway from being overloaded by spam.

Anti-spam software has features that can be implemented to slow down the rate of incoming and outgoing email to ensure that the anti-spam server does not get overloaded. The anti-spam stores incoming email before it is scanned. Once the email is scanned it is sent to the SMTP gateway and removed from the anti-spam server storage. Scanning of incoming email is a CPU intensive process, therefore anti-spam servers should be provisioned with enough processing power to handle the amount of incoming email. The scanning criteria for the anti-spam service should be set according to the organizational standard operating procedures.

In the opinion of the author, the organizational firewall should have a rule on the SMTP port that forces incoming SMTP traffic to first go through the incoming anti-spam server for filtering [43]. Another firewall rule should exist that blocks any outgoing SMTP traffic that does not originate from the SMTP gateway. This will ensure that, if a botnet exists inside the organization, the email that is generated from the botnet is blocked before leaving the intranet.

### 2.5.3 Archiving server

Just as paper based documents, containing organizational information, should be archived to adhere to legislative requirements, email must also be archived [43-45]. Same as with other organization documentation, the content of emails should be classified and archived according to the classification. Most organizations choose to implement digital archives, since digital archives take up less space and can be better protected. Paper based documentation is digitised and archived with other digital documentation.

It is easy to change data in a digital format, therefore extra measures must be put in place to ensure that the email archives are not tampered with and that the integrity of the email can be proven. It is suggested that an organization investigate the procurement of an email archive system that is certified to collect and store the email in a tamper proof and verifiable archive.

### 2.5.4 DNS Server

The Domain Name System (DNS) has been mentioned a few times already and is one of the most important components in the internet service architecture. DNS's importance lies in the fact that it is the mechanism that is used to translate or resolve a human understandable resource name like www.up.ac.za to an IP address, since the IP protocol accepts only IP addresses – not human-readable addresses [46].

DNS is more than 20 years old and based on a number of RFCs [47]. Three major components make up the DNS namely the Domain name space, the name servers and the resolver programs [48]. The domain name space contains the resource records that are used to specify name space that is used to store the domain names and the data associated with the domains, in a tree structure. The name servers hold the information about the domain tree structure and how the human-readable addresses are linked with the IP addresses. The resolver programs are accessed by user systems to resolve a domain address to an IP address, using the said name servers.

DNS is based on domain name notation which is an alphanumeric string that describes the domain name [46], where the different levels of the domain are defined using alphanumeric characters separated by dots. An example would be up.ac.za where "za" gives the root of the domain tree; "ac" gives the second level of the domain tree and "up" gives the final or owner level of the domain tree. This domain name tells a resolver to first locate the "za" domain on the name servers, then locate the "ac" sub domain and finally locate the "up" sub domain. The root address for www.up.ac.za will be associated with the "up" leaf domain entry, with an added leaf for "www". A request for www.up.ac.za will return the address of the www.up.ac.za IP address. There will also be leaf entries for other resources, like SMTP server names, which will be denoted with an SMTP leaf entry which will contain information about the SMTP gateway.

DNS was designed in such a way that extra data and queries could be added to the tree structure [49] without needing changes to the original DNS structure. An example of this design would be the inclusion of a Sender Policy Framework (SPF) record or a Domain Key Identified Mail (DKIM) signature record [50, 51]. SPF and DKIM are discussed in more detail in the following chapter. The SPF and DKIM records are stored in the DNS domain name tree as extra data associated with the email servers and can be accessed with specific SPF and DKIM queries. Implementing SPF or DKIM signatures ensures that an organization's domain is protected from being used to spoof email. The following section

gives a short overview of the implications that information security place on email systems, due to the fact that email might contain confidential organizational information that should be secured to protect the organization from harm.

## 2.6 Information security and email systems

Information security focusses on the protection of organizational information to minimize the risk that organizational information could be stolen [45]. According to ISO/IEC 27002:2005 information that is sent and recieved using email, should be appropriatly protected. The risk involved with email must be understood and managed using appropiate protection mechanisms. An email policy that discusses the use of email and the information that can be sent and recieved using email systems, should exist for the organization. Information categorization should be used to limit what information can be sent with email and to whom the information can be sent. An example would be the sending of personal information to clients in such a way that the information cannot be stolen during delivery.

The organization can also implement secure email to further safegaurd classified information. The usage of secure email includes the implementation of encryption for the email being sent, that only the recipient of the email can decrypt. Implementing secure email will add to the security of the information in the email. The following section gives an overview of the legislative responsibilities when implementing an email system, since email is accepted as legal documents [44].

## 2.7 Legislation and governance with regard to electronic communication

Depending on the jurisdiction the email used, email can be used as evidence in a court of law since it is seen by the law as legal documents [44]. The main implication of legislation on email systems is that certain policies and procedures must be put in place to ensure compliance, which can help in finding and collecting digital evidence from email systems. According to Thornton[44], ever increasing legal responsibilities are being placed on email systems with regard to retention, destruction and restoration of emails. Email can be submitted as a legal document, therefore effective managnment of email systems must be implemented to ensure that email that is sent and received by an organization, is kept for the same period of time that physical documents are kept.

The ECT act [52] contains rules on how electronic communications should be managed in order to maintain and prove the integrity of electronic communication. This includes the use of being able to provide email documents when requested and the implementation of information security policies spesific to email. If an organization cannot supply the requested email documents or information is stolen due to inadequate protection, the organization will be liable for damages and penalties. The damages and penalties are a legal matter which is not the focus of this research. The following section gives the conclusion for the chapter.

## 2.8 Conclusion

This chapter discussed the different components that are supposed to make up an email system. Each component performs a specific function and has a specific interaction with other components in the email system. It is important to understand the function of the different protocols and components of an email system so that the collection points of digital forensic data in the email system can be defined. An overview was given about the standards and legislation that govern email systems. Understanding the interaction between the different email system components and the standards and legislation that should be adhered to is important for performing an email forensic investigation and also for implementing a digital forensic readiness process for an email system.

From the investigation into email system architecture it was concluded that email systems can logically be broken into three parts. The first section is the send receive section, which include devices such as the underlying network, firewall and anti-spam servers. The function of the first section is to ensure that the email is transported from the sender to the correct receiver. This section is controlled only by device configurations and system policies. This section would possibly be the best place to implement a DFR process since the possibility that the SMTP header could have been changed is less likely than for the other sections.

The second section is the scan and deliver section which include the anti-spam server, SMTP gateway and email Post-Office. This part is responsible for the scanning and delivery of email. The anti-spam server scans the incoming email for spam and handles the scanned email according to the filtering and policy that it was configured with. Once the email is scanned it is sent to the SMTP gateway to be delivered to the email Post-Office. Since filtering and resending of the email is done in this section it can be concluded that some of

the SMTP header data can be lost in this section. It would be better to implement a DFR process before the filtering and resending of the email is done.

The third and final section is the user controlled section which includes the email Post-Office, user mail box and email client. The activity of the third section is mostly controlled by user actions that are controlled by the organizational policies. Users can read, send, archive and delete email only as policy allows. Implementing a DFR process in the third section of the email system might not provide the required results since the email has already travelled through a number of email devices which could have altered the original SMTP header data and also filter out some of the email evidence.

The different spam and anti-spam scenarios that have been detected and developed are discussed in the following chapter.

# Chapter 3 : State of the art of Spam and Anti-Spam

## 3.1 Introduction

Spam has an economic implication for organizations. The implementation of anti-spam strategies has cost implications in terms of system resources needed to perform anti-spam functions and administration time to implement and maintain anti-spam systems. Since most anti-spam strategies are implemented either on the user's email box or on the organization's email servers, the spam needs to be downloaded from an Internet Service Provider (ISP) before it can be scanned and classified as spam [53]. Such downloading of potential spam has a direct impact on the bandwidth cost for an organization.

The effectiveness of anti-spam systems does not stop the spam from being sent. According to "ELEVEN Integrated Message Security", spam made up 73.9% of the total email load that was sent through their client's email systems in November of 2012 [54]. According to Symantec, 70.6% of email received by the email systems of their clients were considered spam[55]. Spam and Anti-spam have been caught in a type of co-evolution development life cycle, meaning that every time anti-spam systems find a way to block spam the spammers find a new way to bypass anti-spam systems.

The volume of spam received could also overload the email system of the organization, causing denial-of-service (DoS) of the email system. Organizations use email to send information electronically to vendors and customers. While email communication is hampered in this fashion, organizations cannot send information, which has a direct impact on productivity and can be seen as an economic loss for the organization. All this cost is carried by the organization that implements the email service.

Anti-spam systems could potentially become an inconvenience when some of an organization's email traffic is falsely identified and/or blocked as spam. Anti-spam systems are never completely accurate[53]. Anti-spam systems may still produce false positives, i.e. blocking legitimate email, or false negatives, i.e. allowing spam to be delivered. Blocking legitimate email could be detrimental to the operation of an organization. An example is, when an organization needs to send clients very important information but the email containing the legitimate information is wrongfully blocked and clients never receive the information. Anti-spam systems do not solve the spam problem; anti-spam systems are

merely a way to manage the spam problem and should still form part of an organizational email system. To truly solve the spam problem, spammers must be deterred from sending spam in the first place. Spammers use a technique called spoofing to hide from being detected, which makes it difficult to trace spam back to the spammer. The ability to successfully trace the origin of spam will discourage spammers from sending spam but only if there are consequences for spammers.

The state of the art anti-spam strategy makes use of intelligent filtering or content scanning. Intelligent filtering is build on all the anti-spam strategies developed earlier, including strategies like content scanning, black-listing and white-listing. Section 3.2 discusses inteligent filters in more detail. With each strategy developed there are still situations where intelligent filters give false positives and false negatives. False positives occur when legitimate email is marked as spam, for example medical correspondence including black-listed words. False negatives are when spam is not detected by the spam filter because of picture content or misspelling of black-listed words. Section 3.3 looks at the process of content scanning where the content of an email is analised according to set rules and algorihms, to determine if the content can be labeled as spam. Section 3.4 looks at black-listing, white listing and mailbox authentication. These are all different types of intelligent filtering that anti-spam services employ to detect and block spam.

Another state of the art anti-spam strategy is to ensure that spammers can make use of spoofing to hide the origin of the mail when sending spam. Email spoofing is the act of editing or falsifying the SMTP header information to hide the true origin or root of an email[16]. Spoofing is also used to add fake validity to the content of an email by using a well-known and trusted domain as the originating domain in order to perpetrate a phishing attack. The Sender Policy Framework (SPF) and Domain Key Identified Mail (DKIM) are SMTP extensions that were designed to combat spoofing in email [17]. Section 3.5 discusses SPF and DKIM in detail.

Honeypots are devices that are set as traps to entise attackers to attack the device [56]. The attacker may try and make the honeypot device into a zombie and once that is done the botnet can be infiltrated from the honeypot device. Section 3.6 looks at honeypots. Botnets are used to send spam using a distributed sending network so that a single point cannot be detected and black listed. Botnets consist of infected PCs, the invected PCs are called zombies, that work together to aid in cyber crimes[57, 58]. The botmaster controls these botnets from a

central point. The problem of eliminating botnets and botmasters is firstly to find zombies in the botnet and then to trace them to the botmaster. It is possible to trace botnet activity using digital forensics and trace the activity back to the botmaster. Section 3.7 looks at botnets and their use. The chapter ends in Section 3.8 with the conclusions reached in this chapter.

## 3.2 Intelligent Filters

Intelligent filters are software applications that are installed as part of a user's mail application or as part of anti-spam service or both[53],[59]. Intelligent filters use a set of anti-spam strategies to improve the success of the filter such as content scanning and listing which is discussed later, in Section 3.3 and Section 3.4. Intelligent filters can be trained to reduce the amount of false negatives and false positives. The idea is that the user or groups of users give input to the filter to train it. Anti-spam software vendors claim that intelligent filters can be trained to block 99.9% of all spam[53],[59]. Although this is really good it is still not perfect. Intelligent filters still need to be trained correctly to achieve this block percentage and will have to be retrained every time a new form of spam content is used. From the claims made, it can still be deduced that not all spam can be blocked at all times.

## 3.3 Content Scanning

Content scanning is sometimes implemented as a separate anti-spam application on the user's email client or as part of an anti-spam server's intelligent filtering process [53, 59]. To give email content a spam probability rating, content scanners use statistical filtering methods such as Bayes or Chi-squared[14]. The algorithm uses key words and key phrases to calculate the spam probability rating. We also refer to these key words and phrases as patterns. This rating categorises the email as spam, possible spam or non-spam. Spam is stopped at the email server. Possible spam could be marked with a spam tag but are still sent to the user. Non-spam is seen as normal email.

To thwart content scanners, spammers use techniques like picture content, HTML tag inlay and misspelling of patterns [53, 60]. Picture content is a series of pixel values and cannot be scanned the same way as text, because it is in the format of an image such as a bitmap or jpeg, or any other picture format. This means that the anti-spam server cannot read the content as text words. When content scanners try to scan text, the scanner ignores the pictures. Email servers can be set so that the ISP will only download picture content if the user gives permission for the action.

With the advent of mail clients being able to parse HTML, spammers started using tag inlay like “Viagra”, to hide patterns. The addition of the HTML tag  does not add or remove content when viewed in an HTML capable email client but confuses the content scanner. The content scanner is confused because, from the content scanner’s perspective, the word does not exist in the spam content list. Misspelling patterns, replacing characters but still making it recognisable to the reader e.g. “Vi@gra” or “V1@gra” for Viagra, causes the same type of problem for content scanners. Users need to add these alternative patterns to the lists of the content scanner each time an alternative is detected, since although there is a finite number of patterns, it will take time to preconfigure all the possibilities, for the content scanner to be able to detect the problem the next time. The following section discusses the different listing techniques and mailbox authentication which is also used as part of intelligent filtering.

### 3.4 Block list, black list, white list and mailbox authentication

A block list is built by individual users[53]. Users block email senders or email domains from the users’ own mail application. Block lists block email from being downloaded to a user’s mailbox in future. White listing is when users set up a list of allowed email accounts and email domains to be downloaded to their mailboxes.

A blacklist is generated at ISP level where email traffic is monitored for indications of bulk mail originating from a single source[53],[13]. The ISP will blacklist email domains suspected of sending bulk mail. There are also internet organizations that set up and maintain blacklists such as the Spamhaus project [61].

Faynberg, [62] proposed a method for authenticating email called mailbox authentication. Mailbox authentication relies on the fact that spammers falsify or spoof the “From” and “Reply-to” tags of an email. A message is sent to the mailbox in the “From” or “Reply-to” tag. If either the “From” or “Reply-to” mailbox does not exist, the mailbox cannot be authenticated and the email is marked as spam. Each gateway and relay server authenticates email by sending a query to the originating mail server, asking if the email received originated from the specified mailbox. Each email forwarded needs to be logged before the email is forwarded. This log checks if the respective server sent the mail, when there is a query about the sent mail. Support for this process was removed when SMTP servers started sending email from a sending server to a receiving server using IP routing.

An addition to the proposal is to make use of an Authentication, Authorisation and Accounting (AAA) server that is trusted to verify an email server. This server certifies that an email server being queried can be trusted to give a true answer. If the AAA server returns with a negative response, the server drops the email regardless of the sending server's response. When a server drops mail, the server's log notes that the drop action has been performed on the mail.

To get past email block lists and mail box authentication, spammers replace the "From" tag with the "To" tag in the email header[53],[59]. According to the email profiler, the mail appears to originate from the users' own mailbox. The spam email bypasses the block list since it is assumed that the user would not block its own mail address or domain. As long as the user's domain is not blacklisted, the spam email bypasses the black list. The spam email bypasses the white list because the white list automatically adds the user's address when it creates the list. Since the user's mailbox does exist, the spam email's "From" mailbox is authenticated. The following section discusses extensions to SMTP focussed on combatting spoofing.

### 3.5 Anti-spoofing SMTP service extensions

As stated before, email spoofing is the act of editing or falsifying the SMTP header information to hide the true origin or root of an email. Spoofing is also used to add fake validity to the content of an email by using a well-known and trusted domain as the originating domain in order to perpetrate a phishing attack. Anti-spoofing is therefore a process of detecting or blocking spoofing attempts. RFC4406 is an experimental RFC that describes two tests for SMTP servers to perform, to verify that a mail header has not been spoofed.

The first test is the Purported Responsible Address (PRA) test[24]. Lyon [24] describes a way to try and find the PRA inside the SMTP headers. If no PRA can be found, the email has a high probability of being spoofed. If the PRA can be established, it is still not proof that the SMTP header has not been spoofed, since the address used for the PRA is the first well-formed address the PRA algorithm found. The PRA needs to be tested further, to establish its validity.

The second test uses a Sender Policy Framework (SPF) to authenticate if an SMTP client is allowed to act on behalf of the originating domain[25]. Wong proposes SPF as a method to

detect a spoofed email that uses valid domain information to appear legitimate. The supposed sender domain and the routing information in the header is authenticated by the DNS of the domain owner, to determine if the SMTP client's domain has the authority to act on behalf of the supposed sending domain. If the DNS returns a failed authentication, the email is marked as possibly spoofed.

RFC4871[51] proposes Domain Key Identified Mail (DKIM) Signatures. DKIM defines a domain-level process that domain owners can use to verify that a message that was supposedly sent by the domain owner was actually sent by the domain owner. The verification process uses public-key cryptography and key sever technology to authenticate the message sender.

### 3.6 Honey-pots

Even[56] states "honey-pot systems are decoy servers or systems set up to gather information regarding an attacker or intruder into your system". Spitzner, according to Obied[63], defines a honey-pot as information system resources whose value lies in unauthorised or elicit use of that resource. The author defines a honey-pot as a trap set to detect, deflect, or in some other manner counteract attempts at unauthorised use of information systems. The information gathered by the honey-pot is used to track where the authorised access originated and what exploits were used. If the honey-pot is not accessed, it is of no use. A honey-pot logs access information in accordance with digital forensic information gathering techniques. This means that the content of the log files can be verified and validated. The design, implementation, placement and monitoring of a honey-pot is crucial to the effectiveness of the honey-pot.

Honey-pots have been deployed as open proxy servers and open mail relays, to gather information about the spammers that use them[63]. Honey-pots have also been employed to gather information on botnets[63]. The next section discusses the history of botnets, as well as advances made in the development of new strategies to trace and combat botnets.

### 3.7 Botnets

Botnets are created by infecting computers with Trojans. Once a computer is infected the Trojan could create an SMTP account on the local machine or use the email account that is configured on the computer. This account can then used to send spam and any other electronic content. The Trojans in a botnet used IRC (internet Relay Chat) connections to

receive information from the controler. According to InternetNews [64] the new tactic is for the bots to communicate with each other using Peer-to-peer connections that are set up in a family tree fashion to relay information and commands. With the IRC method the address of the controller is hard coded into the Trojan and can be extracted during dissection. The family tree method of control is when no "child" Trojan knows any of its forefathers other than its direct "parent". The family tree method makes it harder to find the controller.

Since closing McColo [64], a US-based ISP accused of being a major hub for spammer activity, spammers have learned to hide their activity behind the same technology used for secure networking. The biggest botnet closed in late 2008 and was called Sirbizi. According to Walters [65] the infection rates are increasing. InternetNews [64] states that MessageLabs is currently monitoring a number of botnets including Xarvester, Cutwail and Mega-D. Spammers use botnets to create low-volume-high-node-count mail senders. Low-volume-high-node count means that the nodes are only used to send a small subset of the mails to stay under the radar of bulk mail detectors. Mega-D was detected because it overutilised its bots.

According to Obied [63], Microsoft used a zombie machine as a honeypot to detect and trace spam activity. The machine was infected with a botnet's trojan and quarentined. Activity to and from the zombie was monitored. The information gathered by the zombie honeypot helped to track the command and control source of the botnet. This tracking information was used in a lawsuit against 13 spam operations. Using a zombie as a honeypot is only possible if the controller of the botnet is unaware of it that one of the zombies is being used as a honeypot. Botnets employing P2P connections between the different zombies make it harder to use on the zombies, to track the controler. The following section is a short conclusion of the discussions in this chapter.

## 3.8 Conclusion

The state of the art of spam and anti-spam given in Chapter 3 looked at the processes that are employed by spammers and anti-spam technology used to try and block spam. The background showed the co-evolution development life cycle that spam and anti-spam are caught in. Although anti-spam technologies are successful to a large degree, large amounts of spam are still being sent and some of the spam still make it to the users' mailboxes, which implies that anti-spam technologies are not deterring spammers from sending spam. It is for this reason that the research for this dissertation was conducted.

Looking at the focus of anti-spam strategies it can be seen that anti-spam techniques mainly focus on blocking the delivery of spam and not blocking the sending (origin) of spam. Even the anti-spoofing strategies such as SPF and DKIM do not block spam from being sent. It is the opinion of the author, however, that the only truly effective way of combatting spam is by blocking the spam from being sent in the first place and that can only be achieved if the source of the spam can be found and stopped. Tracing the origin of spam is not an easy task due to the techniques that spammers use, like spoofing. Once an email header is spoofed the data in the SMTP headers cannot be used to trace to origin of the email.

Digital forensic techniques to collect digital evidence from email systems could help trace the origin of spam and help get those sources blocked for good. The following chapter gives background about Digital forensics, Email forensics and digital forensic readiness.

# Chapter 4 : Digital Forensics, Email forensics and Digital Forensic Readiness

## 4.1 Introduction

This chapter describes digital forensics and digital forensic readiness to convey the implications of using digital forensic techniques to collect data. Understanding how to collect data and what type of data to collect is important during a digital forensic investigation. Once the process for collecting and analysing digital data is understood, a process for collecting the data automatically could be implemented.

This chapter gives an overview of concepts that are used in the rest of the chapters in this dissertation. The chapter gives a short background of digital forensics in Section 4.2. Email forensics is discussed in Section 4.3 and Digital forensic readiness is discussed in Section 4.4. There is also an added topic that forms part of digital forensic readiness namely Big-Data which is discussed in Section 4.5. The conclusion of the chapter is given in Section 4.6.

## 4.2 Digital Forensics

Digital forensic science is a relatively new field of study that evolved from computer security and became part of forensic science. According to the Oxford Dictionary[4], digital forensic science is the systematic gathering of information about electronic devices, that can be used in a court of law. Digital forensic science is more popularly called digital forensics and sometimes also called computer forensics. Palmer [66] defines digital forensics as " the use of scientifically derived proven methods towards the preservation, collection, validation, identification, analysis, interpretation, documentation and presentation of digital evidence derived from digital sources for the purpose of facilitation or furthering the reconstruction of events". Palmer's definition describes the digital forensic process whereas Oxford describes digital forensic science. Kohn [6] states that "any digital forensic process must have an outcome that is acceptable by law".

Digital evidance is defined as information or data, that is stored or transmitted in binary form, that may be relied on as evidence [18]. There is also potential digital evidence that is defined as digital evidence that has not yet been determined, through the process of examination and analysis, to be relevant to an investigation [18]. For example, when a hard drive is seized, not

all of the data on the drive will be used in a court case, but only pieces of it. Therefore, the entire hard drive is seen as potential digital evidence until, through the process of examination and analysis, the particular pieces of data, e.g. digital photos, will become the actual digital evidence. When, in the dissertation, the researcher uses the concept of digital evidence or digital forensic data it actually means potential digital evidence until the digital evidence becomes part of legal proceedings. Digital forensics is not employed to stop spam directly but used to gather information that can be used to indirectly assist the blocking of spam. The following section defines email forensics.

### 4.3 Email forensics

Conducting Digital forensics on email is called email forensics. Email forensics is defined as the study of the source and content of an email to identify the author or sender, the recipient, the date and time and the origin of an email message[3]. Email forensics can also be defined as the process of collecting and analysing email data such as archives and server logs, to establish a sequence of email communication events[9].

Owing to the ease of use of email it has now become the cornerstone of modern communication [61]. The ease of use also makes it easy to use for cybercrimes like e-mail spamming, phishing, cyber bullying, racial vilification, child pornography, and sexual harassment[67]. The different types of cybercrimes that are conducted using email necessitate the development of email forensic processes that supports the investigation of these cybercrimes. Email forensics does not only support cybercrime investigations but are also used in other types of crime where email was used. Email evidence is viewed in the same light as other forms of documentation by a court of law.

Email forensics has been used to calculate the probability that an author is the actual author of a specific email or to authenticate a user while the user is using their email application [68, 69]. Both these processes of validating the author of an email needs sample data to build the validation. The process of validating an email author becomes more precise, the more sample data can be collected and used.

Email forensics has also been used to trace the origin of email messages. The problem with tracing the origin of any email is that SMTP header data can be manipulated due to the fact that, by default, it is not encrypted. This means that the SMTP header data integrity needs to

be validated in some way to ensure the data was not manipulated. If the SMTP header data was manipulated, tracing the origin of the email would be impossible.

In addition, to validate SMTP header data, email evidence could have been lost by the time an incident is reported. By the time an incident is reported, evidence of the incident could already be lost due to illegal user or system action. The following section discusses Digital Forensic Readiness which is intended to address the problem of lost evidence.

## 4.4 Digital Forensic Readiness

Rowlingson[7] defines digital forensic readiness (DFR) as consisting of two objectives. The first objective is to maximise the environment's capability of collecting digital forensic information and the second objective is to minimise the cost of a forensic investigation. ISO 27043 [18] defines DFR as the process of being prepared for a digital investigation before an incident occurs.

Preparing any environment to be digital forensically ready, a mechanism will need to be added to preserve, collect and validate the information contained in the enviroment. The information gathered from the environment can then be used as part of a digital forensic investigation. The rest of the dissertation shows the research done by the author.

Two possible DFR processes are discussed in this disertation. The first, discussed in Chapter 6 Section 6.3, makes use of performance monitoring software to collect digital forensic data. The second, discussed in Chapter 7, uses network tools to collect TCP/IP packet data. Background on designing a DFR process is given in Section 4.4.1. Section 4.4.2 gives background for using performance monitoring tools to collect digital forensic data from the email systems. Using network monitoing tools to collect digital forensic data is discussed in Section 4.4.3.

### 4.4.1 Designing a DFR process

According to ISO 27043 [18] designing a DFR process is broken into a number of processes that need to be performed to ensure that the DFR process is digital forensically sound. Figure 4.1 depicts the Readiness design process steps, as discussed in ISO 27043, which depicts three process groups called the planning process group, the implementation process group and the assessment process group. The first step of the process is to define the scenarios

where digital data might be required. For this dissertation the scenario is to collect digital data to perform an email forensic investigation.

The next step is to identify possible sources of digital evidence that could be needed in an email forensic investigation. After the possible sources for digital evidence are determined, planning for the collection, storage and handling of the digital evidence needs to be defined. Once the planning for pre-incident collection, storage and handling is defined, the pre-incident analysis of the data needs to be defined. This step depends on the scenario and the digital evidence sources.

Pre-incident analysis can be used to define an incident detection process. The incident detection process could be implemented using a performance monitoring tool that reads the analysed data for a spesific incident definition. Next, the information system architecture, that the DFR is being defined for, needs to be created. The information system architecture refers to the organizational structure of the information system including all the hardware, software and personnel that form part of the sytem. The information system architecture design needs to capture the aims of the DFR as well as what digital evidence sources will be collected and how to collect the evidence.

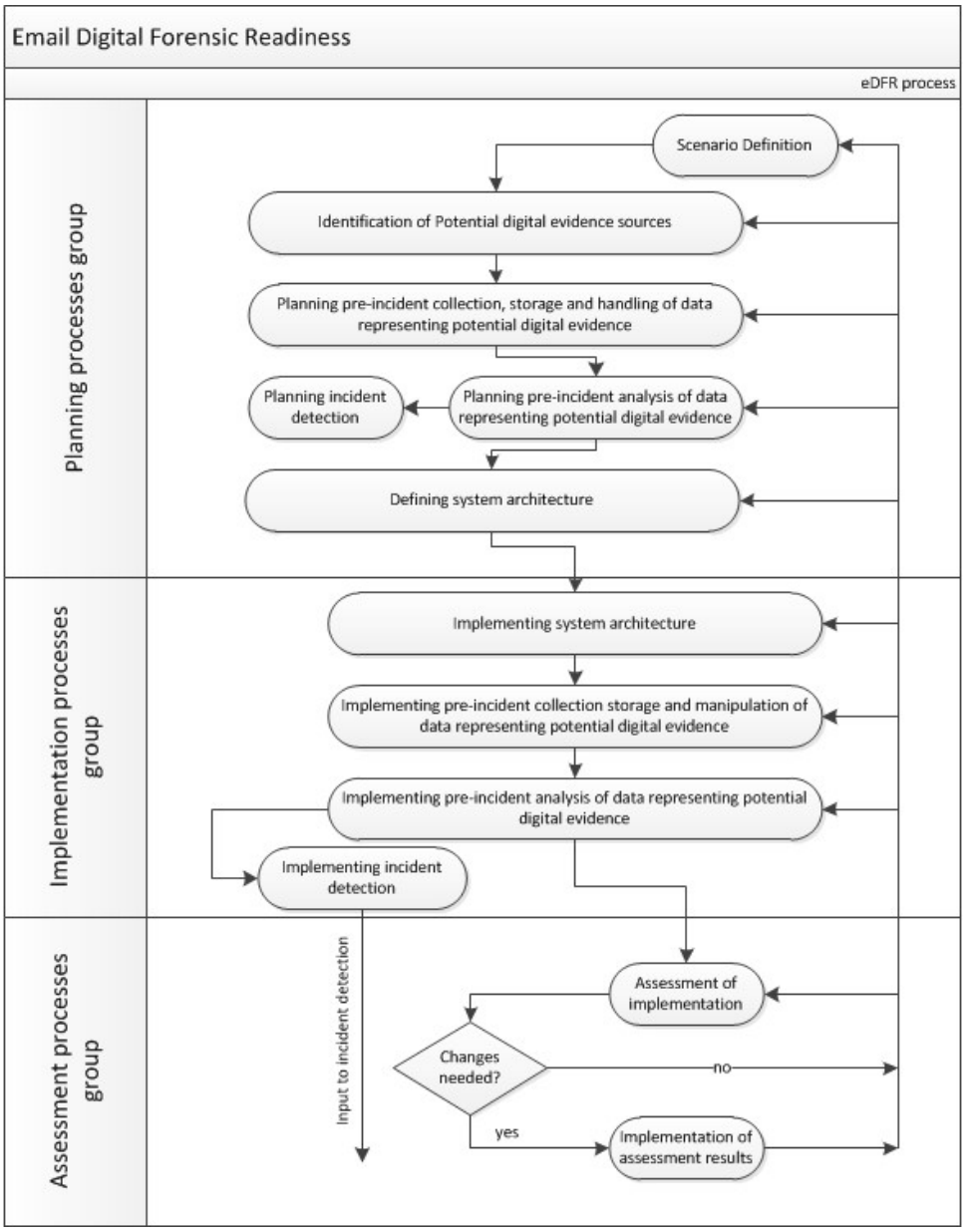


**Figure 4.1: Readiness design process as defined in ISO 27043 [18] reprinted with permission.**

After the information system architecture is defined, the next step is to implement the design. This process includes the installation of new hardware, software and/or the policies needed to support the DFR process. The first step of implementing the design of the DFR is to implement the pre-incident collection, storage and handling of data, as defined in the information system architecture during the designing of the DFR. The implementation of the pre-incident analysis and incident detection designs are next. As part of the incident detection

implementation, an interface must be created to the monitoring system, that will send a notification of a possible incident.

After the DFR is implemented, the implementation process should be assesed to ensure that the aims of the DFR are actually met. This should be a technical as well as a legal review of the DFR process. Any problems, changes or enhancments are documented and the DFR process must be adapted to include the changes. All changes to the DFR should be documented and assesed to ensure that the DFR does in fact support the aims that it was designed for. If the scenario changes or new aims are defined the DFR should be adapted accordingly. The design of the email system DFR is based on this process and will be discussed later in the dissertation.

### 4.4.2 Performance monitoring tools

One possibility for using performance monitoring tools is to collect and store digital forensic data for later analysis. Section 4.4.2.1 gives a general overview of performance monitoring tools. The author feels that it is important to understand how performance monitoring tools work so that the value of using them as a digital forensic tool can be understood. Section 4.4.2.2 gives an overview of the protocols that are used by performance monitoring tools. This is to show that performance monitoring tools use forensically sound methods for collecting performance data.

#### 4.4.2.1 Overview of Performance Monitoring tools

Performance monitoring tools are designed to collect data about software systems in order to report on performance, uptime and availability[70, 71]. Live data about the monitored systems is used to detect system problems and pinpoint the source of the problem. The data is collected from all the different layers in the system from the network layer to the application layer to give a holistic overview of system performance.

Security monitoring tools are implemented in a system to collect data on security concerns like intrusion detection events, malware detection and performance attacks. Each device in a system contains a security probe that is used to collect security data from each device. The data collected is used to generate warning events if needed and for system security reports.

Data used by monitoring tools are categorised into live data, historical data and custom data. All data collected are stored in read-only tables on the monitoring system database. Live data is data that was collected during the latest completed collection cycle and is used to display the current system state. Live data can be kept for a period of time to show the changes in system state over time. After a set period of time the live data is reformatted and moved to the historical data set. Historical data is used to calculate the performance, uptime, and availability and produce security reports on enterprise systems.

Custom data is data collected by the performance monitoring system but not used for displaying system state or reformatted to become historical metric data that trend analysis is performed on. Monitoring systems normally do not understand the meaning of custom data unless extensive customisation is done to the monitoring tool. Descriptors can be created for custom data, to explain the meaning of the data to the monitoring tool and to generate alerts that activate, based on the collected data.

A new trend in software management is to combine security and performance monitoring into an integrated system. These integrated systems are used to implement System Performance and Security Management (SPSM). SPSM is implemented with a central control system so that all aspects of enterprise systems can be monitored and controlled from a central console. Monitoring tools use different external protocols to ensure the data collected is validated and verified. The following section gives a short overview of these protocols.

#### 4.4.2.2 Protocols used by performance monitoring tools

Some performance monitoring tools make use of the Simple Network Monitoring Protocol (SNMP) that is specified in RFC5590[72]. SNMP specifies a method for connecting to and collecting data from servers, firewalls and network devices, mainly for the purpose of performance monitoring.

The Lightweight Directory Access Protocol (LDAP) [73] defines a way to access distributed directory services. This paper makes reference to an open-LDAP implementation. Open-LDAP is an open source implementation of the LDAP specification, used as a single point of authentication for users accessing network resources. Performance monitoring tools use LDAP to authenticate the system users that are used to run the system monitoring probes.

According to RFC4251 [74] Secure Shell (SSH) is used to implement secure communication over an insecure network. Performance monitoring tools use SSH to establish communication between the probes and the performance monitoring server. This is dependent on the performance monitoring tool that is implemented and the architecture of the performance monitoring tool. The implication is that the integrity of the data sent to the performance monitor by the probes, is assured.

### 4.4.3 Network monitoring tools

Network monitoring tools make use of programs like tcpdump [75] to collect TCP/IP packets from the network interface and store it in a standard .pcap file. The pcap file is based on the libpcap standard which was originally developed by the tcpdump team at Lawrence Berkley Laboratory [76].

WireShark is an open source application that is used to analyse network traffic[77] and is used by the digital forensics community to analyse network data captured using libpcap[78, 79].The '.pcap' files generated by tcpdump can be loaded into WireShark and analysed. Part of the WireShark analysis process is to reconstruct the TCP/IP packets to view the actual payload. Chapter 7 makes use of tcpdump and WireShark to show how research was conducted by collecting TCP/IP packets that contained SMTP data. The following section discusses the use of big-data with DFR processes.

## 4.5 DFR processes, Big-Data and Hadoop

Big-data as a concept originates from a paper [80] by NASA scientists, describing a problem that they had with visualization. In this paper big-data is described as "data sets are generally quite large, taxing the capacities of main memory, local disk, and even remote disk. We call this the problem of big-data. When data sets do not fit in main memory (in core), or when they do not fit even on local disk, the most common solution is to acquire more resources." The Oxford English Dictionary describes big-data as "data of a very large size, typically to the extent that its manipulation and management present significant logistical challenges." Gartner [81] defines big-data as "high-volume, high-velocity and/or high-variety information assets that demand cost-effective, innovative forms of information processing that enable enhanced insight, decision making, and process automation."

Some definitions try to differentiate between big-data and big-data analytics. The concept of big-data has to do with the storage of large structured and unstructured data sets in such a way that they could be analysed [82]. Big-data analytics is the process of using structured and unstructured data sets to generate usable insights that can be used in decision making [83].

From these few definitions the author deduced that big-data is a concept used to describe a situation where the data collected from a process is more than the resources that can be used to analyse the data in a reasonable amount of time. Big-data has been a problem for IT since the beginning of computer processing. Without going deeply into the history of computing, two examples of IT trying to face the problem of big-data are the invention of the mainframe and data storage arrays which can be used to analyse large data sets to produce reports.

Digital Forensics have the same problem in that implementing DFR processes for digital systems causes a huge amount of system data to be collected and stored for analysis[84, 85]. Most of the data collected by DFR processes might never be used during a digital forensic investigation but knowing what data to keep and what data to discard is almost impossible[86]. The saying “Better to have it and not need it than need it and not have it” comes to mind. DFR processes can almost be seen as digital hoarding. DFR processes are sometimes the only way to collect digital forensic data that might be needed during a digital forensic investigation.

An example of such data would be log files from email systems that contain a record of sent and received emails. The log files are not kept for a long time due to constraints on system resources but sometimes, while investigating a system problem or an incident, it would have been beneficial to have log files that are already overwritten. A DFR process can be implemented to collect the data from the log file and store it somewhere else.

Collecting the log files using a DFR process would allow the log files to be available when they are needed. The problem is that collecting and storing all the log files, to possibly use them later, means that a great amount of storage is used and huge amounts of data would need to be scanned to find the data that was needed[87]. Big data solutions can help to analyse the log files after collection or even during collection, to look for possible incidents[83].

There are a number of big-data analysis tools that can be used to perform analysis of large data sets. The one used as part of the research is Apache Hadoop[88]. According the Apache Hadoop wiki page Hadoop is a framework that allows for the running of applications on hardware in such a way that the hardware needs of the application can be adjusted according to the needs of the application. In other words, Hadoop allows for the scaling of application processes according to the hardware resources of the application. This allows for application processes to be distributed over as many physical hardware resources as needed by the application and managed in such a way that optimum use of the hardware resources is achieved.

Hadoop implements a distributed file system and process scheduler that allows multiple applications to perform processing on distributed hardware platforms without knowing that other applications are performing processing. This means the Hadoop application processing management separates application processing in such a way that if the application is not designed to share data resources other applications will not be able to access the data resources. The implication of using Hadoop for big-data analysis is that the process developed to perform the data analysis can be deployed and adjusted according to the size of the data set, without applications intruding on each other. The design of the Hadoop deployed application and the function of the application is discussed later in the dissertation.

The following section gives a short conclusion of the chapter.

## 4.6 Conclusion

Digital forensics, Email forensics and Digital forensic readiness were discussed in this chapter. The research conducted for this dissertation is based on the digital forensic process but focuses mainly on a DFR process for Email defined later in the dissertation.

The DFR process for Email is concerned with identifying sources of digital evidence in a digital system. Email forensics is focussed on performing digital forensic investigations on digital evidence gathered from email systems. Different options for collecting digital forensic data from email systems are discussed in the dissertation. The use of performance monitoring tools and network monitoring tools were described in this chapter and is used later in Chapter 6 and Chapter 7 as part of the research that was conducted.

A number of sources showed that DFR processes causes a huge amount of data to be stored that then have to be analysed by investigators. The analysis of these large data sets needs to be automated to make it easier for investigators to perform investigations. That is why Big-Data and how big-data technologies could help as part of a digital forensic process was also discussed. The following chapter shows the research flow and how that fits with the remaining chapters.

**Part III: Planning**

**Part III** of the dissertation contains **Chapter 5** which discusses the design of the **email Digital Forensic Readiness (eDFR)** architecture that is based on the readiness design process described in ISO 27043 [18]. The eDFR was designed to capture digital forensic data from email systems. Digital forensic data is needed to trace the origin of spam.

# Chapter 5 : Planning the eDFR architecture

## 5.1 Introduction

Planning of the email Digital Forensic Readiness (eDFR) architecture is discussed in this Chapter. Figure 5.1 shows the Readiness design process that is adopted from the harmonised digital investigation process schema of ISO 27043 [18] as discussed in Chapter 4. Figure 5.1 indicates how the eDFR architecture is implemented, discussed in each chapter that follows. The eDFR architecture should enable email forensic investigations, for example tracing the origin of email as long as the SMTP header data is not spoofed, or allow for the detection of spoofed SMTP header data.

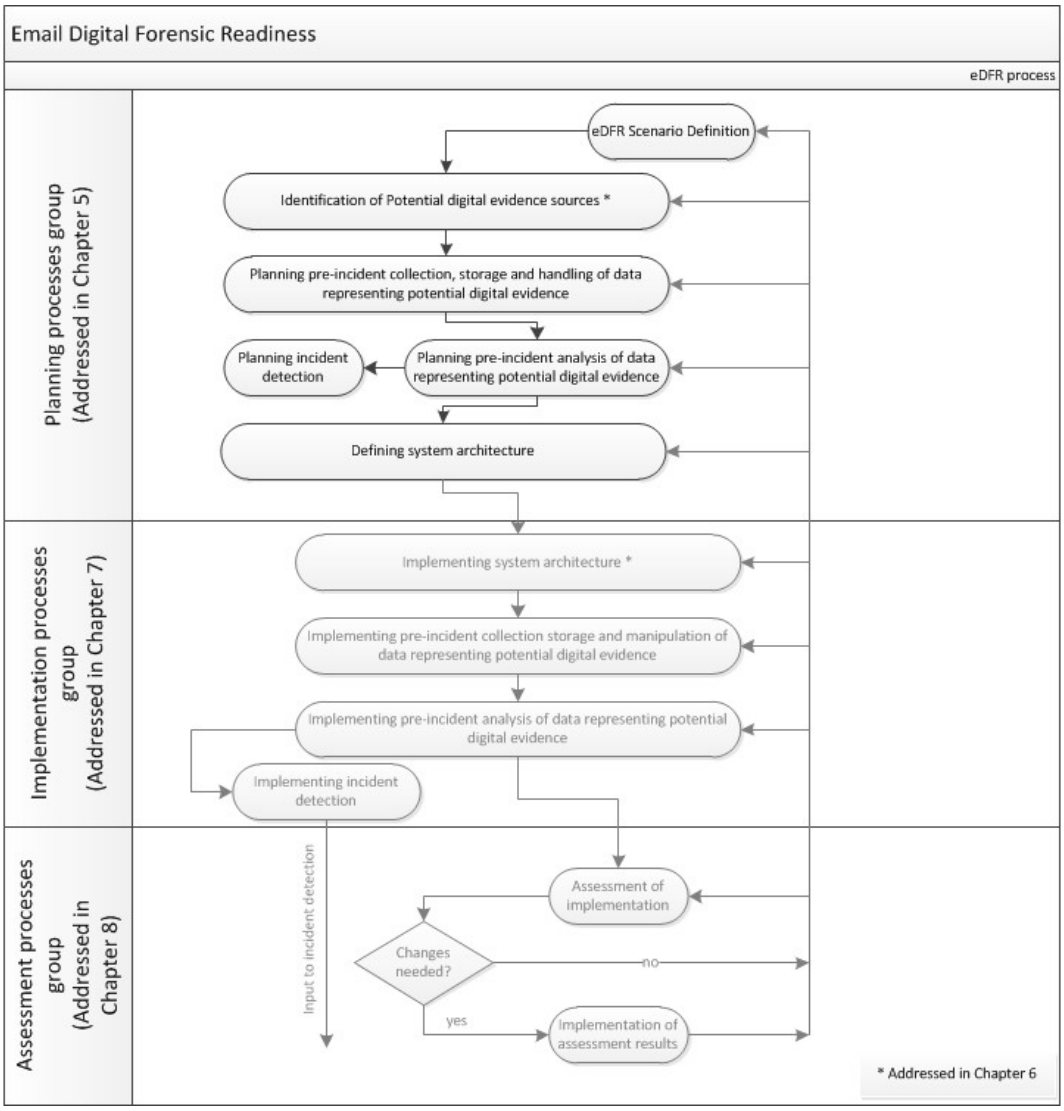


**Figure 5.1: eDFR creation process according to ISO 27043**

Figure 5.1 depicts three process groups called the planning process group, the implementation process group and the assessment process group. This chapter focuses on the planning process group, i.e. to enable the planning of an eDFR architecture. The following sections discuss the different processes that form part of the planning processes group as shown in Figure 5.1.

The first step of the planning process is to define the scenario that the readiness process is being designed for, which is discussed in section 5.2. Section 5.3 discusses the different potential sources of digital evidence that could possibly be collected. Planning the collection, storage and handling of the digital evidence is discussed in Section 5.4. The planning for the

analysis of the data is discussed in Section 5.5. Part of the analysis of the data serves to detect possible incidents in the information system that the DFR collects data from. The planning for the incident detection is discussed in Section 5.6. The last part of the planning processes group is to define the information system architecture which is discussed in Section 5.7. The chapter ends in Section 5.8 with the conclusions reached.

## 5.2 Creating the Scenario definition

The readiness process starts by setting the scenario that the eDFR architecture must cater for. Data needed for an email forensic investigation needs to be collected while the email system is active. The data collected should support multiple email forensic investigations for it to be economical to implement.

A number of case studies for email forensics were consulted to review the data that was used for email forensic investigations. Email forensic investigations are conducted to confirm the author of an email, to find the origin of an email, to determine the flow of information between suspects, called discussion threads, and/or to find proof of a cybercrime[67, 68, 78, 89, 90].

When confirming the author of an email, a number of emails, that are known to be written by the author in question, are used to profile the author's writing style[91]. The writing style profile is used to determine if the email in question was in fact written by the author in question. For this type of investigation a sample set of emails from the author is needed to create the writing profile. These emails should come from a source that can verify the authorship of the emails.

Investigating the origin of an email is historically done manually. The SMTP header of the email is collected, containing data such as the received header data, from a mailbox and analysed to determine the origin of the email. The problem with this method is that the received header data could contain spoofed address information. As discussed in Chapter 2, the data in the received header can easily be spoofed. It is conceivable to then assume that all received header data is spoofed unless it can be determined that it is not spoofed. The data collected from the received header should be tested for spoofing before it can be used in an investigation to trace the origin of an email. The type of data that would be needed for this investigation is the SMTP headers of the email in question and some form of corroborating data that can show the SMTP header data is not spoofed.

Determining the flow of information using email makes use of a set of emails that form an email discussion thread. The emails are linked together using the sender and receiver data in the SMTP headers as well as the data in the body of the email. This type of investigation needs all the emails that could be part of a discussion thread or one email that contains the entire conversation. Some email clients such a google web client has the capability to structure the contents of the email inbox into discussion threads. This happens when a user replies to an email. The old body of the previous email is kept in the reply email unless the reply email is edited and the old email text is removed. If a user forwards and email to another recipient the email is seen to be leaving the original conversation thread.

To find proof of cybercrime, for example phishing attacks, investigators make use of email that has been determined to be suspicious, by a user or an investigator or an automated process like an anti-spam service. The full email is analysed to collect data that will confirm the suspicious nature of the email. The type of data needed would be the full suspicious email with all the attachments and the body of the email.

Each of the different types of investigations uses different aspects of email data. Therefore, the scenario for the eDFR architecture is to collect digital evidence that can be used in all email forensic investigations.

From the definition of the scenario the aims of the eDFR architecture is defined as:

- Collect data that will support:
    - Authorship investigations
    - Determine the flow of information
    - Detect cybercrime activity (for example phishing attacks)
    - Trace the origin of spam
    - Detect spoofed email

Now that the scenario is set to as broad as possible the next step is to identify the potential digital evidence sources. The scope of the scenario is defined as the email forensic investigation that the eDFR architecture will support. Future mentions of email forensic investigations in this dissertation are meant to mean the scope of the scenario defined here. The following section discusses the identification of possible digital forensic sources.

## 5.3 Identification of potential digital evidence sources

Suitable points to collect potential digital evidence should be identified so that the data collected can be used for all the email forensic investigations defined in the previous section. One collection point for possible digital evidence is the SMTP headers. The SMTP headers should contain data that will be useful for email forensic investigations. The problem is that the SMTP header can be spoofed; therefore, a mechanism should be implemented to safeguard the digital evidence in the SMTP header. The email body and attachments are also sources of possible digital evidence which should be collected with the SMTP headers. Possible digital evidence could also exit in the email system log files. The log files contain information on the activities of the email system.

The identification and collection of suitable digital evidence in email systems, is discussed in detail in Chapter 6. A number of different collection options and collection points are investigated to find the source of potential digital evidence with the best potential to contain all the data needed to support the scenario definition. A collection example would be to collect SMTP header data from mailboxes or to collect log file data from the email system components. The following section discusses how the data should be collected, stored and handled as part of the eDFR architecture.

## 5.4 Planning the collection, storage and handling of data

The collection of the potential digital evidence should be designed in such a way that the collection process does not impact the email system. The data should be collected while the email system is active so as to minimize downtime. The collection process should also collect data before there is a possibility that the data could be lost. Loss of potential digital evidence could occur when a user deletes an email from a mailbox or when log file data is cleared from the email system to open up space on the system drives due to storage drive space limitations.

The storage of potential digital evidence should be done away from the control of the email system so that the potential digital evidence collected could not be changed or removed from any mechanism in the email system. This will help ensure that the collected data will not be lost accidentally or on purpose when maintenance is performed on the email system. The data storage medium that is used to store the data should ensure that the collected data cannot be changed, removed or replaced under any circumstances.

The handling of potential digital evidence should be controlled in such a way that the original data is never removed from storage. When an investigator needs data for an investigation, the original data in the database should be cloned and the cloned data should be provided to the investigator, so that the original data should remain intact.

The mechanisms needed to ensure that the collection, storage and handling of the data are correct, is discussed in detail during the implementation process group discussions in Chapter 7. The following section discusses the planning for analysing the potential digital evidence.

## 5.5 Planning the analysis of the data

The analysis of the collected potential digital evidence is dependent on the type of investigation that could be performed. As stated before, the investigation types that need to be supported are authorship investigations, information flow investigations, cybercrime detection investigations, detecting spoofed email and tracing the origin of email.

One analysis process that must be implemented is to detect spoofed SMTP headers. The potential digital evidence can only be used as part of a digital forensic investigation if the evidence collected can be validated. The analysis process should be able to detect spoofing by using the data collected from the different collection points, as pointed out in the previous section. The detection of spoofing is discussed in detail in Chapter 7.

The following section discusses the actions that should be performed when an incident is detected.

## 5.6 Planning incident detection

When a spoofed email is detected an incident alarm could be raised. The alarm could be an input to the anti-spam system to adjust anti-spam rules. The incident alert could also be sent to the receiver of the original email to alert the receiver of possible spoofing. The incident alarm should contain a reference to the analysis data stored in the database. The following section defines the architecture of the DFR process.

## 5.7 Defining system architecture

The eDFR architecture defines the structure of the eDFR architecture that will be implemented. The architecture defines the resources needed for the implementation and includes the hardware, software, storage and integrations of the eDFR architecture. The

design starts by reviewing the aims of the eDFR architecture. The aims were set when the scenario for the eDFR architecture was defined. Figure 5.2 depicts the physical hardware and connectivity requirements for the implementation of the eDFR architecture.

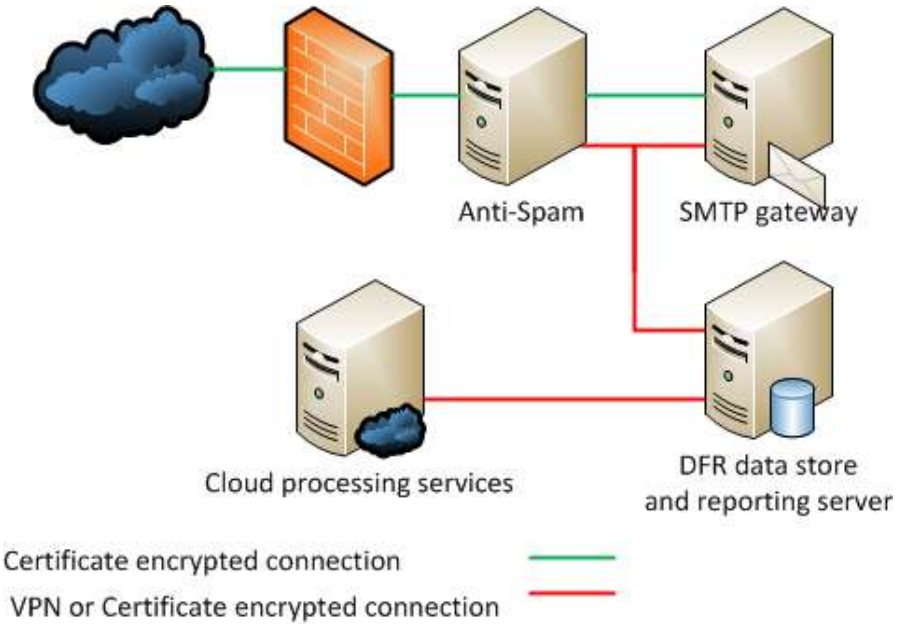


**Figure 5.2: Hardware and connectivity needed for the eDFR architecture**

Hardware resources such as the internet connection, Firewall, anti-spam server and the SMTP gateway should already exit since the existence of these components necessitated the creation of the eDFR architecture. Each of the existing components should be linked with a network connection that implements secure communication between the components. Secure communication can be achieved by using Public Key Infrastructure (PKI) encrypted connections as, depicted in Figure 5.2.

Vacca [92] states in his book that PKI enables users of insecure public networks such as the internet to securely and privately exchange data through the use of a public and private cryptographic key pair that is shared through a trusted authority. PKI provides for digital certificates that can identify individuals or organizations and directory services that can store and, when necessary, revoke certificates.

The added components needed for the eDFR architecture implementation is the eDFR data store and reporting (DFRDSR) server and the cloud processing service. The DFRDSR server is used to store the data collected from the anti-spam and SMTP gateway servers and the data

produced by the different analysis processes. The connections between the anti-spam and SMTP gateway servers and the DFRDSR server must be secured to ensure the data cannot be intercepted or changed during transmission. The secure connection can be implemented using Virtual Private Networking (VPN) or using certificate encryption.

The cloud processing service depicted in Figure 5.2 can be implemented as a self-hosted or externally hosted service depending on the organization. If the cloud service is self-hosted the communication between the cloud processing service and the DFRDSR server can be implemented the same way as the connection between the anti-spam server and the DFRDSR server. If the cloud processing service is implemented using an externally hosted solution the external host must ensure the security of the processing environment and must ensure that the chain of custody is preserved. How this is achieved is outside the scope of this dissertation. The implementation of the data collection process is discussed in more detail in Chapter 6.

## 5.8 Summary

This chapter discussed the planning process group of the eDFR architecture. The planning set the outline for the research that is discussed in the rest of the dissertation. This chapter does not provide extra insights or conclusions for the dissertation but rather sets the framework that describes the reasoning for the layout of the dissertation.

Section 5.2 discussed the scenario for the eDFR architecture. The scenario for the eDFR architecture was set so that the eDFR architecture could be implemented for all possible email forensic investigations. Designing the eDFR architecture to support all possible email forensic investigations increases the value of the proposed architecture. Due to the fact that the economic viability of architectures [93] are measured by the value that they add, the higher the value of a DFR process, the higher the probability that the process would be successfully implemented.

Section 5.3 discussed the identification of possible digital forensic data in email systems. According to ISO 27043 the implementation process and the assessment process should feed back into the planning process. The different sources of potential digital evidence were not discussed in detail as part of the planning process for the sake of brevity but, as stated in this section, are discussed in more detail in Chapter 6.

Section 5.4 discussed the planning that was performed to collect, store and handle the possible digital evidence that could be collected. The main requirements stated were that the collected potential digital evidence should be kept on a different server with the appropriate data protection in place to ensure that the data cannot be tampered with or removed.

Section 5.5 discussed planning the analysis of the collected data. As stated, the main focus of this dissertation is to trace the origin of spam, therefore the main analysis focuses on detecting spoofed SMTP headers and tracing the origin of email. These analysis processes are discussed in more detail in Chapter 7, as part of the implementation process.

Section 5.6 discussed the planning for incident detection. The focus of this dissertation is not to create incident alerts with regard to email but a proposal was made to implement an incident alert when spoofing is detected. As stated, this alert could serve as an input to anti-spam services to add rules to block email spoofed in such a way. The main concern would be that the analysis process would not be responding fast enough to alert the anti-spam system in real time.

Section 5.7 discussed the planned system architecture that would be needed to implement the eDFR architecture. This focussed mainly on the hardware that would be needed and not on the software requirements. The software requirements are discussed in Chapter 7 and are based on the research conducted in Chapter 6.

Part IV of the dissertation follows, containing Chapter 6, which discusses the research conducted to find possible digital evidence, and Chapter 7, which discusses the implementation process group for the eDFR architecture.

**Part IV: Implementation**

**Part IV** of the dissertation contains Chapter 6 and Chapter 7. **Chapter 6** discusses the research that was conducted to **find digital evidence** in email systems. Defining the digital forensic data that could be collected from email systems is part of the **implementation process** group of the ISO 27043 Readiness design process. **Chapter 7** gives an in depth discussion on the **implementation of the eDFR** architecture according to the planning done in Chapter 5 and the research conducted in Chapter 6. The implementation discussion shows how the data is **collected, handled, stored and analysed** using the eDFR architecture.

# Chapter 6 : Identifying sources of potential digital evidence for the eDFR architecture

## 6.1 Introduction

This chapter endeavours to answer the question of which data sources could be used to perform a digital forensic investigation to trace the origin of spam. The research conducted in this chapter is seen as part of the planning process group and the implementation process group. This chapter is part of the planning process group because it discusses the investigations that were conducted to find sources of digital forensic data. It is also part of the implementation process group because the outcomes of the research conducted in this chapter are used to implement the eDFR architecture.

When conducting an email forensic investigation the data is normally collected from the SMTP header of an email. The problem is that SMTP is a clear text protocol which makes it easy to spoof the data in the SMTP header. Cyber criminals use spoofing to hide the true origin of an email. The easy spoofing of the SMTP header means that the data in the SMTP header cannot be trusted to be accurate, due to its unencrypted nature. Therefore, before the data collected from the SMTP header of the email can be used to perform an email forensic investigation, the data must first be corroborated to ensure the data was not spoofed.

Another problem that was identified during this research was that the SMTP header only contains SMTP routing information when the email system is implemented using SMTP forwarding. The most used email systems today is based on the Internet messaging architecture (IMA) as discussed in section 2.5.1. IMA is based on IP routing so no routing information is kept in the SMTP header.

Automating the collection of SMTP data from mailboxes is discussed in section 6.2 which is based on the manual collection process that is currently being used by current email forensic investigators, to collect digital forensic data. Section 6.3 discusses the addition of hash values to the SMTP and IP headers in order to validate the data in the SMTP and IP headers. Section 6.4 discusses an attempt that was made to use performance monitoring tools to collect digital forensic data from log files. Section 6.5 discusses attempts to collect data that currently exist in email systems, to validate the data in an unaugmented SMTP header. These collection points and methods, i.e. SMTP mailbox, SMTP and IP headers, email system log files and

TCP/IP packet data, were researched to identify the best method and point for collecting potential digital evidence.

Section 6.6 discusses the different investigations that were conducted to find digital evidence and the usability of the proposed collection processes. Not all the identified sources of digital evidence will be used in the implantation of the eDFR but it is felt that the full investigation should be discussed so that the rationalization of the final implementation can be understood. The chapter ends with the conclusions that were reached in Section 6.7.

## 6.2 Automating the collection and storage of SMTP header data

Collection of SMTP header data is normally a manual process. The investigator will copy all the original emails to tamper proof storage and only then start the investigation. Email that is received by an organization should be archived on an email archive server but that is not always the case.

The email needed for an investigation might not exist anymore by the time an investigation is started. Implementing an automated collection process will ensure that all email data is saved in a tamper proof environment, for later use. Owing to the amount of email an organization could send and receive, some organizations do not archive spam, only email that is delivered to the SMTP gateway. Therefore, collecting SMTP data from the archive server would not be helpful for the overall research that was being conducted. A decision was made to collect the SMTP data from the anti-spam server. This would allow the collection of all emails that are sent from or received by an organization.

A program was written using a scripting language that hooked into the anti-spam system. The SMTP data was routed through the script to the anti-spam application. Data from the script was then stored in a custom designed SMTP data table on the DFRDSR server. The SMTP data table was created as an archive table, meaning that data could be added to the table but data could not be edited or deleted [94]. Figure 6.1shows the structure of the email database tables.

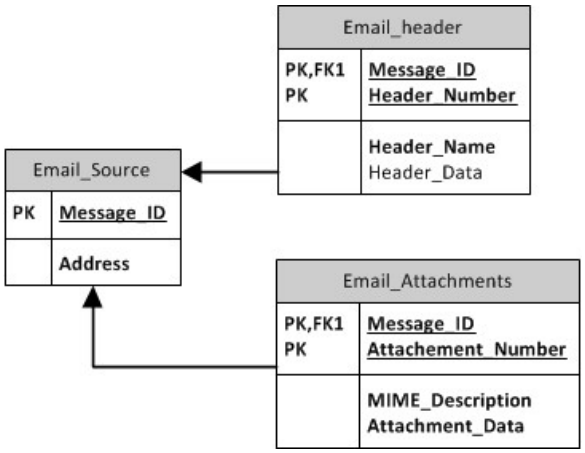


**Figure 6.1: SMTP data collection database tables**

The Email_Source table records a Message_ID and the email address of the receiver. Each email that is sent by an SMTP server is issued with a unique Message_ID. Using the Message_ID as the primary key in the table links the data in the different tables together.

From experimentation it was concluded that not all email systems use the same number of SMTP headers or contain the same structure of SMTP headers. Due to the difference in SMTP header structure of different email systems each SMTP header was saved as its own record in the database, with a header name and header data row. The Email_Header table is used to store all the email header data. A third column was added to store the message-ID of the email, to keep all the headers of the same email together. A header number was added so that the email headers can be placed back in order to recreate the email. The header number and the message-ID together make the primary key of each record.

Another table was created to store the attachment data. Attachment data can be any type of media that could be described in a MIME header. To ensure that attachment content is stored effectively it needs to be stored using a media data type. The Message_ID and Attachment_number are used to sort the attachments of an email. Figure 6.2 shows the process that the script uses to collect and store the SMTP header data.

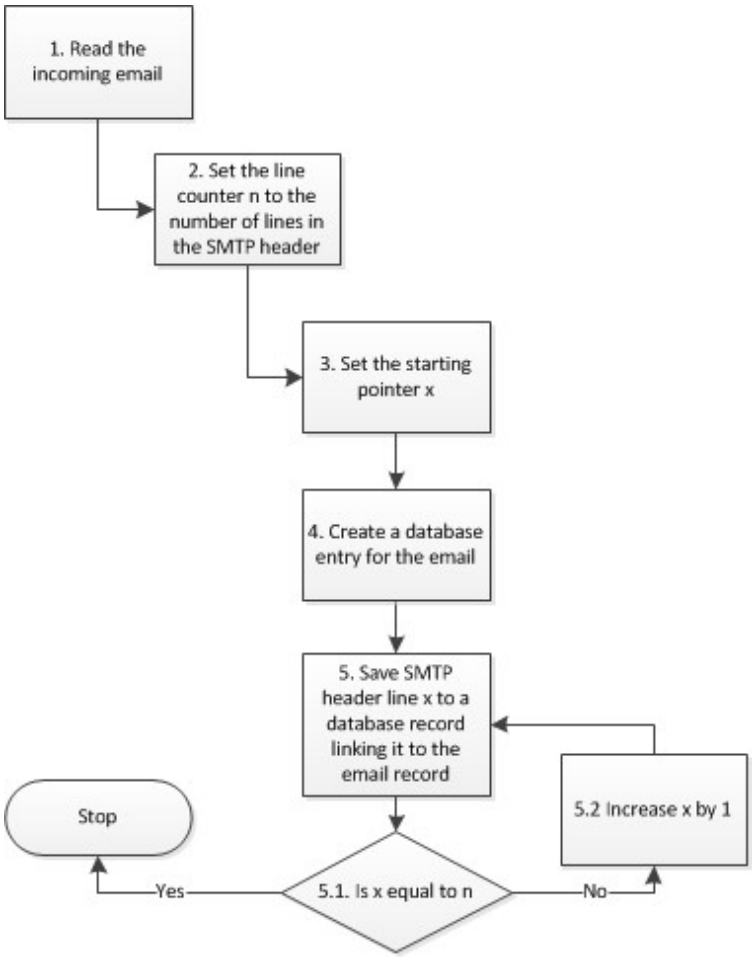


**Figure 6.2: SMTP header data collection process**

Step 1 of the SMTP header data storage process, as depicted in Figure 6.2, is to read the incoming email. This is achieved by setting the script to listen for traffic on the SMTP port or by routing SMTP traffic to the port that the script listens on. Step 2 of the process is to set up a counter for the number of lines that exist in the SMTP header. The counter is needed since not all emails have the same number of SMTP headers as not all SMTP servers implement the SMTP specification in the same way. Step 3 sets the starting point for reading the SMTP header data.

Step 4 creates a record of the email in the database that will be used to connect the SMTP header data together, to enable the rebuild of the email. Step 5 reads the first SMTP header and saves it to the database as a single record. Step 5 is repeated until all the header data is saved in the database.

The storage process was designed so that an email can be rebuilt from the stored data. This means that the stored data can be verified by comparing the original email and the rebuilt email. The database tables are defined as archive tables which imply that data can be added to the tables and read from the tables but the data cannot be edited or deleted. Storing the data in the database helps with searching for specific data sets. Data mining processes could also be implemented to search for patterns in the data. The field of data mining is not within the scope of this research and is not discussed here. The following section discusses the augmentation of the SMTP and IP headers to validate the data in the SMTP header.

## 6.3 Adding validation to SMTP header

Initial attempts to collect digital forensic data from email consisted of augmenting the SMTP and IP headers to add hash values to the headers [95]. The hash values could then be used to validate the data in the SMTP and IP headers.

Section 6.3.1 discusses the augmentation of the SMTP header and the results that were achieved mapping to subsection 1.1.1 of the research progress diagram. The process to augment the IP header and the subsequent results achieved are discussed in Section 6.3.2, mapping to subsection 1.1.2 of the research progress diagram.

### 6.3.1 Adding digital forensic data to SMTP

Adding digital forensic information to SMTP requires the addition of a hash value to verify the trace information contained in the SMTP header. According to the background discussion of Chapter 4, digital forensic information must be verifiable. Looking at the trace header, as shown in Chapter 2, the best place to store the hash value would be in the received header. Figure 6.3 shows the augmented description of the received header.

```
Received        = "Received:" 1*1received-token";" date-time CRLF

Received-token ="from" (word / angle-addr / addr-spec / domain) ":" hash
                "by" (word / angle-addr / addr-spec / domain) ":" hash
```

**Figure 6.3: Augmented Received and Receive-token Rule**

Figure 6.3 also shows how two hash values are added to the original received rule. The first hash value appears after the "from" line that contains the value of the sending domain hashed

together with the DNS lookup IP of the sending domain. The second hash value appears after the "by" line that contains the value of the host's domain, hashed together with the DNS lookup IP address of the host's domain. Creation of the hash value is done using MD5 hashing to preserve the integrity of the received-token. MD5 was chosen as an example but it was assumed that newer hashing cyphers could be implemented if needed. Figure 6.3 also depicts the changed received header, to contain at least one and only one received-token entry per line. The received header must contain at least one received token so that the digital forensic information is always present. Only one received-token entry is allowed per line to simplify the information extraction process. The addition of the received header hash token has a number of implications. The hash value will help to add digital forensic validation to the SMTP trace header but the addition of the hash value will increase the size of the SMTP header. This will impact the efficiency of transporting the email.

Adding the hash value does not guarantee that the data in the SMTP header can be validated. SMTP is an open protocol, which means that the information contained in the headers are human readable and easy to edit. This implies that anything can be added to the header or be removed from the header at any point during the SMTP transmission. Even if the hash value is added to the SMTP header by the sending server it can be removed at any point of the transmission. The removal of the hash value will only be known if the initial SMTP header is compared with the delivered SMTP header. There is no way to ensure that the hash value will not be removed. There is no way to enforce the addition of the hash value since the SMTP specification is defined in such a way that SMTP services need not implement all the services described in the specifications or the extensions but only needs a process to tell a calling server what services are available. As mentioned before, if one is able to implement PKI this problem can be solved, however, the cost would be too high for the beneficial gain implemented.

Safeguarding the token information in the header would have increased the complexity of implementing SMTP. There was, therefore, no way to ensure that the tokens were not removed or replaced during message transportation. Augmentation of the SMTP header would only be adopted if the augmentation added more value to SMTP than the cost of changing SMTP. The decision was made to abandon the augmentation of the SMTP header since it was believed that the augmentation will be too costly to implement and will not really

add value to the SMTP process if the hash value could not be secured using PKI [92] or digital signatures based on PKI, therefore the hash process was not adopted.

Since adding a hash value to SMTP is not a viable option, it was decided to find another layer where it would be better to add the hash value. IP, however, already has security extensions built into the IPsec standard. At this stage the researcher explored the next avenue of the research by attempting to add hash values on the IPsec protocol since the hash should be secured using IPsec extensions without the need to use PKI. The following section discusses the addition of a hash value to secure data in the IP headers.

### 6.3.2 Adding digital forensic data to IP

A decision was made to try and add the hash value to the IP header instead. The idea is that the IP header might be a better place to try and implement the addition of the hash value since the IP header is lower down in the network transportation stack. The lower down in the transportation stack the harder it would be to spoof the protocol headers due to the level of expertise needed to spoof lower level protocols. The proposal is to create a new optional header called the messaging header that will capture the hash value. Table 6.1 and Table 6.2 show the field names and sizes of the messaging header for IPv4 and IPv6 respectively. Hash data is added to the routing header to create the messaging header. The messaging header is an extension of the routing header in that it contains the same data as the routing header, except for the added hash value.

**Table 6.1: Message header field names and sizes for IPv4**

| Field name | Type | Length | Pointer | Route data | Hash data |
|---|---|---|---|---|---|
| Size in bits | 8 | 8 | 8 | length * 4 | length * 4 |

**Table 6.2: Message header field names and sizes for IPv6**

| Field name | Next Header | Length | Type | Segments left | Address data | Hash data |
|---|---|---|---|---|---|---|
| Size in bits | 8 | 8 | 8 | 8 | length * 8 | length * 8 |

Information in the messaging header is updated by each router before the message is sent to the next router. Figure 6.4 gives a small example of the transportation process focussing on the message header. The messaging header only shows the IPv4 addresses but it will also work for IPv6 addresses.

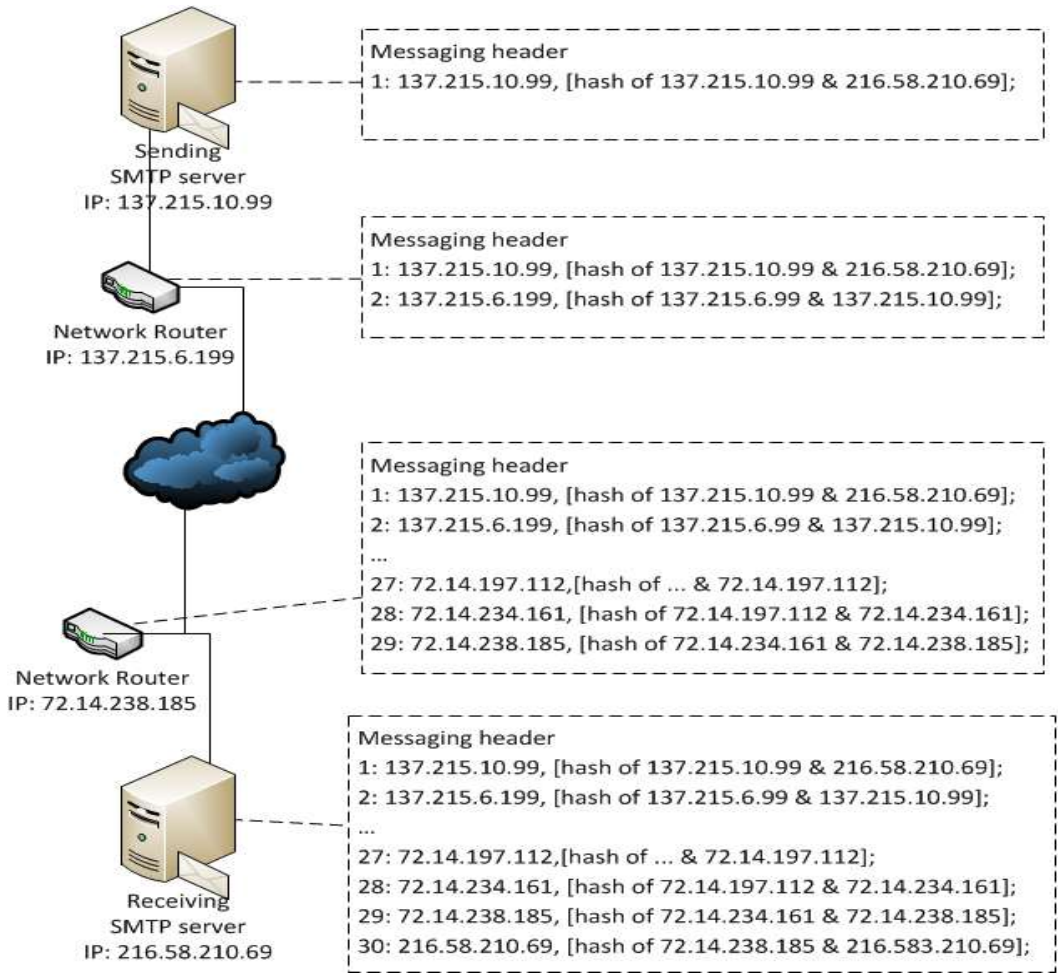


**Figure 6.4: Showing how the message header grows**

At the origin, the messaging header is loaded with the origin's address and a hash value of the origin address and the destination address. This can be seen in Figure 6.4 where the origin address is 137.215.10.99. The hash value is not shown in the figure but a place holder is shown. Generating the hash value is done by using the MD5 hashing function. Initially, the hash value is generated using the origin address, 137.215.10.99, and destination address, 216.58.210.69. After the initial entry, every hash value is generated using the sending router address and the receiving router address. This means that the hash value in the second line in the message header is created by hashing 137.215.10.99 and 137.215.6.199 together.

Although Figure 6.4 does not show IPv6 addresses the process for both IP versions are the same except for the number of bits of the hash value stored in the message header. The output is reconfigured according to the IP version of the datagram. If the datagram is IPv4 the hash value is reduced from 128 bits to 32 bits. If the datagram is IPv6, the hash value is kept at 128 bits. Before a router forwards the message, the message header is loaded with the current router's address and the hash value.

When an email is delivered to the final host, the IP header, containing the messaging header information, is saved for further processing. The last step in Figure 6.4 shows an example of the messaging header that is saved. If the email is received without the messaging header, the email is marked with a high spam probability. During the experimentation with the augmentation of the IP header it was found that saving the IP header data is more complex than previously thought. The IP header data needs to be captured before the SMTP packets are delivered because once the SMTP packets are delivered, the IP header data is lost.

A problem with adding the hash value to the IP headers was that the IP headers could possibly get very large if the route between two SMTP gateways were long. The final message header in Figure 6.4 shows a shortened list of addresses. From the shortened list it can be seen that the messaging header stores a large amount of data. If for example we assume that a route between two SMTP servers consists of 30 hops, that would mean that the message header would have 30 address and hash value pairs. IPv4 packets will have an added 960 bits in each packet of the sent SMTP packet set. IPv6 packets will have an added 3480 bits in each packet of the sent SMTP packet set. This load will increase for larger email messages since more IP packets will need to be sent to transfer the email.

Another problem was to find a way to safeguard the messaging header while it was being transported. IPsec was investigated to determine if the security services in IPsec could be used to safeguard the messaging header. Making use of IPsec secure transportation services meant that the IP headers that included the messaging header could be safeguarded.

IPsec offers a service called data origin authentication that can be used to validate the origin of the IP packet, which could be used instead of the messaging header, to validate the origin of the IP packet. When using IPsec the IP routing header can also be safeguarded which means that the messaging header would not be needed. The message header would add extra overhead and the fact that the header could be replaced by implementing IPsec led the

researcher to conclude that the messaging header would probably not be implementable. There were, however, a number of lessons learnt during the creation and testing of the messaging header, such as when IP headers are lost during the transportation process.

After the collection of the digital forensic information, produced by the addition of the messaging header, the researcher noted that the amount of data collected is too much for an investigator to analyse manually. The gap detection algorithm was therefore developed, to overcome this problem of having to deal with too much data. Although the addition of the hash values to the SMTP and IP headers did not entirely deliver the desired results, the gap detection algorithm did work in tests and could still be used as an eDFR analysis process. The next section discusses this gap detection algorithm.

### 6.3.3 Gap-detection Algorithm

The gap detection algorithm was originally developed by the researcher for pre-filtering the information contained in the received header of SMTP, as discussed in Section 6.3.1. However, after the messaging header was implemented, as discussed in Section 6.3.2, the gap detection algorithm was altered so that it can be used with the messaging header. Section 6.3.3.1 discusses the gap detection algorithm created for SMTP. Section 6.3.3.2 discusses the Gap detection algorithm that was created for IP.

#### 6.3.3.1 The SMTP Gap-detection algorithm

The SMTP gap-detection algorithm is used to detect gaps in the digital forensic data stored in the received header and to store the filtered tracing data for later use. The send-receive-pair shows one send-receive sequence during the sending process of the mail. The SMTP gap detection algorithm was published in a paper by the researcher[95].

The researcher defines a gap as a break in the receiving header when one or more send-receive pairs are not detected in the digital forensic information. The gap detection algorithm is defined as follows:

- Step 1. Store the received send-receive pairs, from last entry to first, in a queue.
- Step 2. Remember the receiving host address of the first entry in the queue.
- Step 3. Look at the next send-receive pair and compare the sending host address with the stored receiving host address. If the addresses are the same, proceed to step 4. If the addresses are not the same store the sending address and proceed to step 5.
- Step 4. Store the receiving host address in the proven list and proceed to Step 6.

- Step 5. Store the receiving host address in the proven list. Store the sending host address in the gap list and set the gap found flag to true. Proceed to step 6.
- Step 6. If the next send-receive pair is null proceed to Step 7, else store the next receiving host address and proceed to Step 3.
- Step 7. If the gap found flag was set to true, end the algorithm with the output message "gap detected" else end the algorithm with the output message "no gap found".

At the end of the algorithm two lists exist: the proven list showing the hosts that implement the augmented receiving header and the gap list showing the last known hosts that did not implement the augmented received header.

The lists can be used to trace the origin of an email if no gaps were detected. Another scenario is that the data can be used to detect spamming servers if there are gaps but the list still exists. Since the list cannot be safeguarded it might be that there will be no list or that the list itself is spoofed. This means that the data in the header and by extension the data in the SMTP gap detection lists cannot be trusted. The next section discusses the IP gap detection algorithm.

#### 6.3.3.2 The IP Gap-detection algorithm

The purpose of the gap-detection algorithm is to detect gaps in the digital forensic data, captured in the messaging header, using the hash values stored in the messaging header. Information filtered by the gap-detection algorithm is stored for later use. If the hash value cannot be confirmed, the routing information cannot be verified. The author define a gap as a break in the send-receive sequence, where one or more hash values are false.

Figure 6.5 depicts the gap-detection algorithm. Step 1 is to load the origin address, which is the first address in the messaging list, into the variable "sender". In step 2 the destination address is loaded into the variable "receiver". Initially the hash value was generated using the origin and destination address; therefore the first hash value must be verified using the same information. Step 3 generates a hash value from the sender and receiver variables to test the hash value contained in the messaging header. If the IP version is IPv4 the hash value is reduced to 32 bits.

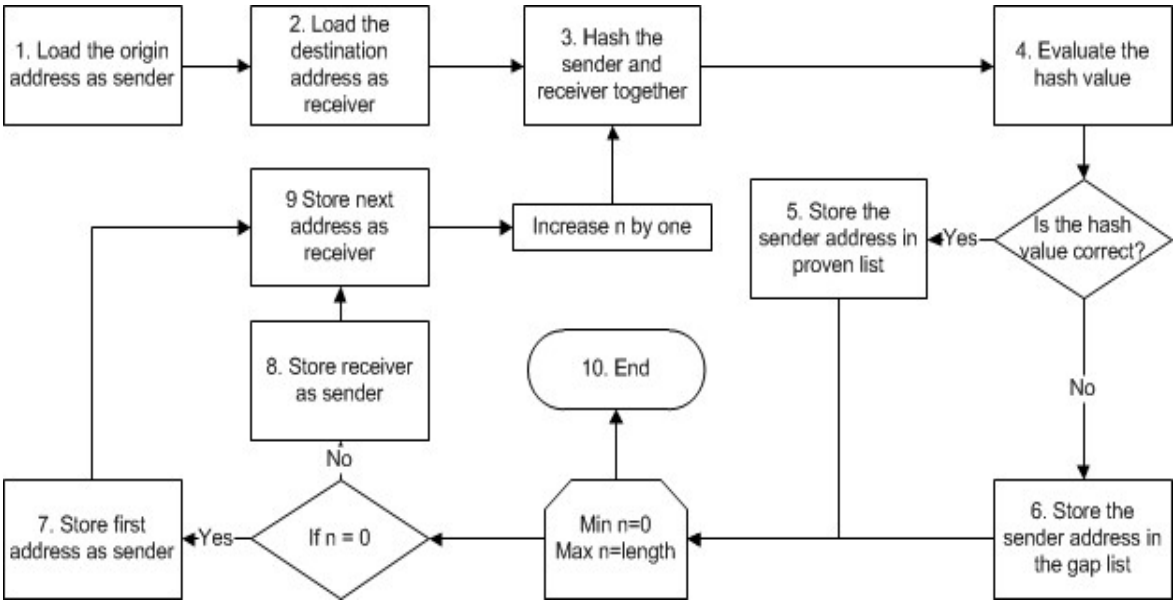


**Figure 6.5: Gap-detection algorithm depicted in a flow diagram**

Compare the created hash value with the hash value stored in the messaging header. If the hash values are the same it means that the send-receive pair is correct. The algorithm proceeds to step 5. If the hash values are not the same it means that there is a gap in the send-receive pair and the algorithm proceeds to Step 6.

Step 5 is reached if the sender address has been verified, so it is stored in the proven list. Step 6 is reached when a gap has been detected between the sender address and the receiving address and the sender address is stored in the gap list. Step 5 and step 6 feeds into a loop that steps through the address list in the messaging header. The loop starts with n equals zero and will end when n equals the length of the messaging header. If n is equal to zero the algorithm proceeds to step 7. If n is greater than zero and lower than the length of messaging header, the algorithm proceeds to step 8.

Step 7 loads the first address in the messaging header, which is supposed to be the origin's address, as sender. Step 7 is only performed once to ensure that the destination address is not used again until the end of the algorithm. Step 8 loads the current sender address as the receiver address. Step 7 and step 8 proceed to step 9 in the algorithm. Step 9 stores the next address in the messaging header, as the receiver address. The value of n is increased by one before the algorithm proceeds to step 3.

When n reaches the value of the length of the messaging header, the proven list and the gap list are safeguarded and the algorithm exits. At the end of the algorithm, two lists exist: the proven list and the gap list. The gap-detection algorithm can detect gaps in the messaging header but it cannot detect if the messaging header has been spoofed, if the hash values were also replaced. The following section discusses how information stored by the gap-detection algorithm is analysed and utilized.

#### 6.3.3.3 Utilizing the information generated by the IP gap detection algorithm

The IP gap detection algorithm produces two lists, as already discussed. The information stored in the lists can be used in one of two ways. If there were no gaps detected for a specific message, the information in the list pertaining to the message can be used to digital forensically trace the origin of the email by following the verified send-receive pairs in the messaging header. If the origin in the messaging header cannot be used to trace the origin of the message, the logical assumption can be made that the routing information has been spoofed.

The second use of the stored list only works with large data sets, gathered from messages that originated from many different domains. The information is valuable even if gaps were detected in the messaging header. Using the information in the proven list, a network map is generated, showing the routing lines that are known to implement the messaging header with the digital forensic information. The routers on the network map can be called a trusted network, meaning that the information in the messaging header of messages, which travel through the trusted network, is more trustworthy.

By analysing the created trust network and by adding the information in the gap list, gaps that continually occur, can be identified. Depending on the size of the data set, paths that bypass the gap areas in the network can be created. Routers that form part of the trust network can be given a trust value, indicating that the routers are a preferred path for messaging.

Messages routed by trusted routers can be given a trust value based on the trust value assigned to the trusted router and adjusted as the message passes other routers in the network. The trust value determination and assignment is executed when the email is delivered to the final SMTP gateway. The trust value assigned to a message can be used to determine how much anti-spam resources must be assigned to the message, to detect if the message is spam. The creation and assignment of the trust values are outside the scope of this research. The

following section discusses using performance monitoring tools to collect log file data which might contain digital evidence.

## 6.4 Implementing log file sniffing using a performance monitoring tool

This section is based on work that was published [12, 96] and describes the new implementation that is used to collect data from log files for forensic readiness purposes. The prototype log file collection process was not originally created for email systems but the researcher assumed that if it works for one type of log file it should work for all types of log files. The log files are located on different servers and in various places on the servers.

The intent of collecting the data from the different log files is to find if the log files contain digital forensic evidence that can be used as part of a digital forensic investigation. The process described in the following sections focuses on the original development of the log file collection process that was developed to collect log file data from java web applications. The intent is to adapt the process for email systems.

The setup of a performance monitoring tool to collect digital forensic data is discussed in section 6.4.1. Section 6.4.2 discusses the method used to collect data from log files. Section 6.4.3 discusses an experiment that was performed with the digital forensic data collected, using a performance monitoring tool, to determine if the data could be used in a digital forensic investigation.

### 6.4.1 Setup of the Performance monitoring tool to collect digital forensic data

The implementation of the performance monitoring tool implementation focuses on a web system which is made up of a collection of web applications. All the web applications were developed using the Java programming language and use the same user session information to grant users access to services. The Java application server uses Apache Tomcat application containers to manage the Java applications. Users connect to the main web server to gain access to the access-controlled Java applications. The main web server creates connections to the Java application server using a reverse proxy server to hide the internal URLs of the Java web applications. Figure 6.6 depicts the elements that are included in the investigation experiment.

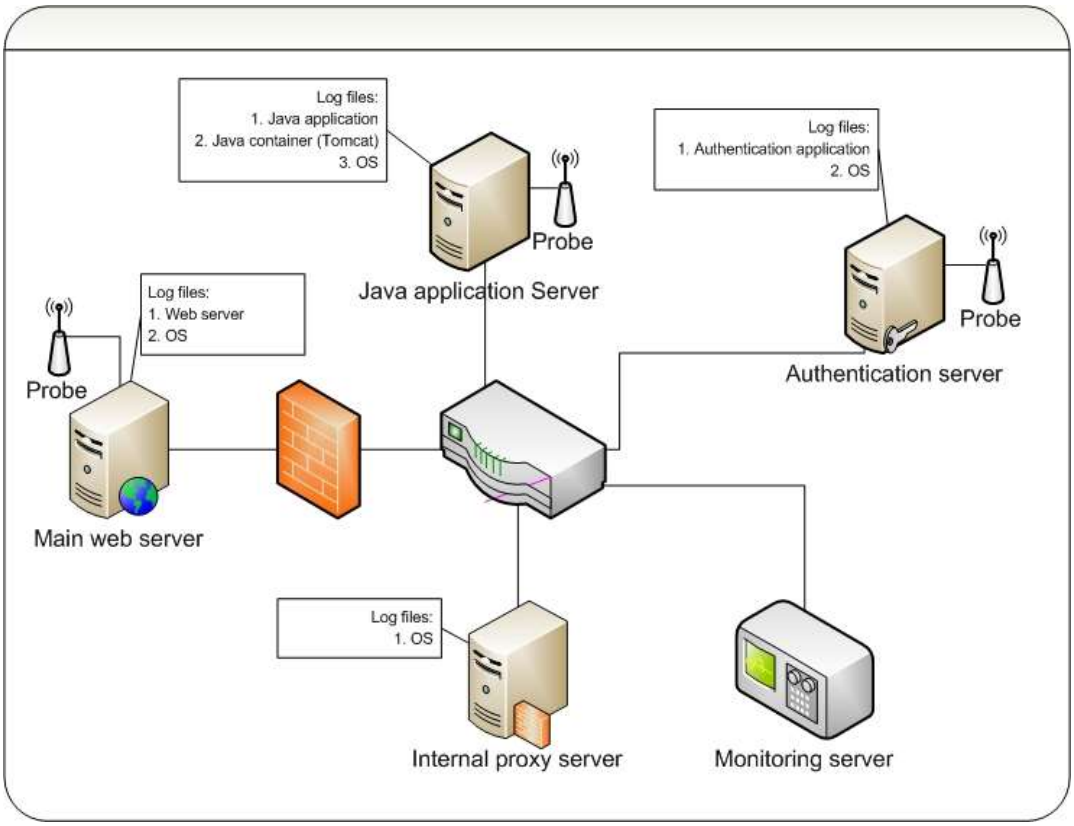


**Figure 6.6: Experiment representation**

The diagram in Figure 6.6 shows five servers: the main web server, the Java application server, the authentication server, the internal proxy server and the monitoring server. Authentication is implemented using an open LDAP system, as described in the background section. An application on the Java application server does the authentication lookup and session management when users log in to any of the Java applications.

Probes are installed on the main web server, the Java application server and the authentication server. A probe is simply a device that collects data for the monitoring server. The probes are set up to on the three servers to collect log file entries for user login activity data and user session data. More specifically, the user login activity data and user session data consist of the following: The web server log files contain data about user web resource requests. Log files on the Java application server contain data about user sessions and application access and authentication requests. The outcome of authentication requests is stored in the log files of the authentication server.

During a user-initiated login process, a connection is established from the main web server, through the firewall and internal proxy server, to the Java application server. The main login page is displayed in the user's browser. When the authentication application successfully authenticates the user on the open-LDAP, the authentication application generates a session object that is saved by the authentication application and sent to the user's browser.

System log files can contain data about system state, actions performed and errors encountered. Web applications that make use of containers like Apache Tomcat or web-logic have additional logs that are kept by the containers. Application log files and container log files are sometimes located in different locations on the server. System experts can help, during an investigation, to locate different log files and describe the information that is contained in the log file. Documenting where log files are located and what information they contain is helpful, but not always available. Sometimes system documentation does not exist or does not contain log file description information.

Therefore, if the system experts are not available and no documentation exists, the investigator will have to locate and scan the log files themselves. During any investigation much time can be spent looking for and scanning log files that might contain data needed for an investigation. The process of setting up probes to collect the log file information is a way of capturing the knowledge that system experts hold and saves time during the investigation's data collection phase. The log file data collected by the probes is used to investigate a specific user's authentication process. Section 6.4.2 discusses the process that is used by the probes to collect data from the different log files and store the data in the performance monitor's database.

### 6.4.2 Collecting data from the log files using custom probes

This section discusses the process that was used to extend the probes to collect data from different log files. Section 6.4.2.1 discusses the working of the probes. Section 6.4.2.2 discusses the collection of the data in the log files, using these probes.

#### 6.4.2.1 The probes

Different monitoring probes exist, to monitor different data collection points like system performance, database tables and log files. Probes normally do not perform any processing but connect to a data point and periodically reads the value of the data point. A data point can

be a system value, also called a standard data point or it could be a software generated data point, also called an extended data point. Standard data points like CPU, memory usage and drive space usage exist as part of the operating system. Extended data points are small applications that can be created to capture and expose data from log files and databases.

The probe captures the values and sends it to the listening agent. The listening agent is a software application that is installed on the monitored resource to allow for Secure Shell (SSH) communication between the performance monitoring tool and the monitored resource[12]. The communication connection is established from the listening agent to the monitoring server using a specific port and username and password authentication.

Communication between the listening agent and the performance monitoring tool is established periodically, according to the timed events, as set by the system administrator in the performance monitoring tool. If, for example, the timed event is set to five minutes, then the performance monitoring tool will communicate with the listening agents every five minutes.

Figure 6.6 depicts three probes that are already installed as part of a performance monitoring strategy. Each of the probes has been configured to provide specific information such as CPU, memory and disk space usage, about the server it is installed on. The probe on the main web server collects data from extended data points about the web server, like requests per second, response time per request and the average response time of each of the web pages.

The probe on the Java application server, in Figure 6.6, collects data from extended data points about the Java applications and the Apache Tomcat container. This data is used to profile the Java applications in order to do optimisations to the code and scale the Java environment. Data on the authentication process, like logins per minute and authentication system performance, is collected by the authentication server probe from an extended data point. The next section discusses the collection of the data contained in the different log files.

#### 6.4.2.2 Log file Probes

The probes, depicted in Figure 6.6, access the different log files using a log file sniffing application. Each log file has its own log file reading application. When the probe is polled for data, it actually executes the log file reading applications. The log file reading applications remember what the last line in the log file was, that was read. The log file

reading applications then read every line from the last read point to the end of the file. Each line is corroborated using the date-time stamp, log file name and server name of where the log file line was recorded.

The probes receive the two lines, as described in the previous example, in a single string that it sends to the performance monitoring agent. The listening agent returns the received string to the performance monitoring server. As stated in Section 4.4.2.2, the agent connects to the monitoring server using an SSH connection. This means that the information could not have been tampered with during transmission. The monitoring server stores the validated log file information in a read-only database table. A read-only database table only allows the data to be stored using SQL INSERT statement and the data to be read, using the SQL SELECT statement. The read-only table does not allow the data to be edited using an SQL UPDATE statement or deleted using an SQL DELETE statement. The following section discusses different uses for the collected log file data.

### 6.4.3 Using the data for a digital forensic investigation

This section discusses the data that was collected from the log files and the types of investigations supported by the data. It is important to note that, although the data was stored in a read-only table in the monitoring server's database, the performance monitoring tool could not be used to analyse the data. To allow the performance monitoring tool to understand the data, the data description must exist in the performance monitoring tool's data dictionary. Augmenting the performance monitoring tool's data dictionary was deemed outside the scope of the research because it would have meant rewriting a large part of the performance monitoring tool and the proposal for this research was to use the performance monitoring tool "as-is".

Section 6.4.3.1 discusses the data that was collected using the performance monitoring tool. Proof that the data could be used for an investigation was gained by using the collected data to investigate an experimental brute force attack. The outcome of the investigation is discussed in section 6.4.3.2.

#### 6.4.3.1 The data collected

Web server requests and responses are stored in the web server log file according to the design of the web server that was being monitored. The monitored log files were collected

from the main web server, depicted in Figure 6.6. Each line entry in the log file contains: Date-time stamp, requesting address, requested resource, HTTP request type (GET/POST) and request outcome.

The authentication application log file is used to store user authentication and session management data. Data in the authentication application log file contains: data-timestamp, log level, username, session ID and the authentication process outcome. Table 6.3 depicts two lines from the application log file.

**Table 6.3 Example data contained in the application log file**

| |
|---|
| 30-Jul-2011 11:00:00; INFO; s98278178; NULL; Authentication request received |
| 30-Jul-2011 11:00:01; INFO; s98278178; tQy1TRvGm53vbJFRyt11JmRnhCNvyYyq81Fy2Zy8vrn8CPTpt3pz; Authentication request successful |
| 30-Jul-2011 12:15:29; INFO; s12345678; NULL; Authentication request received |
| 30-Jul-2011 12:15:29; INFO; s12345678; NULL; Authentication request failed |

Every authentication attempt is stored in the log file. Session Ids, as shown in Table 6.3, are issued according to the username and date-time stamp when an authentication request is successful. The session Id is used to generate a session object that is referenced by all the web applications to verify a user's authentication status. If the user session expires, the session object is destroyed. When a session ID is not stored in the log file, it means that the request was made for a new authentication session, as shown in the first line of Table 6.3. When a session ID is stored, the authentication request succeeded, as depicted in the second line of Table 6.3. The last line in Table 6.3 shows the session ID as NULL and indicates that the authentication request failed. Authentication requests can also be a request to change a user's password.

Authentication requests are sent from the authentication application to the authentication server, depicted in Figure 6.6, and are logged in authentication server log file. The log file data contains: date-time stamp, username, session object ID and the outcome of the authentication requests.

Table 6.4 depicts the data contained in the authentication server log file. Session validation occurs when a user browses from one web application to another. If a session validation

request's outcome is negative it normally means that the user does not have permission to access the requested application. Line 2 of

Table 6.4 depicts a session validation request.

**Table 6.4 Example of data contained in the authentication server log file**

| |
|---|
| 30-Jul-2011 11:00:01; INFO; s98278178; authentication successful; tQy1TRvGm53vbJFRyt11JmRnhCNvyYyq81Fy2Zy8vrn8CPTpt3pz; |
| 30-Jul-2011 11:04:00; INFO; s98278178; validation successful; tQy1TRvGm53vbJFRyt11JmRnhCNvyYyq81Fy2Zy8vrn8CPTpt3pz; |
| 30-Jul-2011 12:15:29; INFO; s12345678; failed; |

The following section discusses the first investigation experiment that was conducted by using the live log file data.

#### 6.4.3.2 Investigating a brute force attack

The experiment is to detect evidence of a brute force attack and try to determine where the attack originated from. It is important to mention that the authentication application does not lock a user's account after a number of failed login attempts. After the log file probes were in place a mock brute force attack was launched against a test account. The attack was initiated from a computer inside the organization's network. The mock attacker was told to find a random area to attack from, without the knowledge of the investigator.

Using the log files from the authentication application and the authentication server, evidence of a possible brute force attack was found. A search for all the authentication requests, that contains the test account ID, was performed on the performance monitoring tool's database. The data retrieved from the database was split into two data sets. The first data set contained the data from the authentication application log file. The second data set contained the data from the authentication server log file. After analysing the first data set, it was found that from date-time stamp $t_0$ until date-time stamp $t_1$ there was a significantly large number of failed authentication requests.

A search for all the web server log file data between $t_0$ and $t_1$ was performed. According to the experiment scope the brute force attack originated from one position. A search was performed to find a requesting address that showed a large number of authentication resource

requests within the same order of magnitude as the number of authentication requests in the first data set. It was decided not to use a slow brute force attack due to time constraints that was placed on the experiment.

The brute force attack initiated by the mock attacker produced a very high number of authentication resource requests compared to the normal number of authentication resources requests, so only one requesting address was found during the data search. Confirmation from the mock attacker was used to prove that the workstation in question was the actual attacking source. The experiment was conducted four times; using different locations at different times, with the same outcome.

### 6.4.4 Collecting digital evidence from email system log files

Although it was proven that log file data could be collected using performance monitoring tools, it was found that the data collected from email system log files will not provide the needed digital evidence to support all email forensic investigations. After the prototype log file collection process was implemented and tested, it was extended to collect log file data from the different servers that make up the email system. Most of the data contained in the log files of the email system, had more to do with system performance and functions, than the activity of the email system to send and receive emails.

One log file on the SMTP gateway recorded the connection requests that were being made but all the connection requests originated from the anti-spam server. Another log file, discovered on the Post office servers, recorded the sender and receiver addresses of the different email and the timestamps of when the email was delivered.

A decision was made to try and find a more complete set of digital forensic evidence elsewhere in the email system. The following section discusses the initial process of collecting and storing IP header data.

## 6.5 Collecting IP header data

As discussed, it was found that using the data from TCP/IP packets, the SMTP header data could be validated so that the data could be trusted. The author defines SMTP header validation as the process of using corroborating information from other sources to confirm the data in an SMTP header.

Cyber criminals use spoofing to hide the true origin of an email. During an email forensic investigation, the possibility that the email was spoofed should be recognised. SMTP header validation is, therefore, a necessary addition to email forensics. Moore[97] states that a new process should be measured by the value it has over the existing process and the cost effectiveness of implementing the new process. Therefore, not only does the SMTP validation process need to add value to the email forensic process, it also has to be cost effective to implement.

The first step in designing an effective SMTP validation process is to find the correct data source to collect the extra data from. Using only data collected from an email does not always give the answers the investigator is looking for, especially if the information collected is spoofed. It is therefore proposed to use data from other sources, in addition to emails, to perform an email forensic investigation. Finding the correct data sources means that the data sources either already exist or the data sources can easily be created with the least amount of impact to current email system implementations.

The IP packet collection process, described in Chapter 6, used different collection points to determine if the collected IP packet data could be used in an email forensic investigation. The author decided to experiment further with IP packet data, in addition to looking at the SMTP headers, to validate SMTP header data. Each IP packet consists of an IP header and its IP packet payload. Such IP packet header and IP packet payload data may be potential sources of digital forensic information. IP packet data can easily be obtained using a network sniffer tool, as discussed in Chapter 6, and can possibly be used to validate SMTP header data without changing the implementation of the current email system.

Section 6.5.1 discusses the development that was conducted to improve the IP packet collection process. The experimentation scenario is discussed in section 6.5.2, to set the scope of the SMTP validation experiment. Section 6.5.3 discusses the lessons learnt during the implementation of the different IP packet collection points. A short overview of the data collected from the different collection points is given in section 6.5.4. Section 6.5.5 presents the experimental analysis of the collected IP packet data and the SMTP header data. A summary of the experimental results is given in section 6.5.6.

### 6.5.1 Creating the IP packet data collection process

IP header data is only available during the transportation of IP packets across a network. Once the packet is delivered to the indented destination, the IP header data is lost unless a record of the IP packet is kept. Network monitoring tools could be used to keep a record of IP packets, however, since there can be numerous IP packets that are sent across a network, network monitoring tools normally only collect IP packets when the data is needed for trouble shooting a network problem. The first task was to find a way to collect the IP header data, this task is discussed in section 6.5.1.1 Section 6.5.1.2 discusses the second task which is to determine where to collect the IP header data from and to ensure that the IP header data contained the data needed to validate the SMTP header.

#### 6.5.1.1 Data collection tool

While attempting to find a solution for how to collect the IP packet data, a study of network monitoring tools was conducted. The study of network monitoring tools pointed to the use of a simple application named tcpdump[98] that has the required functionality of recording TCP/IP packet data directly from the network interface of a network attached device. When a recording of the network traffic on a server's network interface card is needed, the tcpdump application is run on the server itself. The tcpdump application collects all the network traffic that is received by the network interface on the corresponding server. Tcpdump can be used to collect the IP header data according to user-set filters. Setting filters for tcpdump allowed the author to collect only the IP packet data that was related to SMTP traffic on the network. The following section discusses the selection of the different collection points.

#### 6.5.1.2 Data collection points

Focus moved to the next task: where the IP packet data should be collected from. IP header data only exists while the IP packets are being transported to the intended receiver. The data collected from the chosen collection point, therefore, had to contain the data needed to validate the SMTP header. After reviewing the network architecture of an email service, it was decided that the possible points to collect IP packet data from would be either at the firewall, the anti-spam server or the SMTP gateway.

The first potential collection point chosen was the firewall because it was assumed that all network traffic must pass through the firewall. IP packets used to deliver email to the email system will travel through the firewall and can be collected. Since the firewall redirects SMTP traffic to the anti-spam server only, the anti-spam server was chosen as a second

potential collection point. The IP headers collected from the anti-spam server should also contain the data needed to validate the SMTP header.

Some email systems do not use an anti-spam server. In the cases where an anti-spam server is not used, the email is routed from the firewall to the SMTP gateway. The SMTP gateway was included in the experimentation to test the ease with which the SMTP gateway collection point could be implemented. Since the experimental setup included an anti-spam server, it was thought that the SMTP gateway collection point would not produce the needed IP header data. The reasoning for the previous statement is that the anti-spam server scans the incoming email for spam before resending the SMTP packets to the SMTP gateway. When the SMTP packets are sent from the anti-spam server to the SMTP gateway, the IP header data will probably indicate that the origin of the email is the anti-spam server due to the anti-spam server resending the SMTP packets.

The experimentation was conducted on the live email system of our organization. An experimental email account was created to receive the experimental emails. The experimental email account was created to ensure that the experimentation process will not impact on the other users of the email service within the organization, since it is a live email service. Figure 6.7 provides a graphical overview of the email service setup.

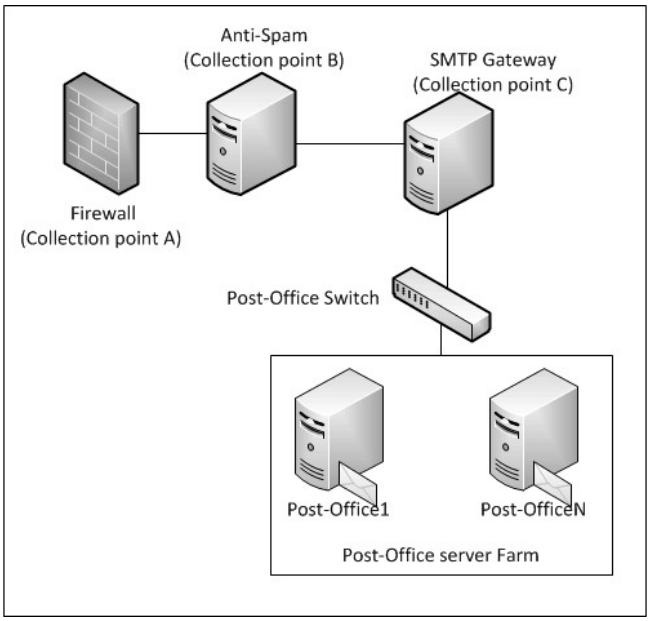


**Figure 6.7: Setup of Experimental Email System**

In Figure 6.7, firewall rules force all SMTP traffic intended for the SMTP gateway server to the anti-spam server. SMTP traffic that was not intended for the organizational SMTP gateway is blocked by the firewall. The anti-spam server filters the incoming SMTP data for spam, blocking the perceived spam and forwarding the rest of the SMTP data to the SMTP gateway. The experimental email account was hosted on one of the post-office servers in the post-office server farm. The following section discusses the collection of the IP header data from the three potential collection points previously mentioned.

### 6.5.2 Scenario for collecting IP header data

The spoofed email test data was generated to conform to a basic phishing attack scenario. It was decided to try and emulate a phishing attack scenario to truly test the capabilities of the proposed SMTP validation process. The scenario is described as follows; a suspicious email was received by a user, asking for personal information of the user to be updated by following a link in the email.

The email was purported to originate within the whitehouse.gov domain which made the user suspect foul play. The anti-spam server might also have marked the email as spam, by adding some kind of spam tag to the email, which might also have added to the user's suspicions of the email. The user reports the email to the investigative team and asks that the source of the email be verified.

During the creation of the scenario it was assumed that an email forensic investigation will only be conducted when an investigation task is given to the investigation team. The SMTP validation process is currently intended as an email forensics process that is only performed during an email forensic investigation. To support the scenario, the IP header data collection process is implemented as a DFR process and, therefore, it is not the intention to have this part automated in any way.

To undertake the scenario, test emails were sent using a telnet session. The telnet session was used to connect to the email service and was sending the test emails using a normal TCP/IP connection to simulate the connection between two SMTP gateways. Section 2.3 of the background explains how the telnet session was used. Using a telnet session to generate spoofed emails was seen as the easiest way to generate test data for this scenario. Although more sophisticated methods for spoofing exist, discussing different methods of spoofing is not within the scope of this paper.

The experimentation was done on a live email system and the tcpdump tool was set to collect data for all incoming SMTP traffic at the three collection points simultaneously. The telnet sessions were hosted on a computer that was connected to the organizational network. Due to the fact that a large email system receives a lot of data, and the fact that collection was done at three points in the system, the experiment was conducted over a short time so that the tcpdump files did not grow larger than 500MB. Large files are cumbersome to move to the analysis machine and experimental storage resources for capturing data were very limited. The next section discusses the implementation of each of the three collection points.

### 6.5.3 Implementing the collection points

In order to collect data from collection point A in Figure 6.7, i.e. at the firewall, tcpdump needs to be installed on the firewall. The setup of the tcpdump process on the firewall, however, proved to be problematic, because a proprietary firewall appliance is used in our organization, which did not have the tcpdump application installed. Installing tcpdump was not an option since it would mean that the proprietary firewall appliance software is changed. Such a change, however, is not allowed by the proprietary firewall appliance vendor as part of the vendor's terms of use of the firewall appliance. It was, therefore, decided to try and capture the data from collection point A in Figure 6.7, using a device called the OptiView® XG Network Analysis Tablet [99] which is simply referred to as a network sniffing device in

the rest of this dissertation. The network sniffer is a physical device which was plugged into an open port on the firewall device and set to record the network traffic after it was sent through the firewall to the internal network. The firewall is set up to allow all SMTP traffic to pass through and hence, no SMTP packets will be dropped by the firewall. Therefore, it does not matter whether the sniffer device is placed before or after the firewall.

The sniffer device is not actually intended as a monitoring device but is a troubleshooting tool used to solve network-related issues; therefore it has very limited storage resources. Due to the limited storage resources on the sniffer device, only a limited amount of data, as mentioned previously, could be sniffed at a time. The network sniffer, therefore, could only be used for a short time (about 90 minutes in our organization's network) before its storage resources would be depleted.

Another drawback of this sniffer device is that it needs to be unplugged so that the data can be downloaded to a separate storage device for the data to be analysed by any external analysis process; the data cannot be transferred and analysed while the sniffer is in the process of capturing traffic. The limitations of proprietary devices and limited storage on the network sniffing device, meant that the implementation of a collection point at the firewall would not be feasible in our organization’s scenario. Therefore, the firewall collection point could not be used as a DFR implementation for this experiment. Should these limitations be overcome by another organization’s different scenario (i.e. non-proprietary firewalls or less-limiting storage issues), the firewall could certainly be considered as a valid collection point.

The implementation of collection points B and C, in Figure 6.7, i.e. at the anti-spam server and SMTP gateway respectively was not a problem. Tcpdump could be installed successfully on the anti-spam server and on the SMTP gateway, since both of the servers are running open source software as the core operating system, which meant that alterations to these servers were permitted, without the limitations evident at collection point A, i.e. the firewall. Although the collection point implementations were run for a limited period of time, the permanent implementation of the two collection points would be feasible as part of a DFR process. Each collection point produced a tcpdump file that contained IP header data. The data contained in each tcpdump file is discussed in the next section.

### 6.5.4 Reviewing the data collected from the three collection points

A tcpdump file was collected from the firewall, anti-spam server and the SMTP gateway. The IP packet data dumps created by tcpdump at the three different collection points was analysed using WireShark, shown in Figure 6.9.

From the initial visual inspection it showed that each collection point did produce IP header data that could be used to validate SMTP header data. An example of data collected from one collection point can be seen in Figure 6.9. Line 448 shows that SMTP data was indeed captured as part of the IP header data collection process, which shows that the tcpdump application did collect IP header data that corresponds to the sending of email.

Investigation of the tcpdump files from collection point C, (the SMTP gateway) in Figure 6.7, showed that the originating address for the recorded IP packets came from the anti-spam server. This observation verified that, as suspected earlier, the IP packet data collected from the SMTP gateway would not contain the data needed for SMTP validation. This observation is only true if SMTP traffic is routed through an anti-spam server before it is sent to the SMTP gateway. Otherwise, if the latter observation did not hold true, data collected from the SMTP gateway would also have been a valid collection point.

In the case of our scenario, however, IP packet header data is lost, since the SMTP routing process routes all incoming SMTP traffic to the anti-spam server first. The data loss is not due to the routing action but rather the delivery action. The IP packet header data gets lost because when the IP packets are delivered to the intended receiver of the particular IP packets, the data contained in the IP packets are delivered to the intended recipient. The IP header data for all the delivered IP packets, according to the internet protocol (IP), are discarded as soon as the data is delivered at its final destination, because then they are not needed anymore. Since the anti-spam server is the first delivery point in the destination organization's email system, it would be the final destination for the initial SMTP delivery process. The data collected from the SMTP gateway, therefore, contained newly-manufactured IP header data as created by the anti-spam server. For this reason the IP header data collected from the SMTP gateway could not be used to validate the data in the SMTP headers.

A review of the tcpdump file from collection point B in Figure 6.7 showed that the IP routing data was captured. The tcpdump from collection point A also contained the IP routing data since the firewall was not the receiving host but merely a routing point to the intended host,

i.e. the anti-spam server. The review process also showed that the data from collection points A and B were virtually identical when the emails originate from outside the organizational network. Internal email was found not to pass through the firewall but was routed to the anti-spam server. The firewall did contain email data that was sent from the organizational SMTP gateway but was not found in the collected anti-spam server data. This is because outgoing email from the organization is not sent to the anti-spam server. The researcher concluded that the IP header data, needed to validate the SMTP header data in our organization's scenario, would probably exist in the tcpdump files collected from collection point A and B and that further investigation should focus on these points.

Further investigation showed that, with some minor system changes, collection point B would be the better choice as a collection point. The system change is to force all SMTP traffic to pass through the anti-spam server. The value to the organization would not only be the collection of potential digital forensic evidence but also to ensure that organizational resources are not being abused to send unsolicited electronic communication such as from compromised workstations. Another reason for choosing collection point B was the problems with the implementation of collection point A, i.e. the fact that the tcpdump application could not be implemented on collection point A directly. The analysis of the IP header and SMTP header data, using the data from collection point B, is discussed in the following section.

### 6.5.5 Analysing the collected header data

The analysis of the header data started with collecting the IP header data and the SMTP header data from the following sources; the tcpdump files, from collection point B in Figure 6.7, were collected and hashed to ensure the preservation of the files. SMTP header data of the suspect email was collected from the email account in question and saved to an email export file. The email export file was hashed to ensure the preservation of the file.

The SMTP header data in the email export file was analysed using visual analysis by opening the email export file in a text editor and viewing the email source information. Figure 6.8 shows the SMTP header of the exported email. The email source information contains the SMTP header data and the email body data, but Figure 6.8 only shows the SMTP header data, since that is the data that needs to be validated; the data of the email body is not relevant. The visual analysis was performed in order to try and find the true source of the originating email. The spoofed email purported to originate from the domain whitehouse.org, as can be seen on

line 1 of Figure 6.8. Using an IP lookup utility like nslookup on the originating IP address, in line 9, shows that the originating IP address is from the up.ac.za domain. The fact that the return path domain, on line 1, is purported to be from whitehouse.org and the purported originating domain, on line 2, is up.ac.za shows that the SMTP header data was spoofed.

```
1   Return-path: <the.test@whitehouse.org>
2   Received: from kendy.up.ac.za ([137.215.101.101])
3       by gwise.up.ac.za with ESMTP; Fri, 23 Aug 2013 14:14:42 +0200
4   Received: from op.up.ac.za ([137.215.209.16])
5       by kendy.up.ac.za with esmtp (Exim 4.63)
6       (envelope-from <the.test@whitehouse.org>)
7       id 1VCqGH-0001el-Vl
8       for p4160886@gwise.up.ac.za; Fri, 23 Aug 2013 14:14:41 +0200
9   Received: from [137.215.124.175] (helo=whitehouse.org)
10      by op.up.ac.za with smtp (Exim 4.63)
11      (envelope-from <the.test@whitehouse.org>)
12      id 1VCqGG-0004yB-7x
13      for ruan.vanstaden@up.ac.za; Fri, 23 Aug 2013 14:14:41 +0200
14  Message-ID: <E1VCqGG-0004yB-7x@op.up.ac.za>
15  X-Scan-Signature: 1e6b9c517c62f3e60a17c68d3359d0fc
```

**Figure 6.8: Extract of the SMTP header from the email export file**

The next step of the analysis process is to match the IP header data captured using tcpdump to the SMTP header data contained in the email export file. Matching the IP header data to the SMTP header data is achieved by using the SMTP message-ID. The message-ID should exist within the collected IP packet data, which also contains the SMTP data. The identification process was achieved by matching the message-ID from the SMTP header in Figure 6.8 line 14 to the message-ID in IP header data shown in line 468 of Figure 6.9.

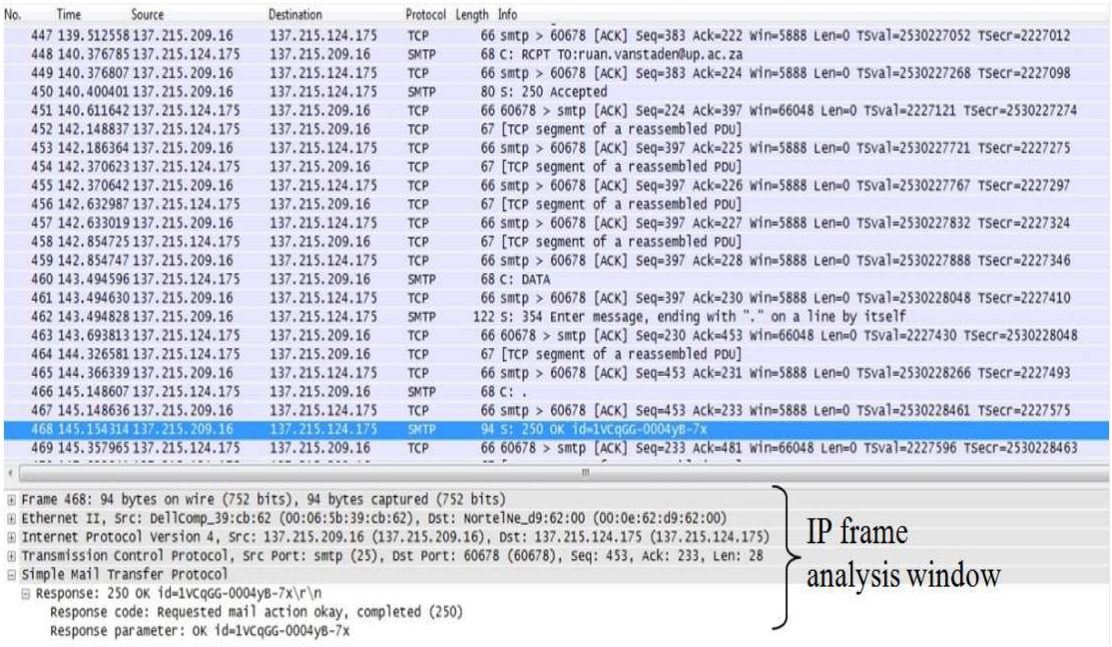


**Figure 6.9: Screen capture of the IP packet data as viewed in Wireshark**

After the IP packets, which were involved in the delivery of the suspect email, were identified, a comparison was made between the IP address in the SMTP header and the IP address in the IP header. The origin data in line 468 of Figure 6.9 matched the origin IP of the SMTP header in Figure 6.8 line 9, i.e. IP 137.215.124.175. It was found that the IP address was a match, which validated the IP address in the SMTP header as the source of the email.

The assumption that the IP address in the IP header is not spoofed is based on the process that is used to send and receive email. The sending and receiving SMTP gateways have a connection exchange at the start of the sending process. If the initial IP address in the IP header is spoofed this initial connection could not be completed. Therefore, the IP address recorded in the initial packet of the SMTP exchange must be the true origin of the IP packets.

The following section reviews the different collection points, to identify the best collection point to collect the IP header data from.

### 6.5.6 Defining the collection point

Collecting IP header data from several points; the firewall, the anti-spam server and the SMTP gateway, was not an efficient or an economic use of system resources. This statement is motivated as follows. The data collected from the SMTP gateway was shown not to contain

the IP header data needed to validate the SMTP header data for incoming email. IT did however contain IP header data needed to validate the SMTP header data for internal email and email being sent from the organization. The data collected from the firewall and the anti-spam server was virtually the same for external SMTP traffic and internal SMTP traffic was not recorded. The firewall traffic had SMTP traffic for email being sent by the organization that was recorded on the SMTP gateway but not the anti-spam server.

Each collection point only had a subset of the SMTP traffic data. The appropriateness of the different collection points can be measured by looking at three factors; the cost effectiveness of implementing the data collection process, the quality of the data collected and, finally, the efficiency of analysing the data collected from the particular collection point. Since all of the collection points had only a subset of the total SMTP traffic, the quality of the collected data was an issue for all collection points. Table 6.5 shows the overview of the different collection point measurements.

**Table 6.5: Summary comparison of collection points**

| Collection point | Implementation | Data collection | Analysis |
|---|---|---|---|
| Firewall | • Difficult<br>• Need special tools | • Incoming email traffic was in the recording<br>• Sent email traffic was in the recording<br>• Internal email traffic was not in the recording | • Depends on the analysis tool<br>• With the correct filtering, data was limited to needed data only |
| Anti-Spam Server | • Easy<br>• No special tools | • Incoming email traffic was in the recording<br>• Sent email traffic was not in the recording<br>• Internal email traffic was not in the recording | • Depends on the analysis tool<br>• With the correct filtering, data was limited to needed data only |
| SMTP Gateway | • Easy<br>• No special tools | • Incoming email traffic was in the recording but had the anti-spam server as originating IP address in the IP header<br>• Sent email traffic was in the recording<br>• Internal email traffic was not in the recording | • Depends on the analysis tool<br>• With the correct filtering, data was limited to needed data only |

The problem with the implementation of the firewall collection point, as mentioned in section 6.5.3, is the effectiveness of implementing the collection process since tcpdump could not be installed on the proprietary firewall device. A different method was needed to collect the IP header data from the firewall and the only option available in our scenario was the use of a network traffic recording device. The limitations of the network traffic recording device meant that the collection process could not run for a long period of time and, thus, could not be implemented as a proper DFR process.

The SMTP traffic data recorded on the SMTP gateway contained data for incoming email, internal email and outgoing email. The problem is that the incoming email contained the IP address of the anti-spam server as the originating IP address of the SMTP traffic but not the SMTP header. This meant that the IP header data could not be used to validate the originating SMTP header data.

From the comparison of the collection points, seen in Table 6.5, the author found that the anti-spam server is the most desired collection point. The IP address of the source of the SMTP traffic is still intact and can be used to validate the source in the SMTP header. There is still the problem of the anti-spam SMTP traffic recording which does not contain internal or outgoing SMTP traffic data. A change to the routing of SMTP traffic on the organizational network, that routed all SMTP traffic through the anti-spam server before it travels through the SMTP gateway, was proposed.

The selected collection point was best suited to the setup of our origination's email system but other organizations might find that the other collection points might be better suited to their email system setup. Collecting the IP packet data was shown to be usable as part of the SMTP validation process, as long as the IP packet data is collected from a suitable collection point. From the experiments performed, the author was able to construct an SMTP validation process model. The SMTP validation model is discussed in Chapter 7. The following section gives the conclusion of this chapter.

## 6.6 Discussion of the research

This chapter endeavoured to answer the question of which data sources could be used to trace the origin of spam. From the background on SMTP and IP it is known that both protocols can be edited and the information contained in the headers, spoofed. Due to the ability to edit the

data in the SMTP and IP headers, a method was needed to validate the data in the SMTP and IP headers.

A process that automated the collection of SMTP header data from mailboxes was discussed in section 6.2. The process is based on the manual process that is currently used by investigators to collect potential digital evidence from email. A database design was implemented to store the collected email data, for this purpose. The data tables were set as archive tables which meant that the data could not be tampered with once it was in the database. Using a database made it easier to search the collected data for potential digital evidence. While implementing this process it was discovered that not all email systems implement SMTP in the same way or by using the same number of SMTP headers. Therefore, it was decided to store each SMTP header as a separate record.

The initial attempts to add validation to the data collected from emails were discussed in section 6.3. Although the initial attempts to add validation data to the SMTP header failed, the researcher feels that it is important to note why the initial attempts failed. The main problem was that the ability to edit the SMTP headers necessitated the introduction of a data validation mechanism also meant that the hash value, that was introduced to perform the validation function, could not be safeguarded. The hash values that were added, to the SMTP headers, could be spoofed by an attacker in the same way that the SMTP header data could be spoofed. Another problem was that the implementation of the hash value in the SMTP header or the IP header would have added extra overhead to the different protocols that would have made them too expensive to implement. Using IPsec services, the hash values that were added to the IP headers could be safeguarded by using IPsec, therefore the message header was not needed. This meant that implementing a safeguard for the added IP header hash values showed that it was not needed if IPsec was implemented, since IPsec services already provided the needed data validation. The remaining problem was that not all organizations implemented IPsec for IP based services. The experience gathered from the experiments conducted on the SMTP and IP header showed that adding data to the SMTP header or the IP header might not be a viable option but recording SMTP traffic might be a viable option. This led to the development of the SMTP header validation process.

Before the development of the SMTP validation process an attempt was made to collect digital forensic evidence from email system log files using performance monitoring tools. Performance monitoring tools were employed, as mentioned in section 6.4, to create an

automated data collection process that could be used to collect log file information and store it in a central place. The log file collection process was first tested by collecting web access log file. The collected log files could be used to trace a user's activity on the web systems. This scenario showed that it was feasible to use performance monitoring tools to collect digital forensic data but email system log files did not contain the data needed to validate the data in the SMTP headers.

Section 6.5 discussed experimenting with collection IP header data that could be used to validate SMTP header data, specifically the source address contained in the SMTP header. Experimentation showed that IP header data is only available during the transportation of IP packets across a network. Once the packet is delivered to the intended destination the IP header data is lost unless a record of the IP packet is kept. The problem of collecting the IP header data was solved by employing an application called Tcpdump, as described in section 6.5. Tcpdump makes a recording of all the IP packets that pass through a specific network interface. Setting a filter for Tcpdump to filter out all IP traffic except for SMTP traffic reduced the amount of data that was collected. The experiment showed that the best place to collect IP header data was the anti-spam server. It was demonstrated that the data collected could be used to validate the data in the SMTP header and add extra data to the email forensic process. The following section gives the conclusions reached from this chapter.

## 6.7 Conclusion

Although augmenting the SMTP and IP headers by adding hash values to validate the data contained in them failed, it lead to automating the collection of email data for email forensic investigation. Using performance monitoring tools to collect log file data showed that digital forensic data could be collected from a number of log files in email systems and stored securely. Although using a performance monitoring tool to collect email log data did not have the outcome that was expected. These failures lead to the creation of an added SMTP validation process which uses IP header data and SMTP header data to validate the SMTP header data.

The data collected could be used to extend the email forensic process beyond just using SMTP data to conduct an investigation. It was also found that collecting the IP packets involved in the transfer of SMTP data meant that the SMTP header data was also collected. This meant that a SMTP header data collection process would not be needed but it would still

be an advantage to save the SMTP data from the collected IP packets in a database. The following chapter discusses the implementation of the eDFR architecture based on the planning that was discussed in Chapter 5 and the research conducted, as discussed in this chapter.

# Chapter 7 : Implementation of the eDFR architecture

## 7.1 Introduction

This chapter describes the implementation of the eDFR architecture by referring to the implementation process group of the ISO 27043 readiness process depicted by the highlighted area in Figure 7.1. Like with any newly proposed process, the success of the eDFR architecture is dependent on the adoption of the process. Moore[93] states that a new process should be measured by the value it has over the existing process and the cost effectiveness of implementing the new process. Ozment [100] stated that adoption of any security technology only occurs when the advantage of adoption outweighs the cost of adoption. Therefore not only does the eDFR architecture need to add value to the email forensic process it also has to be cost effective to implement in order to be successfully adopted.

The implementation process group is divided into four separate implementation steps. The first step is implementing the system architecture, which was discussed in Chapter 5. Implementing the collection, storage and handling of the potential digital evidence is the second step of the implementation process group. The third step is the implementation of the automated data analysis processes. Implementing the incident alert process is the final step of the implementation process group.

The implementation of the system architecture is discussed in Section 7.1 and focuses on the implementation of the DFRDSR server and the cloud service needed for the data analysis automation. The cloud service for data analysis is needed due to those large data sets that could be extracted from email systems. The research, in Chapter 6, showed that the best place to collect potential digital evidence from is by capturing the TCP/IP packets on the anti-spam server's network interface. Section 7.2 discusses the implementation of the collection process, the data storage and the way the data is handled, according to the results of the research reported in Chapter 6.

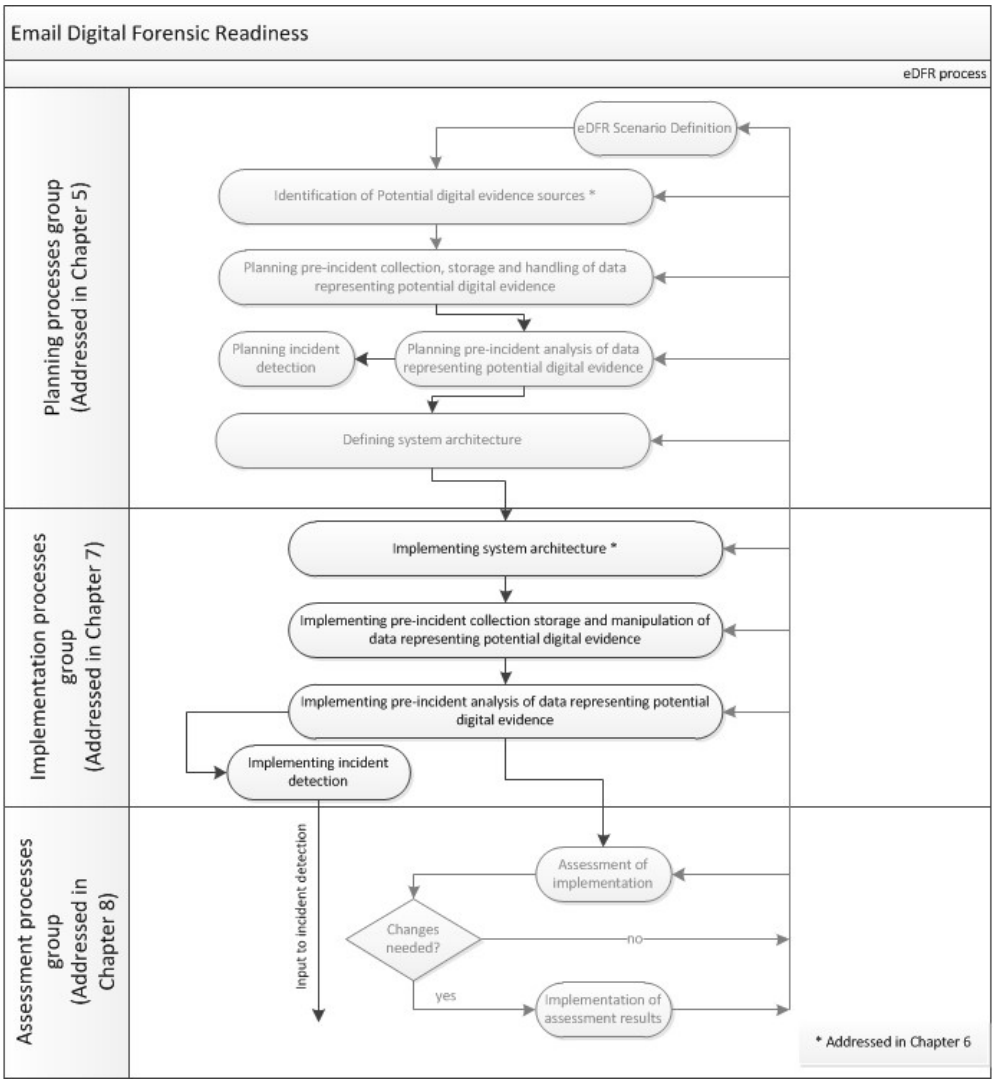


**Figure 7.1: Readiness process according to ISO 27043**

There are two data analysis processes defined for the eDFR architecture implementation. First is the reconstruction process which is used to reconstruct the collected TCP/IP packets into emails with SMTP headers, email bodies and MIME attachments. SMTP validation is the second analysis process that is defined for the eDFR architecture implementation and is used to detect if SMTP headers were spoofed. The implementation of the data analysis processes are discussed in section 7.3. The SMTP validation process is used to implement the incident detection integration that sends alerts when a spoofed email is detected. Section 7.4 discusses the creation and implementation of the incident detection integration.

A summary of the implementation process group is discussed in section 7.5 and the chapter conclusion is given in section 7.6

## 7.2 Implementing system architecture

As discussed in Chapter 5, the system architecture of the eDFR architecture needs a DFRDSR server. The DFRDSR server was implemented on a database server that is not part of the database server cluster and with limited access control so that only administrators have access to the server. The server firewall was set up to only allow communication between the anti-spam server, the Cloud processing server and the DFRDSR server. Investigators would be able to access the DFRDSR server using a SSH session. Communication between the anti-spam server and the DFRDSR server was implemented using a VPN with private key encryption. This meant that the communication line was reasonably secure. The same type of VPN connection was setup between the Cloud processing service and the DFRDSR server. The DFRDSR server was setup to allow the sending of email from the server to a specific group of email addresses so that reports and incident alerts could be sent.

The Cloud processing service was self-hosted and setup on a resource sharing system. The system is normally used for batch processing of financial data so that auditing of the processing is active. The Cloud processing service uses a resource schedule allocation to dynamically assign processing resources as it is needed, according to a prioritization scheme. During the evening the financial batch processing is given priority to use resources and during the day other batch processes are given priority. How the resources are managed is not within the scope of this dissertation. The following section discusses the implementation of the data collection process.

## 7.3 Implementing pre-incident collection storage and manipulation of data representing potential digital evidence

There are many ways of collecting and storing data with regard to email. The data can be collected directly from the mailbox or from the SMTP gateway or from the anti-spam server, as discussed in Chapter 6. The point of data collection and the method of data collection can have an impact on the type of data that can be collected. Different methods of data collection have also been proposed by different researchers. An example would be Aiello et al [101] which describes a process of collecting network data using a Pearl script and storing the captured data in .pcap files. The .pcap files are then parsed to a database.

Chapter 6 showed a similar process of capturing .pcap files using tcpdump. The data collection process has been changed from the proposed implementation, in Chapter 6, in that

the packet data is directly stored in a database and not captured in .pcap files. The capability to enable the parsing the .pcap files data into a database was kept in case it might also be needed. Collecting data from the network layer has the added advantage that the collected data not only contains SMTP data but also TCP and IP data. As has been shown, the IP and TCP data can be used to validate the SMTP header data.

The eDFR architecture is mainly a software process that can be broken into two applications. The first is the data collection application and the second is the analysis application discussed in Section 7.4. Figure 7.2 shows a flow diagram of the data collection process.

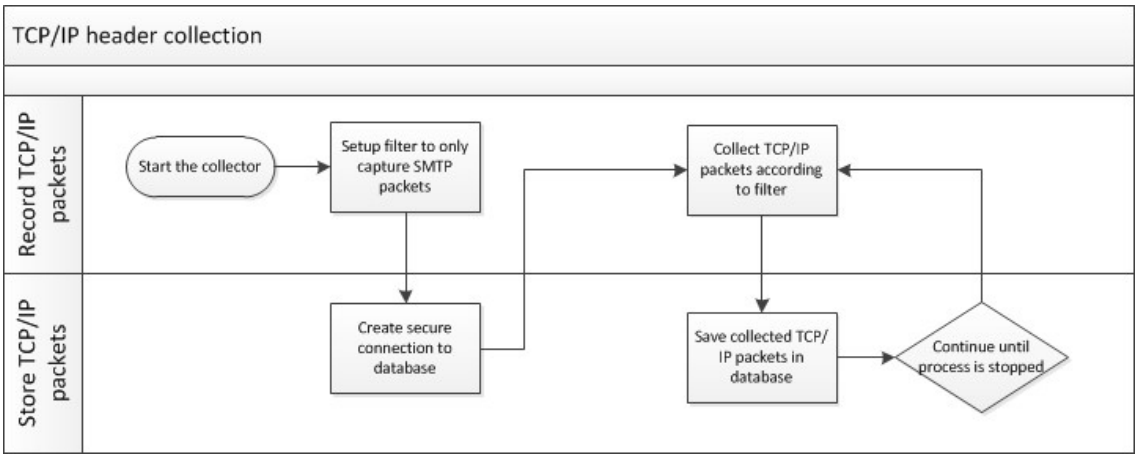


**Figure 7.2: Data collection process**

The data collection process need not have a user interface since it will only be used to collect the TCP/IP data from the anti-spam server network interface. The program starts by setting up a collection filter to only collect the SMTP packets that are sent to and from the anti-spam server. The filter will ensure that only SMTP data is collected but that all SMTP data is collected. The store process opens a secure connection to the database on the DFR data store and reporting server, shown in Figure 7.2. Once the connection is made, the collection process starts and TCP/IP packets involved with SMTP transmissions are collected. The collected packet data is sent to the database on the DFRDSR server and saved. The data collection process continues until it is stopped by an external action.

Figure 7.3 depicts the database design that is used to store the packet data recorded from the network layer. The database design is based on the pcap file format, the IP datagram, the TCP datagram and the previous database design for the SMTP data capture process. Pcap files are

used by packet recorders like WireShark to store captured packet data. For this reason the decision was made to add the ability to store .pcap file data in the database.

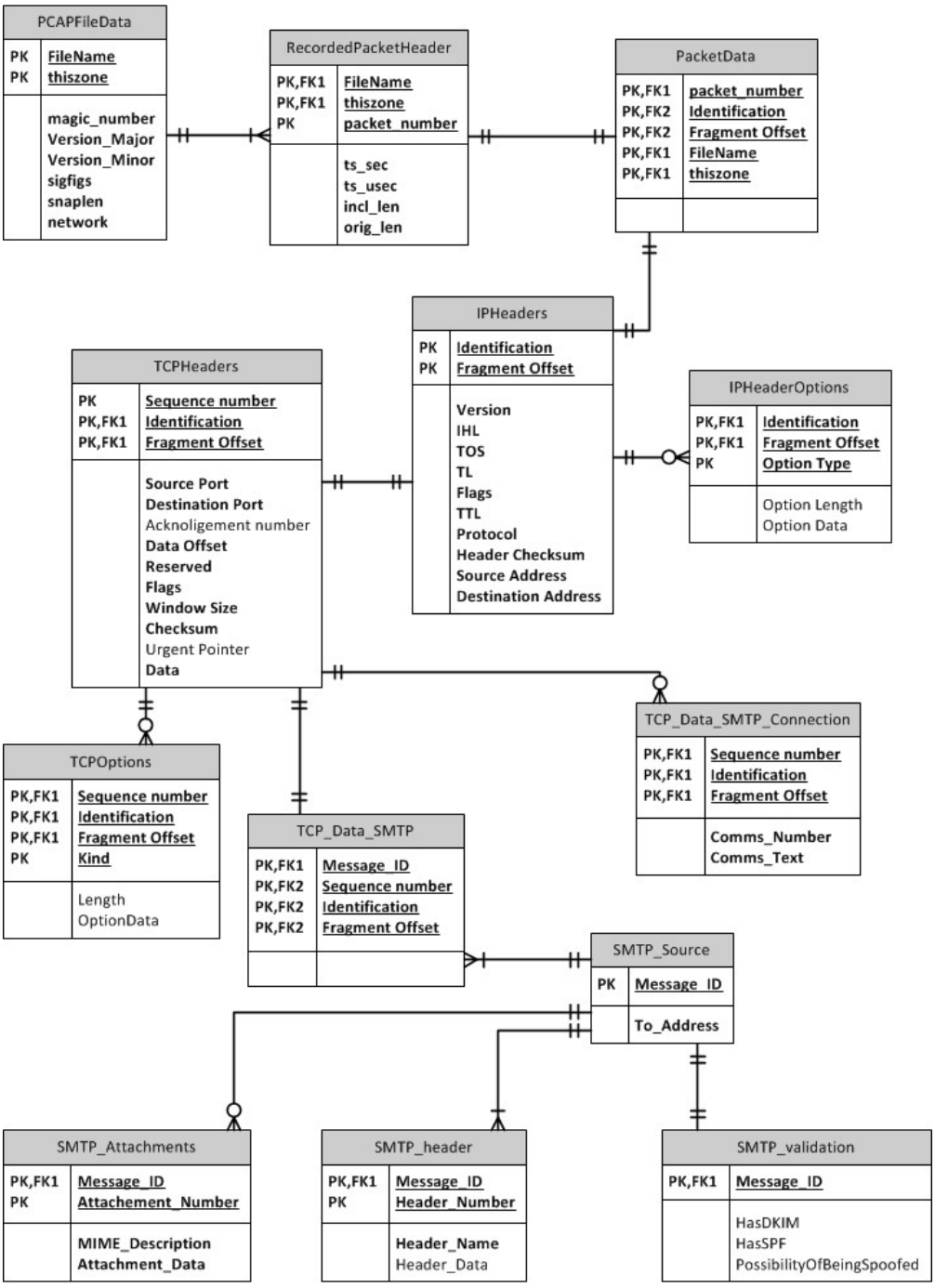


**Figure 7.3: Database design**

The layout of a pcap file is given in Figure 7.4. The pcap file is structured into three distinct sections namely the global header, the recorded packet header and the data header. The global

header records information about the packet recording session like the version of the pcap file, given by the major and minor version, the local time offset of the location that the packet recording was performed at and the sample length that was specified for the recording. All this information is recorded in the database table PCAPFileData depicted in Figure 7.3

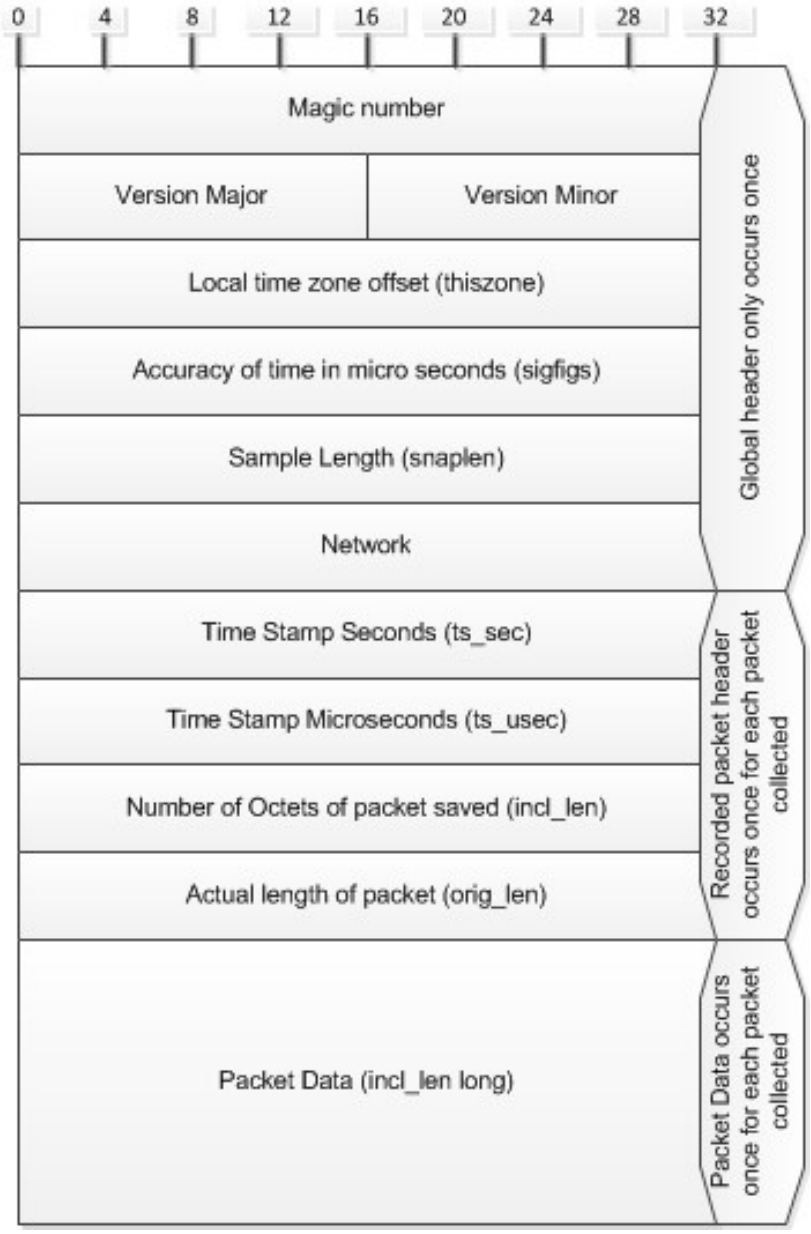


**Figure 7.4: PCAP file format**

Each packet that is captured in the pcap file also has a recorded packet header that records extra information, as depicted in Figure 7.4, about the packet, like the actual size of the

packet and the recorded size of the packet. The recorded packet header is a separate table in the packet recording database, called RecordPacketHeader, depicted in Figure 7.3.

Each packet that is recorded is saved as an IP datagram shown in Figure 7.5. The IP datagram is broken into a header, options and payload or data area. The header contains information about the IP packet, for example the protocol contained in the packet the time to live (TTL).

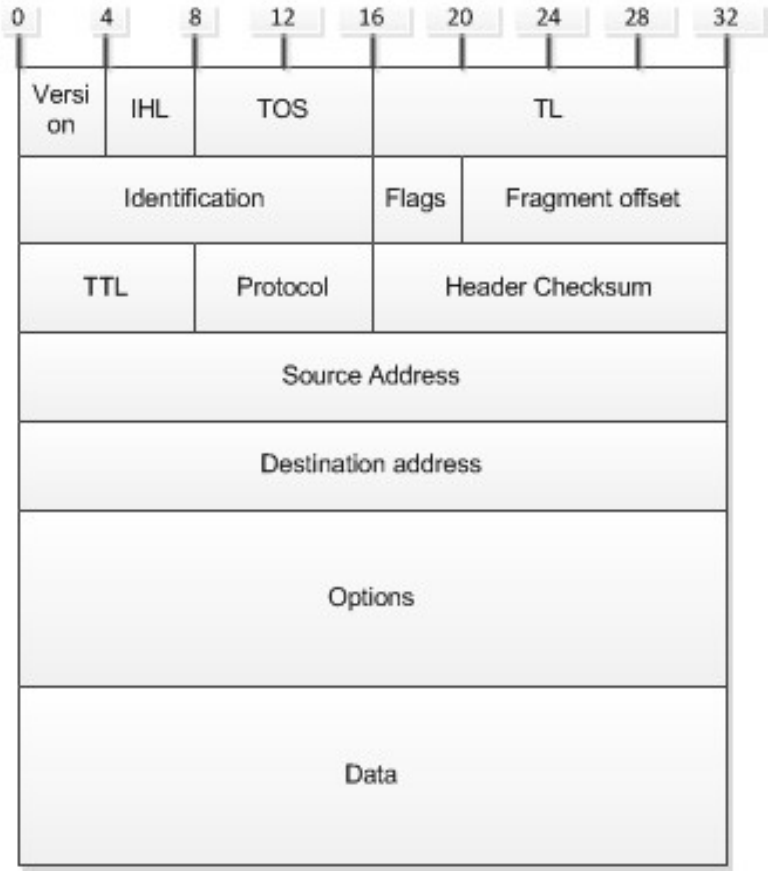


**Figure 7.5: Depiction of an IP Datagram**

The database shown in Figure 7.3 has a table called IPHeaders that store all the IP header data for a captured IP packet. It only stores the header data since that is the only part of the IP packet that has a standard length. The options section and the data section of the IP packet have variable lengths depending on the options that are recorded and the data that is contained in the IP packet. Table IPHeaderOptions, from Figure 7.3, stores the different options in the IP packet, each in its own database record. The data in the IP packet is a further packet of a higher level protocol like TCP or UDP.

The packet capture tool was designed to capture SMTP traffic so that the database only captures packets that use TCP as the IP protocol option. Table TCPHeaders, from Figure 7.3, is used to store the TCP packet headers. Figure 7.6 shows the TCP datagram design. A TCP packet can contain a variable number of options but each option has a set structure. It was decided to save the TCP options in its own DB table, depicted in Figure 7.3, called TCPOptions.

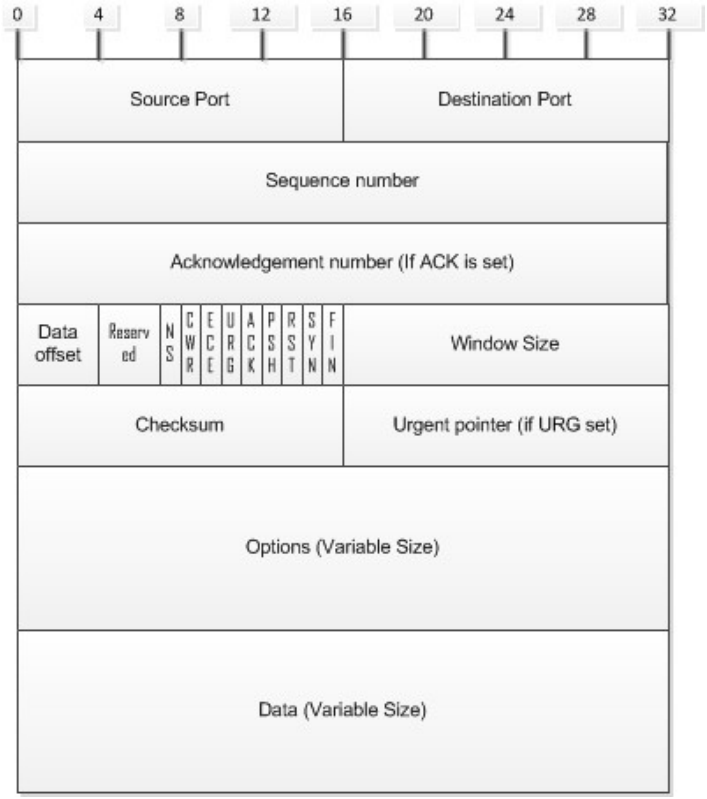


**Figure 7.6: TCP Datagram**

The SMTP database tables that were designed to hold data about email and were discussed in Chapter 6 were added to the main database design. The tables are not used during the packet capture process since restructuring TCP data while running a packet capture would slow down the capture process and impact on the email system. The SMTP tables are used during the data reconstruction process. The database design can be used to store data from pcap files or to programmatically capture and store the packet data directly to the database. The reconstruction process can then be used to generate the SMTP data contained in the TCP/IP packets and stored in the DB as well.

When the reconstruction process was first run it was noted that not only does the email data get captured but also the SMTP server connection process communication data between the SMTP server and the SMTP client. The TCP/IP packets that involved communication between the SMTP server and the SMTP client are also saved in the DFRDSR server database. These SMTP communication packets are kept, since they could also be used as part of an email forensic investigation and should also be reconstructed and saved in the database. The TCP_Data_SMTP_Connection table is used to store the communication data between the SMTP server and the SMTP client. The Comms_number and Comms_type fields in the TCP_Data_SMTP_Connection table are used to store the communication data as it originates from the SMTP server and the SMTP client.

Although the database is an archive database it has been decided to include a hash value for each of the TCP/IP packet data records to ensure that the record did not change. The data tables will also be encrypted so that unauthorized access of the data file will not expose the data. The design process added a filter to the collector that will allow the collection of only TCP/IP packets involved with the transfer of SMTP data. Normally the SMTP client server exchange would be logged on the SMTP server in a log file. Collecting the TCP/IP packets directly, the SMTP client server exchange is also captured and there is no need to collect digital evidence from the SMTP server log files.

The SMTP packet data can be reconstructed to produce the full email, containing the SMTP headers, email body and MIME extensions describing the email attachments. This fully reconstructed email can then be used for email forensic analysis. The data contained in the TCP and IP packets can be used to validate the data in the SMTP header data and build a routing graph for the sent email. Saving the data in an archive database, as discussed in Chapter 6, ensures that the collected data can't be changed or removed. This is a requirement to ensure the data is suitable to be used in a digital forensic investigation. The following section discusses the implementation of the analysis processes.

## 7.4 Implementing pre-incident analysis of data representing potential digital evidence

Section 7.3 gave a description of the data storage process that can be used to store and reconstruct the TCP/IP packet data for later analysis. Reconstruction and analysis of the TCP/IP packet data is not a difficult process but the amount of data that is generated during the recording process makes reconstruction and analysis a time consuming process. The

amount of data that needs to be reconstructed and analysed necessitated the use of big-data processing tools. Hadoop is a technology developed by the Apache group to allow for the analysis of big-data using distributed resources. Hadoop is discussed in Chapter 4.5 but the main point to remember is that Hadoop is a framework that allows for the scheduled distributed processing of data. The Hadoop application was developed by referencing a Github[102] project that gives an example of a Hadoop application that can be used to reconstruct .pcap files.

The data collected during the DFR process can be scheduled for processing at a later time when processing resources are available or the processing assigned can be increased or reduced depending on the needs of other systems. The first step is to reconstruct the SMTP data, which is discussed in Section 7.4.1. Section 7.4.2 discusses the use of Hadoop to run email forensic analysis on the reconstructed data.

### 7.4.1 Email reconstruction process

The process of TCP/IP reconstruction is implemented in programs like WireShark and there are also libraries that can be used with most programming languages like Java, C++ and Pearl. The process of reconstructing TCP/IP packets is therefore well known. During the network transportation process the email is broken into packet size blocks that will fit into the data area of the TCP packet. These TCP packets containing the different data blocks are then transported using IP packets which encapsulate the TCP packets[29]. When the packets are delivered to the receiving network interface the packets are reconstructed and the TCP/IP packet data is sent to the application layer the data is intended for. The normal reconstruction process makes use of buffers in the network interface to store the TCP/IP packets while the reconstruction process is taking place but after the reconstruction is complete the TCP/IP packets are removed from the buffer.

The removal of the reconstructed TCP/IP packets is a problem since the data, which could be used as part of a digital forensic investigation, is lost. It is for this reason that the DFR process captures the TCP/IP packets from the network interface before the network interface reconstructs the TCP/IP packets to keep the TCP/IP packet intact. Storage of the captured packets could be accomplished by either storing the packets in a pcap file or storing the TCP/IP packets in a database. Both of these storage options will ensure that the TCP/IP packet data is not lost. Since the raw packet data is captured and stored, the packet

reconstruction process needs to be implemented for the captured TCP/IP packets to obtain the contained SMTP data.

Experiments conducted to reconstruct the TCP data into email data showed that the reconstruction process can take a long time. Reconstructing one email might be a relatively fast process but reconstructing multiple emails can take a long time. Using threading to reconstruct multiple emails at a time also sped up the process but not to the point where a full day's email could be reconstructed in a fast and efficient way.

It should be noted that the time taken to reconstruct email directly from a pcap file was found to be dependent on the size of the pcap file and not the size of the emails since the entire pcap file must be read into memory to find all the TCP packets that form part of the email. Placing the data into a database made the amount of time needed for the reconstruction process dependent on the size of the email and not the amount of TCP/IP packets that were captured. This is due to the fact that the data in the database is structured in such a way that TCP/IP packets that together make up an email can all be extracted from the database without scanning through all the data in the database.

The data reconstruction process is the same as what would happen on a normal network interface as described by [103] with the difference being that the TCP/IP data is saved in a database before the reconstruction process starts and that the reconstructed email is saved in the database so that the email data can be traced back to the TCP/IP packets that carried the data. Reconstructing the email data from the database is not done live but is scheduled so that it is run as a bulk process once or twice a day when other system resources are available to assist with the processing. Figure 7.7 shows the process of reconstructing the emails from the TCP/IP packet data.

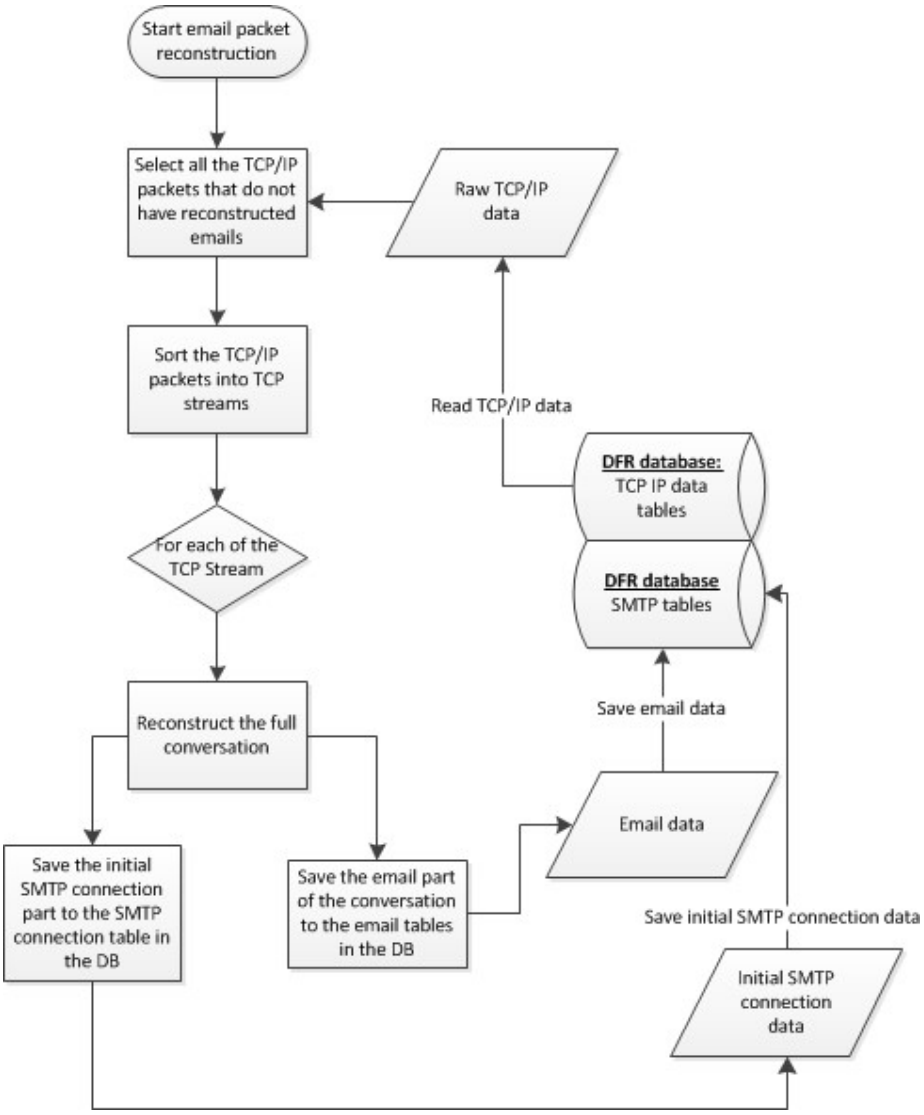


**Figure 7.7: Packet reconstruction process**

The reconstruction process starts off by extracting all the TCP/IP data that have not been used to reconstruct emails before and are sorted into TCP Streams. A TCP Stream is a specific packet flow between a source address and a destination address that makes up the SMTP delivery process for delivering and email. Each TCP Stream should contain the full SMTP exchange between the sending and receiving SMTP server. This exchange includes the actual email that was delivered and the server connection process. Refer to Chapter 6 Figure 6.9 depicting a TCP stream.

Once all the TCP Streams have been identified the email reconstruction happens for each of the TCP streams. The email is then saved in the database, in the data tables associated with

email data. The reconstruction process, as well as the analysis process was implemented to run on cloud processing services using Hadoop.

The new addition is to use Big-data tools like Hadoop to perform the reconstruction process for large data sets. Programming libraries like the one provided by RIPE-NCC[102] can be used to implement TCP/IP reconstruction from pcap files. This library also has an extension to query pcap files with SQL like queries. Changes were made to read the TCP/IP data from a database instead of a pcap file but the pcap file reader was left intact.

The Hadoop framework is used to perform the SMTP reconstruction process on multiple TCP/IP packet sets contained in the database. The number of processing objects is determined by the amount of processing data and the amount of processing resources available. Some TCP/IP packet payloads are encrypted using SSL certificates. The application protocol such as SMTP is still known since only the TCP/IP packet payload is encrypted and not the TCP/IP packet headers. Decrypting encrypted SMTP is the same as decrypting HTTPS if the SSL certificate is known. Part of the TCP/IP packets contains the certificate used to encrypt the SMTP contents of the TCP/IP packets. Part of the analysis process must be to decrypt the SMTP data after it is reconstructed from the TCP/IP packets. The following section discusses using the Hadoop framework to perform analysis on the SMTP data.

### 7.4.2 SMTP validation process model

SMTP validation is a process of scanning a suspect email and validating the data in the SMTP header using a source of data that corroborates the data in the SMTP header. Chapter 6 discussed experimenting with the collection of SMTP and IP header data. Lessons learned from the experimentation, discussed in Chapter 6, were used to develop a two stage SMTP validation model.

The flow diagram in Figure 7.8 depicts a graphical overview of the processes involved during the SMTP validation process. The validation process uses header data gathered from SMTP headers as well as IP headers.

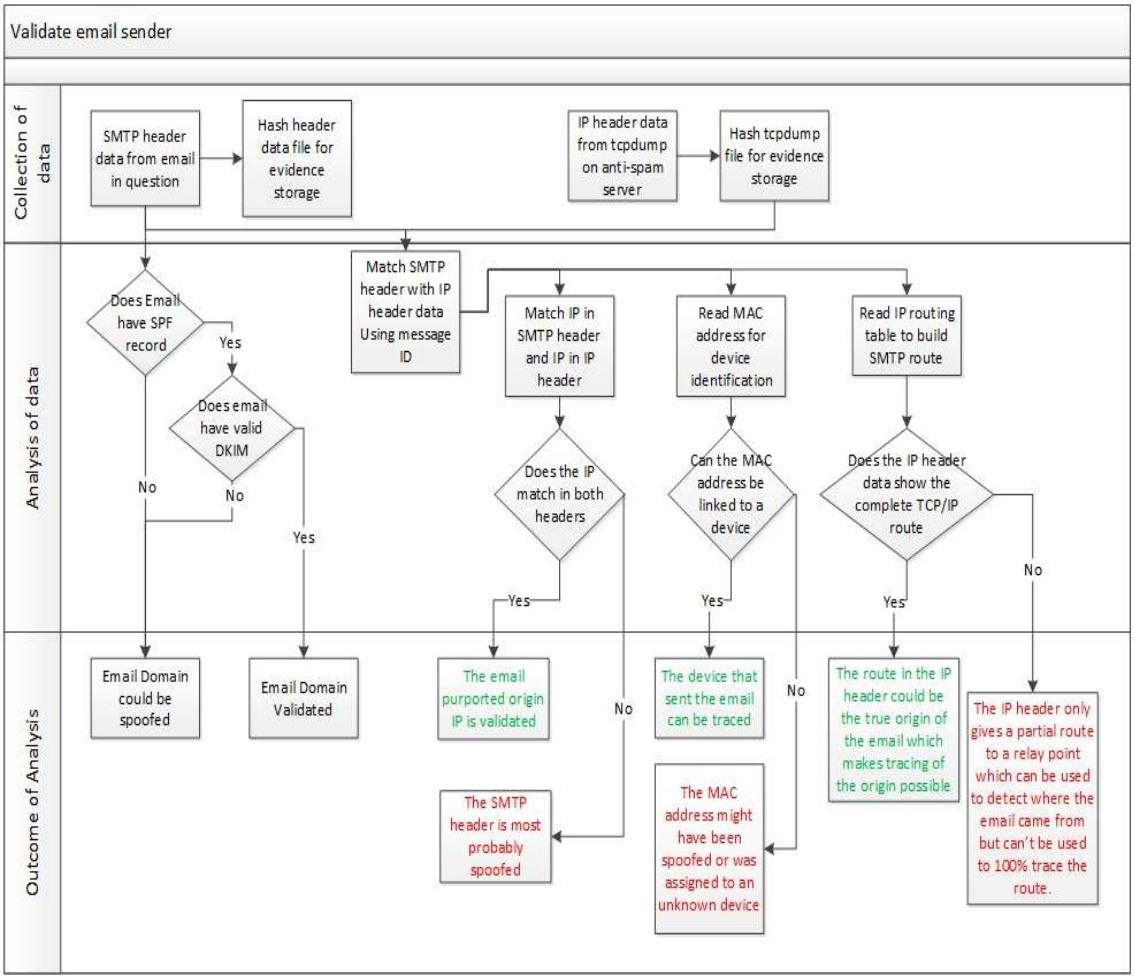


**Figure 7.8: Email validation process**

SMTP header data can be loaded from different email systems. Loading the data from different email systems might mean that the data needs to be reformatted into a common data format. The conversion process is not described in this dissertation since processes to achieve this conversion, already exist [104]. After the data is converted to the common data format, the header information is stripped from the content. The previous section discussed the collection of the IP header data. The next step is to analyse the SMTP header data and the IP header data that was collected.

The analysis starts by determining if the email header contains a Sender Policy Framework (SPF) record. The SPF record is issued by the sending SMTP server, to validate that the sending SMTP server is allowed to send email for the email domain[105]. If the SPF record exists and the validation check passed, the chance that the email sending domain has been spoofed can be reduced.

The second point of analysis is to determine if the email contains a Domain Key Identified Mail (DKIM). The DKIM is used by the sending server to sign the email to prove that the email originated from the sending domain. The DKIM record in the header will also contain the key signature that can be verified by sending a DKIM request to the sending domain DNS server. A record of the DKIM and the validation call is recorded in the SMTP header. To ensure that the DKIM was not spoofed the DKIM record can be used to do a second validation to ensure the DKIM is true by calling the purported sending domain's DNS server and comparing the DKIM that is returned with the DKIM in the SMTP header. The presence of the SPF and the DKIM only validates the sending domain of the email and not the original sender of the email. The sending SMTP gateway might have a record of the original sender of the email, which could be used to validate the original sender.

Validation of the email origin is done using the data from the SMTP header and combining it with the data from the IP header. Since the IP header data and the SMTP header data is in the database and linked to each other, matching the source IP address in the IP header and the source IP address in the SMTP header means running an SQL query in the database.

The first validation that was done was validating the purported IP address in the SMTP header with the IP address in the IP headers. A positive validation outcome would mean that the IP address in the SMTP header is the true origin address of the email. A negative validation could mean that the true origin address was changed to hide the true origin of the email. Another explanation could be that IP address in the IP header could be from the SMTP gateway where the IP packets originate, which will not be the same as the purported IP address in the SMTP header, but the domain address should still be the same.

Implementing the SMTP validation process using Hadoop is basically running the same SMTP validation process on all the SMTP data in the database and storing the result. The SMTP_Validation Table, shown in Figure 7.8, was used to store the outcome of the SMTP validation process for each email connecting it to the original email data. During the analysis process the actual email data is never changed.

More analysis processes could be implemented in the same way using the Hadoop framework. It should be noted that Hadoop does not affect the validity of the original data used in the analysis process since the original data is never altered after it is saved in the database. The following section discusses the implementation of incident detection.

## 7.5 Implementing incident detection

Incident detection was not originally part of this study. From the research that was conducted in Chapter 6, where a performance monitoring tool was used to collect log file data, the determination was made that the performance monitoring tool could be used to implement incident detection. Connecting the DFR process to a performance monitoring tool to create an incident alert was not such a difficult process. Most performance monitoring tools have integration capabilities that allow them to receive information from sources external to the performance monitoring tool. Using the performance monitoring tool, described in Chapter 6, an integration point was set up that will alert system owners when an alert is received from the DFR process. The alert was set up using an alert switch which is activated when an incident is detected.

The incident alert process was set up in such a way that any type of alert can be created and sent from the DFR process. The creation of an incident alert starts by defining alert criteria that is used to detect incidents as the data is placed in the database. The alert criteria can be based on a search string e.g. a specific email address or a specific set of values in the SMTP header. The alert criteria is stored in the database and is evaluated against the reconstructed email data being saved in the database. When an alert is triggered the trigger point with identifying data is stored in the database.

The alert can also be linked to a corresponding alert created in the performance monitoring system. This means that any new alerts defined for the DFR can be configured on the performance monitoring tool. The SMTP validation process was used as an experiment and the DFR process was set up to report when a spoofed email was detected. At this point it was possible to create the integration so that the severity of the alert could also be defined. This meant that if a possibility of spoofing was detected the performance monitoring tool would not send an alert but note the spoofing occurrence. An alert for a highly probable spoofing event will send an alert to system owners. The alert process on the performance monitoring tool was adjusted to also alert the intended receiver of the email so that they could be warned.

Although notifying users of possible spoofed email should be the work of the anti-spam server, it was felt that implanting the alerting process showed that it would be possible to alert users of possible spoofed email using the eDFR architecture. The incident detection could be implemented to alert on other incidents.

## 7.6 Summary of research

The research described in this chapter focussed on implementing a DFR process for email forensic investigations. The DFR process implementation was broken into three stages. The first stage was collecting the Digital forensic data from the email system. This was accomplished by collecting the TCP/IP packets directly from the anti-spam server interface.

The second stage was reconstructing the SMTP data from the TCP/IP packets. This stage was accomplished using an open source big-data tool called Hadoop. The reason for using Hadoop was that the amount of data collected from the anti-spam server network interface could become too large to reconstruct the SMTP data from the TCP/IP packets, in a reasonable amount of time, using a conventional reconstruction process. Using Hadoop is digital forensically sound because the original TCP/IP data is not lost or changed in any way.

The third stage was to implement the SMTP validation process using Hadoop. Implementing the SMTP validation process as part of the DFR process meant that an investigator can immediately see if the captured SMTP data was spoofed or not. The following section gives the conclusions of this chapter.

## 7.7 Conclusion

This chapter proposed a process of collecting, storing and analysing email data as part of a DFR process. The answer to the question: can the collected data be used in a digital forensic investigation is yes. The point and method of data collection is accepted in the network forensics process. The method of storage of the collected data ensures that the data cannot be tampered with once it is stored. The chain of evidence stays intact since the same process that collects the data is also the process that stores the data. An unintended but added benefit is that the data stored by the proposed eDFR architecture can also be used as a backup for the email system since all the data is captured and stored in a tamper proof way. The eDFR storage can be periodically written to long term backups like tape backup and cleared to keep the eDFR database manageably small.

Any analysis process that is executed on the stored data can read the data but not change it. The results of the analysis are stored either in another table of the database or in another form of storage. Any other form of storage of analysis data is outside the scope of the research and cannot be commented on. SMTP validation data is stored in a database table for later

reference and can also not be changed once the validation process is complete. This might be a problem later if the validation process can be better refined.

The SMTP validation is not the only analysis process that can be implemented. Processes to detect phishing attacks or even to detect spam can also be created. It is also important to note that, with enough resources, the reconstruction process could be done in real time if need be. Reconstruction of the TCP/IP data in real time will allow for the creation of a real time incident detection process.

The following section discusses the assessment process group.

**Part V: Assessment and Conclusion**

**Part V** of the dissertation contains Chapter 8 and Chapter 9. **Chapter 8** critically **assesses** the design of the **eDFR architecture** and proposes revisions that could be made to the eDFR architecture. **Chapter 9** gives the final **conclusion** of the dissertation and reviews the research that was conducted. The outcome of the research is discussed by analysing the research based on the problem statement that was made in Chapter 1.

# Chapter 8 : Assessment of the eDFR architectures

The assessment of the eDFR is intended to find improvements to the eDFR implementation and include them in the eDFR design process. This chapter reviews the research that was conducted for the dissertation and analyses the eDFR implementation to determine if the aims of the eDFR were met and if more refinement of the eDFR design might be needed. Figure 8.1 Shows the Readiness design process as discussed in ISO 27043. The highlighted part of the figure denotes the assessment process group that is discussed in this chapter.

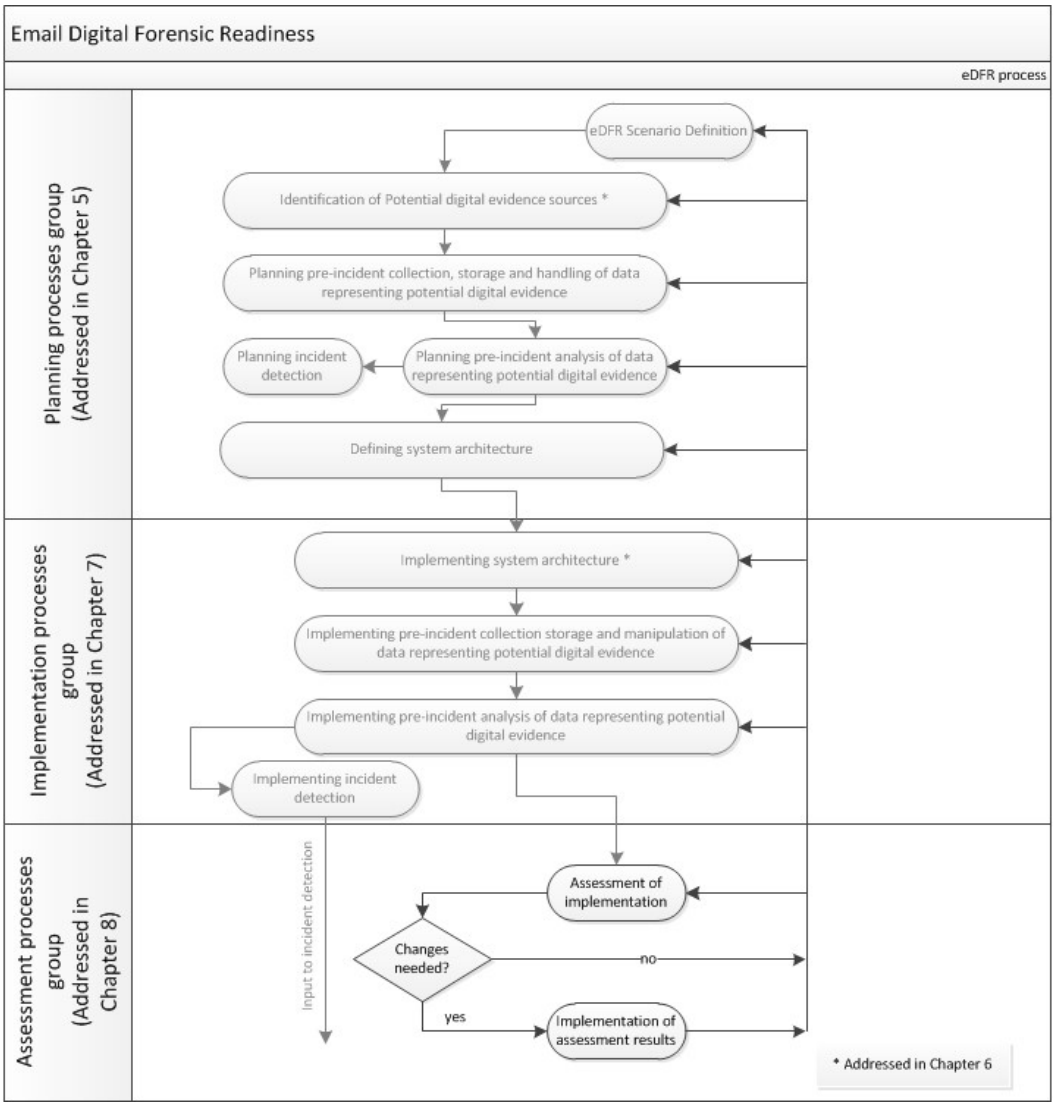


**Figure 8.1 Readiness design process according to ISO 27043**

Assessment of the eDFR architecture implementation is done in Section 8.1. Section 8.2 discusses the proposed changes to the eDFR architecture that were identified during the implementation assessment. Section 8.3 discusses the research that was conducted to implement the eDFR architecture and how the implementation of the eDFR architecture provides support for collecting email forensic evidence that can be used for email forensic investigations.

## 8.1 Assessment of implementation

Planning of the eDFR architecture defined the scenario that the eDFR architecture must support. The scenario created for the eDFR architecture was to collect possible digital evidence that could be used to investigate all the different email forensic investigation scenarios. The aims of the scenario were:

- Collect data that will:
    - support authorship investigations
    - determine the flow of information
    - detect cybercrime activity (for example phishing attacks)
    - trace the origin of email
    - detect spoofed email

Chapter 6 discussed the research that was conducted to collect possible digital evidence from email systems. Initially it was decided that the best course of action would be to add some sort of validation to the SMTP header. The proposed addition, discussed in Chapter 6, was to add a hash value to the received SMTP header. This addition of the hash value did not provide the desired results since the hash value that was added could not be safeguarded from tampering. During the research conducted into the SMTP header, specifications were found that were intended to allow SMTP servers to detect possible spoofing. SPF, DKIM and PSA were discussed in Chapter 2. These specifications were used in Chapter 7 to design the SMTP validation process.

Another attempt was made to add validation for email in the form of a hash value added to the messaging header of IP. It was found that, although it would be possible to safeguard the hash value using IPsec the hash value would not be needed is IPsec was implemented. Implementing IPsec on IP for SMTP meant that the IP packets containing SMTP data could be validated in using transportation security incorporated using IPsec. One problem is that

IPsec is not enforceable since the IPsec specification states that the implementation of IPsec should not interfere with transmission of IP packets that do not have IPsec transportation security implemented. This is the pattern for most security specifications proposed to date. SPF and DKIM implementations are also not enforced. There is, however, an assumption that can be made with regard to security specifications which is that, if an organization implements transportation security like IPsec or anti-spoofing like SPF or DKIM then the organization could be trusted not to send spam. This assumption is based on the idea that if an organization implements SPF and DKIM then the organization wants to use email for legitimate communication reasons and not to send spoofed email or for their email domain to be used in spoofing. This assumption was not tested but was incorporated in the design of the eDFR, specifically the SMTP validation process.

A way was needed to try and ensure that possible digital evidence contained in log files could be collected and stored. Experiments were conducted using performance monitoring tools to collect log file data from different servers. The experiments showed that it is possible to use performance monitoring tools to collect log file data. Investigating the log file data collected from email systems, it was found that the log files mainly contained data about the performance and activity of the email system components. This data did not support any of the aims of the eDFR scenario. It was decided that the implementation of log file collection for email systems, using performance monitoring tools, would be performed if another source for the data in the log files could not be found. The data could be useful for system administrators who maintain the email system but it might not be useful for email forensic investigators.

The lessons learnt while trying to implement the hash value for IP showed that collecting TCP/IP packets might provide the possible digital evidence needed to support the aims of the eDFR. A process was implemented using tcpdump and WireShark to test this assumption. The experimentation with collecting TCP/IP packets lead to the discovery that the full SMTP exchange between the SMTP client and the SMTP server could be collected in this way. This meant that the activity for creating the initial SMTP connection and the actual email could be captured. The creation of the SMTP connection was also found during the collection of log file data which meant that collection of the TCP/IP packets would negate the need for the collection of log file data.

The added benefit of the TCP/IP packet collection process was that the email SMTP header, email body and the email MIME attachments were collected. This meant that the data needed to support the aims of the eDFR could be obtained by collecting the TCP/IP packet data. After experimenting with tcpdump and WireShark it was decided to develop a different process for collecting the TCP/IP packet data instead of using tcpdump and WireShark. The main motivation behind creating the new collection process was that, although tcpdump provided the needed TCP/IP packet data and WireShark reconstructed the TCP/IP packets, neither of the two tools could help to automate the analysis of the collected email data. The implementation of the new collection process was discussed in Chapter 7. Chapter 6 also looked at the best place to collect TCP/IP packet data from. It was found that, for the experimental setup, the best place to collect the TCP/IP packet data is the anti-spam service.

Chapter 7 discussed implementing eDFR by collecting TCP/IP packets directly into a database. The packets were then reconstructed into email data that was also saved in the database. The reconstructed email data contained the email SMTP headers, the email body text and any email MIME attachments. Placing the data in a SQL database meant that the data analysis process could be implemented by creating a set of queries. It was, however, found that the amount of email data collected could become too large to reasonably reconstruct using conventional processing capabilities.

It was decided to investigate the use of Big-Data tools to help create a reconstruction and analysis process for the eDFR architecture due to the large amount of data that the eDFR architecture could potentially collect. This was achieved by using Apache Hadoop to implement the reconstruction and analysis processing. Using Hadoop meant that the processing needs could be adjusted according to the resources t needed versus the amount of resources that were available. New analysis processes could be developed on the same Hadoop platform to allow for the same resource management of the reconstruction process. The SQL database also made the process of selecting and processing data more efficient since the data that needed to be read into memory before processing could begin did not reside in pcap files.

Only two analysis processes were discussed during the implementation namely the reconstruction process and the SMTP validation process. The reconstruction process was developed to reconstruct the TCP/IP packets into email data and store the reconstructed email data in the database for later analysis. The SMTP validation process was designed to detect

spoofed SMTP headers. Detecting spoofed SMTP headers is needed to be able to determine if the SMTP header data can be used to trace the origin of an email. Placing the possible digital evidence in a structured database means that investigators can create data analysis processes that will extract the data that they might need.

Focussing on the aim to detect spoofed email, the eDFR architecture was able to detect spoofed SMTP header data to a degree. The problem is that, if email header data is spoofed, the true origin of the email cannot truly be traced. Spoofing is mostly used by spammers to hide the true origin of spam. Tracing the origin of a single spam email might not be possible but using the data collected with the eDFR architecture it would be possible to create a network map that plots the spoofed emails according to the source addresses contained in the IP headers. The map would show if a correlation existed between the source addresses that point to where the spam might really be coming from.

An analysis process was developed to use possible spoofed and validated SMTP header data to create SMTP network diagrams to determine if a correlation could be seen for the possible spoofed email. The network diagram was created to show if SMTP gateway servers could be connected to specific organizational domains using the SPF or DKIM records in the SMTP header. The network diagram also placed SMTP gateway servers that could not be associated with any organizational domain. SMTP headers that contain resend forward data was also used, to try and see if this could show email being sent between organizations and maybe help detect the flow of email communication. The network map that was generated was reduced to a limited number of SMTP gateways for clarity. The following graph shows one of the network graphs that was generated.

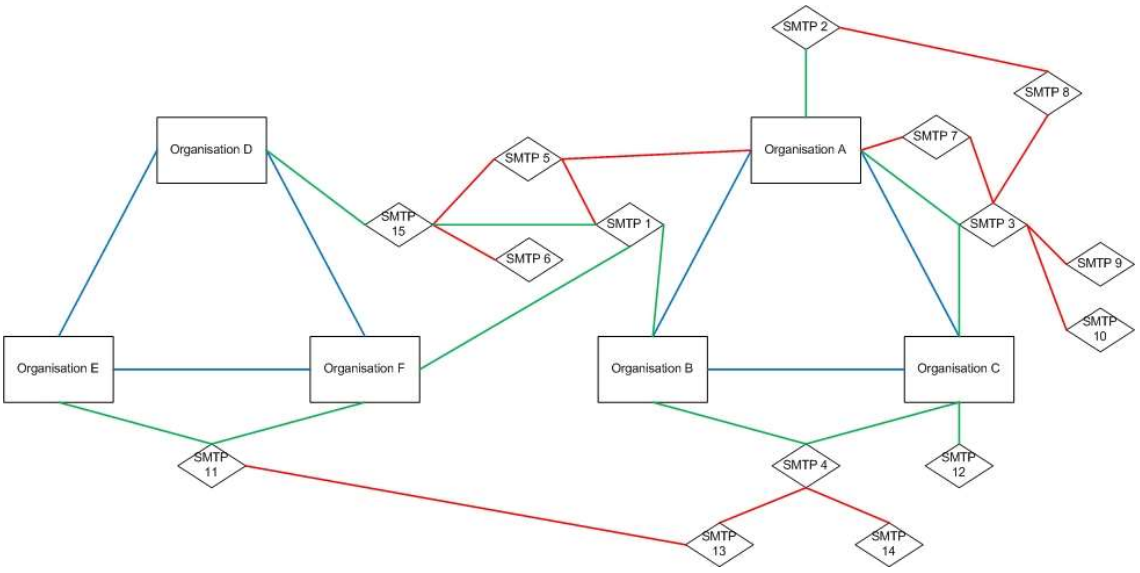


**Figure 8.2 Network graph generated from SMTP header data**

The network graph consists of organization blocks that the name of the organisations were removed from and replaced with place holders. This was done for privacy concerns. Organization blocks represent SMTP gateway servers that were validated as belonging to a specific organization. The diamond shapes represent SMTP gateway servers that could not be mapped to an organizational domain. Blue lines showed communication between validated organizational SMTP gateway servers. The green lines depict communication between SMTP gateways that could still be traced. The red lines shows communication between SMTP gateway servers where the network path was approximated, since only the sending and receiving SMTP gateways were known, but the IP routing data was not complete.

The SMTP gateway servers might be the same server with different names since the sending SMTP gateway servers could not be validated. This is not a complete solution but the researcher feels that it shows the capability of analysing the captured packet data. The following section discusses the implementation of the assessment results.

## 8.2 Implementation of assessment results

The assessment showed that the eDFR architecture supports the dissertation but can still be extended with other analysis processes to support other email forensic investigations. These other email forensic investigations, as mentioned in the scenario, are out if the scope of this dissertation. The following section discusses the outcome of the research that was conducted for the dissertation.

## 8.3 Review of the research

This dissertation addressed a number of topics to implement a DFR process that could be used to collect digital forensic evidence for email forensic investigations. The research was presented in the form of a Digital Forensic Readiness process design called the eDFR architecture. A number of experiments were conducted mainly to find potential digital evidence in email systems. During the pursuit of the research it was determined that the main problem to overcome was to detect, and possibly overcome, spoofing of the SMTP headers. Most of the research focussed on finding digital evidence to detect spoofing. Although mechanisms exist for SMTP, for example SPF and DKIM, which can be implemented to achieve the goal of overcoming spoofing, the mechanisms are not enforced as a standard. This can be seen in the wording of the standards that state that the implantation of SPF and DKIM should not hinder the main function of SMTP when the mechanisms are not implemented. Neither do SPF and DKIM ensure that potential digital evidence is gathered in a forensically sound way.

During the research process a number of lessons were learnt that were used to implement the eDFR architecture. Implementing the eDFR architecture does have benefits to email forensics. One of the benefits is that the eDFR architecture automatically captures digital forensic evidence and stores it in a forensically sound way. This means that there is a very low possibility that email forensic data could be lost. The automation of the data collection also means that the eDFR architecture does not have an impact on the function of the email system since no downtime is needed to collect forensic data. The collected data can easily be analysed to find possible digital evidence that is needed for an email forensic investigation.

Storing all the TCP/IP packet data and the SMTP connection data also means that data, which would normally be discarded during the transportation of TCP/IP packets or during the establishment of SMTP connections, are also saved. This data could potentially be used for other types of analysis processes that might not form part of email forensic analysis. The eDFR has the added benefit of being extendable, specifically by adding new analysis processes to the eDFR architecture to perform pre-analysis of the forensic data. This extendibility could also be used to create analysis procedures for non-forensic purposes.

An interesting observation that was made was that collection TCP/IP packet data seems to be the best way to collect digital forensic data for services that rely on a network connection to

perform the service's intended function, such as email. This would be an interesting exploration to see if this observation holds for other TCP services such as File Transfer Protocol (FTP) or Hyper Text Transfer Protocol (HTTP). The following section gives an overview of the contributions that the research made to email forensics.

## 8.4 Contributions made to email forensics

The different contributions that the research made to the field of email forensics if given in Table 8.1.

**Table 8.1 Contribution to email forensics**

| Contribution to email forensics | Discussion of contribution |
|---|---|
| Create a Digital Forensic Readiness architecture to collect digital forensic evidence from email systems. | Chapter 5, Chapter 7 and Chapter 8 |
| Provide a process to collect and store digital forensic evidence from email systems | Chapter 6 Section 6.5 |
| Provide a process to validate data collected from email systems to ensure the collected data can be used as part of an email forensic investigation. | Chapter 7 Section 7.4.2 |
| Provide an example of what can be done with the collected digital forensic evidence to aid in email forensic investigations. | Chapter 8 Section 8.1 |

The contributions that are given in Table 8.1 are discussed in depth in the sections of the dissertation given in the table. The following chapter gives the conclusion of the dissertation.

# Chapter 9 : Conclusion

## 9.1 Outcome of the research

Huge amounts of data are generated by email systems every day. As soon as a digital forensic investigation is required, digital forensic evidence is manually collected from the email system. The research focussed on creating an automated process to assist digital forensic investigators to sift through the huge amounts of data that are created by email systems. Automating the collection of email data ensures that all possible digital forensic evidence is collected from email systems and organised in such a way as to facilitate email forensic investigations. The analysis of the collected email data can also be automated to reduce the time needed to perform an email forensic investigation. Another major problem, addressed by the proposed eDFR architecture, is the securing and validation of the SMTP headers. The following questions needed to be answered in order to address the problems regarding the creation of the eDFR architecture:

**Q1:** Is there a way to automate the digital forensic data collection process for email?

Yes, by creating a software application to collect the TCP/IP packets from the anti-spam server network interface and store the data in a SQL database the digital forensic data collection process for email can be automated. A software application was created to reconstruct the TCP/IP data and store the reconstructed data with the TCP/IP data in the same SQL database.

**Q2:** What digital forensic data could be collected from email?

Collecting TCP/IP packets from the anti-spam server network interface allowed for the collection of IP packet data and SMTP data, including any media and attachments contained in the SMTP envelope.

**Q3:** Can the digital forensic data be validated and verified?

Yes, a process was developed as part of the eDFR architecture that validates the SMTP header data using IP header data. During the data collection the origin of the data was verified by storing the full TCP/IP packet data in the database.

**Q4:** Can the collected digital forensic data be used successfully for an email forensic investigation?

Yes, as long as the SMTP header data is validated. In the case where the SMTP header data cannot be validated, the SMTP header data cannot be used for the proposed use case of tracing the origin of email since, in such a case, invalidated SMTP header data might be tampered with and will not conform to digital forensic principles.

The collected digital forensic evidence could possibly be used for other email forensic investigations such as email authorship investigations, determining the flow of information and detect cybercrime activity (for example phishing attacks), but this was not fully explored. The collected digital forensic evidence was used to develop an analysis process to detect spoofed email.

The spoofed email detection process called the SMTP validation process was created when it was determined that spoofed email data cannot be used as part of some email forensic investigations, since spoofed email data could not be validated. The proposed eDFR architecture does, however, add to the field of email forensics by showing how data could be collected without causing downtime of email systems. Another benefit of the eDFR architecture is that email is captured before it is blocked by the anti-spam server and before the email could be deleted by a user. The eDFR architecture also negates the need for an SMTP archive server since the eDFR architecture includes an archiving process. The following section gives the future work that could possibly still be performed.

## 9.2 Future Work

Future work should look at ways to trace the origin of spam, using the digital evidence collected from email systems, to build a map of possible spam locations. An attempt was made during the research and initial results looked promising but the research was not completed due to time constraints.

It should be worthwhile to develop other analysis processes that incorporate research from areas such as email authorship analysis and information flow analysis to expand the eDFR architecture. Since the eDFR architecture is based in ISO 27043 it would be advantageous to consolidate the research conducted in email forensics into one process. An interesting research area would be to see if the collection of TCP/IP packets is indeed the best way to

collect digital forensic data for other TCP services for example FTP and HTTP and if that could also be incorporated into a DFR process.

The researcher would like to see the eDFR architecture design discussed in this dissertation to be expanded and improved to become the standard email Digital Forensic Readiness process. The following section gives the final conclusion of this dissertation.

## 9.3 Final Conclusion

Although the researcher could not find a way past spoofing, the proposed eDFR architecture was able to collect email digital evidence and use it to find spoofed email. Using TCP/IP packet capturing as part of the proposed DFR process allowed for the collection of email digital evidence that can be used as part of other email forensic investigations. Using big-data analysis tools to create automated analysis processes also helps to reduce the time an investigator has to spend sifting through the collected digital evidence, to find what is needed.

It is the opinion of the researcher that if IPsec, SPF and DKIM are made mandatory then spoofing of SMTP headers would not be possible anymore. The researcher hopes that ISP's, Network operators and standard writers will see this opinion and make work of it.

## 9.4 Acknowledgement

This work is based on research supported by the National Research Foundation of South Africa (NRF) as part of a SA/Germany Research cooperation program. Any opinion, findings and conclusions or recommendations expressed in this material are those of the author(s) and therefore the NRF does not accept any liability in regard thereto.

I would like to thank Professor Venter for his support and guidance during this process. I would also like to thank my wife for her support. I would also like to thank the University of Pretoria for the opportunity.

# Chapter 10 References


1. Peter, I., *History of Email*, in *www.nethistory.info/History of the Internet/email.html*. 2004, NetHistory Project.

2. *History of Email, How email Works, Email Server, Email*, in *www.vicomsoft.com/learning-center/history-of-email/*. 2011, Vicomsoft.

3. Vacca, J.R. and K. Rudolph, *System Forensics, Investogation and Response*. 2010: Jones & Bartlett Learning.

4. Oxford, *AskOxford.com*, in *AskOxford.com*. 2010, Oxford University Press.

5. Palmer, G.L., *Road Map for Digital Forensic Research*, in *DFRWS Technical Report*, G.L. Palmer, Editor. 2001, Report from the first Digital Forensic Research Workshop (DFRWS): New York.

6. Kohn, M., J.H.P. Eloff, and M.S. Olivier. *UML Modelling of Digital Forensic Process Models (DFPMs)*. [Document] 2009.

7. Rowlingson, R., *A Ten Step Process for Forensic Readiness.*, in *International Journal of Digital Evidence*. 2004.

8. Tan, J. *Forensic readiness*. [Document] 2001; Available from: https://isis.poly.edu/kulesh/forensics/forensic_readiness.pdf.

9. Wiles, J. and A. Reyes, *The Best Damn Cybercrime and Digital Forensic Book Period*. 2008, Safari online: Syngress.

10. Wood, D., *Programming Internet Email*, in *Programming Internet Email*. 1999, O'Reilly Media, Inc. p. 5-7.

11. Joyce, J., et al., *Monitoring Distributed Systems.* 1987. **5**(2).

12. Van Staden, F.R. and H.S. Venter, *Implementing Forensic Readiness Using Performance Monitoring Tools*, in *Advances in Digital Forensics VIII*. 2012, Springer: Berlin Heidelberg. p. 261-270.

13. Lueg, C., J. Huang, and M.B. Twindale. *Mystery Meat: Where does spam come from and why does it matter?* Security in the mobile and networked world [Document] 2006 April 29 [cited 15.

14. O'Brien, Cormac, and C. Vogel. *Spam Filters: Bayesvs. Chi-squared; Letters vs Words*. [Document] 2003.

15. Network-Dictionary, *http://www.networkdictionary.com/security/b.php*, in *http://www.networkdictionary.com*. 2009, http://www.networkdictionary.com.

16. Allman, E. and H. Katz, *SMTP Service Extension for Indicating the Responsible Submitter of an E-Mail Message*, in *RFC 4405 : SMTP Service Extension for*

*Indicating the Responsible Submitter of an E-Mail Message*. 2006, The Internet Society.

17. Lyon, J. and M. Wong, *Sender ID: Authenticating E-Mail*, in *RFC 4406*. 2006, IETF.

18. ISO/IEC, *ISO/IEC FCD 27043*, in *Information technology: Security techniques: Incident investigation principles and processes*. 2014, ISO/IEC. p. 27.

19. Carpenter, B. *The IETF Standards Process*. Internet Engineering Task Force 2015-08-25 [cited 2015 2015/10/01]; Available from: https://www.ietf.org/about/standards-process.html.

20. Postel, J., *Simple Mail Transfer Protocol*, in *Simple Mail Transfer Protocol*. 1982, IETF.

21. Klensin, J., et al., *SMTP Service Extensions*, in *RFC 1425 : SMTP Service Extensions*. 1993, IETF.

22. Klensin, J., *Simple Mail Transfer Protocol*, in *RFC 2821 : Simple Mail Transfer Protocol*. 2001, IETF.

23. *Simple Mail Transfer Protocol*, in *RFC 5321 : Simple Mail Transfer Protocol*. 2008, IETF.

24. Lyon, J., *Purported Responsible Address in E-Mail Messages*, in *RFC 4407*. 2006, IETF.

25. Wong, M. and W. Schlitt, *Sender Policy Framework (SPF) for Authorizing Use of Domains in E-Mail, Version 1*, in *RFC 4408*. 2006, IETF.

26. Allman, E., et al., *DomainKeys Identified Mail (DKIM) Signatures*, in *RFC 4871*. 2007, IETF.

27. Resnick, P., *Internet Message Format*, in *RFC 5322 : Internet Message Format*. 2008, IETF.

28. Postel, J., *Internet Protocol*, in *Internet Protocol*. 1981, IETF.

29. Atkinson, R., *IP Authentication Header*, in *RFC 1826*. 1995, IETF.

30. Deering, S. and R. Hinden, *Internet Protocol, Version 6 (IPv6) Specification*, in *RFC 2460 : Internet Protocol, Version 6 (IPv6) Specification*. 1998, IETF.

31. Rivera, J. and R. Van der Meulen, *Gartner Says 4.9 Billion Connected "Things" Will Be in Use in 2015*. 2014, Gartner: Barcelona, Spain.

32. Atkinson, R., *Security Architecture for the Internet Protocol*, in *RFC 1825 : Security Architecture for the Internet Protocol*. 1995, IETF.

33. Pfleeger, C.P. and S.L. Pfleerger, *Security in Computing*. 2010, Boston: Pearson.

34. Kent, S. and K. Seo, *Security Architecture for the Internet Protocol*, in *RFC 4301*, B. Technologies, Editor. 2005, IETF.

35. *IP Authentication Header*, in *RFC 1826 : IP Authentication Header*. 1995, IETF.

36. *IP Encapsulating Security Payload (ESP)*, in *RFC 1827 : IP Encapsulating Security Payload (ESP)*. 1995, IETF.

37. Crocker, D., *Internet Mail Architecture*, in *RFC 5598*, B. Internetworking, Editor. 2009, Network Working Groupe.

38. Resnick, P., *Internet Mail Format*, in *RFC 5322*, Q. Incorporated, Editor. 2008, Network Working Group.

39. Freed, N. and N. Borenstein, *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*. 1996, Network Working Group.

40. Nilsson, M., *The audio/mpeg Media Type*, in *RFC3003: MIME description for audio/mpeg media type*. 2000, IETF.

41. Crocker, D., T. Hansen, and M. Kucherawy, *DomainKeys Identified Mail (DKIM) Signatures*, in *RFC 6376*. 2011, IETF.

42. B. Ramsdell, S.T., *Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification.* RFC 5751, 2010.

43. Tracy, M., et al., *Guidelines on Electronic Mail Security*, T.A. National Institute of Standards and Technoligy, Editor. 2007, NIST: U.S.A. p. 139.

44. Thornton, L., *Guide to Achiving email compliance - a South African Perspective*, in *thornton.co.za*. 2003, Lisa Thornton Inc.

45. *Code of practice for information security managment*, in *ISO/IEC 27002:2005*. 2005.

46. Harrenstien K, Stahl M, and F. E, *DOD INTERNET HOST TABLE SPECIFICATION*. 1985, IETF.

47. Consortium, I.S. *DNS RFC*. [HTML] 2013 2015/05/15 [cited 2015 2013/06/16]; Available from: https://www.isc.org/community/rfcs/dns/.

48. Mockapetris, P., *DOMAIN NAMES - CONCEPTS AND FACILITIES*, ISI, Editor. 1987, IETF.

49. Mockapetris, P., *DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION*, IETF, Editor. 1987, IETF.

50. Wong, M. and W. Schlitt, *Sender Policy Framework (SPF) for Authorizing Use of Domains in E-Mail, Version 1*, in *Sender Policy Framework (SPF) for Authorizing Use of Domains in E-Mail, Version 1*. 2006, IETF.

51. Allman, E., et al., *DomainKeys Identified Mail (DKIM) Signatures*, in *RFC 4871 : DomainKeys Identified Mail (DKIM) Signatures*. 2007, IETF.

52. *No. 25 of 2002: Electronic Communications and Transactions Act, 2002*, in *Government Gazette*. 2002, Republic of South Africa: Cape Town.

53. www.spam-site.com, *http://www.spam-site.com/your-website-blacklisted.shtml*, in *www.spam-site.com*. 2006, www.spam-site.com.

54. *ELEVEN E-MAIL SECURITY REPORT – DECEMBER 2012*, in *ELEVEN*. 2012, ELEVEN Integrated Message Security.

55. *Symantec*, in *Intelligence report*. 2012, Symantech.

56. Even, L.R. *Intrusion Detection FAQ: What is a Honeypot?* http://www.sana.org.

57. Dagon, D., et al., *A Taxonomy of Botnet Structures.* 2007: p. 325-339.

58. ESET, *Botnet Definition*, in *http://www.eset.com/*. 2009.

59. Mueller, S.H., *Fight spam on the Internet*, in *Fight spam on the Internet*. 18.

60. Naidoo, N. *Introduction to Using Spam Methedology to Initiate Proactive Spam Controls*. [Document] 2007.

61. Linford, S. *The Spamhaus Project*. 2015 [cited 2012; Available from: https://www.spamhaus.org/.

62. Faynberg, I., *Method and Apparatus for Reducing Email Spam and Virus Distribution in Communications Network by Authenticating the Origin of Email Messages*, in *US2005/0203985A1*. 2004, USA.

63. Obied, A. *Honeypots and Spam*. [Document] 2006.

64. InternetNews, *Report Says Spam Arms Race Escalating*, in *http://www.ironport.com/*. 2009, IronPort In the News.

65. Walters, D., *Spam overwhelms email Messages*, in *BBC News / technology*. 2009, BBC.

66. Palmer, G.L., *Road Map for Digital Forensic Research*, in *Road Map for Digital Forensic Research*. 2002, Digital Forensic Research Workshop

(DFRWS).

67. Hadjidj, R., et al., *Towards an integrated e-mail forensics analysis framework.* Digital Investigation, 2009. **5**.

68. de Vel, O., et al., *Mining E-mail content for author identification forensics*, in *SIGMOND Record*. 2001.

69. Gupta, G., C. Mazumdar, and M.S. Rao, *Digital Forensic Analysis of E-Mails: A Trusted E-Mail Protocol.* International Journal of Digital Evidence, Springer, 2004. **2**(4).

70. *Define: Application Performance Managment*, in *http://www.techopedia.com*. 2010, Techopedia.

71. *What is application performance monitoring*, in *http://www.solarwinds.com/it-management-glossary/what-is-application-performance-monitoring.aspx*. 2010, Solarwinds.

72. Harrington, D. and J. Shoenwaelder, *Transport Subsystem for the Simple Network Management Protocol (SNMP)*, in *RFC5590*. 2009, IETF.

73. Zeilenga, K., *Lightweight Directory Access Protocol (LDAP): Directory Information Models*, in *RFC4512*. 2006, IETF.

74. Ylonen, T., *The Secure Shell (SSH) Protocol Architecture*, in *RFC4251*. 2006, Network working group.

75. Jacobson, V., C. Leres, and S. McCanne, *Linux man pages - tcpdump*, in *Linux man pages*. 2009, University of California: Berkeley.

76. McCanne, S., *libpcap: An Architecture and Optimization Methodology for Packet Capture*, in *Sharkfest 2011*. 2013, Riverbed Technology: Stanford University, Stanford, California.

77. Combs, G., *http://www.wireshark.org*, in *http://www.wireshark.org*. wireshark.org.

78. Jones, K.J., R. Bejtlich, and C.W. Rose, *Real Digital Forensics - Computer Security and Incedent Response*. 2010, USA: Addison-Wesley.

79. Volonino, L. and R. Anzaldua, *Computer Forensics for Dummies*. 2008, Safari online: For Dummies.

80. Cox, M. and D. Ellsworth. *Application-controlled demand paging for out-of-core visualization*. in *VIS '97 Proceedings of the 8th conference on Visualization '97*. 1997. IEEE Computer Society Press Los Alamitos, CA, USA ©1997.

81. Inc, G. *http://www.gartner.com/it-glossary/big-data*. 2013.

82. Russom, P., *Big data analytics.* TDWI best practices report, fourth quarter, 2011: p. 1-35.

83. Cárdenas, A.A., P.K. Manadhata, and S. Rajan, *Big Data Analytics for Security Intelligence*, in *Cloud Security Alliance*, A.A. Cárdenas, P.K. Manadhata, and S. Rajan, Editors. 2013, Cloud Security Alliance: https://cloudsecurityalliance.org/download/big-data-analytics-for-security-intelligence/. p. 21.

84. Pollitt, M. and A. Whitledge, *Exploring Big Haystacks*, in *Advances in Digital Forensics II*, M. Olivier and S. Shenoi, Editors. 2006, Springer New York. p. 67-76.

85. Irons, A. and H.S. Lallie *Digital Forensics to Intelligent Forensics*. Future Internet, 2014. **6**, 584-596 DOI: 10.3390.

86. Wang, L. and C.A. Alexander, *Big Data in Distributed Analytics, Cybersecurity, Cyber Warfare and Digital Forensics.* Digital Technologies, 2015. **1**(1): p. 22-27.

87. Technology, N.I.o.S.a., *Guide to Computer Security Log Management* in *Recommendations of the National Institute of Standards and Technology*. 2006, Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology: Gaithersburg, MD 20899-8930

88. Corperation, A. *Apache Hadoop main page*. 2015 [cited 2015 2015-09-08]; Available from: http://hadoop.apache.org/.

89. Casey, E., *Digital Evidence and Computer Crime Second Edition*. 2004, San Diego: Elsevier Academic Press.

90. Farmer, D. and W. Venema, *Forensic Discovery*. 3rd ed. Professional Computing Series. 2008, CapeTown: Addison-Wesley.

91. Estival, D., et al. *Author profiling for English emails*. in *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics (PACLING'07)*. 2007.

92. Vacca, J.R., *Public Key Infrastructure: Building trusted applications and Web services*. 2004, New York: CRC Press.

93. Moore, T., *Economics of digital forensics*, in *Fifth Workshop on the Economics of Information Security*. 2006, Weis: Cambrige, England.

94. MySQL, *MySQL 5.7 Manual*, in *The ARCHIVE Storage Engine*, Oracle, Editor. 2015, Oracle: https://dev.mysql.com/doc/refman/5.7/en/preface.html.

95. Van Staden, F.R. and H.S. Venter, *Adding digital forensic readiness to the email trace header*. 2010, IEEE: Johannesburg.

96. Van Staden, F.R. and H.S. Venter, *Adding digital forensic readiness to electronic communication using a security monitoring tool*. 2011, Information Security South Africa (ISSA): Johannesburg.

97. Moore, T. *Economics of digital forensics*. 26 - 28 June, 2006. Cambrige: Fifth Workshop on the Economics of Information Security.

98. Jacobson, V., C. Leres, and S. McCanne. *Tcpdump*. Tcpdump manpage 2009 26 February 2014 [cited 2014 29 April]; Manual for the use of tcpdump].

99. Fluke. *optiview-xg-network-analysis-tablet*. 2013 [cited 2014 2014-04-07]; OptiView® XG Network Analysis Tablet]. Available from: http://www.flukenetworks.com/enterprise-network/network-monitoring/optiview-xg-network-analysis-tablet.

100. Ozment, A. and S.E. Schechter, *Bootstrapping the Adoption of Internet Security Protocols*, in *The Fifth Workshop on the Economics of Infromation Security*. 2006, Weis: Cambridge, England.

101. Aiello, M., et al., *SMTP sniffing for intrusion detection purposes*, in *Second Italian Workshop Italiano on PRIvacy and Security*. 2007, PRISE: Rome. p. 53-58.

102. Nagele, W., *Hadoop-Pcap.* GitHub, 2015.

103. Deering, S. and R. Hinden, *Internet Protocol, Version 6 (IPv6) Specification*, in *RFC 2460*. 1998, IETF.

104. Hadjidj, R., et al., *Towards an integrated e-mail forensic analysis framework.* Digital Investigation, 2009. **5**(3-4): p. 124-137.

105. Kitterman, S., *Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1*, in *RFC 7208*. 2014, IETF.