

A Forensic Framework for Webmail Threat Monitoring and Log Analysis



By

Abdul Saboor Malik

FALL-2018-MS-IS 0000276573 SEECS

Supervisor

Dr. Muhammad Khurram Shahzad

Department of Computing

A thesis submitted in partial fulfillment of the requirements for the degree of Masters
of Science in Computer Science (MS CS)

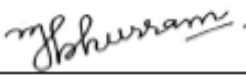
In

School of Electrical Engineering & Computer Science (SEECS) , National
University of Sciences and Technology (NUST),
Islamabad, Pakistan.

(April 2022)

THESIS ACCEPTANCE CERTIFICATE

Certified that final copy of MS/MPhil thesis entitled "A FORENSIC FRAMEWORK FOR WEBMAIL THREAT MONITORING AND LOG ANALYSIS" written by ABDUL SABOOR MALIK, (Registration No 00000276573), of SEecs has been vetted by the undersigned, found complete in all respects as per NUST Statutes/Regulations, is free of plagiarism, errors and mistakes and is accepted as partial fulfillment for award of MS/M Phil degree. It is further certified that necessary amendments as pointed out by GEC members of the scholar have also been incorporated in the said thesis.

Signature: 

Name of Advisor: Dr. Muhammad Khuram Shahzad

Date: 15-Apr-2022

Signature (HOD): _____

Date: _____

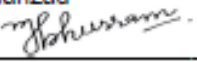
Signature (Dean/Principal): _____

Date: _____

Approval

It is certified that the contents and form of the thesis entitled "A FORENSIC FRAMEWORK FOR WEBMAIL THREAT MONITORING AND LOG ANALYSIS" submitted by ABDUL SABOOR MALIK have been found satisfactory for the requirement of the degree

Advisor : Dr. Muhammad Khuram
Shahzad

Signature: 

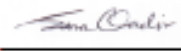
Date: 15-Apr-2022

Committee Member 1: Dr. Mehdi Hussain

Signature: 

Date: 14-Apr-2022

Committee Member 2: Dr. Sana Qadir

Signature: 

Date: 14-Apr-2022

Committee Member 3: Dr. Hasan Tahir

Signature: 

Date: 14-Apr-2022


Dedication

This thesis is dedicated to my parents, siblings and my teachers with love and gratitude.

Certificate of Originality

I hereby declare that this submission titled "A FORENSIC FRAMEWORK FOR WEBMAIL THREAT MONITORING AND LOG ANALYSIS" is my own work. To the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at NUST SEecs or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST SEecs or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics, which has been acknowledged. I also verified the originality of contents through plagiarism software.

Student Name: ABDUL SABOOR MALIK

Student Signature:  _____

Acknowledgments

I would like to extend my gratefulness to my supervisor, Dr. M Khuram Shahzad, for keeping me inspired and guiding me comprehensively during this process. This research work would not have been possible without his kind supervision. Thanks to all the GEC committee members: Dr. Mehdi Hussain, Dr. Hasan Tahir, and Dr. Sana Qadir for their valuable insights and recommendations.

Thanks to my family and friends for their inspiration and support.

Abdul Saboor Malik

Table of Contents

1. INTRODUCTION	1
1.1 Email Communication System.....	1
1.1.1 Components of E-Mail System.....	2
1.1.2 Email Client-side protocols	3
1.1.3 Simple Mail Transfer Protocol (SMTP)	4
1.1.4 MIME format.....	5
1.2 Cloud Based Email Services	6
1.3 Email Security Threats and Challenges	6
1.3.1 Common Threats	7
1.4 Motivation	8
1.5 Problem Statement	8
1.6 Research Objectives	8
1.7 Scope	9
1.8 Research Challenges	10
1.9 Thesis Outline	10
2 Summary	11
2. BACKGROUND AND LITERATURE REVIEW	12
2.1 Email Security Domains	12
2.2 Email Security (Literature).....	13
2.3 Email Forensics, Tools and Webmail	17
2.4 Summary	18
3. RESEARCH METHODOLOGY	19
3.1 Nature of Research	19
3.1.1 Quantitative research	19
3.1.2 Qualitative research	19
3.1.3 Inductive Research:	19
3.1.4 Deductive Research:	20
3.2 Research Cycle	20
3.3 Research Phases	21

3.4 Research Method.....	22
3.4.1 Problem Identification	22
3.4.2 Literature Review	23
3.4.3 Research Based Questions.....	23
3.4.4 Research Design	23
3.4.5 Environment Setup and Implementation	24
3.4.6 Analysis	25
3.4.7 Presentation	25
3.5 Summary	25
4. PROPOSED SCHEME.....	26
4.1 Proposed Framework Methodology	26
4.1.1 Single Process Dumping	26
4.1.2 Parent Process Identification.....	26
4.1.3 Full memory dumping.....	26
4.1.4 Analysing email evidence	27
4.1.5 Single Process Identification.....	27
4.1.6 Process memory dump.....	28
4.1.7 Using windows systinternals.....	28
4.2 Proposed Solution (Overview)	28
4.2.1 Malicious insider case scenario	29
4.3 Framework Implementation	31
4.3.1 Webmail Logging Tool	31
5. PERFORMANCE EVALUATION OF PROPOSED FRAMEWORK	34
5.1 Experimental Setup	34
5.1.1 Systems and Software Used	34
5.1.2 Experiment.....	35
5.1.3 Evaluation Metrics.....	35
6. CONCLUSION AND FUTURE WORK	39
6.1 Conclusion.....	39
6.2 Future Work	39

REFERENCES	41
APPENDIX A – WINDOWS SYSINTERNAL AND VOLATILITY TOOLS COMMANDS..	45
APPENDIX B – CODE FOR PYTHON FORENSIC LOGGING TOOL	46

LIST OF TABLES

Table 2.1: Popular Email Forensic Tools	17
Table 5.1: Tool performance.....	35
Table 5.2: Memory Dump Comparison	37
Table 5.3: Comparison from other schemes	37
Table 5.4: Performance Comparison from other schemes.....	37

LIST OF FIGURES

Figure 1.1: Email Protocol Architecture	3
Figure 1.2: IMAP and POP protocol representation.....	4
Figure 1.3: SMTP Protocol Workflow 1.....	5
Figure 1.4: SMTP Protocol Workflow 2.....	5
Figure 1.5: Key Features of our Proposed scheme	10
Figure 2.1: Email Security Categories	12
Figure 2.2: Research Work on Email Security	13
Figure 3.1: Research Cycle	21
Figure 3.2: Research Phases	22
Figure 3.3: Research Design	24
Figure 4.1: Framework Overview	30
Figure 4.2: Algorithm for Pattern matching the New Email	32
Figure 4.4: Python Tool Workflow.....	33
Figure 5.1: Logging Tool Memory Performance.....	36

LIST OF ABBREVIATIONS

POP	Post Office Protocol
IMAP	Internet Message Access Protocol
SMTP	Simple Mail Transfer Protocol
FTK	Forensic Tool Kit
CSP	Cloud Service Permisses
MTA	Message Terminal Agent
MUA	Message User Agent
HTTPS	HyperText Transfer Protocol Secure
IM	Instant Messaging
SSL	Secure Sockets Layer
URL	Uniform Resource Locator
APT	Advanced Persistent Threat
PII	Personal Identifiable Information
OS	Operating System
ASCII	American Standard Code for Information Interchange
DMP	Memory Dump File
RAM	Random Access Memory
MX	Mail Exchanger
CLI	Command Line Interface

ABSTRACT

Today, webmail is being deployed in many organizations for all kinds of normal and important communications. Several cyber threats involving phishing, malicious insider, and ransomware attacks are primarily targeted through webmail. This poses challenges and limitations for forensic investigators in the analysis as compared to email clients, where they have access to email files. The majority of work on email forensics and detection focuses on email client artifacts stored on a disk. To gather artifacts about email activity from webmail used in browsers, volatile memory forensics is gaining popularity. Few research work, utilizing memory forensics approach are focused on external email threats such as spoofed email detection, to create user activity logs and gather artifacts. The present work lacks a generic framework with some new tools which can perform the tasks periodically and efficiently in terms of performance and storage. Moreover, present schemes are not applicable to detect internal email threats, where a malicious user can send a new email containing confidential information. In this work a novel method is proposed, to monitor, detect, log and gather information about new email activity using volatile memory forensics. In our research work, a framework is proposed to address the internal threat related to webmail. The proposed scheme, efficiently creates user activity logs from browser parent process memory as the user creates a new email. To implement and test the framework a python tool was developed that perform the tasks periodically with good performance efficiency from previous schemes in terms of memory dump size, file sizes, and logs creation time. The framework is equally applicable for both public and private browsing. The proposed method can also be applied to create logs about spoofed email as proposed in previous schemes. Our proposed method provides forensics investigators with a novel webmail logging tool that can be used to gather artifacts about malicious email activity from insiders.

Keywords: Email Forensics, Malicious Email, Memory forensics, Webmail Threat Det

Chapter 1

1. INTRODUCTION

Over the years, the use of Email communication is much increased in all our day-to-day communication, encompassing the personal and commercial space of our lives. As our lives are connected to the internet all the time through social media, e-commerce, and entertainment, we are always getting notified of new email messages. Considering the cybersecurity threats in today's digital world, the free and open nature of email services brings new areas of security gaps to protect the information the email users and the information being shared over the email. As more and more migration to cloud-based service continues various challenges arise for security researchers and experts in email threats detection, analysis, and forensic domains. This chapter provides a brief overview of research work on email security and forensics.

- Section 1.1 introduces the basic email communication system.
- Section 1.2 describes the trending shift to cloud email services.
- Section 1.3 gives the details about the latest email security threats and challenges.
- Section 1.4 gives a survey of email forensics challenges and tools.
- Section 1.5 Email security Threats concerning webmail.
- Section 1.6 presents the problem statement.
- Section 1.7 states the research objectives.
- Section 1.8 defines the scope of this research.
- Section 1.9 gives an outline of the thesis work.
- Section 1.10 summarizes this chapter.

1.1 Email Communication System

One of the earliest forms of digital communication since the internet began is E-mail. Email service allowed the message exchange over the internet from one part of the world to another. The email message content consists of text, documents, images, voice, and video. A standard protocol is defined with its system and components into a hierarchical system. The originator of an email is termed as **sender** and the person who receives mail is called **the recipient**. The email system is mapped based on the physical post mail service.

1.1.1 Components of E-Mail System

Email system in its basic form is categorized in the following components. The interworking of these components is also presented in the figure below: User-Agent (UA), Message Transfer Agent (MTA), Mail Box, and Spool file[25] [36].

The functionality of Each component is defined below.

1. **User Agent (UA):**

The UA is software, program, or interface for the user to send and receive email. Often termed as mail reader. It contains mailbox settings and allows various commands to the end-users of email, to create and delete an email messages.

2. **Message Transfer Agent (MTA) :**

MTA is an essential part of the email system which involved in transferring an email message through different nodes in the Email network. It is a server-side node. Both the sender and receiver of the email should have an MTA that transmits and receives an email. Then the MTA transfers email to the mailbox user. The email is transferred following the SMTP protocol.

3. **Mailbox:**

It is local or remote storage holding the mails. The owner of the email must have a mail box setup. The mailbox allows presenting the email received and modify them. Access must be authenticated to the owner only.

4. **Spool file:**

This file contains mails that are to be sent. The files exist on the email server side as it arrived from the client computer. The outgoing emails are added to the spool file using SMTP. MTA extracts the remaining emails from the spool files to be delivered. E-mail allows a mailing list concept in which one alias can represent various email addresses. If the mailing list is there, a separate message is sent to all entries in the list by the MTA otherwise a single email is sent to the alias itself.

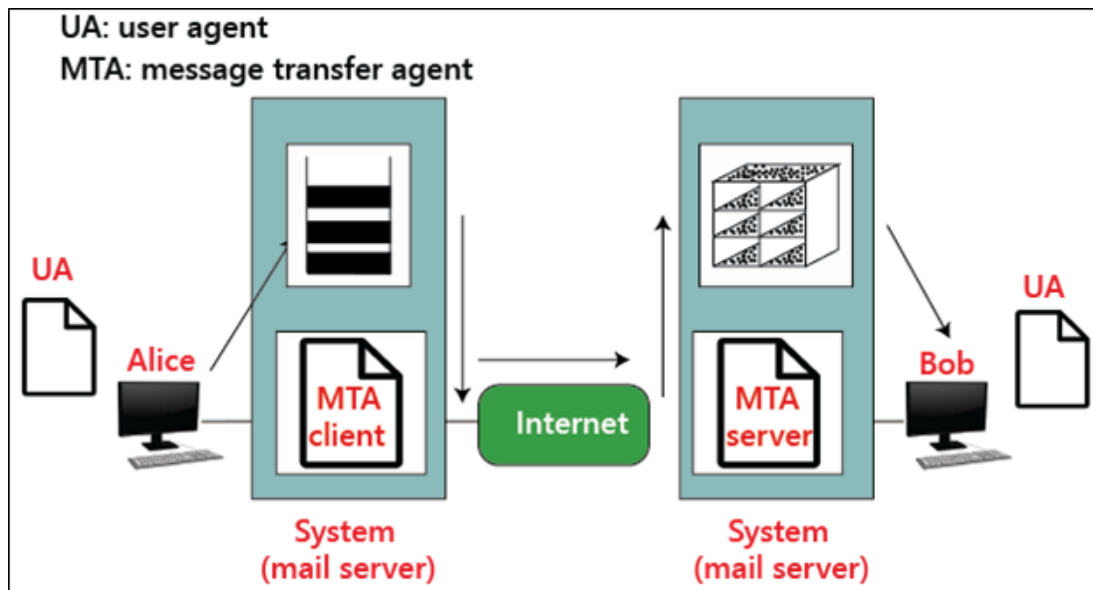


Figure 1.1: Email Protocol Architecture

Email systems follow different protocols to provide message sending and receiving functionality. The Email message from one computer to another computer travels through various stages as defined in Figure 1.1[25]. The SMTP protocol is a push protocol that moved the email to the email server and across other email servers. In order to fetch the emails from the server to the end-user mailbox in an email client software, pull protocols are used. Two of the pull protocols used are IMAP v4 and POP3.

1.1.2 Email Client-side protocols

In the POP3 protocol, both clients and servers are configured with POP3 settings. After the configuring POP setting on TCP port 110. Mail retrieval starts and the mailbox starts loading the email from the server after the user enters his email and password.

Another standard for accessing mail is the Internet Mail Access Standard, version 4 (IMAP4). IMAP4 comes with more features than POP3. IMAP4 is more powerful and more complex. There are several forms POP3 is deficient. It does not require the user to arrange their mail on the server; there are various directories on the server that the user cannot [25]. However, POP3 does not allow the user to review the mail content partially before downloading. The diagram illustrating the protocol is shown in Figure 1.2 [25].

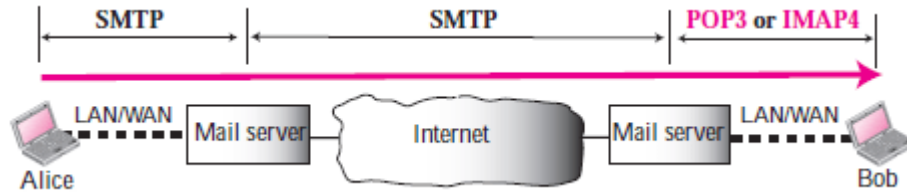


Figure 1.2: IMAP and POP protocol representation

1.1.3 Simple Mail Transfer Protocol (SMTP)

MTAs share the email contents with one another using the SMTP protocol. It is a request and response protocol followed by data exchange between the MTA servers. The SMTP commands and protocol sequence is shown in Figure 1.3 and Figure 1.4 [25].



Figure 1.3: SMTP Protocol Workflow 1

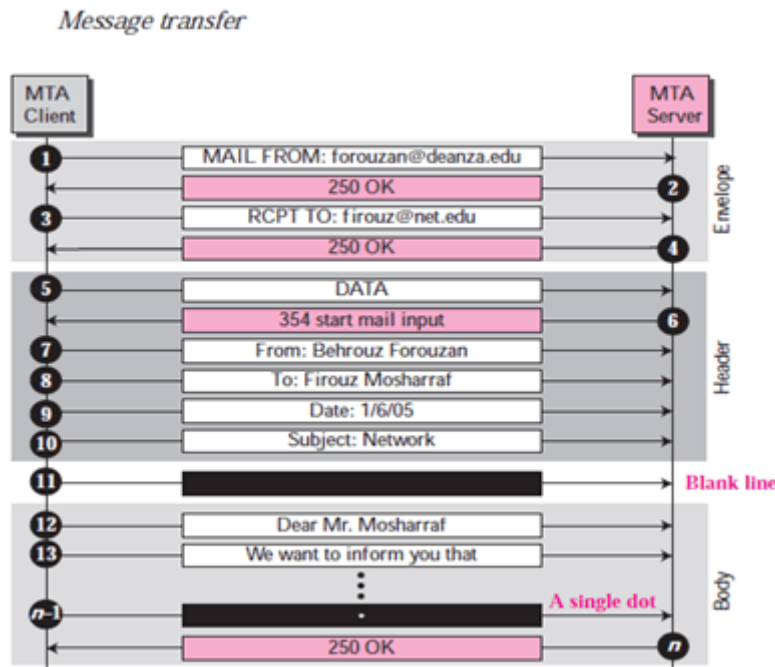


Figure 1.4: SMTP Protocol Workflow 2

1.1.4 MIME format

The Email structure is much different than normal message layout. The message is encoded in 7-bit NVT ASCII format. However, it brings the bottlenecks, as only English language can be well represented and other languages such as (such as French, German, Hebrew, Russian, Chinese, Urdu and Japanese) cannot be used. The audio, video and binary files also cannot be send using MIME [12] [25].

MIME is a supplementary that aids in sending the non ASCII data over email. At the sender location, MIME converts non-ASCII data into NVT ASCII data and transfers it to the MTA client to be sent over the Web. The message is converted back to the original data at the receiving site. MIME can be seen as a set of software functions that transform non-ASCII data into ASCII data, and vice versa

1.2 Cloud Based Email Services

Emails are still as popular as the social media networks and texting. As popular giants provide free mobile and web-based communications for Next to social networks and texting. Enterprise also use the cloud email services like Gmail, Outlook and Yahoo. The data shows that around 293 billion emails are sent every day. Therefore, it is likely to stay here and be an important communication medium. Moreover, new cloud email services are increasing in addition to popular ones. Hence, the security challenges linked with emails are increasing [12] [14].

These three cloud services are used in great numbers as shown by the statistics.

- **Google Gmail:** The most popular, reliable, fast, intuitive and user-friendly cloud based email service. According to Android police, Gmail is the most used service worldwide. According to stats from 2021, Gmail has about 1 billion registrants and 750,000 active users.
- **Microsoft Outlook:** It's in second place with 400 million active users. It can be used independently from the Office 365 platform and has big advantage in that it is pre-installed in Windows PCs. It's very responsive and intuitive to use.
- **Yahoo Mail:** It has 230 million active users. Since the early 1990s, when yahoo was the only email provider, and still used by many before Google in 2000s. It has recently improved its services and still much popular .
- **Yandex Mail:** Yandex is a popular search engine for news reading, claiming about 85 million monthly users.
- **Mozilla Thunderbird:** Thunderbird is another popular email client services available for all 3 popular Desktop operating system and. It offers easy integration and notification for users. It has around million email users.

1.3 Email Security Threats and Challenges

This widely used medium of communication is also a great source for adversaries and hackers to launch cyber attacks, targeting organizations, infrastructures, and individuals [7]. The latest statistics depicts a continued rise in email-borne threats. According to the latest FireEye reports, email phishing and URL-based attacks increased by 17% and 26% respectively

Moreover, file sharing through cloud email services and new impersonation attempts on payroll and supply chains have increased. Therefore, email communications are vital in threat detection and computer forensics for post-incident analysis.

1.3.1 Common Threats

Since e-mail system is widely deployed, well understood, and used to communicate within and outside any organisation, it is frequently the target of various attacks. Attackers can exploit e-mail to gain control over an organization, access confidential information, or disrupt IT access to resources. Common threats to e-mail systems include the following [35]:

Malware: Increasingly, attacker's user bulk email messaging to target organizations and users to trigger link click or downloading of "malicious software," that include viruses, worms, Trojan horses, and spyware. The attacker success may give the malicious entity complete control of the target server or computer. Using this, the attacker gain access to sensitive information, carry out covert monitoring of user activities or execute any commands [35].

Spam and phishing: The unsolicited commercial e-mail, commonly referred to as spam, is the sending of unwanted bulk commercial e-mail messages. Such messages can disrupt user productivity, utilize IT resources excessively, and be used as a distribution mechanism for malware. Phishing is another form of spam, which refers to the use of tricks and social engineering to entice users in responding to the e-mail and gaining access to private information of users. Malicious email systems and servers are setup to send email to target a user and make it look like legitimate.

Social engineering: Before actually attacking user through some malware, the attacker can make use of a lot of email communication to get useful information. A common social engineering attack is e-mail spoofing, a user pretends to be a known person or service by changing the original fields of the email.

Entities with malicious intent: Malicious entities threat is a growing worry and difficult to model for the organizations. Someone having intentions to harm the company as an employee can have easy access to the various computers. Using such access, the user can reach anywhere in the internal network and launch an attack from inside to attack critical network devices such as servers. In such way, the email passwords and other details of various internal users are compromised [35].

Unintentional acts by authorized users: Security compromise via email can be unintentional too. Authorized users may inadvertently send proprietary or other sensitive information via e-mail, exposing the organization to some legal and public defamation.

Data Exfiltration: Transferring useful data from a target computer in single or multiple times refers to data exfiltration. It is often triggers by malware (such as APT), or some internal malicious insider. It is also an unauthorized data excursion. The information can be very critical if coming from compromised servers of user computer. The data can include very critical information such as personally identifiable information (PII), personal financial information and, in some cases, cryptographic keys. These leaks may have undermined privacy laws and incur heavy fines

Protect against high-level BEC (Business E-Mail Compromise): These attacks involve using advance phishing campaigns and techniques such as spear phishing and whaling. The attacker

target the executives of the company and CEOs to make spoofed emails causing compromise of business workflows. These emails are much difficult to be flagged as reported by CSO Online, only 30% of companies equipped to flag such identical emails. This leads to issues like fraud, internal damage to company's reputation and external irregularities with partners [35].

There are various type of email security threats and attack vectors that keeps growing with time. However, with the cloud email system the threats can be divided into two types which is mainly insider threat and outsider threat. Various research work and countermeasures focus on threats detection from the outside. However, the threats from the inside is a growing challenge in modern cloud-based email era. This is a daunting challenge; organizations have to face. Hence, a research gaps exist to tackle these challenges and propose new solutions, schemes and tools.

1.4 Motivation

As we move towards more cloud-based solutions with email services such as Yahoo, Gmail and Hotmail. The security challenges have become more complex. The ease of using email services with webmails, makes a lot of attacks easier which includes attacks from outside of the organization to a user (such as phishing, spam, malware etc). Threats from the inside are also very important to handle and needs strong measures, as they can easily cause data exfiltration from company, by simply sending an email with malicious intent and transfer any unauthorized data to outside world covertly. This makes a challenging task for source attribution for security teams and forensic investigators. Considering this, we aim to bridge the gap and propose new methods that can be used to monitor and log the end user activity as they use webmails service to send email message or some attachment to other users.

1.5 Problem Statement

The cloud-based webmail system has brought challenges in email security for forensics and threat detection teams. The internal (malicious entity) threat surface landscape has become complex with cloud-based webmail services. Therefore, its visibility and analysis at the users end, and forensics acquisition in the form of logs and email evidence from CSPs has become difficult due to global legal, geographical and political implications. Therefore, to overcome this research challenge in webmail, a new framework is proposed to help security teams with visibility and knowledge of evidence about internal webmail threats.

1.6 Research Objectives

Considering the webmail security challenges, our proposed solution will achieve following objectives.

1. Propose a solution to monitor and log webmail activity to detect any malicious insider.

2. Use memory forensic approach efficiently to create logs of communication at client side.
3. Develop efficient automated tool to analyze webmail.
4. Help the security researches and forensic investigators in analyzing and monitoring webmails in the organization for email threats analysis and forensics.

Using volatile memory acquisition of clients computer in efficient way, we aim to achieve our proposed scheme. To automate the various processes of gathering email data, memory dumps and analysis, we use following approach:

1. Python libraries psutil to monitor running browsers processes.
2. Windows sys internal tools procdump and strings utility for memory acquisition.
3. Using Chrome and Firefox Browsers.

Our research objectives are achieved for windows-based computers as they can make use python-based tools with windows sysinternal commands. The tool can run in background and perform operations seamlessly and create logs in text format.

1.7 Scope

After detailed review of the current research work on email threats, email forensics and email tools used in threat detection and forensics. We have defined the scope of our research to webmail services taking the specific case scenario of tracking and logging activity of a malicious insider.

We have used windows-based tools and created the webmail forensics logging and threat detection tool. Our scheme is based on the previous two schemes proposed in literature for email spoofing detection using volatile memory forensics. Our proposed solution provide efficient memory dumps and create automated logs. The tools performance analysis shows our proposed solution efficiency in terms of time, storage size of dumpfiles and log files. Figure 1.5 presents few unique features of our proposed scheme.

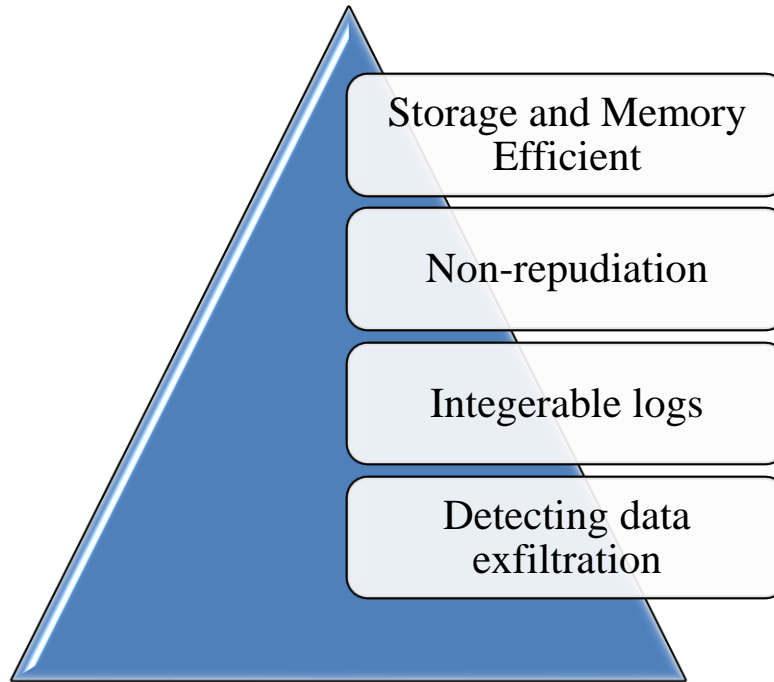


Figure 1.5: Key Features of our Proposed scheme

1.8 Research Challenges

Webmail don't provide evidence of any email files stored on local computer as the email information resides at the CSP side. The only way to gather information about any email is from memory through periodic memory dumps. To find process memory dumps of the current opened tab, various dumps were acquired after sending an email using web services and then finding out the browser process holding the information. The previous proposed scheme made use of windows system to create dumps and Linux systems to analyze the dumps, which was a challenging and time-consuming task.

1.9 Thesis Outline

To organize our thesis work for better understanding, The work is divided into seven chapters. In this chapter, we provide overview about the email system and changing environments and trends in email system and the arising security challenges. Problem statement and research objectives are provided here in depth.

Chapter no. 2, “**Background and Literature Review**” discusses the existing work done on email security. It provides a brief overview of email security measures at various levels and techniques, methods, tools and frameworks used to secure email infrastructure and how the tools are helpful in email threats detection and forensics.

Chapter 3 discusses the “**Research Methodology**” we outlined various phases of our research and key steps taken in our research. In this chapter we categorized our research methodology with generic frameworks and defined various research methodologies such as qualitative, quantitative and deductive research.

Chapter 4 describes the “**Proposed Framework**” which highlights our proposed solution. In this chapter we defined our framework, tools, process work flow, and algorithm. The information on the new tool development, implementation of our proposed tool and its experimental setup is also defined in this chapter.

Chapter 5 consists of “**Performance evaluation and Analysis of Proposed Framework**” in which we gathered qualitative and quantitative evaluation of our framework. It illustrated the our framework and tool performance in terms of time, memory and resources consumption. It also provides the qualitative evaluation of our proposed scheme with results of logs evidence.

Chapter 6 describes the “**Conclusion and Future Work**” where the entire research work is concluded and the future aims are discussed. As concluding remarks, future works are also discussed here.

Chapter 7 covers the “**References**” which provides the bibliography of our research work in IEEE format.

2 Summary

In this chapter, we have provided a detailed introduction to the Email system and evolution into popular cloud based web services. We also discussed various webmail challenges in threat detection and forensics which helped to identify the problems and motivated us to propose new solution to perform webmail threat detection that would be helpful for the forensic investigators and security teams.

Chapter 2

2. BACKGROUND AND LITERATURE REVIEW

Email security is an evolving topic in the research community as the threats increase due to changes in email technology. New solutions, protocols and methodologies are introduced to reduce the security gaps in the current email architecture. This Chapter gives a brief overview of the work on information security and solutions proposed in the research community. This chapter provides a detailed background and review of latest trends in the email security domain. Through our detailed review, new research gaps were identified and provided baseline to work on shortcomings of previous work. Further sections of this chapter are as below.

- Section 2.1 gives broader view of email security and its sub category.
- Section 2.2 gives insight about the different levels of threats in Email system and proposed solution for counter measures.
- Section 2.3 discussed methods, tools and techniques on Email Forensics and Limitations with webmail
- Section 2.4 summary.

2.1 Email Security Domains

The Figure 2.1 provides a broad view point of literature review concerning the email.

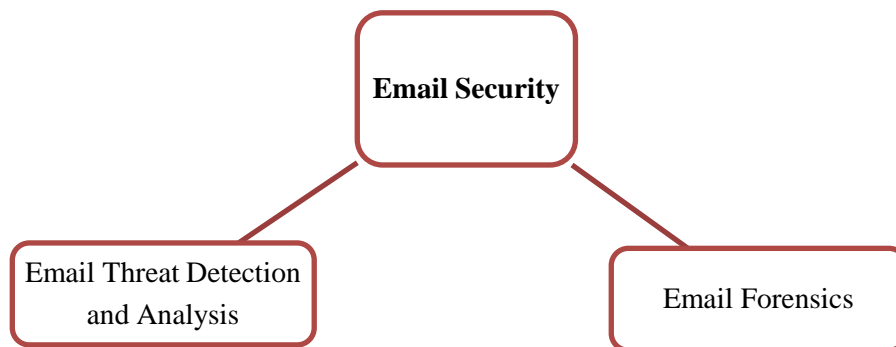


Figure 2.1: Email Security Categories

Email Threat Detection and Prevention: various types of email threats exist and continues to evolve as defined above. These threats are further divided into external and internal threats. Figure 2.2 provides some broad domains in which the email security is improved to address these external and internal email threats. In the external threats, email spoofing and email spamming are a growing threat vectors. Various research work have contributed to proposed solution for it at network level, email service provider leve and email client perspective. In the internal threats, detection and prevention of malicious insider is a growing challenge that needs to be addressed by security researchers and provides a room for more research on client side..

Email forensics: Another popular aspect of email security domain is the ability to perform email forensic.This domain relates to tools, techniques and methods used to investigate the email related evidence. Once the email evidence is obtained, which consist of email files of different formats, investigators use these email files for email header analysis, for source identification in any incident which involves email communication. The survey about these forensic tools are described in section 2.3.

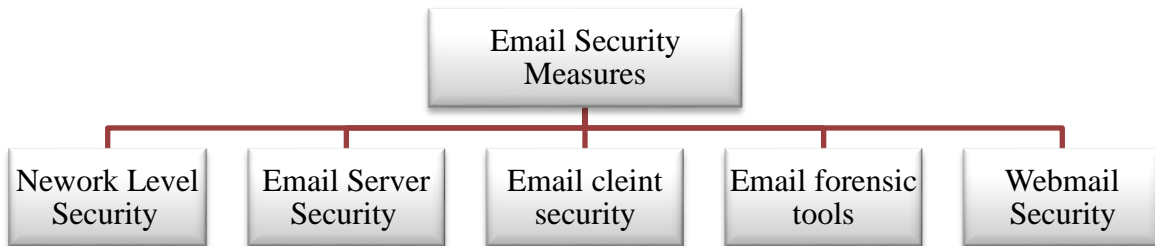


Figure 2.2: Research Work on Email Security

2.2 Email Security (Literature)

Email system follow the client server model. The original architecture design, had many security weaknesses and a lot of research work has addressed the security issues and challenges that keeps

changing as the technology evolves. Email security literature mostly comprises of efforts to improving the security of email architecture at different level as shown in Figure 2.2.

Email system are usually open nature since the beginning as the email follows open ports 25 for plain email communication. Hence, the security researchers have addressed the email security issues at various levels. The network level security relate to improving the vulnerabilities in SMTP protocols and email communication ports. The email server security soltuions, are concerned with the email severs that holds the email information of the users and security settings for email server like email headers, certificates and ports configurations etc. Email client security involve the soltuions and research efforts for improve the email security for end users, as the people using the email services are a direct victimn of these email borne attacks. The Email forensics is another widely research area in email security and various solutions, case studies, frameworks have been proposed. Researchers have proposed various new tools and techniques for email evidence analysis. Few popular tools are tabulated in Table 2.1. Since few years, there is a wide change in cloud-based email service adoption. Individual and organisations have moved from on premises to cloud solutions for their email infrastructure needs. Hence, the new era of webmails security is emerging and bringing new challenges and issues on forensic and threats detection from internal and external sources.

Spam threat is a common type of email security that has been addressed mostly as a server side security measures. In a comprehensive survey about email spam detection in [37] authors have shown various techniques which includes non AI based techniques (such as DKIM, SPF, domain whitelisting, heuristic filtering and content filtering). Authors also discussed some AI based techniques using algorithms such as SVM and Randomforest. These AI and non AI based measures are mostly applicable at server or SMTP architecture level with some shortcomings due to less uniform global adoption of these measures, compliance issues with exsiting email infrastructure, false positives, computations weakness, alogorithm accuracy and low dataset.

Some of the research work focus on obtaining the email headers from intercepting the network traffic in [27][28]. Using the network traffic analysis, the email spoofing threat is detected in [29]. These network level defenses are used mainly to block the users from external sources to check the spoofed email [29]. However, network traffic is often encrypted as in the case of webmail and not allow to view actual content. Authors in [38] carried out the survey of provider email security at network level for secure provider-provider email communication. The use of strong cipher suites and TLS security makes up most the cloud email infrastructure. Hence, from the organizational perspective, the email analysis threat detection and forensic analysis work is shifting more towards the end user side. Hence, various client side detection schemes have been proposed recently in the email security literature

Analysis of email headers is a basic part in email forensics. The headers and body of any email to be investigated are often attained from email files stored on end user or the enterprise servers.

Authors in [39] has used email headers fields for detecting for email forging case and identified the time , sources of email and contents.

Authors in [30][31][32] utilized the network level defenses such as DMARC, SPF and DKIM to address any malicious email from outside and detect the email headers for spoofing attempts.

In an email forensic process, first the forensic copy of complete system(including harddrives and RAM) is obtained and analysed on different system that has tools for email files analysis. The authors in [40] have proposed a PLugsE framework and used evidence from google chrome profiles, cookies information to access the gmail account and gather email information through HTML parsing.

Recently, as the webmail service continues to grow, the traditional threats and forensic landscape is evolving with new schemes and challenges. For forensic investigators, it is challenging to have access to cloud services for email files due to its legal complication. Researchers have proposed live memory forensic acquisition to help attain email artifacts which are stored in RAM. The memory acquisition method has been explored by researchers for email threat detection such as spoofing and email forensics.

Getting information from memory is the only way in various cases since more email, messaging and chat application are browser based. It is very easy for users to delete the browser history. Authors in [41] have discussed scenario for application of live memory forensics which includes Google drive uploads, web link accessed through Tor, malware artifacts from memory, Whatsapp and proton mail communication. In such case, authors have shown the evidence of signatures from volatile memory to identify the activity and its contents.

To address the spoof email directly on the client side, authors in [33] proposed a tool to parse the spoof emails and let the user know about any email spoofing attack they got in their inbox.

In email forensics, the memory acquisition is an integral part as it gives a detailed view about the current state in which the computer is acquired and data is collected. It also holds information about the current open programs. Hence a lot of critical information is retrievable from memory and widely explored in literature. Authors in [34] have used memory forensics approach to gather user passwords from web application. Another research work used the memory forensics to detect the malware and APTs. This approach has also been used in analysis of virtual machines for useful artifacts with a different algorithm. Researchers in [1] have also utilized the memory acquisition approach to gather artifacts of messages from the web messages applications such as skype, messenger etc.

Recently few research works have also focused on applying memory acquisition technique to gather email evidence for webmail in Windows, Android and detecting the email spoofing attacks.

Utilizing memory forensics for email threats detection and forensic has recently got attention among researchers. Various solutions are proposed that incorporate memory acquisition, filtering, and extraction of useful strings to find evidence. Researchers have proposed different frameworks and schemes to capture and analyze the variable nature of data in memory [3]. In [1] a framework for web-based social media and Instant Messaging (IM) application has been proposed that works on software-based memory acquisition. In [2] authors proposed way of detecting email spoofing by taking a periodic dump of Yahoo webmail activity from RAM dump and extractions and filtering string search to detect spoofing on received and replied emails. obtained from memory dump text files.

In [1,4,16] researchers demonstrated the extraction of email artifacts stored in RAM for specific platforms such QQMail and defined where some fields can be found using string search. In [5] authors proposed the RAMAS framework and built the extraction module for round cube email. It relies on the complete windows full size RAM dump to gather the evidence such as message information, author, timestamps, and recipient's email.

Email is an integral part of today's web environment services and required new framework. In [42] authors have discussed the need for new frameworks for web environment. They have identified the limitations and proposed some new framework features which are also applicable for webmail services. These limitations are in associating online persona, evidence access from cloud service provider, relevant context identification and ability for new tools integration. They proposed new framework to support evidence acquisition, analysis space reduction, timeline reconstruction and structured formatting.

In few recent effort, memory acquisition approach is also applied for email spoofing detection. In [2] authors proposed a way of detecting email spoofing by taking a periodic dump of Yahoo webmail, and extractions and filtering string search to detect spoofing on received and replied emails obtained from memory dump text files. These text files are used to create logs from the current dump file about email evidence and content. It used a complete memory dump from Forensic Toolkit (FTK) image software. The memory acquisition process takes a complete dump of the whole memory. So if the memory is 12 GB, the size of the memory dump would also be 12 GB, which leads to storage issues on the client side. On the client side such scheme can cause inefficient use of storage as periodic dumps sizes are large and exhaust the storage capacity of end-user system.

In [8] authors proposed bringing some improvement into memory forensics scheme from [2] and proposed spoof email detection by taking all browser running processes instead of full memory dump and used Mail Exchanger (MX) record to detect email spoofing. The schemes proposed in [2] and [8], are only applicable if the user opened sent item and inbox item before taking the memory dump. These scheme don't provide much information about how the new tools

that performs the automated process. Moreover, these schemes can be improved to create new email logs, which can be helpful to know about new email activity to prevent malicious insider email attempts.

Using the approaches defined in [2], [8] and [42], we have addressed some shortcomings and proposed framework to generate email messages logs from the RAM for the browsing session.

2.3 Email Forensics, Tools and Webmail

Overview of key proposed solutions in different literatures is shown below [26][9]. These forensic tools are dependent on the email format files. These files are further parsed to get the details from headers. As organisations and normal users move to cloud-based webmail services, the threats challenges and areas of email security have expand. Since the webmail store all the data on cloud over the internet and less evidence about any email activity is available on the user side, more research works are utilizing memory acquisition approach to enhance email security and prevent security attacks and incidents over webmail.

The memory acquisition approach also provides help to forensic experts in email security investigations. The memory forensics approach is a bit less explored in email security domain but more in general evidence acquisition. The table below gives a brief overview of latest and popular email forensic tools, however, these tools are much less applicable and helpful in gathering evidence from webmail. However, for the webmails, the email data with few headers information is stored in RAM and use json format and string format. These forensic tools are less capable to parse webmail from memory dump files acquired for email evidence, which we have used in our research. Therefore, this motivates to develop new tools that can be used to parse the webmail content browser process memory dump.

Hence, there is a room for research work to propose new solution that help the forensic experts in investigations involving webmail services.

Table 2.1: Popular Email Forensic Tools

Memory Forensic Tool	Main Features	Limitations
Add4Mail	Analyse various file formats such as PSTs, MBOX etc.	<ul style="list-style-type: none"> • Cannot parse the live webmail • Dependent on specific email format files
Paraben Email Examiner (EM)	-	-
eMailTrackerPro	-	-
MailXaminer	-	-

EmailTracer	-	-
AbusePipe	-	-
FINALeMAIL	-	-

2.4 Summary

In this chapter, we have explored the recent literature work on email security to improve the limitations in the Email system to control and prevent various threats. We have also covered various approaches, tools and working solutions to tackle email forensics. As per our findings, a lot of research work to improve email security, address the Threats and Forensics challenges are based on external threats. Various tools used are dependent on analysis of specific email format when obtained from email client software. However, working with modern webmail services, the malicious insider threat detection, monitoring and forensic analysis is a big open challenge, which needs to be explored further. The thorough literature review has helped us proposed new solution and technique that address these gaps in the Email Security domain, in further sections of our research work.

Chapter 3

3. RESEARCH METHODOLOGY

In this chapter we have defined the methodology of our research and procedure and methods we have followed to provide solution to the problems and fill the existing gap in literature. The research methodology is categorized in four terms. First, the four basic types of research methodologies are defined. Following these research methods we defined out proposed methodology. The subsections of this chapter are as followed.

- Section 3.1 nature of research.
- Section 3.3 research cycle.
- Section 3.4 research phases.
- Section 3.5 summary

3.1 Nature of Research

Nature of research is briefly explained as:

3.1.1 Quantitative research

In this method, the research is described in the quantitative form through graph or numbers. It is used to verify the assumptions and theories.

3.1.2 Qualitative research

In this method, the research is described in form of words. The decisions are based on experiences and thoughts about the topic. This research lets you know about the greater insights into the specific topic. The concepts and theories are explored through detailed literature review.

3.1.3 Inductive Research:

Inductive research is another form which takes when there is no specific theory or related research available on a topic. Inductive research involves below 3 stages.

1. Observation.
2. Pattern observation.
3. Theory development.

The shortcoming such research is that these are theoretical in nature and it needs to be validated and tested based on some experiment.

3.1.4 Deductive Research:

Deductive research starts with a pre-existing theory, procedure or research which gives the base for the rest of the research. In deductive research the new theories are tested through experiments and follow the four step process below:

1. Starting with an existing theory.
2. Formulating a hypothesis.
3. Data collection.
4. Analysis.

The results of deductive research depend on the assumptions set during inductive research but if any of the assumption fails then deductive research cannot be carried out effectively.

So, in our research methodology, to proposed solution of malicious webmail detection and forensics, we followed the deductive research following a base research and used various tools and techniques to collect relevant evidence data and perform analysis and further evaluation.

3.2 Research Cycle

A generic model proposed by Frankfort-Nachmias & Nachmias as shown in Figure 3.1, defines the process of how the key areas to be addressed in the research. To carry out our research we made used of this model to guide us on the process of our research, starting from defining the problem statement and reach to conclusion[43].



Figure 3.1: Research Cycle

3.3 Research Phases

Our work is focused on webmail threat and forensic so applying the above mentioned model in section 3.2, we did our research in various different phases. After thorough research on present literature on webmail forensic and threat detection, we implemented the few schemes defined in some recent paper. A single method was shortlisted that followed memory acquisition . Through experimentation and implementation of current work, a new efficient and automated framework was proposed which we will defined in upcoming chapter. The Figure 3.2 represents salient phases of our research[44].

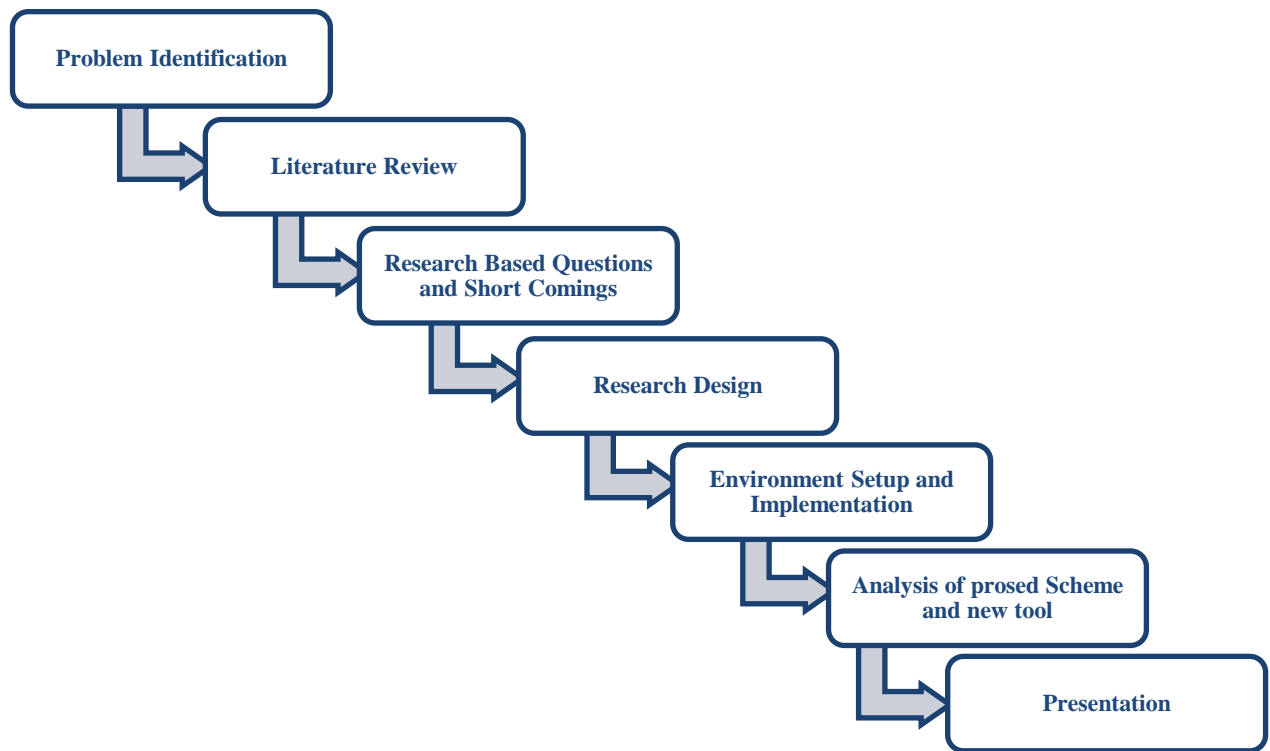


Figure 3.2: Research Phases

3.4 Research Method

It is important to know a holistic view of a problem and solutions to know the complete subject, for which research is necessary. It helps to collect the information and analyze it. Problem identification, experimentation and presentation of information play a key role in research as they provide a new addition to the existing problem. In this section we will discuss the problem statement. Based on literature review, we will define the steps and methods to make a new scheme which we will validate and implement in our upcoming chapters.

3.4.1 Problem Identification

Before starting the research process, it is necessary to define a target problem, in which a specific domain and subdomain is selected and the scope is narrowed down to a further specific problem of interest. This study is focused on proposing a new solution for webmail threat detection to help

the security teams and forensic investigators in any investigations regarding webmails. Our research methods and literature review helped to narrow down the issues and gaps in new tools, threat vectors, and efficient methods and technique that can be helpful in webmail threat detection and forensics.

3.4.2 Literature Review

The problem identification leads to the literature review to know various existing and non-existing solutions which are currently available to solve the problem. The literature review is helpful to gain in depth knowledge of methods, techniques and tools being used and also identify limitations in all of them. Therefore, helping to bring new solutions and ideas to proposed, implement and test.

For our research, we studied various conference papers, generals, and articles on email security, email security tools, email forensic tools and techniques. Then we focused on webmail security and forensic issues concerning the tools, threat vectors, frameworks, schemes implementation and performances and then we used this as a baseline to propose new techniques to work on different threat vector i.e a malicious insider.

3.4.3 Research Based Questions

After the thorough literature review on webmail and email in general following the quantitative and deductive research methodology, the gaps were more apparent in current schemes and work done on webmail threat detection. This provide way to address questions such as if any more efficient and automated scheme can be proposed on different threat vector such as malicious insider and how it performs with qualitative and quantitative results.

3.4.4 Research Design

In this subsection we define and formulate the process to carry out the new technique based on the existing literature work done. This includes working on our proposed framework, devising new technique or tools, and further testing and evaluating our implementation and the complete outcome of our solution in detection and logging of threats in webmails. The brief research design followed for our proposed scheme is highlighted in Figure 3.3.

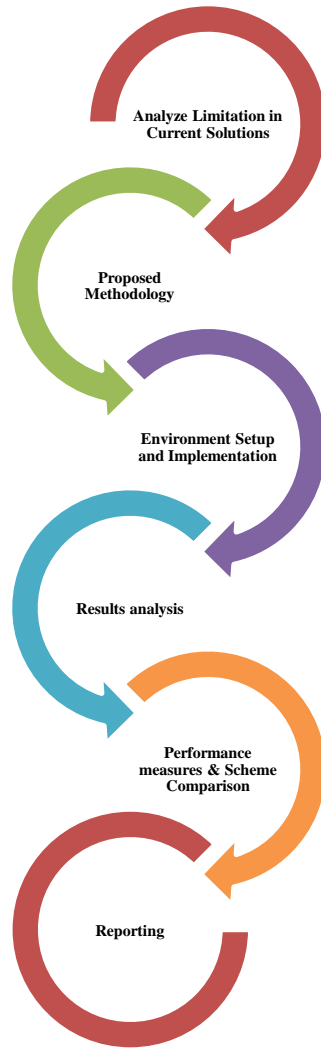


Figure 3.3: Research Design

3.4.5 Environment Setup and Implementation

To perform our proposed solution implementation, we made use of various software tools and methods. To implement and test the shortcoming we used both Linux and Windows based virtual Machine to perform email activity dump operations using FTK tool and volatility software to take memory dump, parse them and identify the process information and retrievable email evidence from those processes. Afterwards, we used the information to develop new tool in python on windows to create automated logs from memory for new emails. We have used procdump and strings sysinternals tool with Python 3.

3.4.6 Analysis

To test our framework and new tool for detecting malicious insider activity. The tool performance was tested using python memory profile library and storage details about log file and memory dumps. The results are compared with previous schemes and were presented in tabular format in coming chapters.

3.4.7 Presentation

The final reporting gives the detail of our scheme implementation and describes our scheme advantages in comparison with previous schemes which used memory based approach. It contains the results from our research in terms of generated logs, logging tool and performance results.

3.5 Summary

Research mythology is an integral part of any research. To propose our novel framework, this chapter gives detail break down of our research methodology which is based on several task starting with the problem identification to proposing and testing new schemes with new tools and obtaining performance results in a brief and concise form.

Chapter 4

4. PROPOSED SCHEME

This chapter contains the following:

- Section 4.1 gives a detailed insights about the steps, algorithms and methods used in our proposed solution.
- Section 4.2 gives our proposed solution.
- Section 4.3 gives the details above framework and its workflow implemented using our custom developed python tool.

4.1 Proposed Framework Methodology

Following our base research papers, this subsection describes details about the methodology followed through experiments that helped us to propose a new framework in section 4.2. Some interesting findings, observations and experiments are described in this subsection.

4.1.1 Single Process Dumping

The schemes proposed in [2][8] utilized periodic memory dump of full memory of complete system and web browser processes memory respectively. To improve the dumping process a single process dumping approach is used in our research. To gather the information about the single process, our methodology is explained below.

4.1.2 Parent Process Identification

Our proposed framework makes use of single parent process dump of browser process memory to gather email evidence. Through experimentations and implementation described below, a new framework is proposed that is more storage and memory resource friendly utilizing parent process based memory dumping. To get a snapshot of information stored at a particular time instance in computers volatile memory, the memory acquisition tools are used. Previous schemes have used FTK imagers for this purpose, which is widely used in memory forensic acquisition.

4.1.3 Full memory dumping.

FTK imager is a popular tool that is used to create full memory dump of windows memory. It is widely used by forensic investigators to create a forensic copy of the complete windows system

memory when analyzing any forensic system for investigations. It can take the complete memory dump for both hard disk and volatile memory of computer. In our case, only RAM dump was required to be obtained, as it helps to know what is stored in volatile memory when the user opens the browsers, opens mailbox and performs activities (like create new email, view an email from inbox or open an email from sent items.). It is also important to have knowledge about the format and layout of email data in RAM and how it is formatted, for automated parsing and extraction through tool. Various iterative experiments were conducted in which the memory dumps were obtained as the user perform email activity on Chrome and Firefox browser through their Gmail, Yahoo and Outlook accounts [2].

4.1.4 Analysing email evidence

FTK Image tool creates a .mem file which is equal to the size of the RAM installed on system. In our experimental setup we used 1 GB RAM on windows 10. Therefore, various .mem files of 1 GB were obtained for each test case of email activity. The tool also allows you to open and parse the .mem content in text format. Using ASCII string search provided by the FTK tool, the email evidence was discovered about email sender, receiver and email message body strings. The experiments helped the digital footprints of email stored in RAM, which helped to explore the evidence at the process level.

4.1.5 Single Process Identification.

FTK imager tool result is limited to only the hexadecimal and ASCII view only. It cannot tell much about the process information. As in the memory, all the content belongs to some process, to further analyze the email information in the context of the process, a detailed process based view of memory information is required. For this purpose, Voatility tool was used.

Volatility tool is a popular memory forensic tool by the volatility foundation. It is an open source tool and available for multiple OS such as (Windows and Linux). It has a newer version (version 3) released in 2020 which is open source python based and available on Github. The version 2 is also available for windows. The tool has multiple features and plugins available which enhance the tool functionality to gain an detailed view to analyze computer volatile and physical memory dumps. An interesting plugin which can be used to find the process information about the certain information in the memory file is the yara plugin [28]. It is primarily used in malware analysis to identify the strings in the contexts of the program process IDs.

To analyze our samples for web browser process information, the memory dump files in .mem files were analyzed using volatility framework yara plugin. Using volatility framework in a separate Linux system the email evidence strings were searched. The results obtained provided details about the process information that holds the corresponding email evidence information. All the other dump files from Yahoo, Gmail and Hotmail memory dumps were also searched using volatility yara plugin [28]. Using this iterative experiment, it was found that only parent process of the browser holds the information about the email created, sent, and viewed as the user makes use of webmail service in the web browser.

4.1.6 Process memory dump

Through our iterative experiments above, the browser processes holding the webmail related information were successfully identified. Hence, instead of taking the dump equal to the size of the RAM which can exceed 8 GB and above for modern computers, it is much feasible to take only the browser process memory dump. The browser process memory dump is memory and resource friendly approach if periodic dumps are required. When the browser starts, the program consists of various child process but single parent process. Each of these process have their own process memory which can be acquired using processed memory dumps.

4.1.7 Using windows sysinternals

Windows sysinternals tools is a bunch of useful tools for windows operating system for various tasks. One of the sysinternals tools used in our research is the Procdump. It runs through the windows command line and allows the process memory dump of running processes in windows. It has different arguments that can be given to use the tool as per the requirement. In our research we have used this tool to write memory dump (.dmp) of browser parent process memory, as soon as the browser is closed. The dmp file extensions need to be converted to text file to parse and analyze the email evidence stored in the dump file to produce logfiles containing email activity and email message details[21][24]. For this purpose, another sysinternals tool, Strings was used. It is a command line tool which provides the text file output from the .mem dump file. A detailed usage of procdump and strings is described in appendix A.

When the browser opens, various processes exist as visible in the task manager. The browser has a single parent process ID but various number of child processes IDs. Therefore, if the parent process is known, the windows sysinternals tools can be used to create memory dump of single process physical memory and get the email evidence through some automated tool.

The above experiments helped to define the key processes for our framework which can be used in webmail threat detection and forensic logging. Moreover, the research method guided and motivated to develop a new tool which implements the framework. To automate the dumping process and working with web browser processes a python based tool was developed. More details on the tool are described in section 4.2 and 4.3.

In the next section, our proposed solution is defined along with the workflow of custom developed tool. As a case study for our framework, a malicious insider threat vector is assumed.

4.2 Proposed Solution (Overview)

To address the threats detection in the webmail and improve forensic analyses capabilities we proposed a new framework as shown in Figure 4.1 below which consist of four stages. Based on the previous research work that utilized the memory forensic approach for email spoofing

detection, we have proposed a novel method to monitor, dump and log the new email message sent using webmail. An interesting use case of our approach is to address the malicious insider who can covertly make use of webmail to craft and send any unauthorized information.

4.2.1 Malicious insider case scenario

As discussed above, it is a challenging task to detect insider making use of webmails to send confidential data, since there are no logs available about the user activity at the client side. As shown in [4], the contents of webmail are very short lived, as the user switch between different tabs, or close the browser after sending email. The contents of email from the memory are only retrievable if the continuous dumps are taken for all the time as long as the webmail tab is opened. Hence, it is very challenging to model and make accurate detection to gather browser memory dump. Some of the possible case scenarios for malicious insider browsing activity is defined below. For our research we have used the S3 malicious insider case scenario:

S1: The user opens the new tab → login to webmail → writes an email consisting of useful information and attachments → sends the email → closes the browser session.

S2: The user sends a malicious email → continues to browse → close the browser tab.

S3: User opens the web browsers → opens webmail service → creates a new email to share some data → close the browser.

The proposed framework is described below in Figure 4.1 in 4 stages taking the S3 case scenario of malicious insider. A brief overview of processes involved in our framework are further described.

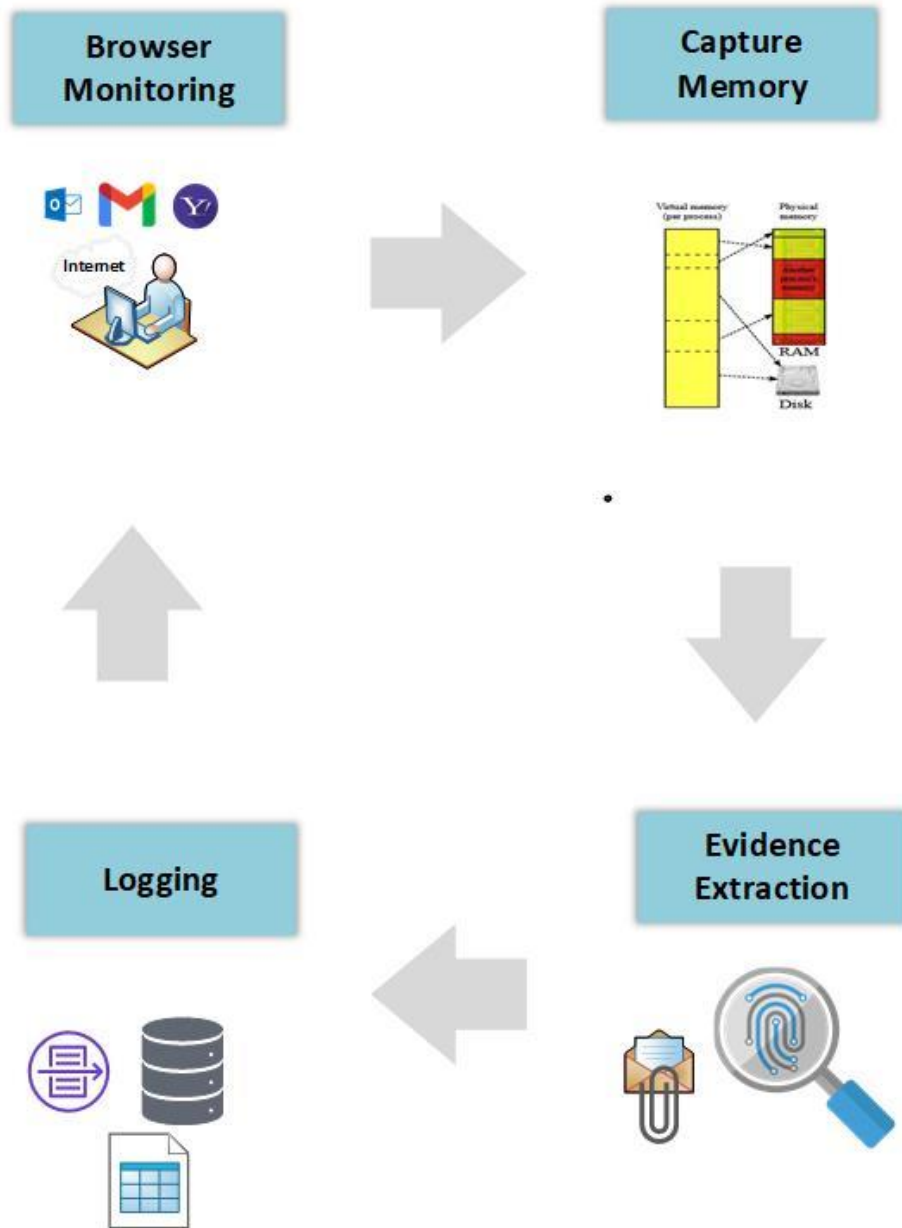


Figure 4.1: Framework Overview

1. **Browser Monitoring:** As the user send any new email using browser. It is necessary to know the browser running state. In this phase, the browser process is monitored as the user opened web browser.
2. **Capture Memory:** As the user sends as email and closes the session(closing the main browser) as described in our case scenario, the browser volatile memory is captured using procdump and creates a string file from the .dmp file. The string file is further used in 3rd phase for evidence gathering.

3. **Evidence Gathering:** In this phase, the string file is parsed to get the evidence about any new email sent using the new email pattern detection defined for each webmail services.
4. **Logging:** In the last stage, after the algorithm finds any new email messages stored in the format, the contents of email are copied into a log file.

The process continues again when a new malicious insider activity starts following all the 4 phases.

4.3 Framework Implementation

To implement our proposed solution, there is a need to develop a tool that follows the workflow and performs the automated tasks in the four phases. For this purpose, a new tool is developed that follows the four stages and is periodically running in the background. The tool is developed in python that automates all these processes and implements the framework.

4.3.1 Webmail Logging Tool

Python is a high-level language that is very popular nowadays. Its popularity index is increasing rapidly and it has a lot of applications in various industries and the latest growing technologies such as AI, ML, DataScience, cybersecurity, web application, etc. Python supports a lot of libraries to work with almost anything to build new tools and software easily. In fact, python is popular among security researchers to make new tools and software to solve various research challenges [18]. We used python to develop a new tool that implements our proposed framework.

Following are the components we used to create our program. Following are the components we used to create our program.

1. To continuously monitor the browser processes, the python psutils library is used. Python psutils library allows to work with Windows processes and fetch details and information about the running processes, memory, and CPU consumption. Using the psutils functions the browser parent IDs are found and continuously monitored as the application runs in background [22].
2. Python allows to communicate with windows shell and execute commands. Using these features the windows Sysinternals programs (Strings and Procdump) are executed from the python tool [24][21]. As the browser session is closed, python executes the memory dump command along with some other arguments and creates the dump file and afterward, uses strings command to produce a string file of the memory dump which is parsed.
3. To gather the evidence and extract the new email, the python tool parse the string file for pattern matching of the email message lines with headers and contents. The pattern matching algorithm is shown in Figure 4.2 below, which is found through the experiment as described above in section 4.1.

```

if "\"newMessage\":true" in line:
    #print("Email service: Yahoo")
    print(line);
    logs.append(line);
if "\"UpdateItemJsonRequest" in line:
    #print("Email service: Hotmail")
    print(line);
    logs.append(line);
if "\"3\", \"2\"" in line:
    #print("Email service: Gmail")
    print(line);
    logs.append(line);

```

Figure 4.2: Algorithm for Pattern matching the New Email

- After pattern matching algorithm, the email message lines with headers and contents details are stored into a separate log file. The log file is created once and periodically updated as the new activity occurs. A sample result of the log file is shown in Figure 4.3 below. The information about email body, some email headers fields and body data are shown below. The data is organised in JSON format. The workflow of the webmail logging tool is shown the in Figure 4.4.

```

Email logs of ... @hotmail.com
{
  "__type": "UpdateItemJsonRequest:#Exchange", "Header": {
    "__type": "JsonRequestHeaders:#Exchange", "RequestServerVersion": "V2018_01_08", "TimeZoneContext": {
      "__type": "TimeZoneExchange", "Id": "Greenwich Standard Time"
    }
  }, "Body": {
    "__type": "UpdateItemRequest:#Exchange", "ItemChanges": [
      {
        "__type": "ItemChange:#Exchange", "Updates": {
          "__type": "SetItemField:#Exchange", "Path": {
            "__type": "PropertyUri:#Exchange", "FieldURI": "ToRecipients"
          }, "Item": {
            "__type": "Message:#Exchange", "Meta-Data": {
              "Importance": "Normal", "IsReadReceiptRequested": true, "IsResponseRequested": true, "IsSendReceiptRequested": true, "IsStatusChanged": true, "Subject": "Test"
            }
          }
        }
      }
    ]
  }
}
{
  "__type": "SetItemField:#Exchange", "Path": {
    "__type": "PropertyUri:#Exchange", "FieldURI": "CcRecipients"
  }, "Item": {
    "__type": "Message:#Exchange", "CcRecipients": [
      {
        "__type": "SetItemField:#Exchange", "Path": {
          "__type": "PropertyUri:#Exchange", "FieldURI": "BccRecipients"
        }, "Item": {
          "__type": "Message:#Exchange", "BccRecipients": [
            {
              "__type": "SetItemField:#Exchange", "Path": {
                "__type": "PropertyUri:#Exchange", "FieldURI": "Subject"
              }, "Item": {
                "__type": "Message:#Exchange", "Subject": "Hello123456"
              }
            }
          ]
        }
      }
    ]
  }
}
{
  "__type": "SetItemField:#Exchange", "Path": {
    "__type": "PropertyUri:#Exchange", "FieldURI": "Body"
  }, "Item": {
    "__type": "Message:#Exchange", "Body": {
      "__type": "BodyContent:#Exchange", "BodyType": "HTML", "Value": "<html><head><meta http-equiv='Content-Type' content='text/html; charset=UTF-8' /><body><div></div></body></html>"
    }
  }
}
{
  "__type": "SetItemField:#Exchange", "Path": {
    "__type": "PropertyUri:#Exchange", "FieldURI": "Importance"
  }, "Item": {
    "__type": "Message:#Exchange", "Importance": "Normal"
  }
}
{
  "__type": "SetItemField:#Exchange", "Path": {
    "__type": "PropertyUri:#Exchange", "FieldURI": "IsReadReceiptRequested"
  }, "Item": {
    "__type": "Message:#Exchange", "IsReadReceiptRequested": false
  }
}
{
  "__type": "SetItemField:#Exchange", "Path": {
    "__type": "PropertyUri:#Exchange", "FieldURI": "IsDeliveryReceiptRequested"
  }, "Item": {
    "__type": "Message:#Exchange", "IsDeliveryReceiptRequested": false
  }
}
{
  "__type": "DeleteItemField:#Exchange", "Path": {
    "__type": "ExtendedPropertyUri:#Exchange", "DistinguishedName": "AQMkADAwATYwMAItZjI1YjMTg3LTAwA10wMAoARgAAA5FaZeS2YKNNkFbzQqixAYsHAAe5n0BU+sdsEsrIAM2zJEnOAAACAQ8AAAAHuZzgVpRHLKvADNisyRjz1QAEBZ00"
  }, "Item": {
    "__type": "Message:#Exchange", "Id": "AQMkADAwATYwMAItZjI1YjMTg3LTAwA10wMAoARgAAA5FaZeS2YKNNkFbzQqixAYsHAAe5n0BU+sdsEsrIAM2zJEnOAAACAQ8AAAAHuZzgVpRHLKvADNisyRjz1QAEBZ00"
  }
}
{
  "__type": "Message:#Exchange", "AlwaysOverwrite": true, "ClientSupportsIrm": true, "SendCalendarInvitationsOrCancellations": "SendToNone", "MessageDisposition": "SendAndSaveCopy", "SuppressReadReceipts": false, "NewMail": true, "PromoteInlineAttachments": false, "SendOnNotFound": true, "ItemShape": {
    "__type": "ItemResponseShape:#Exchange", "BaseShape": "IdOnly", "AdditionalProperties": [
      {
        "__type": "PropertyUri:#Exchange", "FieldURI": "ItemLastModifiedTime"
      }
    ]
  }
}
METADATA

```

Figure 4.3: Email logs result

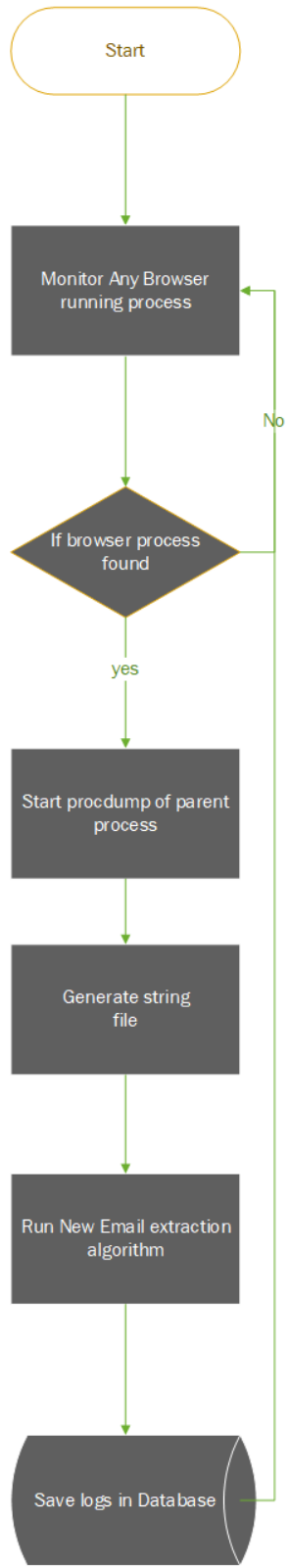


Figure 4.4: Python Tool Workflow

Chapter 5

5. PERFORMANCE EVALUATION OF PROPOSED FRAMEWORK

This chapter described the performance evaluation of our proposed webmail threat detection framework. In the performance evaluation we have evaluated our custom build tool form some metrics. In this chapter, firstly, our experimental setup is described to get the performance evaluation and implement our research, the qualitative and quantitative metrics we used to evaluate our proposed solution along with the results obtained. Then, based on the data we performed the comparison of our proposed scheme with previous schemes utilizing memory acquisition for email threat analysis.

5.1 Experimental Setup

To get our solution working along with the new tool developed, it is important to test its performance in order to get the results about how or proposed scheme is performing. In our experimental setup we made us desktop computers and Laptops as used in the modern enterprise environment. The operating system used were was windows 10. The systems to be used were connected to internet and were used to send email and results were gathered.

5.1.1 Systems and Software Used

So, to test and evaluate our framework, following are the hardware and software specifications of our test environment.

I. Hardware

- Laptop.
- OS: Windows 10
- Processor: Intel(R) Core(TM) i5-3337U CPU @ 1.80GH
- RAM: 4 GB, 8GB and 12GB
- HardDisk : 100 GB SSD

II. Software

- FireFox Web Browser
- Chrome Web Browser
- Python 3.6
- ProcDump (Windows systinternal)
- Strings((Windows systinternal))
- Python Libraries (cprofile, memProfile)

5.1.2 Experiment

The system configured with above specification was used to run our custom python tool that continuously monitored the web browser processes and creates logs. To test the tool working, we used three different accounts each on three popular webmail services (Gmail, Yahoo and Outlook). In the experiment, we logged into each account in a single session of Chrome and Firefox web browser, opened the respective webmail services, created a new email, made attachments with a test file, sent email to other two email accounts of different webmail service and closed the browser. This step was performed iteratively for other two webmail services to obtain to test the functionality and applicability of the tool. The python tool as continuously running in the background, which performed the tasks of memory dumping and logs creation from memory after parsing the string files [21][24].

5.1.3 Evaluation Metrics

To evaluate our proposed solution, we choose both the qualitative metrics as well as quantitate metrics.

Quantitative evaluation: In this type of evaluation, we evaluated the performance of our tool and well as our proposed framework in terms of storage and computation efficiency. The framework performance gives details about the size of the files produced on the user system. These files include the memory dump files, string files and the email log files. Also, the time to create these files was also measured. The values obtained for time and storage are shown in the table below. As compared to the previous two schemes from our base research paper, the file size, time and storage consumptions are much less and gives more efficient results.

Webmail Service	Logs creation Time (sec)	ASCII version of .dmp file size (MBs)	Memory dump size(MB)	Memory consumption (MiB)
Yahoo Mail	52.871318	29	385	26
Gmail	40.953343	29	385	26
Outlook	01:12.256179	29	385	26

Table 5.1 Tool Performace

To measure our tool performance itself, we check the memory consumption of our tool using cprofile and memprofile python libraries. Using these libraries, the performance of our tool was evaluated with some data values that gives information about computing power and memory, utilized by tool as it follows its basic tasks [19]. After the automated process of dumping, memory acquisition, string file creation and log files using our python tool, the time and storage results are

compared with the other two research works using volatile memory forensics on email spoofing detection. Our scheme shows reduced logs creation time which is less than 1 minute, the dump file sizes are much less in size as compared with other schemes. The browser process memory dump is also constant as the dump file is overwritten using procdump in each new session as the user performs activity. The previous schemes follow complete RAM memory dump in large sizes of GBs and all browser processes memory dumps. Compared to that our approach is more storage friendly to be used on end desktop system. Our proposed tool is also tested for RAM consumption and consumed around 25-26 MiB. As compared to previous schemes, we provided an efficient automated tool which can be deployed in organisational desktop environments. The memory profile results of our proposed Logging tool is shown in Figure 5.1 below.

Line #	Mem usage	Increment	Occurrences	Line Contents
70	25.4844 MiB	25.4844 MiB	1	@profile(precision=4)
71				def main():
72				while (1):
73	25.8711 MiB	0.3867 MiB	1	browserppid = if_process_is_running_by_exename("firefox.exe")
74	25.8711 MiB	0.0000 MiB	1	if browserppid != 0:
75	25.8711 MiB	0.0000 MiB	1	start_time = datetime.datetime.now()
76	25.8750 MiB	0.0039 MiB	1	print("---- Initialize Dumping of {}".format(browserppid))
77	25.8750 MiB	0.0000 MiB	1	procdump = "C:\Program Files\Internet Explorer\iexplore.exe -t -ma -o " + str(browserppid) + " proc_dump.dmp & exit"
78	26.0312 MiB	0.1562 MiB	1	output = subprocess.check_output(procdump, shell=True)
79	26.0391 MiB	0.0078 MiB	1	print(output)
80	26.0391 MiB	0.0000 MiB	1	print("Dump Complete")
81	26.0430 MiB	0.0039 MiB	1	generatestring()
82	26.2500 MiB	0.2070 MiB	1	createLogs()
83	26.2500 MiB	0.0000 MiB	1	end_time = datetime.datetime.now()
84	26.2500 MiB	0.0000 MiB	1	print(end_time - start_time)
85	26.2500 MiB	0.0000 MiB	1	break;
86				else:
87				print("NO browser is running")

Figure 5.1: Logging Tool Memory Performance

Schemes proposed in [2] and [8] have used different tools and methods for acquiring memory dump. Table 5.2 below shows different memory acquisition tools and their comparison with our scheme. In [2] Memorize tool is used to take live memory dump. It was used to create full memory dump of RAM. In [8], authors used Magnet Process capture which is a GUI based tool and allows to take running process memory dumps at different interval. It allows to take multiple process dumps, which authors have used to get all browser processes dumps. Both these tools and methods use software which are GUI based; requiring user interaction with less support for command line inputs which limits their usage to build new custom tools. In our proposed scheme, we have used Windows ProcDump tool to create periodic and automated dumps. This tool has a lot of command line options and allows to create a single process dump upon browser closing. This feature is not available in other tools and not explored in other memory forensic schemes. As a CLI based tool, it can be used with python for automated dumping of browser parent process. Our scheme and memory acquisition tool used can be explored further to build new tools. Our scheme and tools improve the memory dumping process which is described numerically in Table 5.1 above.

Volatile Memory Dumping Tool		
Mandiant Memoryze	Magnet Process Capture	ProcDump

Table 5.2: Memory Dump Comparison

Qualitative evaluation: In this type of evaluation, we made a comparison of our proposed scheme with the other similar approach in nature. The Table 5.2 below shows various issues addressed by our proposed scheme in terms of methodology, results, novelty and new tool development. Its gives a brief describes of our scheme evaluation with previous two schemes as proposed in [2] [8]. In Table 5.4, the performance comparison of our proposed tool and method with previous two schemes is provided.

Scheme Comparison			
Salient features	(Iyer 2017)	S. Shukla (2020)	Our proposed scheme
Automated Framework Implementation	-	-	A novel python-based tool proposed for automated process.
Memory Acquisition Method	Creates dump of complete memory	Uses Multiple browser process dumps	Uses single parent process dump
Generic Framework	-	Methods only focused on single email threat	Provides Generic framework for memory based detection and forensics
Malicious insider detection and forensics	Both schemes are based on single threat vector i.e email spoofing		Addressed malicious insider threat and created logs for new email.
Used sysinternal tools	-	Uses different third party tool and utilities	Makes use of default windows provided sysinternal tools
Periodic and Efficient logs creations	-	New email logs not provided	provides efficient logs creation from tool with minimum storage and efficient processing results
Memory Acquisition Approach	Used FTK for creating complete RAM dumps.	Used Magnet Process dumping tool. A GUI tool.	Windows provided CLI based tool. Provides a lot of command line

	Tools doesnot have command line options.	Provides very few command line options for automation.	options. Can be integrated to build new tools and enhancement.
New Tool Integration	No single automated tool proposed and source code shared.	No information on tool developed for automated dumping. No source code shared.	Provided the source code for complete workflow automation and tools required to run on single system. Open to further development on github.

Table 5.3: Comparison from other schemes

Performance comparison with previous schemes			
Salient features	(Iyer 2017)	S. Shukla (2020)	Our proposed scheme
Logs creation Time (sec)	43200	60	40 - 70
Memory dump size(MB)	Equal to RAM size in GBs	400 MB and above	385 MB
ASCII version of .dmp file size (MBs)	Around 400 MB	Greater than 50 MB	Approx. 29 MB
Memory Consumption (MiB)	50	32.7	26

Table 5.4: Performance Comparison from other schemes

6. CONCLUSION AND FUTURE WORK

This chapter our research work is concluded with further sections discussing some open areas to pursue further research work on webmail forensics involving malicious insider. Following sections are discussed in this chapter.

- Section 6.1 Conclusion
- Section 6.2 Future Work

6.1 Conclusion

With the increase in cloud-based services, the user of webmail services provided by various tech giants (Yahoo, Google and Outlook) have become very popular. The adverse side of these services open new doors for the attackers to carry out various forms of attacks through email. A malicious insider can easily take the advantage of the webmail and perform data exfiltration. Working in an enterprise environment, there are no such measures to detect and log such events as the webmail logs are stored at the cloud service providers. This makes a tough challenge for security administrators and forensic experts in investigations.

In our research, a novel framework is proposed to create email logs to aid security investigation and researcher in malicious email detection. Previously, the memory forensics have been used to detect spoof email. To use the scheme further to track the live email sending activity which can be helpful in analyzing the malicious insider email threat behavior, Our proposed scheme detects and logs any new email message sent using famous webmail services. Our proposed scheme is fast and efficient to be used desktop systems and laptops

For making live detection and forensic logging we developed a small tool with python to make the task automate. The tool performance is also tested with Memory Disk and CPU consumption suing python code test libraries. It can be easily deployed on a large scale on enterprise end user systems.

6.2 Future Work

In our proposed solution, we have created new email logs from 3 popular webmail services. These logs can be further analysed by security researchers and forensic experts to get more useful details. As an interesting future work, the logs can be fed into the SIEM solutions to gather details from all endpoints in any organisation. Moreover, the technique can be further improved with network packet analysis to gather the periodic memory dumps for the time when the email browsing is in progress.

The current scheme works for the basic case scenario of malicious insider as the email content loaded into memory are short lived and are not retrievable if the user switch to different tabs. Another interesting future work is to initiate the memory dump as user sent the email and switch to a different tab, so that the email evidence is not lost.

REFERENCES

- [1] Thantilage R, Le Khac N. Framework for the Retrieval of Social Media and Instant Messaging Evidence from Volatile Memory. 2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13thIEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE). 201
- [2] Iyer R, Atrey P, Varshney G, Misra M. Email spoofing detection using volatile memory forensics. 2017 IEEE Conference on Communications and Network Security (CNS). 2017
- [3] Chen L, Mao Y. Forensic Analysis of Email on Android Volatile Memory. 2016 IEEETrustcom/BigDataSE/ISPA. 2016
- [4] Barradas D, Brito T, Duarte D, Santos N, Rodrigues L. Forensic analysis of communication records of messaging applications from physical memory. Computers Security. 2019;86:484-497.
- [5] Bloomberg - Are you a robot? [Internet]. Bloomberg.com. 2021 [cited 17 December 2021]. Available from: <https://www.bloomberg.com/press-releases/2019-06-25/new-fireeye-email-threat-report-reveals-increase-in-social-engineering-attacks>
- [6] SANS Internet Storm Center [Internet]. SANS Internet Storm Center. 2021 [cited 17 December 2021]. Available from: <https://isc.sans.edu/forums/diary/Using+Yara+rules+with+Volatility/22950/>
- [7] Creating Process Dumps with ProcDump — Knowledge Base [Internet]. Kb.acronis.com. 2021 [cited 17 December 2021]. Available from: <https://kb.acronis.com/content/27931>
- [8] Shukla S, Misra M, Varshney G. Identification of Spoofed Emails by applying Email Forensics and Memory Forensics. 2020 the 10th International Conference on Communication and Network Security. 2020
- [9] Devendran V, Shahriar H, Clincy V. A Comparative Study of Email Forensic Tools. Journal of Information Security. 2015;06(02):111-117
- [10] Malik A. webmaill-logging-tool/webmail-logging-tool.py at main · abdolsabor/webmaill-logging-tool [Internet]. GitHub. 2021 [cited 17 December 2021]. Available from: <https://github.com/abdolsabor/webmaill-logging-tool>
- [11] Tariq Banday M. Techniques and Tools for Forensic Investigation of E-mail. International Journal of Network Security Its Applications. 2011;3(6):227-241
- [12] 52 Gmail Statistics That Show How Big It Actually Is In 2021 [Internet]. TechJury. 2021 [cited 24 November 2021]. Available from: <https://techjury.net/blog/gmailstatistics/gref>

- [13] Xu L, Wang L. Research on Extracting System Logged-In Password Forensically from Windows Memory Image File. 2013 Ninth International Conference on Computational Intelligence and Security. 2013
- [14] Preimesberger C. Cloud-based email services: Everything you need to know — ZDNet [Internet]. ZDNet. 2021 [cited 24 November 2021]. Available from: <https://www.zdnet.com/article/cloud-based-email-services-everythingyou-need-to-know/>
- [15] M. Hussain, A. Wahab, I. Batool and M. Arif, "Secure Password Transmission for Web Applications over Internet using Cryptography and Image Steganography", 2021.
- [16] Hassan N. Web Browser and E-mail Forensics. Digital Forensics Basics. 2019;:247- 289
- [17] Holland T. Webmail vs Email Clients: Which one should you choose? [Internet]. Ontrack. 2022 [cited 18 January 2022]. Available from: <https://www.ontrack.com/en-us/blog/webmail-vs-email-clients-one-choose>
- [18] Krill P. Python slithers toward top of language popularity index [Internet]. InfoWorld. 2022 [cited 18 January 2022]. Available from: <https://www.infoworld.com/article/3632862/python-slithers-toward-top-of-language-popularity-index.html>
- [19] Kumar S. Profile Memory Consumption of Python functions in a single line of code [Internet]. Medium. 2022 [cited 18 January 2022]. Available from: <https://towardsdatascience.com/profile-memory-consumption-of-python-functions-in-a-single-line-of-code-6403101db419>
- [20] Price D. 7 Reasons Why You Should Stop Using Desktop Email Clients [Internet]. MUO. 2022 [cited 18 January 2022]. Available from: <https://www.makeuseof.com/tag/stop-desktop-email-clients-stop-opinion/>
- [21] ProcDump - Windows Sysinternals [Internet]. Docs.microsoft.com. 2022 [cited 18 January 2022]. Available from: [https://docs.microsoft.com/en-us/sysinternals/downloads/procdump\](https://docs.microsoft.com/en-us/sysinternals/downloads/procdump)
- [22] Psutil module in Python - GeeksforGeeks [Internet]. GeeksforGeeks. 2022 [cited 18 January 2022]. Available from: <https://www.geeksforgeeks.org/psutil-module-in-python/>
- [23] SANS Internet Storm Center [Internet]. SANS Internet Storm Center. 2022 [cited 18 January 2022]. Available from: <https://isc.sans.edu/forums/diary/Using+Yara+rules+with+Volatility/22950/>
- [24] Strings - Windows Sysinternals [Internet]. Docs.microsoft.com. 2022 [cited 18 January 2022]. Available from: <https://docs.microsoft.com/en-us/sysinternals/downloads/strings>

- [25] Fahad E. Email Architecture, Gmail two Step Verification, SMTP POP3 IMAP [Internet]. Electronic Clinic. 2022 [cited 10 February 2022]. Available from: <https://www.electronicclinic.com/email-architecture-gmail-two-step-verification-smtp-pop3-imap/>
- [26] Ghafarian A. (2019) Capabilities of Email Forensic Tools. In: Arai K., Bhatia R., Kapoor S. (eds) Intelligent Computing. CompCom 2019. Advances in Intelligent Systems and Computing, vol 998. Springer, Cham. https://doi.org/10.1007/978-3-030-22868-2_38
- [27] A. Almomani, B. Gupta, S. Atawneh, A. Meulenberg, and E. Almomani, “A survey of phishing email filtering techniques,” IEEE Communications Surveys & Tutorials, vol. 15, no. 4, pp. 2070–2090, 2013
- [28] V. Ramanathan and H. Wechsler, “phishGILLNET phishing detection methodology using probabilistic latent semantic analysis, AdaBoost, and co-training,” EURASIP Journal of Information Security, vol. 2012, no. 1, p. 1, 2012.
- [29] Anargyros Xrysanthou and Ioannis Apostolakis. 2006. Network forensics: Problems and solutions. E-Democracy: Challenges of the Digital Era (2006), 307–318
- [30] M. Kumar, M. Hanumanthappa, and T. S. Kumar, “A countermeasure technique for email-spoofing,” International Journal of Advanced Research in Computer Science, vol. 4, no. 2, 2013.
- [31] S. Gupta, E. S. Pilli, P. Mishra, S. Pundir, and R. Joshi, “Forensic analysis of e-mail address spoofing,” in The 5th IEEE International Conference on the Next Generation Information Technology Summit (Confluence), 2014, pp. 898–904
- [32] A. Jayan and S. Dija, “Detection of spoofed mails,” in IEEE International Conference on Computational Intelligence and Computing Research (ICCCIC), 2015, pp. 1–4
- [33] T. Fowdur and L. Veerasoo, “An email application with active spoof monitoring and control,” in IEEE International Conference on Computer Communication and Informatics (ICCCI), 2016, pp. 1–6.
- [34] Lijuan Xu and Lianhai Wang. 2013. Research on extracting system logged-in password forensically from windows memory image file. In 2013 Ninth International Conference on Computational Intelligence and Security. IEEE, 716–720.
- [35] Stine K, Scholl M. E-mail Security: An Overview of Threats and Safeguards [Internet]. Library.ahima.org. 2022 [cited 23 January 2022]. Available from: <https://library.ahima.org/doc?oid=99319#.Ye0ncPhRXIU>

- [36] Fahad E. Email Architecture, Gmail two Step Verification, SMTP POP3 IMAP [Internet]. Electronic Clinic. 2022 [cited 23 January 2022]. Available from: <https://www.electronicclinic.com/email-architecture-gmail-two-step-verification-smtp-pop3-imap/>
- [37] Karim A, Azam S, Shanmugam B, Kannoorpatti K, Alazab M. A comprehensive survey for intelligent spam email detection. IEEE Access. 2019 Nov 20;7:168261-95.
- [38] Kambourakis G, Gil GD, Sanchez I. What email servers can tell to Johnny: an empirical study of provider-to-provider email security. IEEE Access. 2020 Jul 14;8:130066-81.
- [39] Guo H, Jin B, Qian W. Analysis of email header for forensics purpose. In 2013 International Conference on Communication Systems and Network Technologies 2013 Apr 6 (pp. 340-344). IEEE.
- [40] Paglierani J, Mabey M, Ahn GJ. Towards comprehensive and collaborative forensics on email evidence. In 9th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing 2013 Oct 20 (pp. 11-20). IEEE.
- [41] Mistry NR, Dahiya MS. Signature based volatile memory forensics: a detection based approach for analyzing sophisticated cyber attacks. International Journal of Information Technology. 2019 Sep;11(3):583-9.
- [42] Mabey M, Doupé A, Zhao Z, Ahn GJ. Challenges, opportunities and a framework for web environment forensics. In IFIP International Conference on Digital Forensics 2018 Jan 3 (pp. 11-33). Springer, Cham.
- [43] Trappen S. [Internet]. Sandratrappen.com. 2022 [cited 3 March 2022]. Available from: <https://sandratrappen.com/2018/06/23/research-methods-in-criminology/>
- [44] Parija SC, Kate V. Why write a scientific research paper. In Writing and publishing a scientific research paper 2017 (pp. 3-8). Springer, Singapore.

APPENDIX A – WINDOWS SYSINTERNAL AND VOLATILITY TOOLS COMMANDS

- Procdump Commands

`procdump.exe -t -ma -o " + str(browserppid) + " proc_dump.dmp`

-ma	Write a dump file with all process memory. The default dump format only includes thread and handle information.
------------	---

-o	Overwrite an existing dump file.
-----------	----------------------------------

-t	Write a dump when the process terminates.
-----------	---

- Strings commands:

`strings.exe -n 6 -nobanner " + filein + " > " + fileout + " & exit`

Parameter	Description
-a	Ascii-only search (Unicode and Ascii is default)
-b	Bytes of file to scan
-f	File offset at which to start scanning.
-o	Print offset in file string was located
-n	Minimum string length (default is 3)
-s	Recurse subdirectories
-u	Unicode-only search (Unicode and Ascii is default)
-nobanner	Do not display the startup banner and copyright message.

APPENDIX B – CODE FOR PYTHON FORENSIC LOGGING TOOL

```
import psutil as psutil
import subprocess
import time
import datetime
import datetime
from memory_profiler import profile

browsingprocess = ""
browserppid = ""
prcdump = ""
filein = "proc_dump.dmp"
fileout = "strfile.txt"
getmail = ""
logs = []

def generatestring():
    print("----- Generating Stringfile -----")
    gsf = "START /B strings.exe -n 6 -nobanner " + filein + " > " + fileout + " & exit"
    output = subprocess.check_output(gsf, shell=True)
    print("Generated Successfully")

def if_process_is_running_by_exename(exename):
    for proc in psutil.process_iter(['ppid', 'name']):
        # This will check if there exists any process running with executable name
        try:
            if proc.info['name'] == exename:
                b = proc.info['ppid']
                return b
        except (psutil.NoSuchProcess, psutil.AccessDenied, psutil.ZombieProcess):
            print("sss")
    return 0

@profile
def main():
    while (1):
        browserppid = if_process_is_running_by_exename("chrome.exe")
        if browserppid != 0:
            start_time = datetime.datetime.now()
            print("--- Initialize Dumping of {0}".format(browserppid))
```

```

prcdump = "START /B procdump.exe -t -ma -o " + str(browserppid) + " proc_dump.dmp & exit"
output = subprocess.check_output(prcdump, shell=True)
print(output)
print("Dump Complete")
generatestring()
with open(fileout) as infile:
    for line in infile:
        if line.startswith("Mail") and line.endswith("Outlook\n"):
            print("Email service: Outlook")
            print(line)
        if line.startswith("Inbox (") and line.endswith("Gmail\n"):
            print("Email service: Gmail")
            print(line)
            getmail = line.split(" - ")[1]
            print(getmail)
        if "\"newMessage\":true" in line:
            #print("Email service: Yahoo")
            print(line)
            logs.append(line)
        if "\"UpdateItemJsonRequest" in line:
            #print("Email service: Hotmail")
            print(line)
            logs.append(line)
        if "\"3\", \"2\"" in line:
            #print("Email service: Gmail")
            print(line)
            logs.append(line)
infile.close()

file1 = open("mallogs.txt", "a")
file1.writelines("\n\nEmail logs of" + getmail + "\n")
file1.writelines(logs)
file1.close()
print("logs updated")
end_time = datetime.datetime.now()
print(end_time - start_time)
else:
    print("NO browser is running")

if __name__ == "__main__":
    main()

```