SSL based Webmail Forensic Engine

Article *in* International Journal of Advanced Computer Science and Applications · January 2017

DOI: 10.14569/IJACSA.2017.080123

7 authors, including:

Manesh T
Prince Sattam bin Abdulaziz University
32 PUBLICATIONS **26** CITATIONS
SEE PROFILE

Mohemmed Sha
Prince Sattam bin Abdulaziz University
19 PUBLICATIONS **6** CITATIONS
SEE PROFILE

Mohamed Yacoab
Prince Sattam bin Abdulaziz University
7 PUBLICATIONS **16** CITATIONS
SEE PROFILE

Abraham Varghese
ASIET,Kalady
12 PUBLICATIONS **18** CITATIONS
SEE PROFILE

SSL based Webmail Forensic Engine

Manesh T, Abdalla A Alameen, Mohemmed Sha M, Mohamed Mustaq Ahmed A, Mohamed Yacoab M.Y.
Department of Computer Science
Prince Sattam Bin Abdulaziz University
P O Box.54, Saudi Arabia.Pin:11991

Bhadran V K
Resource Center for Cyber Forensics, Language Technology
Center for Development of Advanced Computing
A Scientific Society of the Ministry of Communications
Government of India, Vellayambalam, Trivandrum, Kerala,
Pin: 695033

Abraham Varghese
Department of Information Technology
Higher College of Technology,
Muscat

***Abstract*—In this era of information technology, email applications are the foremost and extensively used electronic communication technology. Emails are profusely used to exchange data and information using several frontend applications from various service providers by its users. Currently most of the email clients and service providers now moved to secured data communications using SSL or TLS security for their data exchanged. Cyber criminals and terrorists have started by means of this mode for exchanging their malicious information in their transactions. Forensic experts have to face greater difficulty and multiple challenges in tracing crucial forensic information from network packets as the communication is secured. These challenges might affect the digital forensic experts in procuring substantial evidences against such criminals from their working environments. This research work revels working background of SSL based webmail forensic engine, which decrypt respective communication or network session and also reconstruct the actual message contents of webmail applications. This digital forensic engine is compatible to work with in proxy servers and other computing environments and enables forensic reconstruction followed by analysis of webmail clients. Proposed forensic engine employs is a high-speed packet capturing hardware module, a sophisticated packet reformation algorithm; restores email header and messages from encrypted stream of SMTP and POP3 network sessions. Proposed forensic engine also support cyber investigation team with generated forensic report and prosecution of culprits by judiciary system of the specific country.**

***Keywords*—Forensics; Network Sessions; Packet Drop; Secure Data Aggregation; Sensor Nodes**

I. INTRODUCTION

With advent of email applications, this technological era has changed the style of communication in all the facets of current social and business environments. Such applications provide great handiness to users in exchanging multimedia contents cost effectively. People normally use these applications for their day-to-day transactions. From the time of inception of email and messaging applications until the introduction of SSL or TLS security over such communications, cyber criminals were very rarely using such communication platforms, as it was easy for forensic investigators to trace them with substitutable evidences. Now communications through Email applications are secured with use of SSL or TLS [2]. Though SSL encrypt the transaction to ensure security and privacy of communications, the process of encrypting messages brings following two serious challenges to forensic investigators frameworks. Firstly, it increases the burden of collecting and decrypting the network session with targeted email communications. Secondly, encryption reduces the chances of procuring accurate forensic details from the network packets as well as regenerating the contents of network packets [2]. Since the likelihood of being traced in using email messages, cyber intruders and criminals are in full swing in misusing this security infrastructure for their criminal communication and activities. This paper introduces a complete webmail forensic engine, which not only decrypts networks session with email transactions over SSL successfully but also traces available forensic details of communication effectively, which are sufficient to pin point malicious users and prosecute them. The proposed digital forensic engine works based on the concepts of network forensics.

Network forensic investigation is a process of regenerating a complete network session collected and processed as packets. This procedure sketches any network anomalies and traces all available network forensic details by analyzing the session of packets. There are two types of network forensic investigation techniques. One is offline, where network packets of session is captured followed by tracing anomalies in it. Second one, is online, where forensic activity is done during the live capturing of network session. Currently this proposed framework implemented for offline packet analysis as decryption process of network session is complicated in online packet analysis.

A. Challenges Addressed in Digital Forensics of Email

Popular email service providers including Gmail, Yahoo and Hotmail etc. have entirely moved to SSL based communication for ensuring increased security and privacy. Traditional forensic methods usually fail in tracing malicious email communication and regenerating email contents from encrypted stream of network session. When a particular email

Prince Sattam bin Abdulaziz University, Kingdom of Saudi Arabia

communication uses SSL, following are the major challenges addressed while analyzing, developing and implementing this proposed forensic engine for email communications.

1) Collect network session with SSL based email communication. High-speed packet capturing mechanism is needed to collect as much as network packets without any loss by wireless or wired methods.

2) Sufficient disk storage is a must to store network sessions in the form of PCAP files consists of more than 20-40 lacks of packets.

3) Parsing of SSL/TLS handshake mechanism to Identify and categorize SSL encrypted network sessions to construct its session keys, session certificate and its private key, session's cipher suite and its protocol details, random number details of SSL client and server, details of premaster secret and key exchange message of SSL client with its public key. Getting public and private key of SSL

4) Reordering the SSL packets of a particular network session followed by subsequent decryption of session using traced public and private keys.

5) Dissecting the packets in a network session to separate header and body parts of packets and to trace available forensic details.

6) Combine decrypted body of network packets according to its order to reconstruct the actual email message.

B. Contribution of the Proposed Framework

This paper primarily sketches the complicated steps involved in collecting, analyzing, decrypting and regenerating the contents of email from an encrypted (SSL) network stream as part of forensic investigation process. This paper describes a novel method to decrypt a particular encrypted stream by tracing cryptographic details successfully for regenerating all available email communications. Forensic investigator will analyze regenerated email message to trace malicious contents in it. A powerful packet rearrangement procedure designed especially to address encrypted network stream, which throttles proposed forensic approach. Proposed structural framework hosts a self-sufficient high-speed packet-capturing module of its own, which work independently or can associate with third party high-speed packet bagging software and hardware.

The various sections of this paper is ordered as follows. Section II summaries related work. Section III describes architecture of proposed webmail forensic engine. Section VI discusses results and major GUIs of the proposed framework. Section V outlines conclusion. Future enhancements are discussed in section VI followed by references.

II. Research Background

This section briefly summarizes significant and related works in the area of email forensics with greater influence for the development of proposed engine. These techniques analyze the content of email through various log file investigations and available communication framework for studying email behaviors and patterns.

Wang Wen Qi, et al.(2009) proposed an email forensic algorithm using fuzzy matching for forensic analysis of SMTP and HTTP protocol. Their contributions led us to understand the forensic processing of SMTP protocol. [1]

M. Tariq Banday (2011) proposed a forensic email architecture, where which describes roles of email actors and components which various protocols. This work enlightened us to know more about email headers and related metadata. [4]

Hong Guo, et al. (2013) discussed email header construction mechanism for forensic investigation process. Their work motivated us to capture various authentic procedures to reconstruct the email headers from network session. [6]

Justin Paglierani, et al. (2013) introduced a collaborative forensic for evidence collection of email which includes identification of non-oblivious artifacts of email and retrieval of data from ISP and analyses email evidences. Their focus helped us to know more about email identification patters. [7]

Lili Xie, et al. (2014) proposed an investigation and data analysis method for Foxmail client, which revels a participle algorithm for content retrieval of, email message and headers to trace suspicious users. This contribution helped us to understand structure of headers and its retrieval process. [9]

Sridhar Neralla, et al. (2014) proposed forensic approach for authenticated author of an email through parameter minimization using sylometric investigation technique. Their findings enhanced our understanding about structure of email and email parameters used for forensic analysis. [10]

Vamshee Krishna, et al (2015) presented a comparative study of available forensic tools for email, which gives an insight into details of its capabilities and scopes. [11]

Yanhua Liu, et al. (2015) proposed a set of solutions for forensic analysis of email contents for analyzing email traffic and email accounts using a centrality algorithm. Their solutions made us more informative about communication architecture of email. [12]

III. Proposed Digital Forensic Engine

The architecture of proposed forensic engine for SSL based webmail applications shown in Fig. 1, which implements forensic reconstruction of email exchanged though webmail clients. The online forensic analysis part of the architecture trace out SSL parameters to calculate session key while offline forensic analysis performs reconstruction after decrypting the session. This engine successfully decrypts the captured network session encrypted by SSL/TLS [2]. This engine intelligently traces cryptographic credentials between web client and BIG-IP or web server for a particular session of communication with help of its own certificate handle mechanism. This engine identifies and recreate contents of all email, which uses SMPTS and POP3S protocols. Proposed digital engine hosts a collection of forensic analysis and investigation of webmail communication through various stages to trace evidences against spiteful communications and efficiently pinpoints malicious users of particular networks. Following section explores significant modules and submodules of proposed digital forensic engine.

A. Forensic Source Pool

Forensic source pool is the major segment of this forensic engine, which collects network stream of packets through either wired or wireless or proxy server's Network Interface Cards (NIC). This segment of proposed engine has a high-speed packet capturing hardware from Napatech coupled with it. This hardware collects as many packets through its faster technology and avoids packet loss from any such attached NICs. This digital engine will perform email forensic investigation for individual computers as well as computer labs with proxy servers. This engine is well suited for proxy servers as it collects network packets to and from a suspected IP address or machine under proxy environment. Once packets are collected, this segment saves the packets in PCAP format for further forensic analysis in the forthcoming segments. [13]

B. Forensic Session Reconstruct

This segment performs core forensic accomplishments by means of reconstructing the suspected network stream from webmail users. This segment rebuilds all available emails communicated through SMTPS and POP3S protocols. This segment has following series of sub segments each of which perform well-defined technical forensic tasks. Following three immediate tasks are online forensic activities.

1) Webmail Filter

Webmail filter identifies source and destination IPs and port numbers present in the PCAP file forwarded to it from the Forensic Source Pool. On finishing this task, this sub segment will filter network packets with application layers protocols of webmail such as Simple Main Transfer Protocol Secure (SMTPS) and Post Office Protocol 3 Secure (POP3S). This segment also categorizes email based on HTTPS as well as webmail. As proposed engine targets SSL based Webmail, the engine will make a PCAP file which contain SSL based webmail network packets and forward to SMTPS and POP3S Email Analyze sub segments.

2) SMTPS Email Analyze & POP3S Email Analyze

The SMTPS Email Analyze and POP3S Email Analyze sub segments respectively filter SMTPS network packets with port number 465 and POP3S network packets with port number 995. Each of this segment then stores filtered network packets as another set of PCAP files for further parallel process. The main goal behind incorporating this parallel segment in this forensic engine is to trace out all the emails sent and received by a suspected user. This segment also saves PCAP files with respect to each email communication sessions between user or web client and BIG-IP or server.

*3) TLS Handshake Decode & Certificate Handl*e

This component is the vital part of proposed engine as it identifies TLS handshake process of specific sessions traced during the monitored suspected network activity. The ultimate goal of this segment is to capture certificates exchanged between web client and BIG-IP, significant cryptographic details, asymmetric keys used by web client and BIG-IP and finally tracing the symmetric key or session key used to encrypt the communication. It is very crucial to acquire the above cryptographic information from the TLS handshake mechanism before the web client and BIG-IP starts encrypted communication. [2]

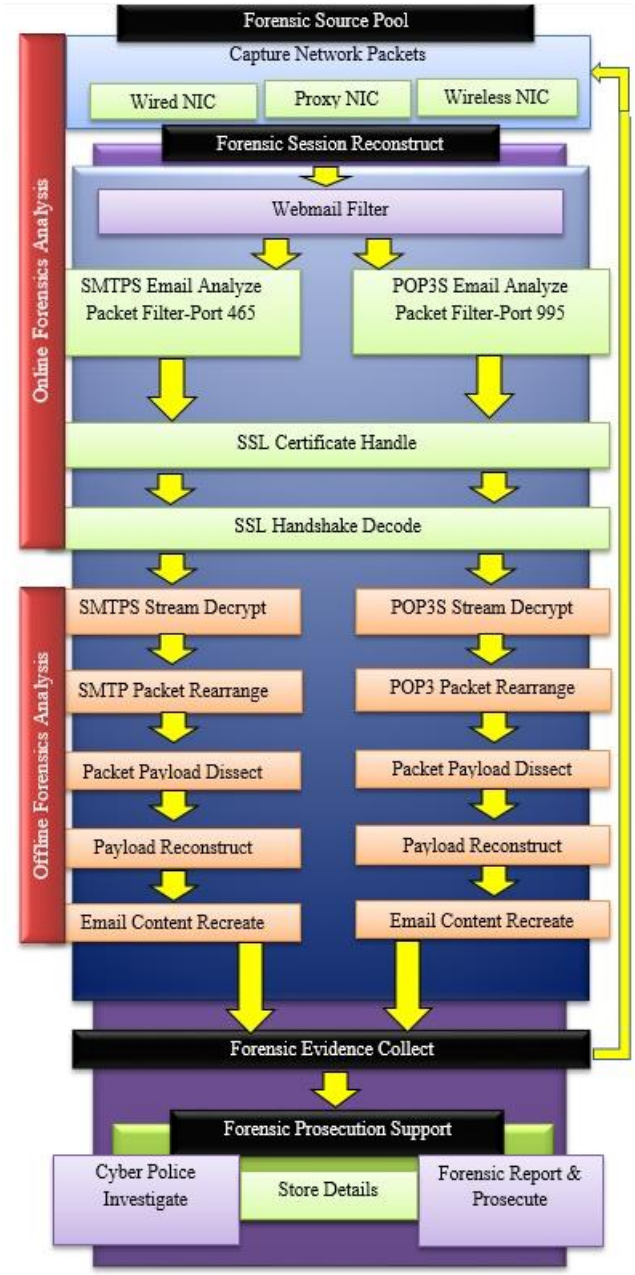

Fig. 1. Architecture of Webmail Forensic Engine

This segment effectively analysis the initial TLS handshake communication in each email establishment sessions, analyses each steps of transaction of TLS handshake mechanism as shown in the Fig 2 and traces session keys for each session. Following section demonstrates detailed working of TLS handshake mechanism and the steps adopted by this segment to gain cryptographic session keys or Master Secret for subsequent decryption of network stream. For identifying the network packets with TLS handshake information, this segment carefully filters and collects initial communication packets of each session between packet tags

SYN and FYN where SYN represents communication start packet and FYN represents communication end packet. To understand how to gain shared session key using TLS certificate handle segment of the proposed engine, following section and Fig 2 provides a brief breakdown of TLS handshake mechanism.

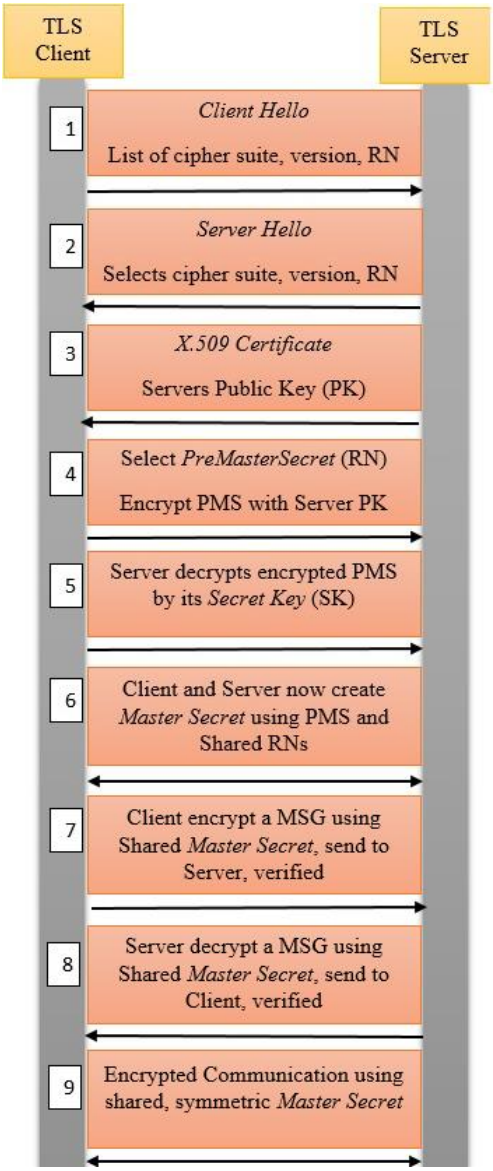

Fig. 2. TLS Handshake Mechanism

As shown in figure 2, TLS client indicates the web client or web browser of the user where TLS server indicates BIG-IP or web server. Initially, TLS handshake mechanism adopts asymmetric key cryptographic methods to exchange a shared key or master key effectively with the public and private keys of the web server. Eventually, TLS handshake mechanism changes to symmetric key cryptography using shared master secret as symmetric encryption is significantly faster and more efficient compared to asymmetric key cryptography. This TLS protocol maintain a state full connection between client and server. Initially as shown in the step 1, the client sends a *Client Hello* message to the TLS server indicating that the client is willing to start an encrypted communication with the server. In this message, details of TLS protocol version, a set of options that the client is willing use in order to communicate with server named as cipher suite containing combinations of cryptographic methods & specifications, compression methods etc. Client also select a 32-byte random number *RN* and send to the server. On receiving a *Client Hello* message, as shown in step 2, server also generate a 32-byte random number RN based on date and time stamp and reply to client by making choice on selected cipher suite from a list of ciphers sent from client hello session ID, TLS protocol version and compression methods. So at the end of hello packets from both the ends, client and server exchanges RN generated at both ends along with attribute for further communication.

Now in step 3, server sends a X.509 certificate with its public key. This X.509 certificate revels the identity of the certificate issuer, version, serial number, algorithm details, issuer name, validity period and other significant details of PKI. At the client side, on receiving server certificate, client verifies name, validation date etc. to ensure server identity. At the end of this phase, client has RN generated at both ends and public key of the server. The server also has both RNs and its pre-existing public certificate. Until now, everything is transmitted in plain text and is vulnerable to sniffing through network packets. This segment of proposed engine brilliantly sniffs all such details from network packets with help of JPCAP library and OpenSSL. Once the negotiation terms are decided and identity of both sides are verified, as shown in step 4, the client generate one more random number RN as PreMasterSecret (PMS). The client encrypts this generated PMU with the public key of the server and sends to TLS server.[14]

In step 5, the server decrypts the encrypted PMU as it hold its private key to obtain original PMS generated at the client side. At the end of this communication, both client and server hold same ingredients like RNs, PMS and public certificate of the server. With all these inputs to algorithm and negotiated attributes, both sides generate same *Master Secret* (MS). This happens in step 6. Further communication makes use of this MS for encrypting data. However, before moving further, both sides need to verify whether they have same *Master Secret* generated at either end. To confirm generation of same MS at other end, client sends some data encrypted with MS generated at his end along with *end SSL handshake* packet as shown in step 7. If the server also generated same MS, server will be able to decrypt the message with MS at his end. As in step 8, to reconfirm for the sake of integrity of connection, server re-encrypts the data using his MS and send back to client with *end SSL handshake* packet. The client and server will encrypt further communication using the shared Master Secret. [14]

Proposed segment of this forensic engine identifies TLS handshake packets and records cipher used, its key length, protocol version along with compression methods by dissecting network packets using JPCAP and OpenSSL software libraries. As discussed above, ultimate aim of this segment is to calculate 48-byte *Master Secret* (MS) or session key for encrypting the inbound traffic. From the TLS handshake, it is clear that Master Secret is crafted using RNs

of client and server, PMS with negotiated cipher suites using Pseudo Random Function (PRF). This segment wisely acquires X.509 certificate from the server with help of specially designed certificate handle mechanism. It can also detect the certificate format such as pkcs#7, pkcs#12 and convert it to hex form for decoding process. Once this segment identifies the certificate format, it parses the contents inside it by searching the label text in the packet as *BEGIN PUBLIC KEY* and *END PUBLIC KEY*. TLS handshake decode segment extracts the text between these two word pairs and saves in the form of plain text or hex format.

TLS handshake decode segment has a collection of well programmed key exchange algorithms such as Diffie-Hellman, Elliptic Curve Diffie-Hellman, RSA which are in the list of ciphers exchanged between client and server. This segment traces *Client Hello*, *Server Hello* and *Key Exchange* network packets from the traffic. As shown in step 1 of fig 2, TLS handshake decode segment parses the *Client Hello* message and traces details of protocol version, session ID, random number RN of the client with GMT timestamp, list of ciphers proposed by client which define encryption and hashing methods etc. Latest version of TLS 1.3 make use of RSA along with 128/256 SHA, RC4 as selected by the TLS server. From the step 2 of the Fig 2, TLS handshake decode extracts server random number, protocol versions and confirms selected cipher suites. From the step 3, TLS handshake decode segment extracts public key present in the X.509 certificate as mentioned earlier, and identifies size of PreMasterSecret created which is normally 128 bytes in size with RSA key size of 1024 bit keys. [14]

Once certificate reaches client from the server, it also receives key exchange messages, which support both RSA, or Diffie Hellman based key exchange methods. Proposed segment is designed to work with both types of key exchange methods to get public key of the server and encrypted PreMasterSecret. Step 4 & 5 of TLS handshake mechanism calculates shared MAC key or *Master Secret* for encrypting SSL traffic sessions. Pseudo Random Function (PRF) on server side generates *Master Secret* by using PreMasterSecret, random numbers RNs of client and server. In order to obtain the Master Secret, this segment is in need of decrypted PreMasterSecret.

The RNs of client and server is now available to this segment from the TLS handshake decode sub segment, but exact PreMasterSecret is not available to this sub segment. In order to avail this PMS, this segment requires the private key of the server. From the X.509 certificate with pkcs#12 format, X.509 TLS certificate handle sub segment as shown in Fig 3 raises private key of the message to perform decryption of the PMS utilizing recorded cipher suite negotiations to attain unique PMS. This sub segment works a proxy server between suspected user machine and the web server. This segment thus redirects all the inbound traffic between client and server for a particular email session through it. The X.509 Certificate Handle now will access the genuine X.509 certificate from the BIG-IP or TLS server where it returns its own certificate to suspected user machine and creates two distinct TLS connection lines.

One is from web client to X.509 Certificate Handle segment and second one is from X.509 Certificate Handle to TLS server. Both certificates are in the form of pkcs#12 format.

Since the certificate handle segment acts in the middle of network traffic between client and server, the root X.509 certificate is made available to the suspected user machine. In each email communication sessions, this segment creates its own dynamically signed certificate for the server with its private key.

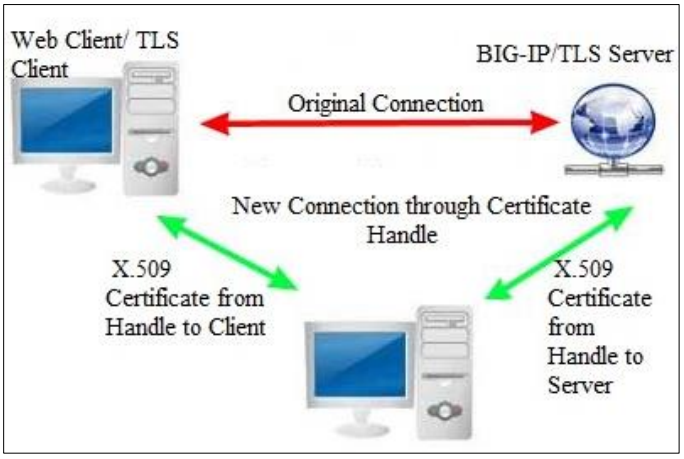

Fig. 3. TLS Certificate Handle

OpenSSL module associated with this segment now traces the private key of the server from X.509 certificate. No SSL error messages are existing as current segment establishes a trusted network traffic between web browser and server.

After acquiring private key of the server, this sub segment decrypts the encrypted PreMasterSecret to get actual PreMasterSecret. Thus, segment calculates *Master Secret* (MS) with appropriate pseudo random function applied to PreMasterSecret, Random Number RN of web client and BIG-IP or web server. This *Master Secret* acts as session key for the encrypted channel between web client and BIG-IP. This segment successfully traces this session key for decrypting the stored network sessions in the form of PCAP files.

4) SMTPS Stream Decrypt & POP3 Stream Decrypt

These parallel sub segments decrypt respective stored network communication sessions. The SMTPS Stream Decrypt and POP3S Stream Decrypt sub segments decrypt network packets stored as PCAP files with SMPTPS and POP3 network sessions using the traced session key and further creates a PCAP file with a stream of unencrypted packets.[14]

5) SMTP Packet Rearrange & POP3 Packet Rearrange

These parallel sub segments rearrange the corresponding SMTP and POP3 network sessions based on its sequence number and timestamps intelligently by analyzing duplicate as well as retransmitted network packets. Following section provides an insight into various processes involved packet assembly algorithm as shown in fig 4.

6) Packet Payload Dissect

This sub segment dissects each network packet of both SMTP and POP3 network streams and separates packet header and payload sections and stores as another temporary PCAP files for further processing. [16]

OFFLINE FORENSIC PACKET REFORMATION ALGORITHM

Algorithm 1: SMTP and POP3 Time Stamp

Input: Pcap file, Source IP, Destination IP, Source Port, Destination port
Output: Pcap file of decrypted SMTP Session

```
Initialize next=1, seq=0;
While packet!= null do
        Read the packet from the Pcap file SMTP and POP3 separately
        If packet= EOF, Filter the packets based on IP and Ports end if
          While packet! =null do
                new.packet=fetch. Next packet, packet = new.packet
                Affix a new time stamp to packet header using Jpcap.packet.header.access.
         end while
 end while
```

Algorithm 2: Separate Retransmitted Packets

Input: Pcap file, Source IP, Destination IP, Source Port, Destination port
Output: Pcap file of retransmitted packets

```
Initialize next=1, seq=0;
While packet! = null do
        Read the packet from the Pcap file
        If packet= EOF,  Filter the packets based on IP and Ports end if
        If syn flag then seq=packet. Sequence, Seq2=Packet. Sequence-seq end if
         While  next==1 do
           current=seq2, next =current + Packet.datalength end while
       If seq2>=next, next=seq2+Packet.datalength
          else Write that packet to the temp file//retransmitted packet end if
 end while
```

Algorithm 3: Separate Duplicate Packet

Input: Pcap file, Source IP, Destination IP, Source Port, Destination port
Output: Pcap file of duplicate packets

```
Initialize next=1, seq=0;
While packet!= null do
        Read the packet from the Pcap file
        If packet= EOF,  Filter the packets based on IP and Ports end if
        If syn flag then seq=packet. Sequence, Seq2=Packet. Sequence-seq end if
         While  next==1 do
            current = packet. fetch(PCAP).
            If current.datalength=next.datalength
                  If current.data=next.data, delete next packet   end if end if
          current=seq2, next =current + Packet.datalength end while
       If seq2>=next, next=seq2+Packet.datalength
          else Write that packet to the temp file//retransmitted packet end if
 end while
```

Algorithm 4: Packet Payload Reconstruction

Input: Pcap file, Source IP, Destination IP, Source Port, Destination port
Output: Reordered and reconstructed pcap file,

```
Initialize next=0, seq=0, flag2=0, contentcalc=0;
While packet!= null do
   Read the packet from the Pcap file
   if packet= EOF or null then exit from the loop
     if packe.source ip==source_ip and packet.source port=source port
       if syn flag is set seq=Packet. Sequence, seq_relative=Packet. Sequence-seq end if, end if
      if next=0, Write reordered file, next=sequence relative + packet.datalength  end if
      if seq_relative= next do following two steps, Write packet to the reordered file,
         next=sequence relative + packet.datalength  end if
      if  seq relative >next, Read each packet (Packet1) from the temporary file until the
        last packet is reached
        if Packet1.sequence-seq=next, write the packet to the reordered file,
          next=next+packet1.datalength end if
    end if,  end while
```

7) Payload Reconstruct

This sub segment reconstructs the actual network session by combining reassembled packet payloads for both SMTP and POP3 sessions and stored separately. [17]

8) Email Content Regenerate

This sub segment displays the regenerated header and body content of the email communication in its console

9) Offline packet Reformation Algorithm

The packet reformation algorithm is a set of four sub algorithms, which ultimately reorganize and reconstruct the SMTP and POP3 network communication sessions after decryption. This algorithm intelligently separates duplicate and retransmitted TCP packets in the stored network sessions using time stamp method. Initially, foremost algorithm affixes a time stamp using a hex value to the packet header in order to identify its arrival to the session along with sequence number. In this algorithm, processes packets using "Jpcap.packet. header.access" from JPCAP library [17]. The second algorithm separates all retransmitted packets. The packet variable "current" represents current packer under processing. It compares its sequence number along with sequence number of the next packet considering its time stamp and data length. The algorithm identifies retransmitted packets by comparing current and expected sequence numbers along with available timestamp and packet data length. The third algorithm identifies duplicate packets in a session by comparing its sequence numbers along with hex values of packet payload. The second and third algorithm work together to reorganize the stored SMTP and POP3 sessions. Finally, the fourth algorithm reforms the actual network session of SMTP and POP3 email communications by combining the packet's payloads.

C. Forensic Evidence Collect

The Forensic Evidence Collect segment of the proposed framework preserves all the data regenerated by packet reformation algorithm. It also traces all available IP and port numbers involved in the malicious or suspected session using a separated log file process. It stores regenerated content of email communication using hex value notation as well as plan text format. This section has a loop back process to forensic source pool to conduct forensic processing for particular session more than once to refine the results.

D. Forensic prosecution Support

The Forensic Prosecution Support segment plays significant role in presenting the digital forensic evidences in a systematic way. It conducts a preliminary evidence credibility test to identify best output of regenerated network sessions, which is rich in forensic information. It further categories the evidences according to its quality and credibility and further report to cybercrime investigation team. The segment finally incorporates the investigation report of the cyber police with its digital forensic report to the court as part of prosecution formalities.

IV. GUIs and Results

This section presents important User Interfaces such as Email Header Display and Email Message Display Consoles developed as part of the proposed forensic framework. The Email Header Display Console displays the content of a decrypted email communication involved in the SMTPS network communication session. The Email Message Display Console displays the content of the email communication.

From the Email Header Display Console, Investigator gets clear information regarding various parameters fetched from respective TCP packet of Email stream after subsequent encryption. The date and time indicates communication period. This console also traces our significant IP address and other forensic details are separately displayed. This header acts as credible evidence against any malicious communications. POP3S analyze button displays corresponding decrypted communications POP3 network stream.

The Email Message Display Console indicates forensically regenerated email message content from respective TCP packets after decryption of network stream. It clearly projects the email subject part with email ID of sender and receiver. Proposed engine fetches IP address of network packets from which sender's email message is traced. This acts as strong evidence against the malicious user. This console can also display decrypted stream in the form of Hex values.

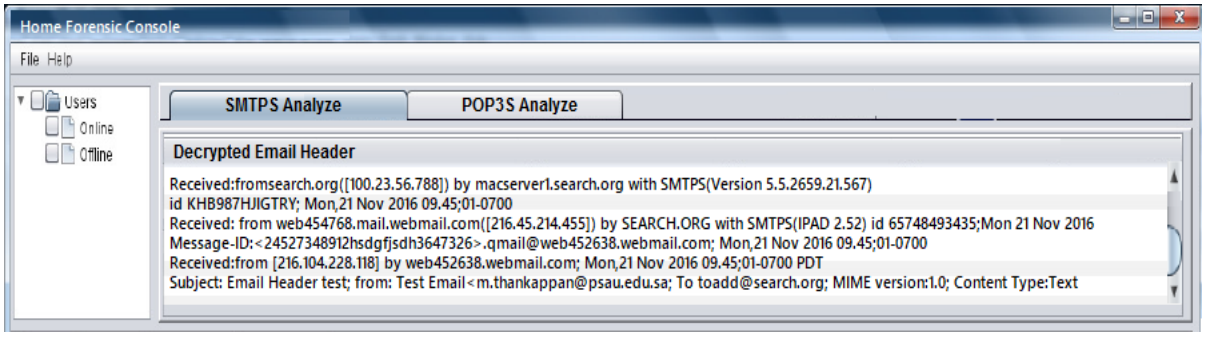

Fig. 4. Email Header Display Console

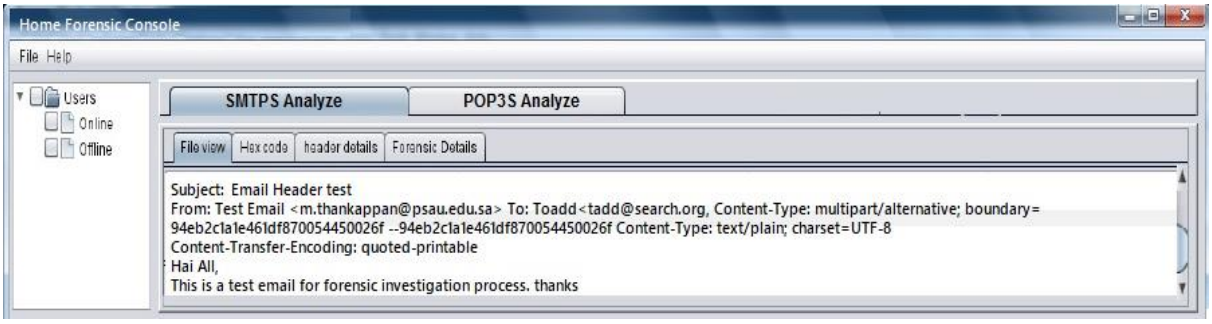

Fig. 5. Email Message Display Console

A. Computing Environment

Fig. 6. Computing Environment

The Fig 6 shows the computing environment for proposed SSL based forensic engine for webmail communications. The yellow colored labels show the different network points, which deploys proposed framework for tracing malicious activities when reported. Standalone PCs or proxy or network servers effectively use this framework to collect packets followed by its forensic reconstruction. Once network packets are collected, filtering of packets are done to get needed packets that follow protocols from email communication,

B. Usefulness of the Proposed Forensic Engine

This framework identifies the network packets associated with Webmail communications though their well-defined port numbers, categorize them, and prepare such packets for subsequent reconstruction of the network session. Currently the anticipated framework retrieves the header details and email contents, which uses SMTP, POP3 protocols, though encrypted channel, and pinpoint the malicious user's credentials like IP addresses and port numbers and regenerated webmail contents etc. It further helps police investigation process and prosecution by court of law.

V. Conclusion

The outlined digital forensic engine for SSL based Webmail application is successful in tracing significant forensic details from its encrypted network sessions. This engine framework primarily calculates the session key used for SSL connections. The sketched framework fetches all SSL parameters during online communication through the suspicious user's device. While tracing the SSL parameters, the framework also stores all the TCP packets in the form of PCAP files for further offline processing. Using the session key of SSL channel, framework decrypts TCP packet stream to trace out email header and message details.

VI. Future Enhancements

Currently the framework works only with SSL based webmail communications. It needs many refinements in tracing all available email communications from the decrypted session. Now, the authors extend the framework to decrypt the SSL based email communications for other popular email clients such as Gmail and Yahoo. The authors also extend the work towards decryption of IMAP as well as Instant messaging applications.

References

[1] Wang WenQi, Liu WeiGuang, (2009) "The Research on Email Forensic Based Network" IEEE International Conference on Information Science and Engineering, Dec 2009

[2] Hai Xin,Duan,(2010) "SSL-Do: A Rootkit of Network Based SSL and TLS Traffic Decryptor", IEEE CTC Workshop, July 2010.

[3] Wang Hui, (2009) "Network Data Packet Capture and Protocol Analysis on Jpcap Based". Proceedings of IEEE International Conference on Information Management and Industrial Engineering, vol 3,No.4, May 2009

[4] M.Tariq Banday, (2011) "Techniques and Tools for Forensic Investigation of Email" International journal of Security & its Applications(IJNSA),Vol.3,No.6 Nov 2011.

[5] Ali M, (2012)," Digital Forensics Best Practices and Managerial Implications", International Conference on Computational Intelligence, Communication Networks, Jul 2012

[6] Hong Guo, Bo Jin, Wei Qian (2013) "Analysis of Email Headers for Forensic Purpose" IEEE International Conference on Network Systems and communication Technologies, April 2013

[7] Justin. Paglieranim, Mike. Mabey, Gail-Joon Ahn "Towards comprehensive and collaborative forensics on email evidence" IEEE International Conference on Collaborative Computing, Networking, Applications & Worksharing, Oct 2013.

[8] Sant Paul (2013)," The Forensics Edge Management System ", IEEE International Conference on Ubiquitous Intelligence & Computing, Jun 2013.

[9] Lili Xie, Guolong Chen(2014) "A Forensic Tool of Foxmail Client" IEEE International Conference on Systems and Informatics, Nov 2014.

[10] Sridhar Neralla, D. Lalitha. Bhaskar,(2014) "A Stylometric Investigation Tool for Authorship Attribution in E-Mail Forensics" Proceedings of the 48th Annual Convention of Computer Society of India, Advances in Intelligent Systems and Computing, Springer, pp 543-549, Oct 2014

[11] Vamshee, Krishna, Devendran, Hossain S, Victor, Clincy, "A Comparative Study of Email Forensic Tools" Journal of Information Security (ARES), Vol.6,No.3 April 2015

[12] Lili Xie, Yanhua Liu, Guolong Chen (2015), "A forensic analysis solution of the email network based on email contents", IEEE

International Communication Conference on Fuzzy Systems & knowledge Discovery, Aug. 2015

[13] Manesh T, B Brijith, Mahendra Prathap Singh, "An Improved Approach towards Network Forensic Investigation of HTTP and FTP Protocols", International conference on Advances in Parallel Distributed Computing Communications in Computer and Information Science, Springer, Vol.1,July 2011.

[14] Manesh T, Brijith B, Braguram T(2013) "Network Forensic Investigation of HTTPS Protocol " International Journal of Modern Engineering Research, Vol. 3, Issue. 5, Sep - Oct. 2013.

[15] Manesh T, M Mohammed Sha, K Vivekanandan, (2014) "Forensic investigation framework for P2P protocol " IEEE International conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), pp-256- 264, July 2014.

[16] M Mohemmed Sha, T Manesh, (2015) "Forensic Framework for Skype Communication" International conference on Advances in Intelligent Systems and Computing, Springer July 2015.

[17] M Mohemmed Sha, T Manesh, (2016) "VoIP Forensic Analyzer" International Journal of Advanced Computer Science and Applications, Vol.1, No.1 Feb 2016.

[18] Agarwal S (2014) " A Hybrid approach for spam filtering using support vector machine and artificial immune system", IEEE International Conference on networks and Soft computing, Jun 2014

[19] GuoLong Chen, Lili Xie, "An Email Forensic Analysis Method Based on Social network Analysis", IEEE, International Conference on Cloud Computing and Big Data, July 2014

[20] Gori. Mohammed, Mohammed. Mohideen (2014) "E Mail Phising-An Open threat to Everyone", International Journal of Scientific and Research Publications, Vol.4,No.2,Feb 2014.

[21] Ravi Tomar (2014), "Taxonamy of Email Security Protocol", International Journal of Innovative Research in Computer and Communication Engineering, Vol.2,No.4, April 2014.

[22] Husak, M, Cermak, M ,; Jirsik, T ,; Celeda, P,(2015).," Network-based HTTPS Client Identification Using SSL/TLS Fingerprinting", IEEE International Conference On Availability, Reliability And Security, Aug 2015.

[23] Eldewahi, AEW, Sharfi, TMH, Mansor, AA, Mohamed, NAF, Alwahbani, SMH (2015), "SSL/TLS Attacks: Analysis and Evaluation" IEEE International Conference On Computing, Control, Networking, Electronics And Embedded Systems Engineering, Sept 2015.

[24] Husak, M, Cermak, M, Jirsik, T, Celeda, P (2016) " HTTPS traffic analysis and client identification using passive SSL/ TLS fingerprinting", Eurasip Journal On Information Security, Vol.1, No.14, Feb 2016.

[25] Rolf Oppliger, "SSL and TLS Theory and Practice", ARTECH HOUSE, ISBN-13 978-1-59693-447-4, 2009.