Netfox Detective: A tool for advanced network forensics analysis

J. Pluskal, P. Matoušek, O. Ryšavý, M. Kmeť, V. Veselý, F. Karpíšek, M. Vymlátil

{ipluskal,matousp,rysavy,ikmet,ivesely}@fit.vutbr.cz, {xkarpi03,xvymla01}@stud.fit.vutbr.cz

Faculty of Information Technology
Brno University of Technology
Brno, Czech Republic

Abstract

Network forensics is a process of capturing, collecting and analysing network data for the purposes of information gathering, legal evidence, or intrusion detection. The new generation internet opens novel opportunities for cybercrime activities and security incidents using network applications. Security administrators and LEA (Law Enforcement Agency) officers are challenged to employ advanced tools and techniques in order to detect unlawful or unauthorized activities. In case of serious suspicion of crime activity, network forensics tools and techniques are used to find out legal evidences in a captured network communication that prove or disprove suspect's participation on that activity.

Today, there are various commercial or free tools for network forensics analysis available, e.g., Wireshark, Network Miner, NetWitness, Xplico, NetIntercept, or PacketScan. Many of these tools lack the ability of successful reconstruction of communication when using incomplete, duplicated or corrupted input data. Investigators also require an advanced automatic processing of application data that helps them to see real contents of conversation that include chats, VoIP talks, file transmission, email exchange etc.

Our research is focused on design and implementation of a modular framework for network forensics with advanced possibilities of application reconstruction. The proposed architecture consists of (i) input packet processing, (ii) an advanced reconstruction of L7 conversations, and (iii) application-based analysis and presentation of L7 conversations. Our approach employs various advanced reconstruction techniques and heuristics that enable to work even with corrupted or incomplete data, e.g. one-directional flows, missing synchronization, unbounded conversations, etc.

The proposed framework was implemented in a tool Netfox Detective developed by our research group. This paper shows its architecture from functional and logical point of view and its application on reconstruction of web mail traffic, VoIP and RTP transmissions.

Keywords: network forensics, forensic tools, network traffic analysis, Web mail, SIP, RTP

1 Introduction

Network forensics is a discipline that deals with obtaining and analysing digital evidences from network sources. It is an extended phase of network security where the main goal of network forensics is to track and analyse network data in order to detect security incidents and present evidences of these incidents to security administrators or investigators. Network forensics use different supporting tools and devices that (i) obtain and collect data (firewalls, IDS systems, capturing tools), and (ii) process, analyse and reconstruct captured data. Network forensic tools are mostly used by security administrators and LEA officers that try to search network data for legal evidences of unlawful behaviour. The aim of the analysis is to establish high level facts such as attribution, intent, identity, timelines and other information which may be relevant to the security incident.

Tools for network forensics can be classified into two main groups: Network Forensic Analysis Tools (NAFTs) that allow administrators to monitor network, gather all information about the traffic and assist in network crime investigation, and Network Security and Monitoring (NSM) tools that are focused more on network monitoring and management. There is a wide range of commercial and open-source NFATs and NSM tools [1]. The primary motivation behind NSM tools is network security from perspective of system administration. NSM tools are very useful in processing large amount of data in short time with limited functionality concerning application protocol dissection. NMS tools include (i) IDS/IPS systems for detection or prevention of malicious activity on network, (ii) statistical tools used for data retention to store meta-information about the traffic, (iii) packet capture and analyses tools that capture communication on local networks and analysing it. The most common NSMs focused on packet capturing and analyses are Wireshark, TCPdump, or Microsoft Network Monitor. These tools are also used for basic network forensic analysis. However, they are mostly oriented on simple analysis of internet and transport layers of TCP/IP model. Some of them even contain an application layer protocol dissector, but the provided information is a context-free parsed internal protocol structure.

In this work, we focus on NFATs. NFATs offer a wide range of research challenges in domain of analysis and reconstruction of captured traffic. Research challenges cover (i) network stream reassembling that include detection of TCP/UDP streams, dealing with out of sequence data, missing or corrupted packets, timestamps overflow, combing streams into bi-directional conversations etc. [2]; (ii) advanced identification of L7 applications using AI techniques, data mining or statistical methods [3]; (iii) processing and analysis of L7 application using application dissectors, (iv) identification and statistical processing of encrypted or tunnelled traffic, (v) efficient storage of big network data with parallel computation, (vi) correlation of different input data, etc.

This paper describes architecture and implementation of a network forensic tool Netfox Detective developed by our team in frame of security research supported by Ministry of Interior of the Czech Republic. The tool is designed for advanced reconstruction and analysis of captured network data with focus on emails (including web mails), HTTP reconstruction and intelligent detection and reconstruction of Voice over IP. Our framework combines advanced techniques and heuristics for assembling captured data, identification of L7 traffic, reconstruction of original conversations, and presentation of L7 objects to an investigator. The proposed framework uses modular programming environment with well-defined API so new modules (application dissectors, processing engines) can be added without a need to re-build the entire application. It also supports parallel processing with efficient data storage.

2 Related Work

Network forensics was formally defined in 2001 on the First Digital Forensic Research Workshop [4] where also major issues were identified: (i) time, i.e., synchronization and integrity of data and time associated with events being analysed; (ii) performance, i.e., speed and effectiveness of processing and computation; (iii) complexity, i.e., general environment with multiple operating systems, network devices, different data formats, and (iv) collection, i.e., who will collect data, when, and what to be collected?

After a decade of innovations and research, general process model for the network forensic analysis has been introduced [1]. General model was composed of blocks with separated functions and was divided into two layers: (i) *lower layer* that included preparation, detection, collection, and preservation; and (ii) *upper layer* containing examination, analysis, investigation, and presentation.

Overview of different frameworks based on distributed systems, soft computing, honeypots, graphs, formal methods, or aggregation can be found in [1]. In that paper, Pilli et al. present a survey of current network forensic frameworks. Most of discussed frameworks were designed to as research tools to prove advanced approaches and techniques in the area of network forensics. Our tool presented in this paper employs some of these ideas but its development is driven by practical usability and deployment.

On the field of free tools, there are several applications that were observed. NetWitness filters captured traffic by processing frames and creating a lexicon of identifiers found in different L3-L7 layers, e.g., IP addresses, email addresses, URIs, etc. An investigator searches this lexicon to filter interesting captured content. The result can be stored as filtered captured traffic or analysed by another NFAT. Another popular tool is NetworkMiner[1] developed by Erik Hjelmvik, an author of Statistical Protocol Identifier (SPID) algorithm for application protocol detection [3]. NetworkMiner processes captured or online communication with an instantaneous analysis of application protocol. The analysed content is grouped into categories based on its characteristics, e.g. images, messages, credentials, files, frames, hosts, sessions. The tool lacks detailed views of captured data and is not able to backtrack objects to its original representation in captured packets. Xplico[2] is an open source NFAT platform composed of functional blocks. Application data are prepared by traffic decoder and then processed by manipulators. Xplico supports various application protocols, e.g., HTTP, SIP, IMAP/POP3/SMTP, FTP, etc. with ability to provide congruent investigation for multiple investigators at once. The tool provides a user interface via a web browser which is simple to use, but it is not suitable for advance analysis, e.g., advance filtering, getting data integrity statistics, etc. Nevertheless, Xplico is the most advanced open source NFAT available.

[1] See http://www.netresec.com/?page=NetworkMiner.
[2] See http://www.xplico.org.

3 Netfox Detective Architecture

By testing available NFATs we discovered that none of these tools is sufficient to accurately extract incomplete network data. In addition, advanced processing of application protocols with user-friendly presentation was mostly missing and limited large deployment of these tools for investigators. To overcome these limitations, a new network forensics framework was proposed with advanced parsing features.

Netfox (NETwork FOrensiCS) Detective is a NFAT framework operating upon four upper layers of generic process model of NFATs as described in [1]. The tool processes input network data stored in different PCAP formats[3] using a generic algorithm that respects L2-L7 encapsulation of PDUs. As described in [2], advanced heuristics is employed to extract maximal amount of information from PDU headers.

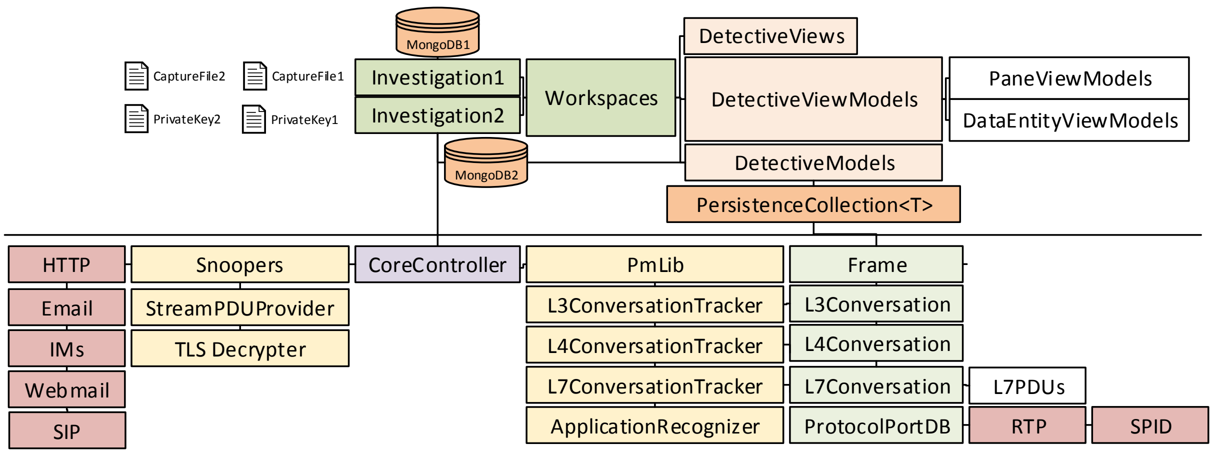

Fig. 1: Functional architecture and data model of Netfox Detective.

Netfox Detective has been designed to be used on Windows 7+ platform. To ensure proper behaviour and modular architecture as shown in Fig.1, the Model-View-Viewmodel (MVVM)[4] design pattern has been chosen with asynchronous programing provided by .NET 4.5.2 and C# 6. When launching the tool, a new workspace is created or a recently used workspace is re-loaded. The workspace represents a directory structure in a file system where all data related to the workspace are stored. The workspace contains one or more investigations that can consists of one or more PCAP files, see Fig. 2. Data processing is controlled by *Core Controller* that communicates with *PmLib* module, *Conversation trackers* and *Snoopers*. *Application Recognizers* use different techniques to identify L7 applications, see below. Application analysis and presentation of the results is implemented using L7 *Snoopers* over HTTP, Emails, IMs, Web mails, or SIP.

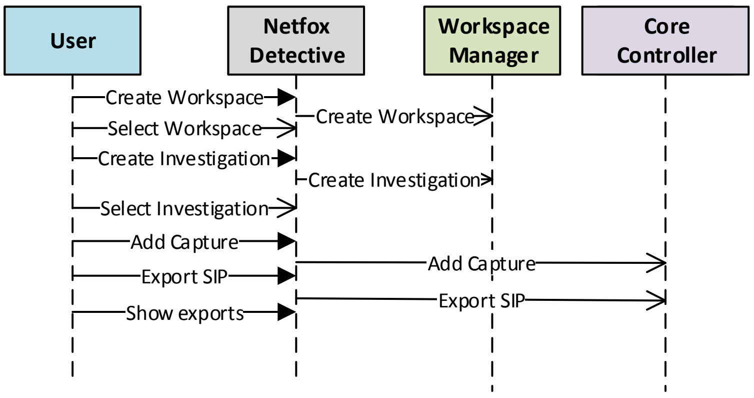

Fig. 2: Logical interactions of presentation model.

[3] E.g., see LibPCAP and PCAP NG at https://wiki.wireshark.org/Development/LibpcapFileFormat (PcapNg), or MS Network Monitor PCAP at http://blogs.technet.com/b/netmon/p/downloads.aspx.
[4] See https://msdn.microsoft.com/en-us/library/hh848246.aspx.

NFATs usually require flexible design with extensibility that allows addition of new features and propagation of these updates throughout the modular architecture without changing internal data structures. This can be implemented using document-oriented database that processes dynamic semi-structured data types in contrast to pre-defined types in relational databases where relations between data are fixed and must be defined in advance. Netfox Detective framework employs document-oriented database system MongoDB[5]. This approach ensures persistence across the entire framework using only one implementation for each data model. Basic data models *Workspace*, *Investigation*, *Capture*, and *InvestigationInfo* are listed in Fig. 3.

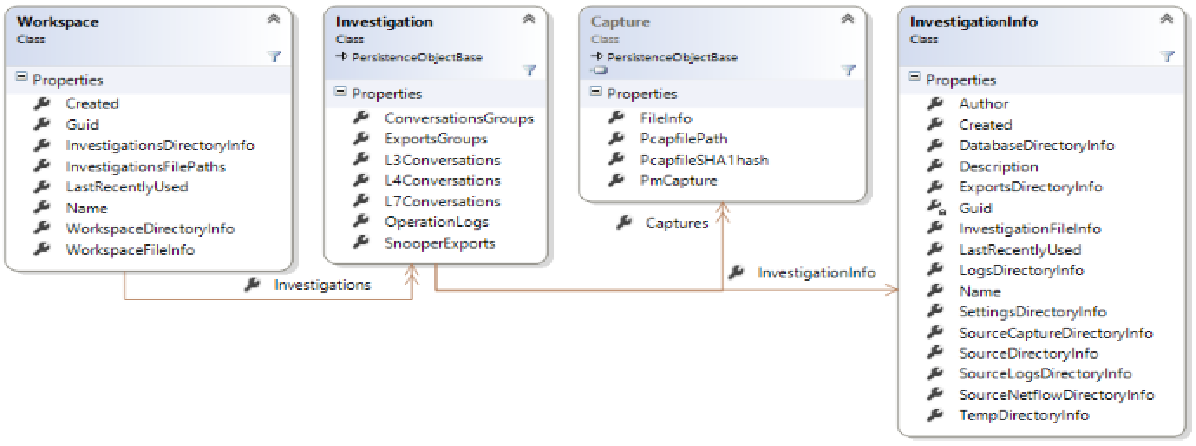

Fig. 3: Database models used in Netfox Detective to ensure persistence of workspaces and investigations.

Captured network data are processed using a pipeline that extracts crucial information for further analysis, see Fig. 4. At first, a PCAP file is added to an investigation and parsed in *PmLib* module that builds a frame collection. Each module *LxConversationTracker* asynchronously processes every new frame and creates an appropriate *PersistenceCollection* for *X-th* level conversation, e.g., for L3, L4, or L7 layer. The *L7ConversationTracker* builds application layer conversations over TCP or UDP sessions, and creates application protocol messages called *L7PDUs* without any syntactical knowledge of the particular application protocol. Conversation tracking and reconstruction uses port numbers and TCP sequence numbers to detect missing data or unclosed sessions. It also employs timestamps to increase accuracy of reconstruction. Detailed description of packet reassembling is described in [2].

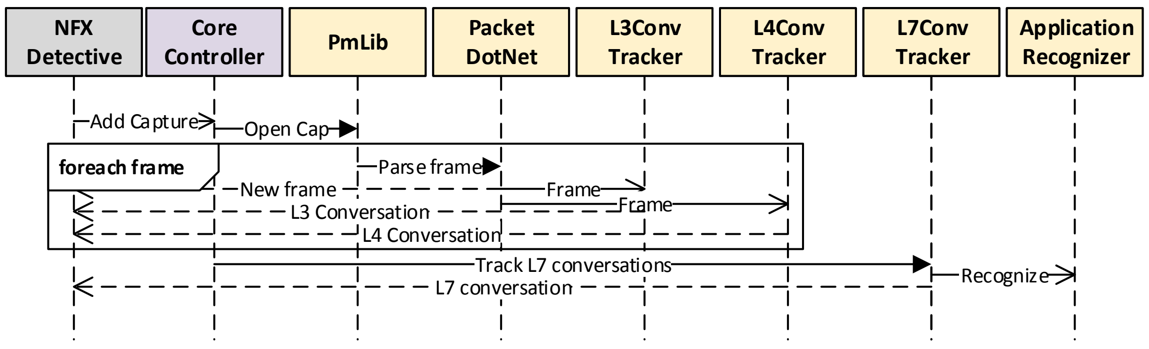

Fig. 4: Asynchronous capture file processing pipeline.

The key issue for successful L7 analysis provided by application extractors (*snoopers*) is a correct identification of *L7Conversations*. Identification is provided by the *Recognizer* that assigns one or more application tags to a L*7Conversation*. The algorithm uses extended IANA database of well-known ports, RTP recognizer for dynamic RTP streams [5], or SPID algorithm [3] using statistical based identification.

[5] See http://www.mongodb.org/about/.

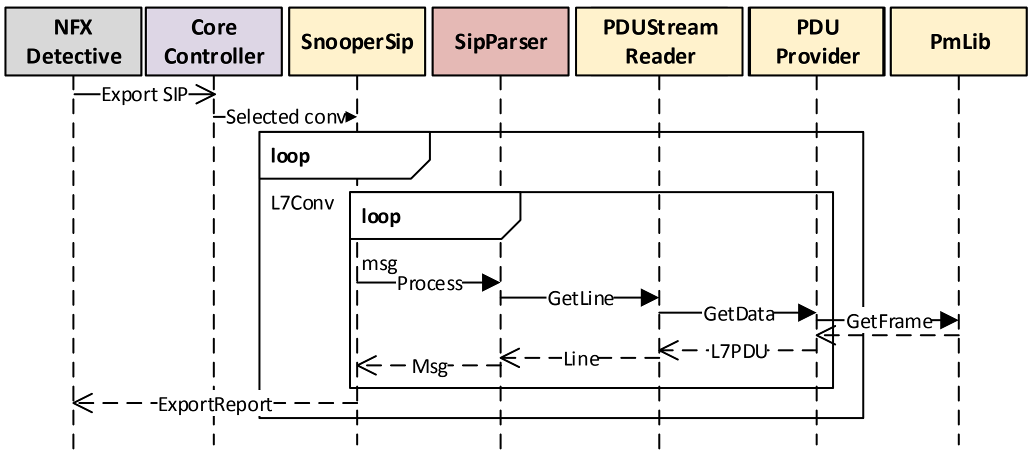

Fig. 5: SIP application protocol data extraction using the SIP Snooper module.

Snooper modules are dynamically loaded to Netfox Detective, therefore, no recompilation is needed when a new application parser (*snooper*) is added. The *snooper* is a reconstruction engine of the application protocol. Outputs of one *snooper* can be chained into another *snooper* for further reconstruction, e.g., outputs of HTTP analysis can become inputs of web mail *snooper*. *Snoopers* export the contents of conversations with corresponding meta-data obtained during the application protocol processing into a current investigation. Each *snooper* defines its own models, views and view-models to provide a detailed presentation of reconstructed data, e.g, HTTP snooper shows reconstructed web pages, an email snooper lists reconstructed emails, VoIP snooper describes VoIP session with RTP streams to be replayed, etc. Example of SIP snooper processing is at Fig. 5.

As mentioned above, s*noopers* provide a syntactical analysis of communication. Until this point, data processing has been based purely upon information obtained from layers L3 and L4. The *snooper* analyses a particular application protocol, i.e., it parses application messages. The *snooper* communicates with low level modules as *PDUStreamReader*, or *PDUProvider* that deal with missing or overlapping segments, TCP sequence number overflow, missing SYN and FIN packets, IP defragmentation, etc. The *snooper* processes logical *L7PDU*s as soon as all conversations have been successfully restored over L4. It receives data from the *PDUStreamReader* module. *PDUProvider* prepares input data for *PDUStreamReader* using one of four strategies shown on following example, see also Fig. 6:

1. *Broken Interlay* – The first application message consists at maximum of PDU1 and PDU2 transmitted in Frame 1, 2, 3. The arrival of Frame 4 on client side signals that the application message has ended. This is typical for request/response protocols. The second application message is contained only in PDU3 and the third in PDU4.
2. *Continued Interlay* – The first application message consists at maximum of PDU1, PDU2, and PDU4 without taking into account frames arriving in opposite direction.
3. *Mixed Interlay* – The first application message might consist of PDU1, PDU2, PDU3, and PDU4. This mode mixes PDUs from both directions into one bi-directional stream.
4. *Single Message Interlay* – Every application message consists only of one single PDU.

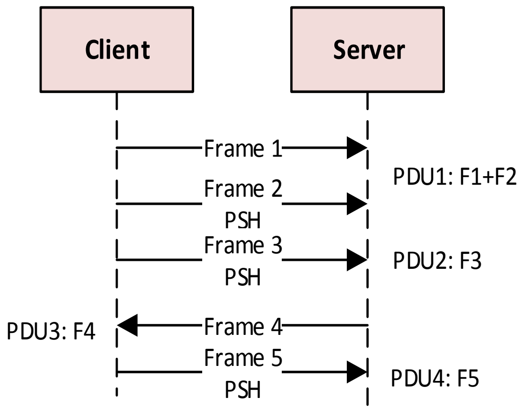

Fig. 6: Processing PDUs

Generally, one application message can be composed of one or more PDUs. When some frames are missing, a virtual frame is created in order to complete proper PDU processing by a snooper. Using this approach, succeeding un-corrupted message will be properly reassembled in contrast to MS Monitor that might misinterpret succeeding messages.

4 Extracting Application Data

Application protocol data extraction is a process of analysing application layer data streams, i.e., payloads of L7 conversations. This analysis requires knowledge of application protocol syntax as well as semantics to extract significant information for forensic analysis. Following examples of application processing demonstrate how L7 parsing is implemented in NetFox Detective. They also describe advanced techniques for reconstruction of incomplete or corrupted application data.

4.1 Web mail

Communication using emails is necessity for everyone today. A majority of users uses web browser to access their mail boxes and to operate with their mail accounts. Therefore, HTTP protocol is mostly used to tunnel web mail communication. Traditional email protocols like POP3, IMAP, and SMTP have been mostly put aside from the end user perspective, even though they are still used among email providers.

In this study, we have focused on web mail traffic analysis in order to create a general model that would be able to process web mail independently on particular service used. As it is seen in Table 1, following operations similar for all analysed web mail services can be identified despite the fact that general structure of web mail is not standardised and web mail providers implement various transmission methods how to deliver web mail contents, e.g., using RPC sessions, JSON applications, etc. Table 1 shows how basic web mail operations can be identified in URL or HTTP header payload using simple pattern matching.

Operation	*Web mail patterns used in URL or HTTP header*
New Message	Keywords: from, to, subject, cc, bcc, content/body, SendMessage.
Message manipulation	URL request/HTTP header: move, delete, MoveMessageToFolder.
Email header request	URL request/HTTP header: list, search, GetInboxData.

Table 1: Common operations and methods of their detection.

Web mail services can be divided based on data privacy protection into three groups: (i) web mail service with unencrypted authentication and mail transmission, e.g. *zoznam.sk, tiscali.cz* (ii) web mail services with encrypted authentication and unencrypted mail transmission, e.g., *centrum.cz*, *atlas.cz* and *mujmail.cz* (iii) web mail services with encrypted authentication and encrypted mail transmission, e.g., *seznam.cz*, *gmail.com*, *email.cz*.

When web mail authentication is encrypted, web mail communication cannot be identified using standard URL analysis but other techniques can be employed. One possibility is to use client's header extension in SSL/TLS handshake where Hello message contains the server name. The server name might indicate that following SSL/TLS communication transmits web mail. Also, DNS resolution can be employed to detect web mail service, see Table 2.

Web mail	*Server name*	*Encoding*
seznam.cz, email.cz	email.seznam.cz	FastRPC
Gmail	mail-attachment.googleusercontent.com	application/x-www-form-urlencoded;charset=utf-8
Yahoo	mail.yahoo.cz	application/json multipart/form-data – incl JSON
MS Live		application/x-www-form-urlencoded
Centrum/Atlas/Mujmail	mail.centrum.cz	application/x-www-form-urlencoded
Roundcube	<private service hostname>	application/x-www-form-urlencoded
Horde	<private service hostname>	multipart/form-data;

Table 2: Identification of particular web mail service.

4.2 Voice over IP

Voice over IP (VoIP) is a technology for transmission of phone calls over IP infrastructure Main advantage of VoIP is that uses the same infrastructure for both data and voice transfers which save money but also reduce maintenance requirements. From point of view of network forensics, VoIP creates a new challenge for detection and interception of suspect's calls. Traditional call interception on telecommunication networks was subjected to strict and well-known rules. VoIP works in flexible environment of IP networks with a large variety of application protocols and codecs. The most common VoIP technologies are SIP [6] for call signalling and RTP [7] for media transmission. Following section describes how SIP and RTP protocols can analysed.

4.2.1 Signalling protocols

Session Initiation Protocol (SIP) is an application layer protocol for signalling and controlling multimedia sessions over IP networks. It is mostly used for voice/video calls and instant messaging. It defines messages that establish, modify and terminate sessions between end points. SIP is a text-based protocol with some similarities to HTTP or SMTP. It serves mainly for user registration and establishing VoIP connection. Media streams (voice or video) are transmitted using RTP protocol [7] or its secured version SRTP [8]. Description of transmitted media stream is encoded using Session Description Protocol, SDP [9].

SIP communication is independent on transport protocols and may use TCP, UDP or SCTP transport. The protocol utilizes a transaction based communication. Each transaction is represented by a request and at least one response. SIP protocol usually communicates on TCP/UDP ports 5060 or 5061 (encrypted sessions).

4.2.2 SIP analysis

The extraction algorithm iterates over *L7 conversations* identified by an application recognizer. Whenever a valid SIP message is obtained, it is processed by SIP snooper that extracts meta-data related to the call. SIP messages with the same Call-ID form a SIP event. Generally, SIP snooper uses two basic methods *INVITE* for call establishment and *REGISTER* for authentication. However, this trivial processing is not sufficient when some messages are corrupted or missing.

1	INVITE sip:10.10.10.109 SIP/2.0
2	Call-ID: D99151DA-1DD1-11B2-B23A-BC0375BD6E00@10.10.10.214
3	From: "unknown"<sip:10.10.10.214>;tag=30652209562016038532
4	To: <sip:10.10.10.109>
5	... c=IN IP4 10.10.10.214 ... m=audio 49152 RTP/AVP 3 97 98 110 8 0 101 ...

Table 3: Example of data transmitted in a SIP message

Table 3 shows what kind of information can be obtained from SIP protocol:

1. *Request method or response code – this can be used to recognize a call.*
2. *Call-ID – a unique identifier used for grouping corresponding messages.*
3. *From header – identifies caller party.*
4. *To header – identifies calling party.*
5. *SDP body – identifies media stream, codecs, RTP ports, etc.*

For network forensic purposes, several SIP message are interesting to get meta-data about call exchange, e.g, *INVITE*, *BYE*, and *REFER* as requests and *100* (Trying) and *180* (Ringing) as responses. Using these requests and responses, we are able to extract SIP calls even if captured signalling is incomplete. As depicted on example in Fig. 7, even if *INVITE* message is lost, the same information can be obtained from related messages (marked by red dot).

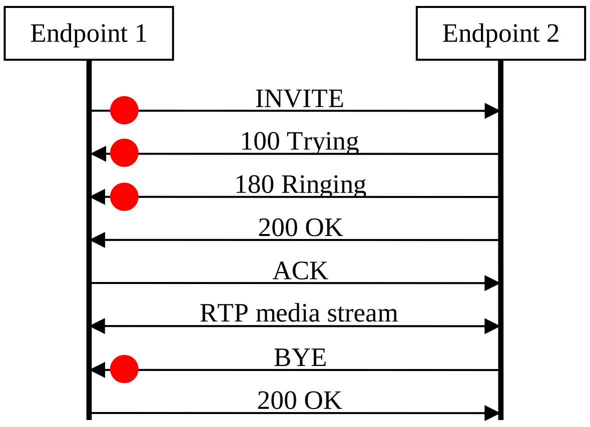

Fig. 7: Typical message exchange during a SIP call.

Another issue is pairing incomplete signalling data with media streams. Network Detective implements heuristic based on RTP and TCP timestamps that result in probabilistic correlation of reconstructed VoIP calls. Utilizing these strategies, we are able to provide better reconstruction in comparison with other tools, see Table 4.

FILE	*NFX DETECTIVE*	*WIRESHARK*	*NETWITNESS*	*PACKETSCAN*
Complete PCAP	2	2	2	2
PCAP without INVITE	2	0	2	0
PCAP without 200 OK	2	2	2	2

Table 4: Detection of VoIP calls when INVITE or 200 OK messages are missing**.**

4.2.3 Real-time Transport Protocol (RTP)

RTP [7] is a stateless application protocol used to transfer media streams over the network. The RTP also provides simple detection of lost packets and multiple streams synchronisation with minimal overhead. It is usually transferred over UDP due to minimal overhead and stateless behaviour. RTP does not retransmit lost packets because even if they had eventually arrived, they would have not been needed any longer. RTP detection is not easy due to the dynamic port assignment. As a part of RTP standard is description of RTP Control Protocol (RTCP) messages that are used to deliver additional control session data, e.g., stream source description, sent data size counter, packet loose, jitter, etc.

4.2.4 Detecting RTP without signalling protocols

Common VoIP concepts separate signalling data (SIP/SDP) from media streams (RTP). Both protocols use their own PDUs and paths through the internet. When signalling data are missing, it is generally not easy to detect RTP stream with dynamic UDP ports and identify what kind of codec is used for voice or video data transmitted. Netfox Detective uses advanced detection algorithm to identify RTP as follows. For full algorithm, see [5]:

1. *RTP header contains a fixed version 2.*
2. *Mostly all current VoIP applications use only UDP transport protocol with ports greater than 1024.*
3. *Observed packets should have a minimal packet length as required by the standard unless extension flag is set.*
4. *Typical RTP stream is collection of large number of small packets with the same SSRC identifier.*

RTP header contains *Payload type* (PT) for codec identification. This field is mostly used for statically mapped codecs like G.711, GSM, G.722, or G.729, see [10]. Dynamically assigned codecs like Speex, G.726, AMR, or Silk require identification information transmitted in signalling protocols. If signalling protocols are not present in a captured file, it is hard to identify the codec. In such case, it is possible to use an identification method based on ratio between payload size of RTP packets and timestamp differences between two successive packets. Since this ratio usually does not change, this method is sufficient for codecs identification without signalling data [5].

4.2.5 Incomplete RTP streams

In case of incomplete or corrupted RTP packets, advanced reconstruction techniques have to be applied. Following case studies present some solutions how to reconstruct such data.

The first case study (see Fig. 8) shows communication between Alice and Bob where a link towards Bob is lossy. In this case, Bob's phone will miss two RTP packets 2 and 4. When naïve approach to decode a received audio stream is applied, audio tracks would not be synchronized, see Fig. 9. This will complicate further reconstruction and forensics analysis.

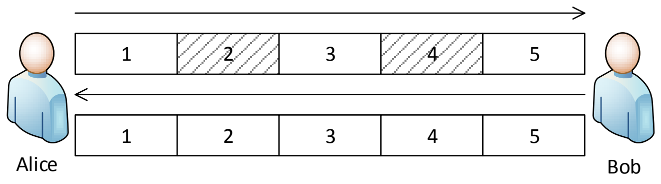

Fig. 8: Incomplete RTP streams

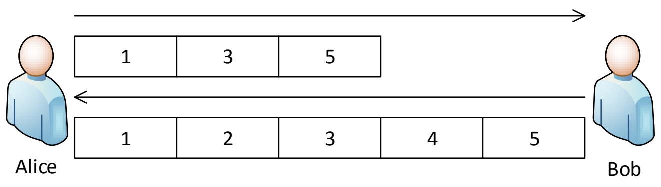

Fig. 9: Naïve RTP Reconstruction

For advanced RTP reconstruction, a following procedure is proposed:

1. *Compute the number of lost samples*.

   Using RTP timestamps a difference between the last received packet and the next one after the loss can be calculated. Then, correlation between real-time and timestamp difference indicates how many packets were lost. Although this correlation is codec dependent it can be used for reconstruction. For example, if the last received packet had timestamp 1000 and the next received packet had timestamp 9000, we may assume that 8000 audio samples were lost.

2. *Reconstruction of missing samples.*

   The knowledge of a codec used is important to encode raw audio data since the codec specifies the sampling rate that has been used. For example, codec G.711 uses sample rate 8000 Hz. In case of 8000 lost samples with sampling rate 8000 Hz one second audio is missing. Therefore, lost packet can be substitute with silence audio or white noise right after decoding to fill the specified gap and synchronize bi-directional audio steams.

Example of RTP streams after reconstruction is depicted in Fig. 10. As it is seen now, timeline of both RTP streams is properly aligned that is important for proper forensic analysis.

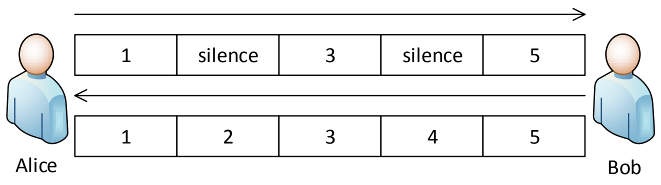

Fig. 10: Reconstructed RTP streams.

5 Conclusions

This paper presented a new framework for network forensics analysis developed during security research. This framework has modular architecture with focus on two important areas: stream reassembling and application reconstruction. Stream reassembling is an important part of the tool. If not done properly, some packets can be skipped without proper analysis. On the other hand, some streams can be reconstructed incorrectly and include frames that do not belong to the reconstructed stream. The main benefit of our study is proposal of different heuristics and techniques that are able to build streams from captured packets even if some packets are missing without a need to parsing application protocol. Proposed heuristics are used to detect missing SYN and FIN packets, to identify lost packets within a stream, to detect overlapped conversations, etc., so that TCP and UDP streams are properly reconstructed for further network forensics analysis.

Following application reconstruction is provided by independent application snoopers that parse reconstructed L7 streams, extract application based meta-data, and visualize results to an investigator or security administrator. Application snoopers also implements advance techniques for proper reconstruction of incomplete application data as presented on web mails and VoIP communication. At the moment, Netfox Detective is able to work with any IP, TCP or UDP streams. It supports reconstruction of web pages, web mails, emails using SMTP, POP, or IMAP protocols, instant messaging protocols (XMPP, ICQ, Yahoo), and VoIP (SIP, RTP). The user interface allows an investigator to filter required conversations and expert interesting data for further analysis.

In this research, we concentrated more on accurate data reassembling, parsing and reconstruction. Future research will be focused on efficient analysis of big data, distributed parsing and employment of advanced detection methods using machine learning, statistical based detection, etc.

6 Acknowledgment

Presented research results were supported by project "Modern Tools for Detection and Mitigation of Cyber Criminality on the New Generation Internet", no. VG20102015022 granted by Ministry of the Interior of the Czech Republic and by project "Research and application of advanced methods in ICT", no. FIT-S-14-2299 granted by Brno University of Technology.

References

[1] S. E. Pilli, R. Joshi and R. Niyogi, "Network forensic frameworks: Survey and research challenges.," *Digital Investigation,* pp. 14-27, 2010.

[2] P. Matoušek, J. Pluskal, O. Ryšavý, V. Veselý, M. Kmeť, F. Karpíšek and M. Vymlátil, "Advanced Techniques for Reconstruction of Incompleted Network Data," in *International Conference on Digital Forensics & Cyber Crime*, Seoul, 2015.

[3] E. Hjelmvik and W. John, "Statistical protocol identification with SPID: preliminary results.," in *Sweedish National Computer Networking Workshop.*, 2009.

[4] G. Palmer, "A Road Map For Digitial Forensic Research," in *First Digital Forensic Research Workshop (DFRWS),* Utica, New York, 2001.

[5] P. Matoušek, O. Ryšavý and M. Kmeť, "Fast RTP Detection and Codecs Classification in Internet Traffic.," *Journal of Digital Forensics, Security and Law,* pp. 99-110, 2014.

[6] H. Schulzrinne, J. Rosenberg, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley and E. Schooler, *SIP: Session Initiation Protocol,* IETF RFC 3261, 2002.

[7] H. Schulzrinne, S. Casner, R. Frederick and V. Jacobson, *RTP: A Transport Protocol for Real-Time Applications,* IETF RFC 3550, 2003.

[8] M. Baugher, D. McGrew, M. Naslund, E. Carrara and K. Norrman, *The Secure Real-time Transport Protocol (SRTP),* IETF RFC 3711, 2004.

[9] M. Handley and V. P. C. Jacobson, *SDP: Session Description Protocol,* IETF RFC 4566, 2006.

[10] H. Schulzrinne and S. Casner, *RTP Profile for Audio and Video Conferences,* IETF RFC 3551, 2003.