Miroslav Voznak - Ivo Zbranek - Miralem Mehic - Dan Komosny - Homero Toral-Cruz - Jerry Chun-Wei Lin *

COVERT CHANNEL IN RTP PAYLOAD USING A POINTER IN SIP HEADER

The paper addresses the issue of hiding data in the network flow. The authors discuss a new proposal of the steganographic approach in IP telephony: transmitting texts using the pointer in the SIP header by means of a data stream within the RTP protocol. This method is based on tagging the binary sequences in payload of the RTP packets, with the individual binary sequences representing individual steganogram characters. Subsequently, the position of the binary sequences is recorded in the SIP header in the Via field and the branch parameter. The proposed way of hidden data transmission cannot be detected by existing anomaly detectors; and does not represent an approach to statistical detection of covert channels. In fact, it is a new contribution to covert communication in ordinary VoIP traffic.

Keywords: *Steganography, VoIP, RTP, payload, SIP, pointer.*

1. Introduction

In the context of the constant rapid development of the data network and related services, the security of transmitted data is increasingly important. One of the most progressive services offered by these networks is IP telephony, using protocols applied across the Internet network. Consequently, real threats such as sensitive personal data, internal or corporate data leaks or server overloads causing the unavailability of the service, etc. arise. One of the possible ways to transmit sensitive or classified data is to use a covert communication channel. The creation of covert communication channels is the subject addressed by the scientific branch referred to as steganography the results of which can also be applied in IP telephony [1]. The most significant results of the research in this area were published by Mazurczyk and Szczypiorski and yield methods which enable creating a covert communication channel [2 and 3]. All data communication in digital networks is realised by means of a data transmission described in the binary scale. This means a vast amount of zeros and ones transmitted one after another. Given the large data volumes, the probability that a sequence, which would allow us to interpret the subsequent information in the required form, is found, is high. Such sequences can also be found within voice communication that uses the IP protocol. The highest volume of data to be exploited is the data stream itself, serving to transmit voice data.

Individual characters which we need to transmit are transformed to binary sequences according to the ASCII table and the transmitted RTP packets within session are locally stored by the sending User Agent. Subsequently, binary sequences representing individual characters are found out in local stored RTP packets and these position are recorded in absolute/relative values. The ASCII table is again applied to transform characters to HEX format and the obtained chain is used in the parameter "branch" as a randomly generated chain which is transferred within ordinary nearest transaction. The receiving User Agent can easily store incoming RTP packet and realized reverse process. Robustness can be enhanced by using shared key for the transform table combining steganography and cryptography.

2. Related Work

The idea of hiding data within the network flow combines two elements: utilisation of unused packet fields and information encoding in traffic behaviour. The first element

* [1]Miroslav Voznak, [1]Ivo Zbranek, [1]Miralem Mehic, [2]Dan Komosny, [3]Homero Toral-Cruz, [4]Jerry Chun-Wei Lin
[1]VSB-Technical University of Ostrava, Czech Republic
[2]Brno University of Technology, Czech Republic
[3]University of Quintana Roo, Mexico
[4]School of Computer Science and Technology, Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, China
E-mail: miroslav.voznak@vsb.cz

is a well-known technique that emerged from old Xmas packets. These packets, with every single option set for the used protocol, are included in a well-known nmap network scanning tool, and they were named Xmas packets because they resemble bright bulbs on a Christmas tree. These packets can be easily detected by intrusion-detection systems (IDS), or more advanced firewalls with an anomaly detection feature [4 and 5]. The second element, i.e. encoding information in the traffic behaviour, was first presented in [6] and this idea was further modified in [2] by Mazurczyk and presented as the LACK (Lost Audio Packets Steganography) solution for VoIP communication.

Covert channels were first observed and defined in the mid-80s as a result of the rapid development of communication networks. Lampson from Cambridge classified communication channels into three categories: Storage, Legitimate, and Covert. He also gave the first definition of covert channels stating that covert channels are the channels which are used for information transmission even though they are neither designed nor intended to transfer information at all [7]. In the last ten years, a large number of covert channels were introduced, and a further development of new techniques is expected in the upcoming years. All these techniques can significantly affect the level of security and reputation that certain communication solutions offer. Viewed from the client's point of view, it is reasonable to doubt the safety and quality of a particular communication solution which has weak points in the system that enable unnoticed leaks of confidential data. Because of that, covert channels are under a close supervision of governments and security companies that seek to prevent these leaks. One of the first papers on this subject [6] presents an approach to statistical detection of covert channel embedded in network packet delays. This simple technique implies the existence of clear differences between the packet delay and it is based on the probability of the existence of a covert channel, which is calculated as follows (1):

$$P_{CovChan} = 1 - \frac{C_{\mu}}{C_{\max}} \quad (1)$$

where C_{μ} is the packet count at the mean and C_{max} is the maximum packet-count of the histogram formed from the number of packets received with a given time, the experiment is explained in [6]. According to Mazurczyk and Szczypiorski [2 and 3], there are several types of steganographic techniques in VoIP networks but all of them can be classified into the following three categories:

- Packet modification steganography;
- PDUs time relations;
- Technique that requires hardware modification of device.

The first one represents a technique which takes advantage of unused fields in the protocols, mostly in IP (Internet Protocol), UDP (User Datagram Protocol), TCP (Transmission Control Protocol) or even RTP and RTCP (RTP Control Protocol) packets [2]. This method is susceptible to detection by IDS. Instead of using a separate RTCP flow, the authors proposed embedding the control information into the actual RTP flow. Unused bits in the IP/UDP/RTP headers signal the type of parameters, whereas the parameter values are embedded as a watermark in the voice data [8 and 9]. The second method is based on the deliberate data delay, since the VoIP content is very sensitive to delay and jitter variations. This method is related to the LACK technique mentioned above that is based on deliberate delays of the VoIP packets. The receiver will only consider the packets that are delivered on time and discard any packets that are delayed. Using this technique, instead of discarding the delayed packets, the receiver will read them for the purpose of a steganographic analysis. This method is quite hard to identify in the network, and equally hard to implement. There are a few variations of this technique such as affecting the order of packets, modifying the inter-packet delay or introducing the intentional losses [6]. The third method consists in using modified hardware for the purpose of a steganographic analysis. This technique is referred to as HICCUPS (Hidden Communication System for Corrupted Networks) for the VoWLAN (Voice over Wireless LAN) specific environment. However, the steganographic method is quite difficult to implement and detect, since it takes advantage of the imperfections of the transmission medium environment [10]. In addition, the authors of this paper have published the results related to the IP telephony security analysis and monitoring [11 and 12]. Their recently published papers discuss the computation of the available steganographic bandwidth in a VoIP covert channel and proved detecting irregularities in SIP flows caused by the injection of SIP headers or by the increased amount of SIP messages [13 and 14].

This article is organised as follows: the main idea is explained in the third section, the experiment and the results are presented and discussed in the fourth section and the conclusion is provided in the fifth section.

3. Main Idea Explanation

The RTP protocol has been designed to ensure media data transmission. In our case voice data was digitalised using codec G.711 and packets are generated every 20ms.. So within each individual packet, RTP transmits 160B of data, enabling to describe voice data, the so called payload. A five-minute phone call has potential of 2400000 sequences that can represent the character required by us. To convert data, internationally

standardised (ASCII) or self-designed converting tables can be applied. In the latter case, the self-designed tables need to be shared with the party receiving the steganogram. In case we intend to describe the entire 7-bit ASCII table, we need to establish 128 unique sequences represented in the binary code. Each character from the ASCII table can be described using 7-bits, but the converting table assigns each character 8-bits, where the first most important bit equals binary 0. Where a further reduction of characters to be used is required, e.g. alphanumerical character incl. lower case characters, all we need is 62 unique sequences. To separate and precisely identify the position of a particular sequence, 3 to 4 specific characters should be allocated.

To identify the packet in which the necessary sequences are located, the *Sequencenumber* value provided by the RTP protocol can be used. This value is unique and allows for uniquely determining the position of data. Individual sequences that correspond to the required character can be looked up in the payload of the packet concerned and we can record information about their position. Where the packet does not contain a particular sequence, we need to use a different packet. Since we need to provide the counterparty with the information which packet is the relevant one and the position of the sequence within the payload, we use the header of the SIP protocol, placing the information about the packet and position into the payload of the given packet. Once the information contained within the SIP header has been transmitted, the counterparty, once it has been provided with the converting table, can start to search for particular sequence positions representing individual characters. Once the binary sequences have been converted using the converting table, a steganogram can be drawn, sent by the first party or the sender. This method consists of several individual phases as is depicted in Fig. 1.

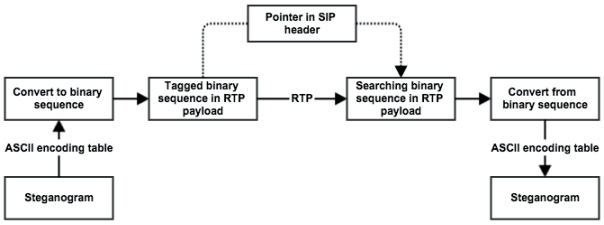

Fig. 1 The phases of the proposed steganographic approach

First, individual characters (ASCII) are converted from a steganogram into corresponding binary values (BIN) using the converting table. Then, these binary sequences are tagged within the payload of the RTP packet. Individual binary sequences stand for individual characters within the steganogram. The position of particular binary sequences together with the packet sequence number is recorded in the pointer located in the SIP header. In order to enhance the confidentiality, the pointer in the SIP header is re-converted from the decimal into the hexadecimal format. Using the information contained within the pointer, the counterparty is able to look up the binary sequences in the payload of the RTP packet. Lastly, the binary (BIN) sequences are transmitted back into alphanumeric characters ASCII) which form the steganogram.

3.1 Steganogram Conversion to Binary Sequence

To convert a steganogram into the binary format, both internationally standardised and self-developed converting

		A	B	C	D	E	F	G	H	
1	0000h:	01010101	11100001	10101111	10000001	10000000	11110010	00000000	11101110	U.......
2	0008h:	01010101	01010011	01011001	01000101	01000111	01000110	01000101	01011001	USYEGFEY
3	0010h:	01010011	11001101	11110001	10101111	10000001	10000000	11110010	10100111	S.......
4	0018h:	01010011	11001101	11	11	01000111	01000001	01000110	01000101	.W][GAFE
5	0020h:	01011110	01010000	00000000	10000110	11110011	10000010	10000010	11110010	^P......
6	0028h:	10100111	10000101	01010001	01011110	01000101	01000001	01000000	01000110	..Q^EA@F
7	0030h:	01011010	01011111	01010110	10000101	10000110	10101111	11110010	11110011	Z_V.....
8	0038h:	00000000	11101110	01010101	01010011	01011001	01000101	01000110	01000110	..USYEFF
9	0040h:	01000101	01011000	01010010	01010111	00000000	11101110	00000000	00000000	EXRW....
10	0048h:	00000000	11101111	01010110	01011101	01011000	01000100	01000111	01000111	..V]XDGG
11	0050h:	01000101	01011000	01011100	01010000	01010100	00000000	10011101	10000101	EX\PT...
12	0058h:	11001101	01010110	01011101	01011011	01000110	01000011	01001101	01000010	.V][FCMB
13	0060h:	01000000	01000100	01011001	01010010	01010110	11001101	10000101	10000101	@DYRV...
14	0068h:	11101111	01010111	01011101	01011010	01000000	01001110	01001000	01001000	.W]Z@NHH
15	0070h:	01001111	01000000	01011010	01011100	01010110	11001101	10000101	00000000	O@Z\V...
16	0078h:	10000100	11101111	01010111	01011101	01011010	01000000	01001111	01001001	..W]Z@OI
17	0080h:	01001111	01000011	01000101	01011101	01010100	10000100	00000000	10100111	OCE]T...
18	0088h:	00000000	10100111	11110001	10000101	01010101	01010011	01011111	01011011	....US_[
19	0090h:	01011010	01011000	01011101	01010100	11101110	10101111	10000010	11001011	ZX]T....
20	0098h:	11100101	11001100	11001011	10101110	10000000	11110011	00000000	00000000	....

Fig. 2 The payload of an RTP packet (sequence number 39796)

tables can be applied. In the latter case, the tables need to be provided to the counterparty. Using such a table, we obtain binary sequences that correspond to individual alphanumeric characters. In order to encode basic English alphanumeric characters, including lower case ones, a total of 62 unique sequences is required.

To separate and unambiguously identify the position of the sequences in the payload of the RTP packet, it is necessary to allocate 4 specific characters, where the relative positions of binary sequences are recorded, or 3 specific characters, where absolute positions of binary sequences are recorded. Using 7-bit ASCII tables, we obtain 128 unique sequences, an amount sufficient to transmit basic alphanumeric characters. When converting the characters from the steganogram, a character of the steganogram is looked up in the ASCII column in the converting table ASCII [15]. Its corresponding binary sequence value can be found in the Binary (BIN) column. This procedure is repeated until we obtain binary sequences for all characters of the steganogram. Subsequently, the binary sequences are tagged in the payload of the RTP packet, as is depicted in Fig. 2 in the real RTP packet for a selected word STEGANOGRAM.

3.1 Tagged Binary Sequence in RTP Payload

To tag the required binary sequence in the payload of the RTP packet, the packet sniffer should be applied (tcpdump is enough). It enables to display the payload of an RTP packet in the binary format. Even better results can be obtained by applying the network protocol analyzer which allows for filtering individual protocols and separating the payload of individual packets from each other. Based on the sequence, we subsequently look up the number of the RTP packet in which the required binary sequences are located. A particular sequence is tagged as follows: a binary sequence of a particular character obtained by means of the converting table is looked up in the payload of a particular RTP packet and its position is tagged. Where the required binary sequences failed to be identified in the packet, we proceed to search through further packets which contain the required binary sequence. To determine the position of a binary sequence we can apply both relative and absolute values. Afterwards, the sequence numbers and the position of individual binary sequences are recorded into the pointer in the SIP header.

3.1.1 The relative position of a sequence

The relative position of a sequence is defined by the first position of the binary sequence representing the first character of the steganogram. This position (the first position in the binary sequence) is the starting point for the determination of the distance between the first character of the octet and subsequent binary sequences. To determine the individual position, the following equation is applied (2):

$$P = B - F \tag{2}$$

where P is the final position of the given character, B is the number of the starting position of the sequence representing the given character and F is the number of the position of the sequence of the given character. We repeat this procedure until we have found all the necessary binary sequences, representing all the characters within the steganogram. Where the binary sequence of the first character is located at the end of the payload of the RTP packet and the subsequent characters are positioned before the first binary sequence, the relative distance values can be negative.

3.1.2 The absolute position of a sequence

When applying the absolute position, there is no need to determine the distance of the positions between individual binary sequences. All we need to establish is the value of the position of the given octet. This approach simplifies and speeds up the tagging of a particular binary sequence, as is depicted in Fig. 3. It also reduces the number of the specific characters required since it is not necessary to use the minus sign. The positions of individual binary sequences may gain 160 different values in a particular payload of the RTP packet.

	A	B	C	D	E	F	G	H
1	1	2	3	4	5	6	7	8
2	9	10 [S]	11	12	13	14	15	16
3	17	18	19	20	21	22	23	24
4	25	26	27	28	29	30	31	32
5	33	34	35	36	37	38	39	40
6	41	42	43	44	45	46 [A]	47	48
7	49	50	51	52	53	54	55	56
8	57	58	59	60	61	62	63	64
9	65	66	67 [R]	68	69	70	71	72
10	73	74	75	76	77	78	79	80 [G]
11	81 [E]	82	83	84	85 [T]	86	87	88
12	89	90	91	92	93	94	95 [M]	96
13	97	98	99	100	101	102	103	104
14	105	106	107	108	109	110 [N]	111	112
15	113 [O]	114	115	116	117	118	119	120
16	121	122	123	124	125	126	127	128
17	129	130	131	132	133	134	135	136
18	137	138	139	140	141	142	143	144
19	145	146	147	148	149	150	151	152
20	153	154	155	156	157	158	159	160

Fig. 3 The absolute position of a sequence – the payload of an RTP packet (sequence number 39796)

3.2 The pointer in a SIP header

In order to obtain information about the position of the binary sequences within the payload of the RTP packet,

we need to share such information with the counterparty. The information necessary to unambiguously identify a binary sequence representing a particular character of the steganogram is the *Sequencenumber* or the *relative* or *absolute* position of individual binary sequences within the RTP packet concerned. To share information, we use the *branch* parameter in the *Via* field. Subsequently, a random number of indicators can be shared with the counterparty using any SIP method.

3.2.1 Branch parameter

This parameter serves to unambiguously identify SIP transactions. Its value needs to be unique in time and space for all and any requests sent by the User Agent. The structure of the *branch* chain needs to contain alphanumeric characters. A typical length of the *branch* chain generated using the IP telephony application referred to as softphones differs, see Table 1.

Implementation of Branch Lengths String in Softphones Table 1

Softphone	String length (without magic cookie)
Media5-fone/4.1.3.3034 iOS/8.3	17
SessionTalk Version 5.11 iOS/8.3	28
X-Lite 4.8.0 75950-02930038 OSX	28
YATE/5.0.0 Linux	9

The maximal length of the *Via* field may return values from 0-65535 B. The typical length, however, is much lower. For instance in the SNORTSIP pre-processor, it is pre-set to 1024 B. The Branch parameter contains a so-called *magiccookie*, a chain which is the prefix for every transaction. This chain consists of the following characters: *z9hG4bK* and is constant. We use this parameter to transmit information about the sequence position in selected packets, where the requested information is found after prefix *z9hG4bK*.

3.2.2 Encoding pointers in a SIP header

When applying the *branch* parameter, it is recommended to convert the indicator values so that they correspond better with the standard values typical for this parameter. This approach enhances the chances that the covert communication channel will not be detected. A converting table is used to encode the pointer. Individual data from the indicator is encoded into the hexadecimal format. 7-bit ASCII table provides 128 values. 122 out of these 128 values are used to encode information contained directly in the pointer. Another 3 or 4 values are used for specific characters depending on whether relative or absolute binary sequence positions are applied (Table 2). Every specific character is encoded independently. Hexadecimal values can be recorded using lower case (a, b, c, d, e, f), upper case (A, B, C, D, E, F) characters or a combination there of which further inhibits the possibility to uncover the covert communication channel.

Special Characters for Encoding SIP Header Table 2

ASCII Character	HEX	Function
{	7b	Beginning set of pointers in RTP packet
| |	7c	Separation of individual pointers in RTP packet
}	7d	Ending set of pointers in RTP packet
~	7e	Character minus

When encoding, we go through the pointer (the branch chain), taking three characters at a time. Where the numerical value is higher than 122, we take away one character from the right and we encode only the two-character value. This approach is applied to the entire pointer with the exception of specific characters that are encoded independently.

Encoding the pointer in a SIP header -the relative position. To encode the pointer using the relative positions to tag a binary sequence, four specific characters need to be applied (see Table 2.). The character tagging the start of the pointer set referring to the binary sequences relating to a particular packet (Sequence number), the character for tagging the end of the pointer set referring to the binary sequences relating to a particular packet (Sequence number), the character for splitting individual pointers (binary sequence positions) within the RTP packet concerned and the character representing the minus sign.

Encoding the pointer in a SIP header – the absolute position. To encode the pointer using the absolute positions to tag a binary sequence, three specific characters need to be applied, since the specific character representing the minus sign is omitted. Individual octets representing binary sequences are tagged using values ranging from 1 to 160, by which their position in the payload of the RTP packet is determined.

3.2.3 Transfer of pointers during a call

The Re-INVITE (INVITE) method of the SIP protocol can be used to transmit information during the call or with any next transaction in the SIP dialogue (SIP method BYE). By changing the parameter of an established connection using the Re-INVITE method, a SIP message with a modified header in the *branch* field can be transmitted repeatedly

and it is a new transaction in the established SIP dialogue. The *branch* field of the INVITE method allows for verifying whether the counterparty is ready to receive the steganogram; or the field can remain empty (magiccookie). With the Re-INVITE method, the *branch* field already contains an encoded chain which represents the positions of binary sequences corresponding to the steganogram. Any number of *Re-INVITE* methods can be created during a call as necessary. The Re-INVITE method serves to modify the parameter of an established connection. Fields From, To, Call-ID are usually equally set as in the original INVITE message, but the remaining parameters can be modified.

3.2.4 Searching a binary sequence in the RTP payload

Searching for individual binary sequences that represent particular characters of the steganogram is done based on the information from the pointer in the SIP header in which such information is encoded. Decoding information is done by means of the conversion of hexadecimal values to decimal values using the converting table in which the HEX value (Hexadecimal value consists of two characters) has a corresponding value in DEC. Thus we obtain non-coded information in the pointer, where one can see individual binary sequence positions in the payload of the RTP packet identified through a sequence number.

3.2.5 Convert a steganogram from a binary sequence

Once the pointer in the SIP header has been decoded, the sequence number of the given packet and the positions of individual binary sequences in this packet are displayed. Binary sequences are converted into ASCII again by means of the converting table in which each binary section has a corresponding ASCII representation. Once all binary sequences have been converted, we obtain a steganogram sent by the counterparty.

4. Experimental Verification and Discussion

To verify a feasibility of the steganographic approach, the codec G.711 A-law is used with packetisation period of 20 ms. Values of a single sample may range from 0 to 255, or from 00000000 to 11111111 in the binary format. These values describe the amplitude of the given sample. For this reason, it would be great to obtain a phonetically diverse analogous voice signal, thus increasing the likelihood that many different binary sequences would occur. As a part of the experiment, a standard five-minute audio call was simulated. Asterisk software, to which the two stations were connected, was used as a SIP softswitch. The SIP softswitch itself ran on the Linux (Ubuntu 14.04) operation system. The first station hosted Windows 7 OS and softphone application Yate 5.4.2. The second station hosted OS X 10.10 with softphone application X-Lite 4.8.0.

Binary Representation and Sequence Position of Steganogram with Text Steganogram - Everything Needed For Transmission Steganogram Table 3

Char Sequence number	Binary sequence	Relative position	Absolute position
S 39796	01010011	73	10
T 39796	01010100	600	85
E 39796	01000101	568	81
G 39796	01000111	560	80
A 39796	01000001	288	46
N 39796	01001110	800	110
O 39796	01001111	824	113
G 39796	01000111	560	80
R 39796	01010010	456	67
A 39796	01000001	288	46
M 39796	01001101	680	95

Bob (Originator) sends a steganogram with text *STEGANOGRAM* to Alice (Recipient). Bob converts the characters in the steganogram by means of the converting table [15] into the binary format as seen in the table (Table 3). A connection is established and lasts 5 minutes. Afterwards, Bob goes through the payload of individual RTP packets. Once he finds the binary sequences (depicted in Fig. 2) corresponding to the text of the steganogram, he notes down the sequence number of the packet concerned. All binary sequences for all steganogram characters were found in a single packet with sequence number 39796. Now, Bob can apply any of two approaches to record the position of individual sequences, i.e. the relative or the absolute approach. The non-coded pointer in the SIP header (the branch parameter with the absolute position of a sequence) is as follows:

branch=z9hG4bK39796[10|85|81|80|46|110|113|80|67|46|95]

We can check positions in Figs. 2 and 3, now the coded pointer (branch) in the SIP header is as follows:

branch=z9hG4bK274f067b0a7c557c517c507c2e7c6e7c717c507c437c2e7c5f7d

Table 3 shows that using the absolute positions to tag binary sequences is more efficient. Subsequently, these positions are recorded together with the sequence number of the packet concerned into the branch parameter using the specific characters (Table 2). Using the converting table, the pointer is then encoded into hexadecimal format. Once Alice receives the pointer in the SIP header, she must first decode the pointer from the hexadecimal format into the decimal format using the converting table [15]. Thus she obtains the original non-coded pointer which allows her by means of the sequence number 39796 to determine where the relevant binary sequences are positioned in the payload (Figure 2) of the RTP packet. Last, invidiual binary sequences (Figure 3) are converted back into the text format (ASCII) using the converting table [15].

The total amount of data transmitted in the covert channel reflects the codec used to encode the voice, the frequency of packet generation, the size of the payload of the RTP packet and the length of the call. G.711 codec with A-law logarithmic compression is used in our experiment and the amount theoretical transmission capacity of the covert channel can be determined using the following equation (3):

$$CS = \frac{PC}{PF} \cdot t \quad [\mathrm{B}] \tag{3}$$

where CS is the total amount of data transmitted in the covert channel [B], PF is the frequency of packet generation [s], PC is the size of the payload within a single packet [B] and t is the length of the call [s]. Entering the values in the equation returns 2,400,000 B in case of 5 minutes long call, it is the maximum number of sequences which can appear in RTP packets. In reality, we only have a few opportunities to send a steganogram within one SIP session due to the existence of only several transactions. Although it is possible to apply the re-INVITE method to insert a new transaction in SIP dialogue, the anomaly detectors would be able to detect these transactions too often. On the other hand, known techniques cannotcannot detect such steganograms in ordinary transactions and nowadays no methods to detect existence of such a covert channel exist. Of course, we can take into account the possibility to enhance security based on the shared secret and the total amount of data transmitted in the covert channel can be enhanced by optimising the converting table.

Next issue to be addressed is the length of the branch value. A typical exact value for the length of the branch parameter has not been defined. For each softphone, the implementation of the branch parameter is different. In the softphone group tested, the length of the branch parameter ranged from 9 to 28 characters (without magiccookie). If we draw on information included in definition RFC 3261, each implementation of a SIP protocol by means of UDP can process messages of up to 65535 B in size. The basic length of the branch parameter within the SIP pre-processor for SNORT is set to 1024 B, but can be adjusted within the range from 0 to 65535 B. OpenSIPS implementation refers to the length of 32 B. If we want to prevent the application of the method described in this paper by limiting the length of the parameter to a particular value, e.g. the value corresponding to the typical length of this parameter, it could, under certain circumstances, restrict the VoIP communication as the implementation specification defined in RFC 3261 had not been adhered to. Where the value of the length of the branch parameter is preset to a typical value due to security reasons, the pointer in the SIP header could be adjusted to such limitations by shortening the length which is considered typical. Where the pointer in the SIP header has the typical length of the branch parameter, this steganographic method becomes virtually untraceable.

Another issue is a limitation due to the placement of B2BUA (Back-to-back User Agent) into the communication path. The described steganographic method allows establishing a covert communication between two end users. B2BUA inserted between two communicating parties terminates the connection from the one UA and establishes a completely new connection targeted towards the second UA. By establishing a new connection, new information is fed into the SIP header. As a result, information about the indicators contained within the branch parameter in the Via field is lost. Similar limitations also apply where a SBC (Session Border Controller) element is inserted into the communication path. Nevertheless, such limitations are not restrictive, since a new communication path containing two SIP proxy, which are interconnected by means of SIP trunk, could be created; or a communication could be set up directly between the communicating UAs. The existence of B2BUA or SBC elements in the communication path can be easily detected and communication can be set up directly between both parties.

5. Conclusion

This paper discusses the steganographic method enabling to cover communications within the data stream of the RTP protocol. This method consists in tagging binary sequences in the RTP stream and placing tags into SIP header. These sequences present characters encoded using a converting table. The position of individual binary sequences within the

packet in which it is located is reflected in the branch value, in the Via field in the SIP header. In order to enhance the protection against uncovering the covert communication channel, the pointer in the SIP header has been converted into the hexadecimal format. The transmission of the pointer series in the SIP header can be performed by means of the Re-INVITE method, nevertheless it is recommended to insert only ordinary transactions in the SIP dialogue and not to insert more Re-INVITE modifications artificially since these are prone to be detected as an anomaly. Certain limitations of this method occur when the communication path is interrupted by an element which impairs the original value of the branch parameter of the Via field such as B2BUA or SBC. The benefit of the steganographic method is that it does not modify the transmitted data, thus no degradation or alteration of the transmitted data occurs. In addition, this approach prevents the use of the steganoanalysis. Where the length of the branch parameter corresponds to the commonly used typical values, this method becomes virtually untraceable.

Acknowledgement

The research leading to these results received funding from the grant of SGS reg. no. SP2015/82 conducted at VSB-Technical University of Ostrava, Czech Republic and was supported by the National Sustainability Program under grant LO1401. This research was also partially supported by the Moravian-Silesian Region within the project VSB-Technical University of Ostrava activities with China. The research was performed using the infrastructure of the SIX Centre.

References

[1] KLIMO, M., KOVACIKOVA, T., SEGEC, P.: Selected Issues of IP Telephony. *Communications - Scientific Letters of the University Of Zilina*, 6 (4), 2004, 63-70.

[2] MAZURCZYK, W., SZCZYPIORSKI, K.: Steganography of VOIP Streams, *Lecture Notes in Computer Science*, 5332 LNCS (PART 2), 2008, 1001-1018.

[3] MAZURCZYK, W., SZAGA, P., SZCZYPIORSKI, K.: Using Transcoding for Hidden Communication in IP Telephony. *Multimedia Tools and Applications*, 70 (3), 2014, 2139-2165.

[4] NEVLUD, P., BURES, M., KAPICAK, L., ZDRALEK, J.: Anomaly-Based Network Intrusion Detection Methods. *Advances in Electrical and Electronic Engineering*, 11 (6), 2013, 468-474.

[5] VOZNAK, M., SAFARIK, J., REZAC, F.: Threat Prevention and Intrusion Detection in VOIP Infrastructures. *International J. of Mathematics and Computers in Simulation*, 7 (1), 2013, 69-76.

[6] BERK, V., GIANI, A., CYBENKO, G.: *Detection of Covert Channel Encoding in Network Packet Delays*. Dartmouth College : Hanover, Technical Report TR536, 2005.

[7] LAMPSON, B.W.: Note on the Confinement Problem, Communications of the ACM, 16 (10), 1973, 613-615.

[8] JANICKI, A., MAZURCZYK, W., SZCZYPIORSKI, K.: Steganalysis of Transcoding Steganography. Annales des Telecommunications/Annals of Telecommunications, 69 (7-8), 2014, 449-460.

[9] MAZURCZYK, W., KOTULSKI, Z.: New VOIP Traffic Security Scheme with Digital Watermarking. *Lecture Notes in Computer Science*, 4166 LNCS, 2006, 170-181.

[10] SZCZYPIORSKI, K. HICCUPS: *Hidden Communication System for Corrupted Networks*. Proc. of Intern. multi-conference on Advanced Computer Systems, October 2004, 31-40.

[11] SAFARIK, J., VOZNAK, M., REZAC, F., MACURA, L.: IP Telephony Server Emulation for Monitoring and Analysis of Malicious Activity in VOIP Network. *Communications - Scientific Letters of the University of Zilina*, 15 (2A), 2013, 191-196.

[12] REZAC, F., VOZNAK, M., TOMALA, K., ROZHON, J., VYCHODIL, J.: Security Analysis System to Detect Threats on a Sip VOIP Infrastructure Elements. *Advances in Electrical and Electronic Engineering*, 9 (5), 2011, 225-232.

[13] MEHIC, M., SLACHTA, J., VOZNAK, M.: *Hiding Data in SIP Session*, Proc. of 38th Intern. Conference on Telecommunications and Signal Processing (TSP), 2015, 1-5, doi: 10.1109/TSP.2015.7296445.

[14] MEHIC, M., MIKULEC, M., VOZNAK, M., KAPICAK, L.: Creating Covert Channel using SIP. *Communications in Computer and Information Science*, 429, 2014, 182-192.

[15] ASCII CONVERSION CHART, online available url https://designthatsit.files.wordpress.com/2013/12/ascii20conversion20chart.gif.