2010

An Exploration of covert channels within voice over IP

Patrick Lloyd

An Exploration of Covert Channels Within Voice Over IP

By

Patrick Lloyd

Thesis submitted in partial fulfillment of the requirements
for the degree of
Master of Science in
Networking and Systems Administration

Rochester Institute of Technology

B. Thomas Golisano College
of
Computing and Information Sciences

May 4, 2010

Thesis Reproduction Permission Form

Rochester Institute of Technology

B. Thomas Golisano College of Computing and Information Sciences

Master of Science in Computer Security and Information Assurance

An Exploration of Covert Channels within Voice over IP

I, Patrick Lloyd, hereby grant permission to the Wallace Library of the Rochester Institute of Technology to reproduce my thesis in whole or in part. Any reproduction must not be for commercial use or profit.

Date: 05/04/2010 Signature of Author:

Rochester Institute of Technology

B. Thomas Golisano College of Computing and Information Sciences

Master of Science in Computer Security and Information Assurance

Thesis Approval Form

Student Name: Patrick Lloyd

Thesis Title: An Exploration of Covert Channels within Voice over IP

Thesis Committee

Name	Signature	Date

Bo Yuan
Chair

Daryl Johnson
Committee Member

Peter Lutz
Committee Member

Table of Contents

Abstract

In the following thesis, an overview of covert channels within Voice over IP is given and then expanded upon by presenting an experiment which proves the ability to hide messages within the Session Initiation Protocol (SIP) and Session Description Protocol (SDP) of a Voice over IP packet. The plain text nature of the SIP and SDP packets allow for an easily embedded message to be encoded into the expected data, while also being "hidden in plain sight" due to the packet only being sent once per VoIP session. While previous papers [15] have proposed the ability to hide covert messages within the plain text SIP and SDP packets of a VoIP call stream, this thesis is the first to carefully analyze and test the ability to embed data in these packets and send a covert message, based on an agreement between the sending and receiving parties. Results include the success for covert messages to be hidden within the Max-Forwards field, a field used for the total number of hops between sender and receiver, the V field, a field used for the version of SIP being used, the T field, usually used for the time a session becomes active on the sending and receiving ends, and finally the O field which designates the owner the call was originally sent from. This success was met with equal failure of previously proposed abilities to hide messages [15] in the Branch statement, tag field, and Call-ID field. A method for systems administrators or network administrators to detect covert channels coming in over a VoIP enabled network using a simple, modified java based packet capture tool is then presented with the ability to check the Max-Forwards, V, T and O fields, due to their low entropy and easy detectability. Using this method, a discussion is given regarding the detectability of covert channels as compared to previous research papers.

1. Introduction

In the original thesis written for proposal to embed covert messages into voice over IP, it was claimed that voice over IP packets have the ability to be modified to include covert messages in a number of fields throughout the session initiation protocol (SIP) and session description protocol (SDP) [15]. The plain text nature of the SIP and SDP packets and rarely implemented encryption for these packets on software based phones, and hardware alike, make them targets for attackers who possess the knowledge of intercepting, modifying the packets which contain the initiation and description, and then forwarding the packets onward, almost instantaneously to ensure that there is little change in the call as compared to a normal call without modification. It has also been claimed [15] that fields within the SIP and SDP packets have the ability to be changed without affecting the quality of the call, while still being able to hold a significant number of characters which will allow for covert message strings to be sent to a receiving party. On account of no proof or testing found to back up these claims, this thesis attempts to prove these theoretical claims and then given a systems administrator or network administrator the ability to easily detect covert channels being sent via their own network, merely by running a Java based application and reviewing the logs produced.

Within this follow up thesis, results and experimental setup will be explained in addition to an explanation of the tools developed to detect these covert channels. The basic structure of the thesis will be as follows: In section 2 it will cover the background of covert channels and the closely related topic of steganography including steganography throughout history, present day steganography, covert channels and how they relate to steganography in sections 2.1, 2.2, 2.3 and 2.4, respectively. In part 3 a discussion of applications to Voice over IP will be given with

specifics given about the protocols contained within Voice over IP and how covert messages are able to be embedded in each. In part 4 the experiment setup will be explained and followed, as well as the results of the experiment being given in section 5. Further discussion regarding advantages and disadvantages of this method of covert channels will take place in section 6. In section 7 an exploration of entropy will be done, followed by section 8 which will explain a Java based packet sniffing tool, modified to detect and easily present covert channels to a systems administrator. A brief exploration of tools developed by colleagues at Rochester Institute of Technology will take place in section 9, followed by ideas for future work and the concluding paragraph in sections 10 and 11.

2. Background

2. 1 A Brief History of Covert Channels and Steganography

Steganography, or "concealed writing" as roughly translated from Greek [25], is the art of embedding a message into a textual or graphical piece of work so that the message is unreadable and virtually unrecognizable to anyone without the knowledge of how to retrieve and view it. Steganography has a rich past and has been used for ages within the textual and graphical media areas, and can easily be used within the digital media arena as well. It was the precursor to digital covert channels and thus can be said to be quite similar.

The history of covert channels date back to before modern methods existed for detecting or decoding hidden messages within written media. Specifically, creating and decoding hidden messages was first recorded around 480 BC when the Greeks used covert channels to hide

messages to inform the Spartans of the plans for an invasion of Xerxes by removing the wax cover from a wooden table and writing the plans for the invasion on the wood before "covering the message over with the wax again. In this way the tablets, being apparently blank, would cause no trouble with the guards along the road…." [11]. This method has also made its way into pop culture through TV shows reenacting the battles of World War II, showing the allies passing messages into Italy on the backs of wine bottle labels. Whether this was a true practice could not be found and therefore could not be documented past this pop culture reference [24]. One method during the current time period is the sending of a message by a German spy which, in most other contexts, wouldn't be seen as a suspicious message. While the message read:

> "Apparently neutral's protest is thoroughly discounted and ignored. Isman hard hit. Blockade issue affects pretext for embargo on by products, ejecting suets and vegetable oils."

A message was actually sent which indicated: "Pershing sails from NY June 1.", found by taking the second letter of each word within the original message [11].

A historical example of steganography dates back to the 16th century when Joannes Trithemius wrote three pieces of literature in Latin, which are said to be the first works which discuss cryptology and the ability to embed hidden messages in a textual or graphical manner. Upon further review of these pieces of work, scientists found that the third book contained multiple steganographical messages including the equivalent of "The quick brown fox jumps over the lazy dog" as well as "The bearer of this letter is a rogue thief. Guard yourself against him. He

wants to do something to you," both in Latin. Although these messages were meant to be apparent to the other party to whom Trithemius was sending the message to, it was not discovered and translated until 1996 by two German Researchers, Dr. Thomas Ernst and Dr. Jim Reeds. Given that it took so long to find these hidden messages, this example shows the power of steganography to hide a message in plain sight [12].

Other message types included the use of quilts during the Civil War to guide slaves over a given path or pass on messages without the ability for slave owners to recognize the patterns and symbols within the quilt. As seen in Figure 1, one example is the use of the "bear claw" to indicate to slaves a specific path to follow to their freedom [12]. The crew of the U.S.S. Pueblo also used various hand positions while being photographed to send the message of "snowjob" to their allies after being captured [12].

Figure 1: The bear claw quilt to indicate a safe route to slaves [12]

2. 2 Present Day Covert Channels and Steganography

In terms of present use and the justification behind the project described in this thesis, there has been an ever growing interest in the use of covert channels and steganography after the events of 9/11/2001. With terrorism being a very real threat and the ability for terrorists to use

both technologically simple and complex means to hide messages to be sent to followers, an interest has been taken in covert channels and steganography, as well as the ability to carry messages without the knowledge of a middle party [14]. There have been reports of the possibility of messages being hidden within pornographic websites as well as in eBay auctions and images, but thus far no suspected images or auctions have been made public [13]. In addition to these threats, the proposal has been given that there could possibly be hidden messages in the television or audio broadcasts from Osama Bin Laden [17]. Because covert channels and steganography are so hard to detect, there have not been any known cases released to the public, but research is still going on to address these fears.

While most uses of steganography are seen as negative and only used to bypass common understanding of a specific piece of work, there are also legitimate uses for steganography as well. The use of one way keys in the forms of SHA1 and MD5 keys has become a prevalent way for software users to know if the software they are using is the same as the one released by the software company. Use of steganography in email has become popular in organizations and even countries which do not allow for the use of encryption on their email. This steganography can be in the form of line shifting, word shifting, or hiding messages within different parts of the message, as can be seen in Figure 2 which is a note sent from California Governor Arnold Schwarzenegger to members of the California State Assembly (note the first letter of each line). Watermarking on images and audio files allow for the true ownership to be determined and kept while the image or audio file is exchanged on the Internet as seen in Figure 3 [2].

To the Members of the California State Assembly:

I am returning Assembly Bill 1176 without my signature.

For some time now I have lamented the fact that major issues are overlooked while many unnecessary bills come to me for consideration. Water reform, prison reform, and health care are major issues my Administration has brought to the table, but the Legislature just kicks the can down the alley.

Yet another legislative year has come and gone without the major reforms Californians overwhelmingly deserve. In light of this, and after careful consideration, I believe it is unnecessary to sign this measure at this time.

Sincerely,

Arnold Schwarzenegger

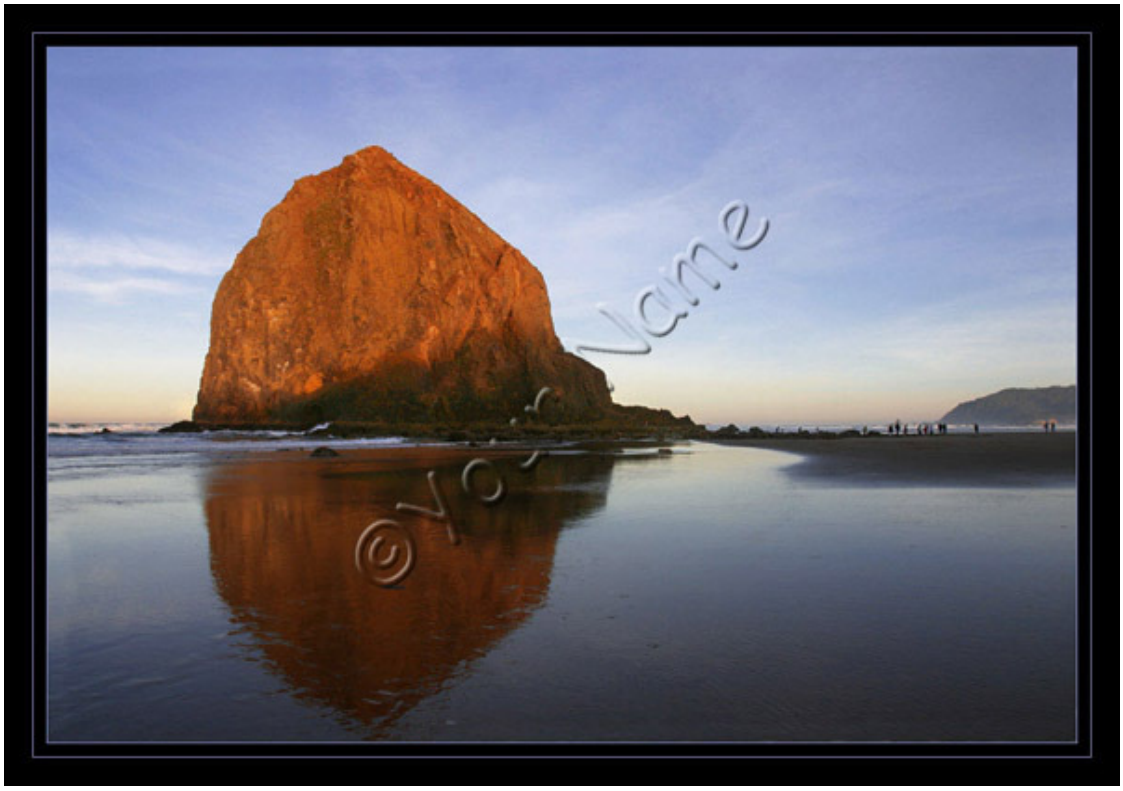


Figure 2: Note from Arnold Schwarzenegger to California State Assembly

Figure 3: Photographical watermark example [23]

2. 3 What is a covert channel?

While VoIP communication is a new concept, the idea of covert channels is far from new. The idea of covert channels, or hiding a piece of information inside of a communication medium which allows for a bypass of security measures on that communications medium [26] was proposed over 30 years ago. This specific phrasing and usage was first proposed in 1973 in an ACM paper by Butler Lampson. Until this point is was a relatively new idea which no one believed could be implemented in an electronic format. The Department of Defense eventually picked up this idea in 1985; once more research was done [26]. The formal definition and types of covert channels varies greatly between sources. For the sake of this thesis, a covert channel will be defined as: "The ability for a communication to be sent over a communication medium which was either not designed to send alternate communications over, or that allows for the bypass of system security policies within the process" [26]. Because of the amount of research into covert channels by multiple parties with multiple ideas into what they are, a truly formal definition is yet to be established. Thus, the thesis will use the preceding definition and will furthermore define three types of covert channels that can be used within VoIP.

The authors of [26] discuss a number of different types of covert channels, to which this section applies and attempts to explain for the sake of enumerating the relevant types of covert channels. Covert channels have the ability to exist in both a user based model or within a kernel based model. In other words, a covert channel has the ability to run with different types of permission, thus giving it different types of access to the resources of its host and destination machine. As the average user has the ability to write to a "profile" on a source and destination machine, and can only write to the software which supports the machine and its operations, it

is said that the user will generally have a user space covert channel ability. Simply put, this means that a covert channel may be able to be spread through one of the mediums this thesis will cover in subsequent paragraphs, and be able to signal a second party by modifying, creating, writing to or broadcasting through a piece of software or file. This can be the writing to a file on the destination machine, the modification of a protocol as viewed through a piece of software, attempting to access the destination machine on a given port or with a certain protocol, or signaling the second party with a covert message while using a third party application such as a MMORPG game. The second basic type of existence that a covert channel has is in a kernel based model which requires the covert channel to be able to interface with the machine itself, fully bypassing the operating system altogether, but also requiring a very high permission which is generally hard to do. To send a covert channel within kernel mode requires the user to become an authority over the hardware itself, thus being able to leave messages within unused sections of system memory (RAM), modifying processes being run within the central processing unit (CPU) to send a signal, or even modifying properties of the video which the user sees by sending signals through the video card. While there are programs that are available to change some of these values, they are mainly used for system utilities and generally are not used except for locally on a machine and in cases where a recovery or other such utility is needed because of the dangers they introduce to the system. While there are two different types of existences that covert channels can manifest themselves as, there are also three different types of covert channels which can be used to send a message to the user on a destination computer. The first is commonly known as a storage channel, or value based spatial channel as phrased by the authors of [26], utilizes the idea of modifying packets or the area

which packets are stored within the destination machine [14]. These modifications usually consist of writing to the packets and embedding information to a header or footer of sorts, thus to be only received by the receiving party who is aware that there is information hidden within the packet. For example, when sending a packet over a medium through which a covert channel is being utilized, a packet may be of a certain length or may have a certain embedded header with 32 bits of data. If either one of these qualities is true that packet may represent the letter F in the schema of a mod 26 alphabet algorithm. An alternate use of covert channels is using a timing channel or value based temporal channel as phrased by the authors of [26]. This timing channel is true to its name in that it modifies the timing properties of packets and when they are sent or at what point they are sent in regards to their comparison in frequency [14]. For example, a packet sent with a time stamp ending in 1 may represent the letter A while another ending in 5 may represent E. This can also be applied to the frequency, in which the destination times how many packets it receives within a given amount of time and bases lettering algorithms on such. So, if 5 packets are received within a time period of 2 seconds, they may represent E, similar to the previous example. A third type of covert channel has gone unmentioned in most of the sources from which this research section was based. The authors of [26] propose a new type of covert channel which utilizes the transitions between events and how events, whether it is storage channels or timing channels, interact with each other at the destination. For instance, the example of modifying the port to which a protocol is sent is used within [26] where if a packet is sent to port 1234, and once received a second one is sent to port 6464, the destination may know that this is representative of the letter A. As with the rest of the types of covert channels covered previously, the continuation of sending packets will

spell out a message based on the transitions between events as they occur. This was alluded to in [14], in which the authors discuss hybrid covert channels. While hybrid covert channels represent the ability for both a storage and timing channel to interact with the same destination at once, a transitional channel does not necessarily allow for the covert message to be stored or to modify an already existing file, but rather only keeps track of the events which occur.

2. 4 Covert Channels versus Steganography

Within the majority of the papers written on covert channels, there seems to be the ability and practice to interchange the phrases “covert channel” and “steganography.” While this thesis does not propose they are not very similar, and one may be a part of another, it does propose the two be separated and used in different contexts. As given in the definition earlier of a covert channel, a covert channel is “The ability for a communication to be sent over a communication medium which was either not designed to send alternate communications over, or that allows for the bypass of system security policies within the process [26].” On the contrary, this thesis proposes that steganography is: “The process through which a message is hidden within a static medium, which is unable or has a large amount of difficulty to be changed and can only be seen by a readily prepared party.” As discussed previously, this is not to say that covert channels and steganography are not similar in nature. This thesis proposes that a covert channel is within the realm of steganography in itself, but not the other way around. This is based on the conclusion that a covert channel must be contained within a transport medium rather than within a static medium and therefore would be in transport. Thus the covert channel would acquire a fleeting property in that if not captured, it cannot be

proven the covert channel ever existed. On the other hand, a steganographical message will always exist unless completely destroyed, and is readily viewable and analyzable by any viewing party.

The example of a drawing can be used when discussing steganography. A drawing may have a hidden message within it, codes or symbols embedded within it or a 3 dimensional image available to be seen by only those parties who have the knowledge in how to look at the image in a certain way or from a certain angle. While this can definitely be considered a steganographical image, the permanent nature of the image and the everlasting message contained within it do not make it a covert channel because of a covert channel's fleeting properties and need to be captured to become steganography. In the example of the drawing, a message was hidden within the drawing, a medium which generally is not used to send messages in such a readily available form, and is relatively permanent. On the other hand, if a timing channel is sent to a destination machine and the packets are not analyzed as to the number which comes in every second, there is really no evidence which can be tracked at a later date as to what happened within this situation. The destination may have readily available the means to distinguish the message, but the message is not hidden within a static medium, but rather a dynamic one which is constantly changing, if only in the way that there is the change in which packets are being sent. There is also the ability to easily change the packets, in contrast to a drawing which is static and can only be changed by overwriting data, rather than modifying it as proposed here.

3. Applications to Voice over IP

The biggest advantage to Voice over IP and the reason why it has become so popular is because of its real time transfer ability of both voice and data streams. This can be seen in the average VoIP call or video instant messaging applications, which have become more popular as promises of "free long distance calling" have been promoted. The inner workings of Voice over IP are split into two different phases, one being the "signaling phase" and the other being the "conversation phase." Within the experimental section the signaling phase will be addressed as part of the experiments, [but both phases are explained with] SIP and SDP being part of the signaling phase and RTP being part of the conversation phase [16].

3. 1 SIP

The SIP packet is the basic mechanism by which the sending party informs the receiving party that they would like to connect and exchange data. As soon as the number for the receiving party is dialed and recognized, the PBX server, or proxy through which the VoIP phone is allowed access to the Internet and then continues on to the receiving party, sends a SIP message to the receiving end to invite them to take part in the call. This SIP packet contains the basic information that is needed to communicate, similar to most other communication mediums, including the protocol being used in the SIP, maximum number of hops between source and destination, identification of the sender and receiver and the type of data that is contained within the transfer. Once the receiving end acknowledges that they will take place in the call, i.e., the handset of the receiver is picked up, the receiving end sends a session description packet as it needs to inform the sending client which abilities it has and what it understands to be the agreed upon medium and encryption [16]. A SIP packet is also sent at

the end of the session to inform the receiving end that it is disconnecting, therefore ending the session.

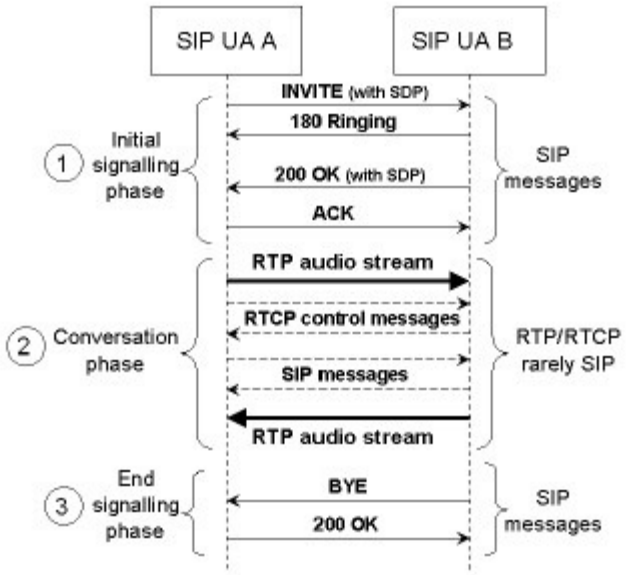


Figure 4: The VoIP call setup and breakdown [16]

21	8.498583	10.37.1.2	10.37.1.5	SIP	Request: ACK sip:1006@10.37.1.5:37504;rinstance=278a967f192798f9
57	8.711116	10.37.1.3	10.37.1.2	SIP	Request: ACK sip:1016@10.37.1.2:5060
931	13.111319	10.37.1.3	10.37.1.2	SIP	Request: BYE sip:1016@10.37.1.2:5060
937	13.217689	10.37.1.2	10.37.1.3	SIP	Status: 200 OK
938	13.217838	10.37.1.2	10.37.1.5	SIP	Request: BYE sip:1006@10.37.1.5:37504;rinstance=278a967f192798f9
940	13.220865	10.37.1.5	10.37.1.2	SIP	Status: 200 OK
2	3.530216	10.37.1.3	10.37.1.2	SIP/SDP	Request: INVITE sip:1006@workgroup, with session description
4	3.983212	10.37.1.2	10.37.1.5	SIP/SDP	Request: INVITE sip:1006@10.37.1.5:37504;rinstance=278a967f192798f9, with session description
10	8.383627	10.37.1.5	10.37.1.2	SIP/SDP	Status: 200 OK, with session description
22	8.498693	10.37.1.2	10.37.1.3	SIP/SDP	Status: 200 OK, with session description

Figure 5: The SIP “BYE” message is sent when the call disconnects

```
(1)    INVITE sip:bob@biloxi.example.com SIP/2.0
(2)    Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
(3)    Max-Forwards: 70
(4)    From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
(5)    To: Bob <sip:bob@biloxi.example.com>
(6)    Call-ID: 3848276298220188511@atlanta.example.com
(7)    CSeq: 12345 INVITE
(8)    Contact: AliceM <sip:alice@client.atlanta.example.com;transport=tcp>
(9)    Content-Type: application/sdp
(10)   Content-Length: 151

(11)       v=0
(12)       o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
(13)       s=-
(14)       c=IN IP4 192.0.2.101
(15)       t=0 0
(16)       k=clear:9123123kjnhdasdoq12e31021n2e4
(17)       m=audio 49172 RTP/AVP 0
(18)       a=rtpmap:0 PCMU/8000
```

Figure 6: SIP and SDP color coded within a SIP packet for clarity [16]

Within the following section, Figure 6 will play a large role and be referred to often. In Figure 6, it can be seen that there is an entire SIP packet which will be sent on initiation of the VoIP session. The packet itself is color coded by the authors of [16] for clarity, with the top section (lines 1 through 10) being in gray representing the session initiation protocol as opposed to the session description protocol (lines 11 through 18) being in white at the bottom of the image. Both of these are contained within the SIP packet that is sent to the destination party when the call is initiated, which allows for the opposite party to begin to process and receive the information that the source is trying to do so with. As can be seen while analyzing the SIP packet, there are a number of different details sent from the source to the destination when the call is initiated. Within the initiation packet, the source sends an invite to join the call to the destination, consisting of the address from which the call came from, the source, destination, maximum number of forwards for the call to take, the identification number, contact, and type of call that is being initiated (voice as opposed to data). As can be seen in Figure 6, there are a

number of fields bolded, which will be discussed in subsequent paragraphs. A similar practice is done in the SDP, which is contained within lines 11 through 18 and has a white background.

The SIP, in general, supports five characteristics of a call including the location, availability, capabilities, session setup and session management, as documented within Request for Comments 3261 [22]. While the first two characteristics, location and availability are relatively self explanatory, the additional three may require some additional explanation. A user's capability is determined based on what media they are attempting to connect to the session through, and the parameters of the media and device they are using to connect. The session setup is the part of the call in which there is a "ringing" [22] and a party is invited to join the session. The client is also informed of the capabilities so that an agreement between the sending and receiving parties can be made. This may be used to invite additional parties into a conference type call, with an additional SIP packet being sent to the desired invitees. While not all of the fields are necessary and the bulk of them are ignored by the SIP completely, they are included in the SIP packet to ensure uniformity across VoIP sessions.

As can be seen in Figure 6, a number of fields are bolded, the bold text representing fields which have the ability to be changed, either because they are ignored by the SIP altogether, or the information that they provide does not lend anything to the call other than a formality. While the fields do have to exist, because of their criticality to establishing the call itself, they theoretically can be changed in a number of different ways to be potential covert channels. As can be seen on the second line of the session initiation protocol, the branch statement is used to form an identification for the transaction that will occur within the call [16]. While in [16] it is

seen that the branch statement does have to begin with “z9hG4bK,” the other 5 digits can be modified to send a covert message, without fear of having the call dropped or the message detected. This lends itself to the ability to send a covert message using the SIP, but there are also a number of other fields which allow for a similar practice. Fields such as Max-Forwards, tag (within the source address), call ID (before the domain name), cseq (the sequence in which the call was originally sent, for identification purposes), and contact (before the actual contact name and domain) can theoretically be changed to allow for covert messages to be sent. This is because none of them are necessarily required for the voice call to successfully take place, but rather are for convenience once the packet reaches the destination [16].

3. 2 SDP

The SDP packet is sent after the SIP packet has been accepted at the receiving end to inform the sending party what protocols can be used within the VoIP call, as well as at intervals throughout the call to ensure that media attributes have not changed. While this interval is not specified in the RFC [8], it can be observed as being sent midway through the call (8 seconds into a 13 second call) in Figure 5. This packet is sent from the receiving end with details such as the media attributes which the two clients are connected by, the owner and session ID, the time which the call and conference calls become active and the name of session, usually populated by the name of the soft phone client. Within the SDP, a number of variables are used, including v, o, s, t, and k. These all represent needed values within the SDP which are also available to be changed and include covert messages. The variables stand for:

V - Version, a field usually ignored by the session initiation protocol

O - The owner from which the message originated made up of random numbers

S - Session name, also ignored by the SIP and SDP

T - Time session active, ignored by the SIP and SDP

K - Potential encryption that is used within the SIP and SDP

Within these fields, each variable has caveats which allow for them to be changed or added to, which is the reason why they are unneeded and can be changed. The “v” or version field is the version of the Session Description Protocol being used within the current call. All version numbers are whole numbers with no “minor version numbers” [8]. Because of the communication of version number, so long as the version number specified is accepted at the receiving end of the VoIP call, the version can be changed to convey a message, assuming both parties establish a way not to convey accidental messages with automatically populated version numbers. The origin field, or “o” variable, is seen on line 12 of Figure 6, and has three sections of its field highlighted in bold. The origin is made up of the user name of the sending party, along with a session id and session version number. The session id, as of the writing of [8], has no uniform implementation, but rather has suggested ways that the equipment processing the VoIP call can use, specifically the suggestion of using a Network Time Protocol (NTP) as the session id to allow for calls to be more accurate based on the time sent and received. The version number is similar to the session id, in that it is randomly assigned, and is only used to establish which message has been sent most recently, especially when looking into proxy announcements. This allows for the receiving party to determine which messages are most

recent, and is therefore also recommended to utilize an NTP timestamp, for the same reasons as the session ID.

The session description protocol is actually quite similar to the SIP in terms of the fields within both. In the first field (line 11 of Figure 6) the version of the SDP protocol being used is displayed but ignored by the destination, other than to establish which version of the SDP is being used and therefore how it will be processed on the receiving end. Therefore, so long as the field is changed to a currently used and accepted version by the receiving party, and a version check is actually done by the receiving party, this version number can be changed to include some sort of single digit covert channel.

Similar to the version number, the owner is made up of a string field and two random identification numbers following it. As with the version number, these owner fields should exist within the SDP packet, but do not necessarily have to be whatever the system assigns to the original. While the end of this field will identify the domain from which the call originated, the owner and two identification numbers can be changed to send a covert message. The session name (variable s) and time the session is activated (variable t) will have a similar quality, while using t to send a covert message may border on being a time based covert channel, with the others being storage based, based on their existence within the protocol when sent. Both of these fields are theoretically able to be modified to send covert channels, mainly because they are made up of assigned strings of characters, or a random string of numbers, respectively [8].

A field which does not exist in Figure 6, but is claimed to be available by the RFC is “i” which is the session description. This description is user defined and can be a string of characters input

by the user, and therefore would be an obvious place to look for covert channels if analyzing the packet. A special case to be analyzed is k. K is the encryption protocol used while sending the SDP, a field which is mostly unused, but can be used when the need for encryption of VoIP call itself arises. Within Figure 6, line 16 shows the k variable as it would naturally be without encryption, stating, as its first string, ”clear” meaning there is no encryption being implemented. While there are very few sources that clearly explain how the encryption or decryption is done when implementing it over a covert channel, the assumption of the author is that both parties can either verify that the key is the same on both ends of the call, to ensure that the message was not intercepted, or the encryption key can be used for some type of software that is implemented to encrypt and decrypt the call for the source and destination, respectively. The key may also be entered by the receiving party, as briefly covered in [8].

3. 3 RTP

In addition to SDP and SIP, there is also the ability for real time transfer protocol, or RTP, to be used within the realm of covert channels as well. Because of the nature of RTP and the need for a codec to be used within it to compress and decompress the voice of the sender/receiver, there are actually two different ways to embed a covert channel within the protocol itself. The first is demonstrated briefly in Figure 7 [26], but still requires a bit of explanation. When the call is established between the source and destination, the invite has been accepted and the process to condition the line to allow for voice transmission begins. This conditioning consists of allowing for the sender to speak into the VoIP device that is available on their end and have their voice compressed into a digital signal. When the signal is compressed using a codec, it is encoded into a stream of bits and then has the ability to be blended with another audio source

if so chosen. The authors of [26] choose to do exactly this, and blend the voice stream as captured from the VoIP device with another digital audio track. In this way, when the two parties are having a conversation, there is another voice track, in the case of [26] a music track, but this can also be another voice, which is playing. This secondary track would be what a third party would hear if they tried to tap the line.

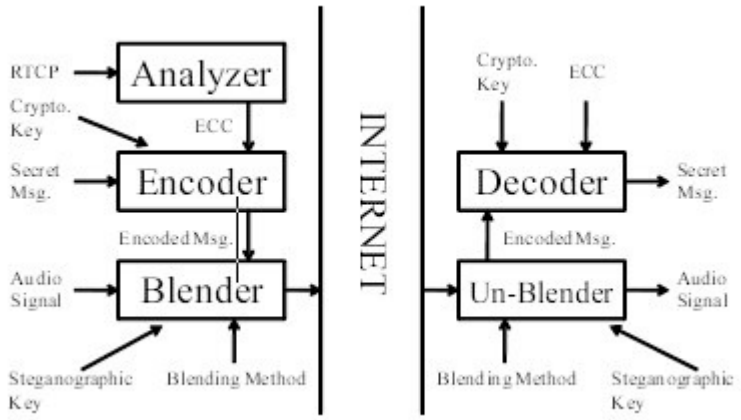


Figure 7: Blending of the original audio signal and another audio signal to embed a privacy covert channel

The second ability for use of RTP covert messages is the use of modified fields, similar to SIP and SDP. As can be seen in Figure 8 [16], a number of fields are labeled optional or recommended and allow for the modification of them to allow for a cover channel to be embedded within the packet itself. The advantage to using this over the blended audio stream is the ability for multiple messages to be sent, one within each packet that is sent out throughout the VoIP session. Because of the nature of the VoIP session, multiple packets are sent out throughout the VoIP call, which allows for a new message to be sent out each time an RTP packet is sent.

When it comes to the actual fields that can be used, there are four main parts of the protocol that can be utilized. The padding field is the first which can be utilized, mainly because it is used for encryption algorithms when implemented. If the encryption algorithm is not implemented, the padding field will be set to 0 and a length of the padding field can be set as the last octet of the padding itself. This octet will tell the destination how many fields to ignore when the data is received, including itself. This is also the situation with the extended header, which is set in field 3 of the first line, similar to the padding field being set to true or false. Similar to SIP, the RTP packet contains the ability to modify the sequence field, a relatively random field which specifies the order in which the packet was sent. In addition, the time field can also be used, but must be used on the first packet because this will establish what the value of the field in subsequent packets will contain. Interestingly, the timestamp field can also be used in a second way to send covert messages. The value of the timestamp field determines when the packet left the source and will be analyzed by the destination when it arrives.

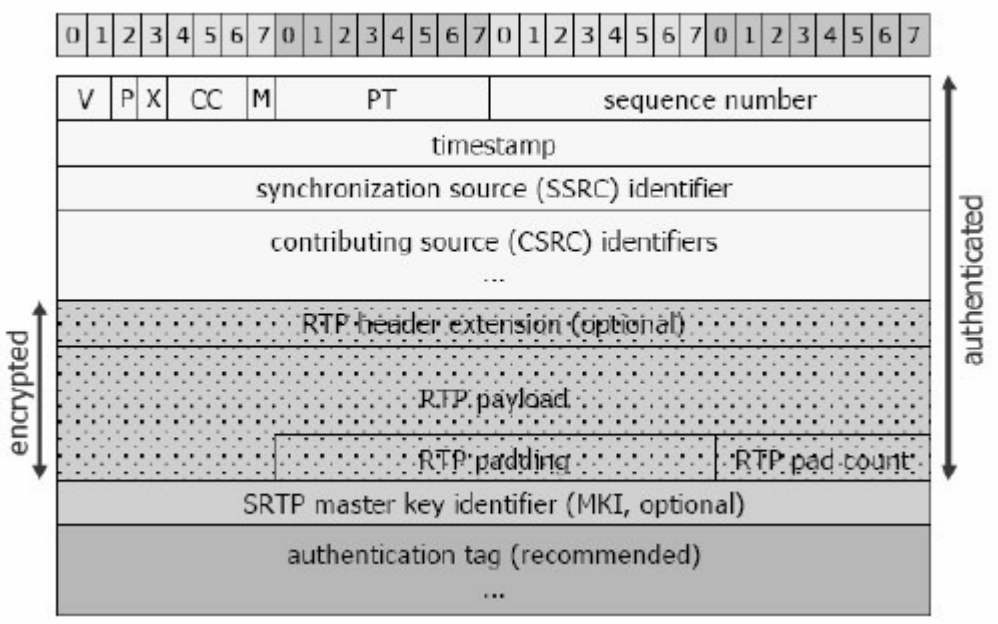


Figure 8: Illustration of the possibility to modify the RTP protocol to send a covert message

The authors of [14] propose that if a packet is sent from the source to the destination but is somehow delayed in transit, or the time field modified to represent a type of delay in transit, the packet will still make its way to the destination, but will be dropped and not processed in the VoIP session itself. While this has to be done with great care because of the limited threshold of error that a VoIP stream can handle, there is still the possibility to change the timestamp field resulting in a covert message being able to be hidden inside. Within the timestamp field there is also a least significant bit which can be modified to send covert messages as well.

In addition to being able to modify the timestamp while using RTP, there is also the ability to modify the stream itself and embed datagrams into the stream instead of actual voice. This, again, has to be used with great care, because of the error threshold that each codec, which will be sending and receiving the voice data, has. For example, the voice codec G723.1 has a 1 percent error capability. This may cause problems because of such a small error rate, as compared to G729A which allows for up to a 3 percent error rate. While neither is ideal for sending an excessively large message to recipients, they both add an additional possibility of covert channels [14].

4. Experimental Setup

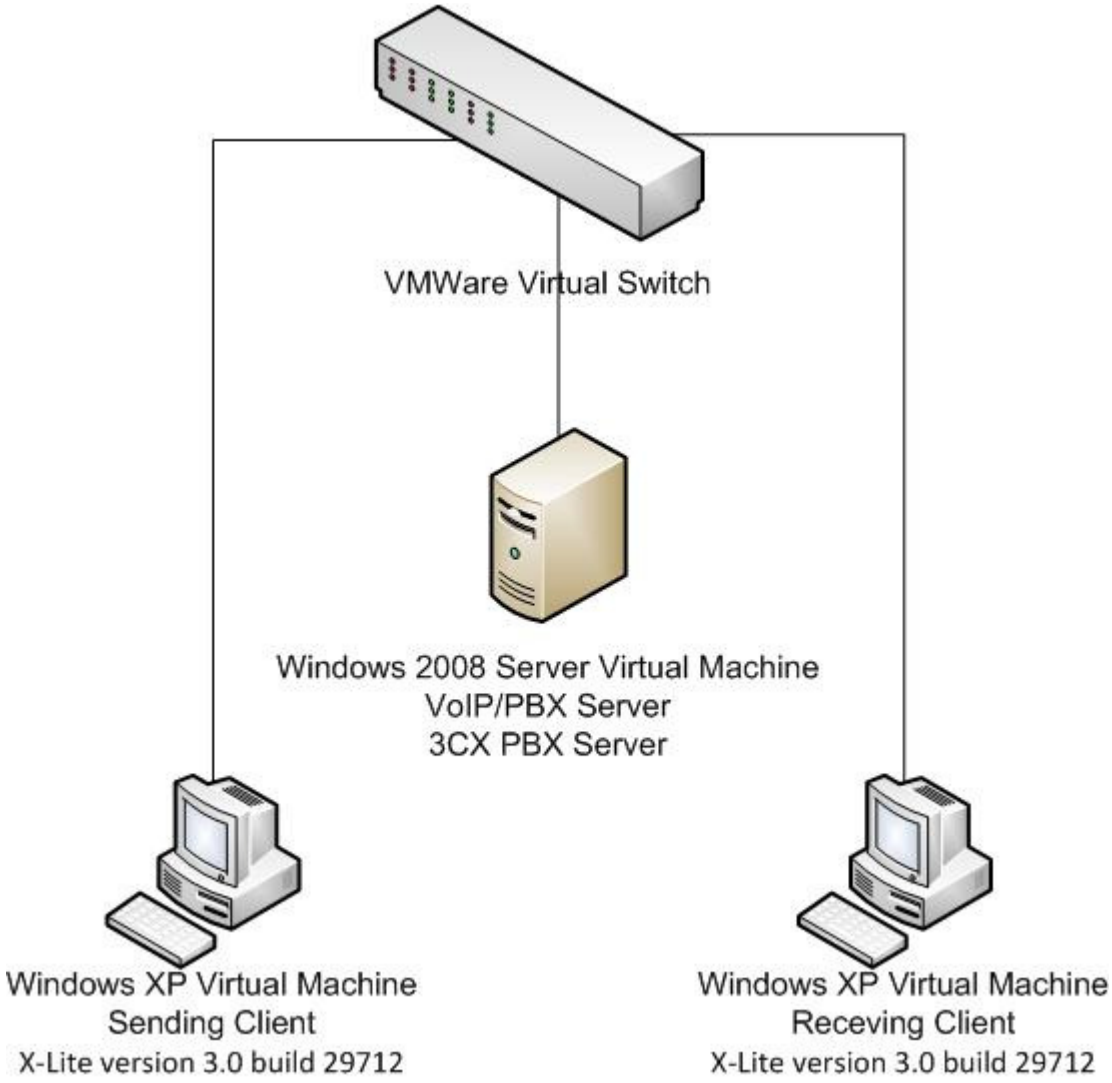
While much research was done to explain the types of covert messages and locations where covert messages could be hidden, there was no evidence that was found to demonstrate that

these experiments were actually carried out. For this thesis, these experiments were carried out within a virtual environment with three virtual machines, one acting as a server and the other two as clients. For the virtual clients, VMWare Workstation 6.5.1 was utilized under an academic license registered to Rochester Institute of Technology. Each client consisted of a Windows XP SP3 operating system, each running with 512MB of RAM AND 64 GB Maximum capacity for their hard drives. Both clients were licensed under the Microsoft Academic Alliance Licensing. For the phones used in this experiment, the decision was made to use software phones to keep the cost to a minimum and to ensure that the SIP protocol was actually used, rather than proprietary protocols used commonly with hardware based phones such as Polycom [20]. With this in mind, X-Lite version 3.0 build 29712 was used on both clients, which used the SIP protocol and also added an extra field to the SDP which will turn out to be beneficial later in the experiment.

For the server, Windows 2008 64 bit edition was used and licensed under Microsoft Academic Alliance Licensing, and was given 1024 MB of RAM and 64GB of maximum hard drive space. The server ran its own PBX, 3CX PBX specifically, which allows for a free server to be installed on any server, interface with an outside voice over IP provider and be managed by a user with little or no experience [1]. It is web console based and allows for individual phones to be added and assigned unique identities, extensions and properties. When connected, clients are registered into the 3CX system and are passed these identities.

As can be seen in Figure 6 and proposed in the original experiment, the authors of [6] proposed that within the SIP and SDP there are a number of fields that can be changed to embed covert

messages. In Figure 6, each field that is bolded was proposed to be able to carry a covert message, without affecting the outcome of the call or the connection that is made between the sender and the receiver. In the following sections, each one of these fields will be examined with the outcome that was produced when the experiment was carried out.



5. Results

```
3       0.0.0.0:42208          :0                      903        SendTo
  0000  49 4E 56 49 54 45 20 73 69 70 3A 31 30 30 40 77    INVITE sip:100@w
  0010  6F 72 6B 67 72 6F 75 70 20 53 49 50 2F 32 2E 30    orkgroup SIP/2.0
  0020  0D 0A 56 69 61 3A 20 53 49 50 2F 32 2E 30 2F 55    ..Via: SIP/2.0/U
  0030  44 50 20 31 30 2E 33 37 2E 31 2E 33 3A 34 32 32    DP 10.37.1.3:422
  0040  30 38 3B 62 72 61 6E 63 68 3D 7A 39 68 47 34 62    08;branch=z9hG4b


4       :0                     0.0.0.0:42208          306        RecvFrom
```

```
00000000 49 4E 56 49 54 45 20 73 69 70 3A 31 30 30 40 77 INVITE sip:100@w
00000010 6F 72 6B 67 72 6F 75 70 20 53 49 50 2F 32 2E 30 orkgroup SIP/2.0
00000020 0D 0A 56 69 61 3A 20 53 49 50 2F 32 2E 30 2F 55 ..Via: SIP/2.0/U
00000030 44 50 20 31 30 2E 33 37 2E 31 2E 33 3A 34 32 32 DP 10.37.1.3:422
00000040 30 38 3B 62 72 61 6E 63 68 3D 7A 39 68 47 34 62 08;branch=z9hG4b
00000050 4B 2D 64 38 37 35 34 33 2D 36 34 36 33 38 32 36 K-d87543-6463826
00000060 35 61 39 37 33 37 64 34 65 2D 31 2D 2D 64 38 37 5a9737d4e-1--d87
00000070 35 34 33 2D 3B 72 70 6F 72 74 0D 0A 4D 61 78 2D 543-;rport..Max-
00000080 46 6F 72 77 61 72 64 73 3A 20 37 30 0D 0A 43 6F Forwards: 70..Co
00000090 6E 74 61 63 74 3A 20 3C 73 69 70 3A 31 30 31 40 ntact: <sip:101@
000000A0 31 30 2E 33 37 2E 31 2E 33 3A 34 32 32 30 38 3E 10.37.1.3:42208>
000000B0 0D 0A 54 6F 3A 20 22 31 30 30 22 3C 73 69 70 3A ..To: "100"<sip:
000000C0 31 30 30 40 77 6F 72 6B 67 72 6F 75 70 3E 0D 0A 100@workgroup>..
000000D0 46 72 6F 6D 3A 20 22 53 6F 66 74 70 68 6F 6E 65 From: "Softphone
000000E0 32 22 3C 73 69 70 3A 31 30 31 40 77 6F 72 6B 67 2"<sip:101@workg
000000F0 72 6F 75 70 3E 3B 74 61 67 3D 31 34 37 32 62 65 roup>;tag=1472be
00000100 33 32 0D 0A 43 61 6C 6C 2D 49 44 3A 20 66 31 30 32..Call-ID: f10
00000110 39 32 33 37 31 66 65 36 34 30 35 35 36 59 7A 45 9237lfe640556YzE
00000120 32 5A 6A 6C 69 4F 44 52 6C 4D 54 51 77 5A 6D 52 2ZjliODRlMTQwZmR
00000130 6C 4D 6A 6C 6B 4F 47 59 34 4E 57 49 35 4E 7A 49 lMjlkOGY4NWI5NzI
00000140 78 4D 47 49 79 4E 47 51 2E 0D 0A 43 53 65 71 3A xMGIyNGQ...CSeq:
00000150 20 31 20 49 4E 56 49 54 45 0D 0A 41 6C 6C 6F 77  1 INVITE..Allow
00000160 3A 20 49 4E 56 49 54 45 2C 20 41 43 4B 2C 20 43 : INVITE, ACK, C
00000170 41 4E 43 45 4C 2C 20 4F 50 54 49 4F 4E 53 2C 20 ANCEL, OPTIONS,
00000180 42 59 45 2C 20 52 45 46 45 52 2C 20 4E 4F 54 49 BYE, REFER, NOTI
00000190 46 59 2C 20 4D 45 53 53 41 47 45 2C 20 53 55 42 FY, MESSAGE, SUB
000001A0 53 43 52 49 42 45 2C 20 49 4E 46 4F 0D 0A 43 6F SCRIBE, INFO..Co
000001B0 6E 74 65 6E 74 2D 54 79 70 65 3A 20 61 70 70 6C ntent-Type: appl
000001C0 69 63 61 74 69 6F 6E 2F 73 64 70 0D 0A 55 73 65 ication/sdp..Use
000001D0 72 2D 41 67 65 6E 74 3A 20 58 2D 4C 69 74 65 20 r-Agent: X-Lite
000001E0 72 65 6C 65 61 73 65 20 31 30 30 32 74 78 20 73 release 1002tx s
000001F0 74 61 6D 70 20 32 39 37 31 32 0D 0A 43 6F 6E 74 tamp 29712..Cont
00000200 65 6E 74 2D 4C 65 6E 67 74 68 3A 20 33 37 32 0D ent-Length: 372.
00000210 0A 0D 0A 76 3D 30 0D 0A 6F 3D 2D 20 31 20 32 20 ...v=0..o=- 1 2
00000220 49 4E 20 49 50 34 20 31 30 2E 33 37 2E 31 2E 33 IN IP4 10.37.1.3
00000230 0D 0A 73 3D 3C 43 6F 75 6E 74 65 72 50 61 74 68 ..s=<CounterPath
00000240 20 65 79 65 42 65 61 6D 20 31 2E 35 3E 0D 0A 63  eyeBeam 1.5>..c
00000250 3D 49 4E 20 49 50 34 20 31 30 2E 33 37 2E 31 2E =IN IP4 10.37.1.
```

Figure 9: A clean SIP/SDP packet as captured by Winsock Packet Editor

5. 1 The branch statement

The first experiment that was run was on the first field noted in [6], or the branch statement. The branch statement is claimed to have to begin with a standardized phrase of “z9hG4bK” but has five characters after which it can be modified according to the authors, rather than the 29 found while carrying out the experiment. In the experimental setup, Wireshark was used in conjunction to Winsock Packet Editor to view the packets and record the results as the packets were edited. While using Winsock Packet Editor, it was found that there is the ability to successfully modify the last five characters of the branch statement by inputting the hex values of the required standardized branch statement and appending a message to them, also in hex format. In this case, z9hG4bK translates into 7A 39 68 47 34 62 4B which was the searched for string when inputting it into WPE.

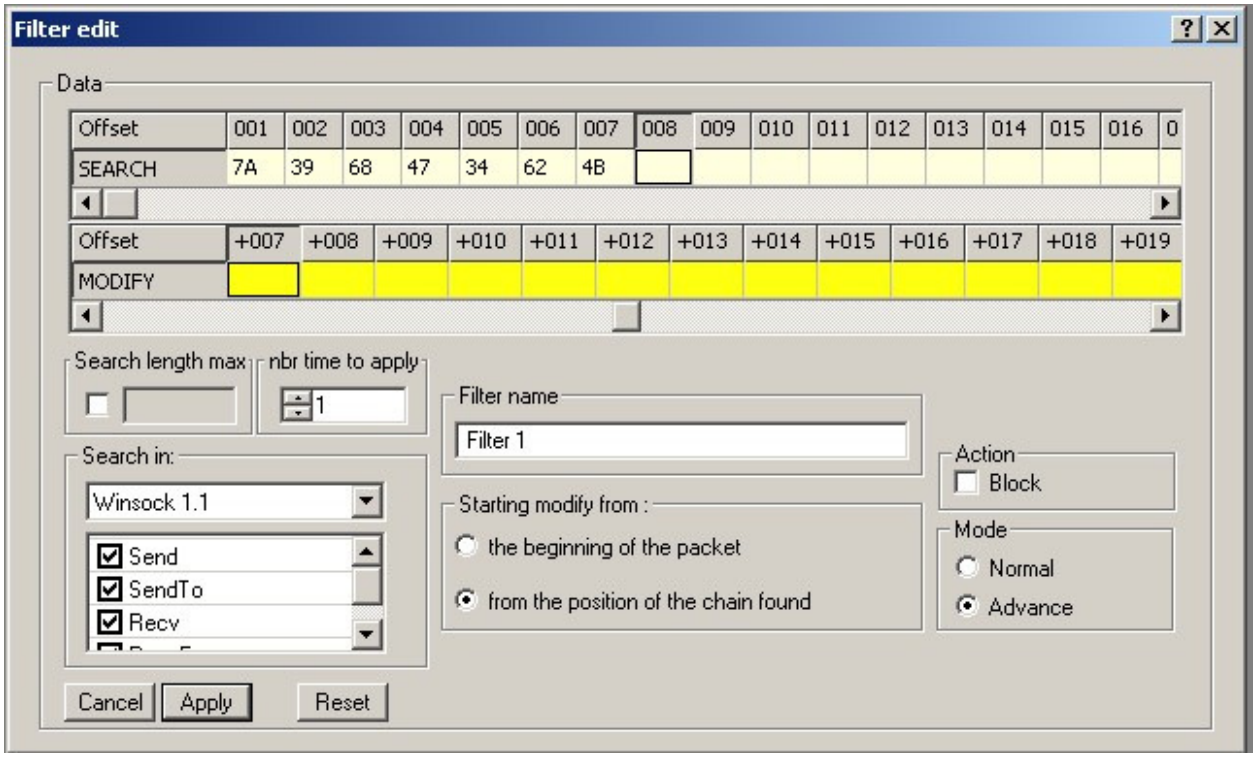


Figure 10: The Hex translation of the “z9hG4bK” Key in the Branch statement

The result of the modification of the branch statement was a successfully connected call for the receiver, but the client not being able to recognize that the call was connected and therefore timing out and dropping the call. According to packet capturing tools, as can be seen in Figure 11, the sender believes that because it was unable to receive a properly formed SIP/SDP packet back from the receiver that the session is not connected and in fact, does not exist. Notice in the Status-Line the message “SIP/2.0481 Call/Transaction Does Not Exist.”

Interestingly, the branch statement is much longer in experiments than originally proposed in [6]. Even with additional experiments run to test the first five characters after the phrase of “z9hG4bK,” the call continues to fail, thus disproving this field is able to carry a covert channel.

Figure 11: Example of a failed call as illustrated through a Wireshark capture of "Call/Transaction does not exist"

5. 2 Max-Forwards

The Max-Forwards line is usually set to a value of 70, indicating the number of times that the VoIP packet should be forwarded between routers while on its route to the destination/receiver. This value is small and may not be enough to send an overly complex covert message, but with the agreement between sender and receiver that the value should be 70 unless there is a covert message embedded in it, this could at least be an indicator that a covert message is embedded in the actual packet.

As can be seen in Figure 12, WPE was set up to filter and change the value of Max-Forwards, based on its hex value and change it up to ten times within any packets sent or received. In Figures 12 and 13, we can see that the packet was successfully changed and resulted in a completed call between the sender and receiver.

While there are drawbacks to using this field for embedding covert messages, namely that there are only two characters that can be used to send either a signal or message, as well as a non-standardized value to be placed in this field, this method of sending covert messages does work and allows for calls to be established and maintained. With careful planning and careful observation by both the sender and receiver, this could hide simple messages which could have dire effects.

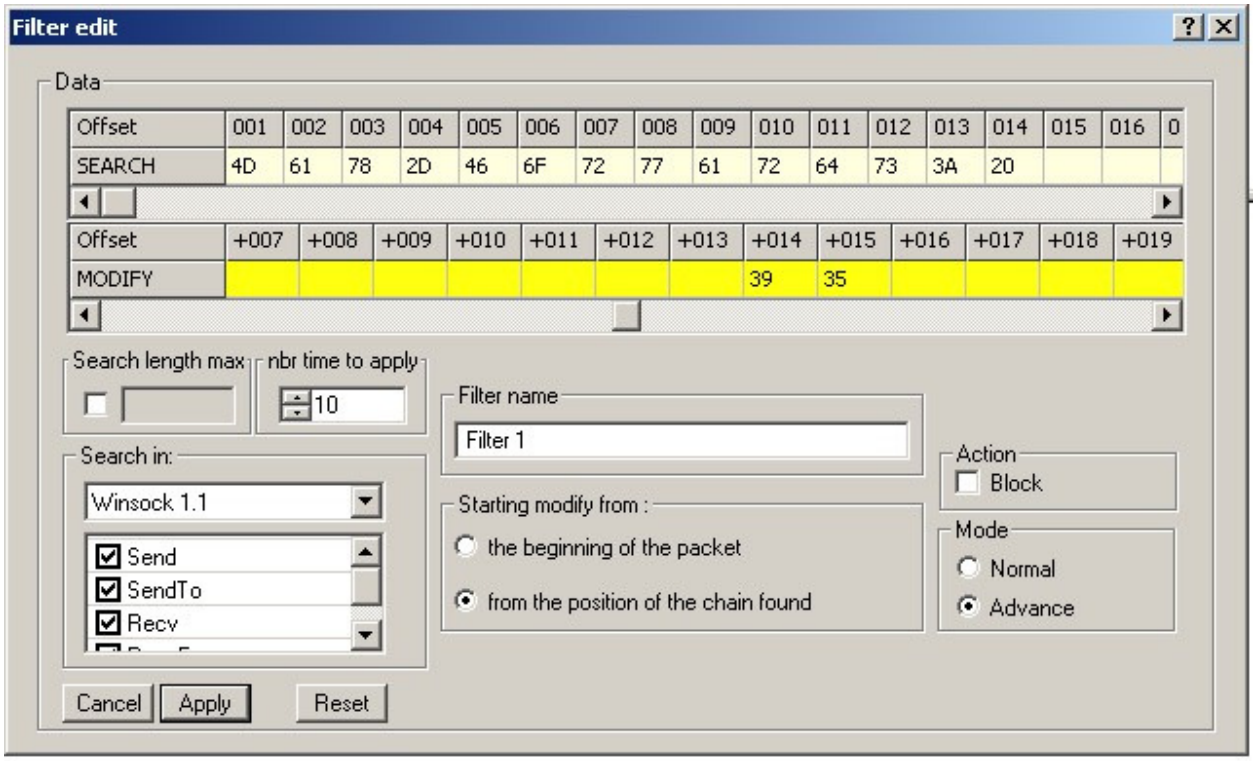


Figure 12: Change of Max-Forwards using the hex input feature of Winsock Packet Editor

Figure 13: A successful call as seen by both the sending and receiving clients

```
00000000 49 4E 56 49 54 45 20 73 69 70 3A 31 30 30 40 77 INVITE sip:l00@w
00000010 6F 72 6B 67 72 6F 75 70 20 53 49 50 2F 32 2E 30 orkgroup SIP/2.0
00000020 0D 0A 56 69 61 3A 20 53 49 50 2F 32 2E 30 2F 55 ..Via: SIP/2.0/U
00000030 44 50 20 31 30 2E 33 37 2E 31 2E 33 3A 35 31 39 DP 10.37.1.3:519
00000040 36 34 3B 62 72 61 6E 63 68 3D 7A 39 68 47 34 62 64;branch=z9hG4b
00000050 4B 2D 64 38 37 35 34 33 2D 30 35 30 33 66 64 36 K-d87543-0503fd6
00000060 65 65 38 30 63 32 30 34 32 2D 31 2D 2D 64 38 37 ee80c2042-1--d87
00000070 35 34 33 2D 3B 72 70 6F 72 74 0D 0A 4D 61 78 2D 543-;rport..Max-
00000080 46 6F 72 77 61 72 64 73 3A 20 39 35 0D 0A 43 6F Forwards: 95..Co
00000090 6E 74 61 63 74 3A 20 3C 73 69 70 3A 31 30 31 40 ntact: <sip:101@
000000A0 31 30 2E 33 37 2E 31 2E 33 3A 35 31 39 36 34 3E 10.37.1.3:51964>
000000B0 0D 0A 54 6F 3A 20 22 31 30 30 22 3C 73 69 70 3A ..To: "100"<sip:
000000C0 31 30 30 40 77 6F 72 6B 67 72 6F 75 70 3E 0D 0A 100@workgroup>..
000000D0 46 72 6F 6D 3A 20 22 53 6F 66 74 70 68 6F 6E 65 From: "Softphone
000000E0 32 22 3C 73 69 70 3A 31 30 31 40 77 6F 72 6B 67 2"<sip:101@workg
000000F0 72 6F 75 70 3E 3B 74 61 67 3D 61 64 32 66 36 39 roup>;tag=ad2f69
00000100 36 62 0D 0A 43 61 6C 6C 2D 49 44 3A 20 37 65 36 6b..Call-ID: 7e6
00000110 65 34 36 34 61 36 37 34 38 34 31 36 37 59 7A 45 e464a67484167YzE
00000120 32 5A 6A 6C 69 4F 44 52 6C 4D 54 51 77 5A 6D 52 2ZjliODRlMTQwZmR
00000130 6C 4D 6A 6C 6B 4F 47 59 34 4E 57 49 35 4E 7A 49 lMjlkOGY4NWI5NzI
00000140 78 4D 47 49 79 4E 47 51 2E 0D 0A 43 53 65 71 3A xMGIyNGQ...CSeq:
00000150 20 31 20 49 4E 56 49 54 45 0D 0A 41 6C 6C 6F 77  1 INVITE..Allow
00000160 3A 20 49 4E 56 49 54 45 2C 20 41 43 4B 2C 20 43 : INVITE, ACK, C
00000170 41 4E 43 45 4C 2C 20 4F 50 54 49 4F 4E 53 2C 20 ANCEL, OPTIONS,
00000180 42 59 45 2C 20 52 45 46 45 52 2C 20 4E 4F 54 49 BYE, REFER, NOTI
00000190 46 59 2C 20 4D 45 53 53 41 47 45 2C 20 53 55 42 FY, MESSAGE, SUB
000001A0 53 43 52 49 42 45 2C 20 49 4E 46 4F 0D 0A 43 6F SCRIBE, INFO..Co
000001B0 6E 74 65 6E 74 2D 54 79 70 65 3A 20 61 70 70 6C ntent-Type: appl
000001C0 69 63 61 74 69 6F 6E 2F 73 64 70 0D 0A 55 73 65 ication/sdp..Use
000001D0 72 2D 41 67 65 6E 74 3A 20 58 2D 4C 69 74 65 20 r-Agent: X-Lite
000001E0 72 65 6C 65 61 73 65 20 31 30 30 32 74 78 20 73 release 1002tx s
000001F0 74 61 6D 70 20 32 39 37 31 32 0D 0A 43 6F 6E 74 tamp 29712..Cont
00000200 65 6E 74 2D 4C 65 6E 67 74 68 3A 20 33 37 32 0D ent-Length: 372.
00000210 0A 0D 0A 76 3D 30 0D 0A 6F 3D 2D 20 37 20 32 20 ...v=0..o=- 7 2
00000220 49 4E 20 49 50 34 20 31 30 2E 33 37 2E 31 2E 33 IN IP4 10.37.1.3
00000230 0D 0A 73 3D 3C 43 6F 75 6E 74 65 72 50 61 74 68 ..s=<CounterPath
00000240 20 65 79 65 42 65 61 6D 20 31 2E 35 3E 0D 0A 63  eyeBeam 1.5>..c
00000250 3D 49 4E 20 49 50 34 20 31 30 2E 33 37 2E 31 2E =IN IP4 10.37.1.
```

Figure 14: Illustration of Max-Forwards changed to 95

5. 3 The tag Field

Similar to the previous fields, Max-Forwards and Branch, the tag field also had the problem of allowing for the call to connect to the other side, but not being recognized on the sender's side as being connected. This is thought to be because of a malformed packet when an acknowledgement is received from the receiving side at the sender side. In tests that were run

on the tag field, similar results were returned as in the error found while testing the branch statement, in which “Transaction does not exist,” was returned.

5. 4 Call-ID Field

The Call-ID field was tested multiple times with multiple values due to a number of different examples and theories by both source authors and this experiment’s author. The Call-ID field in [6] was full of only numbers and thus was tested as such in the first experiment. This resulted in a similar result as the previous fields in which the call went through as seen by the receiver, but an error of “Transaction does not exist” was returned to the sender (in packet form).

In contrast to what was seen in [6], during the experiment it was observed that the Call-ID field was actually made up of numerous alphanumeric characters, as can be seen in the image below.

Therefore, the Call-ID was tested a second time with multiple alphanumeric characters, specifically the phrase “dropthebomb” embedded into the message. With the observation of this also not working, the idea that the first character of the line might also need to remain the same, or at least that of a hex character, as it was in each of the tests. Therefore, an additional test of allowing the first character to remain unchanged, but subsequent characters modified was performed, but also failed. With this change, the Call-ID field read “Edropthebomb” and still failed to connect. Finally, the theory that characters within the Call-ID field must be hex based characters (1..9, a…f) was experimented with, but again resulted in the same conclusion.

5. 5 CSeq

In the testing environment, a discrepancy was seen as compared to [6] in that the CSeq variable was only one numeric character, rather than 5 as proposed in the original source. This is proposed to be because of the very limited number of clients that were involved in the experiment, and the very small number of calls that were made on either client in the experiment. The CSeq number, according to the [8], is incremented based on the number of dialogs per session. Therefore, while this was tested and successfully proved to be a viable candidate for embedded covert channels, rather than multiple numerical characters being able to be embedded into the stream (5 in the example of [6]), it was only a single character. While this may be limiting in the types of covert messages that could be embedded, there would be the ability to at least signal the receiver of the presence of a covert channel.

5. 6 From

From seemed to be one of the most useful fields in the experiment because of its valid ability to be changed and have covert messages embedded in it, as well as how it works when picked up by the receiving client. The From field is in essence the caller id field that would be seen in the average telephone call, and indicates to the receiver who is calling through the display of the softphone as seen in Figure 15. As can be seen, the caller ID field was changed to "Sdropbomb2" from "Softphone2" while keeping the contact information such as email and response address the same, and therefore allowed for an immediate covert message to be popped up on the screen. This call was able to complete successfully, as evidenced in the graphic below, and would therefore be a viable candidate for covert messages. While there is

the ability for this field to be changed, the caveat that any onlookers can easily see a covert channel on receipt should be carefully evaluated before using this field.

Figure 15: Change in the From field (similar to the Caller-ID)

```
2       0.0.0.0:42208          :0                      903        SendTo
  0000  49 4E 56 49 54 45 20 73 69 70 3A 31 30 30 40 77    INVITE sip:100@w
  0010  6F 72 6B 67 72 6F 75 70 20 53 49 50 2F 32 2E 30    orkgroup SIP/2.0
  0020  0D 0A 56 69 61 3A 20 53 49 50 2F 32 2E 30 2F 55    ..Via: SIP/2.0/U
  0030  44 50 20 31 30 2E 33 37 2E 31 2E 33 3A 34 32 32    DP 10.37.1.3:422
  0040  30 38 3B 62 72 61 6E 63 68 3D 7A 39 68 47 34 62    08;branch=z9hG4b

3       :0                      0.0.0.0:42208          306        RecvFrom
  0000  53 49 50 2F 32 2E 30 20 31 30 30 20 54 72 79 69    SIP/2.0 100 Tryi
  0010  6E 67 0D 0A 56 69 61 3A 20 53 49 50 2F 32 2E 30    ng..Via: SIP/2.0
  0020  2F 55 44 50 20 31 30 2E 33 37 2E 31 2E 33 3A 34    /UDP 10.37.1.3:4
  0030  32 32 30 38 3B 62 72 61 6E 63 68 3D 7A 39 68 47    2208;branch=z9hG
  0040  34 62 4B 2D 64 38 37 35 34 33 2D 30 35 33 35 33    4bK-d87543-05353
```

```
00000000 49 4E 56 49 54 45 20 73 69 70 3A 31 30 30 40 77 INVITE sip:100@w
00000010 6F 72 6B 67 72 6F 75 70 20 53 49 50 2F 32 2E 30 orkgroup SIP/2.0
00000020 0D 0A 56 69 61 3A 20 53 49 50 2F 32 2E 30 2F 55 ..Via: SIP/2.0/U
00000030 44 50 20 31 30 2E 33 37 2E 31 2E 33 3A 34 32 32 DP 10.37.1.3:422
00000040 30 38 3B 62 72 61 6E 63 68 3D 7A 39 68 47 34 62 08;branch=z9hG4b
00000050 4B 2D 64 38 37 35 34 33 2D 30 35 33 35 33 34 34 K-d87543-0535344
00000060 38 31 64 32 32 37 38 37 33 2D 31 2D 2D 64 38 37 81d227873-1--d87
00000070 35 34 33 2D 3B 72 70 6F 72 74 0D 0A 4D 61 78 2D 543-;rport..Max-
00000080 46 6F 72 77 61 72 64 73 3A 20 37 30 0D 0A 43 6F Forwards: 70..Co
00000090 6E 74 61 63 74 3A 20 3C 73 69 70 3A 31 30 31 40 ntact: <sip:101@
000000A0 31 30 2E 33 37 2E 31 2E 33 3A 34 32 32 30 38 3E 10.37.1.3:42208>
000000B0 0D 0A 54 6F 3A 20 22 31 30 30 22 3C 73 69 70 3A ..To: "100"<sip:
000000C0 31 30 30 40 77 6F 72 6B 67 72 6F 75 70 3E 0D 0A 100@workgroup>..
000000D0 46 72 6F 6D 3A 20 22 53 64 72 6F 70 62 6F 6D 62 From: "Sdropbomb
000000E0 32 22 3C 73 69 70 3A 31 30 31 40 77 6F 72 6B 67 2"<sip:101@workg
000000F0 72 6F 75 70 3E 3B 74 61 67 3D 39 61 34 63 30 36 roup>;tag=9a4c06
00000100 36 63 0D 0A 43 61 6C 6C 2D 49 44 3A 20 36 32 30 6c..Call-ID: 620
00000110 34 37 32 37 64 39 34 30 38 39 38 32 66 59 7A 45 4727d9408982fYzE
00000120 32 5A 6A 6C 69 4F 44 52 6C 4D 54 51 77 5A 6D 52 2ZjliODRlMTQwZmR
00000130 6C 4D 6A 6C 6B 4F 47 59 34 4E 57 49 35 4E 7A 49 lMjlkOGY4NWI5NzI
00000140 78 4D 47 49 79 4E 47 51 2E 0D 0A 43 53 65 71 3A xMGIyNGQ...CSeq:
00000150 20 31 20 49 4E 56 49 54 45 0D 0A 41 6C 6C 6F 77  1 INVITE..Allow
00000160 3A 20 49 4E 56 49 54 45 2C 20 41 43 4B 2C 20 43 : INVITE, ACK, C
00000170 41 4E 43 45 4C 2C 20 4F 50 54 49 4F 4E 53 2C 20 ANCEL, OPTIONS,
00000180 42 59 45 2C 20 52 45 46 45 52 2C 20 4E 4F 54 49 BYE, REFER, NOTI
00000190 46 59 2C 20 4D 45 53 53 41 47 45 2C 20 53 55 42 FY, MESSAGE, SUB
000001A0 53 43 52 49 42 45 2C 20 49 4E 46 4F 0D 0A 43 6F SCRIBE, INFO..Co
000001B0 6E 74 65 6E 74 2D 54 79 70 65 3A 20 61 70 70 6C ntent-Type: appl
000001C0 69 63 61 74 69 6F 6E 2F 73 64 70 0D 0A 55 73 65 ication/sdp..Use
000001D0 72 2D 41 67 65 6E 74 3A 20 58 2D 4C 69 74 65 20 r-Agent: X-Lite
000001E0 72 65 6C 65 61 73 65 20 31 30 30 32 74 78 20 73 release 1002tx s
000001F0 74 61 6D 70 20 32 39 37 31 32 0D 0A 43 6F 6E 74 tamp 29712..Cont
00000200 65 6E 74 2D 4C 65 6E 67 74 68 3A 20 33 37 32 0D ent-Length: 372.
00000210 0A 0D 0A 76 3D 30 0D 0A 6F 3D 2D 20 35 20 32 20 ...v=0..o=- 5 2
00000220 49 4E 20 49 50 34 20 31 30 2E 33 37 2E 31 2E 33 IN IP4 10.37.1.3
00000230 0D 0A 73 3D 3C 43 6F 75 6E 74 65 72 50 61 74 68 ..s=<CounterPath
00000240 20 65 79 65 42 65 61 6D 20 31 2E 35 3E 0D 0A 63  eyeBeam 1.5>..c
00000250 3D 49 4E 20 49 50 34 20 31 30 2E 33 37 2E 31 2E =IN IP4 10.37.1.
```

Figure 16: Change in the From field as seen by the Winsock Packet Editor

5. 7 SDP V field

THE V field as found in the SIP is usually the version of the SIP protocol being used, to ensure compatibility between different processing systems, i.e., different VoIP phones. This field is generally ignored now that [8] containing information about SIP has been officially accepted and thus was theorized as able to be changed with no adverse effects on the call itself. Through experimentation and editing the V field as based on a "v=" keyword, it was found that the SDP V field was successfully changed to 4 rather than 0 with no effect on the call whatsoever. Although this was done using the client on either side of the VoIP exchange, there should be no adverse effects on the call even using a different receiving software client due to evidence that this field is ignored during the initiation.

5. 8 SDP O field

The O field is used to identify the owner from which a message originated and is formatted in a <user><number><number> format. It seemed while conducting the experiment that these fields were made up of id numbers or time stamps and therefore when they were changed, caused the call to fail. In the original source [6], the authors proposed that this field should be ignored by the receiving client, but when these numbers were changed to different numbers, the call connected on the receiving client's end and failed on the sending client's end, reporting back a "transaction does not exist" error, similar to previously tested fields.

5. 9 SDP S field

The S field is the field of the SIP which gives a session name to the VoIP transfer. Unlike other fields such as the tag or Call-ID field, the S field is ignored and only used when the client is able to read the plain text session name and display it. In the experiment, the S field was ignored and therefore successfully modified with no affect on the call whatsoever. The session name was changed to the common “dropthebomb” phrase that was used throughout the experiments to ensure the conformity with a standard alpha based phrase.

5. 10 SDP T field

Similar to the CSeq field, the T field is used to determine the time that the session became active, both on the sending and receiving ends. Therefore, there are two numerical fields that are used for information with a space between them. According to [6] these fields are supposed to be ignored and therefore not affect the call if changed. When changing the T fields, the call completed successfully with the standard values of “0 0” being changed to “8 9”. The call connected on both ends and the covert channel was sent without interfering with the call.

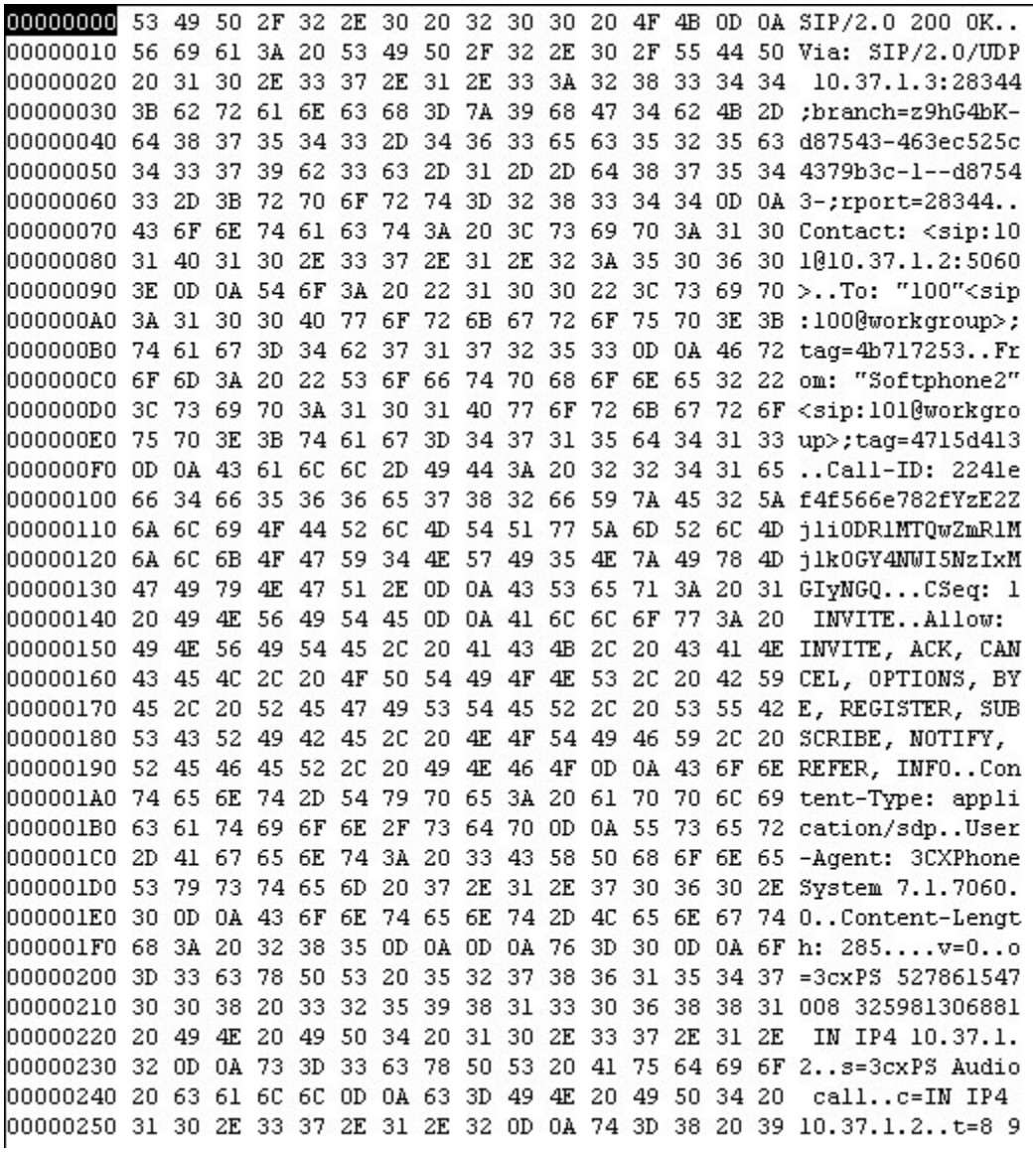


Figure 17: Change in the T field from “0 0” to “8 9”

5. 11SDP K field

While carrying out the experiment, it seemed odd that a K field was not found while monitoring the packets, either while monitoring them with WPE or with Wireshark. When more research was done into this area it was found that the K field may only exist in the SIP’s of clients which support an encrypted call to be made. With free software based phones, this was not possible

and thus there was no K field to modify. A future version of this experiment could be done to try to demonstrate the abilities to hide covert channels within encrypted Voice over IP calls, but is currently cost prohibitive and outside the scope of this overall experiment.

6. Further Discussion

Unlike the RTP transport protocol that is used to carry the actual voice data to the receiving end during the VoIP session, and which also has the ability to carry covert messages within it [5] the SIP and SDP have the advantage of requiring an acknowledgement message to be sent before they will allow for voice data to be sent. While the covert channel that is being sent within the SIP and SDP has to be small because of the limited amount of room within the packet, there is the advantage that the sender will know that the receiver has received the covert message because of the acknowledgement that is returned once the SIP and SDP packets have been received and accepted. Interestingly, this goes against the basic properties of User Datagram Protocol (UDP), which is what VoIP is said to communicate over. UDP's property of making a "best effort" attempt to send the message to the receiving party would usually make for a poor communication medium/protocol to send covert messages over. In contrast, SIP and SDP use the acknowledgement system which makes them a great message carrier.

Another advantage that this method has as compared to other covert channels, including those used in RTP, is that the covert message being sent has to be detected at the very beginning of the call. If a packet capture were to begin after the phone has rung and been picked up, the SIP and SDP packets will have already been sent and there is only a very small likelihood that additional messages will be contained in the maintenance SDP packets. This also lends itself to

the advantage of RTP packets being seen as UDP packets if the control packet, or RTCP packet, is not seen at the beginning of the call. Therefore, they would be indistinguishable from any other average UDP packet being sent across the network.

With these advantages, though, come disadvantages as well. As was seen in the experiments, if the acknowledgement packet is not sent, usually because the SIP or SDP packet is malformed, the call cannot complete and no data is able to be sent. While this does not impact the result of being able to send a covert message across the line in the SIP or SDP packet, it does cause a bit of suspicion if observed by a systems administrator. Given the wealth of knowledge available regarding covert messages and this thesis, a systems administrator could continuously monitor that extension or soft phone for variances from the packet norm.

This leads to another disadvantage: currently to detect a covert channel within the SIP and SDP packets, a systems administrator would have to run a packet capture utility on the VoIP device and continuously monitor until a covert message is sent. Because the call may not come at a specific time or interval, this could be extremely time consuming both in capturing and the effort to monitor VoIP logs.

This thesis also does not account for the ability for a single message to be carried within SIP or SDP packets over multiple calls, or in terms of cover channels “an event based channel over time.” While the tool provided in the following sections does account for any SIP or SDP packet which may have changed values, it is up to human logic to detect the full string through which the message is sent. This could include piecing together the values of Max-Forwards, or any

other SIP or SDP field for that matter, which presents a danger solely based on the time and effort needed to do so.

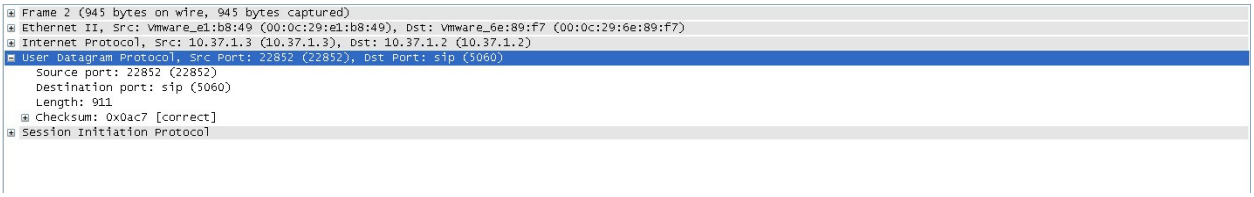


Figure 18: Illustration of VoIP being seen as UDP within Wireshark

7. Entropy

In any communication medium in which there is a message being sent, especially between multiple carriers, we have to worry about entropy. Entropy, put simply, is the uncertainty that the message that is received on the receiving end, is different than that sent from the sending end of the medium. In the realm of probability, entropy is the idea that there is a certain variable which the experimenter is oblivious to, and thus there is a possibility of multiple results occurring the more times the experiment is repeated. Take for example the simple game of a coin being flipped. If we assume the coin is fair and both sides are weighted the same as to produce an equal probability of landing upward, we can confidently say there is a 1 in 2 (1/2) probability that either side will face upward on the coin's landing. However, if we do not have the confidence that a coin is fair and both sides are weighted evenly, another variable is added into the equation. This variable is actually two fold in which we now have a probability of each side landing, thus probability p for heads and q of tails [6].

In the first example in which both sides had an equal chance of facing upward when the coin lands, we say that the coin has an entropy of 1. There is the possibility of either side facing upward because the coin is even and the outcome of either side facing upward is only left up to chance. In the second example of the coin in which there are two probabilities, different possibilities for either side to face upward when the coin lands, we can confidently say that there is an entropy of less than 1, but an exact value is impossible to come up with without the exact probabilities. In a final case that could be made in which the coin does not have a tails side, but rather is double-headed, we can confidently say that the coin has an entropy value of zero because there is only a single possibility and thus no unknown variable of chance which can be factored into the coin toss.

Entropy was originally proposed by Claude E. Shannon in his paper "A Mathematical Theory of Communication," published in 1948 [6]. This paper was based on the work of Lazare Carnot, who originally coined the concept of entropy as it applied to the concept of energy in a working system, in which a certain amount of energy is lost during work in the form of heat [9]. This concept can easily be seen in the example of friction, especially for anyone who has used a grinding wheel to grind metal or sharpen a blade. When the grinding wheel first starts running, it is cold and wastes very little energy while running on its own. When the wheel comes in contact with the piece of metal, wasteful energy can be seen visually in the way of sparks coming off of the metal and through touch due to the heat that is dissipated into both the wheel and the blade itself.

Another more relevant example of entropy in the communications field is the childhood game known as “telephone.” In the game, a number of people line up and a message is verbally given to the first member of the line in a way that will not allow for the rest of the members to hear. This person is given the task to pass this message to the next person in line in the same manner, but cannot repeat the message once it has been given. At the final person (n-1), the message is repeated out loud and an evaluation of the difference between the original message and the one which is received is done. This example is quite similar to the idea of embedding a covert channel, because is it very similar to the process through which voice over IP works. For each member that the message has to travel through in the telephone line, there is a higher probability that the message will not be the exact same one as the message which originally entered the telephone system. Therefore, there is also the ability that each member may embed unneeded data into the message, which may not even be recognizable to the other members, and thus send a covert message either to a member later on in the telephone chain, or just send a false message to the final member of the telephone chain. For each member through which the message has to pass, there is a 50% chance that the exact message understood by them will be that which their sender sent, and thus for each new person that is added to the chain, the probability that the message will continue to be correct at the final receiver is $1/2^{n-1}$. This is exactly the problem with voice over IP and why they are so susceptible to embedded covert messages.

Within each field of the voice over IP communication, there is the ability to compare entropy by viewing the sent message to the acknowledgement message that is received by the second party. This limits the ability for a covert channel to exist undetected once it leaves the sending

party (as would be the case in a man in the middle attack), but when it comes to covert channels purposely sent by the sender, there is the need for analysis of the field as compared to what it should be as based on default settings or common settings which have either no ability or rare ability to be changed on the SIP server itself. In the following paragraphs, it will be attempted to evaluate the amount of entropy in each field that was found to be able to be changed within the original experiment. Because of the needed equations, which will be given in the following paragraphs, entropy will not always be a value between zero and one, and it will be assumed that there are only two parties involved in the call as to not introduce uncertain variables from multiple other parties.

7.1 Branch

The Branch field is a unique identifier that is used by both the sender and the receiver in the form of the Via statement. When a request is made for a client to join a call, in the way of an INVITE packet being sent, both clients will use the same assigned Branch statement throughout the call. The Branch statement must be unique in both the way of the current call and all past and future calls, and must begin with the characters “z9hG4bK” in that order and case sensitive. Interestingly, there is no specific standard for the number of characters that make up the Branch statement, but for the sake of consistency, this thesis will use an assumed 42 characters to make up the Branch statement [22]. Because of the needed structure of the first 7 characters, they can be excluded from the entropy calculation, as well as the needed 6 hyphens able to be excluded. With the assumption that 26 English alphabet characters can be used in either uppercase or lower case and 10 standard numbers, for each applicable character within the 29 character string, there are a total of 62 possibilities. Therefore, the calculation of

entropy for the branch statement will be $\log_2 62^{29}$=172.6717, as rounded to four places after the decimal point.

7.2 Max-Forwards

The Max-Forwards field is a standard field in all SIP sends in which the sending end of the call sets a type of time to live (TTL) for the packet which, if reached, will indicate the packet as not received and the receiver not available within the given specifications. The Max-Forwards field is usually set to 70 as it has been found to be the standard and unchangeable within the 3CX test bed system. Therefore, as the variables given here are constant, we can very easily detect the possibility of a covert message by comparing the current Max-Forwards value to the standard value of 70. In the experimental section, it can be seen in the screen shot that Max-Forwards was changed to 95, indicating a possible message being sent to the second party. While this may impact the call itself, especially if this number is set to a value less the 70, we can still conclude that there is a definite indicator of a covert message being embedded in the call. To calculate the entropy, we assume that the Max-Forwards variable will stay at two digits, as found to be the case in both this experiment and the original case [16]. Therefore with the two digits having ten possibilities each we can calculate the entropy to be as follows: $\log_2 10^2$ = 6.6439, as rounded to four places after the decimal point.

7.3 Tag

The tag field of the SIP is generally used as an identifier for both the sender and the receiver. It is unique to each client within the VoIP network and is randomized "with at least 32 bits of randomness" [22]. Within a given VoIP call, the tag field will be different in the "From" field

(receiver side) and the "To" field, a property which would hold even if the call was sent in the opposite direction. In other words, if client A is assigned id 12345, if the call was sent in the opposite direction, in this case client B, the assigned id would still not be 12345, even for the same call session [22]. When analyzing the tag field, we maintain the assumption that the tag field will always be 8 characters long and have a possibility of 16 characters because of the observed use of only hex based letters and digital numbers within the 3CX test bed system. Therefore, the entropy for the tag field would be given by the equation: $\log_2 16^8 = 32$.

7.4 Call-ID

The Call-ID field is randomly generated for each call and links messages together. As can be seen in the results section of this thesis, it is unable to carry a covert channel because of its specification that it must be the same for every request and response sent by a specific user agent. Even though it must be the same for every call, it also must be randomly selected as to not be chosen by two agents at any given time. Therefore, when we discuss the entropy of the Call-ID field, we can safely say that the entropy is 1 due to no detectable pattern to apply and predict what a given call should have as a Call-ID. Screen shots can be seen below as to the difference in Call-ID's within legitimate calls [22]. Theoretically, if the protocol was changed to allow for manually selected Call-ID's to be assigned, the entropy would be given by the equation: $\log_2 62^{59} = 351.2976$.

7.5 CSeq

CSeq, or the call sequence, is the way that SIP keeps calls and packets in order as they arrive at the server. For each new call made within the VoIP network, the CSeq field is incremented and

sent within the SIP, along with the method for which the packet is being used. The most common method is "INVITE" mainly because the SIP packet is acting as an invite for another user agent to join the call. According to the Request for Comments which addresses SIP, non-registered requests that enter the network to reach a specified client are assigned random CSeq numbers, rather than the sequential ones used for registered calls, and are in the format of a number less than 2^{31}. From this information we can determine that the entropy of the CSeq field can be calculated as $\log_2 2^{31}-1 = 30.9999$ [22].

7.6 Contact

The contact field is populated based on what ID is given to the phone by the system administrator and thus passed down to the phone on its registration with the SIP server. In the experiment, the two softphones were set as Softphone2 and Softphone1, as sender and receiver respectively, and thus the entropy could be discovered by comparing the received identification name of the sender, to that which was assigned to it from the server by the systems administrator. With this characteristic being true, we can conclude that the entropy of the Contact field would be 1. On the other hand, if there was no way to tell if the Contact variable as assigned by the sending party was the same as the one which was received by the receiving party, there is a bit more complication added into the entropy calculation. According to RFC 3261, the contact can be up to 15 characters and be any combination of standard alphanumeric characters as discussed in the branch statement. In addition to an identifier, there is also an IP address appended to the field within angle brackets, and a port number from which the call was sent from. Therefore, calculation of the contact field would consist of $\log_2 62^{15} + \log_2 3^4 + \log_2 10^8 + \log_2 7^1 + \log_2 10^4 = 138.3233$.

7.7 SDP V

As with the Max-Forwards field, in which there was a default value set to 70 which was unable to be easily changed, the SDP V field has a default value of 0, due to its property of being the version of a given announcement sent to the client. Usually, there is only a single announcement sent to the client because the description of the call or data type contained within it voice is unlikely to change, but there is the possibility of the description or data changing. Therefore, even though the field is not readily apparent to the sender or receiver when they are sending or receiving the call, respectively, upon further inspection, it can easily be discovered as to whether a covert message may be hiding in the field by comparing the V field's value to the standard one of 0, especially if the description has not been changed by the user. Based on this evidence, the calculation of entropy would be $\log_2 10^1 = 3.3219$ [8].

7.8 SDP O

The SDP O field is used to identify the originator of the session to which the clients are not taking part in. The O field is made up of the username of the originator, session ID of the call and version, as well as the address of the originator's host. In the experimentation phase of this project, it was found that changing the session ID and call version of the O variable had no effect on the call and still allowed for a call to be completed successfully. Similar to the contact field, we must calculate the entropy of the IP address as well as the two identifier variables and thus are left with the equation: $\log_2 7^1 + \log_2 10^4 + \log_2 7^1 + \log_2 10^4 + \log_2 3^4 + \log_2 10^8 = 65.1054$.

8. Covert Channel Detection Tool

Throughout the experiment, it was found that no tools actually existed for detecting covert channels within voice over IP streams. Unless a systems administrator or network administrator poured over logs regarding the VoIP transactions that took place and meticulously dissected every SIP and SDP packet within those logs, there is little to no ability for the detection of covert channels within Voice over IP. This downfall was furthered by the fact that if a VoIP call was intercepted midway into the call, the packets would not be seen as RTP or even SIP/SDP, but rather be seen as UDP packets because of the ability to identify them with the average packet scanning tool. This is a problem which obviously needed to be fixed, based on the shown ability for attackers with malicious intent to embed messages very simply, using commercial off the shelf software (COTS), or just as easily be able to modify their own software to modify packets as needed. Within the following paragraphs, an open source, java based, covert channel identifying program will be discussed, based on the work of Keita Fujii [7] and then modified by Patrick Lloyd with the help of Sean Madden.

The program works by analyzing captured packets, similar to most other packet capture programs such as Wireshark. While capturing the packets, the program also analyzes them with built in functionality from the original author, to discover the type of packet and break down the packet into its basic characteristics, including whether the packet is IPv4 or IPv6, UDP or TCP, the protocol through which the packet was sent, such as Ethernet, and basic information about the packet including the captured time and the captured length. More technical details regarding the program follow.

Built in functionality included the ability for the program, JPCapDumper by Keita Fujii [7], to capture programs using the JPcap library which was developed for exactly this purpose. The program then displays the frames in a table type format, and if the packet is analyzable based on given characteristics of supported packet formats, it displays these characteristics in the left hand pane in a tree type format. Included formats for packets include: Ethernet packets, IPv4 Packets, IPv6, TCP, UDP, ICMP, HTTP, FTP, Telnet, SSH, SMTP, POP3, and ARP. There is also a properties file which the program reads from to determine how to populate the columns on the top of the windows, based on child properties of each of the aforementioned protocols.

With this framework, the UDP packet analyzer was used as a template to modify its abilities to analyze VoIP packets because of their distinct similarities, and further add in functionality to analyze whether a covert channel is present. This analysis of a VoIP packet containing a covert channel is based on whether the packet contains the keyword "INVITE," a keyword common to all VoIP SIP and SDP packets, because of the list of features that is contained within the packet (see line 00000150 of Figure 9). Based on this proving true, another operation is then triggered to look for a regular expression which will detect and compare the variables: "Max-Forwards:" to be followed by 70, "V=" followed by 0, "T=" followed by "0 0", and "O=" followed by "-". These variables were the most easily changed and most common throughout multiple SIP formats, without affecting the quality and format of the call and packet. If any of these variables do not match their default values, a "found" variable will be changed to a "true" boolean value and populate the characteristic of the VoIP child property Covert Channel, to be "Yes," thus indicating a possible covert channel within the packet.

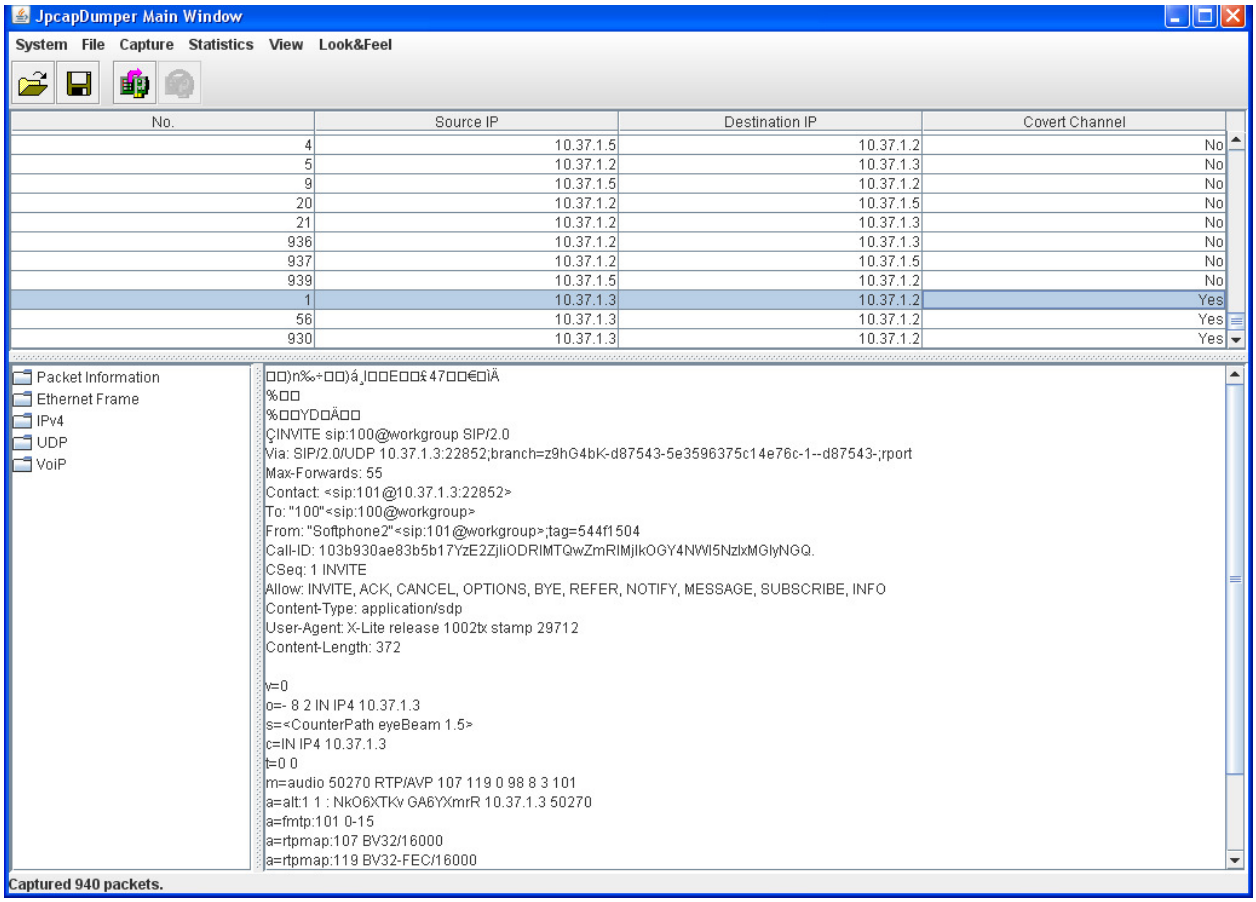


Figure 19: A covert channel detection tool with Packet Number, Source IP, Destination IP and Covert Channel indication

If the keywords are not found, in other words if Max-Forwards is equal to 70, V is equal to 0, t is equal to 0 0, and o is equal to –, or the variables just are not found in the packet, indicating a non-SIP or non-SDP packet, the message of the covert channels column will be populated by a "No" or "Not Available,"thus signaling the administrator to pass over it. The "Not Available" variable will be specific to a non-UDP formatter packet. With the already built in ability to sort the column by values, a systems or network administrator would be able to look over the

record quickly and easily to determine whether a covert message may be present within the current VoIP packet.

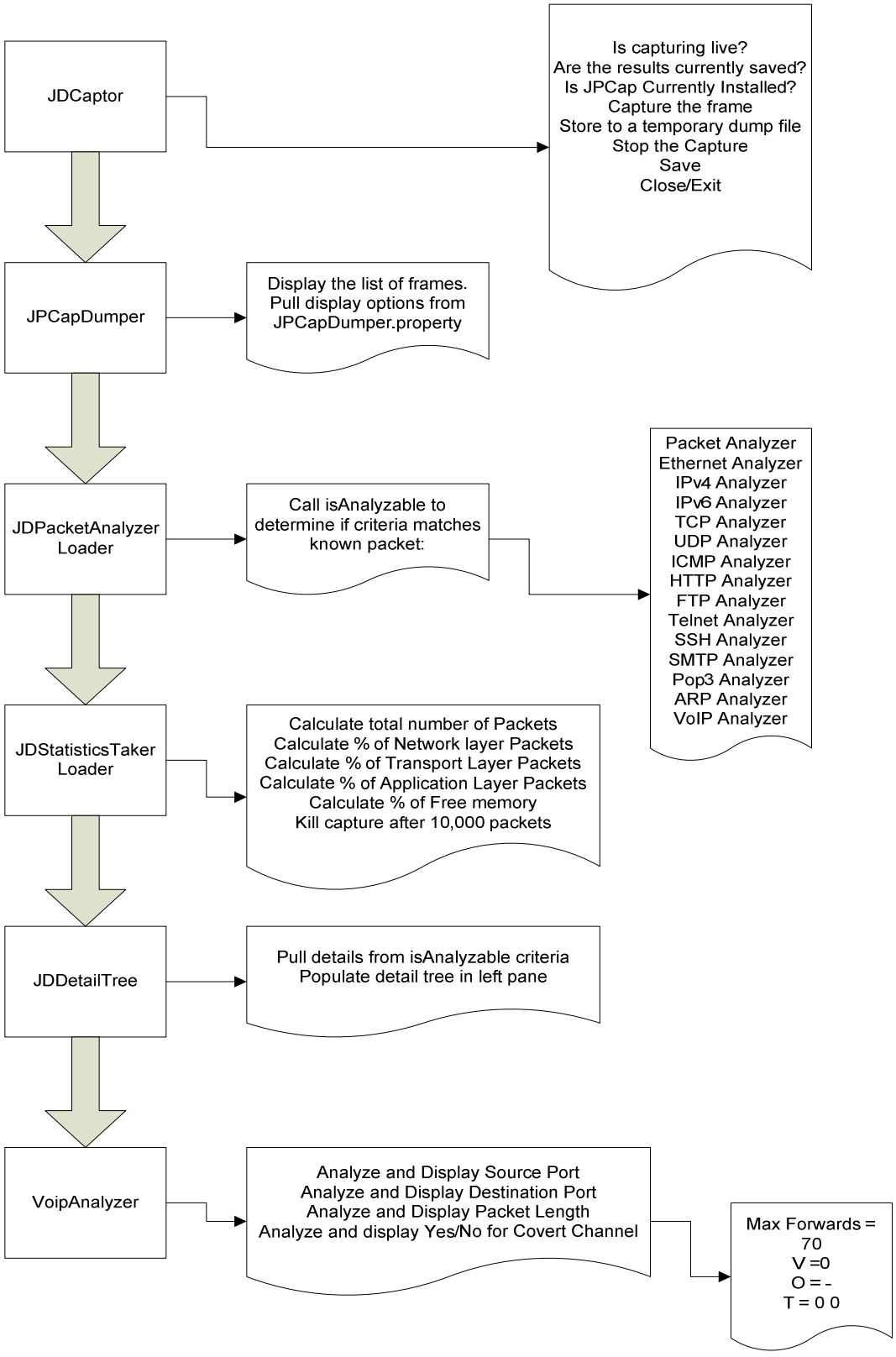


Figure 20: Flowchart depicting the operation of covert channels detection program

9. 3rd Party Developed Tools

Within this thesis, a number of tools were discussed which allow a user to embed a covert message into a SIP or SDP packet and an administrator to detect these messages to protect the integrity and security of their network. Other projects done by colleagues at Rochester Institute of Technology have focused on a similar task but rather than SIP and SDP, there is a focus on the ability to embed covert messages within the timestamp that is included in RTP packets while data is being sent between two clients. Author Christopher Forbes [5], discusses the ability to change the timestamp of a given RTP packet to use a similar method as seen here: using two digits to send an ASCII code to the other user and not impair the quality of a VoIP call. Unlike SIP and SDP covert messages, RTP covert messages can be much more numerous because of the need for encoding that is used to send data to the receiving party. This allows a much longer message to be sent to the receiving party while still using relatively simple methods to send them, mainly by using a C language based program.

10. Future Work

Future work includes more investigation being done on detecting steganography in images and other formats which allow for messages to be sent by being “in plain sight.” By developing a larger understanding of the inner workings of steganography and how it can be impacted by forensic investigation, there is the ability to better understand and prevent covert messages in telecommunication protocols.

11. Conclusion

This thesis has shown the ability to embed covert messages into a Voice over IP stream, merely by modifying the SIP and SDP packets which are sent at the beginning of the call. While these messages must be small in nature and must be embedded into the call by using a packet modification tool, they present a clear and present danger to networks and companies which can be easily combated against by monitoring the packets that flow through a communications network. To aid in the ability to monitor the network, a tool has been provided which allows for the capture of packets flowing through the communications network which also allows for a network administrator to sort the resulting packets to easily find an indicator of a covert channel message. This indicator comes in the format of a simple Yes/No/Not Available option within the program.

12. Works Cited

[1] "3CX: Software based PBX for Windows." *3CX Phone System - software based VoIP IP PBX / PABX for Windows*. 2009. Web. 01 Feb. 2010. <http://www.3cx.com/>.

[2] Arrington, Michael. "Schwarzenegger Gives California Legislature A Hidden Finger." *TechCrunch*. 28 Oct. 2009. Web. 01 Feb. 2010. <http://www.techcrunch.com/2009/10/28/schwarzenegger-gives-california-legislature-a-hidden-finger/>.

[3] "ASCII Table." *Ascii Table - ASCII character codes and html, octal, hex and decimal chart conversion*. Web. 01 Feb. 2010. <http://www.asciitable.com/>.

[4] ”Codec.” Wikipedia, the free encyclopedia. 17 May 2009. 19 May 2009 〈http://en.wikipedia.org/wiki/Codec〉.

[5] Forbes, Christopher R. "A New Covert Channel over RTP." Thesis. ROchester Institute of Technology, 2009. Print.

[6] "Entropy (information theory) -." *Wikipedia, the free encyclopedia*. 31 Jan. 2010. Web. 01 Feb. 2010. <http://en.wikipedia.org/wiki/Entropy_%28information_theory%29>.

[7] Fujii, Keita. "JpcapDumper - Java packet capture library." *Professor Tatsuya Suda's NETGROUP Webpage*. 09 Jan. 2005. Web. 1 Feb. 2010. <http://netresearch.ics.uci.edu/~kfujii/JpcapDumper/doc/index.html>.

[8] Handley, M., and V. Jacobson. "RFC 2327 (rfc2327) - SDP: Session Description Protocol." *Internet FAQ Archives - Online Education*. Apr. 1998. Web. 10 Nov. 2009. <http://www.faqs.org/rfcs/rfc2327.html>.

[9] "History of Entropy." *Wikipedia, the free encyclopedia*. 9 Dec. 2009. Web. 01 Feb. 2010. <http://en.wikipedia.org/wiki/History_of_entropy>.

[10] ”ITU G.723.1 - voip-info.org.” Voip-info.org - voip-info.org. Voip-info.org, LLC. 19 May 2009 〈http://www.voipinfo.org/wiki/view/ITU+G.723.1〉.

[11] Johnson, Neil F. "Steganography - SEC202." *Johnson & Johnson Technology Consultants, LLC*. Nov. 1995. Web. 10 Nov. 2009. <http://www.jjtc.com/stegdoc/sec202.html>.

[12] Judge, James C. "Steganography: Past, Present, Future." *Steganography: Past, Present, Future*. SANS Institute, 2001. Web. 1 Feb. 2010. <http://www.sans.org/reading_room/whitepapers/stenganography/steganography_past_present_future_552>.

[13] Kessler, Gary C. "Steganography for the Prosecutor and Computer Forensics Examiner." *GaryKessler.net Home Page*. Apr. 2004. Web. 01 Feb. 2010. <http://www.garykessler.net/library/ndaa_stego.html>.

[14] Lubacz, Jzef, Wojciech Mazurczyk, and Krzysztof Szczypiorski. Hiding Data in VoIP. Diss. Warsaw University of Technology. Hiding Data in VoIP. Warsaw University of Technology. 19 May 2009 〈http://www.asc2008.com/manuscripts/B/BP-13.pdf〉.

[15] Mazurczyk, Wojciech, and Krzysztof Szczypiorski. "Covert Channels in SIP for VoIP Signalling." *Eglobal_sip_covert.pdf*. Warsaw University of Technology, 2008. Web. 1 Feb. 2010. <http://home.elka.pw.edu.pl/~wmazurcz/moja/art/Eglobal_sip_covert.pdf>.

[16] Mazurczyk, Wojciech, and Krzysztof Szczypiorski. “Steganography of VoIP streams.” Diss. Warsaw University of Technology. Berlin/Heidelberg: Springer, 2008.

[17] McCullagh, Declan. "Bin Laden: Steganography Master?" *Wired News*. 07 Feb. 2001. Web. 01 Feb. 2010. <http://www.wired.com/politics/law/news/2001/02/41658>.

[18] Mercuri, Rebecca T. "The Many Colors of Multimedia Security." *The Many Colors of Multimedia Security*. Security Watch, Dec. 2004. Web. 10 Nov. 2009. <http://delivery.acm.org/10.1145/1040000/1035155/p25-

mercuri.pdf?key1=1035155&key2=8070987521&coll=GUIDE&dl=GUIDE&CFID=60919341&CFTOKEN=10173797>.

[19] O'Neill, Helen. "VoIP Providers List - VoIP Providers USA." *VoIP Providers List Worldwide VoIP Providers Directory*. Web. 10 Nov. 2009. <http://www.voipproviderslist.com/country/voip-usa/voip-providers-usa/sort-byname/65/>.

[20] "Polycom® IP Conference Phones for Avaya." *Avaya Interoperability Guide*. Polycom Worldwide. Web. 1 Feb. 2010. <Avaya Interoperability Guide>.

[21] ”Real-time Transport Protocol -.” Wikipedia, the free encyclopedia. 18 May 2009. Wikipedia. 19 May 2009 〈http://en.wikipedia.org/wiki/Real-time Transport Protocol〉.

[22] Rosenberg, J., H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. "Session Initiation Protocol." *Internet Engineering Task Force*. June 2002. Web. 10 Nov. 2009. <http://www.ietf.org/rfc/rfc3261.txt>.

[23] *SampleDiagonal.jpg*. 2009. Photograph. <Http://www.shutterfreaks.com/Actions/WatermarkSignatures/SampleDiagonal.jpg>.

[24] "Star Trek: The Episode Guide." *"The Killing Game II"* 28 June 2006. Web. 1 Feb. 2010. <http://stvoy.epguides.info/?ID=522&ComingFrom=%3FKey%3D7798&Key=5141>.

[25] “Steganography." *Wikipedia, the free encyclopedia*. 2 Feb. 2010. Web. 02 Feb. 2010. <http://en.wikipedia.org/wiki/Steganography>.

[26] Takahashi, Takehiro, and Wenke Lee. An assessment of VoIP covert channel threats. Diss. Georgia Institute of Technology, 2007. An assessment of VoIP covert channel threats. 21 Sept. 2007. IEEE. 19 May 2009 〈http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4550357&isnumber=4550291>.

[27] "Voice Over IP Statistics - Number and Duration of calls per Year." Greek University Network, 2005. Web. 10 Nov. 2009. <http://voip.noc.uoa.gr/voipstat/query-call-per-year.php?v=1&lang=en>.

[28] ”Voice Over IP - Availability Statistics.” Voice Over IP – Availability Statistics. 2005. Greek University Network. 19 May 2009 〈http://voip-availability.gunet.gr/?lang=en〉.