

UNIVERSITÀ DEGLI STUDI DI MILANO
FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI
CORSO DI LAUREA IN INFORMATICA PER LE TELECOMUNICAZIONI



**ANALYSIS AND IMPLEMENTATION OF SECURE AND UNSECURE
VOICE OVER IP ENVIRONMENT AND PERFORMANCE COMPARISON
USING OPENSER**

Relatore: Prof. Danilo BRUSCHI
Correlatore: Dott. Alessandro ROZZA
Secondo Correlatore: Prof. Miroslav Voznak

Tesi di Laurea di
Antonio NAPPA
matr. 675499

Anno Accademico 2006/2007

UNIVERSITÀ DEGLI STUDI DI MILANO
FACOLTÀ DI INFORMATICA

CORSO DI LAUREA IN INFORMATICA PER LE TELECOMUNICAZIONI

ANALYSIS AND IMPLEMENTATION OF SECURE AND
UNSECURE VOICE OVER IP ENVIRONMENT AND
PERFORMANCE COMPARISON USING OPENSER

Relatore: Prof. D. Bruschi
Correlatore: Dott. Alessandro Rozza
Secondo Correlatore: Prof. Miroslav Voznak
Candidato: Antonio Nappa

ANNO ACCADEMICO 2006-07

Ai miei genitori... A Paola e Grigo compagno di tanti pomeriggi di studio

Acknowledgements

Ringrazio i miei genitori per avermi dato la possibilità di esprimere la mia personalità senza costrizioni e per avermi sempre incoraggiato a scegliere liberamente.

Un ringraziamento speciale per la mia dolce metà, Cinzia, per il suo amore, la sua immensa pazienza e disponibilità.

Ringrazio mia sorella Paola, che mi ha aiutato nei momenti critici con una parola di conforto che solo chi ti sta così vicino può darti.

Ringrazio il professor Danilo Bruschi che ha appoggiato le mie idee e mi ha dato la possibilità di realizzarle.

A special thank to Miroslav Voznak, the person who taught me the bases of the Voice Over IP..

Ringrazio il mio correlatore A.Rozza per il supporto datomi e per avermi dato l'input finale di realizzazione di questo lavoro.

L'ultimo ma non per questo meno importante ringraziamento è per Ilenia, per avermi aiutato nella correzione della lingua inglese.

Contents

Abstract	1
1 Voice Over IP	3
1.1 What is Voip?	3
1.2 Why is it so successful?	4
1.3 Will the Voip replace the current telephone network?	4
2 Voice over Ip protocols	5
2.1 The Tcp/Ip Model	5
2.2 Sip Protocol introduction	5
2.2.1 Sip Architecture	6
2.2.2 Sip Requests and Responses	7
2.2.3 Sip security issues	8
2.3 The Sip protocol and multimedia, introducing the SDP protocol	12
2.4 The transport protocols	12
2.4.1 The Rtp protocol	13
2.4.2 The Srtp protocol	13
2.4.3 The Zrtp protocol	15
2.4.4 The TLS protocol	16
3 Voip security issues and solutions	17
3.1 Introduction	17
3.2 Physical layer	17
3.2.1 Cable hacking	17
3.2.2 Wifi wardriving	18
3.3 Data Link layer	18

3.3.1	Arp Poisoning	18
3.3.2	Vlan flooding technique	18
3.4	Network layer	19
3.4.1	Ip spoofing	19
3.4.2	Ip spoofing: an example	20
3.5	Transport layer	21
3.5.1	Udp flooding	21
3.5.2	Tcp SYN flooding	21
3.6	Application layer	21
3.6.1	Sip protocol hacking	21
3.6.2	Dhcp Exhaustion	22
4	Voip Security:Our Experiments	23
4.1	Experiment summary	23
4.2	Experiment configuration	25
4.2.1	The Simulation	26
4.2.2	Real-Tests	26
4.2.3	Security Measures	26
4.3	Experiment tools	26
4.3.1	IxChariot	26
4.3.2	OpenVpn	26
4.3.3	OpenSer	27
4.3.4	Iperf	29
4.3.5	Sipp	29
5	The Results	31
5.1	Simulated tests	31
5.1.1	Unsecure simulation	32
5.1.2	Result forecast	33
5.1.3	Test result	33
5.1.4	Secure simulation	36
5.1.5	Result forecast	36
5.1.6	Test result	36
5.2	Real tests	38
5.2.1	Unsecure test	39
5.2.2	Result forecast	39
5.2.3	Test result	39
5.2.4	Secure test	42

5.2.5	Result forecast	42
5.2.6	Test result	43
5.3	Conclusions	44
A	Italian resume	47
A.1	Panoramica sul VoIP	47
A.2	Protocolli VoIP	47
A.3	Le problematiche	48
A.3.1	Affidabilità	48
A.3.2	Sicurezza	48
A.3.3	Quality of Service	49
A.4	Il protocollo SIP	49
A.5	SIP e il livello applicazione	50
A.6	Caratteristiche e funzioni del protocollo SIP	50
A.7	Architettura SIP	50
A.7.1	User Agents	51
A.7.2	Sip Server	51
A.8	SIP Requests and Responses	52
A.9	Protocolli associati	53
A.9.1	User Datagram Protocol (UDP)	54
A.9.2	Transport Layer Security (TLS)	54
A.9.3	Session Description Protocol (SDP)	54
A.9.4	Real-Time Transport Protocol (RTP)	54
A.9.5	SRtp	55
	Confidenzialità	55
	Autenticità del messaggio	55
	Key management	55
A.9.6	ZRtp	56
A.10	Possibili attacchi SIP	56
A.10.1	DoS Denial-of-service Attacks	56
A.10.2	Registration hijacking	56
A.10.3	Man-In-The-Middle (MITM)	57
A.11	Sicurezza voip in pratica	57
A.11.1	Il nostro esperimento	57
A.11.2	Le ragioni del nostro esperimento	58
A.11.3	La modalità operativa dell'esperimento	58
A.12	I tools utilizzati	59
A.12.1	IxChariot	59

A.12.2 OpenVpn	59
A.12.3 OpenSer	59
A.12.4 Iperf	61
A.12.5 Sipp	61
A.13 I nostri risultati	62
A.14 I test simulati	62
A.14.1 Simulazione non sicura	63
A.14.2 Previsioni sui risultati	63
A.14.3 Risultati	63
A.15 Simulazione sicura	63
A.15.1 Previsioni sui risultati	63
A.15.2 Risultati	64
A.16 I test reali	64
A.17 I test reali in modalità non sicura	65
A.17.1 Previsioni sui risultati	65
A.17.2 Risultati	65
Man in the middle	65
Sip flooding	65
Udp flooding	65
Sip registration hijacking	66
A.18 I test reali in modalità sicura	66
A.18.1 Previsioni sui risultati	67
A.18.2 Risultati	67
A.19 Conclusioni	68
B Configuration files	69
B.1 Openser.cfg	69
B.2 Openvpn.cfg	70
B.3 Sipp configuration	71
B.4 Iperf configuration	72
Bibliography	73

List of Figures

2.1	The Tcp/Ip model	5
2.2	Sip Registrar	9
2.3	Sip Proxy	10
2.4	Sip Redirect	11
2.5	Zrtp working example	15
3.1	The 802.3 and 802.1Q headers	19
3.2	Explaining the Ip spoofing	20
4.1	R-factor - Mos	24
5.1	Testbed of the simulation	32
5.2	G.711Alaw - NOVPN	34
5.3	G.729 - NOVPN	35
5.4	G.711Alaw - VPN	37
5.5	G.729 - VPN	38
5.6	Sip registration hijacking	41
5.7	Real test - G.711Alaw - NO VPN	41
5.8	Real test - G.729 - NO VPN	42
5.9	Real test - G.711Alaw - VPN	44
5.10	Real test - G.729 - VPN	44

Abstract

In the last five years, the growth of voice services over data networks, and especially over the Web, has reached considerable levels. The fast expansion of a branch of technology usually raises a number of security issues owing to the fact that many of the components of such projects are not generally ready to reach high expansion levels. In this document, we will examine OPENSER - a famous software application used by a large number of Voice over IP providers. This is an open source software application. We choose it in order to focus our attention over the Sip protocol since, according to an accurate analysis we carried out, we found out that the H.323 and other famous protocols were no longer not so commonly in use. Of course, somebody would say the opposite but we had to make a decision and we hope we made the good one. We will take into consideration voice over IP protocols, such as Skype, GoogleTalk, SIMPLE, XMPP, just to properly describe the scenario in which we want to perform our testing and analyses. After such brief introduction, we will introduce the Sip protocol and its way of working, giving a deeper explanation of all the components of the protocol. The analysis will take into account the transport protocol named RTP (Real Time Protocol). We will focus on all the weak points of the protocol and we will describe a secure solution as well as other transport protocols provided with security features that can replace the RTP (ZRTP, SRTP). Following to this, we will represent a real environment either with or without security measures and we will point out all the differences between the two environments and the reasons why they could be immune, or not, from an attack and we will also compare the performance of both solutions.

Chapter 1

Voice Over IP

1.1 What is Voip?

VoIP is the acronym for "Voice over Internet Protocol". Such technology refers to voice traffic over data networks. VoIP can define a lot of different protocols, among which the most famous are: Sip, H.323, H.248, XMPP, SIMPLE, Skype.

Sip: a standard IETF protocol defined in the RCF 3621 and used to transmit data, audio and streaming video files. Such protocol is basically used to initialize, modify and terminate sessions between two or more endpoints. This protocol will be explained in the second chapter.

H.323: a standard ITU (International Telecommunication Union) protocol. This protocol was one of the first VoIP protocols. It is going to be replaced by the Sip protocol.

H.248: protocol defined in RFC 3525 and in H248-1 ITU recommendations. It results from the combination of IETF and ITU protocols. It is used to control data streams between gateways.

XMPP: an instant messaging protocol, based on XML. Associated to a Jingle extension, it can be used to support VoIP traffic. Such configuration is used by GoogleTalk. XMPP is a direct competitor of SIMPLE, an instant messaging protocol based on the Sip protocol

SIMPLE: an acronym for "Session Initiation Protocol for Instant Messaging and Presence Leveraging Extensions". This protocol is being currently developed by IETF. Some parts of it are standardized in the RFC 3428, the features of this protocol should be:

- Registering user presence information and receiving notifications when such events occur, e.g. when a user logs in or turns its status from "away for lunch" into "available".
- Sending short messages, similar to SMS or two-way paging.
- Managing a session of real-time messages between two or more participants.

SKYPE: a complete proprietary protocol against open standards, such as H.323 and Sip. It operates with a peer-to-peer model and was developed by the creators of the file sharing application Kazaa and Joost, the peer-to-peer television.

1.2 Why is it so successful?

The success of technologies for voice transmission over data networks is related to the low costs of data transmission over the earth. If you want to use the VoIP technology, all you need to have is a broadband connection. And if we consider that in almost every house there is a PC connected to the Web nowadays, understanding the reason of such success is much easier. Why using the traditional phone connection if we have a better and cheaper solution? Now, it is also possible to keep your original telephone number and to transfer it over a VoIP line.

1.3 Will the Voip replace the current telephone network?

Everything seems to be really attractive and perfect but let's not forget that the Internet is based on the "Best Effort" service. Hence, some data could be lost. Also, if we don't have enough bandwidth for our voice transmission some delays can occur. Another very important issue is about emergency services. Imagine everybody all over the world is using the Voip technology, it will be difficult to localize the caller owing to the nature of the IP protocol. Also, can you imagine a "Denial of Service" attack against emergency services? What could happen? Fortunately, we are still far from this scenario.

Chapter 2

Voice over Ip protocols

2.1 The Tcp/Ip Model

As you can see in the following picture, the Voip technology uses several protocol. In order to explain in a better way, how this protocols are working together and which is the role of each one, we used the simplified Tcp/Ip model.

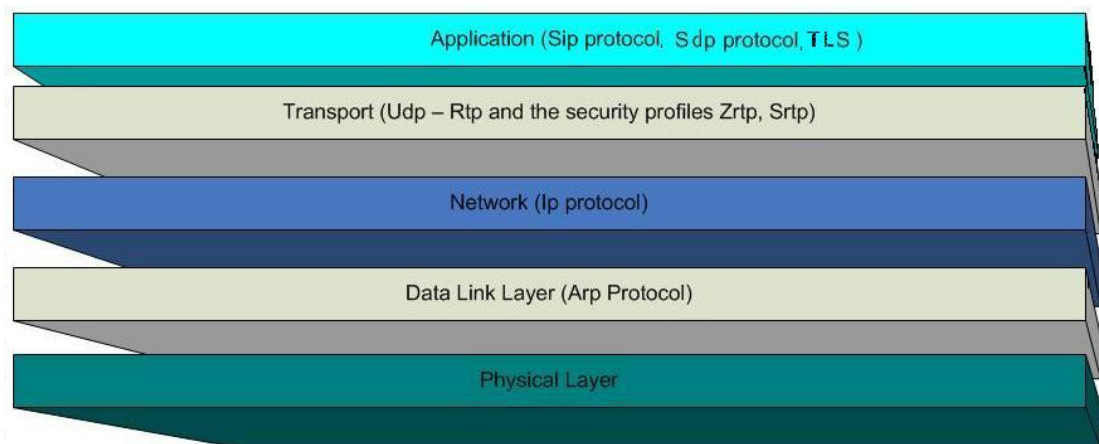


Figure 2.1: The Tcp/Ip model

2.2 Sip Protocol introduction

This application layer protocol is explained in the IETF (Internet Engineering Task Force) RFC(Request for comments) 2543 and 3261. The Sip protocol data is carried either with Tcp or Udp, the common

used port is the 5060. Sip is designed to start, manipulate and close interactive sessions over an IP network. Sip does not directly support the VoIP technology, video conferencing, instant messaging, etc., but it is an important component that helps manage communication factors. Sip works with several network components in order to locate and identify the endpoints of a communication. Proxy servers are used for registering and routing requests to the user's location. Such servers manage the invitations from other users and forward the requests to connect the new endpoints to the existing session. Sip works with URI (Universal Resource Identifiers) to identify users and to connect a simple username, such as foobar@dico.unimi.it, to the correspondent phone number and other information. Sip, which stands for "Session initialization protocol", works in an easy way, and the easier a protocol is, the larger number of applications can support it.

2.2.1 Sip Architecture

The protocol works together with other protocols and with the Sip Servers and the Sip User agents:

- Sip Servers: computers on the network that service requests from clients, and send responses back.
- User Agents (UA): endpoints of a call

Sip Servers

The main feature of a Sip server is the registration of clients, and its ability to put the clients registered with different servers into communication, it resolves usernames to IP addresses. Keeping track of registered users gives a real-time status of users' availability. Therefore, all of the registered clients can learn whether a user is busy. To achieve all of the different tasks that we described above, the server has different roles:

Registrar Server:

it stores the information on the user's location, IP address - username combination, and the presence information, when a user is logged in, the system knows whether he can be called and the connection can be performed.

Proxy Server:

it forwards requests on behalf of other computers. If we call somebody who has not registered with our same Registrar as we are, the Proxy Server can act on our behalf to perform the call toward another server, in order to check if the person we are looking for is there.

Redirect Server:

it gives some information to clients who are looking for someone who is not on the same server. The redirect server responds with the IP address and the username of the client being contacted. This is different from the Proxy Server which acts on our behalf. The redirect just tells us where we have to call if we want to be connected to somebody that is not on our same server.

User Agents:

With this name we can describe each endpoint of a communication, when the user agent performing the call is named UAC (User Agent Client) and the user agent receiving the call is named UAS (User Agent Server). UAS and UAC exchange messages and swap their roles in a similar manner during the session. Without this kind of interaction the communication could not exist. A user agent could be described also as an application software installed on a computer a PDA or a Ip telephone. In any of these cases the user agent act as both a UAC and UAS, because it sends out and receive messages. We can say that when the UA sends a message it plays the role of UAC, and when it receives it acts as a UAS.

2.2.2 Sip Requests and Responses

Sip is a text-based protocol like the HTTP, when a UA acts as like a client, there is a set of requests that it could perform:

- REGISTER It is easy to understand that whenever the client wants to log-in the registrar server, it will use this message.
- INVITE It is used to invite another user agent to communicate and establish a Sip session
- ACK It is used to accept a session and confirm reliable message exchange
- OPTIONS It is used to understand the ability of another user agent, e.g. if the person that we would like to contact is able to make a video call.
- SUBSCRIBE This option is common with instant messaging software. If we want to have updated information about the presence of somebody, to understand whether his status has changed (online, off line, busy, lunch time.etc)
- NOTIFY It is used to communicate to the server the change of status eg:(away for lunch, do not disturb, etc)
- CANCEL It is used to abort a pending request without terminating the session properly
- BYE It is used to terminate the sessiion properly. Both user agents could use this message any time to terminate the communication

It is logical that when a request is made someone has to answer. Hence there are different categories of answers that a user agent could give. For each response there is a different numerical code to understand the meaning of the protocol messages.

- Informational (1xx) The request has been received and is being processed
- Success (2xx) The request was acknowledged and accepted
- Redirection (3xx) The request cannot be completed by this user agent. An additional step is required to accomplish the request.
- Client error (4xx) Not so difficult to understand that some errors occurred in the client's request and the server cannot process it properly.
- Server error (5xx) The request was received but the server, cannot process it. It is an error referred to the server we are contacting. Another server could probably be able to process our request without any problems.
- Global failure (6xx) The request was received and the server is not able to process it.

2.2.3 Sip security issues

Now we are going to describe some call scenarios in order to understand the security issues about Sip communication. After that, it will be possible to appreciate the difference between RTP and other protocols as far as security issues are concerned.

Sip Registration

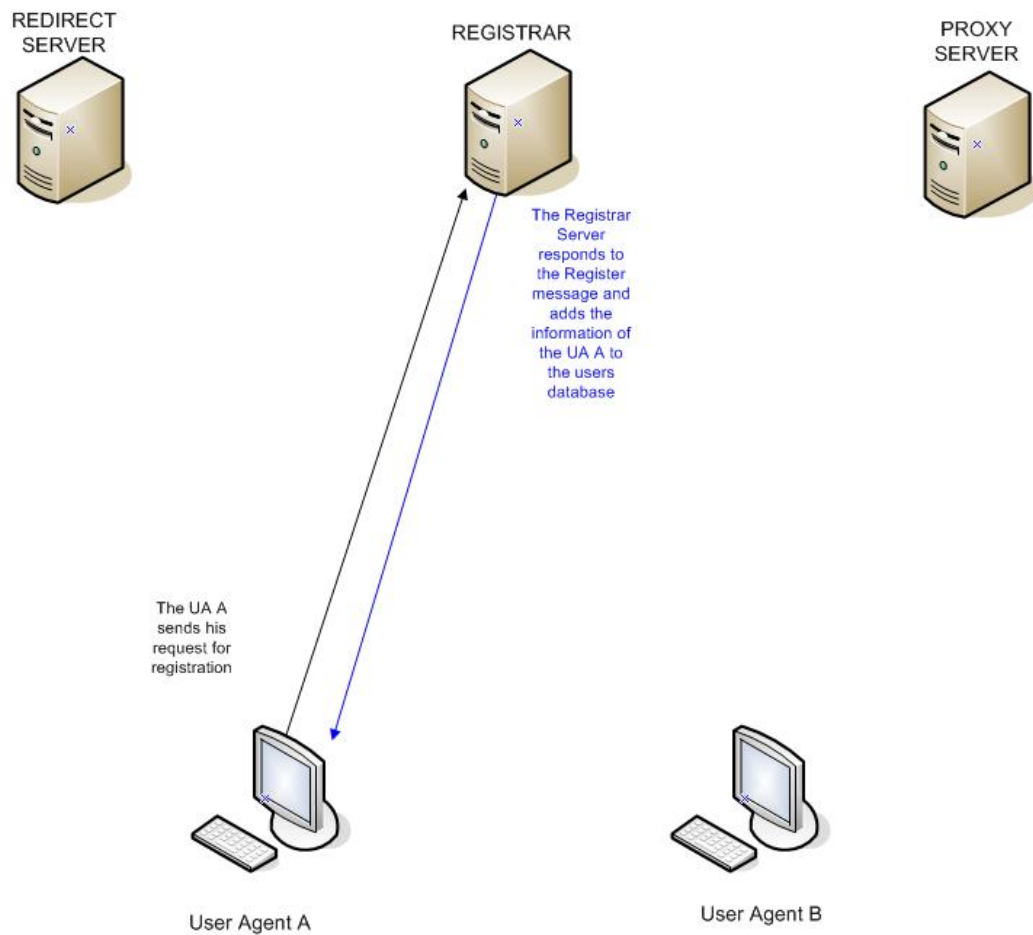


Figure 2.2: Sip Registrar

As you can see in the image above the User Agent A is sending the registration information to the registrar server. Should there be no special security policies every User Agent could be registered without any authentication process. On the contrary, in case there are some authentication methods User Agent A should be registered to the server if he wishes to be connected to the service.

Sip Proxy

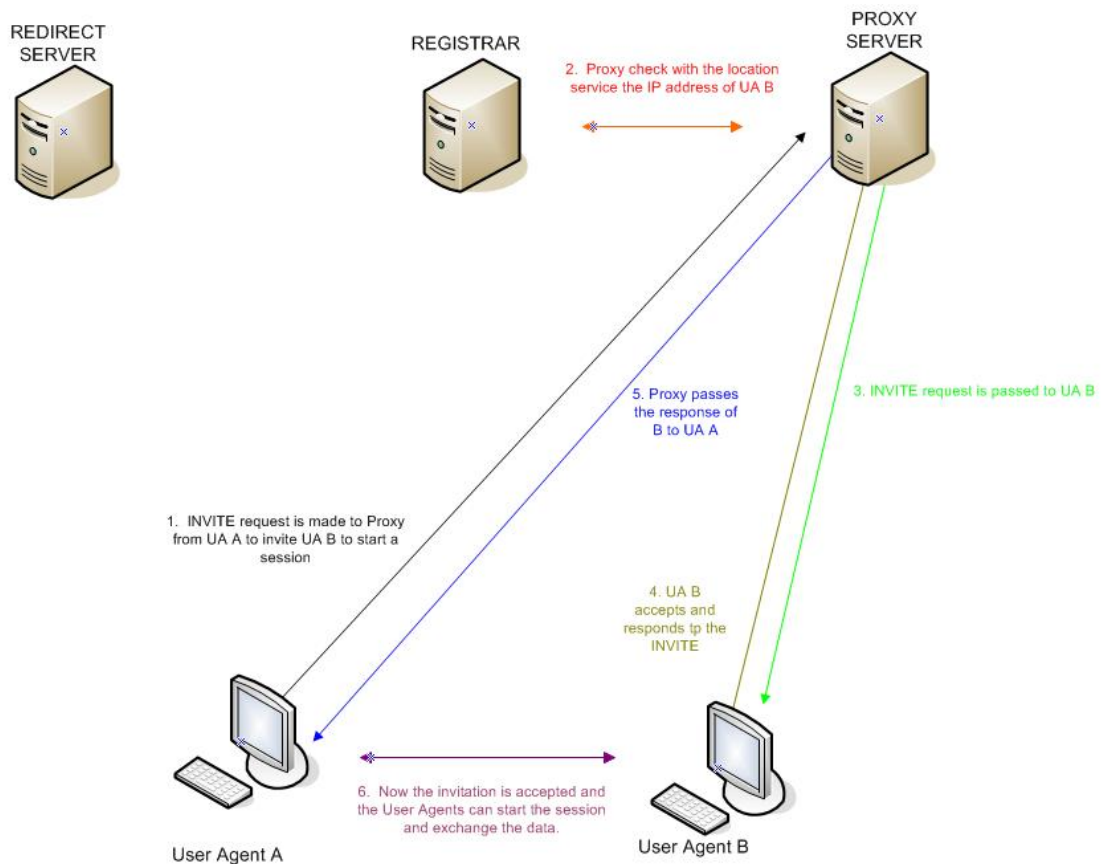


Figure 2.3: Sip Proxy

Once User Agent A has registered, we suppose that he would like to establish a session with the User Agent B. UA A will contact the Proxy to make it act on his behalf in order to contact UA B. Once the INVITE procedure is accomplished, the User Agents could perform the call properly. Now Rtp data can pass between UA A and UA B in two different ways. First, all the data pass through the proxy. This option is not very common, because the amount of data concerning a number of calls can easily overload the proxy. The second option is that the RTP data flow passes directly from one side of the communication to the other, just the Sip signaling (INVITE, Success 200, ACK, etc) pass through the proxy. It is possible to set the proxy in two different manners. In case we wish to keep track of the Sip signals to the proxy, we will use the stateful mode. On the contrary, we will use the stateless mode.

Sip call through Redirect

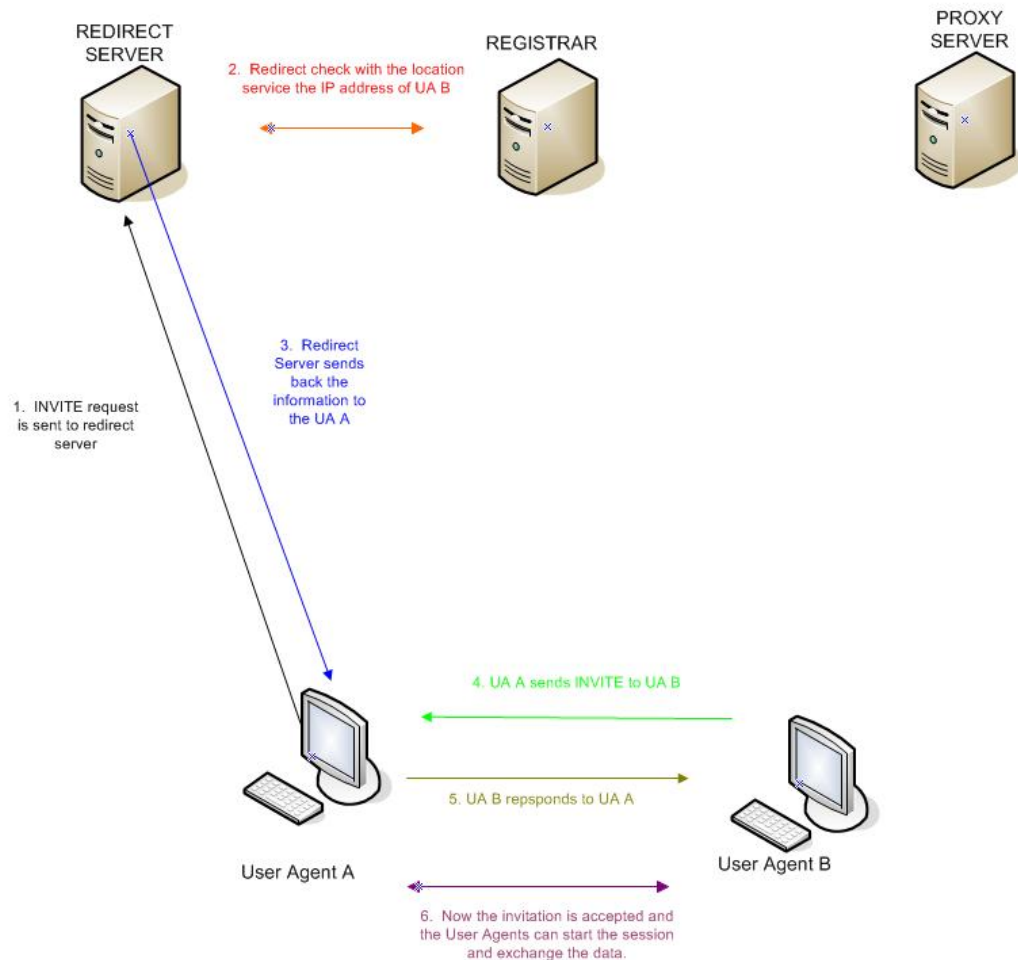


Figure 2.4: Sip Redirect

In this case if the UA A would like to call the UA B, but he does not know UA B's IP address. UA A will ask the Redirect server for UA B's address. The Redirect server will check UA B's using the location service and after that will provide UA A with an answer. Once A knows the information needed to contact B he can start a new session.

As you can see in the above mentioned scenarios, all of the communications are without any kind of trusting mechanism. This means that anyone could pretend to be the server and retrieve the account of any UA. (Identity theft) In this way a malicious user can pretend to be a User Agent and could start a

conversation with somebody who believes to be connected to a person he knows. Other actions like this can be performed. Besides identity thefts, there are some issues connected so Sip signaling and some others connected to Rtp traffic, i.e. real call data flow. Signaling is just the action before the beginning of a real communication. It includes the common Sip action to establish a session (Invite, Ack.Etc). This messages could be appropriately manipulated to finish a communication early or to start a fake communication. there are a lot of possible signaling manipulations. It is quite easy to understand that the traffic can be sniffed without any restriction, using a common software for packet sniffing, such as Wireshark. In this software suite it is possible to listen to Rtp data captured. We just need to install the appropriate audio codec. If we examine what the laws, concerning wiretapping on regular lines says we can discover how hard it is to obtain an authorization to do this kind of job. On the contrary on a computer network it seems that you can do whatever what you want. As the reader may understand the above mentioned problem is common in the TCP/IP architecture, esp. about the IPv4 protocol, since there is no support for secure data transmission. This is just a brief introduction on Voip and security related problems. We are going to analyze the situation in depth in the third chapter.

2.3 The Sip protocol and multimedia, introducing the SDP protocol

Before starting to talk about the transport protocols it is necessary to specify how the Sip protocol interacts with audio/video data. This response is given by the SDP - Session Description Protocol, described in the Rfc 4566. SIP acts as a carrier for the Session Description Protocol, which describes the media content of the session. Such protocol is used to send the needed information data when a multimedia connection is started. The information sent to the SDP packets is:

- Time the session is active
- A description of the media exchanged during the session
- The name and the purpose of the session
- The connection information data (addresses, phone number, etc..) needed to receive media.

2.4 The transport protocols

Following this short description, we can introduce the RTP - Real Time Protocol - protocol. As we may understand by its name, this protocol is used to transport real time data over the network. Either audio or video or audio/video transmission of multimedia streams over an IP network is managed by this protocol. The RTP protocol runs on top of the UDP protocol, which cannot be relied on in terms of transmitted data, but the RTP can be relied on thank to the RTCP - Real Time Control Protocol - protocol,

which monitors the delivery of the data sent between the peers of a communication. Therefore, the user agent receiving the data can detect packet losses and can compensate network delays by managing the jitter buffer dimension.

2.4.1 The Rtp protocol

The Real-time Transport Protocol (or RTP) defines a standardized packet format for delivering audio and video over the Internet. It was developed by the Audio-Video Transport Working Group of the IETF and first published in 1996 as RFC 1889 which was made obsolete in 2003 by RFC 3550. [1] This protocol does not have a standard TCP or UDP port on which it works. The only rule present in the Rfc is that the communication is UDP and is done using an even port while the next higher odd port is used by the Rtcp protocol. Rtcp is used to provide a functionality to detect losses of data and to manage the data rate. As the Rtp protocol is using a random port to transmit data across the network is very hard to manage Rtp traffic through a firewall. As a matter of fact we cannot know before which port will be chosen in the range 16384 - 32767, in advance. To overcome this problem a STUN - (**Simple Traversal of UDP (User Datagram Protocol) through NATs (Network Address Translators)**) - server is usually deployed. This protocol simplifies the communication between the client with NAT/Firewall infrastructure and the external network.

The services provided by the Rtp protocol are:

- Payload-type identification - Indication of the kind of data is inside the payload (audio, video, text, etc)
- Sequence numbering - PDU(Protocol data unit) sequence number
- Time stamping - This is very important to allow packet synchronization and jitter calculations.
- Delivery monitoring

2.4.2 The Srtp protocol

To overcome the weak point of the TCP/IPv4 architecture is necessary to activate a multilevel protection. First of all the attestation of the identities. If we are planning to call somebody, we have somehow to make sure that is the person we want to call. An other but not less important aspect is the confidentiality of the information passing over the network. Surely we do not want that somebody with no authorization to eavesdrop our private conversations. The SRtp protocol was one of the first answer focused on the security issue of the voip communication, defined in the RFC 3711. The SRtp has also a sister protocol named SRtcp which has the same functionality as Rtcp (Rtp data-flow management, packet loss, etc). The SRtp is defined to provide: confidentiality, message authentication, data integrity and replay protection:

- Confidentiality: there is just one cipher suite -AES- that can be used with two different working modes thus turning the originally AES block structure into a stream cipher. The two working modes are:
- F8-mode: that is a variation of the output feedback mode, enhanced to be seekable and with an altered initialization function. The default values of the encryption key and salt key are the same as for AES in Counter Mode. (AES running in this mode has been chosen to be used in UMTS 3G mobile networks.)
- Segmented Integer Counter Mode - a typical counter mode, which allows random access to any blocks, which is essential for RTP traffic running over unreliable network with possible loss of packets. In the general case, almost any function can be used in the role of "counter", assuming that this function does not repeat for a long number of iterations. But the standard for encryption of RTP data is just a usual integer incremental counter. AES running in this mode is the default encryption algorithm, with a default encryption key length of 128 bits and a default session salt key length of 112 bits.
- Authentication, data integrity and replay protection: to accomplish this stage of protection the AES cipher suite is not sufficient, because AES does not provide these features. For the authentication purpose the Srtcp protocol use the HMAC-SHA1 algorithm (RFC 2104), the result of the HMAC-SHA1 operation is 160 bit. Half of this data stream is used to fill the authentication tag loaded in to the packet. The HMAC function is calculated on the packet including the header and the sequence number. The receiver has just to keep the indices of the last messages, compare them to the index of any new packet received, and let in only messages with an index that has not been received earlier. In this way there is a protection against the replay attack. As far as data integrity and authentication are concerned, the HMAC function is enough.

The most common key management protocol used with the Srtcp protocol is Mikey, which stands for Multimedia Internet Keying. Defined in the RFC 3830, this protocol works in three different ways:

- Pre-shared Secret Key (PSK) - It is used to derive subkeys for encryption and integrity.
- Public Key Encryption (PKE) - The caller generates a random encryption key, which is then sent to the remote user. The message is encrypted using the public key of the receiver. With such configuration a central repository for public keys PKI (Public Key Infrastructure) is needed.
- Diffie-Hellmann (DH) - Optional. This method is the most resource intensive, but it is the only one providing the "Perfect forward secrecy" . This method can only be used in peer-to-peer key negotiations and requires an existing PKI infrastructure.

Actually there are few hardphone vendors supporting the Srtcp/Srtcp protocol suite. This is probably owing to the fact that securing Voip does not seem very important now. The Srtcp/Srtcp suite protects

the Rtp flow, but the Sip signaling is completely unprotected. This weakness can be good for malicious user.

2.4.3 The Zrtp protocol

The Zrtp protocol is a protocol for key agreement, based on ephemeral Diffie-Hellman algorithm. It is used to establish the session key and other parameters to set up Secure Real-Time protocol sessions. It was proposed by Phil Zimmerman, the creator of PGP. The key management protocol is without any server and PKI (Public Key Infrastructure). This protocol completely works in an end-to-end mode. The protocol discovers if the caller and the called endpoints supports the encryption parameter. The only implementation of the Zrtp is the Zfone project, which is still in a Beta Version, downloadable from [2]. Zfone it is a plugin for other softphones: X-Lite, Gizmo, Apple iChat AV (audio and video), XMeeting, and SJphone, and it also works with some VoIP service providers: Free World Dialup, iptel.org, and SIPphone.

How does Zrtp works?

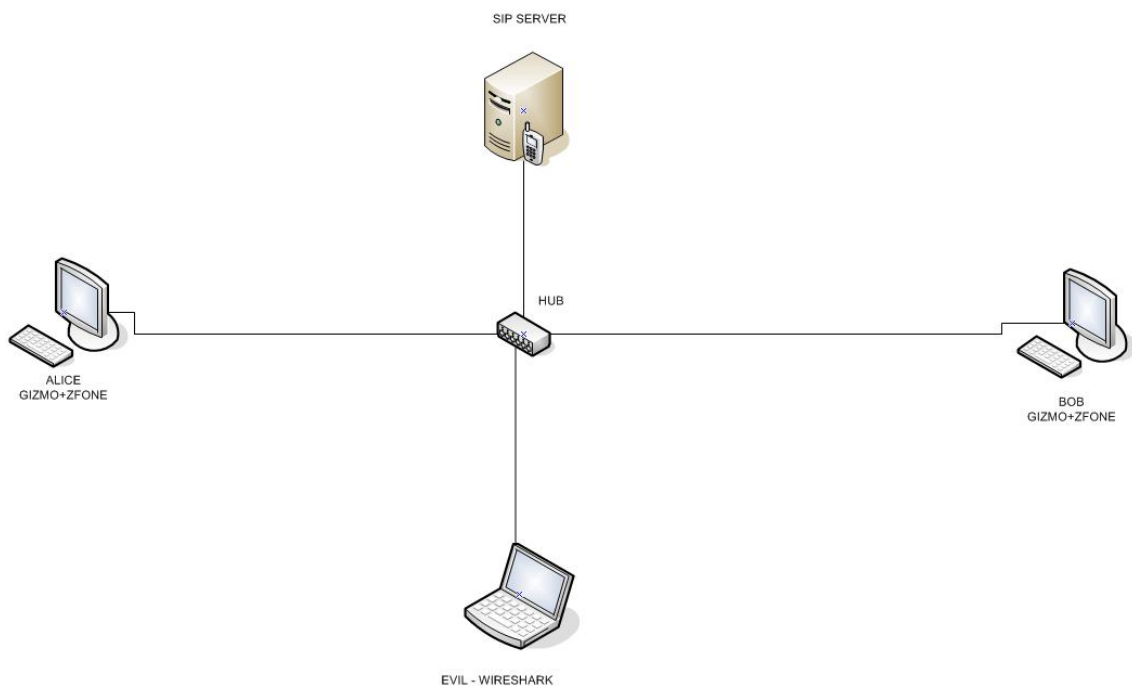


Figure 2.5: Zrtp working example

The picture shows that the Zrtp is an end-to-end protection not implying the components of the Sip architecture.

The user Evil could not perform the "Man in the middle" attack, on the Rtp stream at least. Bob and Alice can use a different softphone like X-lite or Gizmo, for example. After the Sip registration within the REGISTRAR server, the speaking is completely protected from a malicious listener using Zfone. The Zrtp protocol does not refer to Sip signaling and for Evil it is possible to make a "MiTM" attack from this part of the Voip architecture.

As we described above the protection of the Rtp stream is not enough to make secure a voip infrastructure. Let's say that it is "not completely unsecure". In order to protect signaling, it is necessary to use a TLS infrastructure. For example, this may be possible with the last version of OpenSER.

2.4.4 The TLS protocol

The TLS protocol allows network applications to communicate in a secure way, to prevent eavesdropping, tampering, and message falsification. TLS permits to the endpoints of a communication authentication and communications privacy over the Internet using cryptography. Commonly, only the server is authenticated while the client remains unauthenticated. This means that the end user (whether an individual or an application, such as a Web browser) can be sure with who he is talking.

TLS concern three basic phases:

- Peer negotiation for algorithm support
- Key exchange and authentication
- Symmetric cipher encryption and message authentication

During the first phase, the client and server negotiate the cipher suites, which determine the ciphers to be used, the key exchange and authentication algorithms, as well as the message authentication codes (MACs). The key exchange and authentication algorithms are typically public key algorithms, or as in TLS-PSK pre shared keys could be used. The message authentication codes are made up from cryptographic hash functions using the HMAC construction.

Typical algorithms could be:

- For key exchange: RSA, Diffie-Hellman, DSA, SRP, PSK
- Symmetric ciphers: RC4, Triple DES, AES or Blowfish.
- For cryptographic hash function: HMAC-MD5 or HMAC-SHA are used.

Voip security issues and solutions

3.1 Introduction

The voip infrastructure is very complex as we showed in the second chapter. In this chapter we will perform a deep analysis of the security problems regarding this technology.

3.2 Physical layer

3.2.1 Cable hacking

If we are in presence of a Ethernet network provided with a hub or a switch, it's possible to create an invisible sniffer to listen to all of the communication on the wire. Cable hacking is probably a hard core experiment, because it implies cutting some wires of the eight composing the Ethernet cable. There are two different cable hacking methods, the easiest one is to cut the TX- wires, the green and the orange ones respectively, of the T568A and T568B EIA/TIA (Electronic Industries Association/Telecommunications Industry Association) standards cable configuration. With this modification the hub thinks there is somebody connected to the port and sends the data. But in case we would like to transmit data to an other machine, our information is interpreted just as a noise on the wire and completely ignored by the other machines. An other possible cable modification is to build-up a one-way cable, we simply need to connect two wires, the RX- and RX+, the pin three and the pin six, respectively. By so doing, we are absolutely sure that our sniffer machine cannot transmit data over the network, but it can receive them. This is enough to put the network adapter in promiscuous mode.

In case of a switched environment we need a special utility to overload the switch in a fail-safe mode - a hub. The tool that we used to perform the switch fail-safe mode is macof, which uses a mac flooding technique. Macof is a tool included in the dsniff suite, realized by Dug Song.[4]

3.2.2 Wifi wardriving

In case of wireless communication there is a higher possibility of sniffing, because there are a large number of unprotected networks or weakly-protected networks. If a wifi infrastructure is unprotected, it is evident that an attacker would easily sniff the communication. And also it might be easy for an attacker if the network is WEP protected. Owing to the popular weak point of the WEP technology. In such case, it is better to use an effective wireless protection method (WPA-TKIP, RADIUS, etc)[27]

3.3 Data Link layer

3.3.1 Arp Poisoning

This technique is well known in case we would like to sniff some information in a switched environment. The attacker wishing to perform an Arp Poisoning attack should poison the arp cache at both endpoints of a call, or the cache of a SIP server/proxy and a client. This is a possible attack because some operating systems accept the ARP reply messages from the network regardless of fact they have sent an ARP request. In this way, it is possible to modify the arp table of some machines and eavesdrop a communication thus making an identity theft. Sometimes credit-card transactions are made by telephone. In this case the information passing though the wire are very sensitive and a sniffing attack could become a big problem. The first solution to such a problem is to use static arp tables. In this way, is not possible to accept arp reply/request messages from the network. The first suggestion for a network administrator is to install Arpwatch, a useful tool to detect an in-coming arp attack.

3.3.2 Vlan flooding technique

The first positive aspect of adopting the Vlan technology is the possibility to mitigate call sniffing. Because this technology can keep data and voice traffic separate, according to the IEEE 802.1Q standard.

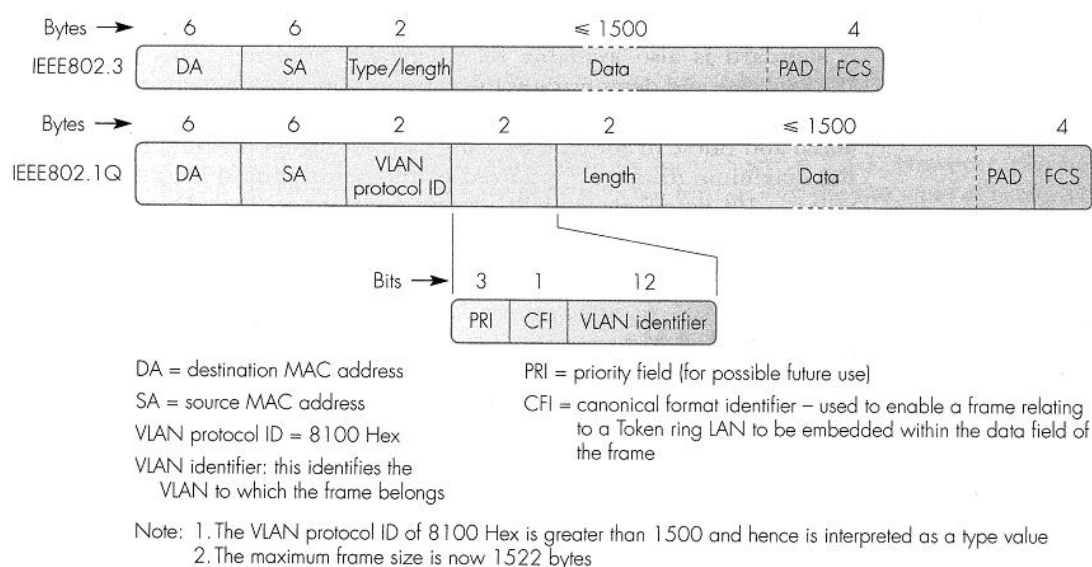


Figure 3.1: The 802.3 and 802.1Q headers

But even if we use Vlans we are not immune to attacks, the first and the easiest way to attack the Vlan environment is by using the mac flooding technique, we can use macof, the tool mentioned above.[4] There are several attacks that could affect a Vlan, it also depends on the vendor of the network hardware that we are using. The most famous is Cisco, all of the information about Vlan vulnerability could be retrieved on [3] the Vlan security white paper. Talking about the QoS the Vlan schema has a special priority field that it could be used to increase the voice packet scheduling, this solution could be helpful to increase the performance and the quality of a voip communication.

3.4 Network layer

3.4.1 Ip spoofing

The Ip spoofing is a technique that was only theoretical until Robert Morris Senior, the father of the author of the Internet Worm, discovered the possibility to predict the Tcp sequence number. The source of this problem is in the nature of the Ip protocol, that has not an internal mechanism to check if the source address in the protocol header is a fake address or it is owned by someone. The Ip spoofing technique is not only based on the above mentioned weak point of the Ip protocol. It also involves the Tcp protocol. For this reason it is considered as an attack on the TCP/IP suite. With reference to old operating systems, the Tcp issue concerned the sequence number random generation. Such number was not effectively generated at random. It was therefore quite easy to retrieve the number and effectively join the communication between two unaware endpoints. The Ip spoofing is another technique to realize

the man in the middle attack. The first attack of this kind was noticed by Tsutomu Shimomura at the San Diego Supercomputing Center, it seems that the author was Kevin Mitnik "The Condor". Actually, with modern operating system is not so easy to predict the Tcp sequence number.

3.4.2 Ip spoofing: an example

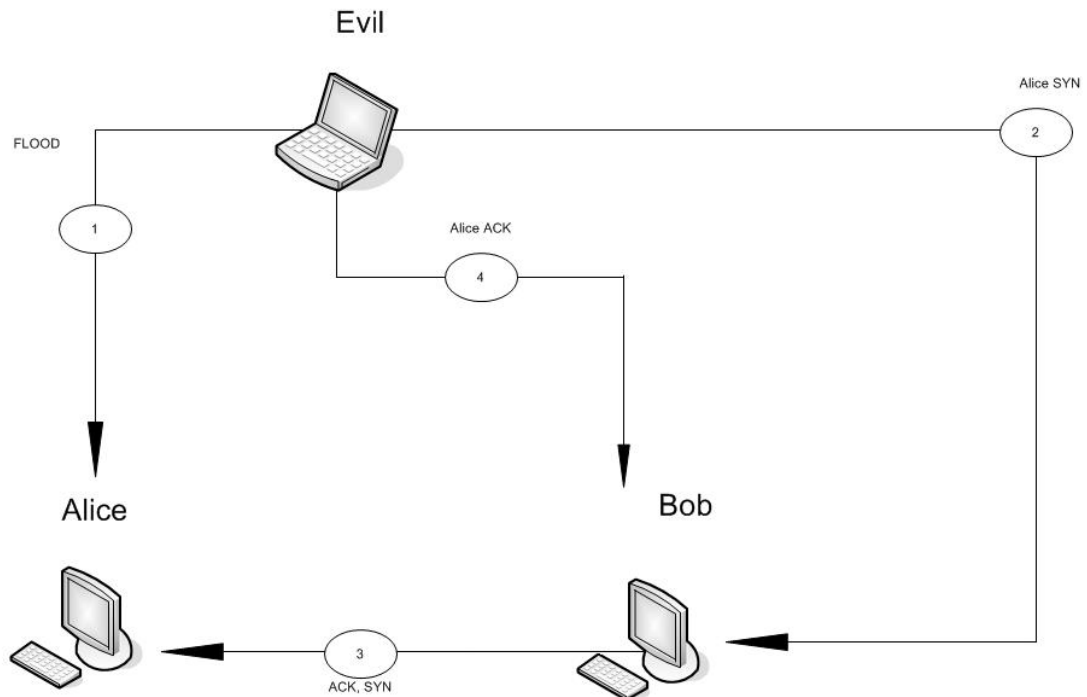


Figure 3.2: Explaining the Ip spoofing

- 1 - The attacker, Evil, opens several Tcp connection to guess the way of generation of the sequence number of Alice, and flood the Alice's machine, in order to block any response from this machine.
- 2 - The attacker sends a SYN message to Bob pretending to be Alice.
- 3 - Now Bob sends ACK,SYN message to Alice, that cannot answer to this message, because is flooded. Now Evil has to guess correctly the sequence number in order to give a correct ACK message to Bob.
- 4 -Now Evil sends the last packet to complete the handshaking procedure, the packet is an ACK message with a correct sequence number. Now the attacker can send some commands to Bob, pretending to be Alice.

3.5 Transport layer

3.5.1 Udp flooding

This attack is directed to the quality of a Voip communication, because the Rtp data flow is transmitted through Udp protocol. The Udp protocol is connection-less oriented. It is therefore easier to introduce a spoofed packet on the network, simply to produce more traffic so as to overload the connection. To prevent and mitigate this issue, the ordinary data on the network could be separated from the voice through the use of Vlans. .

3.5.2 Tcp SYN flooding

This kind of attack could be included in the Denial of Service section since it could be launched to overload an application server (Sip proxy, registrar, redirect, etc) to be unable to accept any more requests. It is an attack consisting of a stream of Tcp packets with a spoofed address and an activated Syn flag. The application server starts a number of new handshaking procedures, which go to timeout, up to the moment the operating system timeout is reached, a lot of resources are engaged and the server cannot respond to normal Syn requests. It is a hard problem to distinguish between hostile and non-hostile messages. The first thing to think is not to use standard Tcp ports. By so doing, only who knows the correct port can send Syn messages to the server. Deploy a TLS authentication system is another solution, to prevent hostile-users and to increase the overall security level.

The goal of a flooding (Denial of Service) attack is to disrupt some legitimate activity, such as browsing Web pages, listening to an online radio, transferring money from your bank account, or even to interrupt a voip call. This denial-of-service effect is achieved by sending messages to the target that interfere with its operation, and make it hang, crash, reboot, or do useless work.

3.6 Application layer

3.6.1 Sip protocol hacking

- Registration hijacking: If the User Agents are registered to a Sip Proxy, the inbound calls are properly forwarded by the proxy, to the correct destination. But if somebody can modify the correct parameters of the registration, and forward the call flow to a rogue application, inbound calls will be listened by somebody else. Also if is used the digest authentication option, there are tools to deploy a off-line password cracking.
- Redirection response attack: If an attacker can reply to a Sip Invite message with certain responses is possible to hijack the inbound calls to another Sip phone or to deny the connection to a user that is fully capable to answer.

- Sip phone reconfiguration: If the administrative password of the Sip phones are not enough strong it is possible to modify all of the parameters. For example is possible to register a phone to another registrar that is acting like a man in the middle, listening all of the conversation.

3.6.2 Dhcp Exhaustion

A lot of Voip phones are configured to directly acquire an Ip address using the Dhcp protocol. If an attacker would like to block some Ip phones, he simply has just to cheat the Dhcp server to exhaust all of the free Ip addresses. The "Internetwork Routing Protocol Attack Suite" (IRPAS) contains a tool named dhcpx that is very useful in this sense. We simply have to insert the Ip address of the Dhcp server.

The solution proposed is easy, we have to configure the Dhcp server so as not to lease addresses to unknown mac address and to untrusted network segments.

Chapter 4

Voip Security:Our Experiments

4.1 Experiment summary

After a careful and deep analysis about the security issues that could affect a voip platform, we tried to realize our platform, in order to understand how the security measures could bias the Quality of Service. The first problem we encountered was the layout of the results. To provide a comparable measurement in the voip area it is necessary to refer to the ITU-IT recommendations. This advises are made to clarify how to evaluate the voice-traffic, the most famous parameters noticed in the ITU-IT recommendations are the MOS "Mean Opinion Score" and the R-Factor. The first one is an evaluation scale between 1 and 5, where 5 is the best quality speech . This parameter is subjective. On the contrary the R-Factor is a scalar and based on the E-Model, a computational model for performance evaluation of data networks. The formula to calculate the R-Factor is very complex and we needed a software suite - Ixchariot - to make a simulations on our platform and retrieve comparable results.

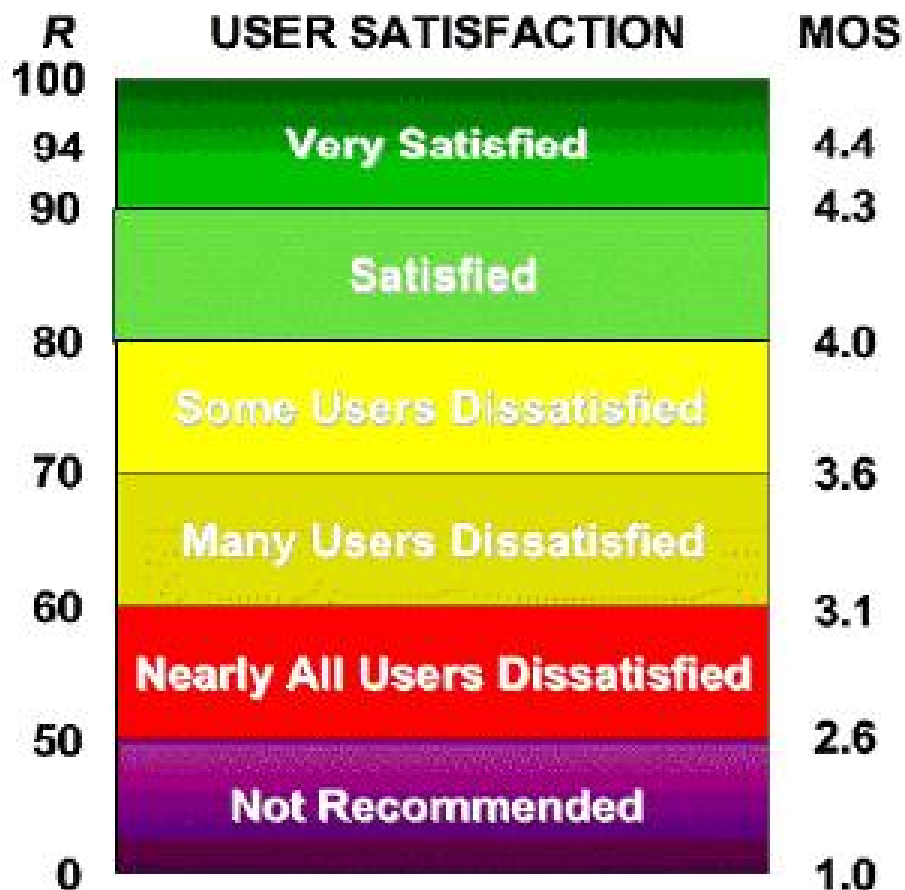


Figure 4.1: R-factor - Mos

The simulated test consists of:

- 10-Mbit network bandwidth in Milano, Italy and 6-Mbit network bandwidth in Ostrava, Czech Republic.
- Use of G.711Alaw and G.729 audio codecs.
- Use of an overloaded network or of an ordinarily loaded network. In order to deploy the overload test, we used Iperf, thus overloading the network with 4-Mbit Udp traffic.
- Activation of security measures or deactivation of security measures. The security solution

adopted is OpenVpn.

- To reduce the margin of errors on the calculation we made the tests five times.

The results of the simulations are documented in [12]. We also carried out real-tests, to make a comparison between the simulation and the measurement of the degree of reliability of the simulator. During the execution of such real-tests we tried some of the several mentioned above attacks. In order to understand the efficiency of our security solution. We made the tests to point out how the security measures could prevent or mitigate the security risks of the Voip technology. The tests were also useful to quantify the amount of bandwidth needed to introduce security measures. In this way we measured the interaction between the voice quality and the introduction of security measures.

4.2 Experiment configuration

The basic configuration of the tests required 2 Pcs provided GNU/Linux Debian. One was based in the Laboratory of Security and Computer Networks in the department of Computer Science of the University of Milano, Italy. And the other Pc inside the Voip Laboratory, part of the VSB-Technical University of Ostrava, Czech Republic. Both universities are connected to the research network Geant2. It is really a significant advantage to be part of high speed network because the end-to-end delay between our endpoints had been approximately 30 ms.

1 ms, cesare.laser.dico.unimi.it
2 ms,159.149.153.254
1 ms,ssr1-ssr7.bone.dsi.unimi.it
2 ms,159.149.251.21
2 ms,159.149.254.25
2 ms,ru-unimi-rt-mi3.mi3.garr.net
2 ms,rt-mi3-rt-mi1.mi1.garr.net
2 ms,garr.rt1.mil.it.geant2.net
14 ms,so-6-3-0.rt1.vie.at.geant2.net
21 ms,so-7-0-0.rt1.pra.cz.geant2.net
22 ms,cesnet-gw.rt1.pra.cz.geant2.net
29 ms,r96-r50.cesnet.cz
29 ms,iptel-21.osanet.cz

4.2.1 The Simulation

To prepare the test-bed environment for the simulation we needed to install Ixchariot endpoints on both PC. For the test configuration we installed the Ixchariot console on a laptop computer. To avoid possible influences on our results, owing to the control mechanism of the Ixchariot console, we choose the batch run test configuration. This allows to send the results to the console only when the test is finished. The Ixchariot software suite is making all the tests using its own endpoints.

4.2.2 Real-Tests

OpenSer was installed in Ostrava. In addition to OpenSer, we needed another software to generate Sip/Rtp traffic. For such reason, we decided to adopt Sipp. [19]

4.2.3 Security Measures

As previously stated the security measures adopted were always the same. Thanks to OpenVpn we were able to protect the Sip signaling and the Rtp data flow, with one simple, security solution.

4.3 Experiment tools

In this section we will present the tools used for our experiment.

4.3.1 IxChariot

This software, produced by Ixia, it is useful to predict device and system performance under realistic load conditions. The test environment is build with the IxChariot console and two IxChariot endpoints (Installable under several operating systems). The console appliance allows to select several test configuration for eg:IPv4 with and without QoS or also with IPv6. At the end of the test, for each possible configuration it is possible to obtain measurement of the throughput, jitter, MOS and R-Factor. The best way to execute a test is by the batch procedure, because in this way the final results are send to the console only at the end of the test. In such way is possible to avoid some influence, due to the result data, during the test.

4.3.2 OpenVpn

This Open Source software it is used to set-up Virtual Private Networks by the TLS protocol, explained in the second chapter. Using this software it is possible to obtain a private network, encrypted with the strongest cryptographic algorithm actually known. To obtain a better performance the asymmetric cryptography is used only for the key exchange, while for the message encryption are still used symmetric algorithms. In this way the encryption procedure is faster.

4.3.3 OpenSer

OpenSer is an open source SIP server. It can be used on systems with limited resources as well as on Service Provider servers. It is able to manage up to thousands call setups per second. This software is written in pure C for Unix/Linux-like systems with architecture specific optimizations to offer high performances. This server can be configured to act like: registrar, location server, proxy server, redirect server; gateway to SMS/XMPP, and also like an advanced VoIP application server. [See chap.2] The latest version supports the user authentication by the TLS protocol, in this way the Sip signaling can be protected. Usually the configuration does not involve the Rtp data flow, that is passing end-to-end, for this meaning, during our test, we choose to protect all of the data with a Virtual Private Network. Is necessary to underline that OpenSer cannot act like a:

- Back-to-back user agent
- Media server
- SIP phone

OpenSer and MySQL

For the registration of the users OpenSer has the possibility to configure the MySQL database. The tarball of OpenSer contains a file named mysql.db.sh, it is necessary to execute this file to start the creation of a new OpenSer database.

The configuration of OpenSer requires an extensive knowledge of the Sip protocol, following we will explain our test configuration.

The configuration file: openser.cfg

The configuration file starts with the following lines:

```
#####OpenSer CONFIGURATION FILE#####  
  
# ----- global configuration parameters -----  
debug = 3  
fork = no  
log_stderr= yes
```

Those three options set the stage for how OpenSer will work. By not forking and logging to STDERR one can run OpenSer directly to gather information about an invalid configuration directive or perhaps why a specific module is not loading as expected.

```
listen = 192.168.1.158
alias = 192.168.1.158
port = 8888
```

The next three instructions inform the software which interface and tcp/udp ports to listen on.

```
children = 4
dns = no
rev_dns = no
```

The children instruction informs OpenSer of how many "child" threads to keep around to process incoming messages. This number has to be decided considering the system performances and the average number of messages that your installation could receive. Usually OpenSer does not need to know about DNS names, but if there is a specific reason for this option it is possible to enable it also in the reverse mode.

After the global configuration parameters in the configuration file, we need to configure which modules to load and for each module the parameters that we want to enable.

```
# ----- module loading -----

mpath="/usr/lib/OpenSer/modules/"
loadmodule "tm.so"
loadmodule "mi_fifo.so"

#-----modules parameters-----
modparam("tm", "wt_timer", 2)
modparam("mi_fifo", "fifo_name", "/tmp/OpenSer_fifo")

# ----- request routing logic -----
```

In this example we just loaded the tm module and the mi_fifo, with their respective parameters. There are several different modules for OpenSer, useful to manage a lot of different situations eg: digest authentication, nat helper, TLS authentication, mysql database, Rtp proxy, etc.

After the above sections there is the most important. In OpenSer every Sip message gets processed through a block of code named route. In our special test configuration we wanted only that our messages had to be forwarded to the application. In our case the application was Sipp, that was waiting for the incoming calls from Milano. So we did not configure any other kind of routing logic.

```
# ----- request routing logic -----  
  
# main routing logic  
  
route{  
    t_relay();  
}
```

In this section is possible to insert a routing logic for each specific Sip message. In this way it is possible to manage and interact with several scenarios, eg: Sip proxy, Sip registrar, Sip redirect, etc.

4.3.4 Iperf

Iperf is a tool to measure maximum TCP/UDP bandwidth, allowing the tuning of various parameters and characteristics. Iperf reports bandwidth, delay jitter, datagram loss. This software permit to generate a pre-defined amount of traffic for a pre-defined duration. This last feature was very important for us, in order to configure the tests over stress conditions.

4.3.5 Sipp

SIPP is a free Open Source test tool / traffic generator for the SIP protocol. It includes basic user agent scenarios (UAC and UAS) and establishes and releases multiple calls with the INVITE and BYE methods. It is possible to build custom XML scenario files describing from very simple to complex call flows. The result of a test are dynamically displayed during the execution (call rate, round trip delay, and message statistics). For complex tests is possible to have periodic CSV file with statistics dumps.

Other advanced features include support of IPv6, TLS, SIP authentication, conditional scenarios, UDP retransmissions, error robustness (call timeout, protocol defense), call specific variable,

One of the most important feature is the possibility to send media (RTP) traffic through RTP echo and RTP / pcap replay. Media can be audio or audio+video. We used this option to generate real calls with G.729 and G.711Alaw audio codecs.

Sipp: the test configuration

Now, let's take a look on our Sipp configuration, in order to understand how this program was interacting with OpenSer. The command that we gave to our Sipp in Milano was:

```
./sipp -sf uac_pcap_729.xml 195.113.113.147:5060 -rsa  
195.113.113.147:8888 -l 70 -m 1000 -r 10
```

As you can see we loaded the XML scenario where we modified some lines to play an audio file encoded with G.729. For the details have a look on appendix B. The address in Ostrava 195.113.113.147:5060 was the answering instance of Sipp, while the 195.113.113.147:8888 was the OpenSer server. The Sip messages generated were passing through OpenSer and after to the instance of Sipp. The "-l 70" directive was to do not overcome the limit of 70 concurrent calls. While the "-m 1000" option was to set the maximum limit of total calls to generate. Instead the last option "-r 10" was to increase the concurrent call number of 10 every second.

The instance of Sipp running in Ostrava had a simple configuration, just listen on the 5060 port and register the statistics.

```
sipp -sn uas -p 5060
```

Chapter 5

The Results

In this chapter we will show the results of our tests. To better clarify our work, the chapter is divided in two macro-areas. One is to explain the simulated tests and the other-one is to depict the real-tests. In every macro-area we showed the test, the relevant comments and the results. Each macro-area is divided into a secure or an unsecure test-mode. This subdivision was necessary to point out the value of the security measures. According to our aim, during the real-tests we tried to attack the system in some of the ways listed in chapter 3.

5.1 Simulated tests

As stated before, the test were made by using the software Ixchariot. The following figure shows the network infrastructure adopted.

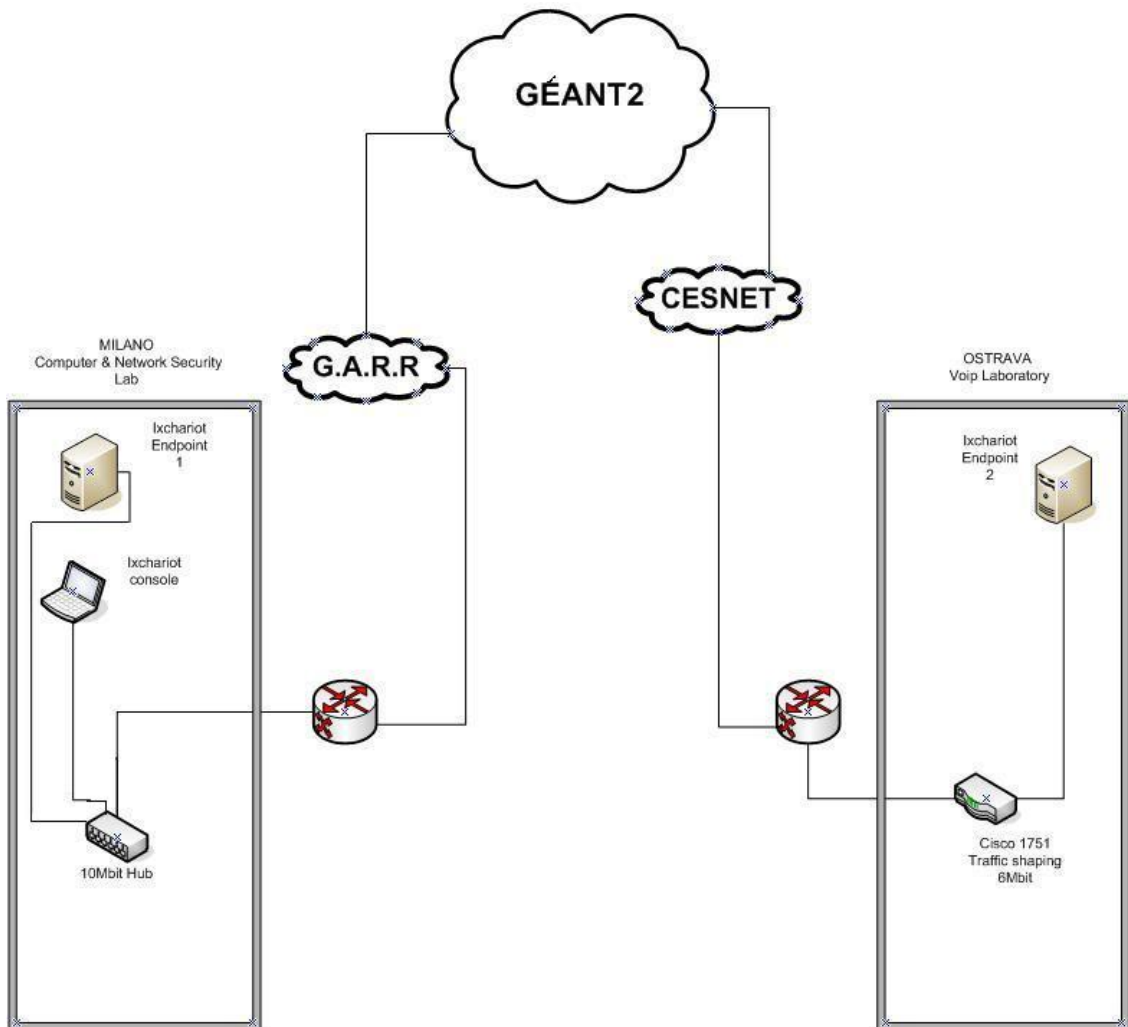


Figure 5.1: Testbed of the simulation

5.1.1 Unsecure simulation

The test was configured to execute 50 concurrent calls using either the codec G.711Alaw or G.729 codec. The available bandwidth was 10Mbit full duplex in Milano and a 6Mbit full duplex in Ostrava. Besides using different codecs we carried out the tests adopting the Iperf traffic generator, so as to understand how the deployed infrastructure can behave in overloaded conditions. The call lasted only one minute. The jitter buffer was 60msec. The audio codecs G.729 and G.711Alaw are two different format of audio compression.[25][26] The first one, is common used with Voip devices, the standard configuration of G.729 operates at 8 kbit/s bitrate, a low bandwidth requirement. This codec is patented by Sipro. The use of G.729 may require a license fee. The G.711Alaw codec offer an higher level of

performance respect of the G.729, but takes more bandwidth. The level of bitrate of this codec is 64 kbit/s. This codec is a standard audio compression format defined by the ITU-T.

5.1.2 Result forecast

As proven in [12] 50 simultaneous calls via the G.711Alaw codec take approximately 9Mbit of traffic. On the contrary, by using the G.729 codec, the bandwidth usage decreases to 5.5Mbit. According to our forecast the unsecure test can be completely supported by the infrastructure adopted, even in overload conditions.

5.1.3 Test result

As it may be understood from the following pictures concerning both of the codecs used, the value, of the R-factor index differs little whether is only 1 or 50 concurrent calls, in ordinary load conditions. But when performing the test with Iperf we appreciated a 5% performance loss with the G.711Alaw codec and 2% with the G.729 codec.

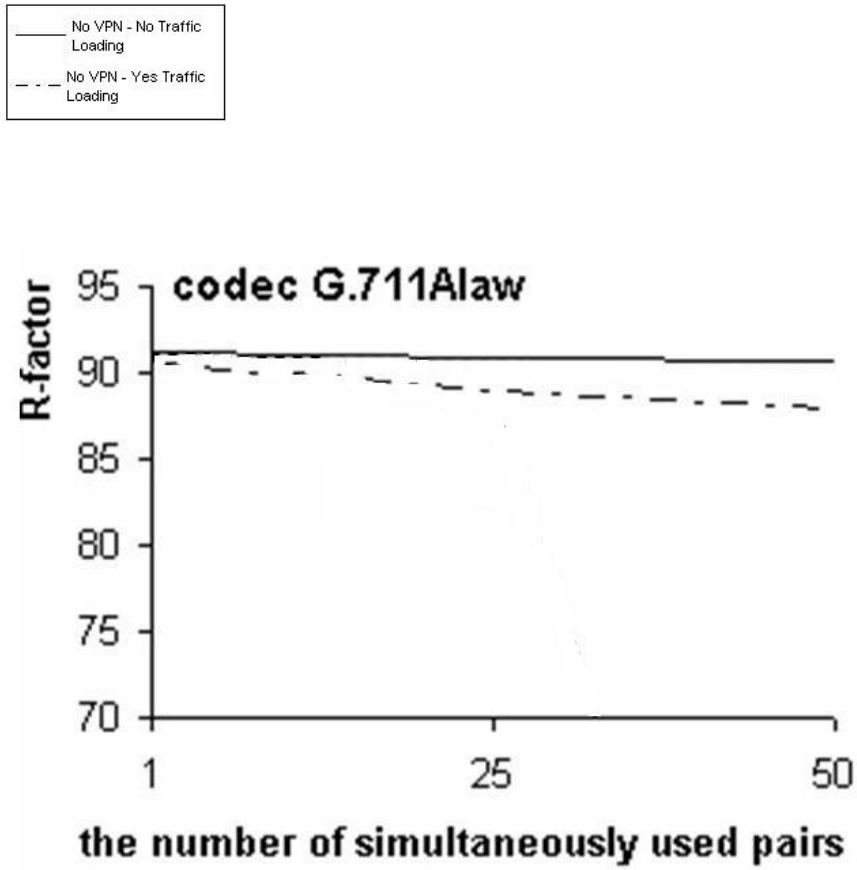


Figure 5.2: G.711Alaw - NOVPN

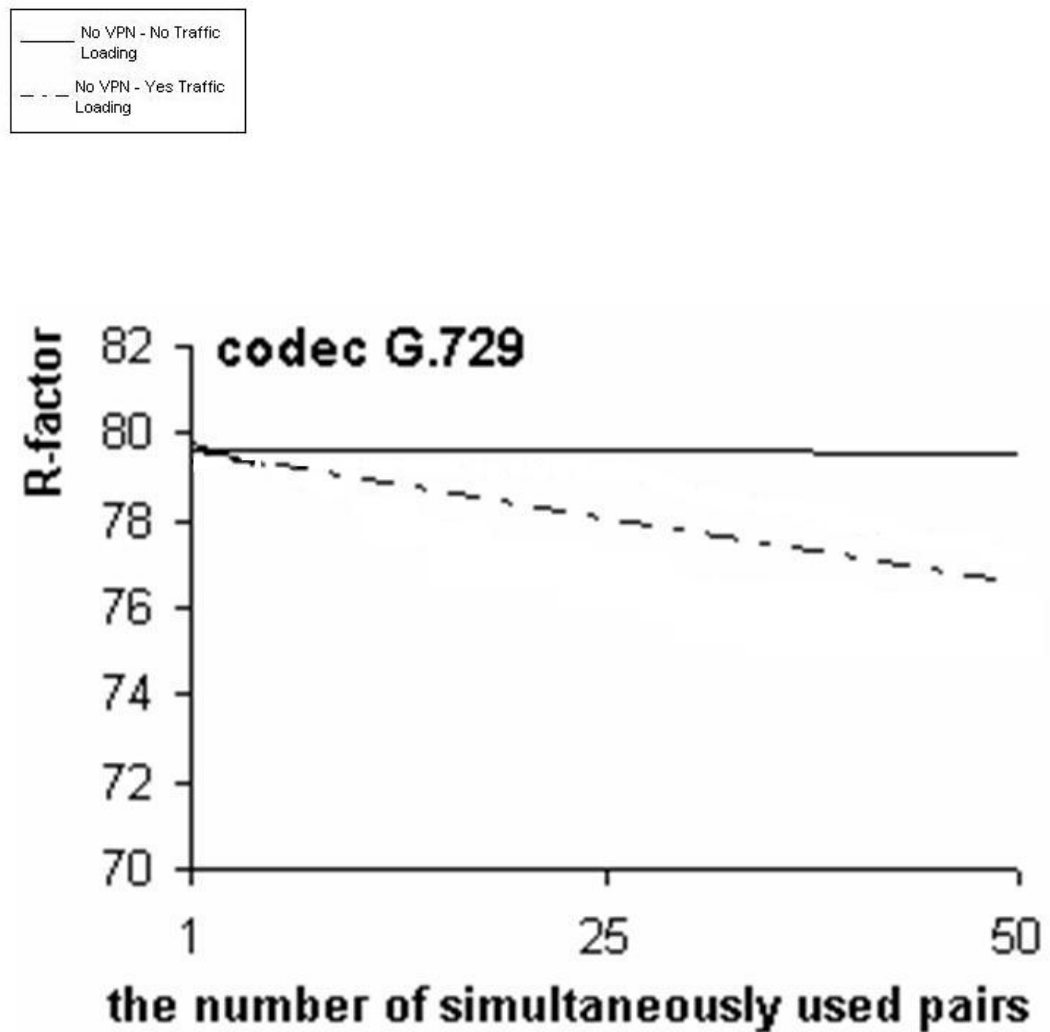


Figure 5.3: G.729 - NOVPN

It is necessary to point out that the two codecs have different performance indexes, because they have different bandwidth requirements. The G.729 codec supported the load in a better way because it has minor bandwidth requirements. But the overall quality of this codec is lower than the one of the G.711Alaw codec.

5.1.4 Secure simulation

The configuration of this test differs from the unsecure one just because this test was launched inside a Virtual Private Network realized with OpenVpn. In the appendix B, it is possible to find the configuration file used to perform the test.

5.1.5 Result forecast

The usage of the Transport Security Layer mechanism, lead to an increase in the request of bandwidth. We were to observe some performance loss and appreciate the differences between the different codecs.

5.1.6 Test result

According to the results obtained we can say that a secure infrastructure needs to be supported by increasing the available resources. To a Service Provider this means an increase in costs . The economic impact of a security measure is usually one of the first issues when an institution decides to adopt security measures. This aspect is well explained in [9]. Using the G.729 codec in normal conditions, the impact of the VPN reported with 50 simultaneous calls equalled just one percent. This difference is so little because, as said before, this codec has a little bandwidth request. But as the reader can see, the maximum R-factor reached with 50 calls is 75, not very good quality. See figure 4.1.

When we used the G.711Alaw codec with no traffic overload the VPN influence recorded was just one percent. But with the traffic overload the network infrastructure collapsed and we reached only 25 concurrent calls with an R-factor index of 90. A very good result, considering the VPN and the traffic overload influences.

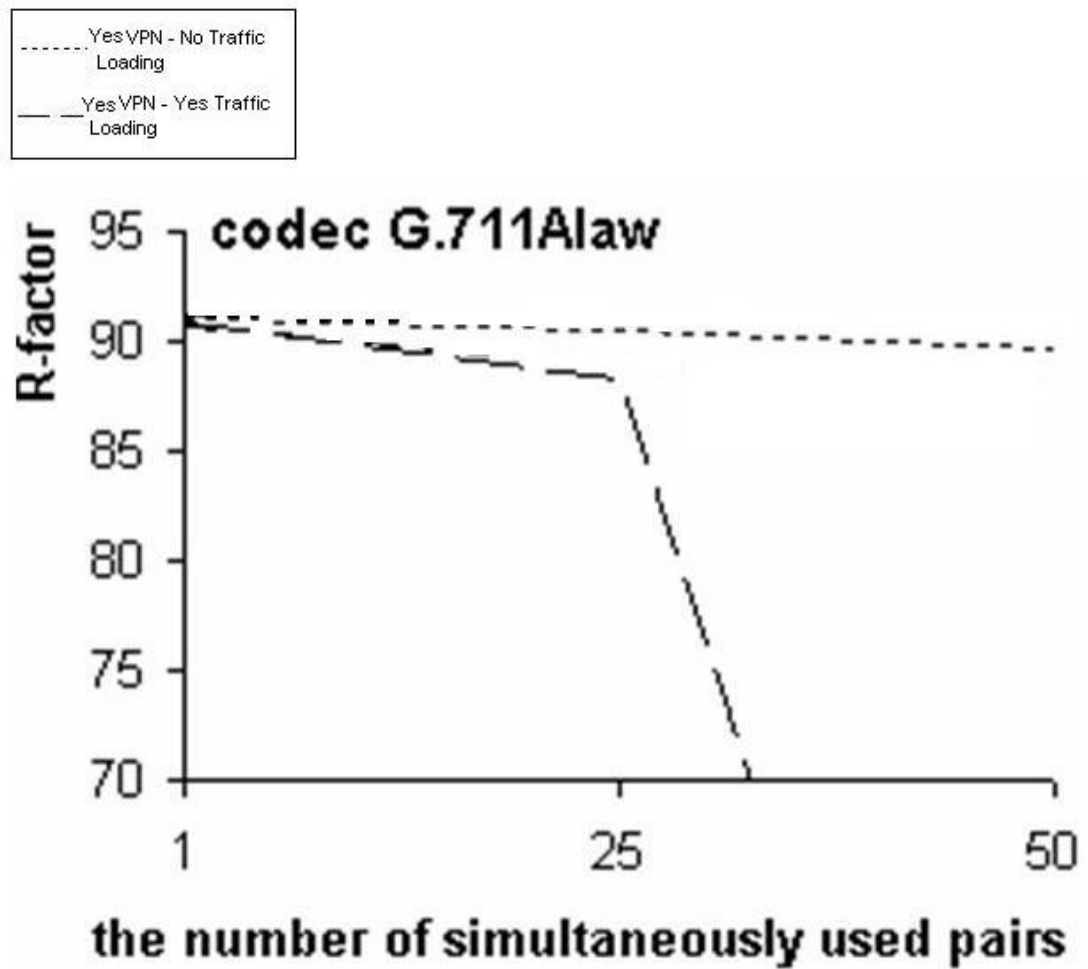


Figure 5.4: G.711Alaw - VPN

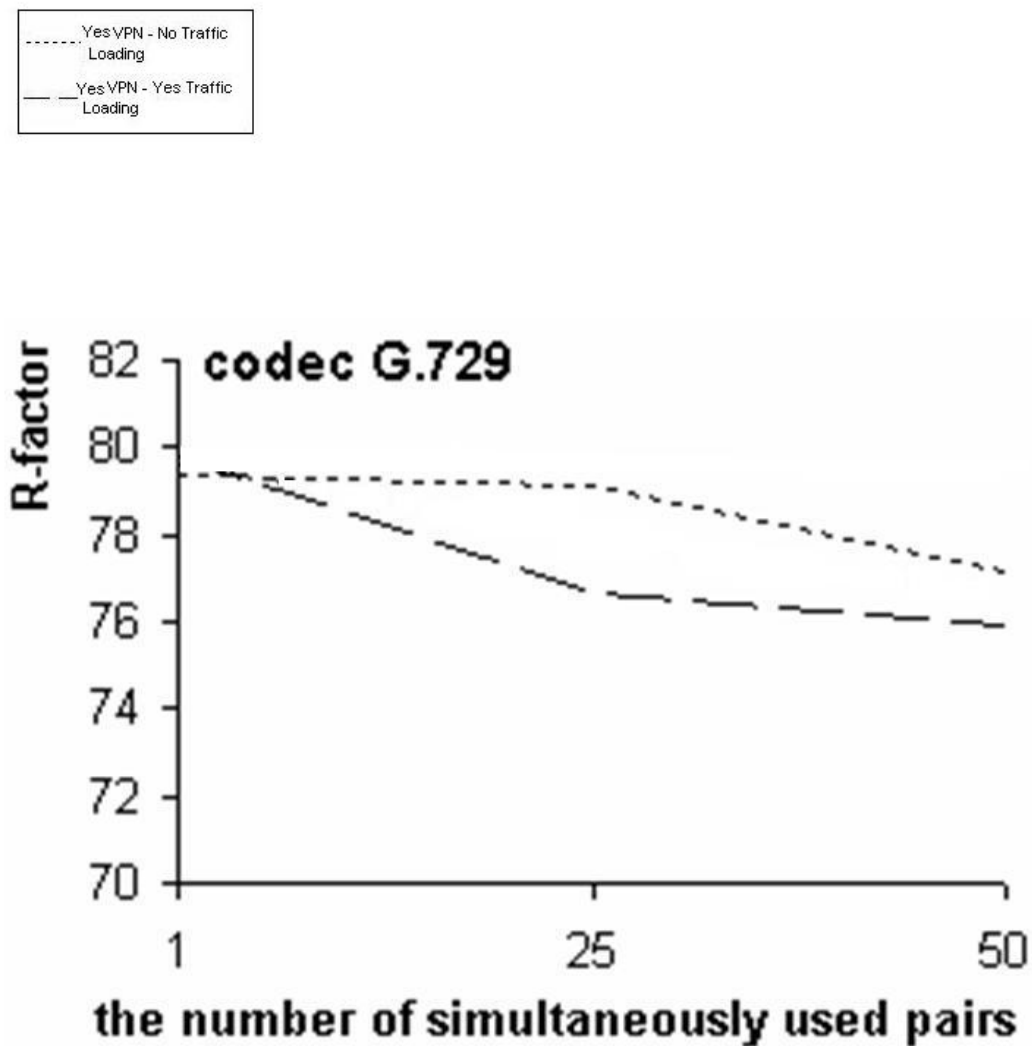


Figure 5.5: G.729 - VPN

5.2 Real tests

To realize this part of the tests we used OpenSer and sipp. The latter is a software that allows the generation of sip/rtp traffic. The configurations details of this software are well described in appendix B. To obtain valid G.711Alaw and G.729 traffic, it was necessary to capture one of our test call with Wireshark. The captured traffic was used during the tests to generate the Rtp traffic. The maximum concurrent call limit in the real-test was 70. At the beginning of the test the simultaneous calls are

10. This number increases by 10 every second, so after 7 seconds we reached the limit of 70. Every call lasted 60sec, once the calls were finished the test was restarted up to when the overall number of generated calls reached 1000. This means that the test of 70 concurrent calls had been repeated about 15 times. While executing the test we tried to attack the network in some of the afore-mentioned ways so as to understand the level of resistance of the network itself. The OpenSer server was based in Ostrava and the Sipp client in Milano. On the machine in Ostrava we also configured another instance of Sipp, in order to answer to the calls addressed by OpenSer.

5.2.1 Unsecure test

During this test we tried the following attacks:

- Man in the middle
- Sip flooding
- Udp flooding
- Sip registration hijacking

5.2.2 Result forecast

We estimated that the amount of traffic generated by 70 concurrent calls can be supported by our architecture, at least with G.729 codec. Our aim was to discover the maximum limit of resistance on a Denial of service attack. Furthermore to confirm the vulnerability to some attacks as MITM.

5.2.3 Test result

Man in the Middle

As you may see in Figure 5.1, in the Lab based in Milano we had a network segment with a hub. By using Wireshark, we encountered absolutely no problems in sniffing all of the conversation passing through the wire, as predicted. Basically, in a network segment provided with a switch, it is not a big problem. By using the Arp poisoning technique it is possible to eavesdrop a conversation. Such attacks do not concern performances. It only was passive eavesdropping.

Sip flooding

With the usage of Sipp it was possible to perpetrate the sip flooding attack. This kind of attack consists of a massive delivery of registration requests to a server. In order to exhaust the available resources. In our case, the OpenSer software was very strong. To obtain a slow call-processing we had to deliver 500 registration messages per second other than the traffic generated by the test. This kind of attack causes

problem to the users of the service, because the management of the signaling can slack down very fast. During the attack, we received some messages of "BYE" to close a call, with a delay of 30seconds.

Udp flooding

With the use of Iperf it is possible to generate tcp/udp traffic and address it to any destination as "IP.ADDR:PORT". The test was executed with a load of 4Mbit. Using the codec G.711Alaw the charge of 4Mbit is too much for 70 concurrent calls and it was impossible to obtain a good level of performances. On the contrary, by using the G.729 codec it was necessary to increase the traffic loading until 4.75Mbit to obtain a considerable performance loss.

Sip registration hijacking

The SiVus [21] tool, is very useful to test the vulnerabilities of a voip infrastructure. We used it to try the Sip registration hijacking technique over our Voip environment. This technique is well explained in [5] and on [23]. To delegitimize the user's registration the sip hijacking works as follows:

- Adopt a Dos technique to attack the Sip Phone victim, to put him in an unavailable status.
- De-registrate of the victim by sending a modified message to the Sip server.
- Generate a race-condition on the Server, in which the attacker repeatedly sends REGISTER requests in a shorter timeframe. In order to override the legitimate user's registration request.

Having deleted the registration of the victim, the attacker replaces it with his registration information. In this way we obtained an identity theft and if the attacker receives a call, it is possible for him to obtain sensible information about the victim.

The following figure is shown how the Sip registration hijacking attack works .

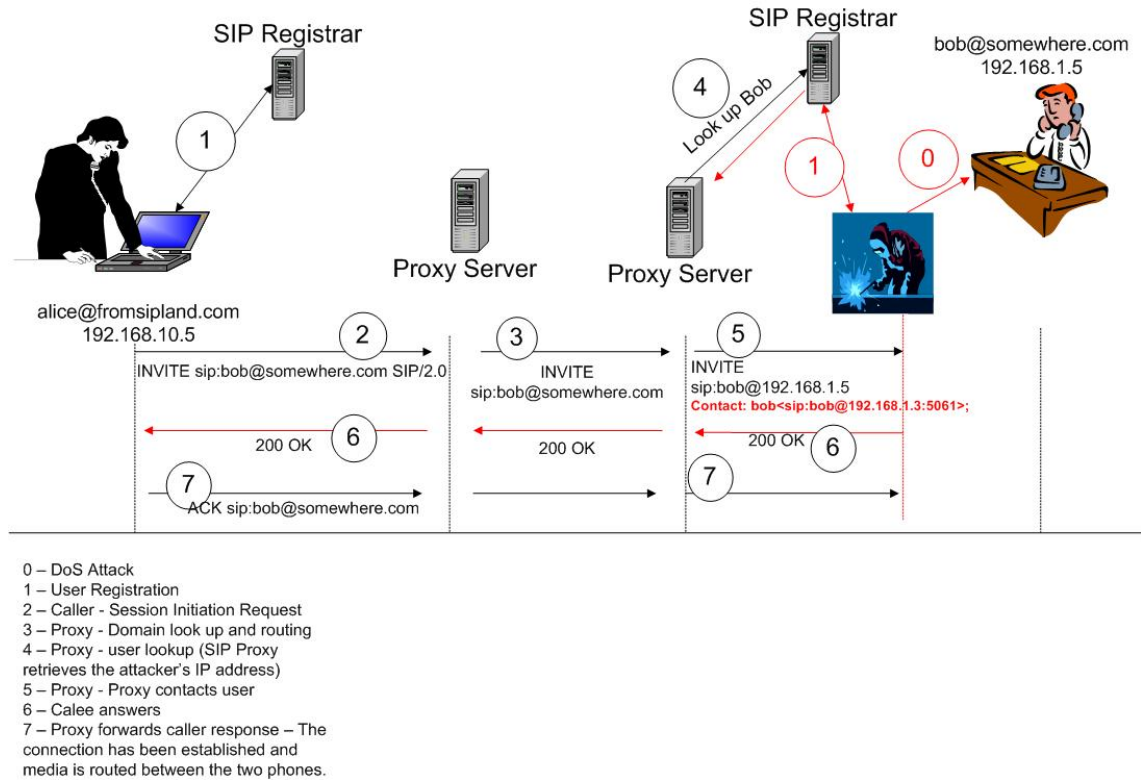


Figure 5.6: Sip registration hijacking

The following tables illustrate the performance of the test during an unsecure execution. To calculate the MOS we used [22]. The value was calculated over a total of 1000 calls generated.

No VPN No traffic G.711alaw max 70 pair	MOS	Total Calls	Success Call	Failed Calls	end-to-end delay
	3,14	1000	863	137	30ms

No VPN Yes traffic G.711alaw max 70 pair	MOS	Total Calls	Success Call	Failed Calls	end-to-end delay
	2,25	1000	729	271	30ms

Figure 5.7: Real test - G.711Alaw - NO VPN

No VPN No traffic G.729	MOS	Total Calls	Success Call	Failed Calls	end-to-end delay
max 70 pair	4,10	1000	999	1	30ms

No VPN Yes Traffic G.729	MOS	Total Calls	Success Call	Failed Calls	end-to-end delay
max 70 pair	3,47	1000	953	47	30ms

Figure 5.8: Real test - G.729 - NO VPN

As we it may be pointed out the results differs from the simulation. With the G.711A law codec, even without traffic loading the quality was low. Probably due to the number of concurrent calls, which is too much for the overall available bandwidth. The results with the G.729 codec got to a result similar to the simulation, owing to its use of less bandwidth.

5.2.4 Secure test

To realize the secure tests we used OpenVpn. This choice was made in order to protect both the Sip signaling and the Rtp flow. Openvpn is a solution that allows to realize a Virtual Private Network based on the TLS protocol. As proven in [10] and [11], if the IPv6 protocol is used to transmit the data, there is a native support for encryption and integrity (IpSec). The encryption is achieved with the ESP (Encapsulated Security Payload) while the integrity control is made by the AH (Authentication Header). Further analyses in [10] and [11] underline that use of IpSec rises however some problems. The bandwidth consumption is very high, due to a lot of auxiliary traffic of the protocol. This network overload makes the infrastructure very sensible to congestion issues. The way this problem becomes harder in case of source-destination hop increasing is also proven.

5.2.5 Result forecast

Our target was to discover how to avoid some attacks and how to mitigate the impact of some other attacks. The TLS protocol used to build the VPN adopts a cryptographic technology with asymmetric keys. This kind of technology would be strong enough to resist at least 100 years. The strength of a cryptographic algorithm with asymmetrical keys is first of all represented by the way of keeping the keys, and consequently in the real goodness of the algorithm. So the main task of the computer security is in the hands of the Man. [9] During the experiment we focused our attention on performance losses owing to the VPN.

5.2.6 Test result

The strength of a Voip platform with Vpn against attacks like Sip registration hijacking or Man in the middle is without doubts very high. The complete encryption of the data, (Signaling and Rtp flow), is a point of defense against malicious users. The Registration hijacking attack is not performable anymore, because the Sip signaling now is completely encrypted. The same is for attack with the Man in the middle technique, of course is possible to sniff the traffic, but in this case the data is completely encrypted. The standard of encryption of OpenVpn are strong enough to resist at least 100 years. A pretty good delay of time to keep a secret. Only if an attacker is somehow able to compromise the certification authority (CA) machine, he can build a valid certificate to penetrate in the communication. Compromising the CA machine has to be, by definition, a really hard job. It is very likely to put the CA machine outside the network.

During the secure tests, without Iperf, the results were similar to the unsecure solution. So the protection mechanism did not considerably influence the performances. In in this way the resistance against DoS attacks is not so different than the unsecure tests.

On the contrary, while using the Iperf we detected a 10% performance loss of the with the G.711Alaw codec and a 5% performance loss with the G.729 codec. With this results it is easy to point out that the strength of the infrastructure against DoS attacks is lower. A possible way to mitigate this issue is not to use standard application ports, so as to try confusing the attacker about the service that is running on that machine. Another solution of this problem could be by configuring trusted network domains. The adoption of too many security measures, however can lead to expensive solutions. Probably this kind of solution could be considered by a government institution, which needs the maximum level of protection.

To achieve a MOS >3 with the G.729 codec we had to switch down the traffic loading to 2,75Mbit. While if we wanted to obtain the same with the G.711Alaw it would be necessary to have more bandwidth or even to reduce the number of concurrent calls.

Yes VPN No Traffic G.711alaw	MOS	Total Calls	Success Call	Failed Calls	end-to-end delay
max 70 pair	3,21	946	833	113	30ms

Yes VPN Yes Traffic G.711alaw	MOS	Total Calls	Success Call	Failed Calls	end-to-end delay
max 70 pair	1,99	1000	659	341	30ms

Figure 5.9: Real test - G.711Alaw - VPN

Yes VPN No Traffic G.729	MOS	Total Calls	Success Call	Failed Calls	end-to-end delay
max 70 pair	4,10	1000	999	1	30ms

Yes VPN Yes Traffic G.729	MOS	Total Calls	Success Call	Failed Calls	end-to-end delay
max 70 pair	2,84	1000	910	90	30ms

Figure 5.10: Real test - G.729 - VPN

5.3 Conclusions

The real-time applications are very sensitive, and each variation occurring on the network can modify and influence the final result of a real-time data transmission, such as a Voip Call. Our tests tried to show up the security weakness of the voip technology, the proposed solution is only one of the several possible. Everytime is necessary to introduce a security measure it has to be considered under different aspects: at least economical, technical and human. Usually the deployment of this measure has to be a compromise between these aspects. So, the choice adopted by us had been a large compromise, between the available environment the low amount of funds and the easy-deployment of the infrastructure. Even if we had got a lot of constraints to build-up our security voip environment, we demonstrate that it is possible to protect about 50 concurrent calls. This with less bandwidth requirements than a Voip Service Provider but keeping a sufficient level of performance, at least with the G.729 audio codec. As read in

[12] the authors will continue the evolution of the security problems relating to the data transmission.

Italian resume

A.1 Panoramica sul VoIP

VOIP è l'acronimo di Voice Over IP, o in altri termini telefonia attraverso reti dati. Tutto questo se è presente una connessione alla rete dati con delle buone prestazioni di Upload/Download. Le ragioni principali per cui l'utilizzo del VoIP sta crescendo notevolmente, si possono attribuire a costi più bassi e alla possibilità di impiegare funzionalità avanzate rispetto alla rete PSTN.

Il suo utilizzo infatti consente una tariffazione delle chiamate indipendente dalla durata e se il destinatario è un computer, indipendente anche dal locazione dell'utente. Ultimamente il numero di provider di servizi orientati al VoIP è in crescita. Questo ha facilitato gli utenti nell'avvicinarsi a questa tecnologia. Sono davvero poche le persone che oramai non conoscono *Skype*. I servizi VoIP sono anche in espansione come supporto ad applicazioni multimediali come video conferenze, streaming video, gaming online.

A.2 Protocolli VoIP

Il numero di protocolli VoIP in uso è aumentato notevolmente nel corso degli ultimi dieci anni. A partire dall'H.323, inizialmente implementato per il trasporto di media all'interno di reti LAN e la cui prima versione è stata rilasciata nel 1996. Successivamente sono stati sviluppati svariati protocolli in grado di scambiare dati/voce tra due o più utenti. L'H.323, implementato dalla ITU Telecommunication Standardization Sector, si può definire come un insieme di protocolli che cooperano per supportare sessioni audio-video.

L'**H.323** è stato il primo ad essere sviluppato, ma ormai è stato soppiantato dal **SIP**. Un protocollo text-based simile ad HTTP e XML, adattabile ad una larga scala di applicazioni. SIP è quindi diventato la scelta della comunità internet ed è stato poi reso standard dall'IETF. Il Session initialization protocol

viene utilizzato per creare, modificare e terminare sessioni tra uno o più partecipanti, è stato originariamente sviluppato da Henning Schulzrinne e Mark Handley nel 1996. L'ultima versione delle specifiche è definita RFC 3261 dall'IETF SIP Working Group. Un altro protocollo di signaling standardizzato è l'**H.248**, conosciuto come Megaco e standard internazionale per il media gateway control. Il suo uso primario è quello di separare il controllo logico delle chiamate dall'elaborazione dei media all'interno di un gateway. Il device che gestisce le funzioni di call control è identificato come Media Gateway Controller mentre il device che gestisce i media è il Media Gateway. Oltre a questi protocolli citati ce ne sono altri non standardizzati, ma utilizzati largamente in rete grazie ad alcuni applicativi che hanno trovato larga diffusione tra gli utenti, (Google Talk, Skype, etc.).

A.3 Le problematiche

La maggior parte dei service provider riconosce nel VoIP la direzione futura da seguire. Per far sì che questo si avveri devono essere garantiti alcuni requisiti fondamentali, per rendere questa tecnologia totalmente equivalente con la rete PSTN:

- Affidabilità
- Sicurezza
- Quality of Service

A.3.1 Affidabilità

Di sicuro il confronto con la rete PSTN in questo momento risulta molto difficile. Dato che quest'ultima ha un livello tale da garantire solo 5 minuti all'anno di inattività. E inoltre è in grado di gestire milioni di chiamate simultanee.

Il divario presente in questo momento tra l'architettura PSTN e quella Voip può essere assottigliato introducendo hardware e connessioni di rete ridondanti. Sarebbe anche necessario aumentare la capacità di ricovero dagli errori da parte dei vari componenti dell'infrastruttura Voip.

A.3.2 Sicurezza

Riguardo alla sicurezza la rete PSTN si è rivelata molto più resistente ad attacchi di sicurezza, di fatti sono pochi i casi di vulnerabilità segnalati di questa architettura. Non considerando che spesso le centraline a bordo strada sono aperte. Un'infrastruttura VoIP invece risulta sicuramente più vulnerabile a determinati problemi. Questo perché si appoggia ad un'architettura, quella IP, che soffre già dalla sua nascita di diversi problemi di sicurezza. Se si considerano inoltre i vincoli di QoS cui questo tipo di architettura deve sottostare, la mole di ostacoli relativi alla sicurezza che devono essere superati per

fornire un servizio comparabile a quello PSTN aumenta. Inoltre se il Voip dovesse rimpiazzare la telefonia tradizionale dovrebbe essere possibile anche effettuare chiamate d'emergenza. Cosa abbastanza difficile, considerando che questo tipo di servizio ha un numero unico sul territorio ma i punti di risposta sono suddivisi in base alla zona dove viene digitato.

A.3.3 Quality of Service

Il Quality of service dipende da tre elementi fondamentali, il ritardo di trasmissione, la varianza con la quale i pacchetti vengono ricevuti e la perdita di pacchetti. Il protocollo IP non dispone di alcun meccanismo in grado di garantire l'arrivo dei pacchetti dati nell'ordine in cui sono stati trasmessi e non offre nessuna garanzia generale relativa alla qualità del servizio.

Le attuali applicazioni nel mondo reale della telefonia VoIP si trovano a dover affrontare problematiche legate alla latenza di trasmissione e all'integrità dei dati per prevenire perdite e danneggiamenti delle informazioni. Risulta quindi indispensabile realizzare una soluzione affidabile e che offra una qualità nella trasmissione/ricezione delle informazioni di livello superiore. Le tecnologie adottabili sono svariate, tra le più conosciute citiamo Diffserv, RSVP e MPLS.

A.4 Il protocollo SIP

SIP è l'acronimo di Session Initialization Protocol, un protocollo che è possibile posizionare a livello applicazione nel modello OSI. Le funzionalità da esso offerte riguardano la creazione, modifica e chiusura di sessioni tra due o più partecipanti. SIP viene sviluppato dalla Internet Engineering Task Force(IETF). La prima proposta di standardizzazione è stata prodotta nel 1999 come RFC 2543, ma nel 2002 lo standard è stato ulteriormente corretto come RFC 3261. La ragione principale che rende necessario il suo impiego, si può ricercare nei programmi che permettono lo scambio di messaggi, voce e media.

Infatti gli utenti che li utilizzano possono accedere ai loro servizi preferiti da diverse zone o computers e possono essere in possesso di diversi account. Le informazioni passano attraverso un server proxy, mediante il quale avviene la registrazione e l'indirizzamento dei messaggi o anche l'invito di altri utenti ad una sessione. La funzionalità più importante è quella di generare e controllare di una sessione tra due parti. Senza quest'ultima capacità la comunicazione all'interno di una rete eterogenea ed estesa come internet sarebbe difficilmente realizzabile. Le sessioni possono riguardare telefonate VoIP, distribuzione di audio/video o multiplayer gaming.

SIP non provvede alla gestione di tutte le funzioni necessarie allo scambio dei dati, però semplifica questo compito, attraverso il meccanismo delle sessioni, la comunicazione tra gli endpoint. Come HTTP e SMTP, SIP usa un Universal Resources Identifier (URIs) per identificare gli utenti, nella stessa maniera in cui un URL è utilizzato per i siti Web. Solitamente l'URI incorpora un numero di telefono o un nome Per esempio (foobar@security.dico.unimi.it) è sicuramente più semplice da memorizzare rispetto a (foobar@159.158.146.151:5060). Gli usernames e i numeri di telefono devono essere univoci

poiché identificano quale account appartiene ad una specifica persona. Questo consente successivamente di comunicare messaggi o inoltrare chiamate verso la parte desiderata. Tutti gli usernames sono salvati sul server, quest'ultimo può quindi determinare quando un particolare utente è raggiungibile oppure no. Generalmente gli URI iniziano con "SIP:", oppure con "SIPS:" che indica la trasmissione delle informazioni attraverso un canale sicuro (ad esempio usando il protocollo TLS).

A.5 SIP e il livello applicazione

Come accennato nel paragrafo precedente, il protocollo SIP invita un utente a partecipare a una sessione e può anche inserirlo all'interno di una già esistente, ad esempio per una conferenza. Il livello applicazione viene utilizzato per identificare i partner della comunicazione e per facilitarne l'autenticazione. Rende inoltre possibile l'interazione con protocolli dei livelli più bassi. Nel caso del SIP la sua funzione di creazione, mantenimento e chiusura delle sessioni viene sfruttata dai software, per la trasmissione dei dati attraverso la rete tra due o più peer.

A.6 Caratteristiche e funzioni del protocollo SIP

Quando il protocollo è stato sviluppato, gli elementi specifici che doveva supportare erano cinque :

- User location, ovvero identificazione e ricerca degli endpoint partecipanti ad una sessione;
- User availability, il partecipante alla conversazione ha opportunità di indicare il proprio stato, (Presente, Occupato, Torno Subito,etc..) e decidere così se vuole iniziare o meno la comunicazione;
- User capabilities, media che verranno usati quando si stabilisce la comunicazione e relativi parametri;
- Session setup, si inizializza una sessione tra 2 o più endpoint;
- Session management, modifica dei parametri relativi alla sessione in corso, trasferimento dei dati, richiamo di servizi specifici e terminazione di essa.

A.7 Architettura SIP

Le componenti fondamentali usate dal Session Initiation Protocol sono le seguenti:

- User agents, ovvero gli endpoint di una chiamata;
- SIP servers, ovvero i computer sulla rete che assolvono le richieste provenienti dai clients e mandano indietro le risposte previste dal protocollo.

A.7.1 User Agents

Per User Agents (UA) si intendono sia il computer/telefono utilizzato per effettuare la chiamata, che il computer/telefono che viene contattato. L'User Agent si può a sua volta suddividere in due ulteriori componenti: un client ed un server. Un User Agent può essere identificato come *Client* o *Server* a seconda che invii o risponda alle richieste di un altro UA. Per instaurare una connessione un UA dovrà invitarne un'altro ad una sessione. Durante questo periodo l'UAC userà il SIP per inoltrare le richieste all'UAS, il quale a sua volta invierà nello stesso modo le risposte. Solitamente come user agents vengono utilizzati softphone, PDA, telefoni USB o normali apparecchi telefonici ma integrati con la VoIP.

A.7.2 Sip Server

Il Sip Server è usato per la traduzione degli URI in IP address, in modo che il messaggio inviato da un User Agent ad un altro giunga senza problemi. Quando un UA si registra con il SIP Server, fornisce il proprio username e l'indirizzo IP, permettendo al server di stabilire la posizione del client nella rete. Per mezzo di tale procedura è possibile anche stabilire quando l'UA è online, consentendo ad altri utenti di invitarlo a partecipare ad una sessione. Un User Agent non è a conoscenza dell'indirizzo IP degli altri UA che desidera contattare, deve perciò interpellare il SIP Server per invitare un utente all'interno della sessione. Il server, una volta interpellato controlla che l'utente sia effettivamente online e successivamente compara l'username all'indirizzo IP per determinarne la posizione. Se l'utente non fosse parte di quel dominio e utilizzasse quindi un altro SIP server, la richiesta verrebbe instradata al server di competenza. Il server SIP è in grado di agire in maniera differente, in base al contesto in cui si trova. Vediamone i diversi ruoli:

- Registrar Server
- Proxy Server
- Redirect Server

Il **Registrar Server** è usato per conoscere l'ubicazione di un UA che si è loggato all'interno della rete. Esso ottiene l'indirizzo IP dell'utente e lo associa con il suo username. Per mezzo di questa operazione viene creata una traccia di tutti gli utenti loggati e della loro posizione. Strettamente legato a questa procedura è il Location Service, usato per creare e mantenere un DB nel quale si tiene traccia degli utenti registrati. Quando un UA si registra ad un Registrar server, il primo invia una REGISTER request. Se il Registrar accetta la richiesta, otterrà il SIP-address e l'indirizzo IP dell'endpoint, aggiungendolo al location service del suo dominio. Per mezzo di questo database si riesce a mantenere un catalogo aggiornato di chi è online e dove si trova.

Il **Proxy Server** è un computer dedicato all'inoltro di richieste per conto di altre macchine. Se il server riceve una request da un client, può inoltrarla a un altro SIP server all'interno della rete o ad un client da lui controllato. Lavorando come proxy possono essere implementate funzioni quali il controllo dell'accesso alla rete, la sicurezza, l'autenticazione e l'autorizzazione.

Infine i **Redirect Server**, sono utilizzati per reindirizzare i client verso gli UA che desiderano contattare. Se uno di loro effettua una request, il Redirect Server può rispondere con l'IP address dell'UA che si vuole contattare. Il Redirect Server svolge quindi un ruolo differente rispetto al Server Proxy, in quanto comunica all'UA come contattare di persona l'altro client.. Una particolarità del Server Redirect, è la possibilità di eseguire la fork di una chiamata, dividendola su più punti. Per esempio se un utente è registrato con più dispositivi, la richiesta viene divisa su tutti, facendoli squillare allo stesso istante. Il primo a rispondere, verrà associato alla sessione. Come specificato dal RFC 3261, i differenti tipi di SIP server possono essere implementati su macchine separate o come parte di una singola applicazione che risiede su una singola macchina. Un'altra caratteristica dei server utilizzati dal protocollo SIP è la possibilità di operare in due modi, stateful o stateless. Il primo tiene conto di tutte le request e le response che invia e riceve registrandole sotto forma di log. Mentre il secondo non ne tiene traccia, cancellandole una volta processate. Solitamente un server operante in maniera stateful viene utilizzato dove risiedono gli endpoint, mentre i server stateless fanno parte della backbone, pertanto ricevono una mole di richieste tale che sarebbe difficile tenerne traccia.

A.8 SIP Requests and Responses

Con il protocollo SIP vengono scambiati messaggi tra client e server, User Agent Client e User Agent Server. I possibili comandi di signaling usati sono:

- REGISTER: un UA passa nello stato online e registra il suo SIP URI e l'IP address con un Registrar server;
- INVITE: usato per invitare un altro UA alla comunicazione e successivamente stabilire una sessione SIP;
- ACK: accettazione di una sessione e conferma dei media stabiliti;
- OPTIONS: usato per ottenere informazioni sulle capacità di un altro UA, in modo da poter stabilire la sessione tra i due endpoint;
- SUBSCRIBE: usato per richiedere aggiornamenti riguardo lo stato di un altro user agent (online, busy, offline etc...);
- NOTIFY: usato per mandare aggiornamenti relativi allo stato dell'UA;

- CANCEL: usato per cancellare una richiesta pendente senza terminare la sessione;
- BYE: usato per terminare la sessione in modo corretto.

Quando viene inoltrata una richiesta SIP le risposte che si possono ottenere sono raggruppate in sei categorie differenti, sotto forma di un codice decimale a tre cifre, il cui primo numero corrisponde alla categoria di appartenenza. I prefissi sono i seguenti:

- Informational (1xx) la richiesta è stata ricevuta ed sta per essere processata;
- Success (2xx) la richiesta ha ricevuto l'ack ed è stata accettata;
- Redirection (3xx) la richiesta non può essere completata e sono richiesti step addizionali;
- Client error (4xx) la richiesta conteneva errori, quindi il server non può processarla;
- Server error (5xx) la richiesta è stata ricevuta, ma il server non può processarla. Errori di questo tipo si riferiscono al server stesso e indicano che un altro server potrebbe eseguirla;
- Global failure (6xx) la richiesta è stata ricevuta ma il server non è in grado di processarla.

Ecco alcuni esempi delle responses più comuni che si possono ottenere.

CODICE	DESCRIZIONE
100	Informational : Trying
180	Informational : Ringing
200	Success OK
404	Client Error : not found
500	Server Error : internal server error
600	Global Failures : Busy everywhere

A.9 Protocolli associati

Il protocollo SIP utilizza diversi altri protocolli per il passaggio di dati tra server, partecipanti e devices di rete. Alcuni di essi supportano il trasporto di pacchetti contenenti informazioni SIP attraverso la rete (UDP), o il crypting degli stessi per assicurare sicurezza alle informazioni trasmesse (TLS). Altri vengono invece utilizzati per la trasmissione di media o per il supporto dei servizi forniti dai programmi di comunicazione. Questi protocolli includono SDP utilizzato dagli endpoint per accordarsi sui codec audio o video da utilizzare. RTP, usato per trasportare effettivamente i dati e RTCP per controllare la corretta consegna dei dati audio/video durante la sessione.

A.9.1 User Datagram Protocol (UDP)

Questo protocollo viene utilizzato per trasportare datagrammi attraverso una rete IP. Offre un servizio di tipo connection-less, il che non assicura la corretta ricezione dei pacchetti da parte del destinatario. Quindi è compito dell'applicazione assicurarsi che i pacchetti arrivino nell'ordine corretto e senza errori. La mancanza di meccanismi di controllo a livello trasporto consente di avere tempi inferiori per la consegna dei pacchetti. Proprio per questo motivo UDP viene utilizzato per la trasmissione del traffico real-time e per l'invio di messaggi di piccole dimensioni. Nel caso del VoIP, video streaming o multiplayer gaming, una piccola perdita di informazioni implica solamente un piccolo inconveniente che non danneggia le performance totali. Il requisito più importante per questo tipo di comunicazioni è senza dubbio la velocità di trasmissione tra gli endpoint.

A.9.2 Transport Layer Security (TLS)

Si tratta di un protocollo che può essere utilizzato per fornire sicurezza tra applicazioni comunicanti attraverso la rete IP. Il TLS cripta i pacchetti per assicurare la privacy, in modo che esterni non possano intercettare e/o modificare i messaggi inviati. Usando il TLS viene stabilita una connessione sicura autenticando il client ed il server (UAC e UAS) e criptando la connessione tra di loro. Il Transport Layer Security è definito all'interno dell' RFC 224. Viene descritto come un protocollo a più livelli:

- TLS Handshake Protocol
- TLS Record Protocol

Il TLS Handshake Protocol è utilizzato per autenticare i partecipanti alla comunicazione e negoziare l'algoritmo di criptazione. Ciò rende possibile al client e al server, di accordarsi prima che vengano inviati i dati effettivi. Una volta eseguita questa operazione viene instaurato un canale sicuro tra di loro. Dopo l'utilizzo del TLS Handshake Protocol, il TLS Record Protocol assicura che i dati scambiati tra le parti non siano stati alterati nel tragitto.

A.9.3 Session Description Protocol (SDP)

Utilizzato per inviare informazioni descrittive necessarie quando vengono inviati dati multimediali. Una volta creata la sessione, SDP stabilisce quali formati audio/video utilizzare e altre informazioni necessarie per organizzare il trasferimento di questi dati. Si tratta di un protocollo text-based, le informazioni che gestisce contengono dati indicanti il nome della sessione, tempo di attività della stessa, la descrizione dei media scambiati e altre informazioni sugli endpoint.

A.9.4 Real-Time Transport Protocol (RTP)

Si occupa di trasportare i dati real-time. Dato che l'RTP si avvale di UDP, che non garantisce affidabilità nella ricezione dei dati, è lui stesso che si deve occupare di questo aspetto tra i due UA. Per quest'ultimo

aspetto viene utilizzato il Real-time Control Protocol che consente di monitorare la consegna dei dati scambiati tra i partecipanti. Grazie a RTCP è possibile verificare se ci sono perdite di pacchetti e di compensare eventuali ritardi introdotti dalla rete.

A.9.5 SRtp

I dati che viaggiano attraverso una rete Ip possono essere facilmente intercettati e alterati. Per questo motivo è stato realizzato SRtp, un protocollo che consente di proteggere il traffico Rtp da occhi indiscreti. Definito nel documento RFC 3711, Srtp è in grado di fornire le seguenti funzionalità:

- Confidenzialità
- Autenticità del messaggio
- Protezione da attacchi di tipo replay

Confidenzialità

Questo aspetto viene garantito attraverso l'utilizzo dell'algoritmo AES, che può essere configurato in modalità F8 o in modalità Segmented Integer Counter Mode.

Autenticità del messaggio

Grazie all'impiego dell'algoritmo HMAC-SHA1 è possibile attestare l'autenticità della fonte di ogni pacchetto. In questo modo se vengono inviati pacchetti creati ad-hoc per cercare di modificare la conversazione tra 2 endpoint, questi verranno automaticamente scartati poichè il codice HMAC-SHA1 sarà differente dagli originali.

Key management

Il protocollo più famoso utilizzato per gestire le chiavi è MIKEY, definito nell'RFC 3830, questo protocollo funziona in tre diversi modi:

- Pre-Shared Secret (PSK)-La chiave pre-condivisa viene utilizzata per generare le sotto-chiavi per garantire l'integrità e la confidenzialità.
- Public Key Encryption(PKE)-Il chiamante genera una chiave casuale per la criptazione, che viene successivamente inviata al chiamato. Questo messaggio viene criptato utilizzando la chiave pubblica del destinatario. Con questa configurazione è necessaria un'infrastruttura pubblica di gestione delle chiavi (PKI).

- Diffie-Hellman(DH)-Questa configurazione è opzionale, ma è l'unica che consente la "Perfect Forward Secrecy". Questo metodo può essere utilizzato per negoziare le chiavi tra due endpoint ma richiede comunque l'esistenza di una PKI.

Per i dettagli di funzionamento di Srtp si rimanda alla sezione 2.4.2

A.9.6 ZRtp

Questo protocollo è un'alternativa al protocollo MIKEY per la gestione delle chiavi. La tecnologia crittografica che utilizza è basata sull'algoritmo Diffie-Hellman effimero e non prevede l'utilizzo di una PKI. Il principio di funzionamento di ZRtp è end-to-end, è sufficiente installare il plug-in Zfone per il proprio Softphone SIP e in maniera automatica tutto il traffico RTP generato verrà protetto da occhi indiscreti. [Vedi figura 2.5] Questo protocollo è stato realizzato da Phil Zimmermann, il creatore di PGP.[17]

A.10 Possibili attacchi SIP

A.10.1 DoS Denial-of-service Attacks

Possono essere utilizzati per sovraccaricare o bloccare un sistema vittima. In un attacco di flooding di tipo UDP, verranno inviati pacchetti verso porte attive di un sistema. Lo scopo di questo attacco è sovraccaricare la macchina bersaglio, cercando di rendere l'host di destinazione irraggiungibile da parte dei client. Per ridurre gli inconvenienti di questo tipo di attacco vi sono svariate soluzioni. Possono essere adottati server proxy o meccanismi di firewalling, nonché tecniche di traffic shaping all'interno della rete per prevenire l'arrivo di traffico indesiderato.

A.10.2 Registration hijacking

Questa situazione si verifica quando un attacker impersonifica un UA valido e si registra su un SIP Registrar, rimpiazzando l'indirizzo del legittimo utente con il proprio. Il nuovo UA potrà contattare tutti gli altri peer registrati sul server e ad esso verranno indirizzate tutte le chiamate destinate alla vittima.

La possibilità di realizzare questo attacco è dovuta al fatto che i messaggi Sip viaggiano in chiaro sulla rete, cosa che facilita lo spoofing delle request. Se sono state abilitate le impostazioni di autenticazione per mezzo di username e password è possibile effettuare attacchi offline con la suite SiVus[21]. Qualora l'attacco andasse a buon fine gli effetti sarebbero la perdita delle chiamate da parte della vittima. Oltre a questo, il falso UA può aumentare la mole di informazioni possedute, immagazzinando autenticazioni o altre informazioni chiave relative al signaling.

A.10.3 Man-In-The-Middle (MITM)

La tipologia di attacco che va sotto il nome di man-in-the-middle consiste nel dirottare il traffico generato durante la comunicazione tra due host verso un terzo (attaccante). Durante l'attacco è necessario far credere ad entrambi gli end-point della comunicazione che l'host attaccante è in realtà il loro interlocutore legittimo.

L'host attaccante riceve quindi tutto il traffico generato dagli host vittima e si preoccupa di inoltrare correttamente il traffico verso l'effettiva destinazione dei pacchetti ricevuti.

Considerando questa situazione all'interno di una rete VoIP, l'utente che effettua l'attacco è in grado di conoscere tutte i dialoghi avvenuti durante la conversazione, decodificandoli per mezzo di appositi plugin. Sniffando il traffico con *Wireshark* è infatti possibile riascoltare le tracce audio, solo se codificate usando il PCMU o G.711. Qualora l'RTP trasmesso utilizzasse codifiche differenti è possibile usare *rtpools* [24], per ascoltare i file audio. Per mezzo di questo applicativo il traffico RTP viene rediretto verso un telefono VoIP che supporta il codec desiderato.

A.11 Sicurezza voip in pratica

A.11.1 Il nostro esperimento

Dopo aver attentamente esaminato i problemi di sicurezza che possono colpire una piattaforma Voip, abbiamo voluto realizzarne una. Per scoprire come le misure di sicurezza possono influire sulla qualità del servizio. Il primo problema affrontato è stato quello di poter avere dei risultati comparabili con i modelli qualitativi espressi nelle raccomandazioni ITU-T. Questi modelli cercano di chiarire come poter valutare le prestazioni del traffico voce, i parametri più famosi che si possono citare per la valutazione del traffico voce sono sicuramente il MOS (Mean Opinion Score) e R-Factor. Il primo è una scala di valutazione da 1 a 5 dove 5 rappresenta la qualità migliore possibile e il giudizio è soggettivo, basato sull'opinione degli utenti. Mentre R-Factor è uno scalare basato su E-Model, un modello computazionale per la valutazione delle prestazioni. Il problema è che la formula per calcolare l'R-Factor è particolarmente complessa e articolata. Quindi c'è stato bisogno di una suite software che potesse aiutarci per poter realizzare questa misurazione. Il software che abbiamo utilizzato è *Ixchariot*, un potente simulatore di traffico per la valutazione delle prestazioni sulle reti. [Vedi Sez: 4.3.1] Le specifiche del test simulato sono le seguenti:

- La rete aveva una banda di 10 Mbit a Milano, Italia, e 6Mbit a Ostrava, Repubblica Ceca.
- Sono stati effettuati test con i codec audio G.711a e G.729[25][26]
- I test sono stati effettuati sia senza sovraccaricare la rete sia sovraccaricandola, utilizzando *Iperf*, con 4 Mbit di traffico udp generico.

- I test sono stati effettuati in modalità sicura e in modalità non sicura, cifrando le informazioni con OpenVpn.
- Per ridurre al minimo eventuali errori di calcolo del simulatore tutti i test sono stati effettuati 5 volte.
- End-to-end delay 30msec sulla rete di ricerca Gèant2, alla quale sono collegati i due atenei.

I risultati ottenuti nei test simulati si trovano in [12] Successivamente alla simulazione abbiamo realizzato dei test reali, per poterli comparare con le simulazioni e misurare il grado di affidabilità delle stesse. Nell'esecuzione dei test reali sono stati provati degli attacchi, per poter valutare l'efficacia della nostra soluzione di sicurezza.

A.11.2 Le ragioni del nostro esperimento

Abbiamo deciso di realizzare questi test per capire come le misure di sicurezza adottate possano prevenire o mitigare determinati rischi. I test sono inoltre stati utili per quantificare il carico delle impostazioni di sicurezza sulle prestazioni totali. Come server sip è stato scelto OpenSer una soluzione adottata da molti Service Provider, adatta a sostenere grandi carichi di traffico. Per quanto riguarda le impostazioni di sicurezza, si è cercato di mantenere una similitudine tra i test simulati e quelli reali. Questo è stato uno dei primi motivi che hanno condizionato la scelta del sistema OpenVpn. Un altro motivo a cui si può ricondurre questa scelta è legato alla possibilità di ottenere un livello di protezione generale più elevato, poiché sia il traffico Rtp che il signaling Sip vengono protetti attraverso la tecnologia SSL/TLS.

A.11.3 La modalità operativa dell'esperimento

Sono stati configurati due Pc con GNU/Linux Debian. Il primo presso il Laboratorio di Sicurezza e Reti all'interno del Dipartimento di Informatica e comunicazione dell'Università di Milano e il secondo presso il Voip laboratory, presso la VSB-Technicka Univerzita di Ostrava in Repubblica Ceca. Successivamente abbiamo installato il software client di Ixchariot su entrambi i Pc. La console di Ixchariot è stata invece installata su un laptop con Windows Xp, questo applicativo consente di preparare e lanciare i test. La modalità di esecuzione scelta per i test è stata quella batch. È così stato possibile ridurre al minimo l'influenza dei dati di controllo che vengono trasmessi alla console durante il test, perché con la modalità scelta tutti i risultati vengono inviati alla console solamente al termine del test.

Per quanto riguarda i test reali, oltre al sopraccitato OpenSer, è stato necessario trovare un software che ci consentisse di generare traffico Sip/Rtp, la soluzione scelta è Sipp, un software che, se configurato correttamente, consente di simulare sia il Sip signaling che il traffico Rtp.

Come precedentemente detto le impostazioni di sicurezza sono state le stesse sia per la simulazione che per i test reali, grazie all'utilizzo di OpenVpn.

A.12 I tools utilizzati

A.12.1 IxChariot

Questo software, prodotto da Ixia viene impiegato per calcolare sperimentalmente le prestazioni di un sistema, simulando delle reali condizioni di utilizzo. I test vengono preparati utilizzando la IxChariot console, per Win32 e due IxChariot endpoints, questi ultimi installabili su diversi sistemi operativi. La console consente di selezionare e configurare molte tipologie di test, tra le quali citiamo: IPv4 con o senza supporto per QoS e anche IPv6. Al termine di ogni test, è possibile visualizzare ogni tipo di parametro relativo alle prestazioni: throughput, jitter, MOS e R-Factor. Il miglior modo di eseguire il test è utilizzando la modalità batch. In questo modo i risultati del test vengono inviati alla console solamente al termine del test. Grazie a questo è possibile evitare influenze dei dati relativi ai risultati durante l'esecuzione del test.

A.12.2 OpenVpn

Questo software OpenSource consente di creare Virtual Private Networks utilizzando il protocollo TLS, illustrato nel secondo capitolo. Grazie a questo software è possibile ottenere una VPN che utilizza le più potenti suite crittografiche conosciute. Per ottenere migliori performance vengono utilizzati algoritmi di crittografia asimmetrici solamente per lo scambio delle chiavi. Mentre per la criptazione dei messaggi vengono utilizzati algoritmi di crittografia simmetrici, poichè questi ultimi sono più veloci.

A.12.3 OpenSer

OpenSer è un Sip server OpenSource. Può essere utilizzato su sistemi con risorse limitate o su server di fascia alta, senza problemi. è in grado di gestire migliaia di chiamate al secondo. Questo software è scritto in linguaggio C per sistemi Unix/Linux con ottimizzazioni specifiche a seconda dell'architettura, in modo da migliorarne le prestazioni. OpenSer può essere configurato come: registrar server, location server, proxy server, redirect server e anche come gateway per SMS/XMPP. L'ultima versione supporta anche il protocollo TLS per l'autenticazione degli utenti, in questo modo il signaling SIP viene protetto.

OpenSer e il database MySQL

Per la registrazione degli utenti OpenSer ha la possibilità di interagire con il DBMS MySQL. All'interno dell'archivio di installazione è presente il file mysqldb.sh, è necessario eseguire questo file per creare un nuove database per OpenSer.

La configurazione di OpenSer richiede una buona conoscenza del protocollo Sip, di seguito spiegheremo la nostra configurazione di test.

Il file di configurazione: opeser.cfg

Il file di configurazione inizia con le seguenti righe:

```
#####OpenSer CONFIGURATION FILE#####  
  
# ----- global configuration parameters -----  
debug = 3  
fork = no  
log_stderr= yes
```

Queste tre opzioni servono a configurare rispettivamente il livello di debugging, il divieto di effettuare una fork del processo e in caso di errore la possibilità di stampare i messaggi sullo standard error. Questi messaggi possono di solito riguardare direttive invalide di configurazione o problemi con il caricamento di un modulo.

```
listen = 192.168.1.158  
alias = 192.168.1.158  
port = 8888
```

Queste istruzioni vengono utilizzate per specificare su quale interfaccia e con quale porta il servizio OpenSer sarà in funzione.

```
children = 4  
dns = no  
rev_dns = no
```

La direttiva `children` è utile per configurare il numero di threads che l'applicazione può inizializzare per la gestione del traffico SIP. Il numero di `children` dev'essere attentamente valutato in base alle performances del sistema dove OpenSer è installato e in base al carico di traffico che dovrà sopportare. Di solito OpenSer non necessita di interagire con il servizio DNS. Ma se per qualche configurazione speciale questo fosse richiesto è possibile abilitarlo.

Successivamente alla parte di configurazione globale, è presente la sezione relativa al caricamento dei moduli e quella relativa ai parametri di configurazione per dei moduli caricati.

```
# ----- module loading -----  
  
mpath="/usr/lib/OpenSer/modules/"  
loadmodule "tm.so"  
loadmodule "mi_fifo.so"
```

```
#-----modules parameters-----
modparam("tm", "wt_timer", 2)
modparam("mi_fifo", "fifo_name", "/tmp/OpenSer_fifo")

# ----- request routing logic -----

Nella nostra configurazione di test sono stati caricati solamente due moduli e configurati i relativi parametri. Ci sono comunque tantissimi moduli per OpenSer, che consentono di gestire gli scenari SIP più impensabili.

La sezione più importante del file di configurazione è quella delle routing logic. In OpenSer è possibile definire un comportamento diverso (route) per ogni messaggio SIP specifico. Nella nostra configurazione di test, volevamo solamente che i messaggi venissero reindirizzati verso l'applicazione che li aspettava, senza essere modificati. Grazie alla direttiva t_relay, il campo relativo al SIP URI in un messaggio SIP viene utilizzato per redirigere i messaggi verso l'applicazione che li deve ricevere. Nel nostro caso Sipp.

# ----- request routing logic -----

# main routing logic

route{
    t_relay();
}
```

A.12.4 Iperf

Iperf è uno strumento per misurare la larghezza di banda TCP/UDP, questo strumento consente di gestire diversi parametri e caratteristiche dei protocolli di trasporto. I dati riportati da Iperf riguardano, la larghezza di banda, il jitter, la perdita di pacchetti. Questo software consente di generare un dato ammontare di dati per un intervallo di tempo prestabilito. Grazie a quest'ultima funzionalità ci è stato possibile effettuare i nostri test per la saturazione della banda disponibile.

A.12.5 Sipp

SIPP è uno strumento OpenSource per testare le applicazioni basate su protocollo SIP. Consente di generare chiamate multiple e grazie alla funzionalità RTP/pcap replay consente anche di generare il traffico RTP associato alle chiamate. Grazie a queste funzionalità è stato possibile generare vere chiamate

Voip, che comprendevano sia il signalling SIP che il traffico RTP. Nella configurazione di base sono presenti degli scenari di default per UAC e per UAS e degli scenari personalizzabili. Questi ultimi sono scritti con linguaggio XML, di facile comprensione. SIPp può essere configurato per funzionare anche con il protocollo IPv6 ed è anche in grado di interagire con server Sip che utilizzano l'autenticazione TLS.

Sipp: la configurazione di test

Ora esamineremo la nostra configurazione di Sipp, per poter meglio capire come questo programma interagiva con OpenSer. Il comando digitato nel Pc a Milano è stato il seguente.

```
./sipp -sf uac_pcap_729.xml 195.113.113.147:5060 -rsa  
195.113.113.147:8888 -l 70 -m 1000 -r 10
```

Come è possibile intuire è stato caricato il file XML con uno scenario personalizzato, nel quale sono state modificate alcune linee per poter creare un flusso RTP codificato con il codec audio G.729. Per i dettagli si guardi l'appendice B. All' indirizzo Ip 195.113.113.147:5060 ad Ostrava, si trovava l'istanza di Sipp che doveva rispondere alle nostre chiamate e registrarne le statistiche. Mentre all'indirizzo 195.113.113.147:8888 si trovava il server Sip OpenSer. I messaggi Sip generati a Milano sono passati attraverso il server e successivamente rediretti verso l'applicazione Sipp che doveva rispondere alle chiamate. La direttiva "-l 70" è stata necessaria per non superare il limite di 70 chiamate contemporanee, mentre quella "-m 1000" per fissare il numero massimo di chiamate da generare. Invece l'ultima opzione "-r 10" è servita per incrementare il numero di chiamate concorrenti di 10 per ogni secondo, fino a raggiungere il limite di 70. L'istanza di Sipp ad Ostrava aveva una configurazione molto più semplice:

```
sipp -sn uas -p 5060
```

A.13 I nostri risultati

In questo capitolo presenteremo i risultati che abbiamo ottenuto durante i nostri test. Sono state definite due macro aree, per spiegare al meglio il nostro lavoro, la prima è dedicata ai test simulati e la seconda ai test reali. All'interno di ogni macro-area sono presentate le modalità operative i commenti e la raccolta dei risultati. Ogni macro area è stata suddivisa in base alla modalità di test, sicura o non sicura. Questo per focalizzare la nostra attenzione alle problematiche di sicurezza.

A.14 I test simulati

Come è stato precedentemente illustrato il test è stato configurato utilizzando l'applicativo Ixchariot, di seguito è presentata una figura che illustra schematicamente l'infrastruttura di testing.

[Vedi figura 5.1]

A.14.1 Simulazione non sicura

Il test è stato configurato per eseguire cinquanta chiamate contemporanee sia utilizzando il codec audio G.711Alaw sia utilizzando il codec G.729.[25][26]

La bandwidth disponibile era 10Mbit full duplex a Milano e 6Mbit full duplex ad Ostrava. Oltre ad utilizzare due diversi codec il test è stato anche ripetuto utilizzando il generatore di traffico Iperf per vedere come il network si comportava sotto stress. La durata delle chiamate era di sessanta secondi, il jitter buffer 60ms.

A.14.2 Previsioni sui risultati

Come documentato in [12] cinquanta chiamate contemporanee con il codec G.711Alaw occupano approssimativamente 9Mbit mentre con il codec G.729 la banda occupata scende intorno ai 5,5 Mbit. Secondo le nostre previsioni il test senza impostazioni di sicurezza poteva essere supportato appieno dalla infrastruttura di rete, anche sotto stress.

A.14.3 Risultati

Come è possibile vedere nei due grafici relativi a G.711Alaw e G.729, i valori dell'indice R-factor variano molto poco tra 1 una e 50 chiamate contemporanee in assenza di sovraccarichi di rete. Quando il test è stato eseguito utilizzando Iperf, invece, è stato possibile apprezzare un peggioramento delle prestazioni di circa il 5% con il codec G.711Alaw e di circa il 2% con il codec G.729. [Vedi figura 4.1]

[Vedi figura 5.2] [Vedi figura 5.3]

È necessario sottolineare che i diversi codec hanno comunque indici di riferimento prestazionale differenti, poiché hanno richieste di banda diverse. Il codec G.729 è risultato più resistente allo stress perché ha un'occupazione di banda inferiore, ma il livello qualitativo offerto da questo codec è più basso rispetto al G.711Alaw.

A.15 Simulazione sicura

La modalità di esecuzione del test è uguale a quella del test in modalità non sicura, la sola differenza è che il test è stato simulato all'interno di una Virtual Private Network realizzata con OpenVpn. È possibile trovare nell'appendice B la configurazione di OpenVpn.

A.15.1 Previsioni sui risultati

L'utilizzo del Transport Security Layer rappresenta una richiesta maggiore di banda all'infrastruttura di rete, la nostra previsione era quella di poter rilevare dei cali prestazionali e apprezzare le differenze tra i differenti codec.

A.15.2 Risultati

I risultati ottenuti hanno evidenziato come l'utilizzo di un'infrastruttura di sicurezza comporti un aumento della richiesta delle risorse disponibili. Questo aumento della richiesta di risorse viene quantificato in un costo economico per un Service Provider. L'impatto economico è uno dei motivi basilari della difficoltà di introdurre infrastrutture di sicurezza. Questo aspetto viene segnalato nei primissimi capitoli di [9].

Utilizzando il codec G.729 senza sovraccarico della banda, la differenza con il test senza impostazioni di sicurezza con 50 chiamate contemporanee è stata circa dell' 1%, lo stesso in condizioni di stress. Questo perchè, come già detto prima il codec G.729 ha una bassa occupazione di banda, va comunque ricordato che ha ottenuto un indice R-factor di 75 che indica un livello qualitativo non pienamente soddisfacente.

Invece con il codec G.711Alaw la differenza nel test senza carico, rispetto al test senza le impostazioni di sicurezza è stata circa dell'1%. In condizioni di stress è possibile notare invece, [Vedi figura 5.4] che l'infrastruttura di rete non è riuscita a supportare il carico richiesto. E il massimo di chiamate concorrenti supportate è stato di 25. Sul grafico è possibile notare come dopo le 25 chiamate vi è un vero e proprio collasso dell'infrastruttura. Va comunque sottolineato che fino a 25 chiamate l'indice R-factor era di poco inferiore a 90. Un ottimo indice prestazionale, considerando la VPN e il traffico aggiuntivo sulla rete.

[Vedi figura 5.4] [Vedi figura 5.5]

A.16 I test reali

Per realizzare questa parte del test ci siamo serviti oltre che di OpenSer, di sipp, un software open source che consente di generare traffico sip/rtp. I dettagli di configurazione si possono trovare nell'appendice B. Lo stesso per quanto riguarda la configurazione del sip server OpenSer. Per poter simulare il traffico Rtp con i codec G.711Alaw e con G.729 è stato necessario catturare del traffico con Wireshark. Le catture effettuate sono state poi utilizzate durante i test, per generare il traffico Rtp. Nel test reale è stato fissato un limite di 70 chiamate contemporanee. Nell'istante in cui il test comincia vengono inizializzate 10 chiamate, ogni secondo successivo alla partenza le chiamate concorrenti aumentano di 10, così dopo 7sec viene raggiunto il limite di 70 chiamate contemporanee. La durata di ogni chiamata è 60sec, una volta che le chiamate sono terminate il test le conclude. A questo punto il test ricomincia finchè non vengono raggiunte 1000 chiamate totali generate. Questo significa che il test di 70 chiamate contemporanee viene ripetuto circa 15 volte, prima di raggiungere le 1000 chiamate totali generate. Mentre i test erano in esecuzione sono stati provati diversi attacchi, per vedere come l'infrastruttura di rete reagiva. Il server OpenSer si trovava ad Ostrava, mentre Sipp configurato come client a Milano. Sulla macchina di Ostrava è stata anche configurata un'istanza server di Sipp, per poter rispondere alle chiamate che venivano indirizzate da OpenSer.

A.17 I test reali in modalità non sicura

Per quanto riguarda il test in modalità insicura, sono stati provati i seguenti attacchi:

- Man in the middle
- Sip flooding
- Udp flooding
- Sip registration hijacking

A.17.1 Previsioni sui risultati

Il traffico generato da 70 chiamate contemporanee, secondo i nostri calcoli poteva essere sostenuto dall'architettura di rete. La nostra curiosità era scoprire quale fosse il limite di resistenza ad attacchi di tipo Denial of Service, oltre che attestare la vulnerabilità ad attacchi del tipo MITM.

A.17.2 Risultati

Man in the middle

Com'è possibile notare nella figura 5.1 nel laboratorio di Milano avevamo a disposizione un segmento di rete con un hub. Grazie all'impiego di Wireshark è stato possibile sniffare tutte le conversazioni dell'utente senza nessun problema, come avevamo previsto. Installando uno switch sul segmento di rete è comunque possibile effettuare lo sniffing delle conversazioni, grazie alla tecnica di arp poisoning. Gli attacchi fin qui descritti non davano luogo a problemi di prestazioni sulla rete, poiché si limitavano a intercettare le conversazioni passivamente.

Sip flooding

Utilizzando invece Sipp, è stato possibile perpetrare l'attacco Sip flooding che consisteva nell' inviare messaggi di registrazione verso il server per cercare di esaurirne le risorse disponibili. In questo caso il software OpenSer si è dimostrato molto resistente e per riuscire a generare un rallentamento nel processing delle chiamate è stato necessario inviare circa 500 messaggi di registrazione al secondo, oltre al traffico generato dal test. Questo tipo di attacco genera problemi agli utenti del servizio, in quanto la gestione del signaling viene estremamente rallentata. È stato possibile notare, durante l'attacco, che i messaggi di chiusura delle chiamate del test sono arrivati anche con 30 secondi di ritardo.

Udp flooding

Con il tool iperf è possibile generare traffico tcp/udp e mandarlo verso una qualsiasi destinazione del tipo indirizzo IP/porta. A patto che ci siano servizi attivi sulla macchina bersaglio. Nel nostro caso

avevamo un istanza server di Iperf che raccoglieva il traffico da noi generato. La modalità di esecuzione del test prevedeva una generazione di traffico pari a 4Mbit, durante l'esecuzione dello stesso. Per il codec G.711Alaw il carico di 4Mbit è già abbondantemente elevato e non consente l'esecuzione di 70 chiamate contemporanee. Invece per quanto riguarda il codec G.729 è stato necessario aumentare il loading fino a 4,75Mbit per apprezzare un calo delle prestazioni.

Sip registration hijacking

Questa tecnica sfrutta un punto di debolezza del protocollo Sip, il fatto che le informazioni di signaling viaggiano in chiaro e che non esiste nessun meccanismo di integrità per scoprire i pacchetti manomessi. Grazie all'impiego di SiVus [21], sulla nostra infrastruttura è stato possibile disabilitare la registrazione di diversi utenti. Questa tecnica oltre a essere ampiamente illustrata su [5] viene brevemente esplicita su [23]. Per poter delegittimare l'utente della propria registrazione è possibile agire in diversi modi:

- Attaccare con metodo DoS il Sip Phone della vittima per renderlo irraggiungibile.
- Effettuare la deregistrazione della vittima inviando un messaggio al server modificato .
- Generare una race-condition sul server, dove l'attaccante invia, a piccoli intervalli regolari, un messaggio di registrazione, per cercare di confondere il Registrar server.

Una volta cancellata la registrazione della vittima l'attaccante la rimpiazza con la propria. In questo modo si compie un furto di identità e inoltre se la vittima viene contattata sarà l'attaccante a rispondere. In questo modo potrebbe venire a conoscenza di informazioni sensibili relative alla vittima.

Nella figura seguente è spiegato nel dettaglio il funzionamento di questo tipo di attacco.

[Vedi figura 5.6]

Nelle tabelle seguenti sono visualizzati i risultati prestazionali dei test in modalità non sicura. Per calcolare il MOS è stato utilizzato [22]. Il valore è stato calcolato sulla capacità di gestire il totale delle 1000 chiamate generate.

[Vedi figura 5.7] [Vedi figura 5.8]

Come è possibile notare i risultati si discostano dalle simulazioni, di fatti con il codec G.711Alaw anche in assenza di traffico non è stato possibile apprezzare una grande qualità, probabilmente il limite di 70 chiamate contemporanee era troppo elevato per la banda a disposizione. I risultati con il codec G.729 invece hanno avuto un andamento più lineare, per via della minore occupazione di banda.

A.18 I test reali in modalità sicura

Per l'esecuzione dei test sicuri è stata utilizzata la soluzione OpenVpn. La nostra scelta è caduta su questo software, perchè si è voluto cercare di proteggere sia il signaling Sip che il traffico Rtp. Il modo

più semplice per poter realizzare questo è attraverso l'impiego di una Virtual Private Network basata sul protocollo TLS.

Come illustrato in [10] e [11], se venisse utilizzato il protocollo IPv6, ormai forse solo un'utopia, sarebbe possibile nativamente abilitare le opzioni per l'ESP (Encapsulated Security Payload), oltre che i controlli di integrità nell'AH (Authentication Header). Quindi con una soluzione che sta a livello 3 del modello OSI, si riuscirebbe a proteggere una piattaforma Voip. Le analisi effettuate in [10] e [11] sottolineano che l'adozione di IpSec comporta comunque dei problemi. Infatti consuma notevolmente la banda per informazioni di servizio e rende tutta l'infrastruttura molto sensibile a problemi di congestione. Questo problema tende ad aumentare con l'aumentare degli hop tra sorgente e destinazione.

A.18.1 Previsioni sui risultati

Le aspettative si sono focalizzate nel capire quale fosse la possibilità di scongiurare determinati attacchi e di mitigare l'effetto di altri. Il protocollo TLS che è stato utilizzato per i nostri test, si avvale di una tecnologia crittografica a chiavi asimmetriche, che dovrebbe essere abbastanza forte da dover garantire la privacy delle informazioni per almeno un centinaio di anni. La resistenza di un algoritmo di crittografia a chiavi asimmetriche sta prima di tutto nella capacità di custodire le chiavi in maniera corretta e secondariamente risiede nella bontà dell'algoritmo stesso. Alla luce di questo è necessario sottolineare come il problema cruciale della sicurezza informatica risieda sempre e comunque negli umani.[9] Un'altra risposta che aspettavamo da questo test era di riuscire a misurare il calo delle prestazioni dovuto all'adozione della Virtual Private Network.

A.18.2 Risultati

La resistenza della piattaforma Voip con Vpn ad attacchi di intercettazione del flusso o al Sip registration hijacking è risultata totale, poiché tutto il flusso di comunicazione viene cifrato all'interno del tunnel Vpn. Solamente se un attaccante riuscisse, in qualche modo, a prendere possesso della macchina dove risiede la certification authority(CA) avrebbe la possibilità di fabbricarsi un certificato valido per poter entrare in comunicazione con le macchine bersaglio. Questo deve essere per definizione un compito difficile, quindi è consigliabile che il ruolo di Ca venga svolto da una macchina esterna alla comunicazione via Vpn. Durante il test in modalità protetta senza l'uso di Iperf per la generazione di traffico, i risultati sono rimasti abbastanza simili ai test senza protezione. L'apporto di funzionalità di protezione all'interno della nostra infrastruttura Voip non ha quindi influenzato le prestazioni in modo considerevole.

Per quanto riguarda invece il test che prevedeva l'inserimento di 4Mbit di traffico sulla rete, con il codec G.711Alaw il decremento prestazionale è del 10% rispetto alla stessa versione del test senza impostazioni di sicurezza. Mentre questo decremento è stato del 5% con il codec G.729. È possibile quindi evincere che con l'adozione di impostazioni di sicurezza basate su VPN, la resistenza della rete ad attac-

chi di tipo DoS diminuisce, perché diminuisce la disponibilità di banda totale. Una possibile soluzione per poter mitigare questo problema è di evitare l'utilizzo delle porte Tcp/Udp standard, per cercare di confondere l'attaccante sui servizi attivi sulla macchina bersaglio. Un'altra possibile soluzione può essere realizzata creando segmenti di rete fidati dai quali accettare il traffico. Elevare troppo il livello di sicurezza può condurre però alla scelta di soluzioni particolarmente costose. Probabilmente utili per un'organizzazione governativa che per necessità ha bisogno del massimo livello di sicurezza.

Nella nostra infrastruttura voip con codec G.729, per mantenere un livello prestazionale che abbia un indice MOS>3, il traffic loading massimo sopportabile è stato stimato in 2,75Mbit. Mentre per quanto riguarda il codec G.711A law sarebbe necessario avere una disponibilità di banda maggiore o ridurre il numero di chiamate contemporanee altrimenti il livello prestazionale rimane davvero basso.

[Vedi figura 5.9] [Vedi figura 5.10]

A.19 Conclusioni

Le applicazioni Real-time sono davvero sensibili alle variazioni che possono avvenire sulla rete, soprattutto quando si tratta di reti eterogenee. I nostri test hanno cercato di mostrare i punti deboli di un'infrastruttura Voip e la soluzione proposta è solamente una tra le molteplici possibili.

Quando si ritiene necessario introdurre misure di sicurezza bisogna considerare il problema sotto diversi punti di vista, i più importanti dei quali sono: fattore economico, fattore tecnico e fattore umano. La messa in opera di soluzioni di sicurezza dev'essere un compromesso tra i diversi fattori sopraccitati.

Quindi la scelta da noi adottata è stata un largo compromesso tra i vari limiti che avevamo: pochi fondi, un'architettura estremamente eterogenea e con gli endpoint a grande distanza, la necessità di una facile configurazione. Nonostante tutti i vincoli citati è stato dimostrato che con requisiti di banda ben inferiori a quelli di un Service Provider è possibile proteggere in maniera abbastanza semplice un'infrastruttura Voip. Questa infrastruttura è stata in grado di supportare almeno 50 chiamate contemporanee, mantenendo un livello prestazionale qualitativamente sufficiente, almeno con il codec audio G.729. Di sicuro interesse sarà osservare come il mercato della telefonia su reti dati si evolverà da un punto di vista della sicurezza. In linea con la conclusione di [12] gli autori continueranno a seguire l'evolversi di questa tecnologia e dei problemi ad essa collegati.

Appendix **B**

Configuration files

B.1 Openser.cfg

```
# ----- global configuration parameters -----

debug=3          # debug level (cmd line: -dddddddddd)
fork=yes
log_stderr=no    # (cmd line: -E)
children=4

disable_tcp=yes
disable_dns_blacklist=yes
disable_dns_failover=yes

# Uncomment these lines to enter debugging mode
#fork=no
#log_stderr=yes
#

port=8888

# ----- module loading -----

mpath="/usr/lib/openser/modules/"
loadmodule "tm.so"
```

```
loadmodule "mi_fifo.so"
#-----modules parameters-----
modparam("tm", "wt_timer", 2)
modparam("mi_fifo", "fifo_name", "/tmp/openser_fifo")

# ----- request routing logic -----

# main routing logic

route{
    t_relay();
}
```

B.2 Openvpn.cfg

```
# Server_side.conf
proto udp
keepalive 10 60 dev tun persist-key persist-tun
ifconfig-nowarn ca ca.cert cert ser.cert key server.key dh dh.pem
comp-lzo
server 10.0.0.0 255.255.255.0
client-to-client
status openvpn-status.log
verb 0

# Clients_side.conf
remote 159.149.153.68
dev tun
client 10.0.0.0 255.255.255.0
tls-client
ca ca.cert
cert cli.cert
key client.key
rport 4096
verb 4
```

```
ping 10
comp-lzo
```

B.3 Sipp configuration

```
./sipp -sf uac_pcap_729.xml 195.113.113.147:5060 -rsa
195.113.113.147:8888 -l 70 -m 1000 -r 10 # g729 Codec
```

```
./sipp -sf uac_pcap.xml 195.113.113.147:5060 -rsa
195.113.113.147:8888 -l 70 -m 1000 -r 10 # g711a Codec
```

```
#Configuration of the XML sipp scenarios
```

```
For g711a
<exec play_pcap="pcap/g711a.pcap"/>
a=rtpmap:0 PCMU/8000
```

```
For g729
<exec play_pcap="pcap/g729.pcap"/>
a=rtpmap:18 G729/8000
```

```
#On the server side the configuration of sipp was
```

```
sipp -sn uas -p 5060
```

B.4 Iperf configuration

```
#Client side
```

```
iperf -c 195.113.113.147 -p 12800 -u -b 40960000 -t 350
```

```
#Server side
```

```
iperf -s -u -p 12800
```

Bibliography

- [1] Voice over Ip fundamentals - J.Davidson J.Peters - Cisco Press, 2000
- [2] Zfone Project - www.zfoneproject.com
- [3] Vlan Security white paper - Cisco <http://www.cisco.com>
- [4] Dsniff suite - <http://monkey.org/dugsong/dsniff/>
- [5] Hacking Exposed:Voip - D.Endler M.Collier - McGraw-Hill - Osborne, 2007
- [6] Practical Voip Security - T.Porter - Syngress, 2006
- [7] Securing VoIP Networks: Threats, Vulnerabilities, and Countermeasures
P.Thermos A.Takanen - Addison Wesley Professional, 2007
- [8] Computer Networking and the Internet - Fifth edition - Fred Halsall - Addison Wesley, 2004
- [9] Computer Security - Second edition - Dieter Gollmann - Wiley, 2006
- [10] D.Bruschi, E.Rosti, R.Barbieri, Voice Over IPsec, 18th Annual Computer Security Applications Conference (ACSAC '02) p. 261
- [11] D.Bruschi, R.Barbieri, L.Pomelli, Studio e analisi delle problematiche di trasmissione di voce cifrata in ambienti con QoS. Tesi di laurea, Università degli studi di Milano, A.A. 2001/2002
- [12] M.Voznak, A.Nappa, The performance evaluation of Voip infrastructure. CESNET IP telephony Seminar, University of Prague, November 16, 2007
- [14] IETF RFC 3830, MIKEY: Multimedia Internet KEYing
- [15] IETF RFC 3550, Real-Time Transport Protocol
- [16] IETF RFC 3711, The Secure Real-Time Transport Protocol

- [17] IETF Internet Draft, ZRTP:Media Path Key Agreement for Secure-RTP
- [18] Internet Denial of Service: Attack and defence mechanism - J.Mirkovic, S.Dietrich, D.Dittrich, P.Reiher - Prentice Hall, 2005
- [19] Sipp project, <http://sipp.sourceforge.net>
- [20] Cain and Abel, <http://www.oxid.it>
- [21] SiVus, www.vopsecurity.org
- [22] Mos Calculator, <http://davidwall.com/MOSCalc.htm>
- [23] Security Focus, www.securityfocus.com
- [24] Rtpptools, <http://www.cs.columbia.edu/IRT/software/rtptools/>
- [25]
- [26]
- [27]