Master Thesis
Electrical Engineering
Thesis no: MEE10:74
June 2010

BLEKINGE INSTITUTE OF TECHNOLOGY SCHOOL OF ENGINEERING

DEPARTMENT OF TELECOMMUNICATION SYSTEMS

Security Issues of SIP

MASTER'S THESIS

Name:	**Gulfam Asghar**
Email:	**asga07@student.bth.se**
Name:	**Qanit Jawed Azmi**
Email:	**qjaz06@student.bth.se**

Blekinge Institute of Technology

School of Engineering

Supervisor: (Doktorand) Mr. Karel De Vogeleer

Examiner: Prof. Dr. Adrian Popes

Table of Contents

Abstract

Voice over IP (VoIP) services based on Session Initiation Protocol (SIP) has gained much attention as compared to other protocols like H.323 or MGCP over the last decade. SIP is the most favorite signaling protocol for the current and future IP telephony services, and it's also becoming the real competitor for traditional telephony services. However, the open architecture of SIP results the provided services vulnerable to different types of security threats which are similar in nature to those currently existing on the Internet. For this reason, there is an obvious need to provide some kind of security mechanisms to SIP based VOIP implementations. In this research, we will discuss the security threats to SIP and will highlight the related open issues. Although there are many threats to SIP security but we will focus mainly on the session hijacking and DoS attacks. We will demonstrate these types of attacks by introducing a model/practical test environment. We will also analyze the effect and performance of some the proposed solutions that is the use of Network Address Translation (NAT), IPSec, Virtual Private Networks (VPNs) and Firewalls (IDS/IPS) with the help of a test scenario.

Objectives

The main objective of this master's thesis is to learn the security issues of SIP and their practical implications. The primary goal of this project is to present the SIP security issues and will analyze the performance of some proposed solutions i.e. NAT/firewall traversal.

Acknowledgments

We would like to thank our supervisor at the Blekinge Institute of Technology (Sweden), who has helped and guided us throughout this thesis work. Our everlasting gratitude and acknowledgements are for our parents for their moral support and encouragement throughout our study period at BTH.

1 Introduction

1.1 Background

The Session Initiation protocol (SIP) is a signaling protocol mostly used for setting up & tearing down the real-time multimedia sessions, for example the voice & video call over the Internet, online games, instant messaging etc. Due to its open architecture and flexible design, SIP has gained enormous acceptance as an Internet telephony signaling protocol for wired and wireless networks. It was approved by IETF as a signaling standard and permanent component of the IMS design for IP based streaming multimedia services in November 2000.

SIP is a text based client-server protocol similar to the HTTP, this text-based nature of SIP messages opens many opportunities for threats like spoofing, hijacking and message tampering. The use of malicious SIP messages is also a possibility which can cause unauthorized access or Denial of Service (DoS). DoS attacks are precise attempt to halt a target thereby preventing genuine users from making use of its services, usually by directing an excessive amount of network traffic at its interfaces. In most of the cases SIP proxy servers face the public Internet in order to accept requests from worldwide IP endpoints which creates number of potential opportunities for DoS attacks that must be recognized and addressed by the implementers and operators of SIP based IP telephony solutions.

As described in [RFC 3261], SIP uses some popular protocols such as TCP, UDP, SCTP for the transport of multimedia sessions. As a result of this, SIP inherits the weaknesses or vulnerabilities of these protocols. For example, considering that UDP is vulnerable to session hijacking, it is most probably that the SIP can face the same type of problems. Moreover, the interconnection of SIP with PSTN (Public Switched Telephone Network) can introduce more opportunities for intrusion, such as attacks on VOIP gateways.

There are two main types of possible threats to a SIP based system:

External threats - it can be defined as the intrusion done by someone who is not participating in the communication or multimedia session.
Internal threats - are referred to as the attacks launched by a SIP call participant.

The basic security services desired for SIP based VOIP networks are:

1. Maintaining the integrity and confidentiality of messages.
2. Preventing replay attacks or message spoofing.
3. Providing the authentication and privacy for the participants in a session.
4. Preventing DoS attacks.[1]

This thesis work will study the security threats to SIP in details and will also analyze the performance of some proposed solutions such as the use of VPNs, NAT and Firewall in SIP based VOIP networks.

1.2 Problem Description

The main objective of this thesis work is to learn the security issues of SIP protocol and their practical implications. This will be done theoretically and by using a model test environment. It is also desirable to compare the performance of some proposed solutions.

Motivation:

SIP is the most favorite signaling protocol for the current and future IP telephony services, and it‘s also becoming the real competitor for traditional telephony services. However, the open architecture of SIP results the provided services vulnerable to different types of

security threats currently existing on internet like spoofing, hijacking and message tampering. The use of malicious SIP messages is also a possibility which can cause unauthorized access or Denial of Service (DoS). DoS attacks are precise attempt to halt a target thereby preventing genuine users from making use of its services, usually by directing an excessive amount of network traffic at its interfaces. For this reason, there is an obvious need to provide some kind of security mechanisms to SIP based VOIP implementations.

Our thesis outlined the analysis of various types of threats that could be used to exploit the SIP by means of exploiting the authentication, confidentiality and integrity of a SIP based VOIP implementation. We have studied the requirements of a firewall to pass SIP signaling in and out of a private network that uses NAT. This subject has been covered in-depth throughout the entire project work.

To secure a SIP based VOIP system, it is necessary to understand the nature of different kinds of attacks and how they can affect to degrade the performance of a SIP system. Many solutions and strategies have been proposed to solve the mentioned security issues, but there is an obvious need to analyze them on the basis of performance.
The important aspect of this thesis was to implement a SIP enable IPSec VPN tunnel to provide greater security for our proposed network which was achieved by means of utilizing Cisco 3745 IOS and Cisco Router and Security Device Manager (SDM) software. Our project also discussed some other security mechanisms theoretically, that can be used with SIP to ensure a greater level of security and privacy.

The Goal of this thesis work is to:

- Study the SIP security mechanisms.

i. Learn about the threats to SIP security.
ii. Analyze the use of IETF standard security protocols such as TLS and IPsec with SIP protocol.
iii. Analyze the use of VPNs, Firewalls and NAT in SIP based VOIP networks.

This thesis will discuss the SIP related security problems and will illustrate some of the identified threats/attacks, their impact on the overall SIP security and will briefly describe the SIP's security mechanisms. In addition, it will also present the analysis of some of the proposed solutions such as the use of NAT, VPNs and Firewalls theoretically and with the help of a practical test scenario.

The practical test scenario which will be based on an open source or trail version of any suitable IP PBX system, a VPN server, a state full hardware firewall, and two or three SIP softphone users. From the firewall perspective, we will analyze the use of some methods/techniques to block the DoS attacks. VPN implementation will analyze the importance of creating secure tunnels to strengthen the SIP security.

1.3 Scope of work

Much attention is paid to enhance the features and interoperability of SIP protocol with less focus on security. A SIP based VOIP network is potentially vulnerable to general IP and VOIP attacks as well as attacks which are unique to SIP.

To secure a SIP based VOIP system, it is necessary to understand the nature of different kinds of attacks and how they can affect to degrade the performance of a SIP system. Many solutions and strategies have been proposed to solve the above mentioned security issues, but there is an obvious need to analyze them on the basis of performance. Our work is an attempt to explore the in depth knowledge of different types of SIP related security vulnerabilities and a comparative analysis of some of the proposed solutions present today.

1.4 Test Environment

We will try to establish the connection between the theoretical and practical aspects of the threats to SIP security, and will also try to compare the performance of some of the existing proposed solutions by performing some test experiments in a sample test system.

The test environment will comprise of:

- An open source or trail version of any suitable software (Cisco Unified Call Manager or Asterisk) based IP PBX system.
- A VPN server will be used for creating point-to-point or point-to-multipoint secure tunnels.
- A state full firewall (Cisco or Juniper) which is capable of providing IPS/IDS functionality.
- Two or three PC based SIP users with soft phones.

1.5 Outline

This report consists of 5 chapters. Chapter 1 will give you the general overview of this thesis work. Chapter 2 will briefly introduce the underlying basic concepts which include basics of SIP protocol, SIP security mechanisms and threats to SIP security. Chapter 3 will discuss the use of different types of firewalls and VPNs in SIP based VOIP networks. Chapter 4 will present the comparative analysis of deploying SIP based VOIP system with NAT, Firewalls and VPNs based on our model test scenario. The chapter 5 will be based on summary, conclusions, results and references.

2 Overview of SIP & Security Considerations

2.1 The Evolution of SIP protocol

The original purpose of SIP was to invite users for Mbone based media sessions when IETF first commissioned it. The protocol then evolved rapidly and is currently a most popular and flexible protocol for almost all types of IP based multimedia sessions including multicast and point-to-point sessions. The current form of SIP protocol (SIPv2) was not designed from the scratch instead it is an outcome of combination of best features of two IETF's proposed protocols namely Session Initiation Protocol version 1.0 (SIPv1) and Simple Conference Invitation Protocol (SCIP).

The Multimedia Conference Control (MMCC) community was developed by Eve Schooler and provided software for point-to-point and point-to-multipoint teleconferences, with audio and video tools. MMCC used Connection Control Protocol (CCP) as a transaction oriented protocol to interconnect the users. The typical form introduced for transaction was consist of one request from the originator and one response from the recipient. The CCP also utilized the User datagram protocol (UDP) for transport to ensure the reliable delivery of packets. These two fundamental multimedia systems gave birth to the design of SIP developed by Mark Handley and Eve Schooler. The first version of SIP (SIPv1) was submitted to IETF in 1996 for consideration and was published as an Internet-draft in 1997. The text based SIPv1 used Session Description Protocol (SDP) to define sessions and UDP to transport protocol messages. The significant feature of SIPv1 was the concept of registrations to conference address servers. When a user registered his or her location to a conference address server, the server can route invitations correctly on behalf of the user and can also provide mobility to a certain level. However, SIPv1 was only capable of providing session establishment functionality and signaling may stopped once the user joined the conversation or session which concludes that mid-conference and tearing down controls were yet to develop.

The draft of another similar protocol, the Simple Conference Protocol (SCIP) was submitted to IETF by Henning Schulzrinne during 1996 which used Transmission Control Protocol (TCP) for transport and was based on Hypertext Transfer Protocol (HTTP). The protocol used email addresses to identify users and was triggered to provide universal

resource identifiers for both synchronous and asynchronous transmission. SCIP was designed to perform signaling after session establishment, to make changes in ongoing sessions and to close existing session. These enhancements were developed by describing a new format instead of recycling the mechanism of SDP.

At the 35th IETF Conference held in Los Angeles, both Schooler and Schulzrinne presented SIP and SCIP respectively. The resulting protocol, Session Initiation Protocol version 2.0 (SIPv2) was the merger of the promising features of both SIPv1 and SCIP. The protocol then achieved proposed standard status and was published by IETF as [RFC 2543] in 1999. The new SIP utilized both UDP and TCP for transport and the mechanism was based on the exchange of HTTP like text messages. It also used SDP to define multimedia sessions. An Internet-draft consists of clarifications and bug fixes was submitted to IETF and then published as [RFC 3261] in 2000 which obsolete or replaced the previous [RFC 2543]. [2]

2.2 Basics of SIP protocol

2.2.1 Definition

The Session Initiation Protocol (SIP) can be defined as an application-level signaling protocol used for the establishment and management of multimedia sessions (example: voice, video conferencing and instant messaging) over an IP network. It is defined as an industry standard protocol by IETF in [RFC 3261].

2.2.2 Basic functionality of a SIP based VOIP system

SIP is a text based client-server protocol similar to HTTP and consists of the following logical entities:

- User Agents
- SIP Proxy servers
- SIP Registrar servers
- SIP Redirect Servers
- Location Servers
- Media Gateways

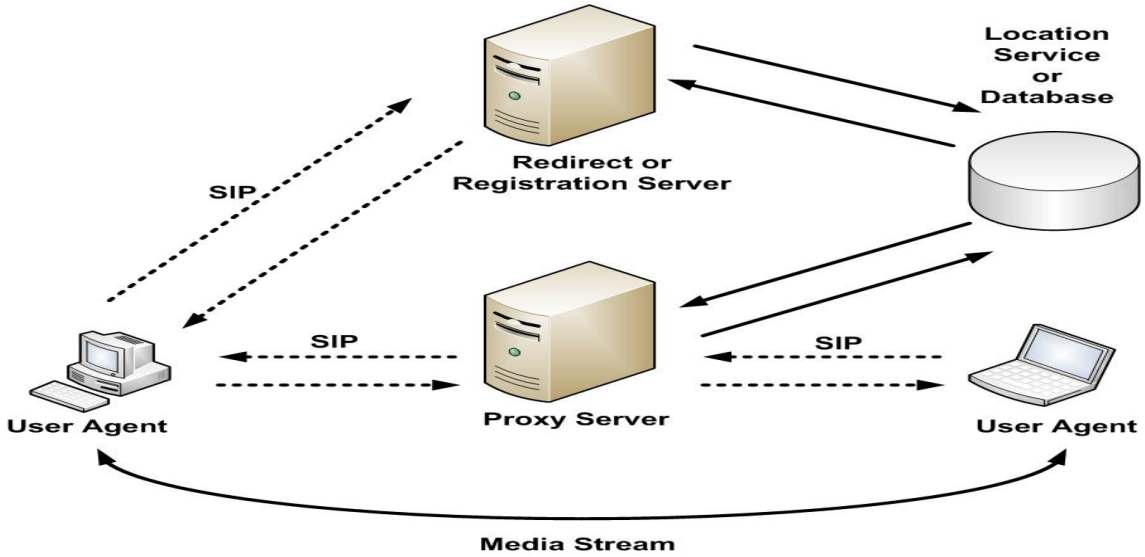


Figure 1. Basic Logical Components of A SIP Based System [3]

A User Agent is an endpoint of a SIP system which initiates a media session on behalf of a user, usually an IP-phone or a Soft-phone. Proxy, Registrar, Redirect, Location and media gateway Servers are the software/hardware applications that make it possible for a User Agent (UA) to communicate with other users or UAs, register himself with the network and to get redirect responses. All these servers are normally implemented physically on one machine. Media Gateways are usually used to provide the interconnection between the Public Switched Telephone Network (PSTN) and IP based packet data network. SIP uses the following IETF standards to ensure basic functionality:

5. Session Description Protocol(SDP): is used to specify the parameters for the media sessions.
6. Real-time Transport Protocol(RTP): is responsible for reliable end-to-end transmission of media streams.
7. RTP Control Protocol: is used to transmits the control data for the RTP streams.
8. Compressors/De-compressors: are used to encode and compress the media for efficient transmission.
9. Feature Standards: SIP is highly flexible and extensible, many standards are used to describe as feature implementation in the [RFC 3261]. But very few are yet realized in many SIP based implementations. [1]

2.2.3 Overview of Operation (A Simple Call Setup Example)

SIP operation is based on exchange of HTTP-like text messages that can be either a request or an acknowledgement to a request which consists of the header fields and the message body. The message body is used to define different session requirements and to encapsulate various types of signaling while the header holds the detailed information about the type of a particular session that is going to establish.

Figure 2. shows the basic message exchange between two SIP-enabled devices. It is also considered that both the devices are connected to an IP based network and are aware of IP addresses of each other. The sender (James), initiates the session by sending a SIP 'INVITE' message to recipient (Maria). The 'INVITE' message is composed of the information about the type of session (voice or multimedia) and contains the following elements:

INVITE sip:maria@energy.org SIP/2.0

Via: SIP/2.0/UDP lab.test-sip.org:5060
Max-Forwards: 60
To: Maria Anderson <sip:maria@energy.org>
From: James Franklin <sip:james@test-sip.org>
Call-ID: 302401884@test-sip.org
CSeq: 1 INVITE
Subject: About the the power outage....
Contact: sip:james@test-sip.org
Content-Type: application/sdp
Content-Length: 140

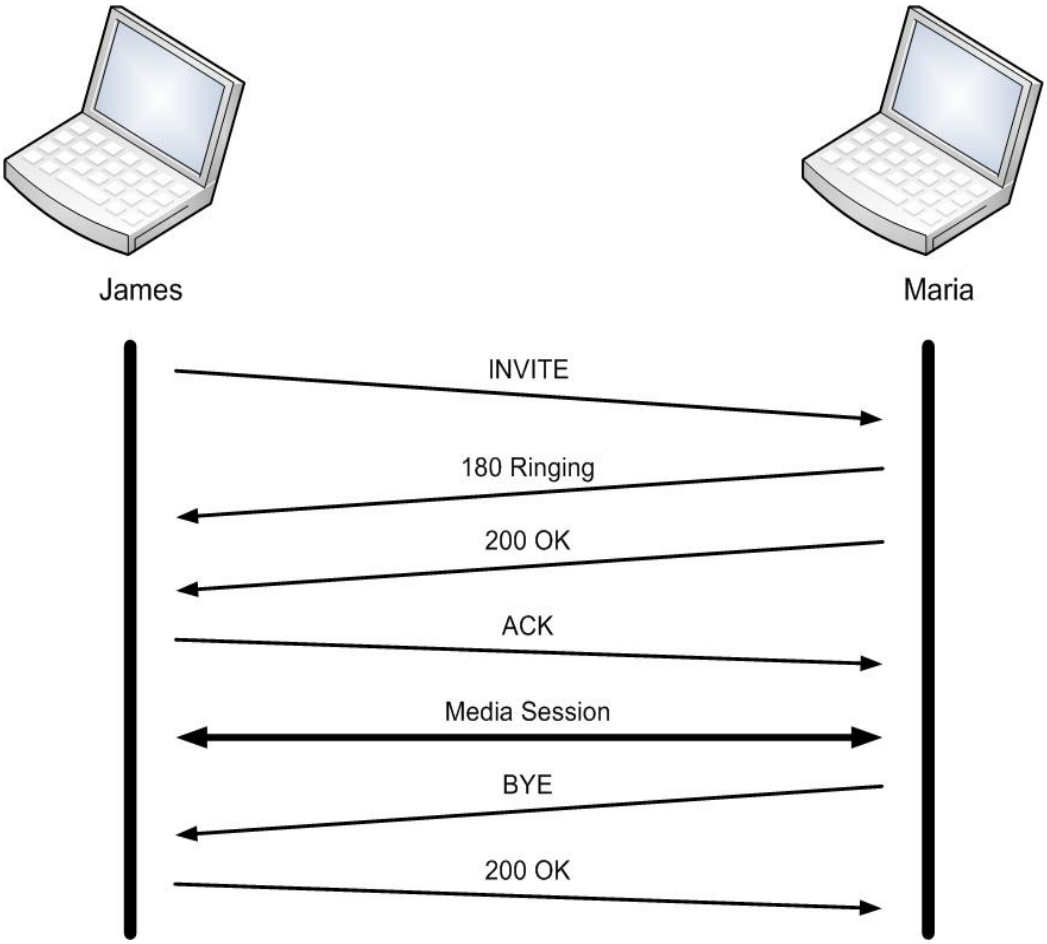


Figure 2. A Simple SIP Message Flow [3]

INVITE sip:maria@energy.org SIP/2.0

Via: SIP/2.0/UDP lab.test-sip.org:5060

Max-Forwards: 60

To: Maria Anderson <sip:maria@energy.org>

From: James Franklin <sip:james@test-sip.org>

Call-ID: 302401884@test-sip.org

CSeq: 1 INVITE

Subject: About the power outage....

Contact: sip:james@test-sip.org

Content-Type: application/sdp

Content-Length: 140

v=0

o=James 289543789 289543789 IN IP4 lab.test-sip.org

s=Phone Call

c=IN IP4 100.103.105.107

t=0 0

m=audio 49260 RTP/AVP 0

a=rtpmap: 0 PCMU/8000

The fields described in upper half of this 'INVITE' message are called headers. The first line define the method which is INVITE, followed by the Request URI (Uniform Resource Indicator) and SIP version number (2.0), all divided apart by spaces. The Request-URI is a special form of SIP URL (Universal Resource Locater) and is used to indicate the address or location of the recipient.

The first header is a 'Via' header which holds an address. A SIP device that initiates or participated in a session, copies its own address into the 'Via' header. This is normally specified as a host name that can be resolved into an IP address by using a DNS (Domain Name Service) query. The 'Via' header also contains the information about the SIP version number(2.0), type of transport protocol (UDP), host name or address and port number (5060). [3][4]

The next header is 'Max-Forwards', which is has an initial value of '60'. This value is decremented each time the message is passed through a SIP server allowing simple loop detection. The next two headers are 'To' and 'From', which identifies the sender and receiver of the message. The 'Call-ID' header uses the same form as an email address but actually is an identifier to keep track of a specific SIP session. The initiator of the request generates a locally unique string and with the addition of its own host name after '@' sign, a globally unique string is formed. In addition to the 'Call-ID' header, both sender and receiver also put up a random identifier which is specific for each call. The identifiers are referred as tags and enclosed in the 'To' and 'From' headers as the session is commenced. The initial 'INVITE' to establish a session contains the 'Call-ID' and 'From' tag while the response to the 'INVITE' has the 'To' tag only. The combination of both 'To'

and 'From' tags together with the 'Call-ID' uniquely identifies an established session, termed as a “dialog.”

The next header shown in the above example is 'Cseq' or command sequence which contains a number followed by the method (INVITE). This number (1) is incremented each time when a new request has been sent. The 'Via' header along with the 'To', 'From', 'Max-Forwards', 'Call-ID' and 'Cseq' headers corresponds to the minimum set of headers any SIP message may contain.

Other headers can also be included to fulfill the requirements of any additional information. For instance, the 'Contact' header contains the SIP URL of the initiator (James) which can be utilize to route the packets directly to him. An optional 'Subject' header is present in this example which could be useful to assist the recipient (Maria) about the nature of the call. We normally do the same type of screening in emails messages as well by using 'Subject' and 'From' headers. The essential information about the media stream is also present in this example which is described in additional headers. The 'Content-Type' and 'Content-Length' headers indicate the type and length of message body which is 'SDP' and '140' respectively. A blank line is used to separate the message body from the header which is present after the header 'Content-Length'. In this example, the message body is composed of seven lines of SDP data defining the media stream parameters that the call initiator (James) wants. These attributes are required as SIP cannot assume about the type of media session such as audio or video to be set up. The SDP fields present in this example are:

iv. IP address (100.103.105.107)
v. Media format (audio)
vi. Port number (49260)
vii. Media transport protocol (RTP)
viii. Media encoding scheme (PCM µ-Law)

SIP message 'INVITE' is just an example of a SIP request message. However, there are five more methods or types of SIP requests are defined in [RFC 3261] and some others are described in extension RFCs.

The succeeding exchanged message in our example is '180 ringing', which is a response sent in favor of the 'INVITE' message. This message identifies that the recipient (Maria)

has received the message and modifying is in progress. The modifying could be the ringing alert, flashing on-screen message or any other way of appealing the attention of the recipient.

Responses are in numerical form and are identified by the first digit of the string of number. A '180 ringing' is an informational response which is identified by the first digit '1'. Informational responses are normally used to express the current status of the call. Response codes of HTTP version 1.1 with some extensions and additions are the basis of many SIP response codes. [3][4]

The '180 Ringing' response is based on the following structure:

SIP/2.0 180 Ringing;

Via: SIP/2.0/UDP lab.test-sip.org:5060; branch=z9hG4bKfw19b

; received=100.103.105.107

To: Maria Anderson <sip:maria@energy.org>;tag=a53e42

From: James Franklin <sip:james@test-sip.org>>;tag=76341

Call-ID: 302401884@lab.test-sip.org

CSeq: 1 INVITE

Contact: <sip:maria@low.energy.org>

Content-Length: 0

The above message was generated by replicating many of the 'INVITE' header fields (Via, To, From, Call-ID and c Seq) complemented by the start line which holds the values of SIP version number, response code and the reason phrase. This method articulates the approach of simple processing of messages used in SIP.

The original branch parameter plus the additional received parameter is present in the 'Via' header. This additional received parameter carries the IP address of the originator (100.103.105.107) which is the same address that is used in the URI of 'Via' header and resolved by using DNS (lab.test-sip.org). Observe that the 'To' and 'From' headers are not reversed as you might anticipate them to be. This message is sent to James to Maria but

the header fields are opposite. This is because the SIP usually identifies direction of the request instead of direction of the message. Since James generated this request, all the following responses will indicate the direction of James to Maria (To: Maria From: James).

As Maria created the 'To' header's tag, all the succeeding requests and responses in this particular session or dialog will hold on the tags constructed by both James and Maria. The response also has a 'Contact' header, which constitutes that the Maria can be reached directly by using the encapsulated address once the session is commenced. When the recipient Maria has accepted the call, a '200 OK' response is sent to James. This response also concludes that the media session type desired by the initiator (James) is acceptable. The message body of '200 OK' holds the media information of Maria:

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP lab.test-sip.org:5060; branch=z9hG4bKfw19b
     ; received=100.103.105.107
To: Maria Anderson <sip:maria@energy.org>;tag=a53e42
From: James Franklin <sip:james@test-sip.org>;tag=76341
Call-ID: 302401884@lab.test-sip.org
CSeq: 1 INVITE
Contact: <sip:maria@low.energy.org>
Content-Type: application/sdp
Content-Length: 155
 v=0
o=Maria 2890844528 2890844528 IN IP4 low.energy.org
s=Phone Call
c=IN IP4 200.203.205.207
t=0 0
m=audio 50800 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

This response is made in the similar fashion as the '180 Ringing' response and holds the same 'To' tag and 'Contact' URI. However, the media related parameters must be encapsulated in a SDP message body. The SDP contains:

- End-point IP address (200.203.205.207)
- Media format (audio)
- Port number (50800)
- Media transport protocol (RTP)
- Media encoding scheme (PCM µ-Law)
- Data sampling rate (8,000 Hz)

The last step before the actual transmissions of media stream begin is to send the confirmation of the received response. This is done with an 'ACK' request before the actual media transmission is about to begin. This exchange of media related information enables the media session to be established using RTP in this case:

ACK sip:maria@low.energy.org SIP/2.0

Via: SIP/2.0/UDP lab.test-sip.org:5060; branch=z9hG4bK321g

Max-Forwards: 60

To: Maria <sip:maria@energy.org>;tag=a53e42

From: James Franklin <sip:james@test-sip.org>;tag=76341

Call-ID: 302401884@lab.test-sip.org

CSeq: 1 ACK

Content-Length: 0

The 'Cseq' header has exactly the same value as the 'INVITE', but this time the method is set to 'ACK'. As both James and Maria are agreed now on different media related parameters, the media session can now start transmitting the real media stream by using RTP. The branch parameter of 'Via' header holds a new transaction identifier because a separate transaction is conceived by sending 'ACK' for '200 OK'.

This message exchange mechanism ensures that the SIP is an end-to-end signaling protocol which does not requires any SIP network or a SIP server. The two end devices

running a SIP protocol stack and are aware of IP addresses of one another, can establish a media session. This example also depicts the client-server based nature of SIP protocol. When James generated the 'INVITE' request, he was behaving as a SIP client and when Maria responded to the request, she was acting as a SIP server, and after the media session is commenced, Maria created the 'BYE' request and behaved as a SIP client. Because of this behavior any SIP-enabled device must contain both SIP-Client and SIP-Sever software. This is where SIP is quite distinct from other client-server Internet protocols (HTTP or FTP).

In Figure 2. a 'BYE' request is sent by Maria to tear down the media session:

BYE sip:james@lab.test-sip.org SIP/2.0

Via: SIP/2.0/UDP low.energy.org:5060;branch=z9hG4bK392kf

Max-Forwards: 70

To: James Franklin <sip:james@test-sip.org>;tag=76341

From: Maria Anderson <sip:maria@energy.org>;tag=a53e42

Call-ID: 302401884@lab.test-sip.org

CSeq: 1 BYE

Content-Length: 0

Again a separate transaction is conceived for 'BYE' message. This is done by using the Maria's host addresses in 'Via' header holding a new transaction identifier. This request is initiated by Maria which is reflected in 'To' and 'From' headers. James can now close the specific media session by using the same local, remote tags and 'Call-ID'. Observe that the value of the entire branch IDs presented so far starts with the string z9hG4bK, which is a special string and is calculated by using strict rules defined in [RFC 3261]. Hence, can be used as a transaction identifier.

The final message in Figure 2. is '200 OK' which is a confirmation response to 'BYE':

SIP/2.0 200 OK

Via: SIP/2.0/UDP low.energy.org:5060;branch=z9hG4bK392kf ;

received=200.203.205.207

To: James Franklin <sip:james@test-sip.org>;tag=76341

From: Maria Anderson <sip:maria@energy.org>;tag=a53e42

Call-ID: 123456789@lab.test-sip.org

CSeq: 1 BYE

Content-Length: 0

This response is an echo of the original 'CSeq' request (1 BYE) which ensures the successful tear down of the call.[3][4]

2.2.4 SIP Call with Proxy Servers

In the previous example, James was aware of the IP address of Maria and sent the 'INVITE' message directly to her. This is not possible in actual practice because the IP addresses are often dynamically assigned. For instance, when you dial-up to an Internet service provider (ISP). An IP address is assigned to your PC with the help of DHCP (Dynamic Host Configuration Protocol), this address is only valid until you disconnect your modem and is different for every time you established a new connection. Even for an “always on” connectivity, a different IP address can be assigned each time you reboot your system. The IP address assigned to your PC does not uniquely identify you, but identifies a node of the physical IP network you are connected with. This entails an unique address is obviously required to reach you, which can identify you no matter where you are and how you are connected.

SIP uses an email-like system independent host name to reach a specific user and does not require any particular IP address to be used. This allows UAs to send and receive SIP messages, irrespective of their IP addresses. To achieve this goal, SIP uses an

addressing scheme called URIs. These URIs are also capable of handling telephone or fax numbers, transport parameters, and some other parameters. A SIP URI can be defined as a name which can be resolved to an IP address to reach a particular user. This is done by using SIP proxy servers and DNS lookups. Similar to proxy in HTTP, a SIP proxy server can not

establish or tear down any media session instead it is used to receive and forward messages on behalf of UAs. Only one proxy server is presented in figure 3. but there can be many proxy servers involved in a signaling path.

Figure 3. illustrates the example of a typical SIP call by using SIP Proxy Servers. In this example, the transaction starts with James's softphone sending an 'INVITE' request destined to Maria's SIP URI. As Jame's softphone was not aware about the IP address of Maria, a SIP proxy server is used. The process starts with a DNS lookup of Maria's SIP URI domain name (bth.se) that gives the IP address of the SIP proxy server (karlskrona.bth.se), the 'INVITE' message is sent to the SIP proxy Server. The proxy server then found Maria's IP address in its database and forwarded the message directly to her with an additional header holding the address of the SIP Proxy Server. This can be defined as a two-step process:

- DNS lookup by UAs to locate the IP address of the proxy server
- Database lookup by proxy server to locate the IP addresses of UAs

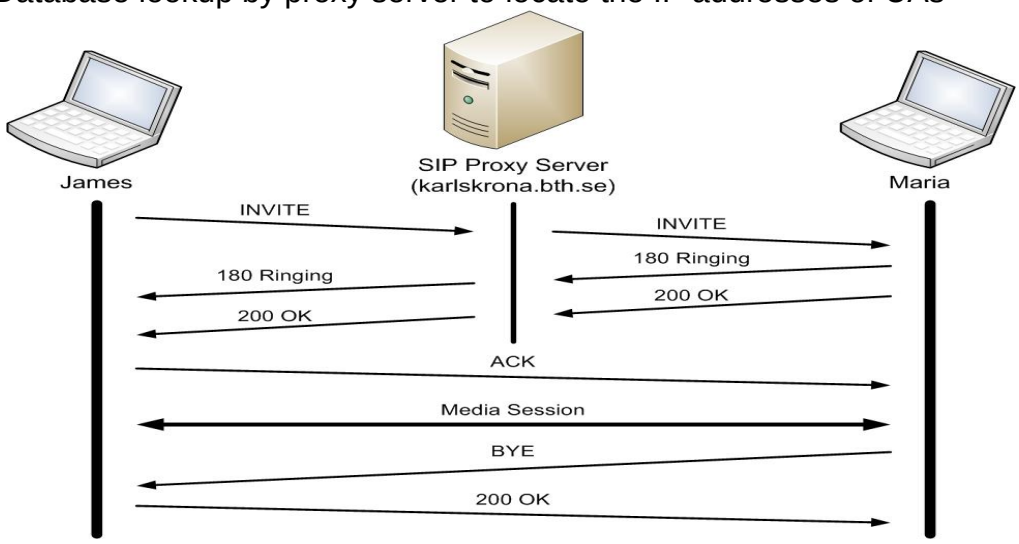


Figure 3. SIP Call Flow with a Proxy Server [3]

The Presence of IP address of SIP Proxy Server also helped Maria to learn that 'INVITE' has been relayed through a proxy server. In response to the received 'INVITE', Maria will

then send the '180 Ringing' to the proxy server. The proxy server received ‘180 Ringing’ and forwards it to James after dispatching the first the header that holds its own address (karlskrona.bth.se). The proxy server then also forwards the '200 OK' message to James as soon as Maria has accepted the call, after removing the additional header once again. The presence of the Contact header field with the SIP URI address of Maria in the '200 OK' will then allow James to send the 'ACK' directly to Maria, bypassing the proxy server. Finally, the media session is closed when Maria sends a 'BYE' message to James. [3][4]

2.2.5 The SIP Registration Process

In Figure 4, the user James sends a SIP 'REGISTER' request to a registrar server. The 'REGISTER' request contains the SIP URI address of James along with the 'Contact' URI which holds the information about his current device and location (IP address). The SIP registrar server uses this information to route the SIP request by updating it to the database of proxies. The 'Contact' URI is then stored and acknowledged by sending a '200 OK' response message to James. This response message returns the current contact information of James with a registration expiry parameter that indicates the time duration of registration validity. If the registration is required beyond that specified period, James must have to send a new 'REGISTER' request before that expiry time. This Registration is normally updated automatically on initialization of a new SIP device and at preferred frequent intervals decided by registrar server. More than one device can also be registered itself against a single SIP URI. In this case, a proxy may forward the request to either one or more devices. Additional register functions can also be realized by using clear registrations or by retrieving the list of currently registered devices. [3][4]

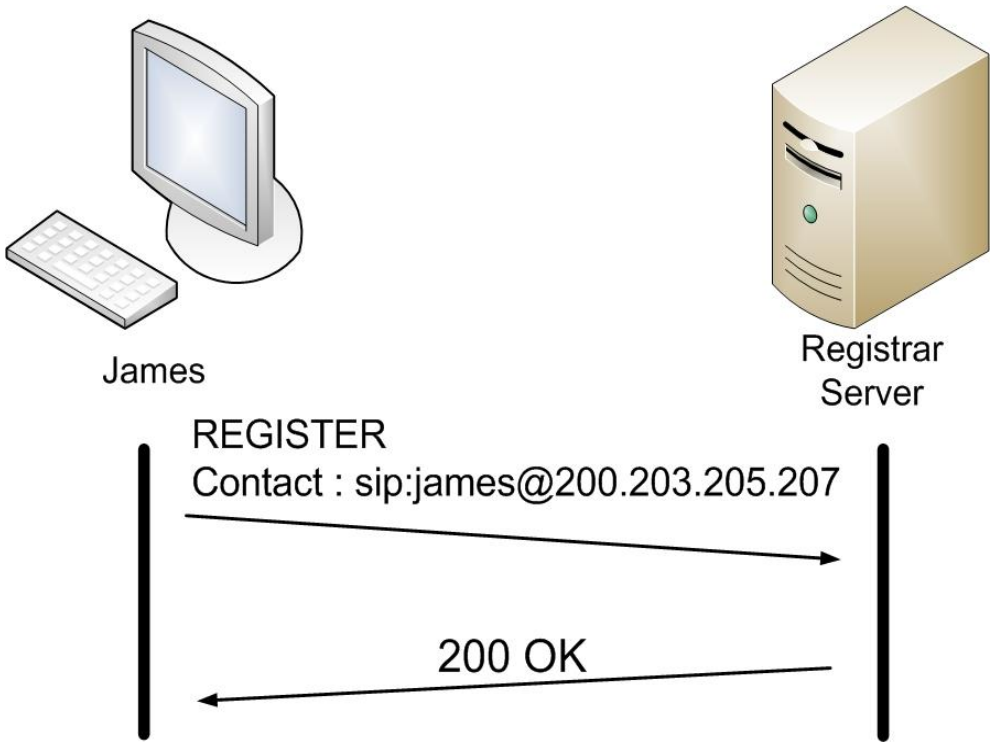


Figure 4. SIP Registration Example [3]

2.2.6 Instant Messaging with SIP

Example in figure 5, illustrates the process of how SIP can work to provide multimedia services such as presence and instant messaging applications. The information related to the state of a user or device at a specific instant, can be described as presence information. This type of services are generally provided in instant messaging applications such as MSN messenger or Google talk, which includes the information about whether a specific user has signed in or not, whether he is active, or idle or away. To accomplish this task, a presence protocol is involved to set up subscriptions or long-term relationships between devices about exchanging status information. However, the actual information exchanged and how it can be presented to a user is application dependent.[3][5][6]

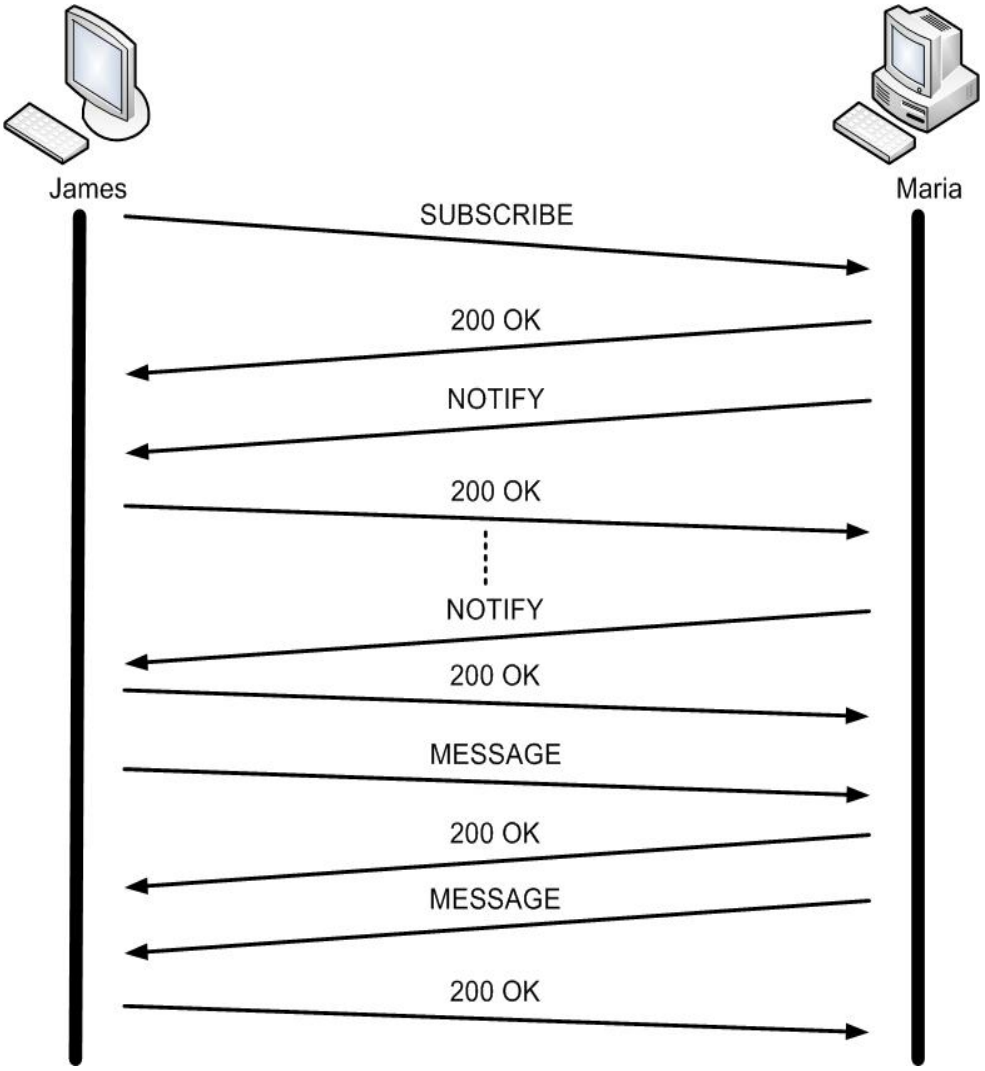


Figure 5. Presence and Instant Messaging Example [3]

As shown in Figure 5, the SIP protocol uses two special type of messages (SUBSCRIBE and NOTIFY) to exchange the presence information. The 'SUBSCRIBE' message is used to retrieve the current status or presence updates from the presence server while 'NOTIFY' performs the delivery of that information to the requester. SIP presence uses the SIP events extensions and instant messaging extensions defined in [RFC 3265] and [RFC 3428] respectively. In this message flow example, James subscribes to Maria's presence information by sending a 'SUBSCRIBE' message to inquire about the current status of Maria. Maria accepted the request by responding back to James with '202 ACCEPTED' message. As there is no proxy server involved, the actual subscription is started when James first received a 'NOTIFY' message. This 'NOTIFY' message holds

the current status (signed-in) of Maria. James then sends a '200 OK' response message to the 'NOTIFY', in order to confirm that he has received the requested notification. Later on, whenever Maria goes off-line, the information will be sent in a new 'NOTIFY' message with change in status. The process goes on and on until the subscription expiration time has reached. Now, as James is aware that Maria is on-line, he can send instant messages to her by using the contact information in the 'NOTIFY' message. Sending instant messages by using the 'MESSAGE' method in SIP is similar to page messaging and is not part of the dialog. Maria replied to the message received from James with a '200 OK', which is also sent outside the dialog.[3][5][6]

2.2.7 Message Transports in SIP

SIP is an application layer signaling protocol which requires use of some standard protocols for the transmission of actual media stream. RFC 3261 has defined the use of TCP and UDP for transport. An extension document RFC 4168 also defined that how SCTP (Stream Control Transmission Protocol) can also be used.

2.2.7.1 UDP Transport

Each SIP request or response message is carried in a single UDP datagram or packet. For messages that exceeds the limit of maximum transmission unit (MTU), there is a compact form of SIP which uses a mechanism to represent common header fields in a short form. The source ports are used from the list available port numbers (usually above 49172) or the default SIP port (5060) can be used. A packet or datagram along with a SIP message can be lost in transmission because of the connectionless nature of UDP transport. However, the UDP checksum ensures discard of erroneous datagrams. The destination port for both request and reply is mostly set to '5060', or the port number listed in the headers.

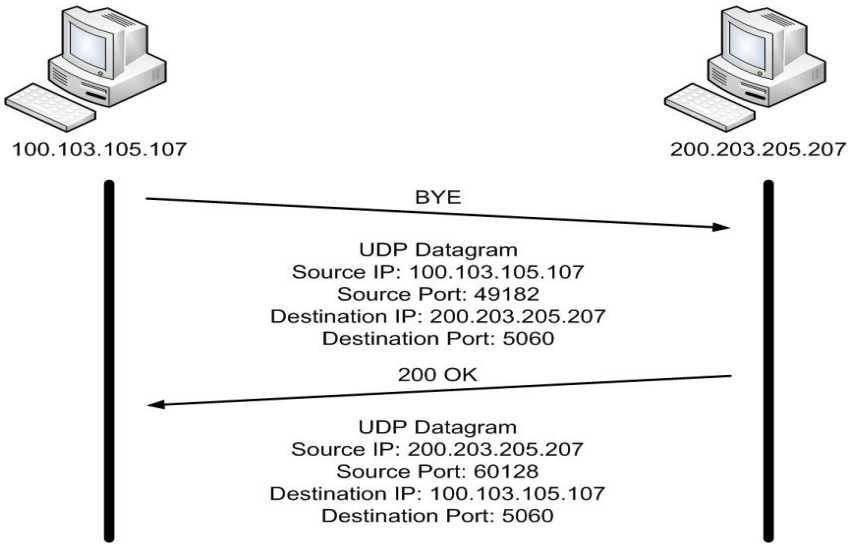


Figure 6, Transmission of SIP messages with UDP [3]

The simple operation of UDP provides an easy way of transport for both UAs and servers, which enables them to work properly without taking too much considerations for transport layer state. UDP does not offer any congestion control mechanism which can result in lost

packets and retransmissions will be required. Heavy packet loss can also push the link into congestion collapse. UDP can not be utilized for SIP if the size of message is larger than the Maximum Transmission Unit (MTU) of the IP network. This means that large SIP messages with multiple message bodies and larger header size must be realized with TCP because SIP can not support fragmentation at application layer. [3][4]

2.2.7.2 TCP Transport

TCP ensures reliable transport for SIP. But complexity and delays are increased as compared to UDP, which can not be avoided for real time transmissions such as voice and video. TCP is utilized for transport of SIP messages in Figure 7. The message exchange starts with an 'INVITE' message sent by an UA at 100.103.105.107 to a SIP redirect server at 200.203.205.207. The SIP redirect server checks its record for the destination address and returns that address in the form of redirection class '302 MOVED' response. The '302 MOVED' is then recognized by the UA with an 'ACK' message. The

next step would be to re-send the 'INVITE' to the address returned by the redirect server which is not shown in this example. As in the previous example with UDP, the destination port number used is '5060' and the source port is picked up from the list of available port numbers. A TCP connection must be established between two end points before any SIP message exchange begin. This is illustrated as a a single arrow in figure 7, but is actually a typical TCP three-way handshake. When the TCP connection is commenced, the messages are sent in the stream. When TCP or another stream-based protocol is utilized with SIP, the 'Content-Length' header must be required in all request and responses because it is used to find out that where one message ends and the next starts on. A new TCP connection is established in reversed direction by the redirect server to send the '302 Moved' on port '5060' as destination port. The connection is then closed by sending an 'ACK' is sent in the TCP stream used for the INVITE. Because this concludes the SIP session. Whenever a TCP connection is closed during a dialog, a new can also be opened to send a request within the dialog. such as a 'BYE' request to tear down the media session. TCP provides reliable delivery of packets and congestion control, and can also transport SIP messages of varied sizes. But with the price of increased amount of delay and complexity. [3][4]

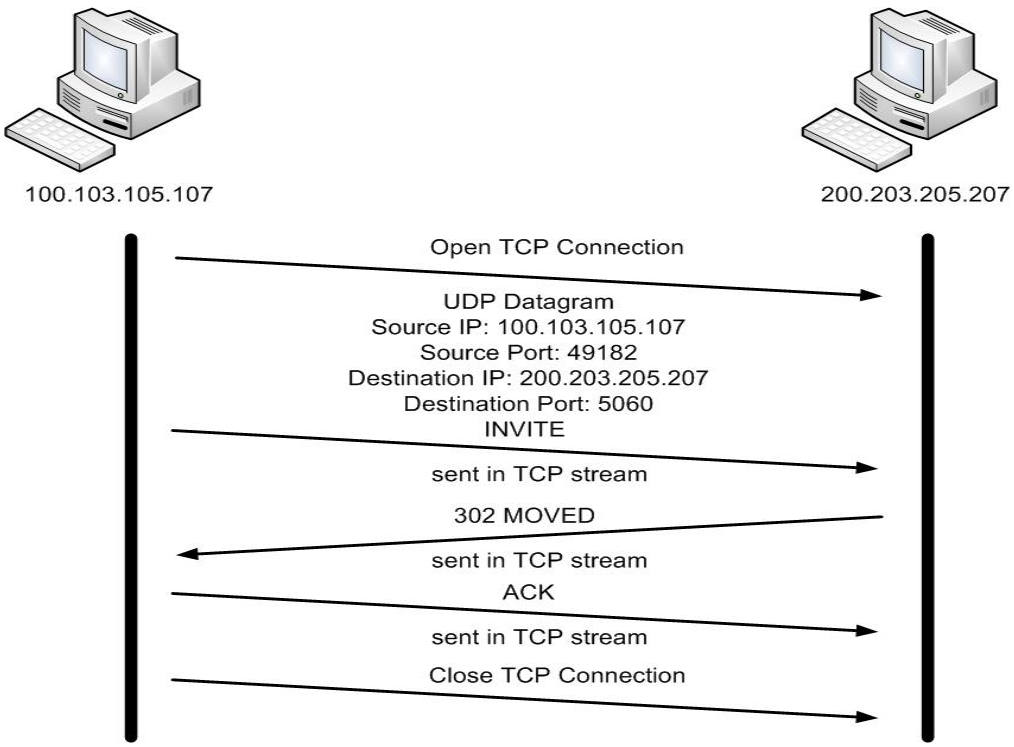


Figure 7. Transmission of SIP Messages with TCP [3]

2.2.7.3 SCTP Transport

The use of Session Control Transport Protocol (SCTP) for SIP transports is described in RFC 4168. SCTP can be used for reliable delivery of SIP messages with number of potential benefits over UDP and TCP:

- Fast Retransmission – The speedy packet loss can be determined by using SCTP for SIP transports. This is done with the help of SACK and the faster delivery of SACK messages in case of any failure, which results in faster detection of lost SIP messages as compared to UDP. TCP also has a similar SACK mechanism with retransmit operation.

- Congestion Control – SCTP manages congestion control by maintaining the aggregate rate of message exchange between two SIP entities which adds up a significant advantage over UDP, which has a less effective congestion control mechanism. However, the similar performance can be realized by establishing number of parallel connections with TCP.

- Transport-Layer Fragmentation – Both TCP and SCTP ensures fragmentation at transport layer by breaking up a large message (bigger than MTU size) into multiple smaller packets. In UDP fragmentation is used at IP layer which increases the probability of packet loss and make it difficult to implement NAT and Firewall. This capability becomes more crucial when the size of SIP messages increases exponentially.

- Message Based Transaction - The message based nature of SCTP enables the identification of different signaling messages at transport layer which lacks in stream based TCP. As TCP only can understand bytes and the assembling of received bytes is done at application layer, the SCTP gains advantage of delivering a message to the application when it arrives and a single message loss does not pose any problem to deliver the other messages correctly. However, if SIP entities are participating in many parallel transactions, SCTP can not ensure better performance as compared to UDP.

- Multihoming: SCTP supports multihoming which means that multiple IP addresses on the same host can be used in a single SCTP connection. This feature makes it possible to migrate from an IP address to another in case of any failure. It is realized when a network failure due to an unavailable interface of a sever does not leads to total service collapse.

It is crucial to observe that the most of the advantages of using SCTP for SIP happens under packet loss conditions. In no loss circumstances, SCTP provides performance in parallel with TCP. However, More research is still required to assess that under what loss conditions the throughput and improvements in setup time can be determined. [7]

2.3 Threats to SIP Security

Almost all the entities (i.e., proxy servers,registrar servers and end-user devices) of SIP based VOIP Networks are potential targets for standard Internet attacks as well as some more types of attacks that are unique to SIP. For instance, SIP Registration Server can not challenge the authenticity of an UA which can cause registration hijacking. This can be done by altering the 'from' header of a SIP registration message, an intruder can portray as a valid UA and can register himself. An Attacker can also launch dictionary attacks to retrieve the password of an UA, and can be able to perform redirection,eavesdropping or monitoring of multimedia sessions. Any SIP server can be duplicated by sending the similar message responses to a soft-phone, the attacker can forward incoming calls to himself. This enables an attacker to track the call and to perform selective DoS attacks.

Fake Proxy Server that is capable of reading and modifying the message header and body can be used to tamper SIP messages, DoS attacks can also be launched by redirecting the messages to non-existing server. A malicious user can forward 'BYE' and 'CANCEL' messages to close down the SIP sessions.[1]

Security threats to SIP based VOIP networks and their impact on overall SIP security will be briefly discussed in the following section.

2.3.1 Registration Hijacking

Registration hijacking takes place when an attacker portray as a genuine UA to a registrar server and substitutes the registration with its own address, which can cause all the incoming traffic for a particular user to be directed to the attacker.[4][8]

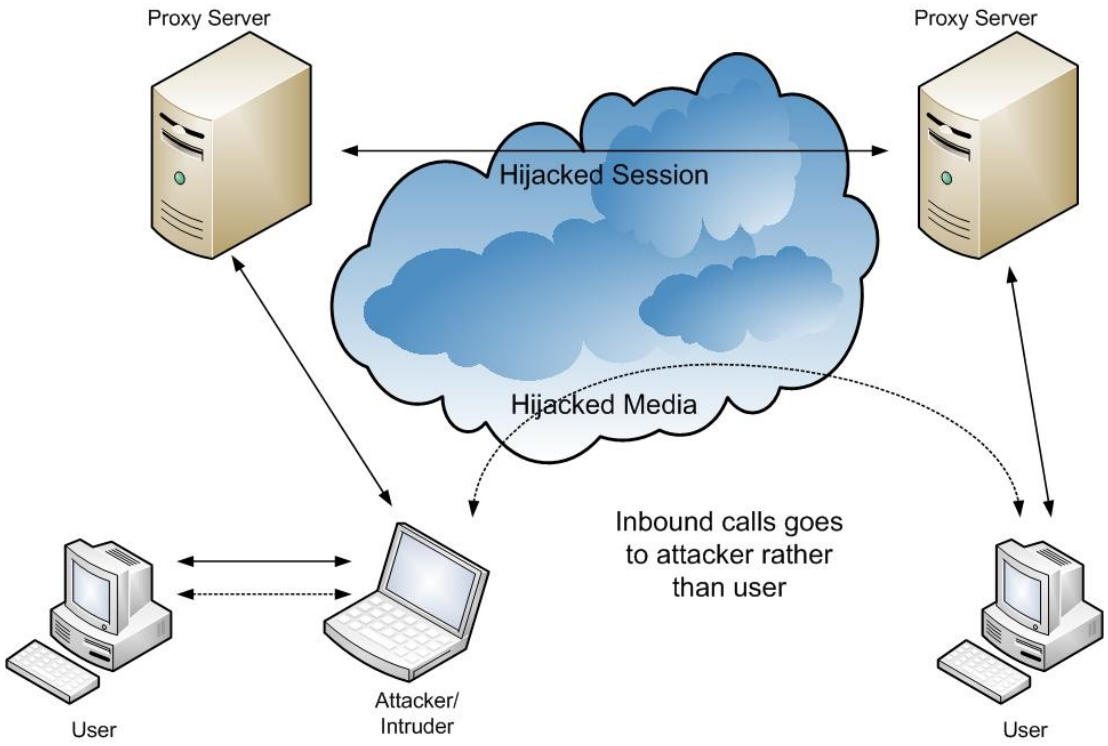


Figure 8. Registration Hijacking In SIP [8]

Registration is mostly done by utilizing UDP for transport, which allows easy message spoofing of SIP requests. Authentication is not present in many SIP based Networks and if present is only requires user-name and password, which can be killed by dictionary attacks. In dictionary attack, the intruder with the help of one of your user-names, can be able to generate a list of most likely passwords based on their information about your organization. The directory of addresses can also be created by scanning for your valid UA addresses. The list of extensions can be developed by using the SIP 'OPTIONS' message to generate a directory of your users. In some cases, organizations may need shared passwords such as the extension number or name with an extra word. These type

of passwords can be retrieved or guessed by an attacker who already has learned one of your passwords. Failed registration requests are not always logged in by registrar servers which opens many possibilities for attackers. SIP proxy servers usually do not notice directory scanning and registration hijacking attempts. Registration hijacking can cause complete service collapse for a genuine UA, which may be a user with soft-phone or a vital resource such as a media gateway, an Automated Attendant (AA) or an Interactive Voice Response (IVR) facility. The attacker can gather critical data such as authentication or key signaling information and can puzzle an authentic user to drop a message by acting as a Voice Mail system. He can also be capable of performing the Man-In-The-Middle (MITL) attacks, in which he sheerly stands between the calling and called party and can be able to retrieve and alter both signaling and media information. The redirection of an inbound call to a media gateway causing toll fraud is another possibility. [4][8]

2.3.2 Impersonating a server

The Request-URI usually determine the domain to which a SIP request is destined for and a server (Proxy or Registrar) in that domain is directly contacted by UAs to deliver the request. However, there is an opportunity that an attacker can act as a remote server, and the UA's request can also be intercepted by some other party. For instance, consider that two domains namely 'karlsrona.com' and 'ronneby.com' are present in a SIP based network and the redirect server at karlskrona.com impersonates as a redirect server at ronneby.com. In this case, when a request sent by an UA to ronneby.com could reached to redirect server at karlskrona.com and responded with proper header fields which resembles the response from ronneby.com. This can maneuver the UA to unsafe resources and can also block the way for the incoming requests to ronneby.com.

User Agents, proxies and registrars usually communicate with each other by using UDP for transport and does not often exchange any strong authentication parameters such as validation certificate or encrypted keys. Therefore, a rouge proxy or registrar server can be introduced in many ways which includes Domain Name Service (DNS) spoofing, Address resolution Protocol (ARP) cache spoofing and by simply modifying the proxy address for a SIP soft-phone. This can leads to full control over calls or sessions and opens the door for similar attacks as for registration hijacking. If redirection of outgoing

calls to a specific domain (karlskrona.com) is done by DNS spoofing, all the outbound calls to that domain can be controlled by means of interception, manipulation, blocking, conferencing or recording. In ARP cache spoofing, an attacker can mislead a user to establish sessions with a rouge proxy server on the internal network. And If accomplished, all the sessions initiated by the UA can be controlled. [4][8]

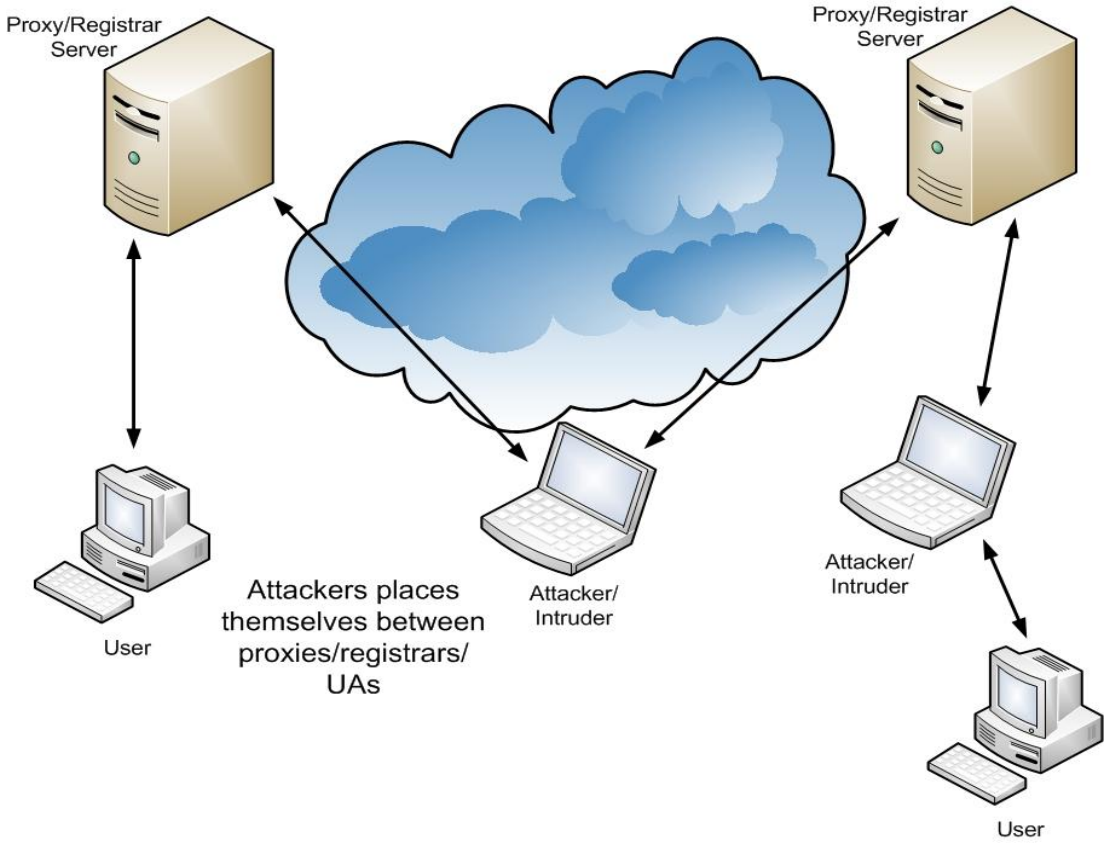


Figure 9. Impersonating a SIP server [8]

2.3.3 Message Tampering

When a malicious user or an intruder grabs and manipulates the SIP messages, it leads to the phenomenon of message tampering. It can takes place by means of registration hijacking, proxy/registrar impersonation, or through an assault on any trusted entity such as a media- gateway or a firewall.

Route requests for SIP UAs are directed by trusted proxy servers. No matter how that trust is formed (authorization or authentication method), a proxy server may trusted by an

UA, but not to perform any inspection or manipulation with the SIP message bodies. For Instance, an UA may utilize the message bodies to exchange session encryption keys for a media stream. Even though, the user trusts the proxy server of a particular domain to deliver the SIP message appropriately, but do not wish the operators of that domain to decrypt any accompanying media session. In the worst scenario, a malicious proxy server can modify session key or security parameters requested by initiator (UA).

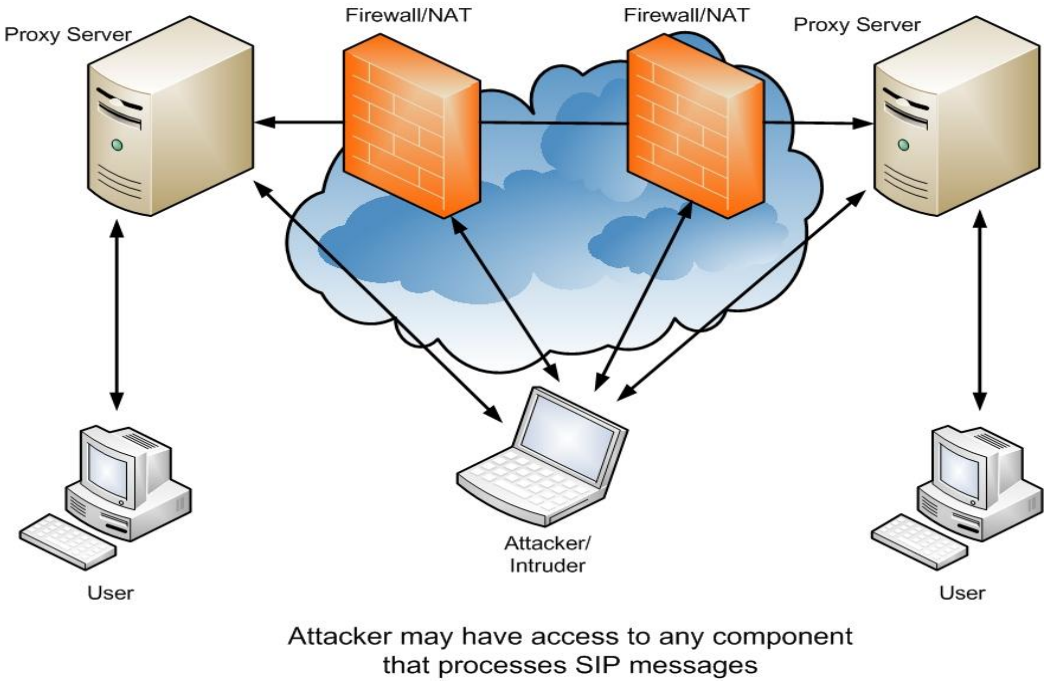


Figure 10. Message tampering scenario [8]

Also, some headers in SIP messages are of significant importance. For example, an UA may be careful of the 'subject' header which can reflect the information about the containing body or media session. However, as proxy servers often modify SIP messages to route the packet properly, not all headers can remain secure end-to-end. For the above discussed reasons, the UA may require secure end-to-end transmission of SIP message bodies and the headers as well in some events.[4][8]

2.3.4 Session Tear Down

When a dialog or SIP session is commenced, succeeding requests can be sent to change the status of the session. It is crucial for the participants to make sure that those requests are not intercepted by attackers.

Session tear down can happen if an attacker is able to send modified SIP 'BYE' messages to the participating UAs after noticing the previously sent messages. This can be accomplished by capturing some initial messages exchanged between the two UAs and through learning session related parameters such as 'To tag', 'From tag'. In this way, the attacker can craft the forged request such that it appears to come from one of the participants. Once the 'BYE' message has reached its destination, the session is pull down immediately. Similar mid-session threats can also be realized by the transmission of properly shaped re-INVITEs that can modify the session parameters most probably to weaken the session security or to redirect the media streams.

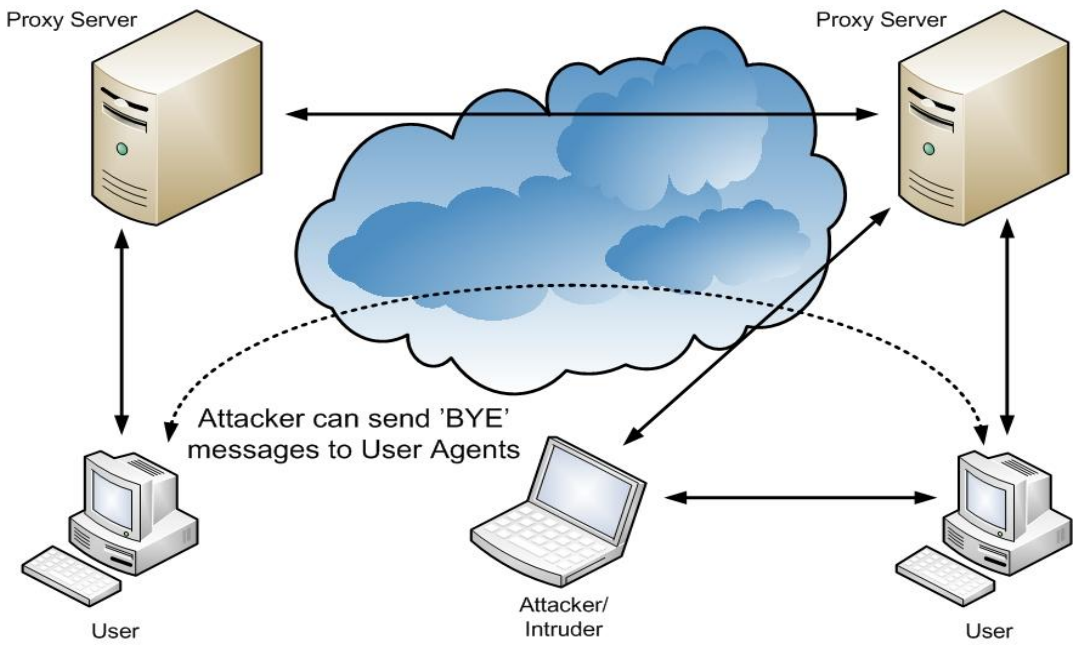


Figure 11. Session tear down by an intruder [8]

If the intruder is aware of the address of a participating media gateway or an Interactive Voice Response, he may send 'BYE' messages which can results in session tear down. Another case of session tear down can be the excessive delivery of 'BYE' messages to a firewall which in turn can close all the UDP ports opened for authorized sessions. A proxy

server may not know about the sessions being closed and will not have appropriate session records is another downside of session tear down. [4][8]

2.3.5 Denial of Service (DoS) Attacks

Denial-of-service attacks are attempts to make a specific SIP network entity unavailable. It is usually done by sending excessive number of packets or traffic to a network interface. It can be realized in the form of modified SIP packets, spoofed SIP states, and by means of flooding SIP request messages such as a “REGISTER” or an “INVITE” message.

An attack which is initiated by a network user and can cause many network hosts to flood a particular host with large amount of network traffic is referred as distributed Denial of Service (Dos) attack. In most of the cases, SIP proxy and registrar servers confront public Internet to accept requests from worldwide users which opens many chances for distributed denial-of-service attacks. Research has already proved that many SIP based networks are extremely prone to these kind of attacks. [4][8]

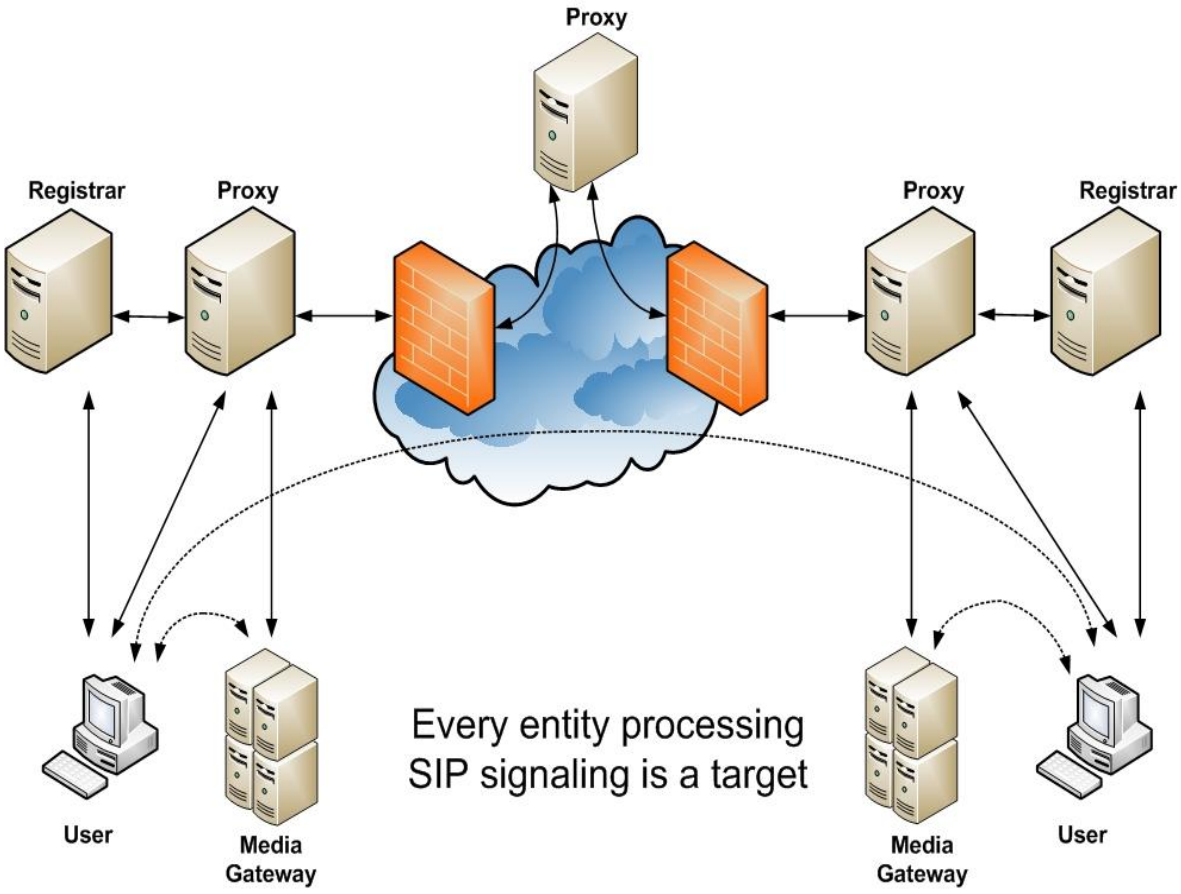


Figure 12. Denial of Service in a SIP network [8]

Attackers can also introduce fake SIP requests which comprises of an altered source IP address and a matching 'Via' header that can be used to identify a targeted host as the initiator of the request and hence can be directed to many SIP network entities, in this way SIP UAs or proxy servers can be cleverly utilized to direct the large amount of denial-of-service traffic towards the target. In the similar fashion, attackers can manipulate Route header field values of a request and then can be able to forward those messages to a forged proxy server that can further amplify messaging sent to the target. The similar consequences can also be realized in case of Record-Route, if the attacker is sure that the SIP dialog geared by the request can lead in number of transactions generated in the reverse direction. Also, unauthenticated and unauthorized 'REGISTER' requests by registrar servers can lead to denial-of-service. This can allow attackers to unregister some or all users in a specific domain which in turn can forbid the users to establish new sessions. The utilization of multi-cast for SIP messages can also introduce number of opportunities for DoS attacks. [4][8]

3 SIP with Firewall/NAT Traversal

The main challenge for SIP based VOIP communication is the Traversal of the Session Initiation Protocol (SIP) and its established sessions with Network Address Translators (NATs). At the same way, many private networks are secured by firewalls employing NAT (Network Address Translation). Establishing SIP sessions with users on private address space (NAT) or with users behind a firewall poses some special problems:

- Network Address Translation does not allow initiating any SIP sessions from public Internet to hosts on a private network.

- The source addresses and port numbers are encapsulated in SIP messages at the application level and are changed by NAT only in IP and TCP/UDP headers which leads to the discarding of messages by the SIP client.

- Many SIP messages can also be blocked by means of port filtering by NAT as SIP uses different combinations of port numbers each time it establishes a new session.

SIP opens many challenges for research which can be resolved in several ways:

- Protocol extensions and modifications
- Modifications for NAT traversal
- Methods and techniques which enables SIP protocol to explore the ways to discover NAT existence or to totally avoid the NAT traversal.[9]

3.1 Firewalls and Network Address Translators (NAT)

3.1.1 Internet Firewalls

"A firewall is a device with two interfaces- one on the "inside" and one on the "outside". Its function is generally to protect devices on the inside from those on the outside, and to sometimes prevent users on the inside from connecting to, or accessing services on the outside."[10]

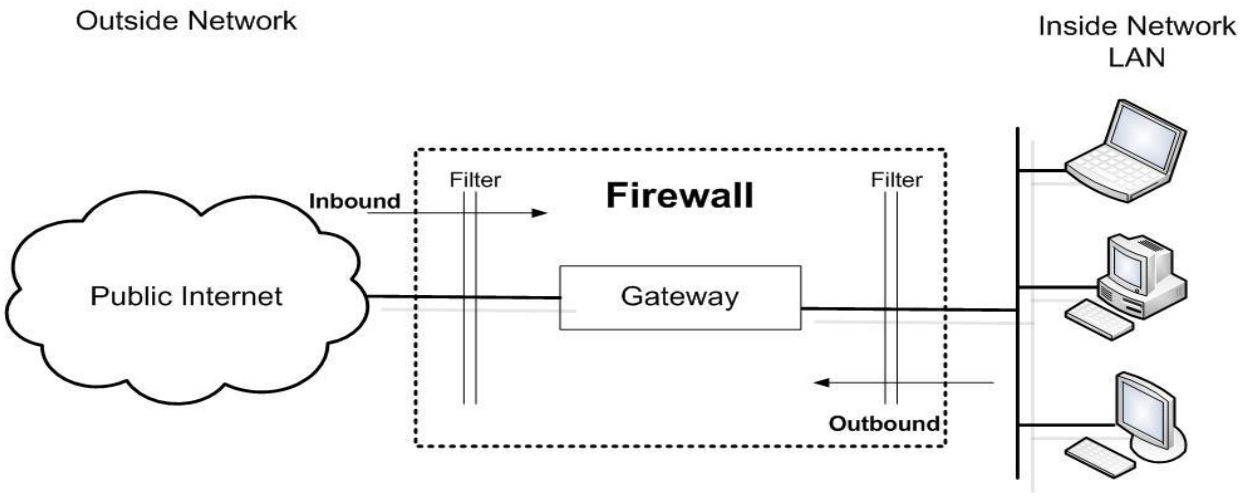


Figure 13. Basic Operation of a Firewall

Firewalls are fundamentally built to protect the access from public Internet to the hosts on a private Network (LAN) while still allowing the users on the LAN to enjoy the benefits of the Internet. This is achieved by applying inbound and outbound rules on the firewall interfaces, entailing that what kind of traffic is permitted to passed through across its interfaces.

3.1.1.1 Packet Filtering Gateways

Packet filtering gateways are of two main types, those that do keeps the record of what was happened previously (stateful) in a session and those who do not (Stateless). In both of the cases, filtering works on the several headers of IP datagrams, i.e. the TCP header, the UDP header and the IP header. The most significant information required for filtering are the IP addresses and port numbers. Other useful data such as the ACK and SYN flag

in the TCP header are of great importance as well. They are used to differentiate between the various stages of a session, i.e. the start, end and an established connection. Figure 14. shows that how a TCP connection is established and tear down. When two hosts set up a TCP connection, a three-way handshake model is used while four TCP segments are required to close the connection.

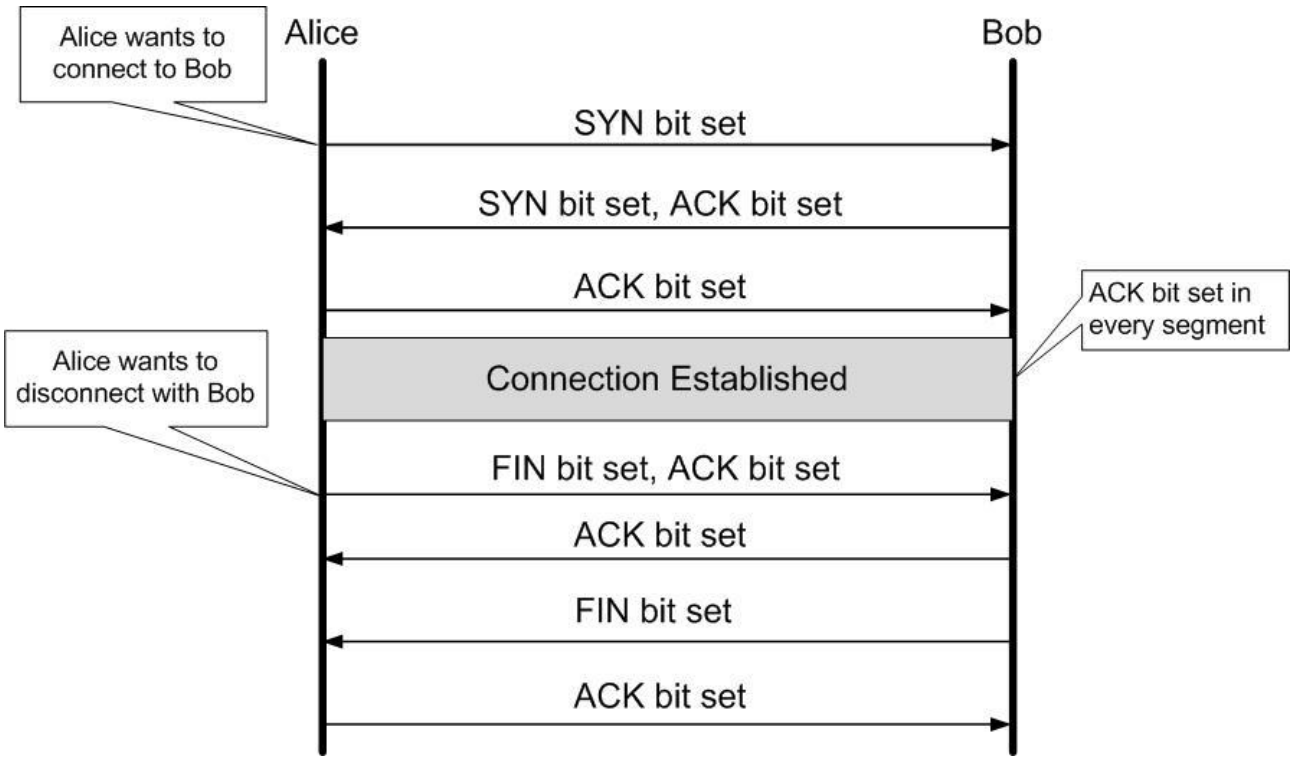
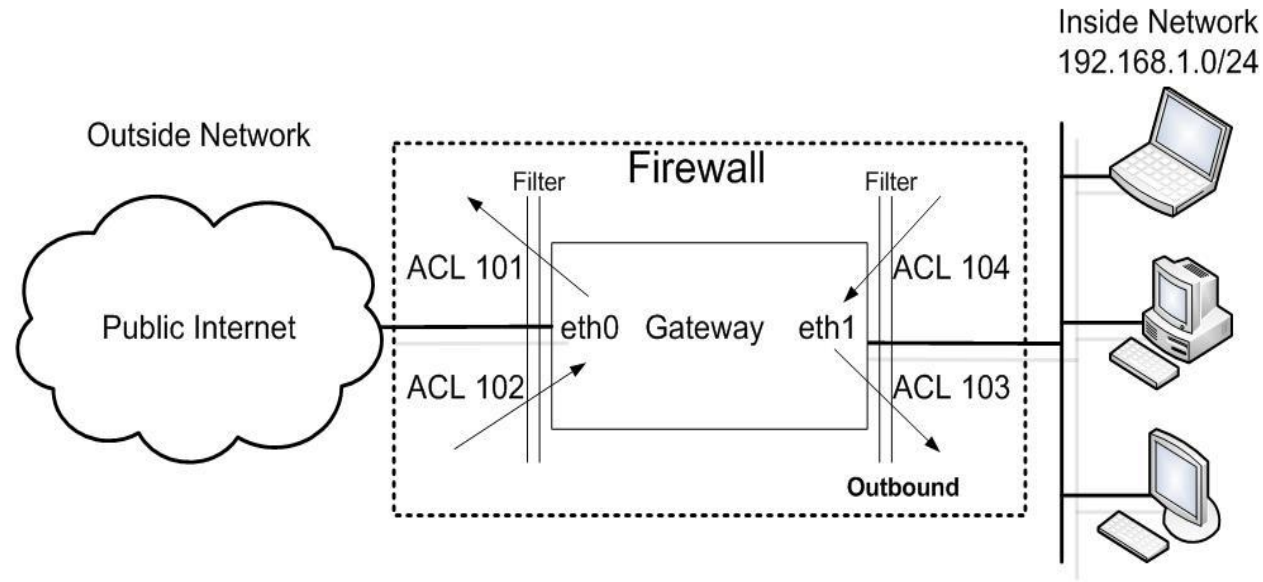


Figure 14. TCP Connection Establishment and Tear Down

The filtering is done in stateful filters by using the information from the application part. This entails that stateful filters are able to realize the application protocols without being looking at whether a specific packet is intended to use a service with a well-known port number or not. The down side is that the packet filtering gateways do not have the capability to alter the application data of the IP datagrams.

The rules and functions for filtering can be applied on every interface and in each direction (inbound or outbound). The rules are also count on the direction of a particular packet, i.e. whether is going towards gateway or departing the gateway. Figure 15. illustrates that how rules can be implemented in Cisco IOS and Linux. The rules are

enclosed in Access Control Lists (ACL) in Cisco IOS while ipchains is the tool for writing rules in Linux. The example in figure depicts that the hosts on the internal network (192.168.1.0/24) are granted access to establish TCP based connections with Web Servers (port 80) by using their web browsers (on port numbers greater than 1023).



CISCO IOS

ACL	Applied to	Direction	Permit/Deny	Protocol	source	port	destination	port	option
101	eth0	out	permit	TCP	10.1.1.0 0.0.0.255	gt 1023	any	eq 80	
102	eth0	in	permit	TCP	any	eq 80	10.1.1.0 0.0.0.255	gt 1023	established
103	eth1	in	permit	TCP	any	eq 80	10.1.1.0 0.0.0.255	gt 1023	established
104	eth1	out	permit	TCP	10.1.1.0 0.0.0.255	gt 1023	any	eq 80	

Linux

```
ipchains –i eth0  -l output  -j ACCEPT  -p tcp  -s 10.1.1.0/255.255.255.0  1024:  -d 0.0.0.0/255.255.255.255  80
ipchains –i eth0  -l input   -j ACCEPT  -p tcp  -s 0.0.0.0/0.0.0.0          80     -d 10.1.1.0/255.255.255.0  1024:  ! –y
ipchains –i eth1  -l input   -j ACCEPT  -p tcp  -s 0.0.0.0/0.0.0.0          80     -d 10.1.1.0/255.255.255.0  1024:  ! –y
ipchains –i eth1  -l output  -j ACCEPT  -p tcp  -s 10.1.1.0/255.255.255.0  1024:  -d 0.0.0.0/255.255.255.255  80
```

Figure 15. Cisco IOS Access Control Lists and Linux ipchains

3.1.1.2 Circuit-Level Gateways

As the name suggests, this type of firewalls works with the virtual circuits and hence can only be utilized for TCP (connection oriented and packets in sequence) based application level protocols i.e. telnet, FTP etc. Support for UDP is under process.

The circuit level gateways are used to relay TCP connections from one side of the firewall (internet) to the other side (LAN). They operate in a way such that a connection has been established by an internal client to a TCP port on the gateway, which then leads to make a connection (TCP or telnet) with the external server. The content of application part of the TCP segment is not interpreted in this exercise and only the relaying of information from one side to another is made possible. [11]

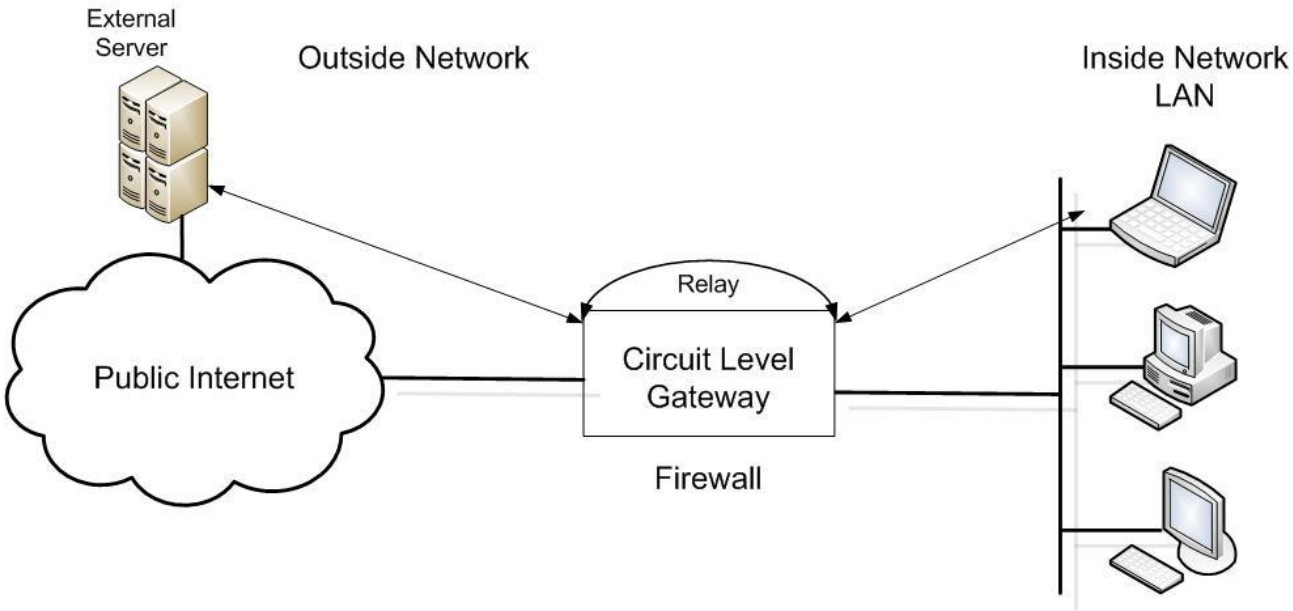


Figure 16. Circuit level Gateways

The addresses of clients on the LAN will not be included in the IP header because they are replaced with the address of the interface present on the public (Internet) side of the gateway. Therefore, port numbers in TCP header are replaced as well.

3.1.1.3 Application Level Gateways

The most intelligent type of firewall is an application-level gateway (ALG). It operates in the similar way as a circuit level gateway but has some key differences. In short, an ALG is described as a program working in coordination with a firewall and performing Network Address Translation (NAT). Some useful features of an ALG are as followed:

1. The special purpose code is utilized by an ALG to support each and every specific service (application). As it realizes the relaying application protocol, a higher level of security can be attained.
2. The ALGs can not only support connection oriented protocols (TCP or telnet), instead they are equally capable of supporting UDP based protocols i.e. TFTP.
3. If an ALG is implemented along with NAT, then it is capable of analyzing the application data for LAN addresses and can substitute it with firewall's external interface address. This practice also enables ALG's support for protocols such as FTP, DNS, and SNMP. [12]

3.1.2 Network Address Translators (NAT)

"Network Address Translation is a method by which IP addresses are mapped from one realm to another, in an attempt to provide transparent routing to hosts. Traditionally, NAT devices are used to connect an isolated address realm with private unregistered address to an external realm with globally unique registered addresses."[13]

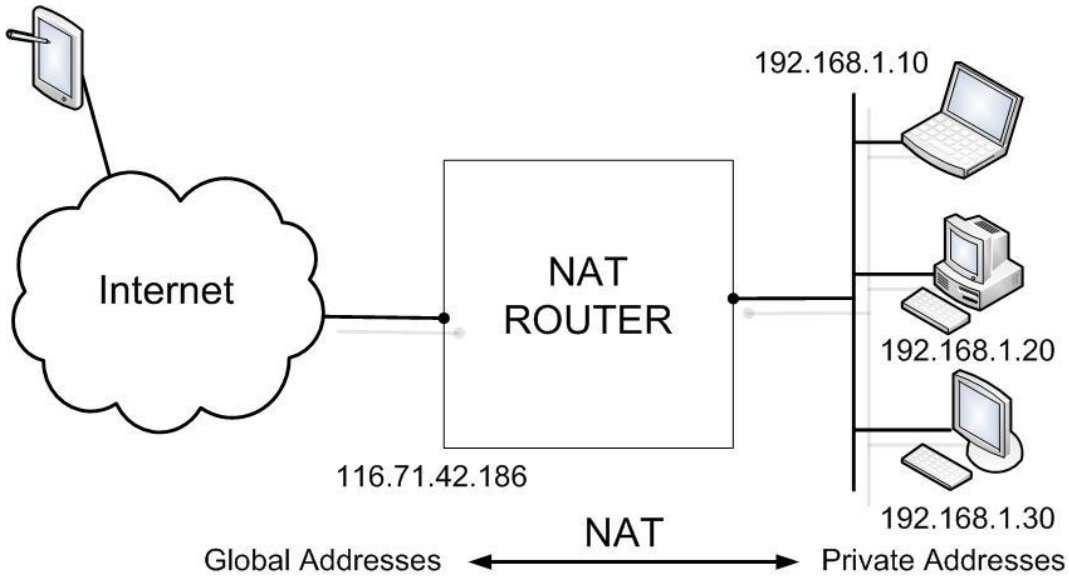


Figure 17. Network Address Translation

NATs are lively agents in the data route, normally embedded in a router or a gateway, whose key role is to reserve IP addresses to be shared between numerous devices. This clearly separates a network into two distinct parts: Inside (private/hidden) - Outside (public Internet). NATs administer every IP packet sent from public internet to local LAN or from local LAN to public internet and makes a decision (based on its IP routing list) to forward it with or without modification, or to discard it. They behave like firewalls as being network topology aware, but unlike firewalls and routers, they are capable of altering the packet before forwarding it. [14]

3.1.2.1 NAT Operation

NAT operation is based on replacement of destination IP address (public) with an ethernet (LAN) address and recalculation of IP and TCP/UDP header checksum, for an incoming packet. For an outgoing packet, NAT replaces the source IP address (LAN) with the address of router or firewall's external interface (Public) and re-computes the checksums of IP and TCP/UDP header.[9]

To establish a session between an external and an internal host, NAT firewall/router makes an entry in its translation table. This is done to ensure that the incoming packets

from one side are rightly mapped to the destined hosts on the other side .i.e. to perform the translation smoothly.

NAT Table	
Internal IP	External IP
192.168.1.1	136.202.148.34
192.168.1.2	136.202.148.32
.	.
.	.
.	.
.	.
.	.

Figure 18. NAT Table

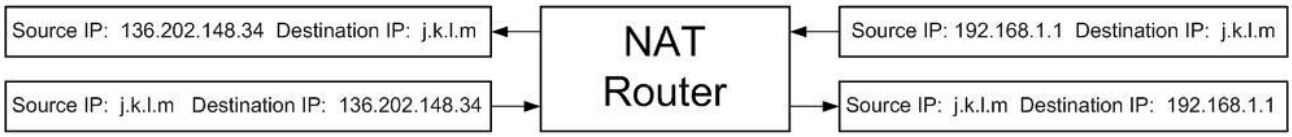


Figure 19. Network Address Translation in practice

3.1.2.2 Basic NAT Traversal

Public addresses in NAT are mapped in a pooling manner. When a packet is sent from an inside host (using LAN IP) to an outside host (using public IP), a public IP address is assigned from the list of available addresses and hence mapped with the address of inside host (entry saved in NAT Table). This new mapped address is now utilized for all the future packets exchange between the two hosts.

When NAT receives an incoming packet from public internet, it examines the destination against the IP mapping table. If an entry is found in the table, the destination address of packet is replaced with matching IP, and the packet is then transported accordingly. The packet will be discarded if NAT does not found a mapping entry.

NAT mapping entries (named as bindings), are produced dynamically and are preserved based on a timer. If no packets arrive for a definite period of time (timer expires), the

particular IP address (for which a binding is reserved in the NAT table) is set free for future use. [9]

3.1.2.3 NAPT Traversal

NAPT (Port Translating NAT) is an evolved and most commonly used version of NAT. It works on the same principles as the basic NAT traversal do but differ in a way that the established bindings are grounded on both the IP addresses and the TCP/UDP port numbers. The bindings are preserved in the NAT table by using the same type of timers.

The beauty of this solution is that many hosts can use fewer external (public) IP addresses. [14]

NAPT uses port numbers as the key reference for address translation instead of IP addresses. The number of connections each IP address can have is up to 2^16 = 65536 (number of ports available). This represents only the theoretical number as various port numbers are allotted to some specific services i.e. telnet, http, TCP/UDP, FTP etc. When a port number is utilized for a translation, the TCP/UDP port numbers are noted as used to make the things simpler. [14]

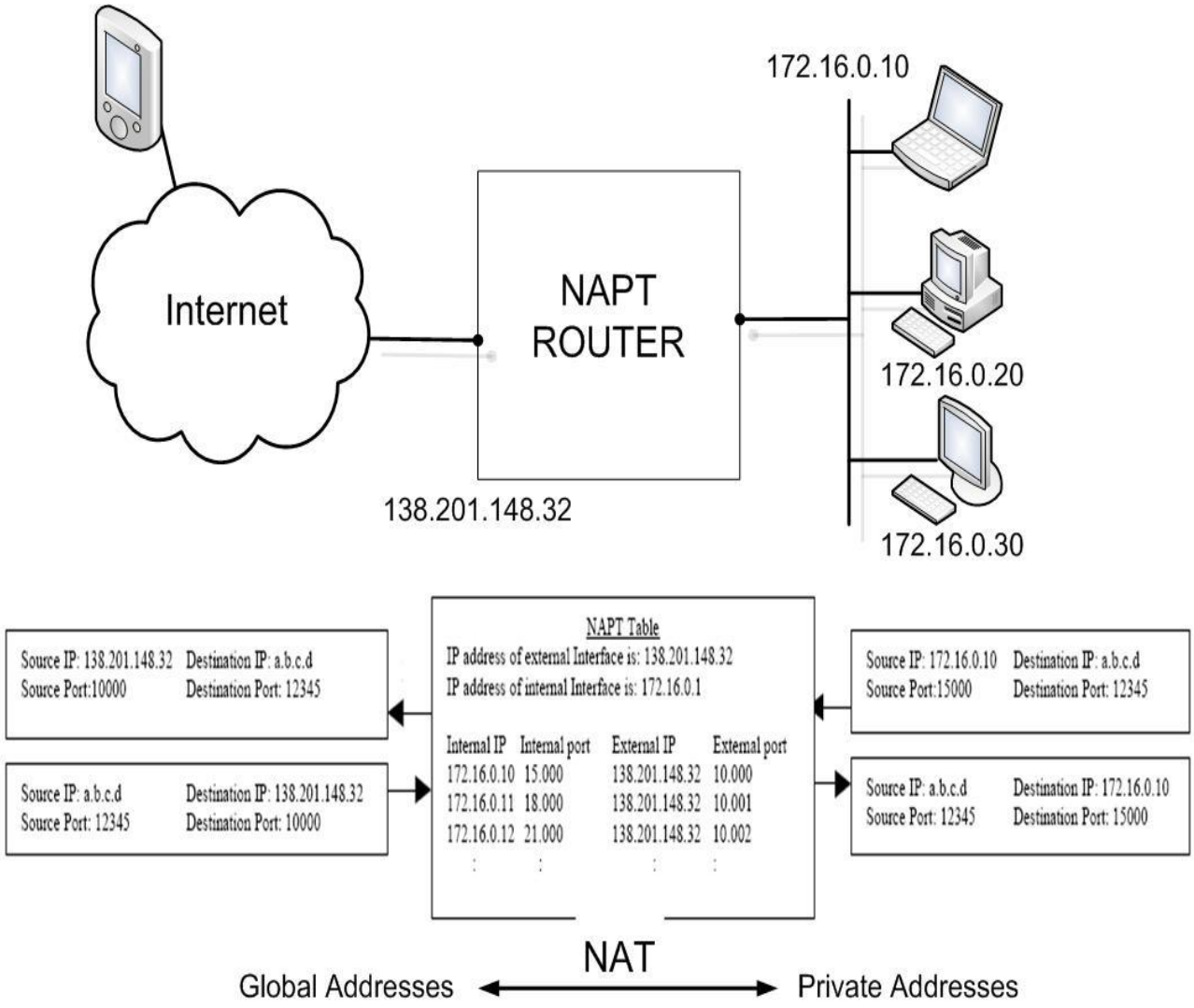


Figure 20. Network Address and Port Translation in practice

3.1.2.4 NAT behaviors and classifications

Due to of lack of standardization, vendors/manufacturers are free to design NATs that may distinct from one another. The difference in design and functionality can not only depend on vendor-to-vendor basis but it could also differ on model-to-model or case-to-case basis for the same vendor. The basic behavioral difference between NATs comprises in a manner in which the binding is performed. [14]

3.1.2.4.1 Symmetric Binding

In symmetric binding, the first outgoing packet actually triggered the mapping of the destination address and it stays unaltered till the timer expires (lifetime of binding). This mapping session is settled on 5-tuple state which includes the LAN IP address and port number, the destination address and port number, and the UDP/TCP protocol. This is termed as the most protective kind of NAT. [9]

3.1.2.4.2 Full-cone Binding

The full-cone binding works in way that an outside host and a port number can utilize the binding of a LAN address and a port number with a public address and port number. It is the least protective NAT in behavior. [9]

3.1.2.4.3 Restricted-cone Binding

The binding in this type of NAT can only be utilized by an outside host. This is done by using any port number for packet sending. [9]

3.1.2.4.4 Port-restricted-cone Binding

The binding in this type of NAT is also can only be utilized by an outside host as it is done in the restricted-cone. But the source port number has to remain same as it is used by the initiator of the binding. [9]

Many NATs are also designed in a way that they can behave differently for TCP and UDP packets and are more complicated in nature. They deal with a TCP packet in symmetric mode and an UDP session in full-cone mode. Those kinds are beyond the scope of this research and hence are not considered in the discussion. [9]

3.1.2.5 Conclusions related to NAT

The behavioral analysis of NAT is *very* vital:

"A NAT has no standard way in which to advertise its presence, nor does it have any standard way to advise protocols and applications of the particular behaviors it applies to packets being passed through the NAT". [14]

3.2 NAT/Firewall Traversal with SIP: ISSUES

The NAT/Firewall Traversal for SIP based communications can be realized in two parts:

- NAT traversal with SIP signaling
- SIP related media with NAT traversal

3.2.1 NAT Traversal with SIP signaling

SIP signaling has some issues with NAT traversal:

- **NAT changes the source port in request message:** The default process for SIP with UDP packets comprises of request messages induced by SIP soft phones (User Agent Clients) and followed by the response messages generated from the SIP registrar/proxy Servers (User Agent Servers). The source address and source port is included in the response message form the SIP ‘via’ header of the request message (‘received’ parameter). The local port (source specified port in request message) has been replaced with an external port by NAT. The SIP proxy severs usually sent the original port numbers to the SIP clients after processing the request messages. The NAT blocks the original port used in the response messages because of the port mismatching.

- **Inbound signaling from public network:** When an IP client is located behind a NAT and the SIP registrar/proxy is on public internet, after the process of registration, any incoming messages from SIP registrar/proxy are blocked by NAT. This is the case because there isn’t any type of mechanisms defined in the SIP

protocol specifications for new generated requests in opposite direction to utilize the same address and port numbers present in the initial requests (registration).

- **Binding timeout:** Keeping a connection alive by means of any kind of request/reply mechanism or by using a time-stamp is not yet included in SIP protocol specifications. This can cause the termination of a previously established connection (through NAT) after a definite period of time (binding timeout); hence it is not possible for the SIP client to entertain any future calls. [9]

3.2.2 SIP related media with NAT traversal

The most favorite media transport protocol used for SIP based communication is RTP (Real Time Protocol). Another prominent protocol which is utilized for the negotiations of the RTP protocol parameters is named as SDP (Session Description Protocol). The address and port numbers of any SIP client are contained in the SDP part of the SIP protocol messages when RTP is utilized for SIP media. As address and port numbers specified are private in nature, the NAT can block the incoming messages because of address and port mismatching. [9]

3.3 NAT/Firewall Traversal with SIP: SOLUTIONS

3.3.1 Symmetric response

The extension to SIP protocol formulated for figuring out the issue of ‘source port changes in SIP messages by NAT’ in SIP signaling is Symmetric response. This method induced a new parameter in ‘Via’ header termed as ‘rport’ which enables a client to suggest the address and port number for a server to send the response messages rather then to deduce the address and port from IP/UDP headers. This technique enforce that the participating SIP entities must be capable of listening and sending SIP request/response messages from/on the same port. [15]

3.3.2 Connection Re-use

This solution can enable inbound requests/calls from public internet to be directed to a SIP client beyond a NAT is termed is Connection re-use. This method is ground on the principles of address and port discovery and re-use and can be done by making a “connection-tuple”. [16]

3.3.3 Symmetric RTP

In this method a SIP based entity can send and receive RTP media traffic on the same address and port number. The address and port number defined in SDP header can be neglected by SIP proxy/registrar on public network and hence address and port numbers used as source from IP/UDP headers for RTP. Symmetric RTP is only used where one of the SIP entities (proxy server or UA client) is known to be on the public internet i.e. not behind a NAT. [17]

3.3.4 Simple Traversal of UDP through NAT (STUN)

The mechanism for dealing with NAT discovery behavior is called STUN (Simple Traversal of UDP). This method also provides the detailed information about external address and port utilized by NAT for binding process. A named STUN server and external server on public internet must be used for discovery. STUN requests are utilized to define response address, changed port and changed address whereas STUN responses are consists of source address, mapped address and changed address.

By sending a combination of distinct requests to a STUN server, a SIP client can realize that whether it is on public network or behind a NAT. [18]

3.3.5 Traversal Using Relay NAT (TURN)

TURN (Traversal Using Relay) NAT is a protocol based NAT which is only can be implemented with symmetric NAT. It enables a SIP entity to get incoming TCP/UDP packets behind a symmetric NAT. The TURN server on public internet provides a public address to the SIP client and behaves like a port-restricted NAT. The public address

provided by the TURN server is called as the transport address, and allows directing the TCP/UDP packets through the NAT for that specific connection. [19]

3.3.6 Interactive Connectivity Establishment (ICE)

It is a complex mechanism which combines the functionalities of both STUN and TURN and offered a unified solution for NAT traversal. ICE (Interactive Connectivity Establishment) incurs typical set of address-port combinations by employing STUN and TURN. These combinations are enclosed in the SDP messages by realizing a new parameter termed as ‘alt’. A unique ‘alt’ parameter is inserted for each and every address-port combination. The use of these kinds of combinations provides SIP adaptability for NAT behavior. [20]

3.3.7 Application Layer Gateway (ALG)

ALG (Application Layer Gateway) are NATs based on a SIP proxy/registrar server or an ALG server. It provides the way to alter the packets ‘on-the-fly’ by overriding the NAT rules and policies. An ALG is a weak node in a network in terms of security and needs alterations in both NAT and SIP protocol. When a modification takes place in NAT or SIP protocol the ALG must be upgraded or altered.

3.3.8 Middlebox/MIDCOM (Middlebox Communications Protocol)

A special kind of ALG is a MIDCOM architecture. It works in way such that signaling functions are separated logical from media transport functions within a middlebox.

“A middlebox is defined as any intermediate, device performing functions other than the normal, standard function of an IP router on the datagram path between a source host and a destination host.” [21]

To permit the media traversal directed through NAT, MIDCOM architecture uses a proxy server equipped with the MIDCOM protocol to communicate with the middlebox NAT. [21]

3.3.9 Universal Plug and Play (UPnP)

The UPnP (Universal Plug and Play) solution is designed for home or office use by Microsoft which provides a method for SIP applications to observe and set up the NAT behavior. In order to utilize this mechanism, the NAT must have to be UPnP enabled. The down side of this method is the lack of any security mechanism. This method imposes many key-holes on NAT to public network with the active dominance of an application which is an obstinate to various security policies.

3.3.10 Session Border Control (SBC)

“In its simplest form a Session Controller enables interactive communication across the borders or boundaries of disparate Internet Protocol (IP) networks. In doing this, Session

Controllers connect islands of IP voice and/or video traffic without requiring all IP traffic to first be converted into TDM at a handoff point between networks. Session Controllers operate at Layer-5 of the network and work with - but don’t replace - devices such as softswitches, NAT devices and firewalls.” [22]

A SBC ensures an authorized connection from the outside (public internet) to inside (private network) by collaborating with the firewalls which overcomes the ‘inbound signaling from the public network’ problem. It executes some of the NAT based functions without interfering with the normal operations, the address and port alteration occurs only for a particular SIP connection while leaving the rest of the traffic being under the control of NAT. The two basic entities of a SBC implementation are:

- **SBC signaling server :** enables SIP signaling between SIP proxy/registrar server and clients behind the NAT. It is organized to act as an active pass through point for SIP signaling messages and accommodates complete transparency over call establishment and control functions. It also controls the keep-alive timers for SIP register update to confront with the “binding timeout” issue. The SBC signaling server performs the following functions:

The process of SIP user registration; The alteration in SIP headers ('contact' and 'via' header), which ensures proper processing of SIP messages; The alteration in address and port numbers to allow NAT traversal; The traffic management and synchronization of SBC media server for reliable packet transfer; Successful DNS resolution for SBC servers.

- **SBC media Server**: It works under the supervision of the SBC signaling server and performed as an active pass through point between SIP clients for reliable delivery of RTP and RCTP packets. In order to permit NAT traversal for media utilizing NAT's pin-holes, it changes SDP parameters without being intrusive to NAT security policies. The SBC media server enables full transparency and control over media packets for each and every SIP connection. It can also execute the functions of dynamic NAPT by concealing the network entities and topology. [22]

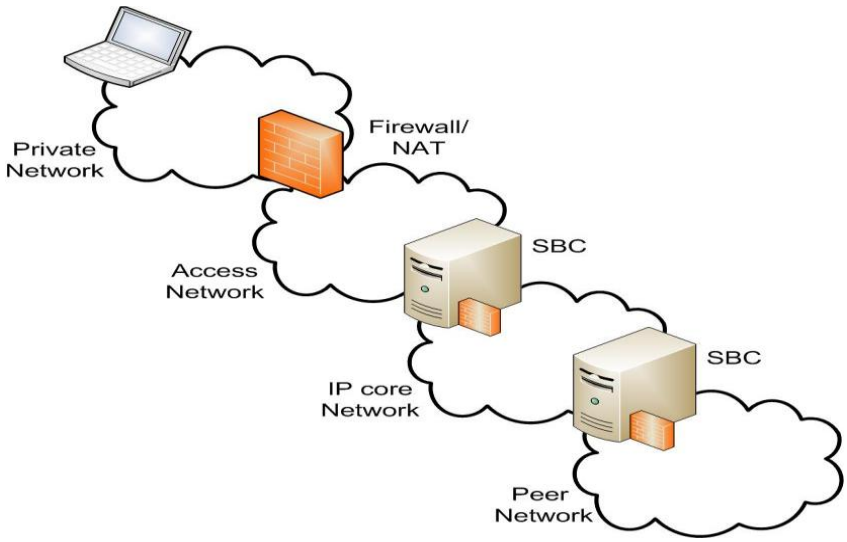


Figure 21. A scalable VOIP network architecture with SBCs

A SBC media server with full visibility and control over various kinds of media sessions can be easily implemented for a scalable VOIP network architecture without being changing the existing equipment while permitting many types of call related functions such as billing management and QoS (Quality of Service).

4 Model Test Scenario

Discrete event based simulations for our proposed SIP based VOIP environment was performed on combination of different software, Emulator and Virtual machines.

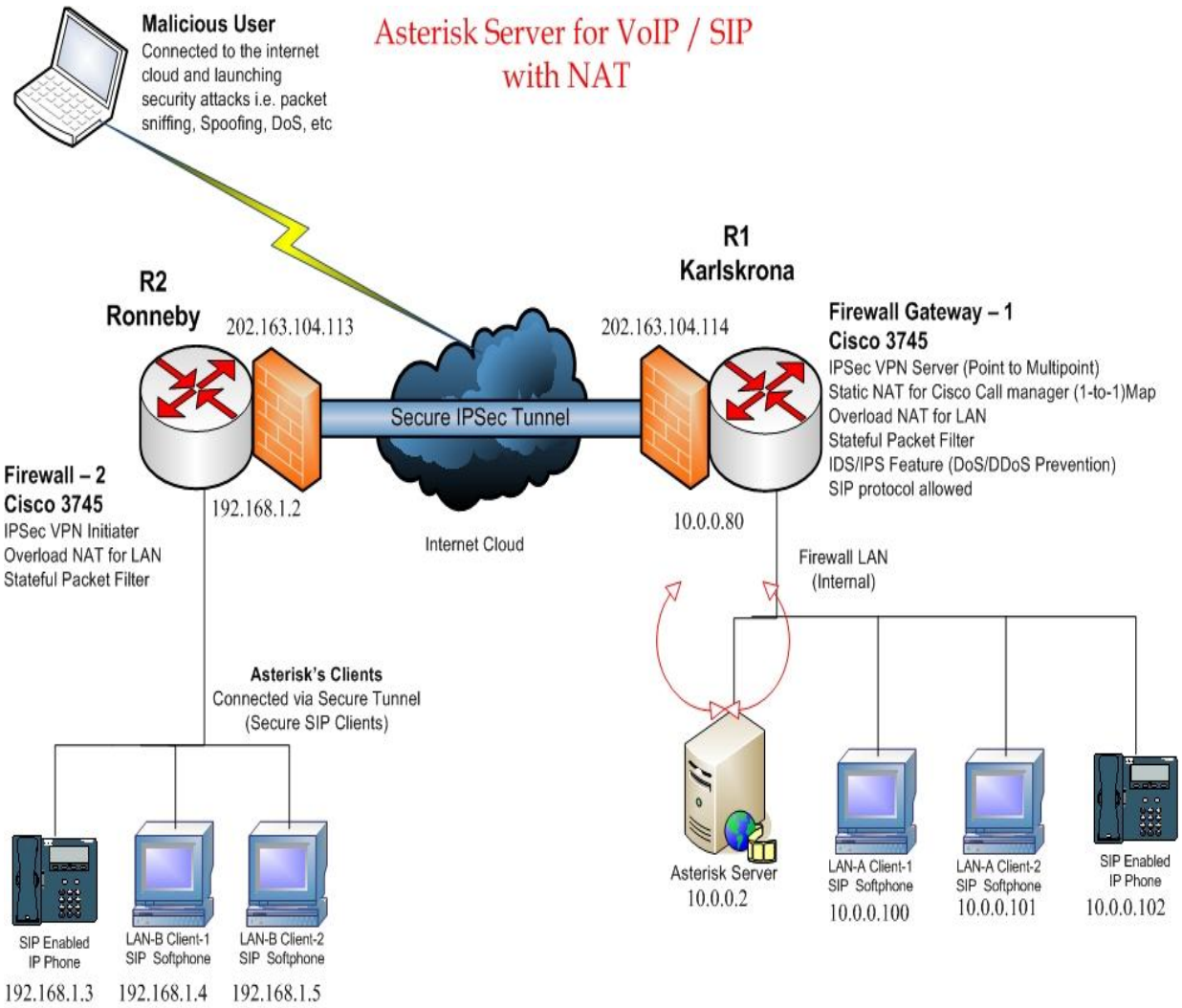


Figure 22. Detailed Network Diagram

Routing					
	Interface	**IP Address**		**IP Address**	**Interface**
R1	Fa0/1	10.0.0.80	R2	Fa0/0	192.168.1.2
R1	S0/0	202.163.104.114	R2	S0/0	202.163.104.113
R1	Lo1	1.1.1.1	R2	Lo1	2.2.2.2

Dynamips / Dynagen

Topology	GNS3
Router IOS	Cisco 3745 Series

Virtual Machines/OS

Windows XP	
Ubuntu-9.04	
Asterisk-1.4.21.2	

IOS features used

IPSec VPN Tunnel
Access Control Lists
static NAT
Threat Protection/IPS

Voice

Asterisk IP PBX
X-lite Softphone users

4.1 Virtualization

Virtualization is a proven software technology that is rapidly transforming the IT landscape and fundamentally changing the way that people compute. Today's powerful x86 computer hardware was designed to run a single operating system and a single application. This leaves most machines vastly underutilized. Virtualization lets you run multiple virtual machines on a single physical machine, sharing the resources of that single computer across multiple environments. Different virtual machines can run different operating systems and multiple applications on the same physical computer.

VMware software provides a completely virtualized set of hardware to the guest operating systems. VMware software virtualizes the hardware for a video adapter, a network adapter, and hard disks. The host provides pass-through drivers for guest USB, serial, and parallel devices. In this way, VMware virtual machines are become highly portable between computers, because every host looks nearly identical to the guest. In practice, a system administrator can pause operations on a virtual machine guest, move or copy that guest to another physical computer, and can resume the execution exactly at the point of suspension. Alternately, for enterprise servers, a feature called 'VMotion' allows the migration of operational guest virtual machines between similar but separate hardware hosts sharing the same storage. Each of these transitions is completely transparent to any users on the virtual machine at the time it is being migrated.

We have used VMware throughout this project to install, configure and connect the multiple entities, which consists of two emulated Cisco 3640 series Routers, a Asterisk based VOIP server and three SIP soft-phone users. This whole setup was installed and configured on three physical machines.

4.1.1 OS level

Windows XP professional®

Windows XP professional® is used as Host Operating System, and Ubuntu 9.04 is installed as VMware based virtual machine in order to install and configure the Asterisk based VOIP server on it. Windows XP® was also used on two different physical machines

for the emulated routers, firewalls, GNS3 and soft-phone users throughout the whole experiment.

Windows XP is the most widely used desktop computer operating system, it is easy to operate and supports a huge variety of softwares; which make it acceptable to a network of a few nodes to a large corporate network having thousands of nodes including mobile workstations.

Ubuntu-9.04

UBUNTU is an enterprise class open source debian based Linux distribution which is widely used in production environments. It does not require licensing for its software updates and technical support. Support is rendered through official emailing lists, internet forums, and chat rooms or paid support can also be possible by contacting Canonical Inc.

Asterisk-1.4.21.2

In 1999 Mark Spencer created a piece of software, known as Asterisk which is an emulation in software for Telephonic Private Branch Exchange (PBX). Asterisk is a software based IP PBX system, through which a softphone phone can make calls to other softphones, and can connect to the Public Switched Telephone Network (PSTN) and VoIP network.

Asterisk works with SIP, MGCP, H.323 and IAX2 protocols. Integration with conventional telephony protocols such as ISDN and SS7 is also supported.

It can be deployed on numerous UNIX family operating systems i.e. NetBSD, OpenBSD, FreeBSD and Solaris.

Asterisk can also provide the VoIP services to GSM and CDMA based cellular networks. Asterisk makes use of the internet to make cheapest long-distance calls (toll bypass). Voicemail, Call conferencing, Interactive Voice Response, Call Queuing, Caller ID, etc.; are the services provided by Asterisk system. However, we have only used Asterisk to make SIP based voice calls in this test lab.

4.1.2 Network Level

The network topology was designed on GNS3. Multiple instances (IOS) of Cisco 3745 series routers were executed in this test lab scenario with the help of Dynamips and Dynagen.

GNS3

GNS3, an acronym for Graphical Network Simulator-3 which is used to emulate the real Networks, by utilizing computer processor to act as a Network device's processor. It works along with Dynamips and Dynagen. It can emulate Cisco Network devices using Cisco IOS as their OS. The latest IOS for a Cisco device might be tested for its new characteristics and the configurations that one wants to do on actual routers can also be tested and verified by using GNS3. It can be used on different operating systems such as Windows, Linux, UNIX and Mac OS as it is an open source software for Router Simulation**.**

Cisco IOS

Cisco IOS is the Operating System for Cisco Network devices that exploits the set of hardware available on the device. c3640-jk9o3s-mz IOS is used for Cisco 3640 Router The Cisco 3600 Series is a family of modular, multiservice access platforms suitable for medium and large-sized organization, and smaller Internet Service Providers. With over 70 modular interface options, the Cisco 3600 router family provides solutions for data, voice video, hybrid dial access, virtual private networks (VPNs), and multiprotocol data routing. The high-performance, modular architecture protects customers investment in network technology and integrates the functions of several devices into a single, manageable solution.

Cisco IOS has a characteristic command line interface (CLI), whose style has been widely copied by other networking products. The IOS CLI provides a fixed set of multiple-word commands — the set available is determined by the "mode" and the privilege level of the current user. "Global configuration mode" provides commands to change the system's configuration, and "interface configuration mode" provides commands to change the

configuration of a specific interface. All commands are assigned a *privilege level*, from 0 to 15, and can only be accessed by users with the necessary privileges. Through the CLI, the commands available to each privilege level can be defined.

In this project, we have also utilized the Firewall and IPSec functionality of Cisco 3600 series platform in addition to the routing capability, we've used 3745 to perform routing infrastructure having IOS of C3745-JK.BIN. The above said IOS has advance Security image with VPN support.

Dynamips

Dynamips is an emulator program which was written by Christophe Fillot to emulate Cisco routers. Dynamips can run on Linux, Mac OS or Windows and can emulate the hardware of the Cisco series routing platforms by directly booting an actual Cisco IOS software image into the emulator. It allows users to build complex network topologies to test the functionality of IOS on a desktop PC, without the need of an actual physical Cisco device. Dynamips currently supports different networking medium such as Ethernet, Serial link, ATM, and POS interfaces for the 1700, 2600, 3600, 3700, and 7200 hardware platforms.

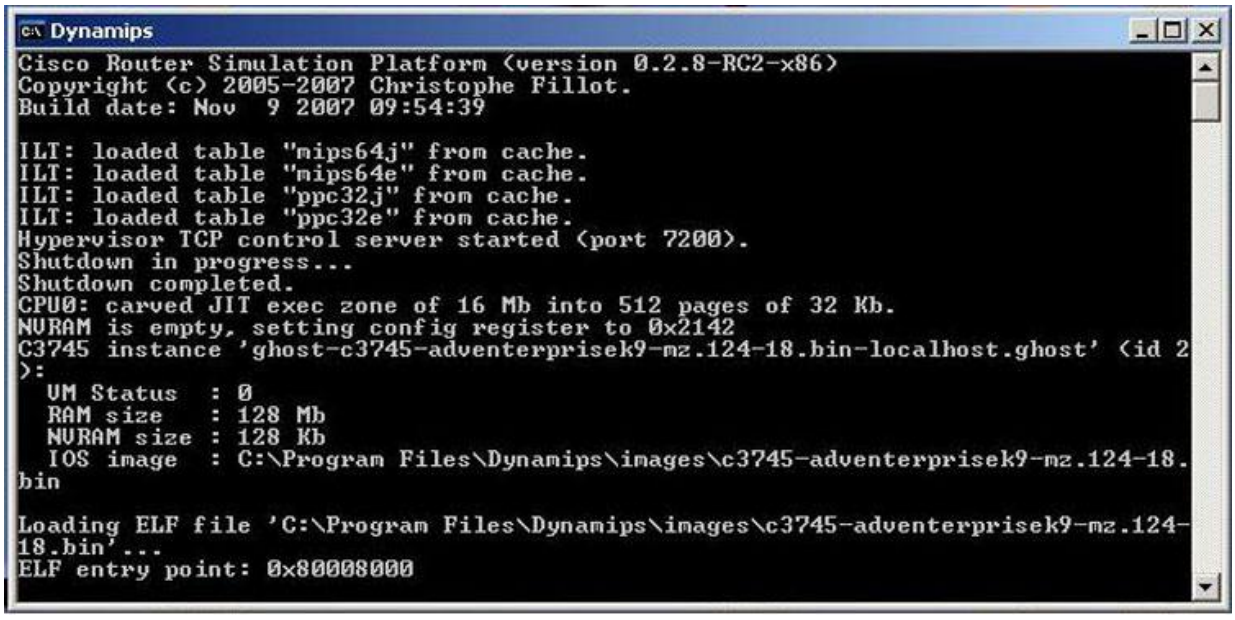


Figure 23. Real time view of Dynamips execution

Dynagen

Dynagen is an active development as an add-on written for Dynamips. It is a front-end add-on that allows the use of an .INI configuration file to provision Dynamips based emulated networks.

Figure 24. Real time view of Dynagen execution

The above shown diagram represents the console view of running instances of router s IOS in our implemented topology. Following are the acronyms that have been used used in this project.

1. Router R1 -- Karlskrona
2. Router R2 -- Ronneby

The running instances of Cisco IOS for both R1-Karlskrona and R2-Ronneby routers can be administered via Dynagen. The Dynamips based instances of Cisco IOS can now be started and stoped by issuing some simple commands.

4.2 Call-Hijacking demonstration

Call hijacking is performed by establishing a SIP based VOIP call between 2 SIP softphone users and an intruder has grabbed the packets by using a packet sniffer tool (wireshark) in the absence of encryption.

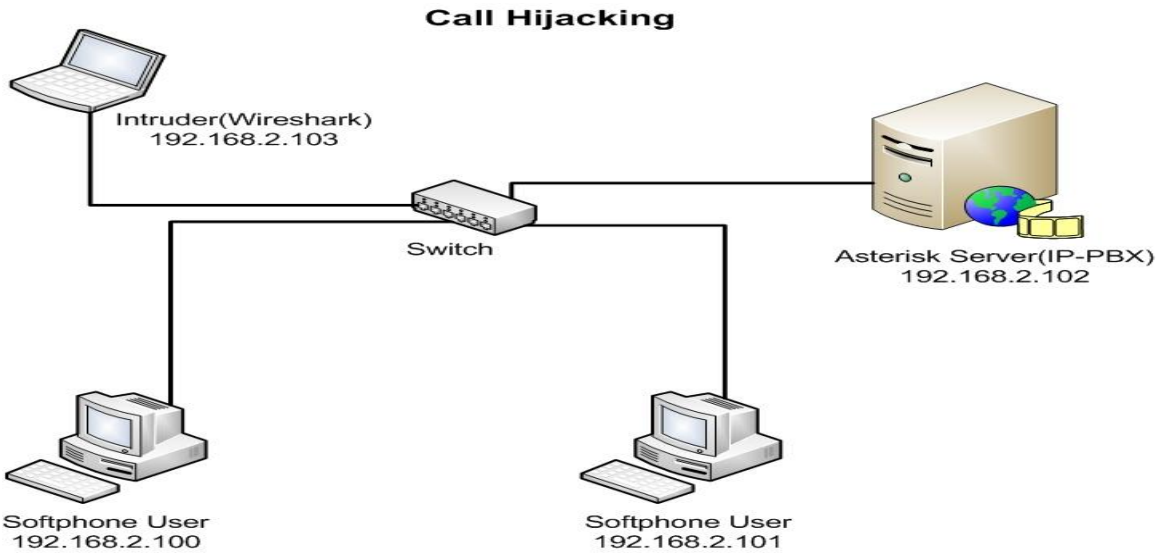


Figure 25. Call Hijacking in practice

Asterisk screenshot - established SIP session between the two softphone users:

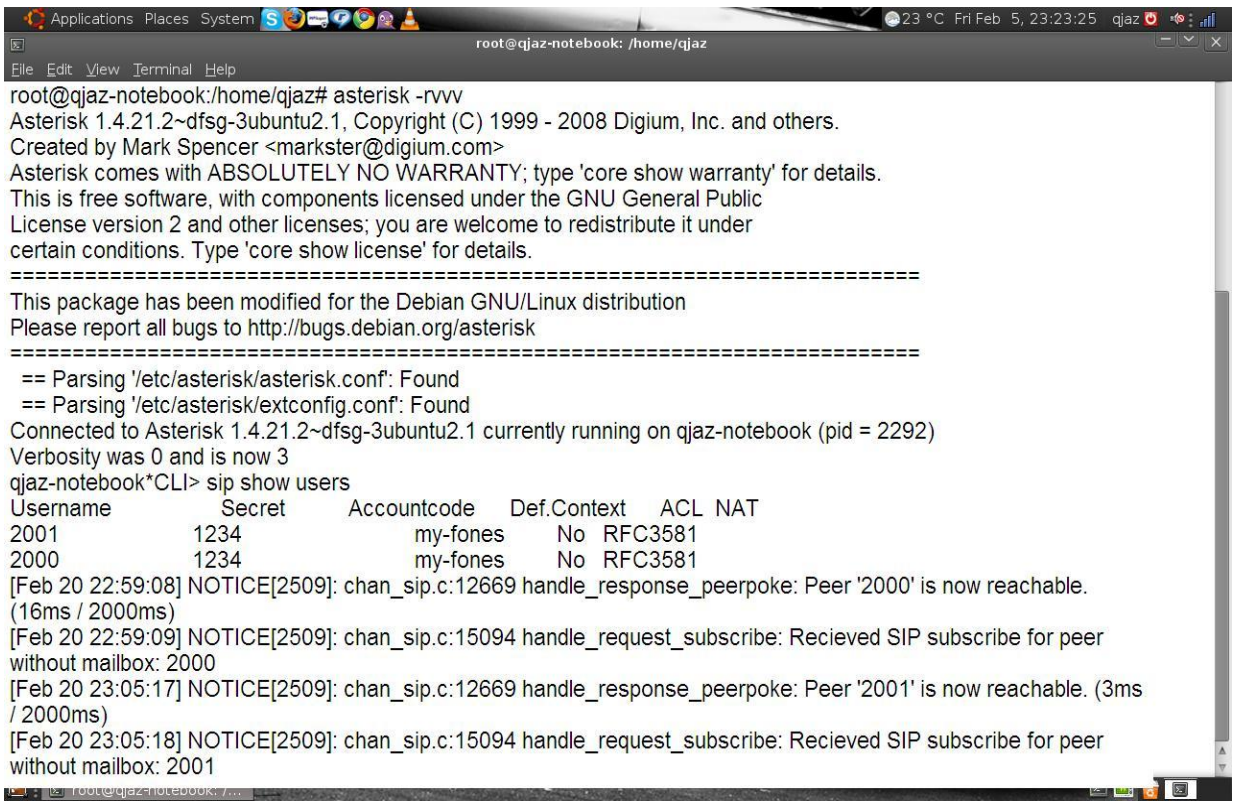


Figure 26. Established SIP session via Asterisk

Packet capturing screenshot (Wireshark), done by intruder:

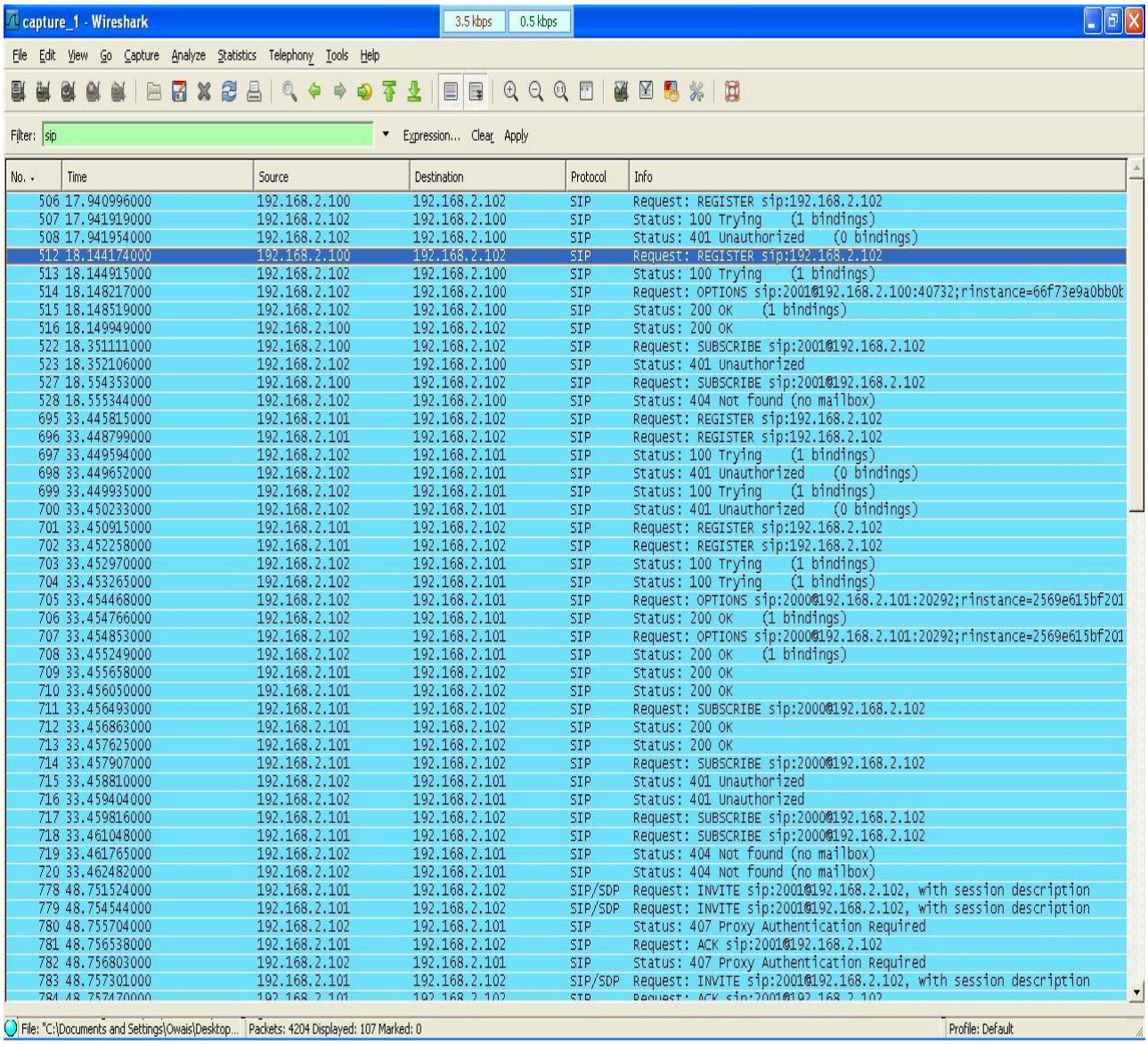


Figure 27. Captured SIP packets(Wireshark)

4.2.1 Packet dumps collected via wireshark

All the SIP packets transferred between sofphone users and Asterisk server were captured by the intruder by using Wireshark:

No. Time Source Destination Protocol Info

506 17.940996000 192.168.2.100 192.168.2.102 SIP Request: REGISTER sip:192.168.2.102

Frame 506 (600 bytes on wire, 600 bytes captured)

Ethernet II, Src: Micro-St_3f:6e:18 (00:1d:92:3f:6e:18), Dst: Intel_50:9b:de (00:16:6f:50:9b:de)

Internet Protocol, Src: 192.168.2.100 (192.168.2.100), Dst: 192.168.2.102 (192.168.2.102)

User Datagram Protocol, Src Port: 40732 (40732), Dst Port: sip (5060)

Session Initiation Protocol

0000 00 16 6f 50 9b de 00 1d 92 3f 6e 18 08 00 45 00 ..oP.....?n...E.

0010 02 4a f4 69 00 00 80 11 be 1e c0 a8 02 64 c0 a8 .J.i.....d..

0020 02 66 9f 1c 13 c4 02 36 9f 5d 52 45 47 49 53 54 .f.....6.]REGIST

0030 45 52 20 73 69 70 3a 31 39 32 2e 31 36 38 2e 32 ER sip:192.168.2

0040 2e 31 30 32 20 53 49 50 2f 32 2e 30 0d 0a 56 69 .102 SIP/2.0..Vi

0050 61 3a 20 53 49 50 2f 32 2e 30 2f 55 44 50 20 31 a: SIP/2.0/UDP 1

0060 39 32 2e 31 36 38 2e 32 2e 31 30 30 3a 34 30 37 92.168.2.100:407

0070 33 32 3b 62 72 61 6e 63 68 3d 7a 39 68 47 34 62 32;branch=z9hG4b

0080 4b 2d 64 38 37 35 34 33 2d 30 35 34 37 38 64 33 K-d87543-05478d3

0090 30 66 37 36 65 63 36 37 34 2d 31 2d 2d 64 38 37 0f76ec674-1--d87

00a0 35 34 33 2d 3b 72 70 6f 72 74 0d 0a 4d 61 78 2d 543-;rport..Max-

```
00b0  46 6f 72 77 61 72 64 73 3a 20 37 30 0d 0a 43 6f   Forwards: 70..Co
00c0  6e 74 61 63 74 3a 20 3c 73 69 70 3a 32 30 30 31   ntact: <sip:2001
00d0  40 31 39 32 2e 31 36 38 2e 32 2e 31 30 30 3a 34   @192.168.2.100:4
00e0  30 37 33 32 3b 72 69 6e 73 74 61 6e 63 65 3d 36   0732;rinstance=6
00f0  36 66 37 33 65 39 61 30 62 62 30 62 37 62 34 3e   6f73e9a0bb0b7b4>
0100  0d 0a 54 6f 3a 20 22 32 30 30 31 22 3c 73 69 70   ..To: "2001"<sip
0110  3a 32 30 30 31 40 31 39 32 2e 31 36 38 2e 32 2e   :2001@192.168.2.
0120  31 30 32 3e 0d 0a 46 72 6f 6d 3a 20 22 32 30 30   102>..From: "200
0130  31 22 3c 73 69 70 3a 32 30 30 31 40 31 39 32 2e   1"<sip:2001@192.
0140  31 36 38 2e 32 2e 31 30 32 3e 3b 74 61 67 3d 61   168.2.102>;tag=a
0150  32 32 34 37 36 34 33 0d 0a 43 61 6c 6c 2d 49 44   2247643..Call-ID
0160  3a 20 34 38 35 63 34 37 32 35 64 63 33 65 32 36   : 485c4725dc3e26
0170  35 63 4e 7a 63 79 4d 44 56 6c 4e 6a 5a 6d 4d 54   5cNzcyMDVlNjZmMT
0180  4e 6a 4d 44 41 7a 4e 32 45 34 4d 57 45 32 5a 47   NjMDAzN2E4MWE2ZG
0190  55 33 5a 54 42 6d 4d 44 6b 78 5a 6d 55 2e 0d 0a   U3ZTBmMDkxZmU...
01a0  43 53 65 71 3a 20 31 20 52 45 47 49 53 54 45 52   CSeq: 1 REGISTER
01b0  0d 0a 45 78 70 69 72 65 73 3a 20 33 36 30 30 0d   ..Expires: 3600.
01c0  0a 41 6c 6c 6f 77 3a 20 49 4e 56 49 54 45 2c 20   .Allow: INVITE,
01d0  41 43 4b 2c 20 43 41 4e 43 45 4c 2c 20 4f 50 54   ACK, CANCEL, OPT
01e0  49 4f 4e 53 2c 20 42 59 45 2c 20 52 45 46 45 52   IONS, BYE, REFER
01f0  2c 20 4e 4f 54 49 46 59 2c 20 4d 45 53 53 41 47   , NOTIFY, MESSAG
0200  45 2c 20 53 55 42 53 43 52 49 42 45 2c 20 49 4e   E, SUBSCRIBE, IN
0210  46 4f 0d 0a 55 73 65 72 2d 41 67 65 6e 74 3a 20   FO..User-Agent:
0220  58 2d 4c 69 74 65 20 72 65 6c 65 61 73 65 20 31   X-Lite release 1
0230  30 30 32 74 78 20 73 74 61 6d 70 20 32 39 37 31   002tx stamp 2971
0240  32 0d 0a 43 6f 6e 74 65 6e 74 2d 4c 65 6e 67 74   2..Content-Lengt
0250  68 3a 20 30 0d 0a 0d 0a                           h: 0....
```

No. Time Source Destination Protocol Info

507 17.941919000 192.168.2.102 192.168.2.100 SIP Status: 100 Trying (1 bindings)

Frame 507 (532 bytes on wire, 532 bytes captured)

Ethernet II, Src: Intel_50:9b:de (00:16:6f:50:9b:de), Dst: Micro-St_3f:6e:18 (00:1d:92:3f:6e:18)

Internet Protocol, Src: 192.168.2.102 (192.168.2.102), Dst: 192.168.2.100 (192.168.2.100)

User Datagram Protocol, Src Port: sip (5060), Dst Port: 40732 (40732)

No. Time Source Destination Protocol Info

514 18.148217000 192.168.2.102 192.168.2.100 SIP Request: OPTIONS sip:2001@192.168.2.100:40732;rinstance=66f73e9a0bb0b7b4

Frame 514 (612 bytes on wire, 612 bytes captured)

Ethernet II, Src: Intel_50:9b:de (00:16:6f:50:9b:de), Dst: Micro-St_3f:6e:18 (00:1d:92:3f:6e:18)

Internet Protocol, Src: 192.168.2.102 (192.168.2.102), Dst: 192.168.2.100 (192.168.2.100)

User Datagram Protocol, Src Port: sip (5060), Dst Port: 40732 (40732)

No. Time Source Destination Protocol Info

515 18.148519000 192.168.2.102 192.168.2.100 SIP Status:200 OK(1 bindings)

Frame 515 (641 bytes on wire, 641 bytes captured)

Ethernet II, Src: Intel_50:9b:de (00:16:6f:50:9b:de), Dst: Micro-St_3f:6e:18 (00:1d:92:3f:6e:18)

Internet Protocol, Src: 192.168.2.102 (192.168.2.102), Dst: 192.168.2.100 (192.168.2.100)

User Datagram Protocol, Src Port: sip (5060), Dst Port: 40732 (40732)

No. Time Source Destination Protocol Info

695 33.445815000 192.168.2.101 192.168.2.102 SIP Request: REGISTER sip:192.168.2.102

Frame 695 (600 bytes on wire, 600 bytes captured)

Ethernet II, Src: de:af:dd:ee:aa:ff (de:af:dd:ee:aa:ff), Dst: Intel_50:9b:de (00:16:6f:50:9b:de)

Internet Protocol, Src: 192.168.2.101 (192.168.2.101), Dst: 192.168.2.102 (192.168.2.102)

User Datagram Protocol, Src Port: 20292 (20292), Dst Port: sip (5060)

No. Time Source Destination Protocol Info

697 33.449594000 192.168.2.102 192.168.2.101 SIP Status: 100 Trying (1 bindings)

Frame 697 (532 bytes on wire, 532 bytes captured)

Ethernet II, Src: Intel_50:9b:de (00:16:6f:50:9b:de), Dst: de:af:dd:ee:aa:ff (de:af:dd:ee:aa:ff)

Internet Protocol, Src: 192.168.2.102 (192.168.2.102), Dst: 192.168.2.101 (192.168.2.101)

User Datagram Protocol, Src Port: sip (5060), Dst Port: 20292 (20292)

4.3 IPSec and Firewall Implementation

We have utilized the built on Firewall/IPS and IPSec features of Cisco 3745 series routers in this setup to achieve the maximum possible security level against various types of SIP related threats i.e. packet sniffing, registration hijacking, DOS attacks etc etc. The policies

have been made in such a way that all the inbound traffic to both of the routers is monitored and only the trusted connections are allowed to occur.

4.3.1 Router R1-Karlskrona

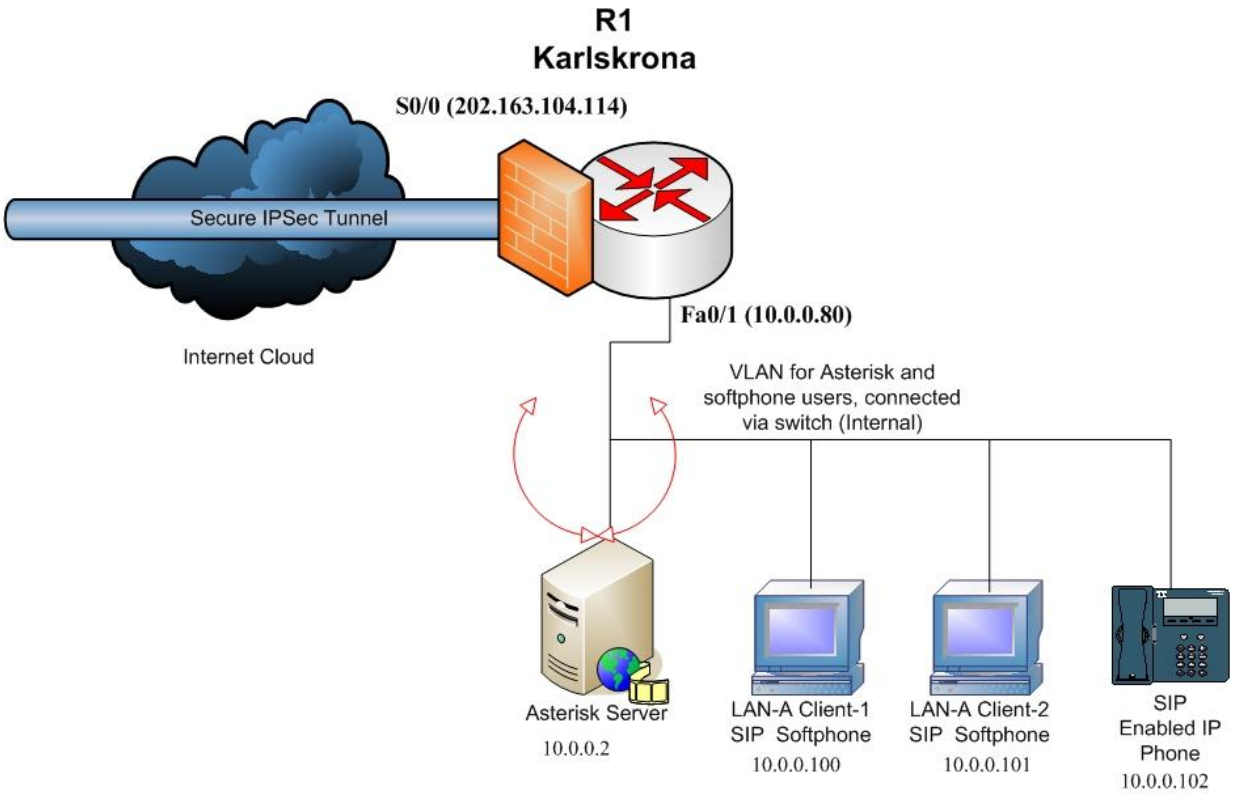


Figure 28. R1 with IPSec tunnel and Firewall

4.3.1.1 IPSec Tunnel

We have configured encryption parameters for IPSec tunnel on router ‘R1’ as mentioned in the network diagram in order to create an IPSec secure tunnel between the two sites i.e. Karlskrona and Ronneby. The configurations on the router reflect the following on R1-Karlskrona:

R1-Karlskrona#show crypto engine connections active

ID	Interface	IP-Address	State	Algorithm	Encrypt	Decrypt
1	Serial0/0	202.163.104.114	set	HMAC_MD5+3DES_56_C	0	0
2001	Serial0/0	202.163.104.114	set	3DES+SHA	0	4
2002	Serial0/0	202.163.104.114	set	3DES+SHA	4	0

R1-Karlskrona#show crypto isakmp sa

dst	src	state	conn-id	slot	status
202.163.104.113	202.163.104.114	QM_IDLE	1	0	ACTIVE

R1-Karlskrona#show ip nat translations

Pro	Inside global	Inside local	Outside local	Outside global
udp	10.0.0.80:520	10.0.0.80:520	224.0.0.9:520	224.0.0.9:520

This shows that a point-to-point IPSec tunnel is active over the internet between the Serial 0/0 (202.163.104.114) port of router ‘R1’ and Serial 0/0 (202.163.104.113) port of ‘R2’. Any packets transferred between the two remote sites will be secured now.

4.3.1.2 Firewall based access control policies on R1-Karlskrona

We have used Cisco Router and Security Device Manager (SDM) software to implement the access control policies and IPS (Intrusion Prevention System) on both of the connected/configured interfaces of the router.

Cisco SDM is a Web-based device-management tool for Cisco IOS® Software-based routers. Routing, switching, security, and quality-of-service(QoS) based services on Cisco routers can be easily configured while enabling proactive management through performance monitoring. It allows quick and easy management of ACLs and packet-inspection rules through a graphical interface.

Inside (trusted) Interface:

FastEthernet 0/1 (10.0.0.80)

Access rule applied to the inbound direction to deny spoofing traffic.

Access rule applied to the inbound direction to deny traffic sourced from broadcast.

Access rule applied to the inbound direction to permit all other traffic.

This configuration was applied via SDM for inside (trusted) network. All the inbound traffic to the local LAN (10.0.0.1/24) will be monitored to deny spoofing traffic and traffic sourced from broadcast. Any other packets will be allowed to pass i.e. In this case, IPSec tunnel traffic carrying SIP packets will be passed through.

Outside (untrusted) Interface:

Serial 0/0 (202.163.104.114)

Access rule applied to the inbound direction to permit IPSec Tunnel traffic.

Access ruled applied to the inbound direction traffic to deny spoofing traffic.

Access rule applied to the inbound direction to deny traffic sourced from broadcast.

Access rule applied to the inbound direction to deny all other traffic.

This configuration was applied via SDM for the outside (untrusted) network. All the traffic coming on to the router's {Serial 0/0 (202.163.104.114)} interface will be monitored to permit only the IPSec secure tunnel traffic. All the other traffic will be denied to achieve the maximum possible security level.

Figure 29. Security Policies configurations summary on R1

4.3.2 Router R2-Ronneby

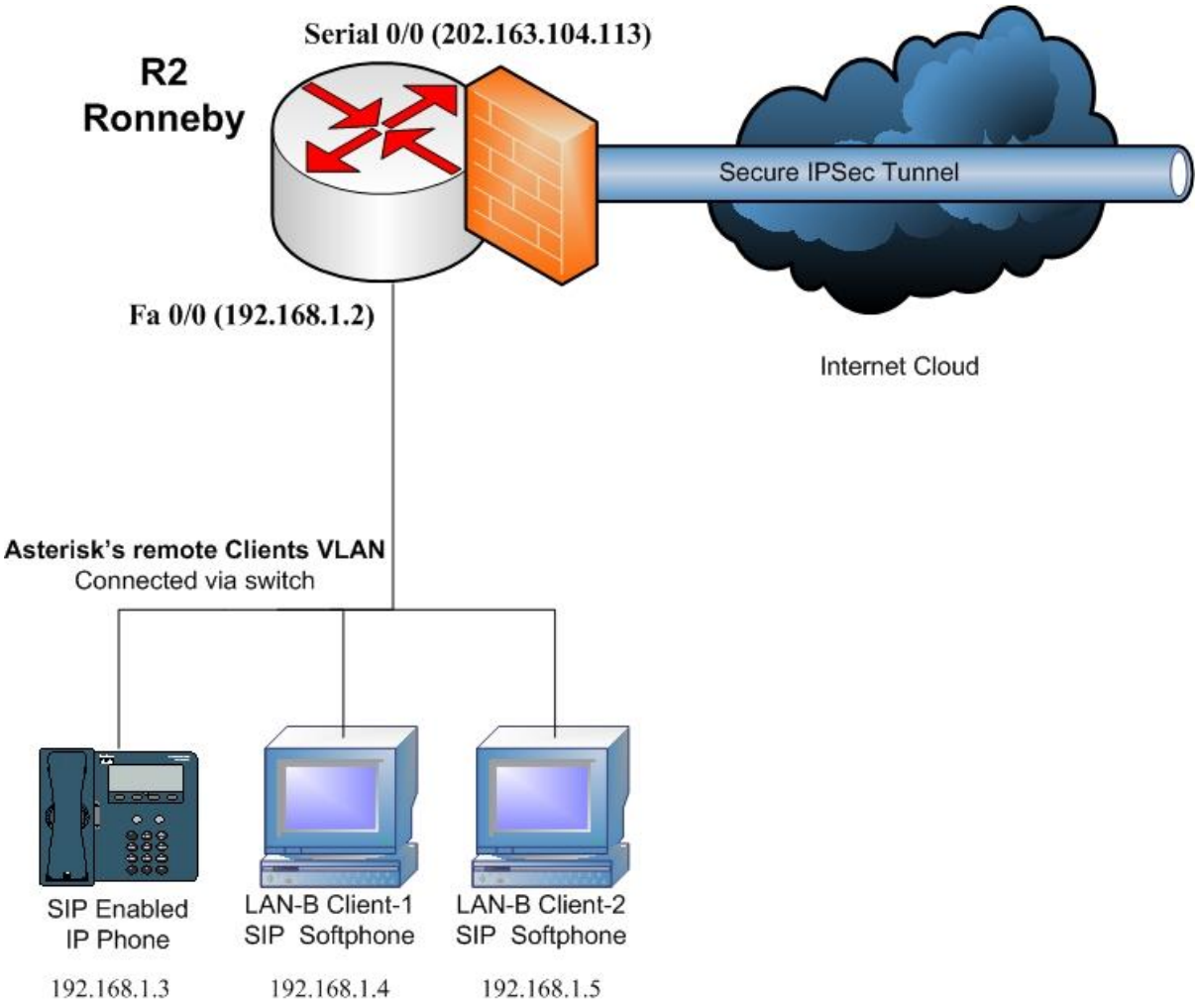


Figure 30. R2 with IPSec Tunnel and Firewall

4.3.2.1 IPSec Tunnel

We have configured encryption parameters for IPSec tunnel on router 'R2' as mentioned in the network diagram in order to provide it the connectivity to previously created IPSec secure tunnel on 'R1'. The configurations reflect the following on R2-Ronneby :

R2-Ronneby#show crypto engine connections active

ID	Interface	IP-Address	State	Algorithm	Encrypt	Decrypt
1	Serial0/0	202.163.104.113	set	HMAC_MD5+3DES_56_C	0	0
2001	Serial0/0	202.163.104.113	set	3DES+SHA	0	9
2002	Serial0/0	202.163.104.113	set	3DES+SHA	9	0

R2-Ronneby#show crypto isakmp sa

dst	src	state	conn-id	slot	status
202.163.104.114	202.163.104.113	QM_IDLE	1	0	ACTIVE

This clearly shows that a point-to-point IPSec tunnel can now be established over the internet between the Serial 0/0 (202.163.104.113) port of router ‘R2’ and Serial 0/0 (202.163.104.114) port of ‘R1’. Any packets transfer between the two remote sites will be secured now.

4.3.2.2 Firewall based access control policies on R2-Ronneby

We have used Cisco Router and Security Device Manager (SDM) software to implement the following access control policies and IPS (Intrusion Prevention System) on both of the connected/configured interfaces of the router.

Inside (trusted) Interface:

FastEthernet 0/0 (192.168.1.2)

Access rule applied to the inbound direction to deny spoofing traffic.

Access rule applied to the inbound direction to deny traffic sourced from broadcast.

Access rule applied to the inbound direction to permit all other traffic.

This configuration was applied via SDM for inside (trusted) network. All the inbound traffic to the local LAN (192.168.1.1/24) will be monitored to deny spoofing traffic and traffic sourced from broadcast. Any other packets will be allowed to pass i.e. In this case, IPSec tunnel traffic carrying SIP packets will be passed through.

Outside (untrusted) Interface:

Serial 0/0 (202.163.104.113)

Access rule applied to the inbound direction to permit IPSec Tunnel traffic.

Access ruled applied to the inbound direction traffic to deny spoofing traffic.

Access rule applied to the inbound direction to deny traffic sourced from broadcast.

Access rule applied to the inbound direction to deny all other traffic.

This configuration was applied via SDM for the outside (untrusted) network. All the traffic coming on to the router's { Serial 0/0 (202.163.104.113)} interface will be monitored to permit only the IPSec secure tunnel traffic. All the other traffic will be denied to achieve the maximum possible security level.

Figure.31 Security Policies configurations summary on R2

Our proposed test network is highly secured as it is now equipped with IPSec tunnel and Firewall with access control policies and IPS. It can now block any type of SIP related attacks i.e. packet sniffing, registration hijacking, message tempering, session tear down, DOS attacks.

5 Summary & Conclusisons

5.1 Summary

The scope of this Master's thesis outlined in Section 1.3, have been achieved. The first goal was to study the security threats to a SIP based VOIP system, which is discussed in chapter 2. This report presented the analysis of various types of threats that could be used to exploit the SIP by means of exploiting the authentication, confidentiality and integrity of a SIP based VOIP implementation. The second one was to study the requirements of a firewall to pass SIP signaling in and out of a private network that uses NAT. This subject has been covered in-depth throughout the entire report. Another important aspect of this thesis was to implement a SIP enable IPsec VPN tunnel to provide greater security for our proposed network which is outlined in chapter 4. This report also discussed the security mechanisms that can be used with SIP to ensure a greater level of security and privacy.

5.2 Conclusions

There exist mechanisms to improve SIP security in every aspect of security. SIP has additional security requirements when compared to PSTN and SS7 signaling, because on public Internet it is really difficult to achieve both security and privacy. SIP doesn't have its own specific security mechanisms rather it relies on some of the industry standard security protocols to achieve the required level of security and privacy. Much focus has been paid to provide new services. During the development of SIP many of the security and privacy issues are recognized. In the end, the implementation of the SIP protocol and the implementations of the security mechanisms demonstrate the security level of the SIP service. Many of the security issues still needs to be resolved and are often present in SIP based networks due to poor implementations.

From the variety of solutions exists today for NAT/firewall traversal for SIP, only the Interactive connectivity Establishment (ICE) method and the Session Border Controller technique (SBC) fulfill the requirements of a unified solution. The non-deterministic behavior of many of the NATs and the extension of VoIP network over multiple physical

networks, each with its own NAT, makes it difficult for ICE to handle the SIP traversal. This makes SBC remaining the only feasible solution. The future large deployment of SBCs depends on a standardization that is still missing, and the need of assurance that the use of SBCs will not introduce any threat to network security, due to the increase of overall network architecture.

List of Figures

References:

[1] Samer El Sawda and Pascal Urien, "*SIP Security Attacks and Solutions: A state-of-the-art review*," 0-7803-9521-2/06 ©2006 IEEE.

[2] Gonzalo Camarillo, "*SIP Demystified*," ISBN- 0-07-137340-3, Copyright © 2002 by The McGraw-Hill Companies, Inc.

[3] Alan B.Johnston, "*SIP:Understanding the Session Initiation Protocol (Second Edition)*," ISBN-9781580536561 © 2003 Artech House Inc.

[4] Rosenberg, J., et al., "*SIP: Session Initiation Protocol*," RFC 3261, 2002, http://tools.ietf.org/html/rfc3261

[5] Roach, A., "*Session Initiation Protocol (SIP)-Specific Event Notification*," RFC 3265, 2002, http://tools.ietf.org/html/rfc3265

[6] Campbell, B., et al., "*Session Initiation Protocol (SIP) Extension for Instant Messaging*," RFC 3428, 2002, http://tools.ietf.org/html/rfc3428

[7] Rosenberg, J., H. Schulzrinne, and G. Camarillo, "*The Stream Control Transmission Protocol as a Transport for the Session Initiation Protocol*," RFC 4168, 2005, http://tools.ietf.org/html/rfc4168

[8] Mark Collier, "*Basic Vulnerability Issues for SIP Security,"* © 2005 SecureLogix Corporation.

[9] Mihai Aurel Constantinescu, Doina Oana Cernaianu, Victor Croitoru, "*NAT/Firewall Traversal For SIP: Issues and Solutions"* 0-7803-9029-6/05 ©2005 IEEE.

[10] J Rosenberg. D. Drew,H. Schulzrinne. "*Getting SIP through Firewalls and NATs*", Internet Draft, draft-rosenberg-sip-firewalls-OO.txt, February 2000.

[11] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, L. Jones, "*SOCKS Protocol Version 5"*, RFC 1928, March 1996.

[12] P. Srisuresh, M. Holdrege, *"IP Network Address Translator (NAT) Terminology and Considerations"*, RFC 2663, August 1999.

[13] P. Srisuresh, M. Uoldrege, "*IP Network Address Translator (NAT) Terminology and Considerations*", RFC 2663, August 1999.

[14] G. Huston, *"Anatomy: A Look Inside Network Address Translation"*, in The Internet

Protocol Journal, vol. 7, Number 3, September 2004, pp. 2-32.

[15] J. Rosenberg, H. Schulzrinne, *"An Extension to the Session Initiation Protocol (SIP) for Symmetric Response Routing"*, RFC 3581, August 2003.

[16] J. Roseriberg, H Hawrylyshen, *"Sip Conventions for Connection Usage"*. Internet Draft, draft-ietf-jennings-sipping-outbound-00 (work in progress), July 2004.

[17] *D.* Wing, "*Symmetric RTP and RTCP Considered Helpful"*, Internet Draft, draft - wing-mmusic-symmetric-rtprtcp-01 (work in progress).

[18] J. Rosenberg, J. Weinberger, C. Huiterna, R. Mahy, *"STUN – Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)"*, RFC 3489, March 2003.

[19] J. Rosenberg, R. Mahy, C. Huitema, *"Traversal Using Relay NAT (TURN)"*, Internet Draft, draft-rosenberg-midcom-turn-06, October 2004.

[20] J. Rosenberg*, "Interactive Connectivity establishment (ICE): A Methodology for Network Address Translator (NAT) Traversal for Multimedia Sessions Establishment Protocols"*, Internet Draft, draft-ietf-mmusic-ice-03, October 2004.

[21] B. Carpenter, "*Middleboxes: Taxonomy and Issues"*, RFC 3234, February 2002.

[22] Session Border Forum, http://www.sessioncontrollerforum.org/.

[23] V.gubani, A.jefferey, "*The Use of Transport Layer Security (TLS) in the Session Initiation Protocol (SIP)*", February 2006.

[24] Travis Russel, "*Controlling Convergent Networks*", Mc Graw-Hill Publications; June 2008.

[25] V.gubani, A.jefferey, "*The Use of session description protocol (SDP) in the Session Initiation Protocol (SIP)***",** February 2006.

[26] Henning Schulzrinne, "*The Real Time Streaming Protocol (RTSP)"* October 2008.

[27] Juanits Ellis, Charles Pursell, Joy Rehman, "*The Convergence of Voice, Video & Netwok Data*", August 2003.

[28] Deauville, " *A perspective on 3GPP*", SEPTEMBER 2001.

[29] D. Kroeselberg Siemens, "*SIP security requirements from 3G wireless networks*", January 2001.

[30] Narayan Parameshwar, Chirs reece, "SIP and 3GPP".

Appendix

Command-line Based Security Implementation on Routers

R1-Karlskrona

```
karlskrona(config)#crypto isakmp policy 10
karlskrona(config-isakmp)#encryption 3des
karlskrona(config-isakmp)#hash md5
karlskrona(config-isakmp)#authentication pre-share
karlskrona(config-isakmp)#group 2
karlskrona(config-isakmp)#lifetime 28800
karlskrona(config-isakmp)#exit
Karlskrona(config)#crypto isakmp key cisco address 172.25.4.34
Karlskrona(config)#crypto ipsec security-association lifetime seconds 28800
Karlskrona(config)#crypto ipsec transform-set test esp-3
Karlskrona(config)#crypto ipsec transform-set test esp-3des esp-sha-hmac
Karlskrona(cfg-crypto-trans)#exit
Karlskrona(config)#crypto map qanit 10 ipsec-isakmp
% NOTE: This new crypto map will remain disabled until a peer
      and a valid access list have been configured.
Karlskrona(config-crypto-map)#set peer 172.25.4.34
Karlskrona(config-crypto-map)#set transform-set test
Karlskrona(config-crypto-map)#match add
Karlskrona(config-crypto-map)#match address ahbr
Karlskrona(config-crypto-map)#exit
Karlskrona(config)#int tunnel 101
Karlskrona(config-if)#ip a
*Mar  1 01:43:21.831: %LINEPROTO-5-UPDOWN: Line protocol on Interface Tunnel101,
changed state to downdd
Karlskrona(config-if)#ip address  10.100.200.101 255.255.255.252
Karlskrona(config-if)#tunnel source 172.25.4.33
Karlskrona(config-if)#tunnel destination 172.25.4.34
Karlskrona(config-if)#
*Mar  1 01:44:15.863: %LINEPROTO-5-UPDOWN: Line protocol on Interface Tunnel101,
changed state to upcry
Karlskrona(config-if)#crypto map qanit
Karlskrona(config-if)#
*Mar  1 01:44:29.091: %CRYPTO-6-ISAKMP_ON_OFF: ISAKMP is ON^Z
*Mar  1 01:44:32.559: %SYS-5-CONFIG_I: Configured from console by consoleh cry
Karlskrona#sh crypto map interface tunnel 101
Crypto Map "qanit" 10 ipsec-isakmp
      WARNING: This crypto map is in an incomplete state!
            (missing peer or access-list definitions)
      Peer = 172.25.4.34
      Extended IP access list ahbr
      Security association lifetime: 4608000 kilobytes/28800 seconds
      PFS (Y/N): N
      Transform sets={
            test,
```

```
        }
        Interfaces using crypto map qanit:
                Tunnel101

Karlskrona#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Karlskrona(config)#ip access-list ahbr
                          ^
% Invalid input detected at '^' marker.

Karlskrona(config)#ip access-list extended ahbr
Karlskrona(config-ext-nacl)#permit ip host 10.2.249.13 host 10.95.4.183
Karlskrona(config-ext-nacl)#exit
*Mar  1 01:45:59.515: %SYS-5-CONFIG_I: Configured from console by consolecry
Karlskrona#show crypto map interface tunnel 101
Crypto Map "qanit" 10 ipsec-isakmp
        Peer = 172.25.4.34
        Extended IP access list ahbr
            access-list ahbr permit ip host 10.2.249.13 host 10.95.4.183
        Current peer: 172.25.4.34
        Security association lifetime: 4608000 kilobytes/28800 seconds
        PFS (Y/N): N
        Transform sets={
                test,
        }
        Interfaces using crypto map qanit:
                Tunnel101

Karlskrona#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Karlskrona(config)#int lo 101
Karlskrona(config-if)#ip add 10.2.249.13 255.255.255.0
Karlskrona(config-if)#^Z
Karlskrona#
*Mar  1 01:46:44.787: %SYS-5-CONFIG_I: Configured from console by console
Karlskrona#
Karlskrona#wr
Building configuration...
[OK]
Karlskrona#
Karlskrona#sh run
Building configuration...

Current configuration : 1486 bytes
!
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname Karlskrona
```

```
!
boot-start-marker
boot-end-marker
!
enable secret 5 $1$D0w3$HGivliSsgx.02RlHeEjAC.
!
no aaa new-model
memory-size iomem 5
!
!
ip cef
!
!
crypto isakmp policy 10
 encr 3des
 hash md5
 authentication pre-share
 group 2
 lifetime 28800
crypto isakmp key cisco address 172.25.4.34
!
crypto ipsec security-association lifetime seconds 28800
!
crypto ipsec transform-set test esp-3des esp-sha-hmac
!
crypto map qanit 10 ipsec-isakmp
 set peer 172.25.4.34
 set transform-set test
 match address ahbr
!
!
!
!
interface Loopback0
 ip address 1.1.1.1 255.255.255.0
!
interface Loopback101
 ip address 10.2.249.13 255.255.255.0
!
interface Tunnel0
 no ip address
!
interface Tunnel101
 ip address 10.100.200.101 255.255.255.252
 tunnel source 172.25.4.33
 tunnel destination 172.25.4.34
 crypto map qanit
!
interface FastEthernet0/0
 ip address 10.95.4.1 255.255.255.0
 duplex auto
```

```
 speed auto
!
interface FastEthernet1/0
 ip address 172.25.4.33 255.255.255.252
 duplex auto
 speed auto
!
router rip
 version 2
 network 1.0.0.0
 network 10.0.0.0
 network 172.25.0.0
 no auto-summary
!
ip http server
no ip http secure-server
!
!
!
!
ip access-list extended ahbr
 permit ip host 10.2.249.13 host 10.95.4.183
!
!
!
control-plane
!
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
!
!
end

Karlskrona# sh crypto map interface tunnel 101
Crypto Map "qanit" 10 ipsec-isakmp
        Peer = 172.25.4.34
        Extended IP access list ahbr
            access-list ahbr permit ip host 10.2.249.13 host 10.95.4.183
        Current peer: 172.25.4.34
        Security association lifetime: 4608000 kilobytes/28800 seconds
        PFS (Y/N): N
        Transform sets={
                test,
        }
        Interfaces using crypto map qanit:
                Tunnel101
```

```
Karlskrona#sh run
Building configuration...

Current configuration : 1486 bytes
!
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname Karlskrona
!
boot-start-marker
boot-end-marker
!
enable secret 5 $1$D0w3$HGivliSsgx.02RlHeEjAC.
!
no aaa new-model
memory-size iomem 5
!
!
ip cef
!
!
crypto isakmp policy 10
 encr 3des
 hash md5
 authentication pre-share
 group 2
 lifetime 28800
crypto isakmp key cisco address 172.25.4.34
!
crypto ipsec security-association lifetime seconds 28800
!
crypto ipsec transform-set test esp-3des esp-sha-hmac
!
crypto map qanit 10 ipsec-isakmp
 set peer 172.25.4.34
 set transform-set test
 match address ahbr
!
!
!
!
interface Loopback0
 ip address 1.1.1.1 255.255.255.0
!
interface Loopback101
 ip address 10.2.249.13 255.255.255.0
!
interface Tunnel0
```

```
 no ip address
!
interface Tunnel101
 ip address 10.100.200.101 255.255.255.252
 tunnel source 172.25.4.33
 tunnel destination 172.25.4.34
 crypto map qanit
!
interface FastEthernet0/0
 ip address 10.95.4.1 255.255.255.0
 duplex auto
 speed auto
!
interface FastEthernet1/0
 ip address 172.25.4.33 255.255.255.252
 duplex auto
 speed auto
!
router rip
 version 2
 network 1.0.0.0
 network 10.0.0.0
 network 172.25.0.0
 no auto-summary
!
ip http server
no ip http secure-server
!
!
!
!
ip access-list extended ahbr
 permit ip host 10.2.249.13 host 10.95.4.183
!
!
!
control-plane
!
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
!
!
end

Karlskrona#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Karlskrona(config)#ip access-list extended ahbr
```

```
Karlskrona(config-ext-nacl)#no 10
Karlskrona(config-ext-nacl)#permit ip host 10.2.249.13 host 10.95.8.183
Karlskrona(config-ext-nacl)#^Z
Karlskrona#co
*Mar  1 01:57:39.167: %SYS-5-CONFIG_I: Configured from console by consolenf t
Enter configuration commands, one per line.  End with CNTL/Z.
Karlskrona(config)#do sh ip int brief
Interface              IP-Address      OK? Method Status                Protocol
FastEthernet0/0        10.95.4.1       YES manual up                    up
FastEthernet1/0        172.25.4.33     YES manual up                    up
Loopback0              1.1.1.1         YES manual up                    up
Loopback101            10.2.249.13     YES manual up                    up
Tunnel0                unassigned      YES unset  up                    down
Tunnel101              10.100.200.101  YES manual up                    up
Karlskrona(config)#
Karlskrona(config)#ip route 10.95.8.183 255.255.255.255 10.100.200.102
Karlskrona(config)#^Z
Karlskrona#ping
*Mar  1 01:58:50.255: %SYS-5-CONFIG_I: Configured from console by
console10.95.8.183

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.95.8.183, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 8/36/88 ms
Karlskrona#sh cry
Karlskrona#sh crypto is
Karlskrona#sh crypto isakmp sa
dst            src            state          conn-id slot status

Karlskrona#
Karlskrona#ping 10.95.8.183 source 10.2.249.13

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.95.8.183, timeout is 2 seconds:
Packet sent with a source address of 10.2.249.13
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 32/83/176 ms
Karlskrona#
Karlskrona#sh crypto isakmp sa
dst            src            state          conn-id slot status
172.25.4.34     172.25.4.33     QM_IDLE             1    0 ACTIVE

Karlskrona#
Karlskrona#sh ip int brief
Interface              IP-Address      OK? Method Status                Protocol
FastEthernet0/0        10.95.4.1       YES manual up                    up
FastEthernet1/0        172.25.4.33     YES manual up                    up
Loopback0              1.1.1.1         YES manual up                    up
Loopback101            10.2.249.13     YES manual up                    up
Tunnel0                unassigned      YES unset  up                    down
```

```
Tunnel101              10.100.200.101  YES manual up                    up
Karlskrona#
Karlskrona#sh ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     1.0.0.0/24 is subnetted, 1 subnets
C       1.1.1.0 is directly connected, Loopback0
     2.0.0.0/24 is subnetted, 1 subnets
R       2.2.2.0 [120/1] via 172.25.4.34, 00:00:22, FastEthernet1/0
                [120/1] via 10.100.200.102, 00:00:17, Tunnel101
     172.25.0.0/30 is subnetted, 1 subnets
C       172.25.4.32 is directly connected, FastEthernet1/0
     10.0.0.0/8 is variably subnetted, 6 subnets, 3 masks
C       10.95.4.0/24 is directly connected, FastEthernet0/0
R       10.95.8.0/24 [120/1] via 172.25.4.34, 00:00:22, FastEthernet1/0
                     [120/1] via 10.100.200.102, 00:00:17, Tunnel101
R       10.100.200.0/24 [120/1] via 172.25.4.34, 00:00:22, FastEthernet1/0
C       10.100.200.100/30 is directly connected, Tunnel101
S       10.95.8.183/32 [1/0] via 10.100.200.102
C       10.2.249.0/24 is directly connected, Loopback101
Karlskrona#
Karlskrona#sh crypto isakmp sa
dst             src             state          conn-id slot status
172.25.4.34     172.25.4.33     QM_IDLE              1    0 ACTIVE

Karlskrona#
Karlskrona#sh crypto map interface tunnel 101
Crypto Map "qanit" 10 ipsec-isakmp
        Peer = 172.25.4.34
        Extended IP access list ahbr
            access-list ahbr permit ip host 10.2.249.13 host 10.95.8.183
        Current peer: 172.25.4.34
        Security association lifetime: 4608000 kilobytes/28800 seconds
        PFS (Y/N): N
        Transform sets={
                test,
        }
        Interfaces using crypto map qanit:
                Tunnel101

Karlskrona#
Karlskrona#sh crypto ipsec sa interface tunnel 1o1
                                ^
```

```
% Invalid input detected at '^' marker.

Karlskrona#sh crypto ipsec sa interface tunnel 101

interface: Tunnel101
    Crypto map tag: qanit, local addr 172.25.4.33

   protected vrf: (none)
   local  ident (addr/mask/prot/port): (10.2.249.13/255.255.255.255/0/0)
   remote ident (addr/mask/prot/port): (10.95.8.183/255.255.255.255/0/0)
   current_peer 172.25.4.34 port 500
     PERMIT, flags={origin_is_acl,}
    #pkts encaps: 4, #pkts encrypt: 4, #pkts digest: 4
    #pkts decaps: 0, #pkts decrypt: 0, #pkts verify: 0
    #pkts compressed: 0, #pkts decompressed: 0
    #pkts not compressed: 0, #pkts compr. failed: 0
    #pkts not decompressed: 0, #pkts decompress failed: 0
    #send errors 1, #recv errors 0

     local crypto endpt.: 172.25.4.33, remote crypto endpt.: 172.25.4.34
     path mtu 1476, ip mtu 1476, ip mtu idb Tunnel101
     current outbound spi: 0x7B3B161A(2067469850)

     inbound esp sas:
      spi: 0xE6433708(3863164680)
        transform: esp-3des esp-sha-hmac ,

Karlskrona#sh crypto ipsec sa interface tunnel 101

interface: Tunnel101
    Crypto map tag: qanit, local addr 172.25.4.33

   protected vrf: (none)
   local  ident (addr/mask/prot/port): (10.2.249.13/255.255.255.255/0/0)
   remote ident (addr/mask/prot/port): (10.95.8.183/255.255.255.255/0/0)
   current_peer 172.25.4.34 port 500
     PERMIT, flags={origin_is_acl,}
    #pkts encaps: 20, #pkts encrypt: 20, #pkts digest: 20
    #pkts decaps: 0, #pkts decrypt: 0, #pkts verify: 0
    #pkts compressed: 0, #pkts decompressed: 0
    #pkts not compressed: 0, #pkts compr. failed: 0
    #pkts not decompressed: 0, #pkts decompress failed: 0
    #send errors 1, #recv errors 0

     local crypto endpt.: 172.25.4.33, remote crypto endpt.: 172.25.4.34
     path mtu 1476, ip mtu 1476, ip mtu idb Tunnel101
     current outbound spi: 0x7B3B161A(2067469850)

     inbound esp sas:
      spi: 0xE6433708(3863164680)
        transform: esp-3des esp-sha-hmac ,
```

```
        in use settings ={Tunnel, }
        conn id: 2001, flow_id: SW:1, crypto map: qanit
        sa timing: remaining key lifetime (k/sec): (4456682/28670)
        IV size: 8 bytes
        replay detection support: Y
        Status: ACTIVE

      inbound ah sas:

      inbound pcp sas:

      outbound esp sas:
       spi: 0x7B3B161A(2067469850)
        transform: esp-3des esp-sha-hmac ,
        in use settings ={Tunnel, }
        conn id: 2002, flow_id: SW:2, crypto map: qanit
        sa timing: remaining key lifetime (k/sec): (4456666/28545)
        IV size: 8 bytes
        replay detection support: Y
        Status: ACTIVE

      outbound ah sas:

      outbound pcp sas:
Karlskrona#
karlskrona#sh crypto
*Mar  1 02:07:20.307: %SYS-5-CONFIG_I: Configured from console by consoleis
karlskrona#sh crypto isakmp sa
dst             src             state          conn-id slot status
172.25.4.34     172.25.4.33     QM_IDLE              1    0 ACTIVE

karlskrona#
karlskrona#
karlskrona#sh crypto ipsec sa interface tunnel 101

interface: Tunnel101
    Crypto map tag: qanit, local addr 172.25.4.33

   protected vrf: (none)
   local  ident (addr/mask/prot/port): (10.2.249.13/255.255.255.255/0/0)
   remote ident (addr/mask/prot/port): (10.95.8.183/255.255.255.255/0/0)
   current_peer 172.25.4.34 port 500
     PERMIT, flags={origin_is_acl,}
    #pkts encaps: 362, #pkts encrypt: 362, #pkts digest: 362
    #pkts decaps: 0, #pkts decrypt: 0, #pkts verify: 0
    #pkts compressed: 0, #pkts decompressed: 0
    #pkts not compressed: 0, #pkts compr. failed: 0
    #pkts not decompressed: 0, #pkts decompress failed: 0
    #send errors 1, #recv errors 0

     local crypto endpt.: 172.25.4.33, remote crypto endpt.: 172.25.4.34
```

```
    path mtu 1476, ip mtu 1476, ip mtu idb Tunnel101
    current outbound spi: 0x7B3B161A(2067469850)

    inbound esp sas:
     spi: 0xE6433708(3863164680)
       transform: esp-3des esp-sha-hmac ,
       in use settings ={Tunnel, }
       conn id: 2001, flow_id: SW:1, crypto map: qanit
       sa timing: remaining key lifetime (k/sec): (4456682/28309)
       IV size: 8 bytes
       replay detection support: Y
       Status: ACTIVE

    inbound ah sas:

    inbound pcp sas:

    outbound esp sas:
     spi: 0x7B3B161A(2067469850)
       transform: esp-3des esp-sha-hmac ,
       in use settings ={Tunnel, }
       conn id: 2002, flow_id: SW:2, crypto map: qanit
       sa timing: remaining key lifetime (k/sec): (4456642/28308)
       IV size: 8 bytes
       replay detection support: Y
       Status: ACTIVE

    outbound ah sas:

    outbound pcp sas:
karlskrona#
karlskrona#sh crypto engine connections active

  ID Interface           IP-Address      State  Algorithm           Encrypt  Decrypt
   1 Tunnel101           10.100.200.101  set    HMAC_MD5+3DES_56_C        0        0
2001 Tunnel101           172.25.4.33     set    3DES+SHA                  0        0
2002 Tunnel101           172.25.4.33     set    3DES+SHA                369        0

karlskrona#
karlskrona#sh crypto ipsec sa detail

interface: Tunnel101
    Crypto map tag: qanit, local addr 172.25.4.33

   protected vrf: (none)
   local  ident (addr/mask/prot/port): (10.2.249.13/255.255.255.255/0/0)
   remote ident (addr/mask/prot/port): (10.95.8.183/255.255.255.255/0/0)
   current_peer 172.25.4.34 port 500
     PERMIT, flags={origin_is_acl,}
    #pkts encaps: 374, #pkts encrypt: 374, #pkts digest: 374
    #pkts decaps: 0, #pkts decrypt: 0, #pkts verify: 0
```

```
    #pkts compressed: 0, #pkts decompressed: 0
    #pkts not compressed: 0, #pkts compr. failed: 0
    #pkts not decompressed: 0, #pkts decompress failed: 0
    #pkts no sa (send) 1, #pkts invalid sa (rcv) 0
    #pkts encaps failed (send) 0, #pkts decaps failed (rcv) 0
    #pkts invalid prot (recv) 0, #pkts verify failed: 0
    #pkts invalid identity (recv) 0, #pkts invalid len (rcv) 0
    #pkts replay rollover (send): 0, #pkts replay rollover (rcv) 0
    ##pkts replay failed (rcv): 0
    #pkts internal err (send): 0, #pkts internal err (recv) 0

     local crypto endpt.: 172.25.4.33, remote crypto endpt.: 172.25.4.34
     path mtu 1476, ip mtu 1476, ip mtu idb Tunnel101
     current outbound spi: 0x7B3B161A(2067469850)

     inbound esp sas:
      spi: 0xE6433708(3863164680)
        transform: esp-3des esp-sha-hmac ,
        in use settings ={Tunnel, }
        conn id: 2001, flow_id: SW:1, crypto map: qanit
        sa timing: remaining key lifetime (k/sec): (4456682/28297)
        IV size: 8 bytes
        replay detection support: Y
        Status: ACTIVE

     inbound ah sas:

     inbound pcp sas:

     outbound esp sas:
      spi: 0x7B3B161A(2067469850)
        transform: esp-3des esp-sha-hmac ,
        in use settings ={Tunnel, }
        conn id: 2002, flow_id: SW:2, crypto map: qanit
        sa timing: remaining key lifetime (k/sec): (4456640/28297)
        IV size: 8 bytes
        replay detection support: Y
        Status: ACTIVE

     outbound ah sas:

     outbound pcp sas:
Karlskrona#
karlskrona# sh crypto map interface ?
  Async            Async interface
  BVI              Bridge-Group Virtual Interface
  CDMA-Ix          CDMA Ix interface
  CTunnel          CTunnel interface
  Dialer           Dialer interface
  FastEthernet     FastEthernet IEEE 802.3
  Lex              Lex interface
```

```
  Loopback          Loopback interface
  MFR               Multilink Frame Relay bundle interface
  Multilink         Multilink-group interface
  Null              Null interface
  Port-channel      Ethernet Channel of interfaces
  Tunnel            Tunnel interface
  Vif               PGM Multicast Host interface
  Virtual-PPP       Virtual PPP interface
  Virtual-Template  Virtual Template interface
  Virtual-TokenRing  Virtual TokenRing

Karlskrona#
karlskrona# sh crypto map interface tunnel ?
  <0-2147483647>  Tunnel interface number

karlskrona# sh crypto map interface tunnel 101
Crypto Map "qanit" 10 ipsec-isakmp
        Peer = 172.25.4.34
        Extended IP access list ahbr
            access-list ahbr permit ip host 10.2.249.13 host 10.95.8.183
        Current peer: 172.25.4.34
        Security association lifetime: 4608000 kilobytes/28800 seconds
        PFS (Y/N): N
        Transform sets={
                test,
        }
        Interfaces using crypto map qanit:
                Tunnel101

karlskrona#
karlskrona#
karlskrona#sh crypto engine ?
  brief          Show all crypto engines in the system
  configuration  Show crypto engine config
  connections    Show connection information
  qos            Show QoS information

karlskrona#sh crypto engine conf
karlskrona#sh crypto engine configuration

        crypto engine name:  Cisco VPN Software Implementation
        crypto engine type:  software
             serial number:  00000000
      crypto engine state:  installed
    crypto engine in slot:  N/A
                platform:  Cisco Software Crypto Engine

   Crypto Adjacency Counts:
              Lock Count:  0
            Unlock Count:  0
        crypto lib version:  19.0.0
```

```
karlskrona# sh cry
karlskrona# sh crypto  ?
  ca               Show certification authority policy
  call             Show crypto call admission info
  debug-condition  Debug Condition filters
  dynamic-map      Crypto map templates
  engine           Show crypto engine info
  identity         Show crypto identity list
  ipsec            Show IPSEC policy
  isakmp           Show ISAKMP
  key              Show long term public keys
  map              Crypto maps
  mib              Show Crypto-related MIB Parameters
  optional         Optional Encryption Status
  pki              Show PKI
  session          Show crypto sessions (tunnels)
  sockets          Secure Socket Information

karlskrona# sh crypto  cal
karlskrona# sh crypto  call ?
  admission  Show crypto call admission info

karlskrona# sh crypto  call ad
karlskrona# sh crypto  call admission
% Incomplete command.

karlskrona# sh crypto  call admission ?
  statistics  Show crypto call admission statistics

karlskrona# sh crypto  call admission stat
karlskrona# sh crypto  call admission statistics
----------------------------------------------------------------------
              Crypto Call Admission Control Statistics
----------------------------------------------------------------------
System Resource Limit:        0 Max IKE SAs:     0
Total IKE SA Count:           1 active:          1 negotiating:     0
Incoming IKE Requests:        0 accepted:        0 rejected:        0
Outgoing IKE Requests:        1 accepted:        1 rejected:        0
Rejected IKE Requests:        0 rsrc low:        0 SA limit:        0

karlskrona#
```

R2-Ronneby

```
Ronneby#
Ronneby#sh crypto isakmp *Mar  1 02:04:10.939: %SYS-5-CONFIG_I: Configured from
console by consolesa
dst             src             state          conn-id slot status
172.25.4.34     172.25.4.33     QM_IDLE              1    0 ACTIVE

Ronneby#
Ronneby#sh crypto ipsec sa interface tunnel 101

interface: Tunnel101
    Crypto map tag: test, local addr 172.25.4.34

   protected vrf: (none)
   local  ident (addr/mask/prot/port): (10.95.8.183/255.255.255.255/0/0)
   remote ident (addr/mask/prot/port): (10.2.249.13/255.255.255.255/0/0)
   current_peer 172.25.4.33 port 500
     PERMIT, flags={origin_is_acl,}
    #pkts encaps: 0, #pkts encrypt: 0, #pkts digest: 0
    #pkts decaps: 196, #pkts decrypt: 196, #pkts verify: 196
    #pkts compressed: 0, #pkts decompressed: 0
    #pkts not compressed: 0, #pkts compr. failed: 0
    #pkts not decompressed: 0, #pkts decompress failed: 0
    #send errors 0, #recv errors 0

     local crypto endpt.: 172.25.4.34, remote crypto endpt.: 172.25.4.33
     path mtu 1476, ip mtu 1476, ip mtu idb Tunnel101
     current outbound spi: 0xE6433708(3863164680)

     inbound esp sas:
      spi: 0x7B3B161A(2067469850)
        transform: esp-3des esp-sha-hmac ,
        in use settings ={Tunnel, }
        conn id: 2001, flow_id: SW:1, crypto map: test
        sa timing: remaining key lifetime (k/sec): (4435014/28483)
        IV size: 8 bytes
        replay detection support: Y
        Status: ACTIVE

     inbound ah sas:

     inbound pcp sas:

     outbound esp sas:
      spi: 0xE6433708(3863164680)
        transform: esp-3des esp-sha-hmac ,
        in use settings ={Tunnel, }
        conn id: 2002, flow_id: SW:2, crypto map: test
        sa timing: remaining key lifetime (k/sec): (4435036/28449)
        IV size: 8 bytes
```

```
        replay detection support: Y
        Status: ACTIVE

      outbound ah sas:

      outbound pcp sas:
Ronneby#
Ronneby# sh crypto engine
% Incomplete command.

Ronneby# sh crypto engine ?
  brief          Show all crypto engines in the system
  configuration  Show crypto engine config
  connections    Show connection information
  qos            Show QoS information

Ronneby# sh crypto engine conf
Ronneby# sh crypto engine configuration

        crypto engine name:  Cisco VPN Software Implementation
        crypto engine type:  software
             serial number:  00000000
       crypto engine state:  installed
      crypto engine in slot:  N/A
                 platform:  Cisco Software Crypto Engine

    Crypto Adjacency Counts:
               Lock Count:  0
             Unlock Count:  0
        crypto lib version:  19.0.0

Ronneby#
Ronneby#sh crypto engine connections active

  ID Interface           IP-Address      State  Algorithm           Encrypt  Decrypt
   1 FastEthernet0/0     172.25.4.34     set    HMAC_MD5+3DES_56_C        0        0
2001 FastEthernet0/0     172.25.4.34     set    3DES+SHA                  0      265
2002 FastEthernet0/0     172.25.4.34     set    3DES+SHA                  0        0

Ronneby#
Ronneby#sh crypto ?
  ca               Show certification authority policy
  call             Show crypto call admission info
  debug-condition  Debug Condition filters
  dynamic-map      Crypto map templates
  engine           Show crypto engine info
  identity         Show crypto identity list
  ipsec            Show IPSEC policy
  isakmp           Show ISAKMP
  key              Show long term public keys
  map              Crypto maps
```

```
  mib             Show Crypto-related MIB Parameters
  optional        Optional Encryption Status
  pki             Show PKI
  session         Show crypto sessions (tunnels)
  sockets         Secure Socket Information

Ronneby#sh crypto ke
Ronneby#sh crypto key ?
  mypubkey      Show public keys associated with this router
  pubkey-chain  Show peer public keys

Ronneby#
Ronneby#sh crypto key mypubkey ?
  rsa  Show RSA public keys

Ronneby#sh crypto key mypubkey rsa
Ronneby#sh crypto isakmp sa
dst             src             state          conn-id slot status
172.25.4.34     172.25.4.33     QM_IDLE              1    0 ACTIVE

Ronneby#
Ronneby#sh crypto ?
  ca              Show certification authority policy
  call            Show crypto call admission info
  debug-condition  Debug Condition filters
  dynamic-map      Crypto map templates
  engine          Show crypto engine info
  identity        Show crypto identity list
  ipsec           Show IPSEC policy
  isakmp          Show ISAKMP
  key             Show long term public keys
  map             Crypto maps
  mib             Show Crypto-related MIB Parameters
  optional        Optional Encryption Status
  pki             Show PKI
  session         Show crypto sessions (tunnels)
  sockets         Secure Socket Information

Ronneby#
Ronneby#sh crypto isakmp ?
  key      Show ISAKMP preshared keys
  peers    Show ISAKMP peer structures
  policy   Show ISAKMP protection suite policy
  profile  Show ISAKMP profiles
  sa       Show ISAKMP Security Associations

Ronneby#
Ronneby#sh crypto isakmp policy

Global IKE policy
Protection suite of priority 10
```

```
        encryption algorithm:   Three key triple DES
        hash algorithm:         Message Digest 5
        authentication method:  Pre-Shared Key
        Diffie-Hellman group:   #2 (1024 bit)
        lifetime:               28800 seconds, no volume limit
Default protection suite
        encryption algorithm:   DES - Data Encryption Standard (56 bit keys).
        hash algorithm:         Secure Hash Standard
        authentication method:  Rivest-Shamir-Adleman Signature
        Diffie-Hellman group:   #1 (768 bit)
        lifetime:               86400 seconds, no volume limit
Ronneby#
Ronneby#sh crypto ipsec ?
  client               Show Client Status
  policy               Show IPSEC client policies
  profile              Show ipsec profile information
  sa                   IPSEC SA table
  security-association  Show parameters for IPSec security associations
  transform-set        Crypto transform sets

Ronneby#
Ronneby#sh crypto ipsec security-association ?
  idle-time  Show this router's security association idletime settings
  lifetime   Show this router's security association lifetime settings

Ronneby#
Ronneby#sh crypto ipsec security-association lifetime
Security association lifetime: 4608000 kilobytes/28800 seconds

Ronneby#
Ronneby#sh crypto ipsec security-association idle-time
Security association idletime: unset

Ronneby#
Ronneby#sh crypto isakmp ?
  key      Show ISAKMP preshared keys
  peers    Show ISAKMP peer structures
  policy   Show ISAKMP protection suite policy
  profile  Show ISAKMP profiles
  sa       Show ISAKMP Security Associations

Ronneby#sh crypto isakmp sa
Ronneby#sh crypto isakmp sa ?
  detail  Show ISAKMP SA Detail
  nat     Show ISAKMP SA NAT Detail
  vrf     Show ISAKMP SA as per VRF
  |       Output modifiers
  <cr>

Ronneby#
Ronneby#sh crypto isakmp sa detail
```

```
Codes: C - IKE configuration mode, D - Dead Peer Detection
       K - Keepalives, N - NAT-traversal
       X - IKE Extended Authentication
       psk - Preshared key, rsig - RSA signature
       renc - RSA encryption

C-id  Local          Remote         I-VRF    Status Encr Hash Auth DH Lifetime Cap.
1     172.25.4.34     172.25.4.33             ACTIVE 3des md5  psk  2  07:52:08
      Connection-id:Engine-id =  1:1(software)
Ronneby#
Ronneby#sh crypto call admission statistics
----------------------------------------------------------------------
              Crypto Call Admission Control Statistics
----------------------------------------------------------------------
System Resource Limit:       0 Max IKE SAs:     0
Total IKE SA Count:          1 active:          1 negotiating:     0
Incoming IKE Requests:       1 accepted:        1 rejected:        0
Outgoing IKE Requests:       0 accepted:        0 rejected:        0
Rejected IKE Requests:       0 rsrc low:        0 SA limit:        0

Ronneby#
Ronneby#sh run
Building configuration...

Current configuration : 1441 bytes
!
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname Ronneby
!
boot-start-marker
boot-end-marker
!
enable secret 5 $1$MYK.$lXiezozROtbQ341jrxYVq.
!
no aaa new-model
memory-size iomem 5
!
!
ip cef
!
!
crypto isakmp policy 10
 encr 3des
 hash md5
 authentication pre-share
 group 2
 lifetime 28800
```

```
crypto isakmp key cisco address 172.25.4.33
!
crypto ipsec security-association lifetime seconds 28800
!
crypto ipsec transform-set qanit esp-3des esp-sha-hmac
!
crypto map test 10 ipsec-isakmp
 set peer 172.25.4.33
 set transform-set qanit
 match address ahbr
!
!
!
!
interface Loopback0
 ip address 2.2.2.2 255.255.255.0
!
interface Tunnel101
 ip address 10.100.200.102 255.255.255.0
 tunnel source 172.25.4.34
 tunnel destination 172.25.4.33
 crypto map test
!
interface FastEthernet0/0
 ip address 172.25.4.34 255.255.255.252
 duplex auto
 speed auto
!
interface FastEthernet1/0
 ip address 10.95.8.254 255.255.255.0
 duplex auto
 speed auto
!
router rip
 version 2
 network 2.0.0.0
 network 10.0.0.0
 network 172.25.0.0
 no auto-summary
!
ip http server
no ip http secure-server
ip route 10.2.249.13 255.255.255.255 10.100.200.101
!
!
!
!
ip access-list extended ahbr
 permit ip host 10.95.8.183 host 10.2.249.13
!
!
```

```
!
control-plane
!
!
line con 0
line aux 0
line vty 0 4
 password cisco
 login
!
!
end
```