SNORT based Intrusion Detection System for SIP/RTP VoIP Service

T0 8049

MS Research Thesis

By

Abdul Waheed
(288-FAS/MSCS/F06)

Supervised By:

Qaisar Javaid
Assistant Professor

Department of Computer Science,
Faculty of Basic and Applied Sciences,
International Islamic University, Islamabad.
(2011)

1. Snort (Software)
2. Computer networks - Security measures

A Dissertation Submitted to the

Department of Computer Science

International Islamic University Islamabad

As a partial fulfillment of requirements for the award of the degree of

MS in Computer Science

International Islamic University, Islamabad

Dated: July 30, 2011

Final Approval

It is certified that we have examined the thesis titled "SNORT based Intrusion Detection System for SIP/RTP VoIP Service" submitted by Abdul Waheed, Registration Number 288-FBAS/MSCS/F06 and found as per standard. In our judgment, this research work is sufficient to warrant its acceptance by the International Islamic University, Islamabad for the award of MS Degree in Computer Science.

Committee

External Examiner

Prof. Dr. Khalid Rashid
Advisor
Department of Computer Science
COMSATS Institute of Information Technology, Islamabad

[illegible]

Internal Examiner

Dr. Muhammad Zubair
Assistant Professor
Department of Computer Science
International Islamic University,Islamabad

[illegible]

Supervisor

QaisarJavaid
Assistant Professor
Department of Computer Science
International Islamic University, Islamabad

[illegible]

DECLARATION

I hereby declare that this work, neither as a whole nor a part of it has been copied out from any source. It is further declared that we have conducted this research and have accomplished this thesis entirely on the basis of our personal efforts and under the sincere guidance of Mr. Qaisar Javaid, Assistant Professor. If any part of this research is proved to be copied from any source or found to be reproduction of some other research work, I shall stand by the consequences. No portion of the work presented in this research work has been submitted in support of any other degree or qualification of this or any other university or institute or learning.

[illegible]

Abdul Waheed
(288-FAS/MSCS/F06)

ACKNOWLEDGEMENTS

All praises to almighty ALLAH, the most merciful and compassionate, and to The Rahmat Al-Aalmeen Holy Prophet Muhammad (Allah's grace and peace be upon him). He enabled me to carry out my research work and provide me with the strength, capacity and guidance to complete this Thesis. He showed me lights of hope when I was not able to find solution of some problem. Without His help I would have never been able to find the way to achieve my goals

To start with, I would like to express my gratitude to **Mr. Qaiser Javaid,** my Supervisor and member of project committee for his valuable guidance for the completion of this project. I thank him for the chance he gave me to work on a subject of a remarkable importance and scientific openness. He always supported me in my research work and helped me to arrange support material for my research.

Secondly, I would like to appreciate the patience, guidance, prayers and love of my dear parents that is still a valuable asset for me and were responsible for my academic career along with achievements. The support of my family always encouraged me to advance in my research work. Their prayers always ensured consistency and harmony in my life. Without their help, I surely would not be able to work in this project.

I would also like to appreciate Mr. Aziz Ahmed Shad, Kerry Braganza, Shahzad Abbasi and other friends for their kind encouragement and valuable help provided to me in my research work whenever required. It was an essential comfort during the whole time, especially drafting this manuscript.

Last but not the least I would like to express my profound thanks to my dear wife Sumra Azhar who supported me throughout the writing of this thesis and by ensuring that my son Muhammad Hasan was cared for without making him feel the deficiency of my presence whilst I complete my research.

I dedicate this thesis to my mother "Sakina", to my father "Dr. Muhammad Shafiq" and my late uncle "Muhammad Refiq Azhar"

PROJECT IN BRIEF

Project Title	SNORT based Intrusion Detection System for SIP/RTP VoIP System
Undertaken By	Abdul Waheed
Supervised By	Mr. Qaisar Javaid - *Assistant Professor*
Completion Date	June 2011
Tools Used	SNORT – version 2.8.5, GCC Compiler
Operating System	Open SUSE 11.3
Software/Packages	MySQL, POSIX C
Hardware	DELL Optiplex Pentium 4 CPU (2.40 GHz) With 1GB RAM and 40GB Hard Disk

LIST OF ABBREVIATIONS

AAA	Authentication, Authorization and Accounting
API	Application Programing Interface
ARP	Address Resolution Protocol
ASP	Active Server Pages
BPF	Berkeley Packet Filter
CODEC	Coder-Decoder
DDoS	Distributed Denial of Service
DES	Data Encryption Standard
DHCP	Dynamic Host Configuration Protocol
DIDS	Distributed Intrusion Detection System
DNS	Domain Name System
DoS	Denial of Service
ENUM	E.164 NUmber Mapping standard
FDQN	Fully Qualified Domain Name
FTP	File Transfer Protocol
GUI	Graphical User Interface
HIDS	Host Intrusion Detection System
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IETF	Internet Engineering Task Force
IIS	Internet Information Services
IMAP	Internet Message Access Protocol
IP	Internet Protocol
IPS	Intrusion Prevention System
IPSec	Internet Protocol Security
IPX	Internetwork Packet Exchange
ISO	International Organization for Standardization

MAC	Media Access Control
MIME	Multipurpose Internet Mail Extensions
MITM	Man In The Middle
NIC	Network Interface Card
NIDS	Network Intrusion Detection System
OS	Operating System
PBX	Private Branch Exchange
POP	Post Office Protocol
PPP	Point-to-Point Protocol
PSTN	Public Switched Telephone Network
QoS	Quality of Service
RFC	Request For Comments
RSVP	Resource Reservation Protocol
RTCP	Real-time Transport Control Protocol
RTP	Real-time Transport Protocol
SAP	Session Announcement Protocol
SDP	Session Description Protocol
SIP	Session Initiation Protocol
SIPS	Secure SIP
SLIP	Serial Line Internet Protocol
SMB	Server Message Block
SMS	Short Message Service
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Object Access Protocol
SPIT	Spam over Internet Telephony
SSH	Secure Shell
ST-II	Internet Stream Protocol, version 2
TCP	Transmission Control Protocol
TFTP	Trivial File Transfer Protocol

TLS	Transport Layer Security
UA	User Agent
UAC	User Agent Client
UAS	User Agent Server
UDDI	Universal Description Discovery and Integration
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
UTF	UCS Transformation Format
VoIP	Voice over Internet Protocol
VXML	Voice Extensible Markup Language
WSDL	Web Service Definition Language
WWW	World Wide Web
XML	Extensible Markup Language

Table of Contents

Abstract

With VoIP we inherit security problems associated with the IP as well as new VoIP specific ones. Both, Signaling protocols such as H.323 and SIP (Session Initiation Protocol), and media transport protocols such as RTP and RTCP, could be the target of a wide set of attacks. Our research is to exploit the strengths of one of the powerful Intrusion Detection System, Snort. Along with many other features it can provide administrators with enough data to make informed decisions on the proper course of action in the face of suspicious activity. But it severely lacks comprehensive intrusion detection capabilities for SIP-VoIP. This makes Snort useless for VoIP security implementation. In this thesis we have studied different aspects of VoIP security issues. The issues have a large variety as VoIP systems suffer security laps for both VoIP protocols (SIP, RTP, RTCP, etc.) and other supporting protocols (DNS, IP, etc.).

We studied different techniques and methods of Intrusion Detection and narrowed down our focus to the techniques which are most significant for VoIP intrusion detection. We have noticed that the techniques already developed for VoIP are focused on a single point of VoIP network. Because many elements are involved to get the work done, we found no coordination of these elements for intrusion detection in previous techniques. Moreover some important attacks detection is overlooked e.g. MITM, SPIT, etc. Therefor to address such issues we have proposed an architecture by which every element of VoIP infrastructure can be monitored independently for intrusions. These elements will share their information with central Intrusion Detection System for further analysis. Our focus will be on Central IDS system which will do correlation of information collected from VoIP elements and detects attacks by looking overall picture of communication. We shall see that this is difficult task if single element or point of network is relied for such intrusions. We shall also propose some structural changes in Snort architecture so that it can effectively accommodate different techniques.

We have tested Central IDS Server of our technique as a proof of concept and implement it on three attacks of different categories. We have seen that our method is a useful way of data correlation, which has been collected from different components of SIP – VoIP system. The results are presented and compared with one of previously developed technique.

Chapter 1
Introduction

1. Introduction

This chapter is divided into two parts. First part gives an introduction of VoIP technology, and protocols used in this technology. As VoIP is an internet technology and carries data into packet switched network, some coding/decoding mechanism is required like other communication mechanisms. We shall present how the data is processed in TCP/IP protocol structure. The emphasis of this thesis will remain on SIP protocol which is de-facto standard of end-to-end signaling over the internet, so it will be discussed in detail. SIP is a signaling protocol and plays an important role in multimedia communication. It provides the key mechanism of establishing, routing, switching and accounting the call, like billing. In VoIP this protocol creates innovative services and exclusive applications. That is why it is mandatory to understand what this protocol can offer and what are its limitations. Lastly media description and media transfer protocols are discussed.

The second part of this chapter presents an overview of intrusion and intrusion detection concepts. This overview is important because of the fact that all intrusion detection architectures and methodologies are based on these basic concepts.

Rest of this chapter is organized as follows: In Section 1.1 we shall present motivation and challenges of this research work. Section 1.2 introduces VoIP technology, Section 1.3 overviews SIP and its different aspects. Media Description and Media Transfer protocols are discussed in Section 1.4. Different Concepts related to Intrusion Detection are presented in Section 1.5. Section 1.6 illustrates our Thesis Outline.

1.1 Motivation and Challenges

The VoIP applications are emerging today as an important component in business and communication industry. VoIP offers the convenience of one network administration, advanced calling features and unified communications so that users can access voice mail, email, text messages, faxes, and other incoming communications using a single interface.

However, security continues to be an important issue and obstacle in implementing VoIP for many individuals and companies. With VoIP we inherit security problems associated with the IP as well as new VoIP specific ones. Both, Signaling protocols such as H.323

and SIP (Session Initiation Protocol), and media transport protocols such as RTP and RTCP, could be the target of a wide set of attacks, ranging from eavesdropping, denial of service, fraudulent usage and SPIT (Spam over internet telephony). Lot more components needs to be protected for comprehensive security. As VoIP is associated with real time propagation of data, therefore quality of service (QoS) issues are essentially considered while implementing security solutions.

There are many tools available in the market, which are in use as Intrusion Detection System. Many methods have been developed that are being accommodated in such tools having their own strengths and weaknesses. Snort is one of such important tool. But, along with others, Snort has very limited support for VoIP, especially SIP. This makes it unfit for the use in VoIP environment. Therefore Snort needs some enhancement in its architecture or technique which covers the broader spectrum of attacks. We shall discuss the challenges which are in the way of Snort and VoIP security in detail in sections 3.3, 4.3 and 4.6.

1.2 VoIP

Voice over Internet Protocol (VoIP), also known as IP Telephony or Internet Telephony, is a technology optimized for the transmission of voice/audio through Internet and other packet switched network. VoIP system takes voice signals, converts them into digital audio signals, apply some compression techniques to reduce the size of samples, encapsulates into data packets and send the data packet stream over IP.

Along with many features similar to PSTN, VoIP can provide many services that are difficult or expensive in traditional telephone networks. Some of these are conference calling, call forward, call divert, automatic redial, etc. Some security protocols (like encryption) can be introduced in VoIP to make call tapping difficult unlike in PSTN. This technology also gives independence of location as you can connect to network from any location, only a high speed internet connection is required. It can also be integrated with other services that are available on internet like file sharing, video conversation, text messages, screen popup, conference calling, address book, and many other services. The feature of VOIP that has attracted the most attention is its cost-saving potential. By moving away from the public switched telephone networks, long distance phone calls become very inexpensive.

VoIP data processing can be divided into four logical steps [1]: Signaling, Encoding, Transport and Gateway Control.

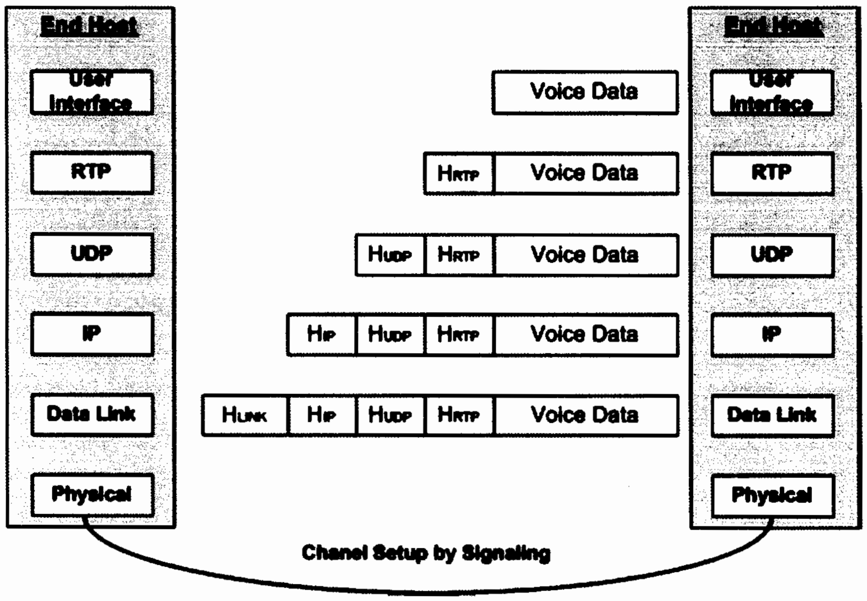


Fig: 1.1 – Data packet encapsulation by protocol headers (Encoding/ Decoding) [1]

A typical VoIP network can be divided into five major components:

- **IP Network:** This is a network for actual transmission of packets from source to destination and work as switching component.
- **End User Devices:** These end devices are used for interaction with users, like initiate or receive calls e.g. soft phone or hard phone, Consoles.
- **Call Processing server/PBX:** This component provides the core functionality of the whole network and manage all VoIP control connections.
- **Media Gateway/Gatekeepers:** Houses the signaling and control services that coordinate the media gateway functions. It also houses the traditional CODEC functions which convert voice calls or conversation into IP sessions which are then transported by RTP over UDP.
- **Session Border Control:** This functionality provides real-time session based traffic control at signaling and transport layers, enabling voice session to cross network borders and domains.

1.3 SIP

Session Initiation Protocol (SIP) is a signaling protocol used to create, modify and terminate multimedia communication sessions like voice or video over the internet. Modification can involve changing addresses or port, changing call features, inviting more participants, adding or deleting media streams, etc. A session can be a two way session or multi-party session like multi-media conference session of one or several media streams. SIP is an Application Layer protocol in TCP/IP model and brought forward by IETF as RFC 3261. Although SIP is still in evolution phase, VoIP community has adopted SIP as protocol of choice.

SIP is a text based request response protocol just like HTTP and SMTP. It also inherits many features from them. Due to text based nature humans can read and analyzes SIP messages. SIP is transport independent and it can work over UDP or TCP depending upon requirement as it is transport independent.

1.3.1 SIP - Advantages & Limitations

Because of its simple and text based nature, SIP is easy to understand, extend and implement. As discussed earlier, SIP messages are very similar to HTTP and much of its syntax in message headers is reused. We can see many headers of SIP having the same meanings as that of HTTP. One example is error "404", error code for address not found. [9]

Similar to HTTP, SIP also used the addressing scheme of SMTP protocol. A SIP address sip:address@domain.com has exactly the same structure as an email address. Similarly SIP also makes use of other components of existing internet structure, such as Domain Name Service (DNS) is used for domain name translation. [7]

Using SIP, service providers can freely choose among different standards and technology. Moreover, users can locate and contact each other regardless of media content and number of participants. At the start and even in middle of a session, SIP negotiates session features so that capabilities can be best utilized. Users can be added or dropped in the middle of a session.

The IETF philosophy is simplicity: specify what you need to specify. SIP is much of this mould. Despite of its many advantages, SIP is not a cure-all. It is neither a session description protocol nor does it provide any conference control. SIP takes services from other protocols for its job to done. For example SIP uses SDP to describe the characteristics and capabilities of the end devices, and RSVP for QoS [7]. SIP works nicely with other protocols like SOAP, HTTP, XML, VXML, WSDL, UDDI, SDP and an alphabet soup of others [10].

There are two fundamental design assumptions for SIP [10]:

- Reusing the existing protocols and protocols design structure *(which is actually mature enough)*, like adopting addressing scheme from SMTP and using DNS for address translation.
- Maximizing interoperability so that it would be easy to bind SIP functions to existing protocols and applications like Web Browsers, E-mail, Messengers, etc. SIP is doing this by obeying the modular philosophy.

1.3.2 SIP - Network Entities

A SIP session utilizes four major components of architecture:

- **SIP User Agents:** These are end devices which are used to create and terminate SIP sessions, e.g. cell phones, Personal Computers, and other hand held devices. A User Agent (UA) is composed of two types: User Agent Client (UAC) – initiates a call; User Agent Server (UAS) – receives call.
- **SIP Registrar Server:** It is a database of the locations of User Agents within a domain. It communicates with SIP Proxy Server to send and receive participants IP address and other information. Users upload their current location to this server for use of SIP Proxy Server. Purpose of the location database is to map sip:bob@b.com to something like sip:bob@1.2.3.4:5060.
- **SIP Proxy Server:** It accepts session requests made by a UA and query a SIP Registrar server to obtain destination addressing information. It then forwards the request to recipient if it is in the same domain or forward to some other proxy server if destination is another domain. It can use DNS to map domain names in the SIP requests. In case if user calls to some PSTN number from

SIP phone, this proxy server uses ENUM to map global (E.164) telephone number to SIP URI's. SIP Proxy Server also authenticates and authorizes the users of the service and implements call routing policies. There are two main types of proxy servers: Stateless proxy servers just forward the received requests and never transmit the last message; while stateful proxies maintain a state of ongoing transactions and keep it until the transaction finishes.

- **SIP Redirect Server:** It allows SIP Proxy Servers to direct SIP session invitations to other domain. These servers may reside on the same hardware domain or some other machine.

1.3.3 SIP – Session Anatomy (Call Example)

There can be two scenarios of session establishment between UAs; Establishing SIP Session within the Same Domain and Establishing SIP Session in Dissimilar Domains. Let's demonstrate both of these:

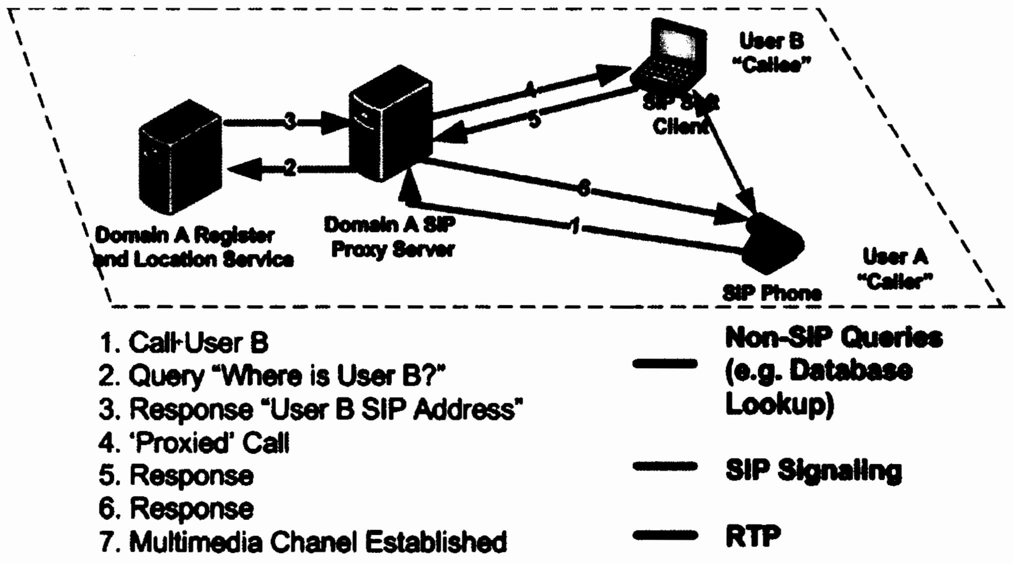
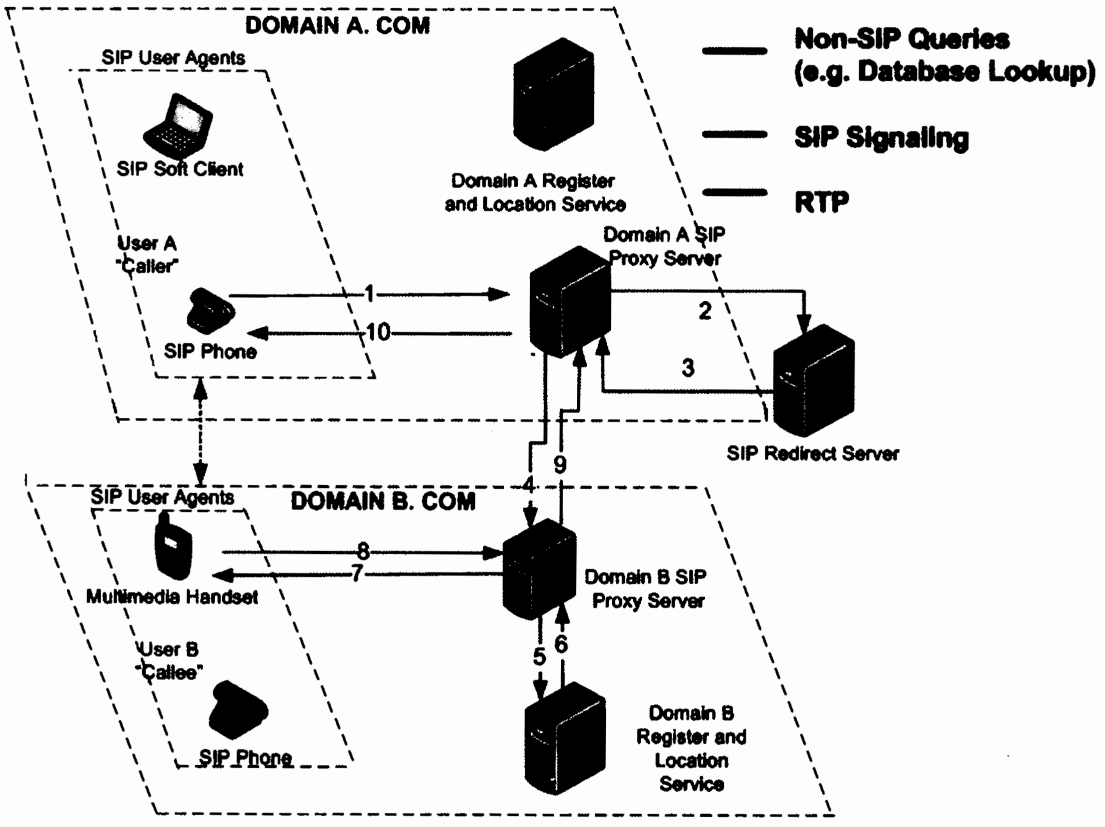

User B "Callee"
SIP Soft Client
Domain A Register and Location Service
Domain A SIP Proxy Server
SIP Phone
User A "Caller"
1
2
3
4
5
6
1. Call User B
2. Query "Where is User B?"
3. Response "User B SIP Address"
4. 'Proxied' Call
5. Response
6. Response
7. Multimedia Chanel Established
Non-SIP Queries (e.g. Database Lookup)
SIP Signaling
RTP

Figure-1.2: SIP Signaling – Users within same domain [8]

a) Establishing A SIP Session Within the Same Domain

Let us suppose that User-A is on phone and User-B is on PC running Soft Client. Upon powering up, both users will register their availability and IP addresses to the SIP Registrar Server. User-A initiates a call and tell SIP Proxy Server he wants to contact User-B. SIP Proxy Server then contacts ISP Registrar Server to get status, location and IP of User-B. SIP Proxy Server then relays the invitation to User-B. This invitation also includes the information of media (using SDP) that User-A wants to use. User-B then informs Proxy Server whether A's invitation is acceptable and he is ready to receive messages. Supported feature negotiation will also be done at this point. Proxy Server communicates the response from User-A to User-B and then a point-to-point RTP link is established between the users. Below is the diagrammatic representation of this procedure.



1. Call User B
2. Query "How to get to User B?"
3. Response "Address of Proxy Controller for Domain"
4. Call "Proxied" to SIP Proxy for Domain B
5. Query "Where is User B"
6. User B's Address
7. Proxied Call
8. Response
9. Response
10. Response
11. Multimedia Channel Established

Figure-1.3: SIP Signaling – Users in different domains [8]

b) Establishing A SIP Session Within Different Domains

In this situation, the difference is that User-A and User-B are located in different domains. For s SIP session the Proxy Server in Domain A recognize that User-B is outside its domain. SIP Proxy Server then query SIP Redirect server for User-B's IP address. This SIP Redirect Server can reside in either Domain. This server responds with User-B information to SIP Proxy Server of Domain-A which then initiates SIP session invitation to SIP Proxy Server in Domain-B. SIP Proxy Server in Domain-B delivers this invitation to User-B and this user will respond with acceptance to User-A along the same path from where invitation has arrived. The diagrammatic description of the session is described above.

c) SIP Call Example

Figure-1.4 shows a typical example *(taken from RFC 3261 [9])* of a SIP message exchange between two users, Alice and Bob. Alice initiates an INVITE request to BOB. The subsequent messages are shown below. The Session Description Protocol (SDP) is responsible for media description negotiation between two parties. In this example, SDP message is embedded into the INVITE message sent from Alice, while Bob inserts its own in 200 OK message (F9 in below figure).

```
                     atlanta.com  . . . biloxi.com
                 .      proxy              proxy     .
               .                                       .
       Alice's  . . . . . . . . . . . . . . . . . . . .  Bob's
      softphone                                        SIP Phone
         |                |                |                |
         |    INVITE F1   |                |                |
         |--------------->|    INVITE F2   |                |
         |  100 Trying F3 |--------------->|    INVITE F4   |
         |<---------------|  100 Trying F5 |--------------->|
         |                |<-------------- | 180 Ringing F6 |
         |                | 180 Ringing F7 |<---------------|
         | 180 Ringing F8 |<---------------|     200 OK F9  |
         |<---------------|    200 OK F10  |<---------------|
         |    200 OK F11  |<---------------|                |
         |<---------------|                |                |
         |                       ACK F12                    |
         |------------------------------------------------->|
         |                   Media Session                  |
         |<================================================>|
         |                       BYE F13                    |
         |<-------------------------------------------------|
         |                     200 OK F14                   |
         |------------------------------------------------->|
```

Figur-1.4: SIP session setup example with SIP trapezoid [9]

The INVITE (message F1 in Figure-1.4) might look like this [9]:

```
INVITE sip:bob@biloxi.com SIP/2.0
      Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bK776asdhds
      Max-Forwards: 70
      To: Bob <sip:bob@biloxi.com>
      From: Alice <sip:alice@atlanta.com>;tag=1928301774
      Call-ID: a84b4c76e66710@pc33.atlanta.com
      CSeq: 314159 INVITE
      Contact: <sip:alice@pc33.atlanta.com>
      Content-Type: application/sdp
      Content-Length: 142

(Alice's SDP not shown)
```

The *Via* contains the address at which Alice is expecting to receive responses of this request, where branch parameter identifies this transaction. *Max-Forwards* represent maximum number of hops that this request can make to reach its destination. *To* contains the address on which original request is heading for. *From* contains the URI of request initiator. The *Call-ID* and tag in *From* header *(randomly chosen)* identify the dialog of the message. The callee adds its tag in To header to fully identify this dialog. Note that *Call-ID* contains a globally unique identifier generated by the combination of a random string and the soft phone's host name or IP address. *CSeq* contains a sequence number and method name. The sequence number is incremented with each new request. *Contact* contains SIP URI that represents the direct route to Alice, usually composed of username at a Fully Qualified Domain Name (FQDN). *Content-Type* shows what type of message body it has and *Content-Length* shows message body in bytes. The detail of this transaction can be found in RFC3261 [9].

d) SIP Methods & Responses

SIP RFC3261 defines the following six Request Methods:

SIP Method	Description
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]

There are some other SIP Method Extensions which can be found in other RFC's. Once the INVITE has been passed to the INVITE client transaction, the UAC waits for responses for the INVITE. There are the following categories of responses. Detailed description can be found in RFC 3261 [9].

- 1xx Informational (e.g. 100 Trying, 180 Ringing)
- 2xx Successful (e.g. 200 OK, 202 Accepted)
- 3xx Redirection (e.g. 302 Moved Temporarily)
- 4xx Request Failure (e.g. 404 Not Found, 482 Loop Detected)
- 5xx Server Failure (e.g. 501 Not Implemented)
- 6xx Global Failure (e.g. 603 Decline)

1.3.4 SIP – Registration Example

As we have already seen in section 1.3.3 that SIP proxy server requires the status, location and IP address of UA so that it can send request to it. For this purpose, the terminal devices or User Agents need to register themselves to Registrar server whenever they come to life. For example Bob is on SIP phone device. When he will connect its SIP phone to internet, it will register itself with Registrar Server of his VoIP domain. A REGISTER message will be sent with Address of Record in TO header. The CONTACT header will contain the IP address where Bob's phone is reachable. Registrar server maintains a location database, may be in the form of table or some other mapping function, where it stores a binding relationship between logical and physical addresses so that proxy server having access to that database can access these records to rout incoming calls to Bob. There is an Expire parameter in CONTACT header which tells the server how long it should keep this record into its database. After that time, a new REGISTER request will refresh this record. In case of soft phone, like a software program on computer e.g. Skype, the registration process is quite similar. When a user login to Skype, it will register itself to Registrar server of Skype domain.

In most deployments, Registrar server asks the user for authentication. The proposed mechanism in RFC3261 is HTTP Digest based authentication. HTTP Digest is a stateless challenge-response protocol in which nonce value is used to challenge the target.

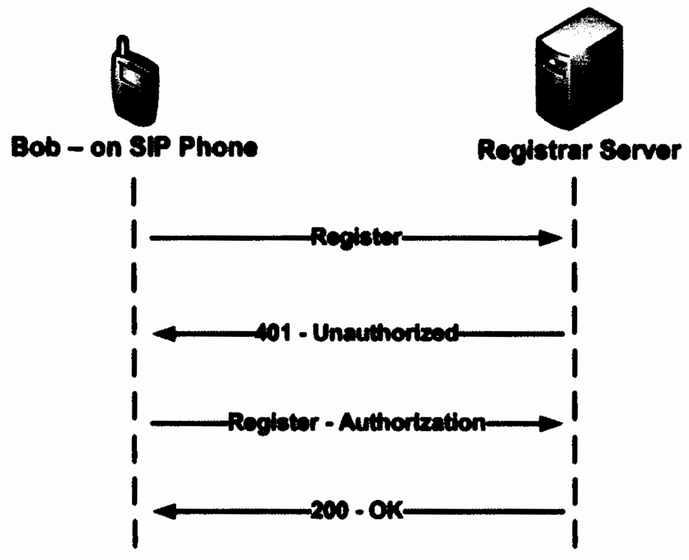


Figure-1.5 : Registration Example

In our example (Figure-1.5), when Bob send request of REGISTER, the Registrar server responds with 401 Unauthorized. This response contains WWW-authentication challenge composed by a realm, a nonce and an encryption algorithm e.g. MD5 hashing function. This header will look something like this:

```
WWW-Authanticate: Digest
      realm="biloxi.com",
      qop="auth,auth-int",
      nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
      opaque="5ccc069c403ebaf9f0171e9517f40e41",
      algorithm=MD5
```

In response to this 401, Bob will resubmit REGISTER request with credentials by including the Authorization (composed by username, password, realm and nonce). Important thing to note is that both parties can challenge each other but in actual implementations only Server authenticates the user not the other way around. The "Digest" authentication mechanism described above provides message authentication and replay protection only, without message integrity or confidentiality.

1.4 Media Description & Media Transfer Protocols

Apart from SIP, some other protocols are also involved in media description, media transfer and Quality of Service QoS. Some of these important protocols are described below to have an idea how VoIP use them to get the work done.

1.4.1 SDP

Session Description Protocol (SDP) is intended for describing multimedia sessions brought forward IETF standard as RFC2327 [11] and revised as RFC4566 [12]. The SDP is used for session announcement, session invitation and other forms of multimedia session initiations to allow the recipients of a session description to participate in the session. Because of its simple and extensible nature, SDP can work with many other protocols like SAP, SIP, RTSP, HTTP, email using MIME extensions. So, SDP is meant to be a general purpose protocol which can be used with a wide spectrum of network environments and applications.

An example of SDP body of a SIP message is given below:

```
v=0
o=bob 22239 77624 IN IP4 lab.supernova.org
s=Conference Session
i=A Seminar on the session description protocol
c=IN IP4 100.101.123.101
t=0 0
m=audio 17333 RTP/AVP 31
a=rtpmap:3 PCMU/8000
a=rtpmap:0 GSM/8000
a=rtpmap:7 PCMA/8000
a=rtpmap:101 telephone-event/8000
```

SDP usage includes [12]:

- Multicast Announcements - For a client to announce a conference session to send packets to a multicast address and port using SAP.
- Email and WWW Announcements – Alternative means for conveying session description by using MIME content type “application/sdp”. This enable automatic launch of application.

SDP messages are text based using ISO 10646 character set in UTF-8 encoding. The SDP messages include [12]:

- Session name and purpose
- Time of active session
- Media involving the session
- Information to receive the media

1.4.2 RTP

RTP is an application layer protocol that defines standard packet format which provides end to end delivery services for data with real-time characteristics, like audio or video, over unicast or multi cast services. It was developed by IETF and published in 1996 as RFC1889 which was made obsolete in 2003 by RFC 3550.

Although there is no restriction for RTP to use UDP or TCP on transport layer, applications typically run on top of UDP to make use of its multiplexing and checksum services [15]. Applications using RTP are less sensitive to packet loss but typically very sensitive to delays, so UDP is a better choice than TCP for such applications. However RTP may be used with other suitable underlying network or transport protocols. Originally it was designed as a multicast protocol for data transfer to multiple destinations, but since then it has also been used in many unicast applications. The target of RTP is to facilitate delivery, monitoring, reconstruction, mixing and synchronization of data streams.

RTP does not provide any mechanism for timely delivery or any QoS guarantees. Rather it uses RTCP as better half to monitor QoS and to convey information about participants in ongoing sessions. RTCP also provide mechanism to arrange received packets in correct order. It also provides information about reception quality for application to do local adjustments like whether application could decide to lower data rate in case of congestion [16].

The goals of developing RTP are to have a protocol which is lightweight (specification and implementation), flexible (provide mechanisms rather dictating algorithms), protocol neutral (can use UDP/IP, ST-II, IPX, etc), scalable (unicast, multicast), separate control/data, and security (support for encryption and possibly authentication).

Although RTP is relatively a new protocol, it is widely used in common applications like Real Network's RealPlayer, Apple's QuickTime and Microsoft's NetMeeting.

RTP Security:

Although many media streams on internet are publicly available, some of them, like video or audio conferencing, usually require confidentiality. In some cases authentication of origin of stream and integrity is also required. Some security mechanisms are shown below. Details can be found in [17].

- To deal with confidentiality, RFC 1889 and RFC 3550 provide flexible facility for encryption of RTP packets. Packet can be split into two parts; encrypted and unencrypted. The encryption algorithms used are DES or CBC, so RTP clients must support them. To prevent known plain text attacks, RTCP headers are encrypted with 32bit random prefix. RTP also support other algorithms for encryption.
- RTP does not specify any standards for authentication rather it rely underlying layers for sophisticated authentication.
- Integrity is verified by wisdom checking decrypted headers. These checks verify known field values like protocol version number, packet length, payload type etc.
- RTP limits issuing certain commands like BYE. This is done when authentication of RTP is ensured by some other means.
- Key management feature is done by using MD5 algorithm which drives key from password. More complex key management is handled by other layers.

1.4.3 SAP, RTCP

Session Announcement Protocol (SAP) is a protocol used to broadcast or announce multicast conference session information. For this announcement, the session creator periodically multicast UDP packets to well-known multicast address and port. SAP typically uses SDP for session descriptions. People who want to know which sessions are going to be active will simply listen to the same well-known multicast group, and receive those announcement packets. More information can be found on RFC2974 [18].

Real-time Transport Control Protocol (RTCP) is an IETF standard defined in RFC3550 [19]. It is based on periodic transmission of control packets to all the participants of the session. This usually used separate port number on UDP as underlying protocol. It provides feedback on Quality of Service in media distribution. RTCP allows the parties of RTP session to exchange statistics within Quality reports like jitter, number of packets sent, number of packets received, lost packets, etc.

1.5 Intrusion Detection Concepts

It is important to understand what really intrusion is. Webster's Dictionary defines an intrusion as *"The act of thrusting in, or of entering into a place or state without invitation, right or welcome"*. In simple terms, intrusion is any system or network activity that is considered unauthorized. Legally, there are no clear and universal standards of intrusion. Infect laws of different countries define such terminologies differently. Many countries don't even have any legislation about computer or cybercrime. Discussion of the legal aspects of Intrusion is out of the scope of this research.

There can be two forms of intrusion: (1) A legitimate user of a system in an organization trying to escalate his privileges and gain greater access to the system than he has been allowed; (2) A remote and unauthenticated user outside the organization trying to compromise a running service to do unauthorized activity. This activity can be to create an account on a system, install a virus running wild your e-mail system, or many other similar scenarios. Intrusion can take many forms, including *(but not limited to)*:

- A virus, worm or Trojan-Horse program that is downloaded from internet through some email or some embedded Active X control or some scripting program.
- A password stolen by sniffer, shoulder sniffing, brute force guessing, cracking the password by some other method or by some key logger program installed on a victim's computer.
- Hijacking a terminal or file transfer session that does not use encryption, e.g. old style telnet, ftp, IMAP, or POP email, etc.
- Exploit vulnerability in a network service or protocol, like FTP, Apache or Microsoft's IIS, SSH, name server, etc.
- Physically accessing a computer and then logging into it as an administrator by some means

There are numerous reasons why your organization's IT assets could be under obstruction. For example Bandwidth (can be used for DDoS), Disk Space (for setting up server for pirated stuff), Political or Emotional Motivation (defacement of opponent country's government website), Financial Gain (stealing credit card numbers), Industrial Spying (Stealing business secretes), Hacktivism (to draw attention), Copycatting, Resentment, or just for fun. Details can be found in [37] and [38].

Intrusion Detection is the practice of spotting threats and, more precisely, attacks to IT assets. These assets may include computers, databases, networks and other devices. The system doing this operation is called Intrusion Detection System (IDS). An IDS monitors resources and upon some intrusive behavior, it perform necessary actions that is pre-configured onto it. IDS provide an internal audit component of a robust security design and policy. They let you know when you are being attacked. You can see the attacks that failed and the attacks that succeed and get real-time notifications of attempted attacks. A network or system administrator can perform detail analysis by inspecting its logs.

Intrusion Detection System (IDS) is different from Firewall. Firewall is usually first line of defense in any network whereas IDS is considered as second level. Firewalls are primarily designed to deny or allow traffic to access a network on the basis of filter, not to alert administrators of malevolent activity. We shall not discuss this topic in detail and focus us on IDS and its characteristics. Although IDS is a vast topic but we shall present some detail of few important concepts related with IDS.

1.5.1 Where to place IDS in your network

This is very important to decide where should an IDS be placed which could monitor the desired asset effectively. An IDS can be classified, on the basis of its placement in a network, into three categories:

a) Network IDS (NIDS):

In this type, an IDS monitors the entire network, network segment or subnet. IDS usually do this by sniffing the network traffic and changing the NIC into promiscuous mode. In promiscuous mode, NIDS can eavesdrop on all communication on the network segment. If all the devices are connected to a hub, all packets are automatically transmitted to all the ports and hence NIDS also be able to get those packets. But when there is Switch, the port on which NIDS is connected should be configured into monitoring mode or use SPAN Port. Wire taps can also be used for this purpose.

The advantage of NIDS is that it does not add any load on the systems or networks it is monitoring. It is also usually stealth to an attacker as he/she doesn't even know it is there. On downside, NIDS is maxing out the bandwidth on the span port itself. If you have 20 100MB ports spanning to one port, you are filling up backplane. In that case Wire Taps are recommended.

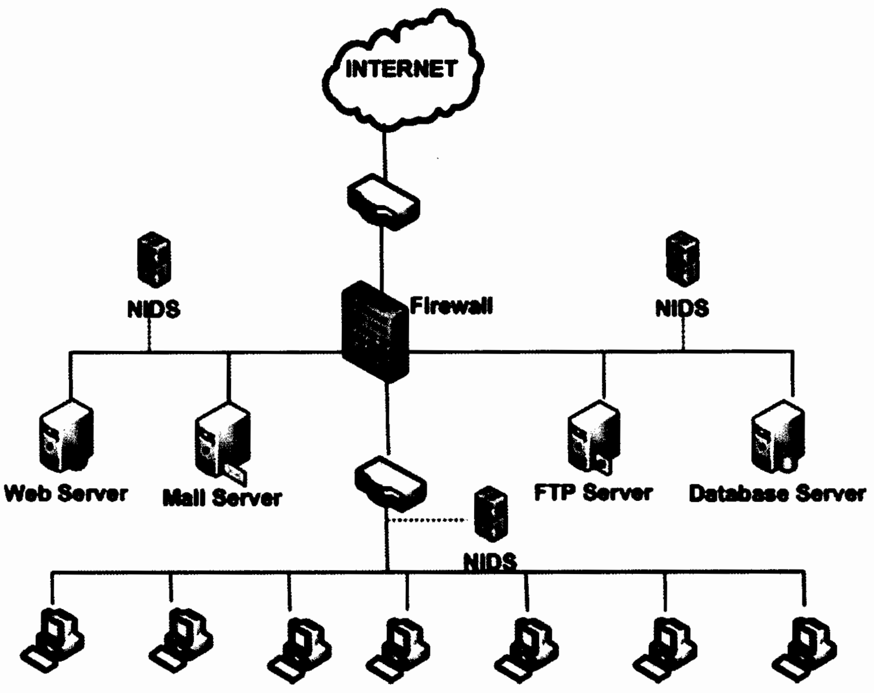


Figure- 1.6: Three NIDS placed in a network on different segments

In Figure-1.6 a network is shown with three NIDS placed at three different strategic network segments and can monitor network traffic for all the devices on that segment.

b) Host IDS (HIDS)

Host-based IDS or HIDS protects the system on which it is installed. It has nothing to do with the other network segments as that of NIDS. Moreover it does not need NIC to operate in promiscuous mode as only network traffic meant for the particular host needs to be checked for attacks. There are many advantages of HIDS over others. The important one is that it doesn't need to check the whole network and a rule set specific for that host can be easily fabricated which is more effective.

For example if HIDS is installed on an Apache Web Server and it is the only service it provides, we need only to fine tune our rule set for Apache web server exploits. It will decrease false positives, hence enhancing the performance and effectiveness of IDS many folds. Other major advantage of HIDS is its capability to detect specific changes to the files and operating system of its host. It can monitor file sizes and checksums to ensure that crucial files are not maliciously changed without someone noticing. On the bigger picture, it can monitor all the activities which are happening on the host system.

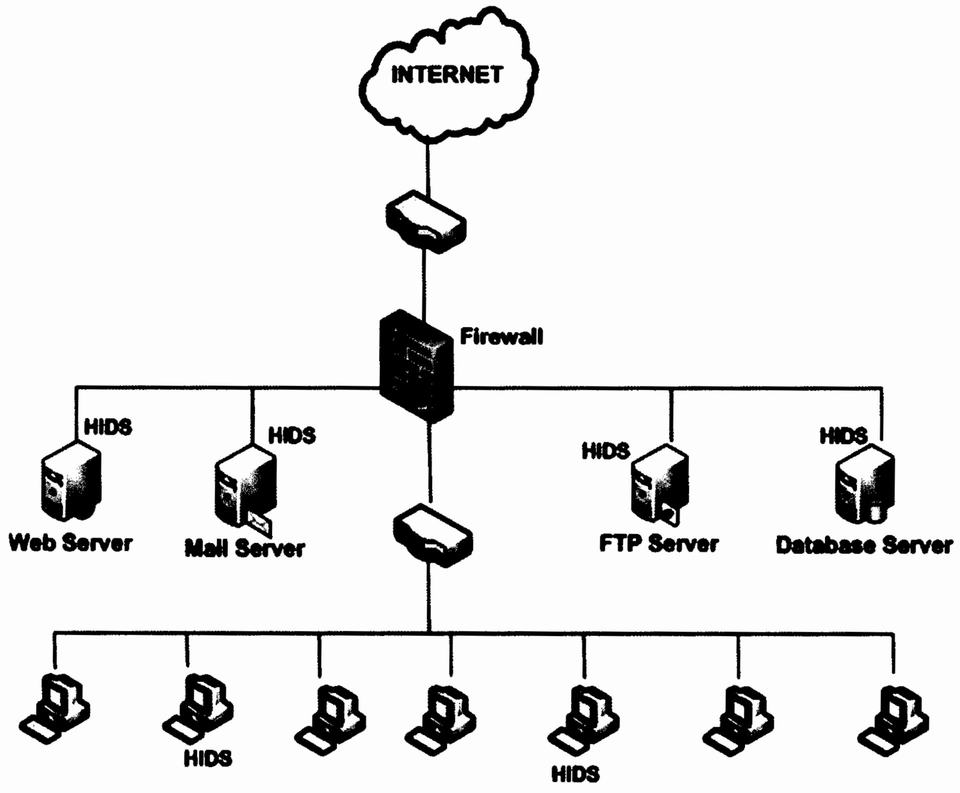


Figure- 1.7: Network with HIDS's installed on servers and hosts

On the downside, you have to choose HIDS that fits in your operating system. If you have many operating systems in a network, it is difficult to use HIDS from the same vendor. Moreover, HIDS will add load on the host on which it is configured as HIDS will consume reasonable amount of resource. It may really become a nightmare if it is one on a busy network server. It is also a challenging issue to maintain a large network of systems with many HIDS's without some centralized management system. Figure-1.7 shows a network using HIDS on specific servers and host computers.

c) Distributed IDS (DIDS)

A Distributed IDS (DIDS) system is a combination of NIDS or HIDS sensors or both, distributed across the whole enterprise all reporting the centralized management and correlation system.

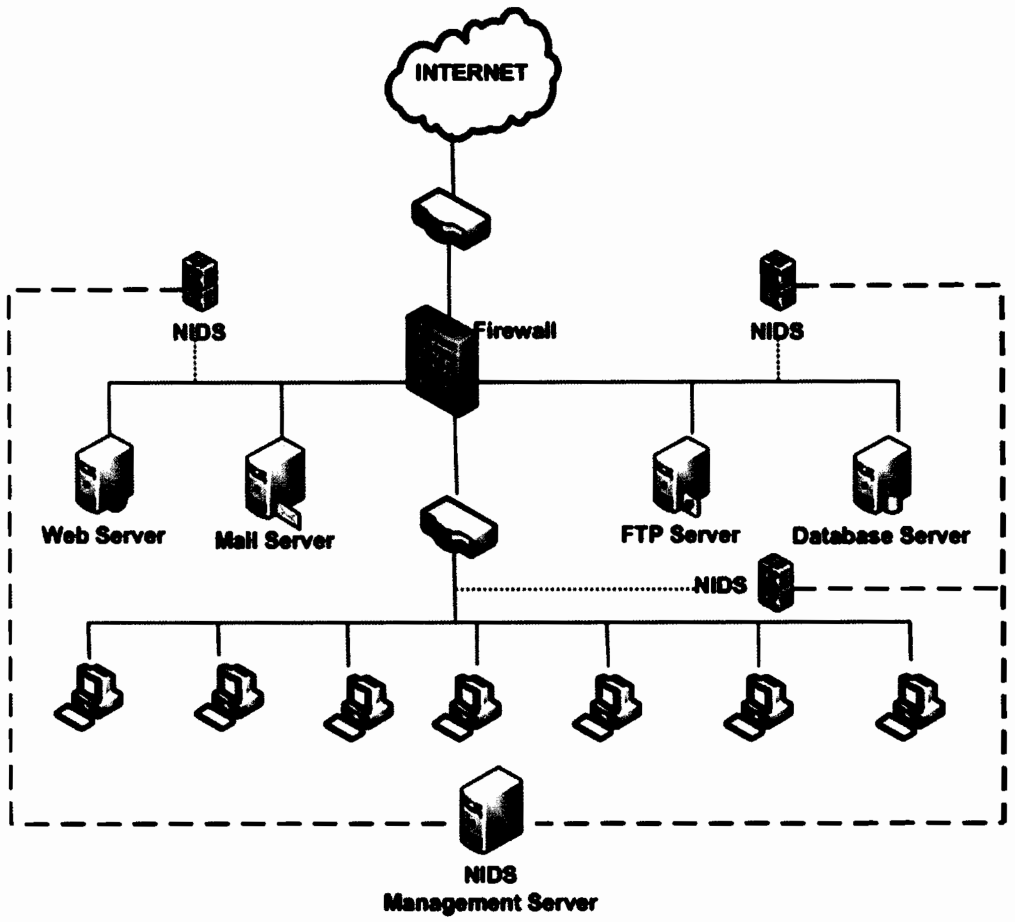


Figure- 1.8: Network with Distributed sensors connected with centralized management system

Sensors collect the attack logs and transmit them to the central server where these can be stored in a central database. New attack signatures are manipulated on the central server and then distributed to the sensors as required on the need basis. The rules of each sensor can be customized to fulfill its individual requirements. Alerts are forwarded to some messaging system. Figure-1.8 shows a DIDS system. Here four sensors are connected with one centralized management system. The network transactions between sensors and manager can be on secure private network or on the existing infrastructure.

1.5.2 What an IDS can gather

Intrusion detection works by gathering information from system and analyzing it for unusual or unexpected events. This data can later be used for prevention, preemption, deterrence or deflection of network traffic and system access. Stephen Northcutt in [37] describes this issue into three categories:

a) Application-Specific Information

All types of IDS are able to watch some sort of application-specific information. For example an NIDS will watch all the traffic which is being transmitted over the wire it is connected with. If the traffic is in clear text, like HTTP or FTP, it should have no problem in matching it against predefined criteria. Even if the traffic is in binary format and packet payload is known, it still can be matched with rules in signature-based rule matching system. For encrypted traffic, host based IDS works pretty well.

b) Host-Specific Information

An HIDS can see the host specific information on which it is installed. As most of the HIDS don't observe all the activities on a particular host, but they can see the behavior they want to monitor, from file creation, system calls, remote procedure calls, call to loop back interface, etc. It is a common practice for IDS to create a local database of system states (as normal activities) at the time of installation (mostly), so that any deviation from baseline can be monitored.

c) Subnet-Specific Information

Mostly, networks have common patterns to their traffic flow. For example, if an email server is installed on some machine in an organization's network, one should not be surprised to see SMTP traffic over the network and especially for that particular server. Similar is the case if you have some web server installed on your network. As time passes, a decent IDS should be tuned to recognize the expected behavior of the subnet on which it resides and sending alerts otherwise. Another valuable component of the subnet traffic is the Layer-2 protocol mapping. Address Resolution Protocol (ARP) is used to map

MAC address to the IP address on the local network. ARP spoofing is a potential threat. This type of traffic can be viewed at the subnet level, depending on the network topology. When network traffic crosses router, the MAC address changes and it is not possible to check ports and locations of MAC address. So this traffic should not go without examination. Therefore if Layer-2 data needs to be captured, NIDS should be on the same subnet on which the machine is located you want to monitor.

1.5.3 How an IDS watches your network?

Without an effective method of collection of data for analysis, any IDS is worthless for security. There can be many possible ways for an IDS to collect data. Some common methods are mentioned in [37]. Each has its own merits and demerits. In an article at ITSecurity website [38] they have mentioned a checklist of IDS actions that it can do to monitor some activity. This list is mentioned here without any detail:

- Activity Monitoring
- Configuration Check
- General Machine Check
- File Authorization Check
- Hidden File Check
- Log File Examination
- System Call Monitoring
- File System Watching
- Packet Sniffer Check
- Password Files Check
- Scheduled Processes
- Services Check
- System Binaries Check
- System and File Integrity Assessment

1.5.4 What an IDS does when it sees an attack?

Although most modern IDS concentrate on automated traffic filtering, blocking or disconnects as a last resort, they have some sort of automated response capabilities. Some IDS also claims to counterstrikes against the attacker, but it is not the preference. The best practice is to identify the attack and back-trace them. There are different approaches to what IDS really do when it identify an intrusive activity. Classically we can divide responses into three categories [37]:

a) Passive Response:

In this type, IDS's are configured to monitor the activities, log to some files and send alerts to administrator if sees some intrusion. These alerts can be sent in the form of SNMP traps, emails, pages or text messages to the administrator or even automated calls on some extreme level of threat. These alerts are configured according to the severity and the frequency of occurrences or on some other criteria. Traditionally this is what we can expect from IDS. Mostly these are set up separately from the other network and have separate management system to alert administrator so that these alerts may not show its presence to attackers.

b) Active Response:

IDS with active response are very similar as for as detection is concerned. However, they see an attempted attack and take some proactive measures against it for which they are configured rather just alerting administrator and wait for him to take some action. These IDS's can be placed INLINE to drop traffic as they see some intrusion, they can send spoofed TCP reset messages to either source or destination systems to rapidly terminate current communication session, or they can send ICMP unreachable messages to the source system in an effort to convince it that target system is unreachable. Some IDS configure Firewalls or routers between the targets and attackers to block the traffic. Some IDS systems try to harvest information about attacking system by namesever lookups or traceroutes. The essence is that you don't need to have a system administrator watching the wire in real time.

c) Inline IDS:

IDS can also be placed INLINE between target system and internet. By putting IDS inline we could use it as an intrusion prevention system. In this way IDS can simply drop the traffic which seems to be malicious and should not allow passing through. But there is also a serious problem associated with it – False Positives can end up with even more disastrous results than other average IPS and performance is also a big question mark. As all the traffic is passing through this one box, a single point of failure is often a serious issue from redundancy and performance point of view.

1.5.5 IDS Capabilities

From implementation point of view, and to get most out of an IDS, it is important to know about the capabilities and limitations; what it can do and what it cannot. Here is a brief overview about the expectations that we can make with an IDS. The detailed discussion can be found in [37].

- **What Will an IDS Do for ME?**

  IDS is an important tool in security infrastructure of any organization. It can give you extraordinary insight about what is really going on in the network and alert in case of intrusion. It can help to monitor network activities, trends of users/system and network, help to enforce company security policies and gives deeper understanding to plan better for future budgeting. It can also tell you where the blind spots and problems in network security are. It can notify the system administrator about possible security threats and even failed attempts. Here is the list of few points about IDS capabilities.

  - Continuously monitor network/system activities
  - Read hundreds of megs of logs and look for specific issues
  - Generate tremendous amount of data as a result of monitoring
  - It will lost its importance if not properly tuned
  - Delicate trends in logs can be figured out by careful inspection that are unnoticeable otherwise
  - Supplement security and protection mechanisms
  - Assist system/network administrator as a partner

- **What Won't IDS Do for Me?**

  An IDS is not the solution of every security problem. It is just another line of defense that keeps your security one step ahead from routine. It will not replace your firewall nor system administrator. It will not secure the physical perimeters of the site, or magically detect small deviation from routine work. Here is the list of few points about IDS limitations.

- Replace the need for security expert
- Detect every attack that occurs
- Prevent attacks from occurring
- Prevent attacks from succeeding (in most cases)
- Replace other protection mechanisms

1.6 Thesis Outline

In this chapter we have tried to give an overview VoIP technology and protocols involved in it. We put special emphasis on SIP protocol and how it is involved in VoIP as this is core of our topic. Later in this chapter we have discuss some important concepts related to Intrusion and Intrusion Detection. This overview is essential to address the topics in subsequent chapters so that concepts can be better understood. However, this should not be considered a comprehensive overview as lot more is needed to be discussed.

Rest of this dissertation is arranged as follows: Chapter 2 will describe Snort in a bit detail so that we can study its strengths, weaknesses and enhancement opportunities. We shall describe a comprehensive Literature Survey which shall include different types of IDS technologies and research work related with our domain in Chapter 3. Requirement Analysis will be presented in Chapter 4 describing the problem domain in detail and research focus. Chapter 5 will present System Design in detail and its implementation is shown in Chapter 6. In Chapter 7 Results and Analysis are exhibited and finally Chapter 8 concludes the thesis.

Chapter 2
Snort – Intrusion Detection System

2. Snort – Intrusion Detection System

As the title of our thesis suggests, Snort is the tool that shall be used for SIP based VoIP security, and hence Snort is our focus. Snort is a security application based on signatures/rules used for intrusion detection. This is highly extendable and more functionality can be added with its core engine. It is being used in small and medium sized networks very efficiently. The beauty of Snort is that it offers ways to customize its preprocessing unit and output components. The idea is to add some capability to Snort so that it can effectively detect intrusions of SIP. Therefore it is important to develop understanding of Snort so that its features can be exploited effectively.

In this chapter we shall introduce Snort and present its features briefly. This chapter is organized as follows: Section 2.1 will introduce Snort as IDS, in Section 2.2 different elements of Snort will be discussed which will also include its components, Section 2.3 will give an insight how Snort can be enhanced to add support for SIP intrusion detection and finally Section 2.4 will conclude the chapter.

2.1 What is Snort?

Snort is a free and open source Intrusion Detection and Prevention System (IDS/IPS) having capabilities to monitor and analyze real-time high speed network traffic. Snort is libpcap-based packet sniffer and logger supported by many add-on programs to provide different ways to gather information, process data and alert user. It is a light weight, cross-platform network intrusion detection tool that can be deployed in small and medium TCP/IP networks to detect a wide variety of suspicious network traffic as well as potential attacks. It has small system and memory footprint and very easy to implement by administrator in a variety of network environments. Although add-ons are not part of core Snort suite but they provide a rich variety of features to administrator. We shall discuss this feature in a bit detail in proceeding sections.

Snort provides a rule based mechanism for logging to perform content pattern matching and detect a variety of attacks like buffer overflow, stealth port scan, SMB probes, web application attacks, OS fingerprinting attempts, etc. Snort has a real-time alerting capacity with alerts being sent to syslog, SMB "WinPopup" messages, in an alert file and even add-on can be added to send administrator an email or even text in case of critical discovery. Snort is configured using command line switches and Berkeley Packet Filter (BPF) commands. Again add-on can be used to get GUI for configuration and management of this package.

Originally Snort was designed as a signature based IDS but later on power of protocol analysis and anomaly detection has been added. It can be rapidly deployed to fill potential holes in network security architecture, for example when a new vulnerability emerges in web browser and vendors are slow to respond to fix it up. This make this tool a favorite choice for a variety of network environments and due to its popularity and thousands of deployments, SNORT has become de facto standard for IPS.

2.2 Snort Elements

Snort's architecture is focused on simplicity, performance and flexibility. It is composed of three subsystems: Packet Decoder, the Detection Engine, the Logging and Alerting system. Libpcap library is used for sniffing packets in promiscuous mode which gives Snort an essential capability of getting packets. In this section we shall describe some detail of these basic components along with optional components which enhances the tool's capabilities. Optional components include Preprocessors and Output plugins which are basically add-ons developed for Snort. Important thing to note is that *snort.conf* is the file that holds the total configuration of Snort and shall be referred in below sections frequently.

2.2.1 Packet Decoder

This is the first component of Snort which receives packet from NIC. The Decoder Engine decides which protocol is in use for a given packet and matches data against allowable behavior. Main function of decoder is to set pointers in packet data for subsequent analysis of detection engine. It also watches the structure of network packets to make sure they are constructed according to specification. It can generate alerts if sees something wrong into it like strange size of packet, improper set options, uncommon settings, etc. As it is very rare to see such problems in normal network conversations, these alerts can be disabled. This can be done in snort.conf file.

Speed is the critical issue in this section as in high speed network dropping packets can be a major problem. After matching packets against decoder, they are forwarded to preprocessor (if configured) or detection engine. Snort provides decoding functionality for Ethernet, SLIP, raw (PPP) data-link protocols. Our focus will be on Ethernet only.

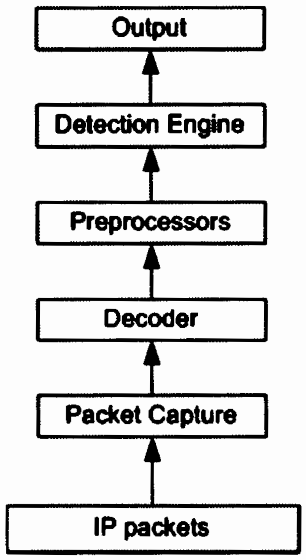


Figure- 2.1: Block Scheme of Snort Architecture

2.2.2 Detection Engine

Detection Engine is considered to have the core functionality of Snort tool. This is mandatory component of Snort that takes data from either decoder or preprocessor and compares it against rules that are configured in snort.conf file. The capabilities of detection engine can be enhanced by adding Detection Plug-In.

The old version of Snort implements a three dimensional linked list of rule headers, rules and detection function. This mechanism was simple but not very efficient when number of rules increases in number. New detection engine was introduced to bring Snort to next level. This engine can check multiple rules in parallel and is capable to perform detections in gigabit network.

In new detection engine, set wise analysis is being used. For example in Snort 2.1, there are four rule groups: TCP, UDP, ICMP, and IP. When a packet comes to detection engine, it first checks whether it is TCP, UDP or ICMP and upon confirmation it apply the particular rule group. If packet does not belong to the three categories, it applies rule sets of a general category of IP. The rule sets are further divided into rule groups. When packet comes in the detection engine, it calls prmFindRuleGroup function which directs it to that particular rule group. Then rule group matching function is called and after that actual rule is fired. We are omitting the detail of how they work and its details can be found in [37]. After all rules are checked, it consults the configuration order specified *(pass, alert, log)* and alerts are generated accordingly.

There are few other concepts that are associated with this detection engine. One of which is Tagging. We can log specific number of packets after a rule successfully fires by using Tag rule option. This is usually done to log additional traffic so that the attack can be analyzed afterwards. Hence it gives an opportunity to better understand the attack and may also lead to forensic investigation.

Another useful feature is Thresholding which allow a user to limit number of events triggered by any rule or rules. For example there can be a noisy rule being fired upon identification of attacks like DoS or by some worm which should be notified only once to administrator and hence avoid to generate millions of alerts every hour. Any clever attacker can hide its actual attack by overloading the logs as an admin might ignore it in millions of alerts.

A need may arise when administrator wants not to be alerted on some event without removing rule from rule set. Suppression option is meant for this. This is a way to tune snort for particular environment or network segment. Another scenario to use suppression is that when security analyst try to figure out loopholes in network and try to exploit some breach.

2.2.3 Alerting and Logging Components

This is the third mandatory component of Snort system. This comprises of two functionalities: Logging (archive the packets that triggered the rules) and Alerting (notify administrator that rule has fired). We can enable these mechanisms and of course either of these options can be turned off.

The logs can be maintained in their decoded form, human readable format (easy to analyze) or tcpdump binary format (faster to record). Alerts can be sent in many ways like SMB Pop-Up windows in MS Windows system, through UNIX sockets over the network, SNMP traps and most important syslog to record in text file. These alerts can also be stored in some database e.g. MySQL, or even applications can be built to send SMS text message, an email or send automated call on cell phone. Output Plug-Ins can accommodate all such functionalities those can be configured by users easily. Details about Output Plugins can be found in chapter-7 of [37].

2.2.4 Preprocessors

Preprocessors are plugins that provide Snort the ability to handle decoded data into different ways according to requirements. They normalize traffic for variety of services so that when packets will enter into detection engine, signatures will recognize the attack with high accuracy. This also does advanced decoding and attack detection. The idea behind the development and use of preprocessor is that inspecting a single packet by detection engine may not detect some attacks, for example if attacker distribute the malicious data into many packets, an IDS shall miss such attacks. This is a way to confuse IDS sensor so that an attack can go unnoticed. So preprocessors are also used for self-defense. Frag2 & stream4 preprocessors are for self-defense. If any preprocessor is configured, packet shall enter and be processed accordingly.

The preprocessors enhances the abilities of Snort many folds and let the user to add new functionality into it very easily. Important feature of preprocessor in which we are interested is that they extend snort's capability to detect anomalies that cannot be detected by simple rule matching. These anomalies may be the sign of intrusion. Snort preprocessors provide us a way to implement anomaly detection logic into it and this feature makes Snort prominent among other IDS solutions.

There are three major purposes why preprocessors are used: (i) Reassembling Packets (ii) Decoding Protocols (iii) Nonrule or Anomaly-based Detection. Many preprocessors come along with Snort package. For example HTTPInspect, frag3, arpspoof, sfportscan, stream5, etc. A developer might build a preprocessor with capability of statistical anomaly detection or even learning abilities. The limit is the imagination of developer. Two of such examples are bo (Back Orifice) and sfportscan (port scan) preprocessors. As we are more interested in preprocessors, let's have a brief overview about two preprocessors "HTTPInspect" and "sfportscan".

a) **HTTPInspect**

HTTP protocol is an application layer diverse protocol having different types of information in HTTP sessions *(html, php, asp, xml, java, multimedia, etc.)*. HTTP traffic should be normalized (e.g. %40 signs or \ in URL) before inspection by detection engine so that it has best chance to match with signatures. HTTPInspect preprocessor is doing all these along with other functions. This preprocessor can recognize both client requests and server responses. A current version of HTTPInspect only handles stateless processing and looks HTTP fields on packet-by-packet basis. This can use some other module for packet reassembling but with limitations. Unlike many other preprocessors, HTTPInspect has a large number of configuration options which shows its diversity and effectiveness. Users can configure individual HTTP servers with variety of options (e.g. IIS and Apache). The configuration can be divided into two categories: Global configuration (Deals with Global Functioning) and Server Configuration (Server specific). Details can be found in chapter-6 of [37].

b) **Sfportscan**

Now let's see how snort can be extended to detect attacks in any way you like. This is an example of Anomaly-based detection. A port scan consists of several probes generally to more than one ports or more than one machine. For example an attacker may need to find open ports on some server or open ftp port on a network. If an attacker has no such behavior, it is very difficult to detect such attack. For example if there is a single request on port 21 comes to a machine which is not actually ftp server, it cannot be considered as port scan as user may have entered wrong IP address mistakenly. However 200 port 21 requests on each IP address on a network (may be arriving in numerical order) are almost certain indication that someone is scanning port 21 on your network. So the factors that distinguish an attacker are (i) Number of destination ports (ii) Number of destination hosts (iii) Time over which the probes are sent.

Now we can clearly see that simple rule-matching mechanism cannot detect such attacks effectively. This preprocessor detects this port scan by watching number of probes in a specific time period and flag if found some anomaly. Important fact in this scan is that most of these queries sent by an attacker will be negative which means that these ports are not open. In any legitimate network, such negative queries are rare and still rare that in so much number in a given amount of time.

This preprocessor shall also alert on stealth port scan probes sent by *nmap*. Important thing to note is that sfportscan preprocessor warns that the information presented at the end of scan will be extensively inaccurate. But at least administrator will come to know about such attacks although not know exactly about the probe count which is not even important to know. We are omitting its detail as our aim is to present that anomaly detection capabilities can also be accommodated in Snort.

2.3 How to Enhance Snort!

Snort is open source, free and effective tool for intrusion detection. Sourcefire *(company developed Snort)* and open source community are continuously working on it to enhance its capabilities. Snort's robust and scalable architecture offer lots of opportunities to mold it to an angle that fits our requirements. Along with the improvement of decoder, detection engine and alerting/logging subsystem, Snort proposes to exploit opportunity of developing add-ons into it. These add-ons include preprocessors, Detection Plug-Ins and alerting/logging Plug-ins. We already have discussed preprocessors and their importance in a bit detail in this chapter.

Research community is already in action to make VoIP infrastructures secure so that commercial users feel comfortable using such services. A project under the title SNOCER [33] "Low Cost Tools for Secure and Highly Available VoIP Communication Services" is an effort in this direction. The authors presented secure and high available software architecture for VoIP infrastructure. In area of security, they use IDS tools specially enhance for VoIP traffic along with VoIP advanced traffic monitoring servers.

Another such effort of SIP intrusion detection and prevention recommendations is presented by S. Niccolini et. al. [44]. They analyzed SIP intrusion detection and prevention requirements and proposed IDS/IPS architecture. They developed a prototype of their proposal by using Snort by extending its basic functionality and make use of preprocessor.

Having said all this, Preprocessors and Detection Plug-Ins are two attractive elements which can be very helpful for adding new capabilities. We can add anomaly detection capabilities in Snort by developing specialized preprocessor for SIP – VoIP systems or Network or can introduce other such mechanism. Preprocessors are very strong and we can implement knowledge based and behavior based intrusion detection techniques into snort.

2.4 Summary

In this chapter we have tried to explore Snort which is the tool to be utilized for intrusion detection for SIP-VoIP. We have discussed Snort's components to understand their functionalities and explore its features keeping in mind their utilization. The ultimate goal of the research is to enhance capabilities of Snort so that it can effectively detect SIP based VoIP attacks. Snort is absolutely capable for accommodating new features which can also include statistical anomaly detection, machine learning and other such techniques. It is customizable and can be molded according to the requirements easily. These features make snort an excellent choice as an Intrusion Detection System.

Chapter 3
VoIP Intrusion Detection Systems: Literature Review

3. VoIP Intrusion Detection Systems: Literature Survey

In start, the networks and applications were not so complex because they were built by keeping into mind about their working in faithful environment. As the time passed, people started finding and exploiting vulnerabilities of these applications and network components. On the other side, developers also focused on fixing the loopholes in their existing and later developments and gradually formulate their procedures for building secure applications. However they also put their efforts in finding the attacks when they occur, at least at identification level. This was the motive of developing Intrusion Detection System (IDS).

Although VoIP attacks are not prominent in discussion forums and security reviews today, but they soon will be as the technology matures. Enterprises, educational research groups, government institutions and open source communities has already foreseen this problem and started work to investigate the best methodologies that should be applied to protect VoIP infrastructure. As part of the process for secure VoIP communication, people have defined general recommendations for safe VoIP deployment and then to analyze security mechanism that should be applied to protect the whole infrastructure.

As the complexity of systems increase, new methodologies of finding intrusions also emerged. In this chapter we shall discuss about intrusion and Intrusion Detection, how it works, and what are their methodologies of detections, so that we are able to proceed further to device methodologies. We shall also discuss intrusion detection methodologies proposed in literature.

Rest of this chapter is arranged as follows: In Section 3.1 we shall discuss different categories of Intrusion Detection Systems, Section 3.2 will present why intrusion detection is difficult for VoIP systems and the challenges faced by IDS in VoIP environment. Recent approaches to deal with intrusions in VoIP and related work will be discussed in Section 3.3 and finally Section 3.4 will summarize the chapter.

3.1 Categories of IDS

IDS are traditionally categorized into two methodologies: Rule Based Detection and Anomaly Based Detection [37]. Rule Based analysis is matching a known pattern to activity seen on the system or network. It is similar as that of antivirus using the signatures for detection and blocking of viruses and malicious activities. This is the most common and widely used approach in commercial IDS technology today. As new attack comes on the surface, new signatures need to be written and included on IDS.

A variant of rule based detection method is Protocol Analysis [37]. Protocol Analysis attempts to define every possible acceptable behavior for a specific kind of activity. Rather to write a simple rule, a detailed description of normal protocol activity is mentioned and any deviation is flagged. But there are some serious problems associated with this method. Protocols are very well defined and tightly written. On the other hand all vendors do not pay attention to everything in protocol definition. The result is that the Protocol Analysis based IDS will complain correctly about something that conflicts with RFC but perfectly normal for that particular application. This is difficult and time consuming process to write good and effective rule.

The second well-known approach is Anomaly Detection. This is based on learned or predefined concepts about normal and abnormal system activity to distinguish anomalous behavior and to monitor or block such anomalies as they occur. Which activity should be considered normal is a big question. Some IDS uses predefined standards about what is normal traffic in a network or system. While some other systems watch the routine activities and develop some baseline profile about normal behavior from that. Any deviation from baseline activity is flagged. There also many downsides associated with this method including high false positive rate.

In a document [39] Peng Ning and Sushil Jajodia discussed in detail about the variations and implementation methods of Intrusion Detection Techniques. Each of these techniques worth a detail discussion but are out of scope of this document. They are presented only to give a taste of technology. The detailed discussion can be found in [39]. Every technique has its own advantage and disadvantage.

3.1.1 Anomaly Detection

a) Statistical Models

This is one of the earliest methods used for intrusion detection. This model works on the assumption that intruder's behavior is noticeably different from that of normal user. These models used to collect and aggregate user's behavior and distinguish an attacker on the basis of this data. There are two statistical models proposed for anomaly detection: NIDES/STST and Haystack.

b) Machine Learning and Data Mining Techniques

This category of analysis is further divided into four methodologies: Time based Inductive Machine, Instance based Learning, Neural Network, and Audit Data Analysis and Mining.

c) Computer Immunological Approaches

This approach is based on human Immune system which can differentiate between self and non-self. The self is represented with system-wide parameters in the form of strings. If string is not matched with self then it is considered as non-self. Naïve approach is used to discriminate these two categories.

3.1.2 Misuse Detection

Misuse detection system looks for well-defined patterns of known attacks or vulnerability and matches them with their repository of threats. Misuse detection is considered complementary to anomaly detection because the known attack patterns can be detected more effectively and efficiently by well-defined rules matching. Misuse detection system can catch even minor or negligible threat that anomaly detection system usually ignore.

We can also consider this system as Rule-Based or Knowledge Based system; however Misuse Detection covers broader aspects of analogy. We have already discussed a little bit about Rule-Based systems earlier. The major issue of these systems is the way to represent known patterns of attacks. Here we shall discuss few methods of representing attacks.

a) Rule-Based Languages

This is the most extensively used method for misuse detection expert systems. The patterns of known attacks are specified as rule sets. Here we discuss two rule-based languages i.e. RUSSEL and P-BEST. Other languages are similar as they all specify known attacks as event patterns.

b) State Transition Analysis Tool-Kit (STAT)

Rule-based languages are strenuous to use. In STAT, state transition analysis techniques are used to facilitate the specification of known attack patterns. It assumes that all penetrations have two common features: (i) The attacker need to have some minimum level of access to the target system (ii) After penetration attacker acquire some ability which he does not have prior to this attack. So, STAT keeps an eye on the sequence of actions performed by some attacker that leads from initial state of a system to a target compromised state.

c) Colored Petri Automata

Misuse detection can also be viewed as a pattern-matching process. In this way intrusion signatures are classified according to signature relationships among the events that composed the signatures. Colored Petri nets can be used to represent attack signatures where guards will represent signature context and vertices will represent system states. These nets are called Colored Petri Automata.

3.1.3 Distributed Systems

As the internet grows up, more and more networks and resources are being added into it. But on the other side it is difficult and complex to secure the entire network with traditional approaches. So, research is going on its way for developing IDS technologies for small and large distributed systems. In distributed systems, the research focus is on two essential issues: scalability and heterogeneity. The research in this regard is being conducted in three main areas. (i) People are building scalable, distributed IDSs or are extending existing IDSs to make them to work with large systems. (ii) Network based IDSs are being developed to take the advantage of the standard network protocols to work in heterogeneous environment. (iii) Standards and techniques are being developed to facilitate information sharing among different, possibly heterogeneous IDSs. The discussion about distributed intrusion detection system is a vast topic and is not covered in this research.

3.2 Why VoIP Intrusion Detection is Difficult

There are many reasons why Intrusion Detection in VoIP is challenging. For example:

- There are multiple protocols involved in providing services to VoIP systems (signaling, media, management, Quality of Service, etc.), each with its own vulnerabilities. This design results in novel and more complex attack scenarios and need cross protocol aware IDS.
- VoIP systems are distributed in nature and need to monitor entry points of networks and interconnectivity links to get comprehensive insight of attacks.
- VoIP systems are heterogeneous having different equipment manufacturers, different configurations and systems maintained by different administrators.
- Traditional PBX system are closed and confined telephony systems while VoIP works on open IP networks supporting end to end VoIP services. This exposed the system an easy access to hackers for traditional internet attacks.
- Real-time media communication in VoIP systems poses tough situation for detection of some attacks like SPIT detection of which is not very fruitful when spam detection is done offline. What is the use of detection a spam call after user has already received it?

3.3 Related Work

There are three parameters to measure the efficiency of an Intrusion Detection system: Accuracy, Performance, and Completeness. Every new system and architecture tries to improve its efficiency and effectiveness. Many approaches and methodologies have been proposed and are available in literature each having its own merits and demerits. Some of them are specifically designed for VoIP infrastructure and some are general purpose approaches. Important aspect is that which approach best fit into the scenario. Here are few important approaches out of which some are specifically designed for VoIP systems and some are generic in nature.

3.3.1 Framework for Detection Anomalies in VoIP:

In VoIP networks, use of SIP is not well recognized as yet but still it is becoming an important protocol of choice in such networks. Because of this fact, security issues are not properly being addressed by academic, research and development community. This lack the tools and methods required to tackle with intrusions and other such issues in SIP based VoIP. Yacine Bouzida and Christophe Mangin in [49] introduced an architecture and technique to detect anomalies in VoIP network. They have particularly designed the system for Session Initiation Protocol (SIP) and therefore it is an important artifact for intrusion detection technologies in our research.

They proposed an intrusion detection technique which consists of three stages. In the first stage VoIP traffic is collected which will contain either normal or attack traffic. The second stage extract attributes out of this traffic which represent most of information related to normal or attack traffic evidence. These attributes are extracted on the bases of RFC3261. In third stage classification process in performed which actually distinguish normal or attack traffic. This classification is based on a model which is built on the bases of previously known attacks. They determined three different profiles used to illustrate SIP signaling flows. These are packet based, transaction based and dialog based. In this way along with signaling based protocols, session description and media transfer protocols are also considered for analysis.

The authors set variety of goals for this mechanism. For example theoretically it detects the whole set of previously known attacks by automatic learning process. It can differentiate between normal or attacks traffic easily. It also has capabilities to detect new anomalous behavior causing by novel attacks. The authors considered this system as a complete one because it also consider VoIP features. It also accommodates stateful and cross protocol detection methodology. The authors also consider this mechanism extensible because it can learn new attacks by simple updates automatically.

One of the important issues related anomaly detection and behavior based learning system is that an attacker can slowly corrupt the knowledge base which serves as an input for such systems. In proposed system model uses attributes based on different statistical measures between current network flow and past flows. It means that past experiences are used for intrusion detection. An attacker can slowly poison the traffic which will serve as reference in future and then manage to escape from detection. This aspect has not been discussed in the paper as there is no way of its mitigation. Some important attack types are not discussed in this methodology, like eavesdropping, connection hijacking & fraudulent usage (billing frauds etc.) and spam issues. Along with it issues of false positive and false negative has also not been discussed.

Although the authors have claimed that the proposed system covers all known attacks detection but after carefully analysis this claims is not seems to be valid. In the paper they have proposed the placement of the system in front of User Agents or Servers or in front of firewall. In this way the system can catch all the inbound and outbound traffic of that element. But in this way it will catch the traffic only meant for that element only. It will not come to know what is going on other VoIP elements. This means that the system cannot accommodate every sort of attack even if we implement the system on multiple elements. There is no way of correlation of information. The argument is also valid for both transactions and dialog validation. What if someone is impersonating a legitimate user and trying to use his account for some service. If she is able to successfully fool the Proxy Server, the proposed system will never catch it. Therefore the system effectiveness is limited only to individual devices in front of which they are implemented.

3.3.2 SIP IDS/IPS – Recommendation:

Niccolini et. al. in [44] analyzed SIP intrusion Detection and Prevention requirements and proposed an IDS/IPS architecture. They also implemented their proposed architecture by developing a prototype in Snort. This prototype extends the basic functionality of Snort by using different components of this software. They make use of preprocessor block especially and try to process data at application layer. Their focus was to accommodate intrusion prevention mechanism rather only the detection. Their focus was also on performance and QoS issues as for prevention the system has to be put in-line to a link path.

In their proposed architecture, the authors recommended the use of network based intrusion detection and prevention system for VoIP infrastructure. They have proposed a two stage technique for detection process. In first stage knowledge based methods are applied while in the second stage behavior based inspection is suggested. Second stage is applied only if first stage does not detect any thing and hence it improves the results of detection. The authors discussed and implement the first stage of their proposal for proof of concept and left the second for future work. Therefore second stage is used to refine the detection ability and effectiveness of stage one. The system is required to be implemented preferably at the entry point of SIP network e.g. SIP-aware Firewalls, Gateways and Session Border Controllers. As the paper describes only the first stage therefore therefor they have only elaborated the same into four block which consists of filtering, SIP syntax analysis, security analysis on mandatory fields and stateful analysis.

There are plenty of goals that the authors want to achieve. First of all they have described the need of implementation of two stage architecture in a way where knowledge based technique is applied before behavior based technique. This improves the effectiveness and accuracy of detection. Secondly, it is a fact that Snort does not have capability to do syntax analysis of application layer. Therefore they have implemented this functionality along with protocol analysis for SIP protocol in preprocessor block. They have specially emphasized on two important issues: much higher number of elements needs to be secured in VoIP; in VoIP infrastructure, to launch DoS attack the servers do not need to be bombarded with heavy traffic. The procedure also houses a level of statefulness into its architecture.

Although the paper proposed very impressive two stage technique and defines in a quite impressive way, but still it has some limitations. The system is proposed to be placed at the entry points of VoIP network and therefore it will only cover that network. User agent based attacks are not discussed in this architecture. The prototype is developed in Snort which is a knowledge based system. In this prototype, only the first step is implemented and the data is handed over to the detection engine. If we want to accommodate second step, then that should be before detection engine which deviates the important design consideration of implementing behavior based technique after knowledge based. They have not proposed any mechanism to deal with these issues. As this system is introduced as intrusion prevention system and put in-line at the entry point of network, therefore if we add behavior based system into it, QoS issues will be a big question mark. The method proposed also does not cover attacks spanning other than SIP protocols and cross protocol detection.

3.3.3 Stateful and Cross Protocol IDS Architecture:

One of the early efforts made towards developing intrusion detection architecture for Voice-over-IP environments is brought forward with the code name SCIDIVE [45]. This was an effort put forward by people in Purdue University and Avaya Labs. Their effort was to develop a stateful and cross protocol intrusion detection architecture. The idea behind the endeavor was to detect attacks which can span on multiple protocols where correlation among them is required, and maintain states of a session so that session awareness can also be considered for attacks spanning over the session.

The proposed structure is designed to detect different classes of intrusions, like masquerading, Denial of Service and media stream based attacks and protect both call management and media delivery protocols. They have made use of two abstractions for IDS, Stateful Detection and Cross-Protocol Detection. The architecture of SCIDIVE consists of Distiller which translates all incoming traffic into information units call Footprints. These Footprints are grouped into trails of same session. An Event Generator maps these Footprints into events which are matched with rules to detect attacks. Important aspect of this architecture is to make use of trails for session maintenance which is ultimately serves as an input for stateful and cross protocol detection.

The most important goals of this architecture is to present a stateful and cross protocol detection methodology. The authors have given some examples to motivate the needs of these paradigms. This architecture covers a wide set of attacks categories, e.g. denial of service attacks, authentication attacks, tool fraud and privacy. It has a strong capability of detection of attacks which span over the whole session or intrusions which can only be detected by looking both session and media protocols. SCIDIVE can be placed on different locations like end point devices or on servers.

The authors have proposed that primarily the system can be placed on both end points and to detect some attacks they have to share information with each other. This is not very much percale idea as exchanging information between end points cannot always be trustworthy. Moreover if one user agent belongs to some other network then this theory becomes invalid. Another proposal for placement of this architecture is an aggressive approach which suggests placing sensors on every component of VoIP network. These nodes will share information with each other. But no mechanism has been identified about how they will share this information and what will be contained in that. Although the architecture covers many types of attacks but it lacks some basic knowledge based detection capabilities like syntax and semantic analysis.

3.3.4 Intrusion Tolerant System:

One of the good approaches to secure any IT infrastructure is to build a durable system resistant to attacks. In this effort Fengmin Gong et. al. [41] presented *Intrusion Tolerant System Using a State Transition Model.* Based on Fault Tolerance system (e.g. commonly deployed in air traffic control) they have tried to propose a system which is Intrusion Tolerance. But in deployment of such systems there are many challenges that make it difficult to realize. The paper discussed these challenges in a bit detail. As this project was funded by DARPA, therefore they focused on one of their pre designed architecture of a project named SITAR.

A state transition approach was used where a system can change its states after some event. This leads to a dynamic behavior of Intrusion tolerant system. In this model they defined nine different states like good state, vulnerable state, active attack state, etc. The system thus enables to accommodate multiple intrusion tolerant techniques into it. It also supports different levels of security necessities. As an example of state transition, the system will enter into Vulnerable state from Good state upon encountering access rights violation, for example. The system will either recover from this Vulnerability and come back to normal state or leads to an attack. Hence, the system is supposed to change its states and present current snapshot at any instant of time.

This approach is based on proactive concept that a system should transition from one good state to another good state. If system transition leads to, active attack state for example, it will either be detected and rectified or intrusion detection system triggered or goes undetected if neither of previous two conditions takes effect.

They also presented case studies by actually applying this model on different vulnerabilities. The considered five classes of vulnerabilities (which leads to actual attacks): (i) Compromise of confidentiality (ii) Compromise of Data Integrity (iii) Compromise of user/client authentication (iv) DoS from external entities (v) DoS by compromising internal entities. They have applied this system against five vulnerabilities: (i) Active Server Pages (ASP) vulnerabilities in IIS 4.0 (ii) Common Gateway Interface (CGI) vulnerability in Samba Server (iii) Sun Java Web Server Bulletin Board vulnerability (iv) 'SITE EXEC' vulnerability in wu-ftpd (v) DoS vulnerability. They traced the transitions by applying these attacks and showed the effectiveness against such attacks.

Although the system is very effective for the attacks presented in their paper, but VoIP should essentially be considered a different scenario based on many reasons. VoIP infrastructure is complex in nature making use of many elements beyond single administrative control which leads to potential vulnerabilities unknown to VoIP security administrators. Moreover, the system or infrastructure proposed should be under one administrative domain and perhaps centralized. On the other hand VoIP system is distributed in nature which leads many other issues. Additionally the paper has not discussed performance issues. It is a big question mark about how system will behave when applied to real-time and high speed infrastructure. It also not covers problems or attacks specifically related with VoIP and SIP.

3.3.5 A Swarm-Intelligence-Based Intrusion Detection Technique:

With increase of network diversity, bandwidth improvement allowing real-time application to work on network, a wide variety of network based applications, the problem of misjudgment and misdetection in real-time environment is a big issue. A very innovative approach to reduce misjudgment and misdetection of intrusions is to make use of Swarm Intelligence approach. Zhou and Liu [42] proposed a concept of Swarm Intelligence based IDS which fulfills above needs and increase real-time response. This technique is based on the fact that gregarious insects like bee, ant, etc. have very low intelligence but their colony can solve many complicated issues and shows high level of swarm intelligence with flexibility, stability and self-organization capabilities

They proposed a swarm-intelligence based system which has the following policies: (i) Simplifying system organization structure – to improve efficiency (ii) constructing independent IDS units – to improve synchronously efficiency and performance (iii) Separating data traffic according to functions – to reduce processing data (iv) Enhancing information exchange among units – to detect complicated attacks. They proposed two different architectures of their model i.e. a network based IDS model (gathering network based information and detecting network related attacks e.g. email, DNS, etc.), and a host based IDS model (gathering host based information and detecting host based attacks e.g. password stealing, etc.). They also proposed the construction of detection unit showing implement framework of detection unit in two models mentioned above. Important thing to notice in these models are that detection units are independent in their nature based on small set of functionality and work on their own. Each detection unit contributes its results to entire detection system to realize swarm intelligence concept.

The authors discussed the realization of their models by clearing some important issues. One is to get user trace under network environment where an attacker can launch attacks by IP spoofing with multiple identities. Second is maintaining performance and efficiency while treatment of system call interception in host based IDS environment. Third is synchronization of network sharing information, where each detection unit frequently need some data not available on local host and hence badly affects efficiency.

The method presented in this paper is very effective generally while considering common network or host attacks. But authors have not implemented or tested their system on attacks or set of attacks. It is not very clear which set of attacks this system is really very effective. Moreover, because of distributed nature of VoIP infrastructure, implementation strategy and resulting efficiency is not known. But this method might be a good starting point for using it in some other system.

3.3.6 Data Mining Techniques for (Network) IDS:

There are a number of drawbacks of IDS, for example: (i) in most of cases IDS are tuned to detect known network attacks which leaves vulnerabilities for novel attacks (ii) data overloading for analyst where megs of log files with millions of record may be needed to assessed every day (iii) False Positives where IDS consider a normal activity as an attack (iv) False Negative where IDS miss an actual attack and does not generate any alert.

Theodoros and Konstantinos [40] proposed a data mining techniques for network intrusion detection systems which is a process of searching a large volume of data to discover patterns using association rules automatically. Data mining also called knowledge-discovery can help improve Intrusion Detection by addressing above mentioned issues. There can be many diversities of this technique but the main function is classification of traffic as normal or malicious. There are many significant advantages in Data Mining technique like Off-Line Processing, Data Mining with Real-Time IDS, Multisensor Correlation, and Evaluation Datasets. The authors has presented different Data Mining techniques including Feature Selection, Machine Learning (Genetic Algorithm, Fuzzy Logic, Neural Networks,Immunological bases techniques, Support Vector Machines), Statistical Techniques, Ensemble Approaches, Predictive Analysis, and some other techniques (e.g. NIDS with Random Forests, Exploiting Infrastructure, etc.). At last authors present some existing systems that apply data mining techniques.

The paper also proposed their own data mining technique that can potentially prove to be good for IDS. They use bi-clustering as a tool to analyze network traffic. Bi-clustering is a problem of finding partitions of vectors and subsets of dimensions such that projections along those vectors are close to each other. They resolved the problem by representing data in a matrix form where each row represents an object and each column represents a feature. They have also given an exampel to illustrate their solution and comment that if it is known in advance that processes in the matrix are malicious, they can give feature set of malicious traces. That set can classify new data collected.

The paper presented a survey of different data mining techniques and also presents their proposal for IDS. But they have not correlated their technique with any category of attacks. Their technique is too general and is just a proposal for incorporating it in IDS. Therefore performance parameters cannot be evaluated on the basis of this work. This technique is also particular for Network IDS and nothing has been said for Host based implementation. Moreover, the method is novel and is not mature enough to be trusted at this stage. It needs to be evaluated for different categories of attacks. As far as VoIP is concerned, lot of work is required to be done to fit this into that particular environment and in any tool.

3.3.7 Anomaly IDS Based on Junction Tree Algorithm:

One of popular methods of intrusion detection model is some graphical representation models. A graphical methodology of Intrusion Detection is proposed Evgeniya Petrova Nikolova et.al. by the name of Junction Tree Algorithm [47] is one of such models. The purpose of this paper is to find methodology for attacks (abnormal activities) recognition during normal system activities. The graph used for message passing is called Junction Tree where nodes represent cluster of variables. The method finds most likely state of distribution and defines anomalies in the obtained sequence.

The paper describes Junction Tree Algorithm in detail. They defined random vectors X with density p in unidirectional graph with nodes having associated random variable X_s. The goal of Junction Tree Algorithm is to find potential representation of graph. This graph can be coupled with some other suitable algorithm to obtain marginal potentials/probabilities. This algorithm also tells the most likely state of the distribution. The authors outline their methodology by using unidirectional maps and obtain the results in the form of most likely state sequence during examined time span. These results are used for calculating state transition probabilities.

They conduct experiments on data obtained from Sun SPARC station system examined during a time period. The data included normal user activity on a system processes with administrative rights along with intrusion data. The results obtained proved that short sequences of system call traces generated by privileged processes are good discriminator between normal and abnormal system activities. The results also proved that short sequences of system call traces are stable and consistent during normal activities of program. The method was applied on synthetic ftp, synthetic lpr, login, named and clock. The discrimination between normal and abnormal activities has false positive rate as 8.6% and false negative rate as 3%.

The method discussed in the above mentioned paper has many advantages. But the authors have not discussed the scope of attack detection. Although method has been implemented on few network and system activities but it does not mean that the method is effective for every set of attacks. As this is a method based on anomaly detection technique, and the tree discussed infers its results by stepping forward and backward, therefore performance of this method is a big question mark. Performance evaluation is not included in this paper. Moreover, VoIP is a distributed environment with multiple servers, therefore implementation of this method in such environment also worth discussion.

3.4 Summary

In this chapter we have discussed different types of IDS technologies and tried to taste different flavors of techniques. This is only an overview and details are omitted to make things simpler. Some important issues have not been studied and some are neglected. In essence, IDS technology is very diverse with more and more innovations incorporating into it as time is passing. But not every technology fits everywhere. We have also discussed why intrusion detection of VoIP infrastructure is difficult. VoIP needs special attention because of its novelty an immature technology. We have also discussed different approaches of intrusion detection techniques that have been brought forward in research literature. It is very important to prepare our security platforms one step ahead of attackers to effectively counter them. Although people have already started giving attention to this side but there is a long way to go to achieve objectives.

Chapter 4
Requirement Analysis

4. Requirement Analysis

The convergence of voice and data networks was a challenge for telecom companies as this will give new dimensions of telecom services. However, along with these advancements, security issues have also become worse than ever. VoIP is based on open standardized protocols technologies and open network architecture which makes it highly vulnerable for a variety of attacks. These technologies are not designed with the focus on security issues in mind. Moreover, VoIP inherits many security problems associated with other protocols that are involved in VoIP communication. The situation becomes more serious when coupled with misconfigured signaling and voice protocols, and poorly developed VoIP applications. As a result, an attacker has a wide spectrum for any malicious activity to disturb or disrupt VoIP services.

SIP is a new technology and many companies are now using this protocol in their applications. But the security problems in SIP and risks associated in VoIP while using SIP are not well known in network security literature. It is very clear that the security requirements of SIP-VoIP are very different on the basis of its distributed and heterogeneous nature.

In this chapter we shall identify security risks and discuss threats that are being faced by VoIP technology. Among all, our focus will be on those specifically related with SIP protocol. We shall also discuss what sort of support is available in Snort for SIP/VoIP. We shall see that Snort is certainly not fit for the use in SIP based VoIP environment. The rest of this chapter is organized as follows: First of all we shall briefly describe security risks associated with Supporting protocols/services and Media Protocols in section 4.1. As our focus will be on SIP throughout this thesis, Section 4.2 is devoted to discuss Security flaws in SIP. Section 4.3 will give a brief overview of security mechanism available in SIP and then its analysis in section 4.4. Section 4.5 will discuss what sort of SIP support is available in Snort. In section 4.6 we shall present Problem Definition and Section 4.7 will describe our Research Objective. Finally Section 4.8 will summarize this chapter.

4.1 VoIP Security Risks

There are many ways of categorization of VoIP threats. Taxonomy of VoIP threats has also been defined by VOIPSA in their document published in 2005 [26]. That is a complete one and also takes into consideration the threats that are not caused by VoIP-specific technical reasons. The document presents a detailed look at threats to give us as much information as possible. Such type of threats is out of scope of this document.

We categorize VoIP security risks into three parts: Risks in Supporting Services, Media Protocols and Signaling Protocols. The objective of this section is to give a taste of attacks other than VoIP services itself.

4.1.1 Supporting Services

VoIP uses many services to get its work done, e.g. ARP, DNS, DHCP and TFTP. These protocols bring their own security risks in VoIP infrastructure. Here we shall discuss the supporting services the associated security problems.

Domain Name Service (DNS) translates a host name (easy to understand and remember by humans) into IP address (difficult to remember). Every host needs to know IP address of other host with which it wants to communicate. DNS service keeps mapping of hostnames and IP address, and updates it periodically, providing a way to know IP address of any host accessible on the internet. DNS Spoofing and DNS Cache Poisoning are two main categories. In DNS spoofing, for example, an attacker spoofs the IP address of DNS and responds with IP address mapping of its own choice. While in DNS cache poisoning, attacker tries to alter DNS cache with false information, usually a wrong record that will map a name with a wrong IP address. This way attacker will make the server to answer something he wants. SIP uses HTTP & SMTP like addressing scheme (i.e. SIP addresses and proxy server names) and hence requires DNS service to route request to destination. More details can be found on [24].

ARP is another important protocol being used by VoIP infrastructure. ARP is a Data Link Layer protocol and translates IP address into MAC address. This introduces a risk resulting from ARP spoofing and ARP cache poisoning. For example, a hacker can fool a host by sending from a rogue network device a fictitious ARP response that includes the IP address of a legitimate network device. Attacker can also create a race condition in order to bind its own physical address with IP address of server. An intruder can also corrupt the ARP cache of some desired network element resulting in misleading communication or re-route traffic to intercept voice and data traffic. ARP flooding is another important attack that can be launched on network switches leaving the network vulnerable to conversation eavesdropping.

DHCP provides a mechanism to dynamically assign IP address within a domain. In our scenario, an IP address can be assigned to UA dynamically by DHCP server of that domain. Similarly TFTP server is used to download new firmware and configuration files onto SIP Hard-phone. ARP cache poisoning can substitutes a rogue TFTP or DHCP server. Important thing to note is that there is a wide range of attack circumstances and only limit is the imagination.

4.1.2 Media Protocols

Media protocols are responsible for actually transmission of voice, video or other data with real-time characteristics. While signaling protocols are most important to be protected, second level of defense is to protect media protocols. Securities of both categories are correlated as all media information is passed through signaling protocols. Real-Time Transport Protocol (RTP) is an application layer protocol and is the most important choice for media transmission used in many applications including audio and video conferencing. After the session is established RTP come into action to transport multimedia data between the communicating parties. This type of transmission has certain requirements for end-to-end data delivery. RTP provides mechanisms for reconstruction of media stream and loss detection but do not provide any mechanism for protecting the transmitted data from eavesdropping and many other attacks. Desired security features in RTP communication are confidentiality, integrity and authenticity. It is essential that all these features should be implemented with minimum delay and jitter.

As VoIP applications are real-time, maintaining QoS is very important. Latency, Delay, Packet Loss and jitter are some important parameters of QoS. For example it is recommended that delay should not be greater than 150 ms for acceptable quality. Similarly packet loss is also very important factor which is strongly dependent on codec. In many codecs, more than 1 % packet loss is considered as degraded quality. If an attacker managed to congest some point in the communication path, the transmission can easily be disrupted or at-least fairly degraded. DOS or DDOS are two important attacks to achieve these objectives.

If RTP stream is not encrypted, then it is easy to sniff the packets. The sequence number of RTP flow can be predicted and extraneous packets can be injected into legitimate media flow. This may lead to unexpected data packets received at the far end of transmission. Message replay is another important issue.

RTP specification does not provide any mechanism for protecting the transmitted data. Rather it suggests other services on underlying layers to be used to achieve these desired objectives. Secure RTP (SRTP) is proposed to use for media transmission which is specifically designed to provide security services like confidentiality, authentication and replay protection. SRTP defines a mechanism for security services, specify encryption algorithm to be used and provides key distribution mechanism. SRTP only encrypts the payload of packets.

Although IPSec can also be used to provide encryption services, SRTP is specifically designed for real-time data and is more efficient than IPSec in terms of bandwidth usage. The detailed discussion on RTP security can be found in [25].

4.2 Security Flaws in SIP

In this section we shall discuss threats specifically related to SIP protocol. Here is the list of most important threats:

4.2.1 Message Interception & Modification

This threat relates to ability of the attacker to intercept the signaling and/or data. SIP INVITE packets contains lots of useful information including source and Destination of the call (To, From, Via, etc). This information can allow call tracking. The duration of a call can be calculated by the time of sending INVITE message and BYE messages. The interception-only attacks empower the attacker for malicious scopes like Call Pattern Tracking, Number Harvesting which may be used for Identity spoofing afterwards, and Conversation Reconstruction which is an issue of privacy. In modification threat, the attacker is supposed to modify the packets which he/she has intercepted. This results in Man-In-The-Middle (MITM) attack and session hijacking. In this case, the malicious scopes of an attacker can be Call Black Holing in which attack can drop INVITE packets resulting call initiation to fail, Call Re-routing in which attacker may redirects the calls to unauthorized nodes, Conversation Altering, and Conversation Degradation. The situation can be worst when attacker manipulates the billing procedure.

4.2.2 Fraudulent Usage

An attacker may try to bypass the billing procedure and may call to PSTN or other VoIP network for free or use some one's legitimate account to make his personal calls. A user may also try to do long distance calls or try to avail some call package for which he is not subscribed. Such attacks are possible due to unsecured gateways, miss-configured dial-phones and billing servers.

4.2.3 Call Hijacking and Man-in-the-Middle Attack

SIP uses HTTP Digest type authentication scheme to authenticate the identity of pear. SIP User Agent, Proxy Server, Redirect or Registrar server may challenge a client user agent or even each other to be sure that both parties knows the shared secret [28]. But in actual implementations, unfortunately, only server authenticates the client. Many sorts of attacks are possible in such scenario and some of which will be discussed in billing attacks.

4.2.4 Denial of Service Attack (DoS)

This is one of the most common categories of attacks and has different approaches to conduct. So, there needs a little bit detailed discussion for this. The majority of DoS attacks are based on exhausting some of a server's resources and causing a server not to work properly for legitimate jobs due to lack of resources. This type of attacks aims to affect availability of services. With SIP server there are three resources required for operation: Memory, CPU and Bandwidth [29].

A SIP server needs to copy each incoming request into its memory buffers to process message. The amount of buffered data and time period the server supposed to keep this data depends upon whether it is operating in stateful or stateless manner. After receiving the SIP message, server needs to process the request and then forward to its destination. The time required for this processing depends upon contents, type of message and server policies. It also depends upon the CPU power of processing data. There are many processes that take lots of CPU resources and can maliciously be used to eat up all processing power. Similar is the case with bandwidth. It involves overloading the communication links connecting SIP server to the internet to such a level where congestion losses occurs. This will result in longer session setup times or even failure of session setup as legitimate requests are not even reaching the server. In this document, rather to follow our discussion according to the above categories, we shall talk about the attacks generally. We can further break up this attack into the following parts.

a) **Flooding Attacks**

State maintenance in SIP servers is one of the easier targets for DoS attacks. Brute force attack is the simplest method of attack that can be mounted against memory of a SIP server by initiating large number of INVITE requests with different session identities.

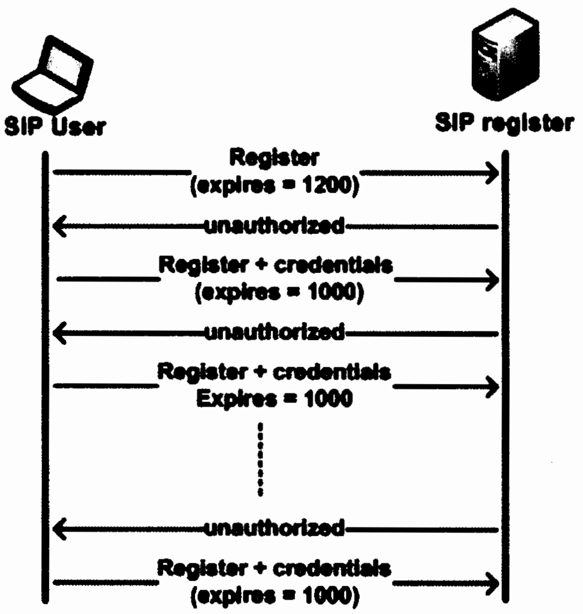

SIP User
SIP register
Register (expires = 1200)
unauthorized
Register + credentials (expires = 1000)
unauthorized
Register + credentials Expires = 1000
unauthorized
Register + credentials (expires = 1000)

Figure-4.1: Attack against SIP Registrar Server [21]

Registrar server, one of critical network element in SIP telephony, can be one of the targets. An attacker can send numerous number of bogus registration requests which can easily cause DoS and paralyze the server. The aim of attacker can also be to guess legitimate user's password [21]. Figure-4.1 depicts the attack scenario against SIP registrar server. To make this attack more effective, parameters of SIP message can be changed which can also help evading IDS's.

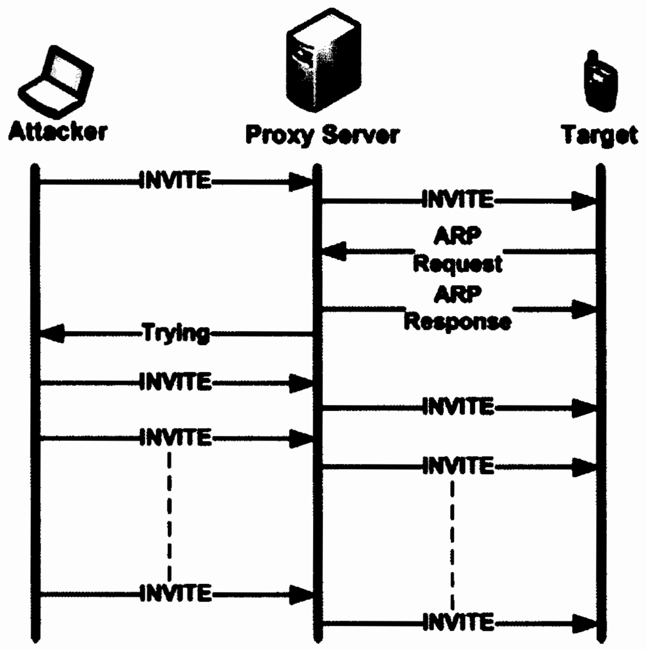


Figure- 4.2: Flood with INVITE Messages [21]

One of the most important and frequently sent messages that must be processed by a SIP Proxy Servers is INVITE. This message is used to establish a connection among two or more participants in a session. SIP Proxy Server has to keep the connection state into its memory for certain threshold time or until the connection is established or dropped manually. RFC 3261 suggest setting timer minimally to three minutes [30]. After this time expires, callee is supposed to inaccessible to establish a call. This fact makes SIP Proxy Server most vulnerable to flooding attack. Server also needs to keep the state of incoming request or wait for termination of its processing in many other scenarios. If a server has forked a request for different destinations, it has to maintain a copy of incoming request and the copy of forked request. The attacker can also launch INVITE flooding attack on end devices or applications. Attacker can generate numerous INVITE requests impersonating a legitimate user, to the end device paralyzing the victim. This situation is illustrated in Figure-4.2.

There can be another scenario in which attacker tries to behave as a legitimate User Agent. The attacker will try different ways to cause DoS either in proxy server or in the end terminal device by attempting to evade any countermeasures or identification mechanism. This type of attack is shown in Figure-4.3. The details of these attacks can be found in [21] and [31].

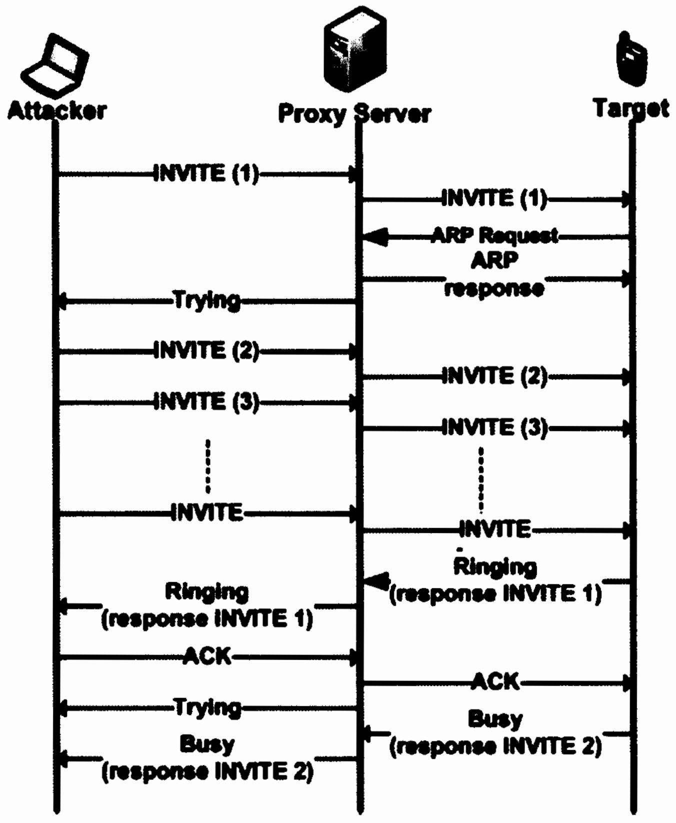
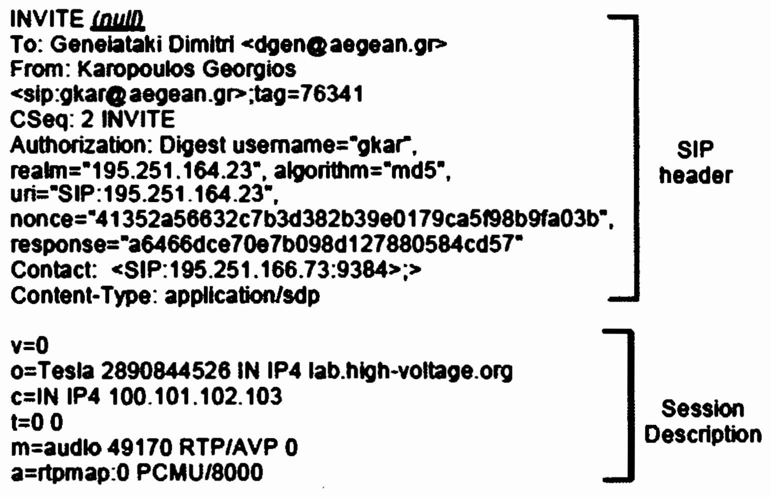


Figure- 4.3: Alternate Flooding INVITE scenario [21]

Since UDP is used as a transport protocol for SIP and media, a large amount of UDP packets can be sent to congest the link and network elements resulting in packet drop which results in legitimate user's packet loss. Consequently, a user will have to wait for long time, trying again and again to make his call.

b) SIP Parser Attack

An efficient parser is required which parse messages up to the point the information is required. However a valid SIP message can be constructed that can obstruct proper parsing. For example, an attacker can construct unnecessarily long messages by simply adding additional headers (like informational header fields) along with a large message body. Many SIP messages may include bodies even when they don't need to be. This will deplete processing power and also use extra memory and bandwidth. The attacker only needs to construct well structured header not to be ignored by the parser. So, server should check messages for their size limits. In some conditions, clients have to send messages using TCP which may open other possibilities of TCP attacks. Due to implementation errors in some circumstances, such attacks can lead to server to crash.



```
INVITE (null)
To: Genelataki Dimitri <dgen@aegean.gr>
From: Karopoulos Georgios
<sip:gkar@aegean.gr>;tag=76341
CSeq: 2 INVITE
Authorization: Digest username="gkar",
realm="195.251.164.23", algorithm="md5",
uri="SIP:195.251.164.23",
nonce="41352a56632c7b3d382b39e0179ca5f98b9fa03b",
response="a6466dce70e7b098d127880584cd57"
Contact: <SIP:195.251.166.73:9384>;>
Content-Type: application/sdp

v=0
o=Tesla 2890844526 IN IP4 lab.high-voltage.org
c=IN IP4 100.101.102.103
t=0 0
m=audio 49170 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Figure- 4.4: Example of Malformed INVITE message

It is also quite possible that an attacker will try various malformed messages to figure out any security flaw or problem towards a SIP victim subsystem. For example the message in Figure-4.4 is invalid and cannot be generated by a standard SIP Protocol Syntax due to the lack of REQUEST-URI which must follow the INVITE method [32].

The attacker may launch a brute force attack of malformed packets; exhaustingly try all possible SIP message combinations. An attacker can follow more focused approach by discovering the target's SIP capabilities, constructing the malformed packets accordingly and then testing these crafty messages against the SIP targets. In this approach the assault cannot be easily identified. More detail about such attacks can be found in [21] and [33].

c) Message Flow Attack

This is another type of DoS attack in which service become unavailable at some moment or the other. SIP protocol specification provides method to terminate a session, cancel an invitation, and redirect a call and update of session parameters. These methods are (i) BYE, (ii) CANCEL, (iii) REFER, (iv) Re-INVITE, (v) UPDATE, (vi) INFO [21]. Let us see some details of attacks possible on the above methods.

i. BYE Attack – The BYE request is used to terminate an established session. An attacker can use this method to tear down a session. The attacker only needs to learn some session parameters (like Session-ID, RTP Port, etc) to launch such attack.

ii. CANCEL Attack – A CANCEL request is used to cancel a previous request sent by a client. More precisely it asks the corresponding server to stop processing the request and generate a corresponding error message. Attacker may use this method to cancel INVITE message or some other message. So, the server will not process that request. Moreover these requests cannot be challenged by the server as they are generated in hop-by-hop manner.

iii. REFER Attack – This method provides a mechanism in which one party provides an arbitrary URI to another party. Assuming that this URI is a SIP URI, the referee will send SIP request (usually INVITE) to that URI. So this method can be used for many applications like call transfer. This scheme enables the referee to act as an eavesdropper giving him the ability to launch MITM attacks. Details can be found in [21].

iv. Re-INVITE Attack – Once a session has been established, its parameters can be modified afterwards. Thus, any unauthorized modification with a forged Re-INVITE can be a potential threat that can cause DoS.

v. UPDATE Attack – SIP UPDATE method gives opportunity to update an existing session like placing call on hold, muting and negotiation of other session attributes. An attacker may send forged

UPDATE message in order to modify initial session parameters to cause DoS.

vi. INFO Attack – This method is used for communicating mid-session signaling information along the signaling path for the call. The message body of INFO is not encrypted hence does not provide integrity and confidentiality. Thus malicious modification of INFO method is possible and it can cause serious problems for communication parties like unauthorized access to a call, DoS for the initial invitation, billing errors and so forth.

4.2.5 SIP Billing Attacks

This is not really a type or category of attack, infect it is the result of MITM attack on SIP server. Billing is critical for any commercial VoIP service and it is very important for any customer of that service to be charged only the amount he or she has used the service. In [23] the authors practically launched attacks on two commercial VoIP services and showed how vulnerable they are. They presented four billing attacks on VoIP Subscribers. We are going to present a brief description of those. Detailed attacks can be found on [23]. An example of normal SIP flow of Call Setup and Tear Down is shown in below diagram Figure-4.5.

They launched four types of attacks:

i. INVITE-REPLAY Attack – As the name suggests, an attacker intercepts INVITE message and use it later on to get unauthorized access to the service. This attack cannot be avoided even if INVITE messages are protected by SIP authentication. The attacker (MITM) who is in-between the SIP phone and the server can observe and intercept all SIP messages in communication between the parties who then replay these messages by modifying INVITE messages according to his own desire.

ii. Fake-Busy Attack – The attacker can surely hijacks VoIP calls of targeted VoIP subscriber through this attack and can also control the call duration. As a result the subscriber will not be able to make the call but still be charged by billing system. Two MITMs at both sides will take over the call soon after actual sender will send INVITE request which will not reach its real destination. Sender will be received BUSY tone and actual destination will never come to know about this call.

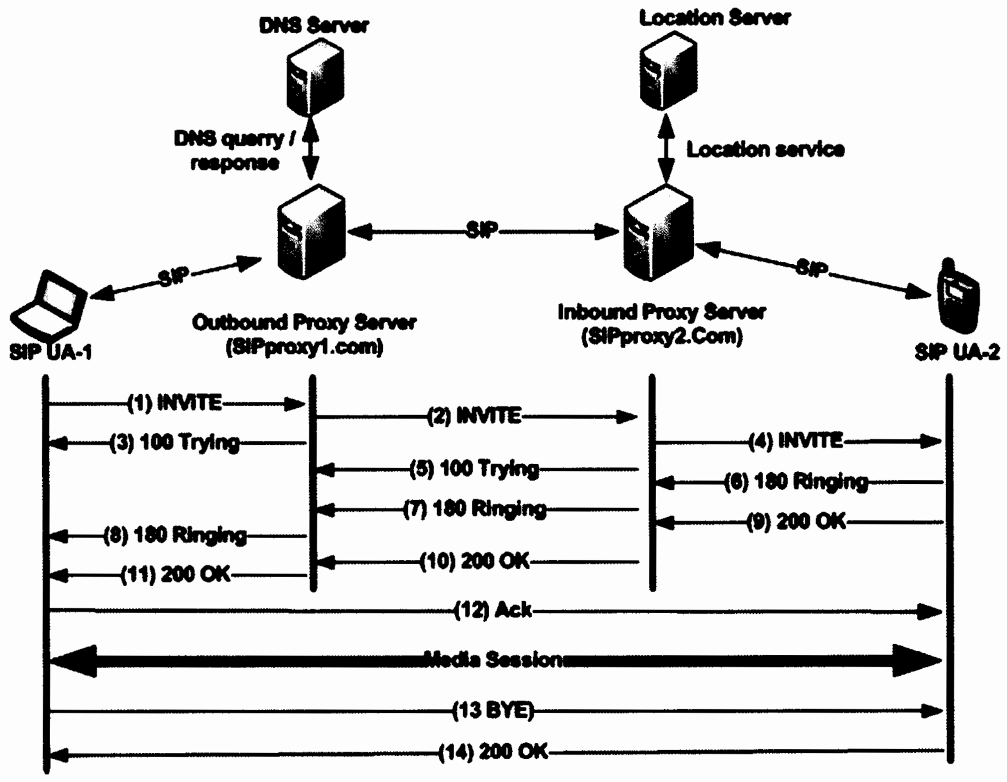


Figure- 4.5: An Example of SIP Call Setup and Tear Down

iii. BYE-Delay Attack – This is another example of MITM similar to previous one. When a caller and callee send BYE message to terminate the call, MITM intercepts it and send 200 OK messages. Caller & callee will have impression that the transmission is terminated successfully, but actually MITM's have taken over the call. The subscriber will be charged by the system for longer period of time as call duration will be prolonged.

iv. BYE-Drop Attack – In this attack, the attacker (MITM) will simply drop BYE message which will prolong the duration of call. To terminate the call MITM will send BYE message.

4.2.6 Social Threats

The problem of ***Social Threats*** has potential to become a major nuisance for VoIP subscribers. This attack has broad spectrum of operation ranging from unsolicited communication to data steeling attacks. The term "unsolicited" is not absolute and strictly relative term as it differs from person to person for which he calls unsolicited. This is the reason why these attacks are difficult to identify.

Any unsolicited communication is referred to as Spam and is actually abuse of electronic messaging system. Spamming is an economically viable solution of advertisement and telemarketing, having nominal operation cost with huge mailing list. Most of these attacks are generated with machines (bot-nets) particularly programmed for such jobs. It can be done on different media like email, instant messaging, mobile phone text messaging, and in current scenario SPAM over Internet Telephony (SPIT). Voice advertising to sell products is an example of SPIT. Unlike email where spam messages simply come in the mailbox and can be checked and removed at any time of day, spam calls are more disruptive as they require momentary response which might become a nightmare for subscriber.

A variant of SPIT is VoIP phishing (Vishing) attack. This is similar to other types of phishing where an attacker masquerade as trustworthy third party and try to acquire confidential information from user. For example a phisher can send an email link to a victim who leads him to a fake webpage which looks like his bank home page and ask to enter some sensitive information. In case of VoIP Phishing an attacker can entice the victim to dial some expensive phone number (like 0900 numbers) to win some prize or participate in lucky draw. The victim may also redirect to an Interactive Voice Responder (IVR) pretended to be trusted and get users personal or confidential information.

Spammers create the environment of untrust and bother users of any particular service. As SIP is an emerging standard and becoming popular day by day, it can be favorite choice of spammers in near future. Identification of spam and mechanisms to deal with it is really crucial at this stage before problem come to surface and becomes monster.

4.3 Available Security Mechanisms for SIP

SIP specification does not include any mechanism for security. Instead it suggests using existing well defined security mechanisms for HTTP & SMTP. The fact is that some mechanism is required to protect the transmitted data against modification, eavesdropping, session disruption and hijacking, imitation, and so forth. These attacks can take place either in signaling or in media transmission. Thus both signaling and media is required to be protected through some security mechanism. As indicated in Figure-4.6, SIP security can be provided wither in hop-by-hop or end-to-end fashion [18].

To implement security in SIP, secure SIP (SIPS) URI scheme is advised. This required the use of TLS hop-by-hop encryption with exception of the last hop which may use some other encryption scheme. SIPS protects the signaling stream against attackers which try to listen on the link. More detail can be found in RFC3261 [9].

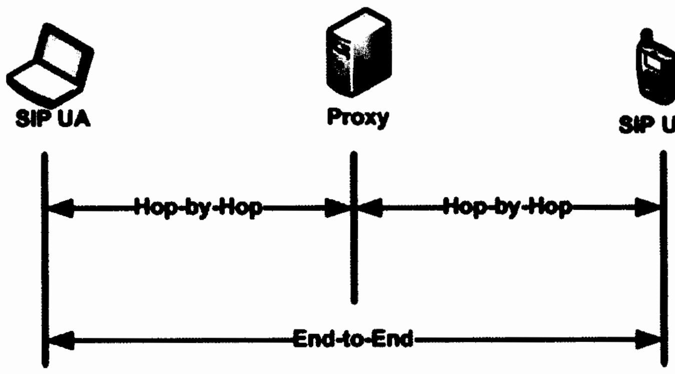


Figure- 4.6: SIP Security Mechanisms

More specifically, the following security mechanisms are described in [21]:

a) **SIP Authentication:** Sip Authentication is inherited from HTTP Digest authentication, specified in RFC 2617 [32], which is a challenge response based protocol. This is the most frequently deployed method with SIP for verifying the identity of users and performing message authentication. A server (Proxy, registrar or redirect) might want to authenticate the sender of SIP request before forwarding. We have already looked on an example of such authentication in section 1.3.4 where a SIP Registrar server challenges a user agent client upon receiving REGISTER message. More detail can be found in [32] and [22].

b) **IPsec:** As SIP use IP protocol on network layer for data transmission, so this transmission is vulnerable for spoofing, session hijacking, traffic analysis and other attacks on IP protocol. To deal with these issues, IPsec is proposed which protect IP from such attacks. It utilizes Encapsulating Security Payload and Authentication Header protocols which provide confidentiality, integrity, data origin authentication, anti-replay and traffic analysis protection.

c) **Transport Layer Security (TLS):** This is another way to protect transport layer. SIP mandates that SP proxy servers, redirect servers and registrar server must support TLS and HTTP Digest based authentication. This authentication is based on SSL and run over TCP enables confidentiality and integrity. During handshake procedure network elements can authenticate each other by exchanging their certificates. Self signed certificates can also be used but not preferred. Both TLS and IPsec work in hop-by-hop manner to secure SIP session.

d) **Authentication Authorization Accounting (AAA) Services in SIP:** Another way to enhance security of SIP based VoIP infrastructure is introduction of Authentication Authorization Accounting (AAA) Server. Rather to store user's credentials and profiles locally on each server, another dedicated AAA server is proposed to introduce in VoIP system which will authenticate and/or authorize users and also provide accounting services. This server has many other advantages, for example this gives the ability to administrator to dynamically configure the type of Authentication and authorization required. More details can be found in [21].

e) **S/MIME and SIP:** SIP messages are capable of carrying MIME bodies. Secure MIME (S/MIME) can be used to implement security services. SIP uses Integrity and Authentication Tunneling and Tunneling Encryption functionalities from the set of many offered by S/MIME. This allows end-to-end encryption and data integrity. The SIP header fields which are needed by intermediate proxies for routing cannot be encrypted. This method prevents eavesdropping and media injection attacks as message bodies containing SDP are encrypted.

4.4 Analysis of SIP Security Mechanism

It is already mentioned that HTTP Digest is the most popular authentication mechanism used for SIP. It can offer one way message authentication and replay protection but cannot support message integrity and confidentiality. Spam calls by malicious user are possible. Moreover due to plain text of SIP messages, MITM attack is also possible as both plain text and cipher text can be sniffed very easily. The downside of this method is that in actual implementation UA cannot challenge any server. Digest authentication also need prearranged trusted environment for password distribution. As there is correlation among the username and SIP URIs, a malicious user may impersonate itself as a legitimate user. More details can be found in [21].

Introducing IPsec in SIP can safeguard signaling and data from the vulnerabilities, provided that some sort of trust relation has been established beforehand between communication parties (e.g. pre-shared keys, certificates) and it can only be utilized in a hop-by-hop fashion. IPsec is implemented at Operating System level. Most SIP clients do not implements this protocol yet. Therefore it can only protect the traffic between the corresponding network servers.

TLS can be used either for one-way or mutual authentication schemes. Perhaps it is more suitable for inter domain authentication. Similar with HTTP Digest, TLS do not need to be supported by UA's. TLS failed to deliver end-to-end security as no mechanism exist to ensure that TLS is being used by all entities along the route. In addition, to maintain many TCP connections simultaneously may be too heavy for proxy servers. Moreover, TLS can only be utilized on connection oriented protocols and could not be used with UDP *(which is preferred protocol for SIP signaling)*.

S/MIME is used to ensure integrity or confidentiality in an end-to-end fashion. But the shortcoming is that S/MIME adds considerable overhead in SIP messages. Moreover, confidentiality or integrity of entire message cannot be ensured due to existing restriction of header modification. This is because intermediate nodes must have access to SIP header to process and rout SIP messages to its destination.

In some cases, security services may require combination of TLS and S/MIME. This is because to get integrity and authentication from TLS, while S/MIME is used to provide mainly privacy for some parts of transmitted data. More details can be found in [21] and [23].

4.5 Snort and SIP support

As Snort is a favorite tool among network and system security community, so they expect to use it in SIP based VoIP systems and networks. In Snort-2.8.5.3 and snortrules-snapshort-2.8 available to date *(when writing this chapter)*, VoIP-SIP rules are available in a rule file *voip.rules*. For example, below is a rule taken from the file:

```
alert udp $EXTERNAL_NET any -> $HOME_NET 5060 (msg:"VOIP-SIP Via header missing SIP field";
content:"Via|3A|"; nocase; pcre:"/^Via\x3A\s+(?!SIP\x2F2\x2E0)/smi";
reference:url,www.ietf.org/rfc/rfc3261.txt; classtype:misc-activity; sid:11975; rev:2;)
```

This rule is checking the header fields of SIP message and alert if find *Via* field missing from header. Via field indicates the transport used for the transaction and contains the address/location where the response is expected to be received. When UAC creates a request it is MANDATORY to insert Via into that request. The details can be found in RFC3261 [9]. This event is triggered when an attempt is made to send a malformed communication to VoIP-SIP device. Although the rule says that its impact is unknown but the request is not legitimate, anyhow. This may be a policy violation.

Some of these rules are addressing issues like some field buffer overflow attempts, known vulnerabilities of CISCO devices, request structure problems, or similar issues that are more related with structure of SIP messages. These are all known types of attacks where we can be sure about a potential intrusion attempt.

Now what is the dilemma with this approach? We have already discussed in previous chapters that VoIP and especially SIP based attacks are diverse in nature due to their architecture. This simple rule matching is certainly not enough to make Snort an effective tool for deployment in such infrastructures. Snort needs some comprehensive and strong mechanism to detect intrusive activities in VoIP architecture and SIP services.

4.6 Problem Definition

As the technology improves issues related with it also become prominent. One of the most prominent issues in IP based applications is security. The security threats and problems in SIP-VoIP are different from traditional technologies and are diverse in nature. This is because of its distributed nature, having components in more than one administrative control, real-time characteristics and involvement of many protocols. For example MITM, Billing Frauds, easy to launch Denial of Service attack, and message or call Spam are few important threats.

Available Intrusion Detection techniques and architectures are either not able to accommodate wide variety of IDS techniques or not well supported in distributed environment like VoIP. The techniques available in literature for VoIP are severely lacking information correlation among VoIP elements and hence are not able to deal with some important attacks.

On the other hand, when we look on Snort for SIP-VoIP system, we come to know that this is not a tool of choice for a security personal for deployment. It lacks the capabilities of intrusion detection in SIP network. Snort needs some sort of extension to cover this gap to fit in SIP-VoIP environment.

4.7 Research Objectives

The objective of our research is to design architecture and mechanism of intrusion detection for SIP based VoIP infrastructure by using Snort. We shall try to present a system which has potential to fulfill security requirements in VoIP infrastructure and can be one of the candidates of choice for security experts. We shall discuss the importance of different SIP elements and their importance in sharing information. Hence the placement of IDS and its sensors will also be discussed. The enhancement of capabilities can be done by adding new misuse detection techniques or some misuse detection techniques. We shall also propose the changing the architecture of Snort to effectively work on selected environment.

4.8 Summary

VoIP is exposed to vulnerable environment which need to understand before taking measures to deal with them. People using VoIP definitely need minimum security and privacy level as offered by traditional PSTN network. They don't concern about complexity of issues. In this chapter we have tried to present a broader spectrum of security issues related to VoIP and focused on the threats particularly related to SIP, which is core of our thesis. A lot more can be said on each issue and as a matter of fact tons of material is already available in literature. Researchers and enterprise community has already started work on security and privacy of VoIP. They are developing methods to build secure and robust VoIP infrastructure and tools for its testing.

Important thing to note is that besides inheriting traditional issues faced by common protocols, communication of voice and data into same network increase complexity and open new dimensions for attackers. Therefore, serious precautions should be adopted by users of services and enterprises aiming to deploy VoIP commercially.

Chapter 5
System Design

5. System Design

In previous chapters we have discovered the features of Snort and seen the opportunities that we have to exploit for the effectiveness of this tool. In this chapter we shall describe our proposed architecture and its design in detail which will be really effective in SIP-RTP based VoIP infrastructure. This architecture accommodates different techniques into Snort. Although there are many technologies available and research community is striving to improve methods of intrusion detection, but some of the techniques are really effective in Snort when applied properly.

We have already discussed in section 2.2 that Snort is basically knowledge based Intrusion Detection System, but it can accommodate behavior based techniques gracefully. We shall discuss a method of anomaly detection implementation keeping in view the effectiveness and advantages of classic Snort methods. We shall discuss how statefull and cross protocol analysis is helpful for SIP based VoIP system and how it can be implemented into Snort. Our proposed system is distributed in nature and every sensor of Snort will send its information to central audit server which will further analyze the traffic.

Rest of this chapter is organized as follows: Section 5.1 will describe the stateless nature of Snort. Section 5.2 will show how Snort can be improved. Section 5.3 will describe some important features related to Snort. In Section 5.4 we shall discuss the design features for our research. We shall describe the general architecture of proposed system in detail in Section 5.5. In Section 5.6 description of each component will be discussed in detail. Two scenarios of system implementation for information gathering will be discussed in Section 5.7. Section 5.8 will describe a proposal for Anomaly Detection technique implementation and finally Section 5.9 will summarize the chapter.

5.1 Stateless Nature of Snort

The basic structure of Snort is very simple and hence efficient. It simply captures packets, decode them, forward to detection engine for rule matching and finally log/alert whatever specified. Hence Snort is stateless in nature. As time passed stateful analysis of packets become a requirement. Therefore people started developing codes (in the form of preprocessors) that performs packets reassembly and maintaining states.

A very famous preprocessor in this regard is Stream5 preprocessor (replaces both Stream4 and flow preprocessor). Its documentation *(in Snort -2.8.5.3)* says that it is a target based TCP reassembly module and is capable of tracking sessions of both TCP and UDP. The feature of UDP packet reassembly was not previously available. This preprocessor also does anomaly detection of TCP protocol e.g. data in SYN packets, data received outside TCP window, etc. Other preprocessors (like HTTPInspect *in Snort - 2.8.5.3*) use stream5 preprocessor to achieve stateful reassembly of packets.

We have noticed that packet reassembly and session maintenance is limited to Transport layer. For example stream5 preprocessor only reassemble TCP/UDP packets that belong to the same session. Snort is not capable of session reassembly of application layers and hence SIP and RTP are being handled packet by packet manner in Snort. Similar is the situation with Decoding engine where syntax analysis (parsing) is only performed on layer 2, 3 and 4. It never goes up to SIP and RTP.

As we have discussed in previous chapters that VoIP is based on session oriented protocols (SIP, H.323, RTP, etc), therefore to achieve effective intrusion detection for VoIP systems it is important to maintain session information and packet reassembly. Some of VoIP attacks are session oriented like billing attacks, session hijacking, etc. which can efficiently be tracked after looking on the sessions. We shall discuss the techniques of maintaining sessions for SIP & RTP later in this chapter.

5.2 How Snort can be Improved?

Functionality of Snort can be improved or enhanced through improvement in decoder, detection engine, adding new rules, improvement in currently available preprocessors, building new preprocessors, etc. In reality the only limitation in extending Snort is the imagination. In present situation we have identified three methods for improving Snort:

a) **Improvement of Knowledge based system**

Although Snort is currently working as knowledge based intrusion detection system and use Rule matching engine to decide about the intrusion, but many improvements can be done in its current methodology, at least at the level of SIP/RTP. We can achieve better results if we introduce some level of statefullness in Snort's current structure, in the form of preprocessors and use it to detect more sophisticated attacks. Detection plug-in can also be developed which can work in collaboration with this particular preprocessor.

b) Implementation of Behavior based system

Apart from typical misuse detection system the idea of introduction of Anomaly Detection in Snort is quite impressive. Misuse detection system has its own strengths and limitations. We can retain its strengths and overcome limitations by using behavior based system.

c) Information Sharing between Sensors in Distributed System

We can also think about the situation where Snort IDS's are being deployed on SIP based VoIP elements in distributed fashion. In traditional approach there is central Snort management system which is used to administer rest of Snort sensors. It does not perform any sort of detection analysis itself. But it will be a good idea if we are able to collect information from all of sensors to this central server and analyze them in correlation approach. In this way tasks of intrusion detection will be divided over different components hence performance will not be degraded a lot and correlation techniques can be accommodated very easily in the system.

5.3 Important traits of Snort

Before discussing the proposal for Snort, we would like to summarize some important features associated with Snort:

- Snort Decoder performs decoding and syntax analysis of layer 2, 3 and 4 of IP packets
- Snort preprocessor is the block where analysis of Application Layer (above layer 4) can be done (e.g. syntax analysis, security checks, etc.)
- The coding language of Snort is C/C++ with available Snort Preprocessor development API and template
- We can develop knowledge based system, behavior based system and also implement stateful analysis in Snort Preprocessor
- Detection engine can only be used to implement knowledge based system because of its reliance on the rule sets defined at startup

- Additional Detection Plug-in can be developed to augment the functionality

By analyzing the above points it is clear that a preprocessor is required to implement the required functionality. We refer again to Niccolini's work [44] for preprocessor architecture. We are not considering IPS functionality in our system.

5.4 Design Features

There are many techniques and architectures that are coming up for SIP/VoIP having very good results for particular types of attacks. But none of them cover the broader spectrum of attacks or they are non-scalable. In most of the cases, proposed techniques cover only some of components of VoIP and specialized to detect a small set of attacks.

In this section, we shall discuss design features for building SIP IDS system architecture. VoIP system is distributed in nature (having distributed network elements) and uses many protocols to get the job done. Attacks to such systems span a large class, from DOS to Billing Fraud. Moreover, VoIP systems are heterogeneous and practically under different administrative domains. Therefore information collection and correlation for audit can bring many attacks onto the surface. Such system can be very helpful to detect more sophisticated attacks. Additionally, cross protocol feature will bring correlation among linked protocols.

5.4.1 Stateful Analysis

Despite of the fact that Stateless protocol analysis is more efficient under heavy traffic load, faster and easy to implement, yet stateful protocol analysis is required and preferred in situations like VoIP. Stateful detection means to assemble states from multiple packets and use the resulting aggregated state for rule matching engine. We have already discussed in section 5.2 that there is no such mechanism available in Snort and states are only maintained at the level of Transport layer of the same session (in case of TCP). In our scenario, we need to keep states of SIP and RTP essentially to get clear picture of what is going on in target system or service. We know that while making an audio or video call by using SIP & RTP, call management protocol (SIP) has to be involved in the total duration of the call until it is terminated. During an established call, different features of current conversation can be changed. Billing is usually done at the end of the call. A stateful IDS is aware of at what stage

VoIP communication session is situated. Moreover, a state presents a complete context about what is going on during a call. By looking all the above facts and features offered by SIP, it is strongly proposed to use some sort of stateful system for intrusion detection.

5.4.2 Cross Protocol Analysis

Similarly, we know that there are multiple protocols involved in VoIP for call management and data delivery. SIP and RTP are the two most important one. SIP establishes and controls a call while RTP transmit voice over the end points. Therefore, both of these protocols work in collaboration. Cross protocol audit means to analyze packets of different protocols involved (SIP/RTP) in a VoIP session. For example, in a simplest scenario between two parties

- At time T1, SIP establishes a call
- At time T2, RTP communication is started,
- At time T3, SIP terminates the call

So, from time T1 to T3 there is one complete session which involved SIP and RTP protocols communication. Suppose if SIP terminates a call and RTP stream is not stopped, it means that there is a problem somewhere in the system. Such issues cannot be detected by simply analyzing SIP or RTP alone and only correlation among protocols reveals these matters. In subsequent portions we shall discuss some more aspects of cross protocols analysis.

5.4.3 Components of SIP-VoIP system need to Audit

There are few things important to understand before starting audit of any system. The most important thing is the critical resources of the system under review. Understanding the VoIP network components is vital in our auditing system. Every component which possesses information regarding a SIP/RTP session can contribute for audit. We have identified the following SIP components which acts as critical information assets in SIP VoIP communication.

- Proxy Server – Accepts session request from source and forward it to destination. This may also keep information of the user rights in that system/domain
- Redirect Server – Allows Proxy server to redirect SIP session invitation to other Domain
- Registrar Server – A database that keeps locations of User Agents within a Domain
- Billing Server – A Server that keeps billing records of its domain users
- User Agents – End devices used to make and receive calls – SIP Soft/Hard Phones

The servers identified above are only logically segregated. Practically they may be part of one single system installed on single server machine. But this is a good idea to separate them on the bases of their functionality. Connections of VoIP servers to external network are also important source of information collection in this audit. We have already identified that the VoIP network is distributed in nature and we have to collect communication information of VoIP from its scattered resources. The second important thing is the vulnerabilities list that should be checked against that system.

5.5 Proposed Architecture

The idea behind this architecture is that there are many components participating in SIP/VoIP communications which possess pieces of information about SIP session and can contribute to detect different types of attacks. Therefore all these components, at least important ones, should be monitored against attacks. Hence our system is distributed in nature to some extent. Moreover, if we gather information from these components and correlate with each other, we may be able to detect some attacks which are not possible otherwise. Therefore our system also gathers information from its distributed sensors to a centralized location so that this information can be correlated to identify attacks with more confidence. We shall discuss these aspects in detail later in this chapter.

There are many requirements for an intrusion detection architecture or mechanism for SIP-VoIP system which we have tried to accommodate in our system. For example:

1. The system should accommodate both anomaly detection and misuse detection techniques.
2. It should differentiate between one types of attack from the other one precisely, on the basis of information collected from different components for the same session.
3. The system should be scalable so that any improvements and enhancements can be accommodated easily.

In addition to these, the proposed system also uses stateful detection technique by looking not on individual packets but to a complete session. It also has capabilities to look into other protocols involved in communication, like RTP and IP, to detect attacks which span on multiple protocols.

Basic Architecture:

Snort has the capabilities to work in distributed environment where Snort sensors are responsible to collect information and process accordingly. In this way only the central server is responsible for managing all components or sensors. This scenario is usually deployed in environment where there are multiple elements involved in some business logic. Similar is the situation with VoIP having many components, some of them are even distant from each other. In Snort sensors, although the central server manages all of them, but they work individually and perform some information processing locally.

We have taken the same concept of distributed architecture. SIP based VoIP components are discussed in section 1.4.2. SIP Proxy Server, Registrar Server, Redirect Server, Accounting Server, and User Agents play important role in a whole session. Whereas, Proxy Server, Accounting Server and User Agents are important, because of their consistent presence in the session. Therefore in this system we have focused our discussion to these elements only. As the system is extendable, therefore other elements can be added into it very easily. This mechanism may be proved as a guideline for building robust security architecture.

The theme of this architecture is that all the sensors will be collecting particular information and process it locally to find some attacks. After that, these sensors convert this processed information into a format, a snapshot, which describes the whole session. At the end the sensors watch for any anomaly or other session problems in these snapshots. There is a central Intrusion Detection Audit Server which is responsible to correlate information of different IDS sensors. All sensors, after their own processing and converting the data into session oriented information, send this snapshot to this central audit server. Some other important information is also included into it. The server will collect snapshots from its sensors and arrange it into a group of same session. It then looks into these snapshots and try to find out integrity of messages and other anomalies.

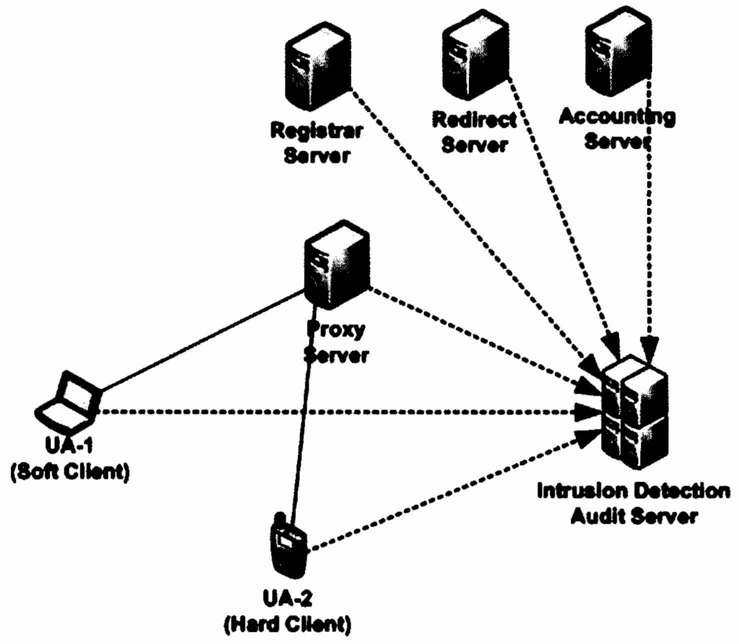


Figure-5.1: Snort Sensors sending Snapshot to Central Audit Server

In every sensor, there are three essential steps that are being performed to make a snapshot. In every step, the captured information is processed before it is forwarded to next abstraction level. Each abstraction level is called profile where small units are combined to form a bigger unit. When packets arrive in the sensor, they are first inspected according to the logic of intrusion detection. These packets are combined to form a transaction which is second level of abstraction. These transactions are then combined to make Dialog which is third stage of Abstraction. We shall use this logic on both Proxy Server Sensor and User Agent Sensor. Figure 5.1 below is showing this concept in generic way. We are going to discuss each component in detail.

5.6 Methodology

The details of the sensor's architecture are below:

a) Sensor on Proxy Server:

Proxy Server is involved as the conversation starts till it ends. Therefore it has the most valuable information regarding the whole session. This is the reason that intrusion detection approaches developed for such nodes covers much more details about the communication and hence cover broader spectrum of Server based attacks. We already have mentioned some facts about Snort in section 5.4 like Syntax Analysis of application layers is not performed in decoder block. For that we need to implement this functionality into preprocessor block. Figure 5.2 shows the schematic block diagram of Proxy Server sensor.

We have to keep in mind that this is intrusion detection system and therefore the result of the sensor is alerting and/or logging only. The functions of these blocks are:

i. Filter: This is the first check on packet entered into Snort sensor which allows only SIP packets to enter into preprocessor block. The decision is based TCP/UDP port number which is already configured at startup time in configuration file. Ports usually in use by SIP based VoIP service is 5060 or 5061. It depends upon the service provider which port he wants to use. If this port number is not matched, it means that traffic is not SIP and hence packets are ignored. Moreover, some other checks are also performed using *oSIP* library to see whether the request is legitimate or illegitimate.

ii. Syntax Analyzer: We have already mentioned that Syntax analysis is not performed by decoder block of Snort. Therefore the message received is parsed according to the rules of SIP. Again *oSIP* library function (*osip_message_parse*) is used. If some problem is found an error or alert is registered.

iii. Security Analyzer: After the SIP message is parsed, it is checked against mandatory fields whether they are present in correct form or not. Size of fields is also checked. If some problem is found an error or alert is registered.

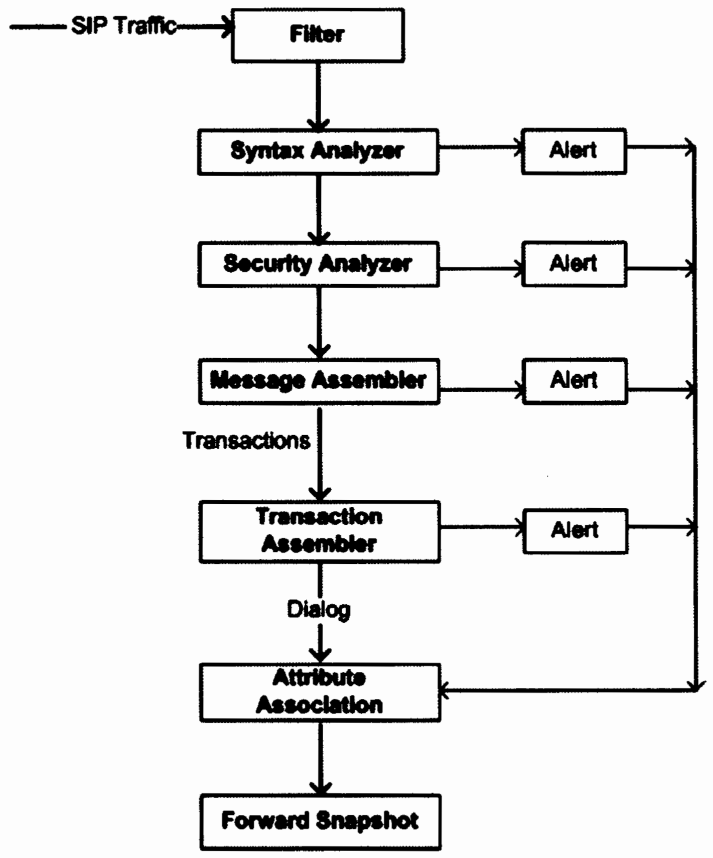


Figure-5.2: Snort Proxy Server Intrusion Detection Architecture

iv. Message Assembler: The messages analyzed in previous step of the same session are grouped or assembled to make a Transaction. RFC 3261 defines a Transaction which consists of a request that invokes a particular method or function on a server and ultimately at least one response is generated. SIP use HTTP like request-response paradigm. This block also checks the correctness and completeness of transaction according to RFC3261. If found some issue an error message is registered.

v. Transaction Assembler: In the next step of abstraction, these transactions of same session are combined to form a complete dialog. All the transactions are added until the session is terminated gracefully. In case there is immature session termination, e.g. one of user agent goes down unexpectedly, the dialog will be considered complete and depicting the same fact. This dialog is a complete picture of a successful or unsuccessful session.

vi. Attribute Association: At the end or completion of dialog some other attributes are associated with it showing additional information gathered by this sensor. These attributes include Call ID of Dialog, session start time, session end time, whether session started with INVITE message or end with BYE message, associate error messages found by this sensor in previous steps, etc. The result of this activity is that a complete snapshot is developed which is comprehensive in nature.

vii. Snapshot Forwarding: At the end of this process, Snapshot created in previous step is forwarded to the central audit server for further inspection and correlation with snapshot received from other components.

b) Sensor on User Agent:

User Agent is the end device which is usually present far away at some remote location. This sensor resides on both User Agent Client and User Agent Server. These are essentially involved in any session from start to end and hence are important to be monitored. They also possess the first hand information about what they are sending to server or other party. This is the reason that information collected from these nodes can be proved very vital for checking integrity of messages. Some of the intrusion detection approaches only rely on information arrived or possessed by this node. Some also prefer to share information with its counterpart. First approach is limited in finding attacks and the second approach is practically not applicable. Therefore in our scenario we have proposed that these nodes should also share its information with central audit server who has the responsibility of information correlation. The architecture of this sensor is similar with that of proxy server sensor. The objective is simple, to get snapshot of the whole session. Figure 5.3 shows the schematic block diagram of the architecture of Sensor implemented on Proxy Server.

The description of each block is:

i. Filter: This block receives the packets and check if they are SIP messages. If found then it forwards them to preprocessor block.
ii. Message Assembler: In this first step of abstraction, the messages of the same session are grouped or assembled to make a Transaction. This block also checks the correctness and completeness of transaction according to RFC3261. If found an issue, error message is registered.

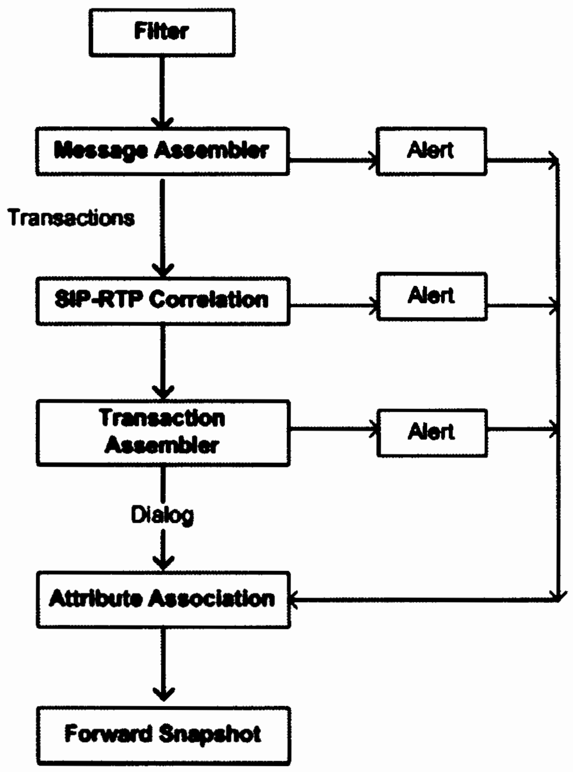


Figure-5.3: Snort User Agent Intrusion Detection Architecture

iii. SIP-RTP correlation: This block is responsible to perform two steps (a) It will check the integrity of RTP messages by investigating whether RTP stream is coming from the same source who is generating SIP messages. This step may not be fully observed if some RTP gateways are present in between client and server. (b) The second step is the attribute extraction from RTP stream. These attributes may include IP-address of the counterpart, etc. This information may be proved helpful if other User Agent does not belong to the same service provider.

iv. Transaction Assembler: In the next step of abstraction, these transactions of same session are combined to form a complete dialog. All the transactions are added until the session is terminated gracefully. This dialog is a complete picture of a successful or unsuccessful session.

v. Attribute Association: At the completion of dialog other attributes are associated with it showing additional information collected by this sensor. These attributes include Call ID of session, session start time, session end time, whether session started with INVITE message or end with BYE message, associate error messages found by this sensor in previous steps, etc. The result of this activity is that a complete snapshot is developed which is comprehensive in nature.

vi. After successful termination of a session/call, there should be no RTP stream received from the other end. This case may arise when this node receive BYE message from other end which is not actually sent by legitimate counterpart. It may have been generated by some attacker to terminate the call from one side to take over conversation. Man-In-The-Middle (MITM) attacks may end up with such situations. This snapshot also included attributes extracted from RTP stream in second step.

vii. Snapshot Forwarding: At the end of this process, Snapshot created in previous step is forwarded to the central audit server for further inspection and correlation with snapshot received from other components.

The call end up in one of these three forms: Call may be completed and end gracefully; Un-Successful call establishment e.g. callee is not available; Immature call termination e.g. User Agent goes down or some attack. Whatever the situation may be, the dialog will be sent to the central audit server.

c) <u>Sensor on Accounting Server:</u>

This is the service which keeps accounting of all subscribers of VoIP service for particular service provider. This service may reside in Proxy Server, but logically it can be separated on the basis of its functionality. It is quite obvious that service providers with large pool of subscribers will go for separate server running accounting and other databases. When a Proxy server receives a request of call, it will consult with Accounting service whether the user has sufficient balance in his account or not. When the call successfully establish with User Agent Server, Accounting service will start keeping record of time and make sure that user does not exceed the limit of balance.

Whether the session/dialog is complete or incomplete the accounting service will be instructed by Proxy Server to stop billing particular user. Now this service will send snapshot of this accounting to the central audit server so that it can counter check with snapshots received from other sensors. This snapshot may include username, phone number, Call ID, call duration charged, add-on services if utilized, etc.

Important thing to note here is that it is not necessary to place a Snort sensor to monitor accounting service. In fact it will be more appropriate that the accounting software send snapshot to the central audit server. This can decrease complexity of the system.

d) Central IDS Audit Server:

This is the server which collects information from all parties involved in VoIP communication and correlates with each other to find out attacks not possible otherwise. This Snort Server resides on independent machine, which is required to be an average machine, residing in service provider's network. Therefore it is solely in control of service provider administration. As the audit server is not working in real-time mode, therefore performance is not a big issue. The server can generate alerts even after certain time span. Moreover, we can also manage sensors through this server by turning features on/off or even updating certain features. Figure 5.4 shows the schematic block diagram of the architecture of Central Audit Server.

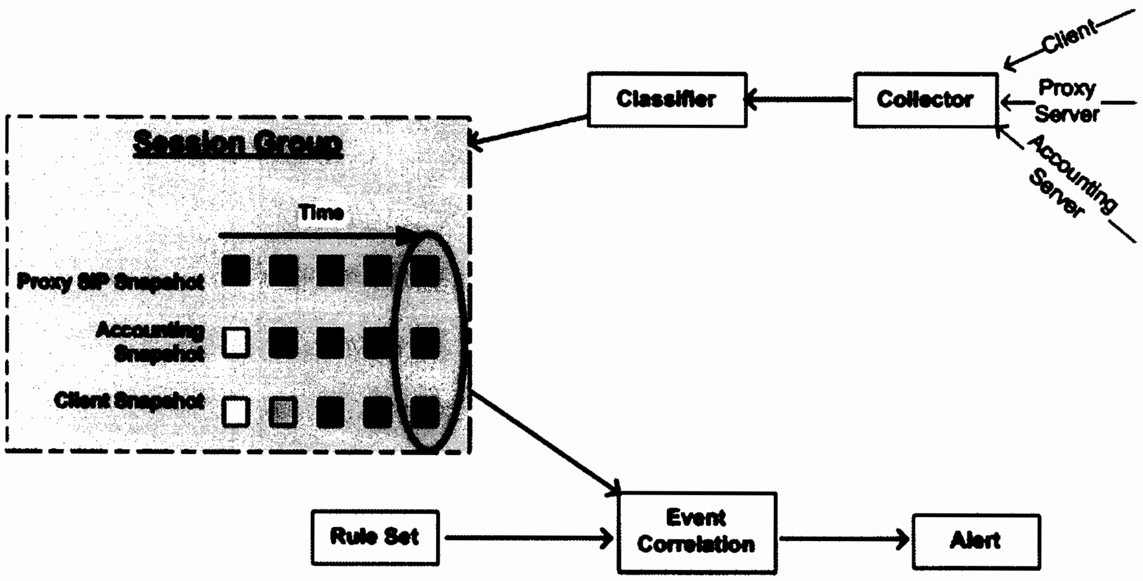


Figure-5.4: Central IDS Audit Server Architecture

The description of each component of this server is:

i. Collector: This block is responsible for collecting snapshots from all elements. In our case, these elements will be Proxy Server, User Agents, and Accounting Server.

ii. Classifier: It will receive the snapshots from collector and separate them according to the session. If there is one session, there will be three snapshots for that call. These snapshots are grouped into a logical unit called Session Group. In addition to this separation and classification, this block also separate attributes associated with snapshots so that these can be used for further analysis.

iii. Session Group: This is a logical block keeping snapshots of all components of same session along with their attributes. At this stage the information collected from sensors are ready for further analysis.

iv. Event Correlation: This block possesses the main functionality of this audit server. This block will perform the following functions:

- If there is a transaction charged in Accounting Service and showing in snapshot, it should also be present in Proxy Server snapshot and User Agent Client snapshot. This block will check its presence. In addition to this, this function will also check the duration of call registered in accounting server and to that showing in user agent's snapshot. By this way billing attacks can be identified e.g. BYE, Call Hijacking, etc.
- Proxy Server snapshot will have the complete dialog showing all transactions. This block will check the integrity of these transactions with user agent dialog transactions. There are multiple checks performed by this function e.g. checking for modification attacks caused by eavesdropping, spoofing attacks, etc.
- SPAM call can be identified by looking into fields of caller or User Agent Client SIP messages. If such calls are made by some caller of the same network, it is very easy to identify. In this way appropriate actions can be taken accordingly. But if SPAM call is being received from outside the network to some number inside this network, it can be identified by other techniques.

v. Alert: The system will generate alert in case found any problem. This alert will also include the error messages found in sensors as they are propagated to this central server along with snapshot.

5.7 Scenarios of Information Collection

It is very important to know at what extent IDS can collect required information. In the technique described above we come to know that information from almost every component of VoIP infrastructure can be helpful in detection intrusion. There are two scenarios possible. In first situation where there is only one domain and every component is somehow under service provider's control or come under one administrative domain and therefore it can be managed to collect information from every element.

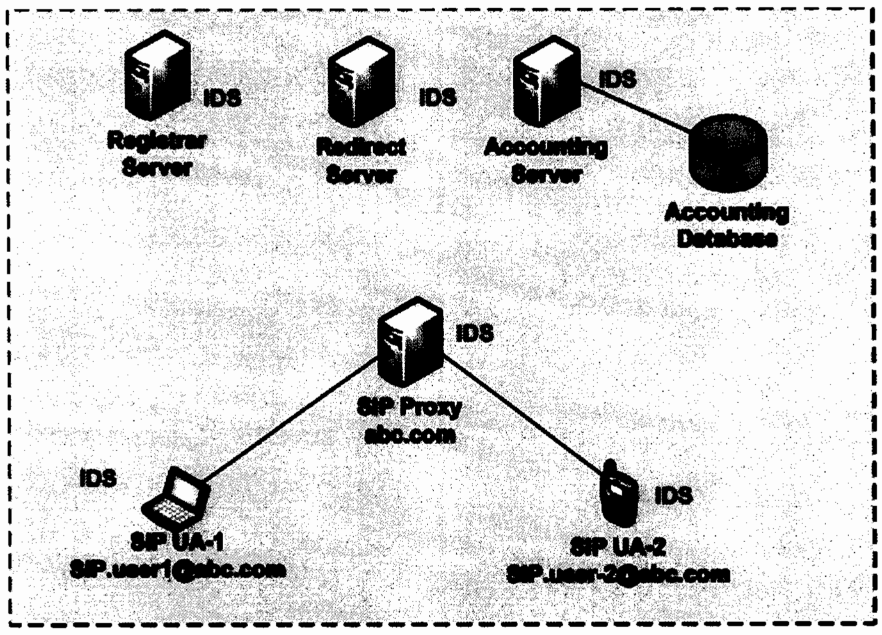


Figure: 5.5 – IDS Implementation in Single Domain

For example, in Skype network where every component is property of Skype having Skype Servers and Skype clients. Here a Skype user can call other Skype user. So, managing every component is very easy, even IDS present in Skype client can be updated online on regular bases. In figure: 5.5 this fact is presented.

But it is not always conceivable. It is possible that some of the components are not under the control of VoIP service provider. For example if there is a network of Skype and a user from Skype messenger tries to call some other network like Yahoo or PSTN. In such case it is not possible to collect information from every component and hence comprehensive analysis is not possible. Skype can implement their IDS on its own components but cannot do anything with Yahoo network elements and PSTN network components. Figure: 5.6 is showing this fact.

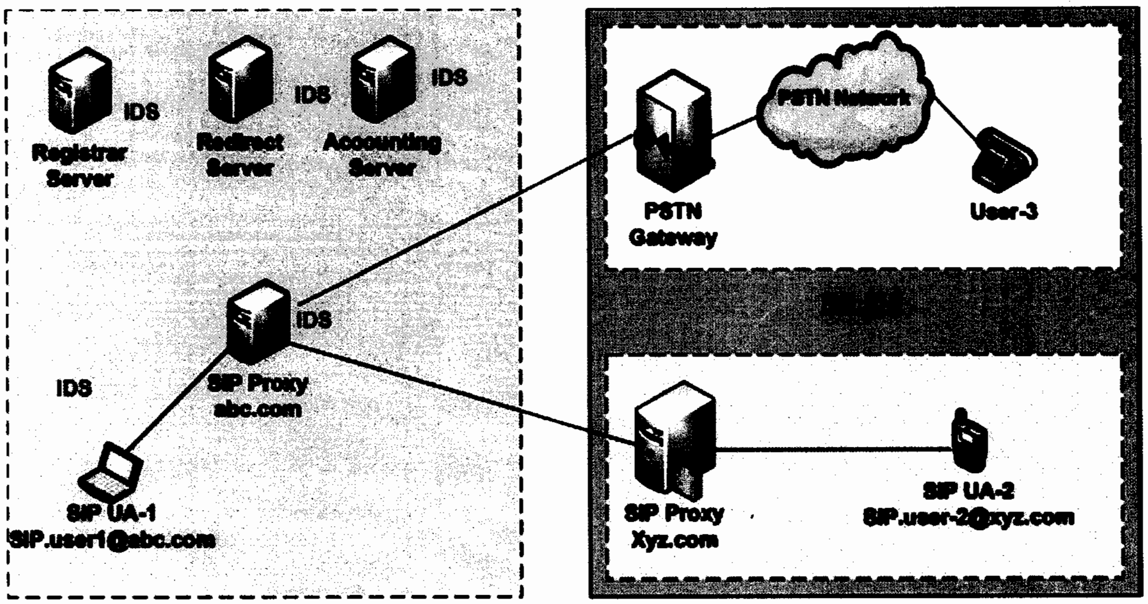


Figure: 5.6 – IDS Implementation in Different Domains

5.8 Anomaly Detection Implementation Proposal

Niccolini in [44] described a 2-stage intrusion detection system, where knowledge based system will be applied first and behavior based system will be applied second. They have developed a prototype in Snort. But the idea cannot be fully realized with the current architecture of Snort. To accomplish step-2, SIP messages has to pass through Snort Detection Engine which is actually high speed module come under knowledge based (rule matching) system. The simple functionality & message flow diagram with preprocessor is shown in Figure: 5.7.

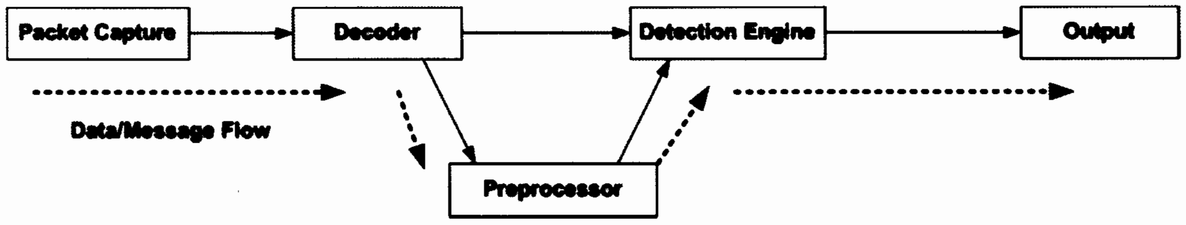


Figure: 5.7 – Message flow in traditional Snort structure

Currently Snort work flow is very simple which is presented in the above diagram. If we apply anomaly detection system in preprocessor (which is in point of fact the only way of adding such functionality right now) then it conflicts with the basic concept of two stage architecture which suggests applying anomaly detection second to rule based system.

Let us suppose that we want to redesign the Snort structure so that it meets architecture's demands. To achieve this there is a possibility to add another module which is actually a postprocessor (similar to preprocessor) which will simply sit in between Detection Engine and Output Module. The schematic diagram is shown in Figure 5.8.

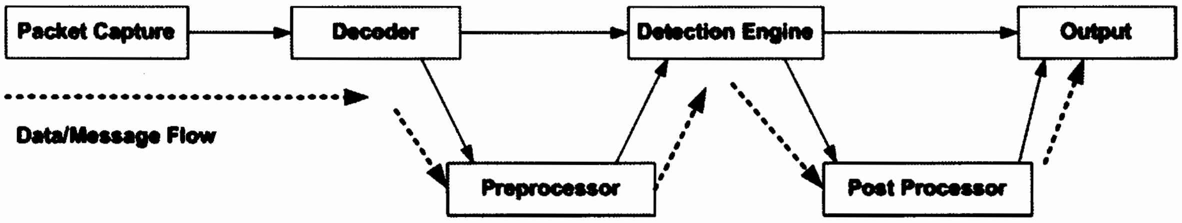


Figure: 5.8 – Message flow in Snort after restructuring

In essence to implement Anomaly detection technique after knowledge based technique, we have to rearrange Snort structure in a way so that:

1. Preprocessor applies first (if configured)and passes data to Detection Engine
2. Detection Engine applies its rule matching procedure
3. If Detection Engine find nothing and if postprocessor is configured, the data will be sent to postprocessor for applying anomaly detection procedure

5.9 Summary

In this chapter we have described what the deficiencies in Snort are and how it can be improved. We also discussed some important features of Snort specifically related with our research. Then we analyze some design requirements including stateful and cross protocol analysis. We also discussed different components that should be analyzed for intrusion and discussed their importance.

As next step, we proposed a distributed Intrusion Detection system based on Snort. The Snort sensors are suggested to be applied on elements of SIP Based VoIP system which are independent in nature and detect intrusions to their own. In addition, they are also responsible to send valuable information to a central Audit server which will further investigate on this information by correlating with each other. In this way some attacks which are hard to detect by single point of impact can be easily brought to surface by looking on analysis of different components. We have described in detail the architecture of each component.

At the end we have described two scenarios of information collection a recommendation of architectural change in Snort structure for implementation of Anomaly Detection methodology in an effective way. Overall the proposal described above can be very effective if applied properly and according to requirements. At least it can serve a starting point for developing IDS architectures on this mode.

Chapter 6
Implementation

6. Implementation

In this chapter, we shall discuss the implementation aspects of prototype development of our project. The architecture discussed in the system design has four components. Development of each one is itself a big task which cannot be possible for a single person to implement in short time period. Therefore we have narrowed down the scope of work to central audit server. We have also defined the data set on which the implementation will be based. We shall see how the correlation method is helpful in detection of attacks which are difficult for conventional techniques. We have discussed three attacks from three categories of attacks to show the effectiveness of proposed system. The references are also given so that the importance of our design can be acknowledged.

Rest of this chapter is organized as follows: Section 6.1 will describe the scope and suppositions that we have made for implementation; Section 6.2 will present the data set used. Section 6.3 discusses the environment in which we shall work. Then we shall discuss different types of attacks and their detection methodology in Section 6.4. In Section 6.5 we have summarize the chapter. Lastly, Section 6.6 presents the Flow Chart and section 6.7 has implementation code.

6.1 Scope and Suppositions

In previous chapter we have discussed the architecture of intrusion detection for SIP based VoIP infrastructure. We proposed that because every component of the network is participating in the communication and possess some sort of information; therefore they can face intrusion incidents. That is why it will be a good idea to place some intrusion detection mechanism on every element. For the sake of simplicity, we have proposed mechanism and architecture for few elements of this network including Proxy Server, User Agents, Accounting Server and Central Audit Server only. Other elements can be included in similar fashion.

For the purpose of prototype development, we limit our scope of work and are only focusing on the central audit server and its functionality. This is because of the novelty of the idea of an offline or delayed analysis of information collected from sensors which already processed to a certain level. Another reason is the time and space constraints due to which all sensors cannot be deployed and tested.

As we have defined our scope to central audit server, therefore we have to suppose some parameters to proceed further. In our scenario, we suppose that sensors have collected all the traffic and analyzed them accordingly. After analysis they have made snapshots of their respective information and send to the central audit server. Now this is our supposition that central audit server possess all the snapshots from the sensors and it is ready to process them for correlation and intrusion detection.

6.2 Data Set

In our scenario there are three servers supposed to send their snapshots to the central audit server. The contents of each snapshot are mentioned below.

a) Proxy Server Snapshot

User Agent Client is the caller and will be referred as UA-A in this snapshot. This snapshot includes the following elements:

Sr#	Element	Description
1	UA-A IP	User Agent A's IP Address
2	UA-A SIP	User Agent A's SIP Address
3	UA-B SIP	User Agent B's SIP Address
4	Call ID	Call ID of Dialog (Unique)
5	Start Time	Dialog Start Time (dd-mm-yy hr:min:sec)
6	End Time	Dialog End Time (dd-mm-yy hr:min:sec)
7	Transactions	Transactions (along with sequence number) separated by CRLF
8	Normal Termination	Dialog Terminated Normally (Yes/No)
9	Error	Error Messages

b) User Agent Snapshot

This will include both User Agent Client and User Agent Server. Both snapshots will have same fields. For the sake of simplicity we are considering only User Agent Client (referred as UA-A in snapshot) because this element will make a call and will be charged for it. This snapshot includes the following elements:

Sr#	Element	Description
1	UA-A IP	User Agent A's IP Address
2	UA-A SIP	User Agent A's SIP Address
3	UA-B SIP	User Agent B's SIP Address
4	Call ID	Call ID of Dialog (Unique)
5	Start Time	Dialog Start Time (dd-mm-yy hr:min:sec)
6	End Time	Dialog End Time (dd-mm-yy hr:min:sec)
7	Transactions	Transactions (along with sequence number) separated by CRLF
8	Normal Termination	Dialog Terminated Normally (Yes/No)
9	Error	Error Messages

c) Accounting Server Snapshot

This snapshot will show the necessary details of calls being made or charged for a particular user. User Agent Client (UA-A in above snapshots) is caller. This includes the following elements:

Sr#	Element	Description
1	User Name	User Name or Phone Number of User (caller) whose account is being Charged
2	Call ID	Unique Identification Number representing particular call
3	Start Time	Dialog Start Time (dd-mm-yy hr:min:sec)
4	End Time	Dialog End Time (dd-mm-yy hr:min:sec)
5	Services	Other Services used (Yes/No)

6.3 Environment

The prototype has been developed under the following environment.

a) Hardware

The personal computer used in this prototype implementation is DELL Optiplex Pentium 4 CPU with speed 2.40 GHz. Its RAM is 1-GB and Hard Disk is 40-GB. Although this system is not very advanced, but this will serve very well for our prototype implementation.

b) Operating System

We have installed Open SUSE 11.3 operating system on this system. Although newer version is also available, but this will serve our purpose. This media contains almost all packages necessary for our project implementation like *pcap* library and other packages related to network. Only thing which are not be available are Snort and its related third party software.

c) Tool

The most important element of our project is Snort Intrusion Detection System. The version which we are using is 2.8.5 which is the latest release at the time of writing of architecture of our proposal. Therefore we shall proceed for the same. This is the tool in which we shall make enhancements by writing its preprocessor which will implement the functionality of Central Audit Server intrusion detection.

d) Language

The native language in which Snort and its components are developed is C. Therefore we shall use the same language for writing our preprocessor. This language is powerful and flexible providing fast program execution and fewer constraints on the programmer.

6.4 Attacks and Demonstration

We implement three attacks to demonstrate the functionality of centralized audit IDS. These attacks can only be detected by using this architecture and no single system can find these intrusions by analyzing the network traffic individually and accurately. Let us discuss these one by one and how they are effectively brought to the surface.

6.4.1 MITM – BYE DELAY Attack

Main-In-The-Middle (MITM) is one of the most difficult categories of attacks for detection. In this category of intrusions the attacker used to sit between the communication channels passively and closely watch and analyze the required traffic. She then become active to some point of time when suitable for. In a paper with title "Billing attacks on SIP based VoIP systems" by Zhang et. al. [23], authors presented their research findings about two VoIP service providers. They have launched four types of attacks by using MITM technique and found that these service providers are vulnerable to these intrusions.

One of those attacks is BYE Delay Billing attacks. The attack detail can be found on that paper. We have selected this attack as an example for demonstration but as a matter of fact all those attacks presented in [23] can be effectively be detected by this architecture.

In this attack scenario, we have two legitimate user agents UA-A and UA-B on two different networks. There are another two user agents acting as main –in-the-middle with the name MITM-1 and MITM-2. The schematic diagram is shown in Figure 6.1. From start of call setup, these MITM's are passively participating into this call intercepting the message flow. When UA-A wants to terminate the call and send BYE message, MITM-1 will intercept it and reply with fake OK message. On the other side, MITM-2 will send BYE message by personating UA-A. The result is that both User Agents are satisfied that the call is terminated, but actually MITM's have taken over the call. Both proxy servers are completely unaware of this disconnection. Now MITM's are free to use the service and the user agents will pay for that.

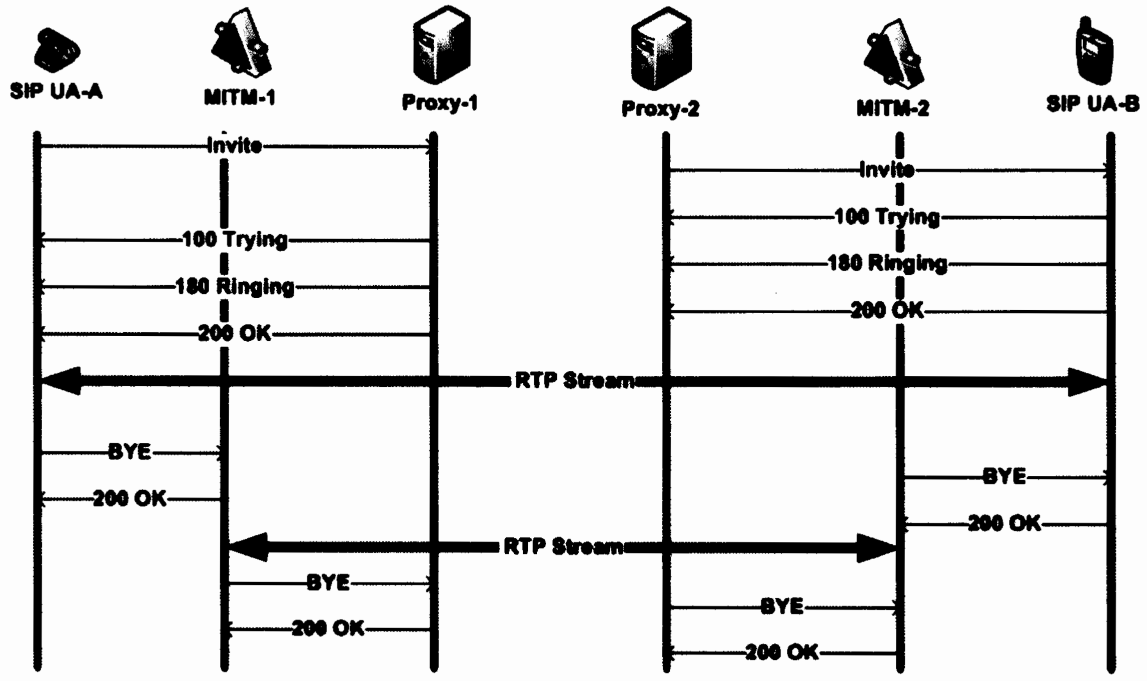


Figure: 6.1 – Schematic of Man-In-The-Middle (MITM) attack

In order to detect this attack, central audit server has the required information received from proxy server, accounting server and user agent. Let us suppose that the audit system is deployed on domain where Proxy-1 and UA-A are located. For the purpose of detection, we have created a rule which will check that a transaction present in the snapshot of accounting server should also be present in snapshot of UA-A. In our case this rule will holds true. It will then check whether the time duration is the same in both snapshots which will return negative results. The classification rule is:

Server Call Duration == Client Call Duration

MITM's have definitely used the connection for their own use and therefore charged for more time duration. This parameter will not match with the time duration of UA-A's snapshot.

6.4.2 Modification Attack – Fake IM

Modification attacks also falls in important categories of attacks in network security. The similar paradigm will follow in the case of SIP-VoIP. Spoofing is the best and well known technique which is used to carry out modification attack. In this type of attack, an attacker impersonate legitimate user by spoofing her identity and send fake messages. The result of such attacks can be diverse in nature.

In our prototype implementation, we have taken an example of *Fake Instant Messaging* Attack. This attack has been discussed by Yu-Sung Wu et. al. [45] in their architecture of intrusion detection for VoIP environments, with the name SCIDIVE. Because they have used host based IDS technique and try to find out this attack by inspecting IP address of the message, therefore their way of inspection is not perfect. They have used the method of verification of sender IP address to detect attack. If attacker also manages to spoof IP address, then this method will fail to detect attack. Moreover, checking IP address of every message in a host based IDS will offer more load on the system. Hence performance of system will decrease to some extent. Therefore, this attack should be detected with more accuracy without degrading the system performance.

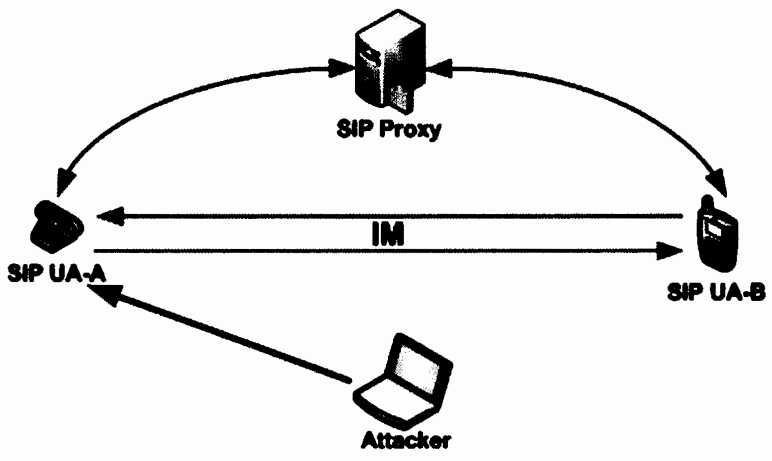


Figure: 6.2 – Schematic of Fake Instant Messaging (IM) Attack

Figure 6.2 is showing the scenario of this attack. There are two user agents UA-A and UA-B communicating with each other through a SIP Proxy. Here we consider that both parties are on single domain. There is another user agent which is acting as an attacker in this scenario. She will send a fake instant message to UA-A, by faking the header. UA-A will believe that this message is from UA-B, though it is actually from attacker.

In [52] it is said that after session establishment client and server will send message directly to each other. But in RFC3428 [51] message flow is suggested to be done through Proxy Server. This means that information in proxy server and UA-A can be useful for us. We shall follow this concept to detect Fake IM attack. If we suppose that this flow has some different route then probably we will have to collect information from some message relay server. In case both users are on same domain it will be good idea to collect snapshot from both and correlate them.

To detect this attack, central audit server has transactions from all parties involved. In our case the stake holders are both user agents and proxy server. Their snapshots possess all transactions which have been seen on these points. These transactions will be compared with each other to know the integrity of messages. In this attack UA-A's snapshot will have IM transaction which UA-B's snapshot and Proxy Server snapshot will not have. It will definitely mean that some third party has sent this message by spoofing the integrity of UA-B. The classification rule is:

Server Transactions == Client Transactions

We have taken the example of single domain. If users are present on different domains, like UA-B is on some other domain, then snapshot of UA-B will not be available. Therefore only proxy server will provide the proof of message integrity.

6.4.3 SPAM Over Internet Telephony

The third example for our prototype implementation, from social threat category, is known as Spam over Internet Telephony (SPIT). We have said enough about this category in previous chapters. Although the issue of spamming is not prevailing as a serious dilemma but it will be so as this technology will be used widely on commercial basis.

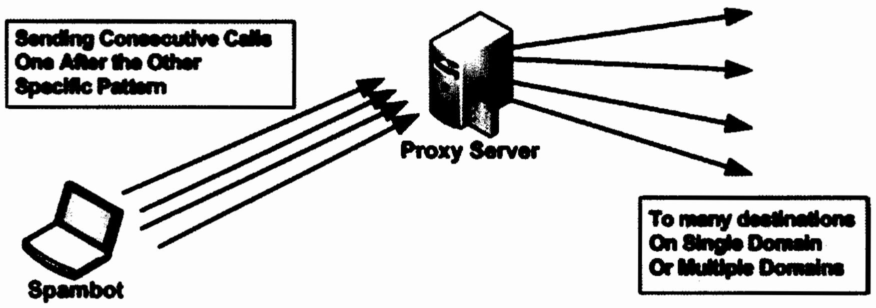


Figure: 6.3 – Schematic of Spam over Internet Telephony (SPIT)

Similar to email spamming, where a service provider does not allow its subscribers to send spams from its network, VoIP service providers will also not allow its subscribers to use the services for such purposes. We need to check whether any user is spamming from a domain to its own or some other domain. Similar paradigm will be used to detect spam from some other domain to a particular domain. The idea is that a user will place many consecutive calls in small period of time having a specific pattern. In most of cases spambots are used to send spams. Therefore a machine or program will be operated with some pattern. Figure 6.3 is showing this scenario.

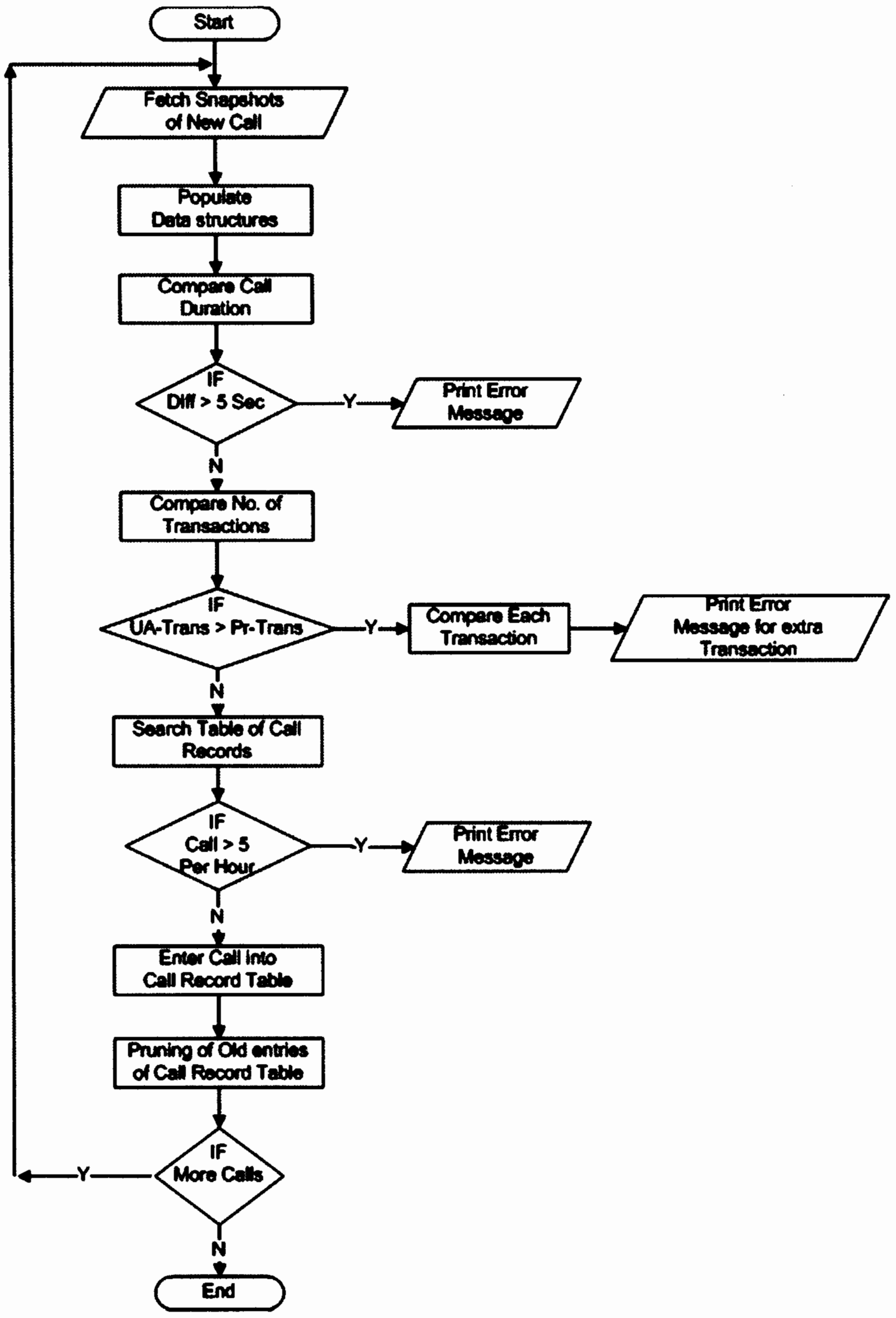
To check this attack, we set a threshold for two factors: (a) if there are more than *x-no* of calls in last 1 hour (b) Calculate gap between each call whether it is the same or have less than 1 minute difference. If both of these conditions holds true, then it is a safe bet to claim such calls as spam. There can also be a third factor which is duration of each which is no more than say 2 minutes. The accuracy of detection can be increased by adding more parameters.

To check these conditions, central audit server can maintain a table which will keep call records for each user depicting user, IP address, time of call, etc. These records will be generated from snapshots received from Proxy server and have validity time period with some expiry date. After receiving snapshot from proxy server, an entry will be made into this table and this check will be conducted.

6.5 Summary

In this chapter we have discussed the implement aspects of this project. First of all we have discussed the scope of implement as developing the full project requires too much time and resource which is not possible. Therefore we have narrowed down the scope to central audit server. We also defined the data set used in our system and the environment in which we are working. Then we discussed three attacks which we have implemented in our prototype to demonstrate the effectiveness of our system. We have also discussed how these attacks were hard to detect for other known system which are either host based of network based. We have used hybrid system to detect them effectively. At last stage, we have shown the Flow Chart which has been followed by Implementation Code.

6.6 Flow Chart


Start
Fetch Snapshots of New Call
Populate Data structures
Compare Call Duration
IF Diff > 5 Sec
Y
Print Error Message
N
Compare No. of Transactions
IF UA-Trans > Pr-Trans
Y
Compare Each Transaction
Print Error Message for extra Transaction
N
Search Table of Call Records
IF Call > 5 Per Hour
Y
Print Error Message
N
Enter Call Into Call Record Table
Pruning of Old entries of Call Record Table
IF More Calls
Y
N
End

6.7 Implementation Code

```
//*************** Header Files ***********
//****************************************

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>


//************** Structures **************
//****************************************

struct SProxyClient   //Snapshot of Proxy & Client
{
      char uaaIP[16];
      char uaaSIP[40];
      char uabSIP[40];
      char CallID[40];
      char CallStart[22];
      unsigned int duration;
      int trans;
      char dialog[150];
};

struct SAccount       //Snapshot of Accounting Server
{
      char uaaSIP[40];
      char CallID[40];
      char CallStart[22];
      unsigned int duration;
};

struct DateTime       //Date & Time elements
{
      //Day-Month-Year & 24 hours time format
      // dd-mm-yyyy 15:40:10
      int dd;
      int mm;
      int yyyy;
      int hr;
      int min;
      int sec;
};

struct CallRecord     //Keep Global Call Record of every session for
certain time
{
      char uaaSIP[40];
      char CallStart[22];
      unsigned int duration;
};
```

```
//****************** Global Variables *************
//***************************************************

#define TimeTh 30;
#define loc 30; //Max. No of Calls found to be involved in SPIT
struct CallRecord cr[100];
int crp=0;
int CallTh=5;



//************ Date Time Conversion *******
//*******************************************

struct DateTime DTConvert(char dtstring[])
{
      int i=0,j;
      char temp[4];
      struct DateTime dt;
      //printf("\nDate Time String is ->%s",dtstring);
      dt.dd='\0'; dt.mm='\0'; dt.yyyy='\0';
      dt.hr='\0'; dt.min='\0'; dt.sec='\0';
//---------------------------------------------
      j=0;
      bzero(temp,4);
      while (dtstring[i]!='-')
            {temp[j]=dtstring[i];
            j++; i++;}
      dt.dd=atoi(temp);
      i++;

      j=0;
      bzero(temp,4);
      while(dtstring[i]!='-')
            {temp[j]=dtstring[i];
            j++; i++;}
      dt.mm=atoi(temp);
      i++;

      j=0;
      bzero(temp,4);
      while(dtstring[i]!=' ')
            {temp[j]=dtstring[i];
            j++; i++;}
      dt.yyyy=atoi(temp);
      i++;

      j=0;
      bzero(temp,4);
      while(dtstring[i]!=':')
            {temp[j]=dtstring[i];
            j++; i++;}
      dt.hr=atoi(temp);
      i++;
      j=0;
      bzero(temp,4);
      while(dtstring[i]!=':')
            {temp[j]=dtstring[i];
            j++; i++;}
      dt.min=atoi(temp);

      return(dt);

}//end of DateTime function



//********** Time Comparison ***************
//******************************************

int TimeCmp(char pTime[], char crTime[])
{

      int pmin=0,crmin=0,temp=0;
      struct DateTime pdt,crdt;

      pdt=DTConvert(pTime);
      crdt=DTConvert(crTime);

      pmin=(pdt.min+(pdt.hr*60));
      crmin=(crdt.min+(crdt.hr*60));

      if(pdt.dd==crdt.dd) //if both calls is in the same day
      {
            if((pmin-crmin)<60)
                  return (1);
      }

      else if(pdt.dd==crdt.dd+1) //if one call is in next day
      {
            temp=1440-crmin;
            temp=pmin+temp;
            if(temp<60)
                  return(1);
      }

      else
            return (0);//if not found


}//end of TimeCmp function
```

```
//************ Check Billing Attack **********
//*********************************************

void billing(struct SProxyClient p, struct SProxyClient c)
{
      int dur;

      dur=p.duration-c.duration;
      if(dur>5)
      {
            printf("-----------------------------------------------
");
            printf("\nBilling Attack Detected on...\n");
            printf("\nIP -> %sUsername -> %sCall ID -> %sRecorded
Time of Call -> %sDuration Billed -> %d Seconds",

      c.uaaIP,c.uaaSIP,c.CallID,c.CallStart,p.duration);
            printf("\n\nCall duration is mismatched. \nDifference is
> %d Seconds\n",dur);
            printf("\nMost probably it is MITM Billing Attack\n\n");
      }

      //Further Inverstigate for this attack which type of attack is
it
      //compare full bye transaction to know this is the same
message

}// end of billing function



//************** Check Integrity Based Attacks ************
//*********************************************************

void integrity(struct SProxyClient p, struct SProxyClient c)
{
      //we need to compare each UA transaction with Proxy
transaction

      char ptrans[50],ctrans[50];
      char *tid,*msg;
      int i=0,j=0,k=0;

      bzero(ptrans,50); bzero(ctrans,50);
      //bzero(tid,10); bzero(msg,10);
      while(c.dialog[i]!='*')
      {
            if(c.dialog[i]!=p.dialog[i])
            {
                  j++;
                  while(c.dialog[j]!='&')
                  {
                        ctrans[k]=c.dialog[j];
                        j++; k++;
```

```
                    }

                printf("-------------------------------------------------
");
                printf("\nFollowing Transaction of Client is not found in
Server\n%s\n",ctrans);

                tid=strtok(ctrans,",");
                  msg=strtok(NULL,",");
                printf("\nTransaction ID -> %s \nMessage Type -%s-
\n",tid,msg);
                if(strcmp(msg,"MESSAGE")==0)
                printf("\nFake IM attack is detected on Address->
%s",c.uaaSIP);
                printf("\nIP -> %sCall ID -> %sCall Date & Time -> %s\n",
                        c.uaaIP,c.CallID,c.CallStart);
                //Checks on other integrity based attacks can be put here
                break;
                }//end if

                if(c.dialog[i]=='&') j=i;
                i++;
            }//end of while


}//end of integrity function



//************ Check SPIT Attack *****************
//************************************************

void SPIT(struct SProxyClient p)
{
        int i,j,level=0;
        //To keep locations of calls record made in last 1 hr by some
client
        int CallLoc[30];

        for(i=0;i<30;i++) CallLoc[i]='\0';

        i=0;
        for(j=0;j<crp;j++)
        {
                if(strcmp(p.uaaSIP,cr[j].uaaSIP)==0)
                {
                        if(TimeCmp(p.CallStart,cr[j].CallStart)==1)
                        {
                                CallLoc[i]=crp;
                                i++;
                        }
                }

        }//end of for loop
```

```
      if(i>CallTh)
           level=level+1;

      if(level==1)
      {
      printf("-------------------------------------------");
      printf("\nSPIT attack is detected from this network... ");
      printf("\nDetection Accuracy=40 Perceeent");
      printf("\nSIP User ID = %sIP Address = %sTime of Call = %sCall
ID = %s\n",
                 p.uaaSIP,p.uaaIP,p.CallStart,p.CallID);
      }


}//end of SPIT function




//************ Preparation of Checking Attacks ***********
//*********************************************************

      int i,j,sLength;
      char tduration[7],ttrans[4];
      char strp[40],strc[40],stra[40],str[30];
      FILE *fp,*cid;
      struct SProxyClient sp,sc;
      struct SAccount sa;
      //printf("\n First Line \n\n");

//set Call Record list to null
for (j=0; j<100; j++)
{
      bzero(cr[j].uaaSIP,40);
      bzero(cr[j].CallStart,22);
      cr[j].duration='\0';
}


      //***** Open file of Call records **********
cid=fopen("calls.txt","r");

while(!feof(cid))
{//error is in below 5 lines.
sLength='\0';
bzero(strp,40); bzero(strc,40); bzero(stra,40); bzero(str,30);
fgets(str,30,cid);
if(str[0]=='*') {break;}
sLength=strlen(str); str[sLength-1]='\0';

strcpy(strp,str); strcat(strp,"/"); strcat(strp,"proxy.txt");
strcpy(strc,str); strcat(strc,"/"); strcat(strc,"useragent.txt");
strcpy(stra,str); strcat(stra,"/"); strcat(stra,"accounting.txt");
```

```
//**** opening proxy file to fetch data ****
      //*********** Initialize structure **********
      bzero(sp.uaaIP,16);
      bzero(sp.uaaSIP,40);
      memset(sp.uabSIP,'\0',40);
      memset(sp.CallID,'\0',40);
      memset(sp.CallStart,'\0',22);
      sp.duration='\0';
      sp.trans=0;
      bzero(sp.dialog,150);
      bzero(tduration,7);
      bzero(ttrans,4);

      fp=fopen(strp, "r");
      if (fp==NULL){perror("Failed to open file ->proxy<-
\n");exit(1);}

      fgets(sp.uaaIP,16,fp);
      fgets(sp.uaaSIP,40,fp);
      fgets(sp.uabSIP,40,fp);
      fgets(sp.CallID,40,fp);
      fgets(sp.CallStart,22,fp);
      fgets(tduration,7,fp);
      fgets(ttrans,4,fp);
      fgets(sp.dialog,150,fp);

      sp.duration=atoi(tduration);
      sp.trans=atoi(ttrans);

      fclose(fp);//closing proxy file



//**** opening useragent file to fetch data ****
      //*********** Initialize structure **********
      bzero(sc.uaaIP,16);
      bzero(sc.uaaSIP,40);
      memset(sc.uabSIP,'\0',40);
      memset(sc.CallID,'\0',40);
      memset(sc.CallStart,'\0',22);
      sc.duration='\0';
      sc.trans=0;
      bzero(sc.dialog,150);
      bzero(tduration,7);
      bzero(ttrans,4);

      fp=fopen(strc, "r");
      if (fp==NULL){perror("Failed to open file ->Useragent<-
\n");exit(1);}
      fgets(sc.uaaIP,16,fp);
      fgets(sc.uaaSIP,40,fp);
      fgets(sc.uabSIP,40,fp);
      fgets(sc.CallID,40,fp);
      fgets(sc.CallStart,22,fp);
      fgets(tduration,7,fp);
      fgets(ttrans,4,fp);
      fgets(sc.dialog,150,fp);
      sc.duration=atoi(tduration);
      sc.trans=atoi(ttrans);

      fclose(fp);//closing proxy file

      //********* Opening Accounting file to fetch data *****
      //************** Initializing structure **************
      bzero(sa.uaaSIP,40);
      bzero(sa.CallID,40);
      bzero(sa.CallStart,22);
      sa.duration='\0';
      bzero(tduration,7);

      fp=fopen(stra, "r");
      if (fp==NULL){perror("Failed to open file ->Accounting<-
\n");exit(1);}
      fgets(sa.uaaSIP,40,fp);
      fgets(sa.CallID,40,fp);
      fgets(sa.CallStart,22,fp);
      fgets(tduration,7,fp);
      sa.duration=atoi(tduration);

      fclose(fp);

      //************ BYE Check *****************
      billing(sp, sc);

      //*********** Fake IM Check **************
      integrity(sp,sc);

      //*********** SPIT Check *************
      SPIT(sp);


//Enter the Call Record into the list
strcpy(cr[crp].uaaSIP,sp.uaaSIP);
strcpy(cr[crp].CallStart,sp.CallStart);
cr[crp].duration=sp.duration;
crp++;


}//end of while
fclose(cid);
```

Chapter 7
Results and Analysis

7. Results and Analysis

In this chapter we shall analyze and assess the techniques proposed in chapter 5. No one technique is curing all and hence multiple methods were proposed. But the question arises to what extent these techniques are effective in detection VoIP attacks and what performance can be achieved by using them. Although it seems that multiple methods impose more loads and required performance may not be on satisfactory level, but it may not be the case. We shall discuss both of these factors in detail in this chapter. As the proposals discussed earlier are previously defined, analyzed and are not newly crafted, we can easily refer analysis done by their authors presented in their paper. It is quite logical that performance will be similar if deployed under the same circumstances.

Rest of this chapter is organized as follows: Section 7.1 will present the effectiveness of our proposed techniques including all proposed components. Section 7.2 presents performance evaluation of the prototype that we have developed. Section 7.3 shows the deployment analysis and finally section 7.5 concludes the chapter.

7.1 Effectiveness

An important question arises while considering an method of intrusion detection system is the effectiveness of that system. A user might want to know what sort of intrusions it will be able to detect. Therefore, we want to discuss the effectiveness of the IDS installed on components of VoIP system. We shall discuss the IDS sensors of Proxy Server and User Agents along with Central Audit Server. Accounting server will only participate ints information for central audit server analysis. Important thing to note is that the system proposed is quite extensible and we can also accommodate other techniques into it very easily.

Usually it happens that when an attacker is crafting an attack, he concentrates only on the results and ignores the correctness of packets. This is a good idea if we check packets for their correctness. Many other checks can be accommodated in the steps to broaden the scope.

7.1.1 Proxy Server

Below mentioned are few benefits of our proposed system of Proxy Server:

- It can detect attacks spanning on the whole state of a communication session. For example if attacker breaks its attack on multiple packets or done it step by step. The method assembles states from multiple packets and use aggregated states for detection engine.
- Syntax analysis is performed and generate alert if found some error
- Mandatory fields in SIP requests are checked for presence, correctness and their size. It also does a cross check Cseq header and SIP method. If something found it raise the flag
- Packets are assembled to make transactions. Correctness and completeness of transactions is also checked on the basis of RFC 3261. If some error is found, alert is generated.
- These transactions are assembled to build up Dialog for stateful analysis. Stateful analysis checks whether the requests being received has some background or not. The examples can be given of DoS attack and Password Guessing attack.
- It also incorporates the basis of Behavior based analysis by keeping traces of active sessions in the form of states. The shared information includes IP addresses, SIP URIs, Ports, Message Rates, etc.
- This method can also detect SPIT attacks to certain level of accuracy by checking INVITE rate of SIP URI and source IP address.
- DoS attacks can be detected by checking SIP message rates of a SIP caller and source IP address

7.1.2 User Agent

SIP User Agent sensor can detect the following attacks:

- This method can detect attacks spanning on multiple protocols like Call Management Protocols (SIP), Media Delivery Protocol (RTP), and even Internet Protocol (IP).

- Packets are assembled to make transactions. Correctness and completeness of transactions is also checked on the basis of RFC 3261. If some error is found, alert is generated.
- These transactions are assembled to build up Dialog for stateful analysis. Stateful analysis checks whether the requests being received has some background or not. The examples can be given of DoS attack attack.
- Media Stream-based attacks are also in scope of this method. RTP based attacks are covered very well even if span on the whole session or are associated with SIP signaling.
- Tracks of RTP stream is also maintained to perform cross protocol analysis. In case of BYE attack, the user agent will see RTP stream coming from its pear even after receiving BYE message.
- Call Tear-Down attacks can be detected by checking CANCEL/BYE whether they are coming from one of the parties involved in the call (IP address). This attack can be done by spoofing only TO, FROM, and Call-ID fields.
- Many other techniques which are based on client side detection can be incorporated into this system very easily.

7.1.3 Central Audit Server

Central audit server will receive information from all sensors and is able to detect these attacks:

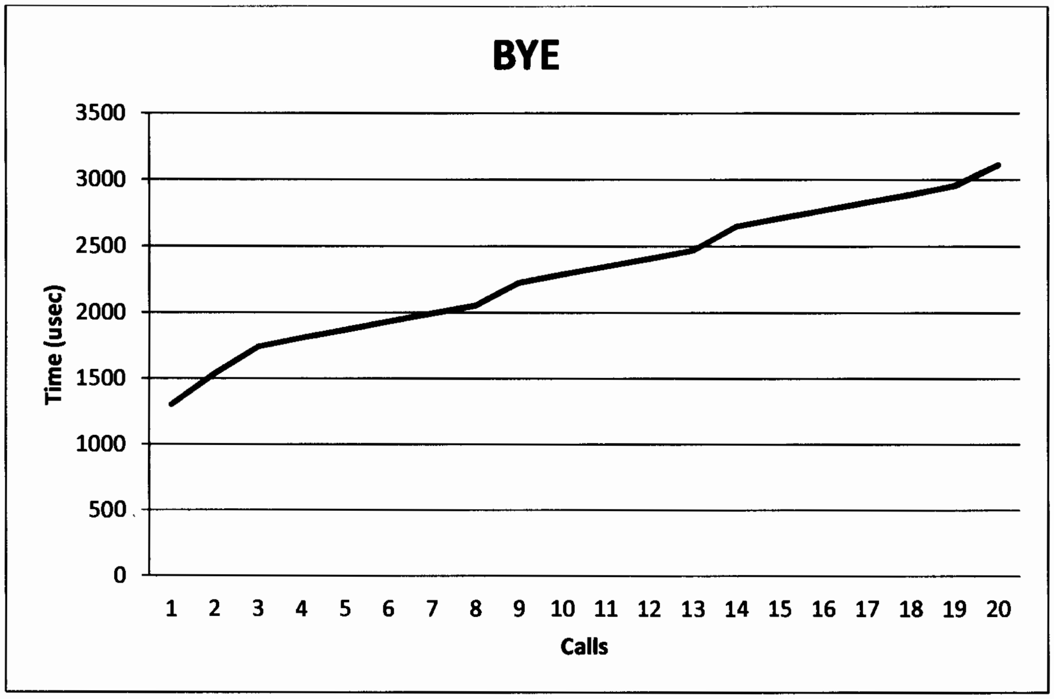
- Masquerading: This type of attacks can be detected after looking into the identity of incoming request and matching it with sender's (who it claimed to be) request. If an attacker masquerade a request, then the actual sender shell not have that request in its log or snapshot sent to central Audit server.
- By the proposed framework, information can be gathered from multiple points and components. Hence more complete, broader and comprehensive analysis can be done. This feature, along with above two, provides the basis of detection many attacks which are very hard to detect otherwise.
- Incomplete dialog break down or Immature call termination can also be inspected by looking at snapshots of all sensors and try to figure out where the problem occurred.

- Billing Fraud where an attacker impersonating a legitimate user and try to use account of other user to make call.
- Other billing attacks which occurred due to Man in the Middle. This is also called Man-in-the-Middle (MITM) attacks category. The examples include (BYE attack, Call Hijacking attacks, Cancel attack, etc.).
- Many Modification attacks after eavesdropping into the network.
- Check integrity of exchanged message for spoofing attacks
- Social Threats can also be detected by this method. We have given an example of Spam call (SPIT) in previous chapter. The technique is equally valid for detection of message spamming and call spamming.
- The central audit server also gathers information about the attacks seen at sensors. Therefore it has the ability to maintain a central knowledge base which is based on experience of all sensors. All sensors can consult this knowledge base to improve their effectiveness for attacks.

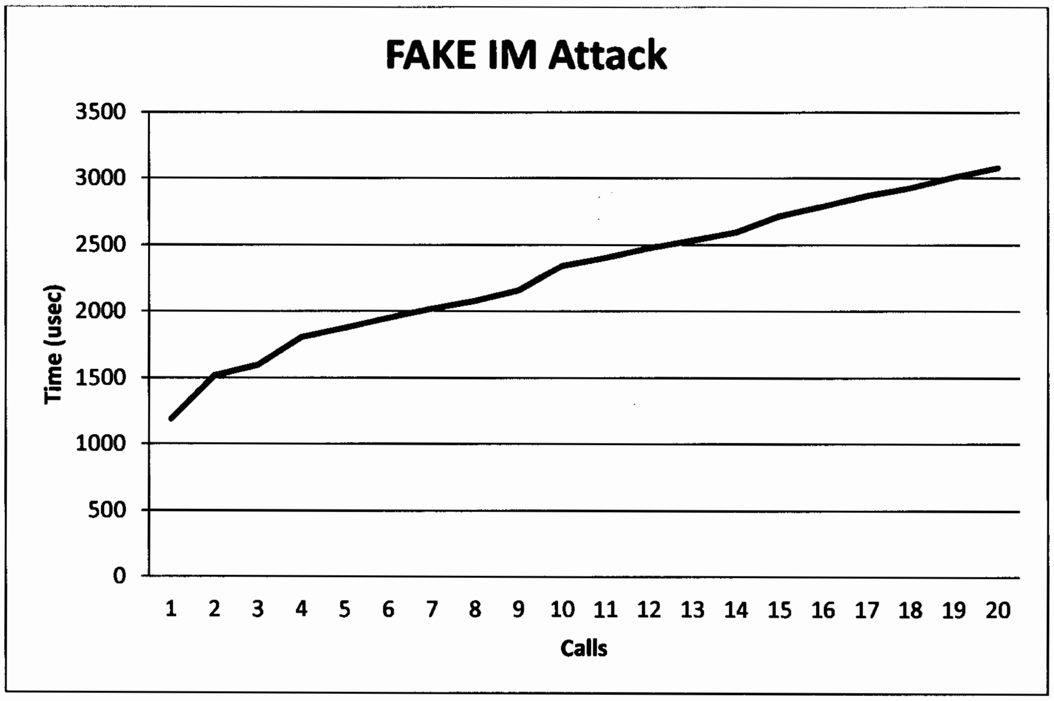
7.2 Performance Evaluation

Although our proposed central audit system is not delay-sensitive but still performance is the point of interest. The prototype that we have developed implements three types of attacks. We have introduced basic checks to avoid complexity. We have supposed that the central audit server has received the snapshots of twenty calls from its sensors. We have introduced the attacks into these snapshots so that testing can be performed. Our results are based on this dataset. We have taken calls on x-axis and time (in micro seconds) on y-axis and plot graphs of BYE, FAKE IM and SPIT attacks. We also have presented comparison of these three attacks.

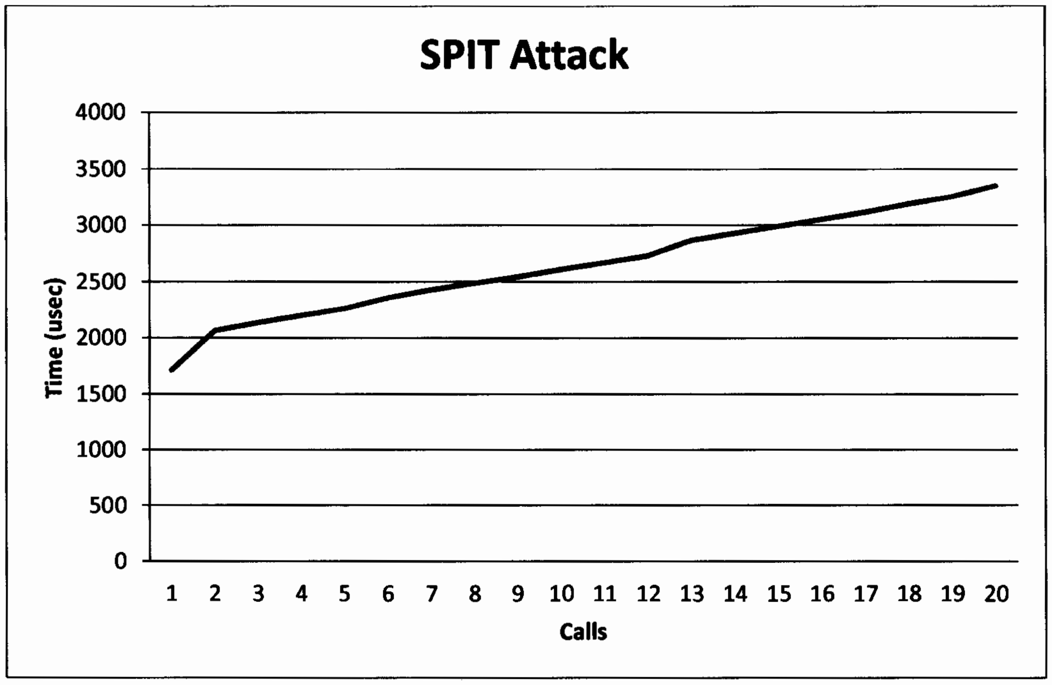
BYE Attack Performance



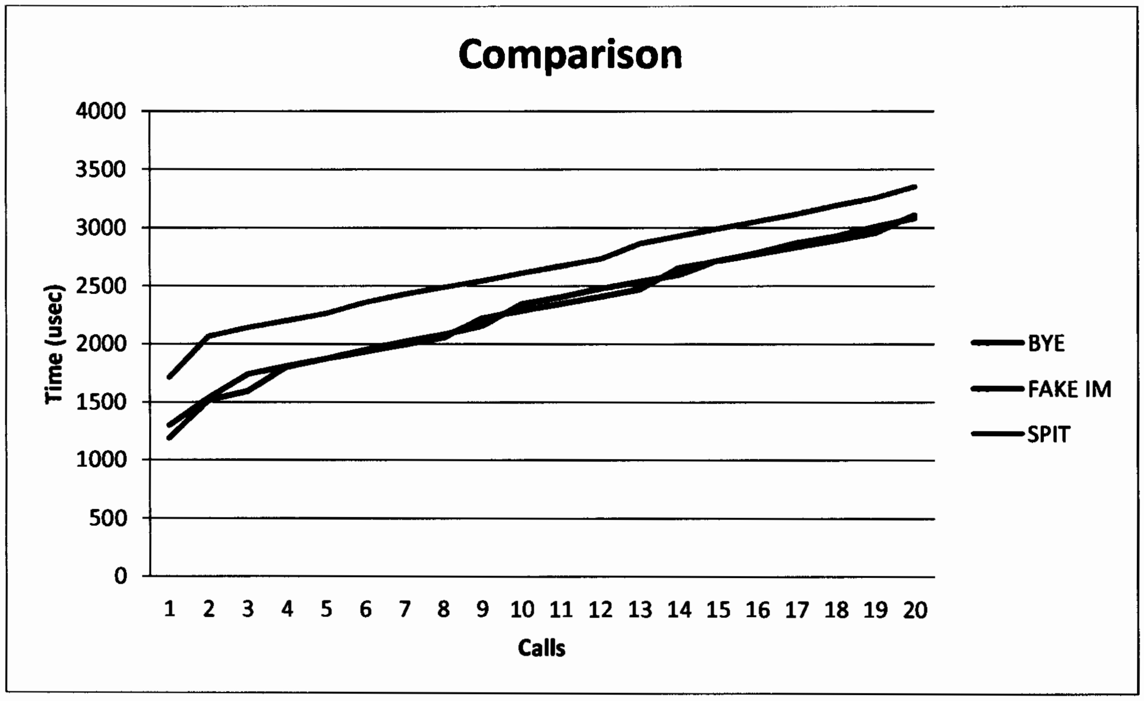
FAKE IM Attack Performance



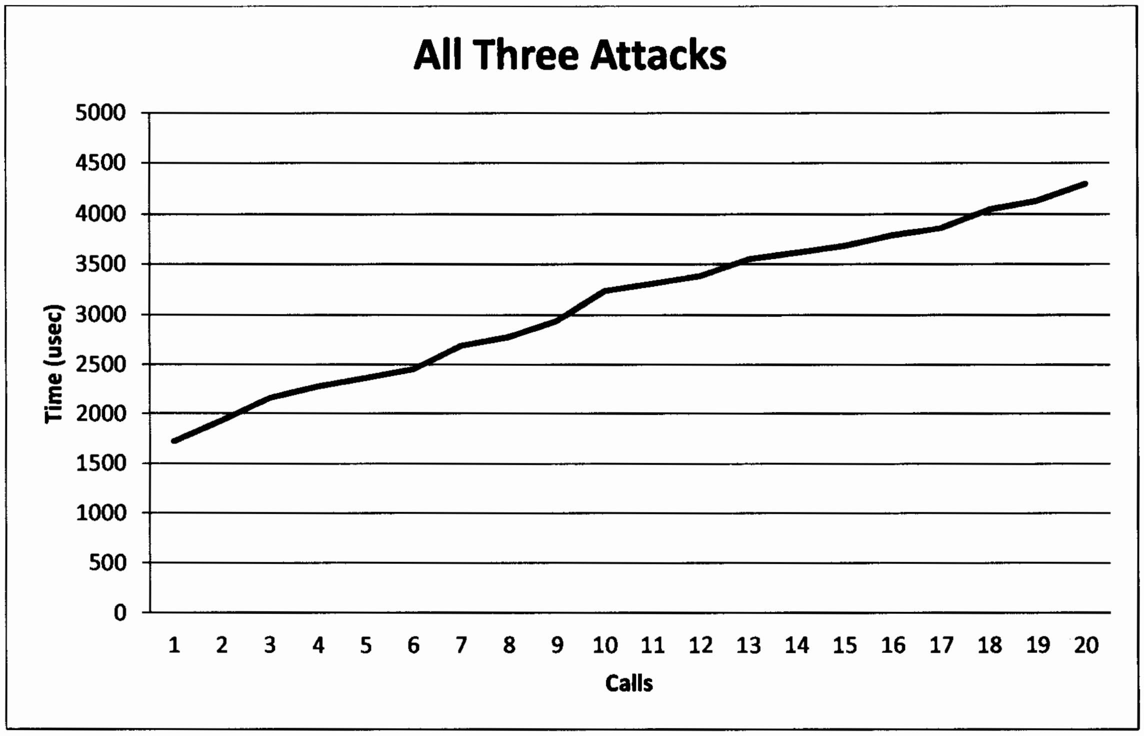
SPIT Attack Performance



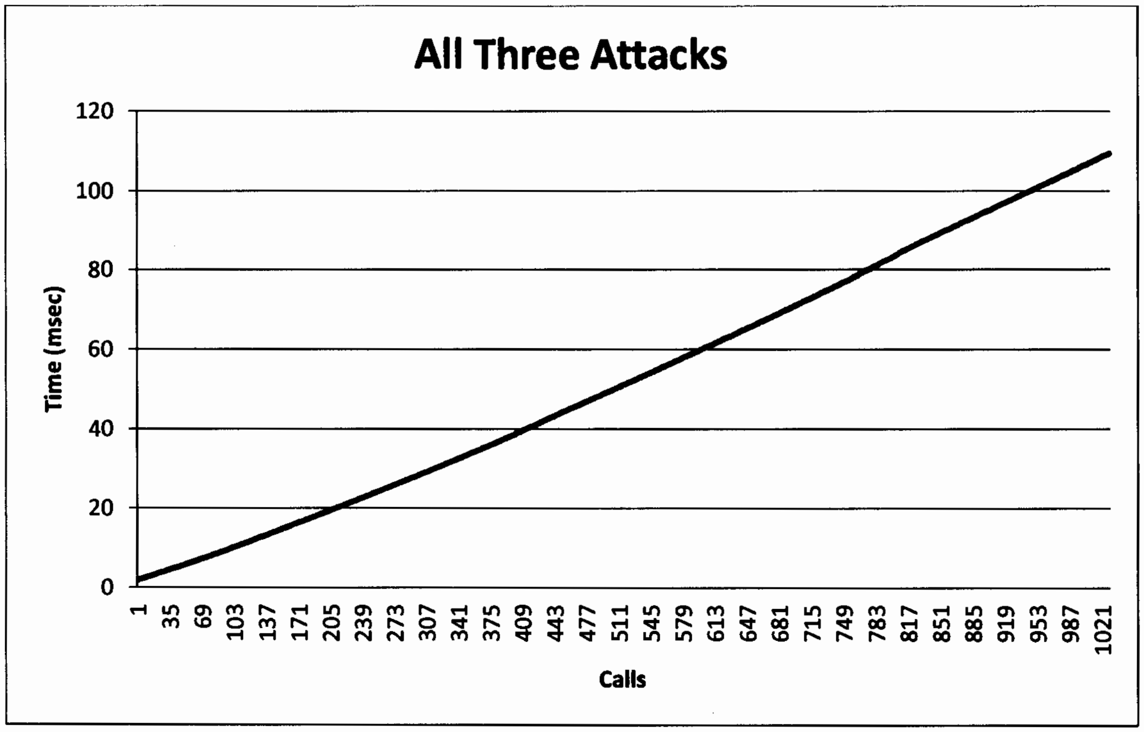
Performance Comparison of Three Attacks



Performance by implementing all three checks in 20 calls



Performance by implementing all three checks in 1000 calls



We can see that the performance time taken by SPIT check is little bit higher than other checks. It is because this check is based on anomaly detection technique and maintains a call record for certain period of time. It uses this record to evaluate the tendency of new call to be marked as spam.

We can see that the above graph of checking all three attacks is almost linear that is a good sign, as far as performance is concerned. Later on, we extended our experiments to 1000 call records and tested the performance. We recorded that total time spent for these 1000 call records is 0.1043 seconds. It means that the system will process 9592 calls in 1 second.

Below is the mathematics of performance measures for these attacks.

- **Bye Attack**

  Detection Delay:

  We suppose that in normal circumstances, server will send snapshot prior than client because of network transmission delay. Therefore, snapshot of server will arrive first to central audit server than snapshot of client. But in case of BYE attack, the situation will be the other way around where client will send snapshot well before the proxy server.

  Let at time Tc, first client snapshot is arrived and at time Ts snapshot of proxy server is arrived. Ts might be after several minutes or ever after hours of Tc. This situation is described in figure 7.1. In case the attack occurs and Tc has been arrived, the system will wait for Ts for m time period. After that it will trigger some other response. Let Tc message is generated at time zero. Nc is transport delay of this message. Ts is arrived after time x. At central Audit Server, the snapshots of client and server are arrived at time T1 and T2 respectively.

  The delay D can be calculated as:

  $D = T2 - T1$

  $T2 = x + Ns$

  $T1 = Tc + Nc$

  $D = ((x+Ns) - (Tc - Nc)$

  $If\ Tc = 0$

$D = x + Ns + Nc$

Where Ns, Nc, & x are random variables.

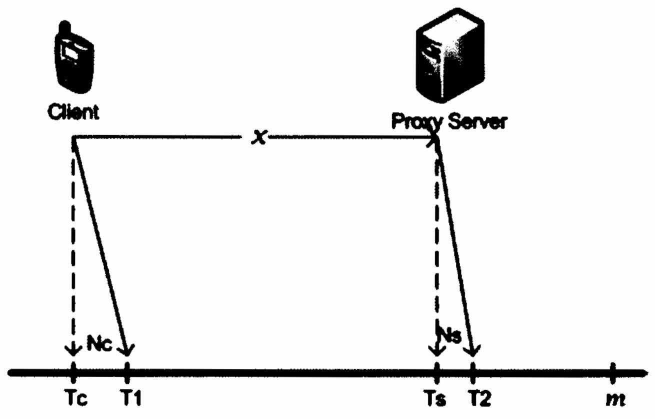


Figure: 7.1 – Delay of BYE attack

Probability of Missed and False Alarm:

There might be a case where attacker arrange to intercept and discard the snapshot message of client.in that case central audit server will receive proxy server snapshot and will wait for *m* time limit. After this time the central server consider it either a network issue at client side or client might have gone down after call termination and is not able to send snapshot. The situation is depicted in figure 7.2.

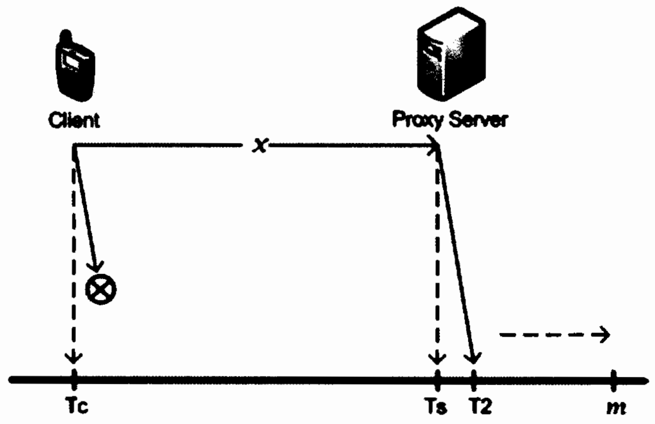


Figure: 7.2 – BYE attack Possibility of Missed Alarm

In such situation the system will miss the attack detection. Here is the probability that the system will not detect this attack:

$Pm = Pr\{ Tc > (T2 + m) \}$

$= Pr\{ Tc > (x + Ns + m) \}$

$= Pr\{ Tc - Ns - x > m \}$

Theoretically, the system cannot generate a False Alarm of BYE Attack.

- **Fake IM Attack**

Detection Delay:

The attack detection depends upon the arrival of snapshots of Proxy Server & Client. We suppose that proxy server snapshot come earlier than client. i.e.

Tc > Ts

Moreover the attack is equally distributed over the whole call and can occure at any time. Thereore Fake IM attack is observed between INVITE & BYE message with distribution given by G (Gaussian Distribution). The situation is depicted in figure 7.3.

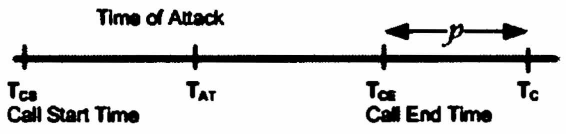


Figure: 7.3 – Delay of Fake IM attack

Therefore the time delay occurred for detection is given as:

Let $T_C > T_S$

$D = T_C + N_C$

$T_C = T_{CE} - T_{AT} + \mathcal{p}$

$D = T_{CE} - T_{AT} + \mathcal{p} + N_C$

Where call is started at point Tcs & terminated at time T_{CE}. Attack is launched at time T_{AT}. System needs some tome to make a snapshot i.e. $\mathcal{p}$.

Probability of Missed and False Alarm:

If the snapshot of client does not reach central IDS server, then the system will not be able to detect the attack. The situation will be like previously discussed BYE attack. Therefore the probability is given by:

$Pm = = Pr\{ Tc - Ns - x > m \}$

7.3 Comparison with other technique

Any technique or architecture is evaluated by comparing it with previously developed methods and techniques. We have discussed many techniques in chapter-3 each having its own procedures for detection of attacks. But we have selected method [44] for our performance comparison. Although there is no apple to apple comparison of both techniques as both have different moods of operation, but we can compare some features with each other.

In [44], the authors have implemented their system in Snort which is in the form of preprocessor. They have used HP Compaq 9005 notebook for their system implementation which is equally as old as we have having similar specs. The important feature of their system is that they have place the IPS as Inline machine and watching the live traffic. Therefore they have to process the data efficiently enough so that the congestion can be avoided. The processing is message by message. In their paper, they have calculated the mean end to end delay which is shown in figure 7.4. It shows that their system can process up to 850 messages per second without any issue. But the data rate exceeds 860 messages per second, their IPS system crashes. It is because the message queue of iptables becomes full and the system rejects the remaining messages coming from sender. The important thing to note is that they have implemented the knowledge based technique into IPS system which is usually light weight as compared to anomaly based techniques.

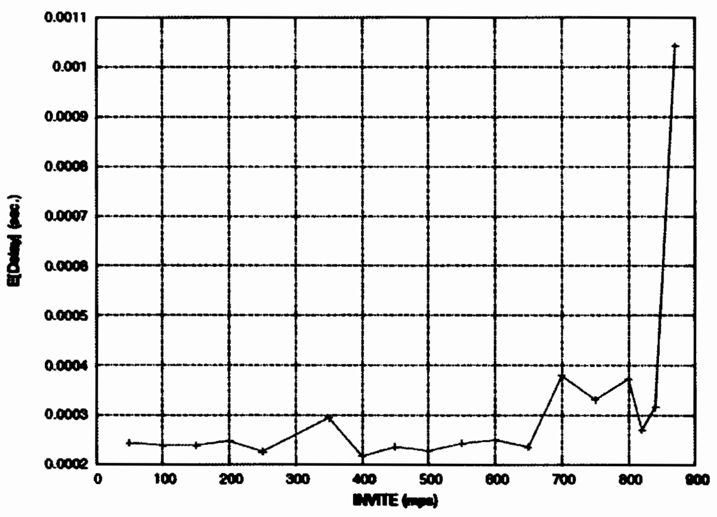


Figure: 7.3 – Delay of Fake IM attack

As we have discussed that our hardware is similar as that of used in [44], but we have used VMware Player to rum Linux system on Windows XP. In this way the resources have been shared among the processes. Moreover, we have not stripped of the OS and there are many programs installed with Linux OS which poses more load on the system. If we stripped off unwanted programs from our OS and install it as a sole OS on computer, then the performance of our system will be considerably be improved.

Our system is based on information correlation of dialogs arrived with snapshots of different SIP elements, instead of each message individually. Moreover our system work in offline mode and installed on a separate machine not involved inside any communication. This means that the traffic or message drop will not be possible. We have already discussed in section 7.3 that in our current implementation, 0.1043 second is required to process 1000 call records. Moreover if we maintain the list of calls to be processed in to the memory, it will also not eat up the memory a lot. In our implementation, we have used 20 call records and the total size of this list is 300 Bytes. Therefore, if we have 1 million call records pending in the list, the memory consumption will be just 14 MB. Another aspect of our system is that we have applied two knowledge based checks as well as an anomaly based check. Usually anomaly detection technique is heavier in processing than knowledge based technique.

In [44] the issue of false results has not been discussed and therefore the parameter of accuracy cannot be compared with each other. As far as our technique is concerned, we already have given the probability of False Positive and False Negative results of our system.

7.4 Deployment Analysis

In broader perspective, we have proposed two types of techniques to implement in previous chapter; Anomaly Based and Knowledge Based. These techniques are further divided into multiple parts. Therefore it is a great concern of security administrators that deployment of all these techniques along with traditional SNORT Intrusion Detection System will greatly overload the system and consume much more resources which might go beyond economical and technical feasibility. These concerns are very much true as both of these factors are important decision parameters when making a verdict how to secure any infrastructure and resources. An important question arises here that why we need so many techniques to be deployed. The answer is simple; there is no single technique that protect VoIP infrastructure comprehensively. Therefore if we are able to justify the deployment of all proposed techniques, both economical and technical point of view, then the proposed solution can be very helpful for VoIP service providers.

As far as the second technique is concerned, VoIP Network Components Audit, this is purely distributed in nature. There is one central audit server which can be a separate machine (preferably) or the machine on which two stage anomaly detection system is installed. The sensors used for collection of valuable data from different VoIP components and sometimes do partial analysis, can be installed on the elements. For example, sensor can be installed on soft or hard client (e.g. Skype) that can also be upgraded online just like antivirus updates itself. In this case there is no need to have a separate machine for running IDS instance; instead Snort instance can be part of the sensor. We already know from previous chapters that Snort is light weight IDS and we can easily reduce its size according to the requirements *(e.g. there is no need to check HTTP or DNS issues)*. Therefore such system will not overload the client machine at all. Similar is the case with other sensor placement like on proxy server, billing and accounting server, etc.

For technical point of view, we can easily see that the load is distributed over components and efficiency is not much affected under heavy loads. It is a natural phenomenon that more systems added in architecture; more the performance will be degraded. But that performance should be acceptable under circumstances in which the system is going to be deployed. Therefore the methods proposed and their way to implement is realistic in nature and can be deployed easily.

7.5 Summary

In this chapter we have seen that our proposed technique constitutes hybrid methodologies and distributed in nature. Despite of this fact that system has to gather information from different components the performance issues are not prominent. This is because of the reason that the work load is distributed over the network and central audit server does not need to check for attack immediately. We have also analyzed the deployment scenario which shows the feasibility of implementation in terms of finance and other factors.

Chapter 8
Conclusion and Future Work

8. Conclusion and Future Work

8.1 Conclusion

Security issues in VoIP are relatively different as compared with traditional IP related technologies. Because of diverse and distributed nature of VoIP infrastructure attack scenarios are relatively diverse and more management and efforts are required to cope with these issues. In this thesis we studied SIP - VoIP technology along with a broad spectrum of threats associated and discussed methodology to counter these attacks.

In first chapter of this document, we introduced VoIP technology and presented a brief overview. We discussed in detail about SIP and its role in VoIP along with SIP network elements and session anatomy. Review of other protocols involved in VoIP like RTP, SDP, RTCP has also been presented. Later in this chapter, intrusion detection systems and its basic concepts were discussed which is required to clear essential concepts. Although VoIP and SIP are not very familiar in commercial technology but their future is very bright in communication industry. To work on these technologies, we need a clear understanding about their components and working.

Chapter two presented an overview of Snort as Intrusion Detection tool as this is our core subject and tool for enhancement. We discussed elements of Snort along with their functionalities briefly. We also studied how capabilities of Snort can be enhanced to make it suitable for the use of SIP communication. Despite of the fact that Snort is a popular tool for intrusion detection, it lacks the capability to detect SIP attacks. Simple rules cannot provide solution for this remedy. But it has great room for enhancements for booth anomaly based techniques and knowledge based techniques.

Then we discussed categories of IDS as our whole future discussion was revolving around these concepts. Later in chapter three we have tried to give brief description of techniques already developed for VoIP intrusion detection and other forms of attacks. This discussion also includes merits and demerits of each technique. The endeavor to secure VoIP infrastructure has been started for a long time since different threats came on the surface. As a result many techniques have been proposed by research and industry. But each one has pros and cons. In most of cases, intrusion detection industry tries to combine more than one method to cope with situation.

Security issues are one of major problems facing in today's communication technology. But after a comprehensive literature survey in chapter four, we have come to know that security issues related to VoIP are different as compared to traditional technologies. This is because of its distributed nature, involvement of more than one management domains, many protocols involvement to get the work done and immature technology. The problem becomes more prominent when we talk about SIP-RTP communication. SIP is based on HTTP and SMTP paradigm which are already vulnerable to many attacks. After discussing attacks specifically related to SIP we can conclude that ordinary measures are not enough and these attacks need special attention. Moreover, single technique cannot protect every element or most of elements in VoIP infrastructure. Therefore some dynamic mechanism is required for securing SIP - VoIP service by using Snort.

By looking this entire situation, we have proposed an intrusion detection infrastructure which is distributed and heterogeneous in nature so that key components of SIP architecture have intrusion detection elements. These elements are independent in working and perform intrusions detection without knowing what is going on elsewhere. For the sake of simplicity, we have proposed architecture for only two sensors which are the most important in all architecture: Proxy Server and User Agent. It obvious that functionality of these elements is different, therefore intrusions and attacks on both will also be different. The result is that different methods are required to detect attacks. After comprehensive studies of different techniques already developed, we have found many methods which we can exploit. Based of those points, we have devised another hybrid procedure for both of these elements. Important aspect of these procedures is that they consider packets individually, along with transactions and the whole dialog as well. By this way state of SIP session is easy to maintain which is helpful for detection of stateful attacks.

Another important concept associated with our proposal is centralized inspection of information collected from all SIP elements on which sensors are installed. We call it as centralized audit server. This server performs correlation of information collected from its sensors and found out issues difficult otherwise. This concept is unfamiliar and has not been used in SIP and VoIP technology so far. Along with effective intrusion detection of some sophisticated attacks, there are many other advantages associated with this method. First and foremost advantage is that performance of this server will not affect VoIP infrastructure. It is neither installed on VoIP elements nor in network path. Instead it is a separate machine and all sensors will send their information to it by just an ordinary way of communication. So even if this server fails to work for any reason, maximum loss will be inability to detect certain type of attacks.

We implemented central audit server for the sake of proof of concept and used sample data which is similar as that of collected from sensors. This is done because of time and resources constraints and to make things simple. In the sample data, three attacks were introduced: BYE attack, Fake IM and SPAM Calls. These attacks belong to three different categories of attacks and are difficult to detect on sensors. Even if they are detected, their accuracy is poor. But by correlating of information collected from different sensors, these are detected with much more precision.

By analyzing the techniques proposed, we come to know that the system has ability to detect a great number of attacks specifically related with SIP. By enhancements into our method, we can potentially counter most of attacks discussed in chapter 4.

Lastly we discussed how to extend our work what can be done future. We have provided the basis of architecture of intrusion detection system for SIP – VoIP infrastructure. But it can be implemented in a correlated way on a test bed. In this way their performance can be measured properly. Moreover, attacks can be simulated against this testing scenario so that real-time effectiveness can be evaluated.

8.2 Future Work

In our research work and the dissertation resulted at the end, we have discussed intrusion detection and different concepts associated with it in a bit detail. We have also discussed Snort, its components, its strengths and weaknesses. We have described a distributed intrusion detection technique which also incorporates centralized audit system. At the end we narrowed down the scope and have tried to implement our method of central audit system under some assumptions.

We have left many things for future work. One of the most important one is to build up a test bed and implement all sensors on different machines. All sensors performs different functions and worth implementation. In this way we shall come to know whether these sensors can detect attack which they are supposed to be. We shall also come to know how these components communicate and collaborate with each other. We can also do many experiments related to our project if we have a test bed deployed. These may include real-time attack simulation on test bed.

Secondly if we implement the framework on test bed, it will be a very good idea to take real world traffic for analysis and checking for intrusions. As we have already said that we can simulate many attacks. Real-world data may include data collected from some VoIP service provider or generated on test bed by making sample calls. But important thing is to implement our proposed methodology onto data which is near to actual SIP based VoIP communication.

Performance is also an important aspect for any IDS and plays an important role in measuring effectiveness and making comparisons. In our work, we have only implemented central audit server where it works on data already collected from its sensors. Therefore performance of this server can only be calculated independently. After implementation on the test bed, performance can be measured of each sensor which is implemented as an additional service running with their primary tasks. Therefore, performance degradation will definitely occur on those sensors. Moreover, when these sensors will communicate with central server, performance issues will also arise there. Both of these aspects are important and need an investigation.

Here are some other points on which future work can be based on:

- More advanced knowledge based and behavior based techniques and detection algorithms of IDS could be implemented
- Developing VoIP alert correlation engine able to detect ongoing attacks
- Design an appropriate protocol for snapshot sharing which is reliable and does not overwhelm the system and network
- Evaluate effectiveness and accuracy through simulated attacks

In last chapter we have discussed a two stage method where knowledge based technique is implemented first and behavior based is implemented second. We have discussed that if we want to accommodate this concept into Snort, then we have to change the basic architecture of Snort. We have proposed to add postprocessor which is similar to preprocessor analogy. Architectural change can impose a big difference on the performance and effectiveness of any system. This proposal can be implemented and its effects can be evaluated.

Bibliography

[1] David Butcher, Xiangyang Li, Jinhua Guo, "Security Challenges and Defense in VoIP Infrastructure", *IEEE Transactions on Systems, Man, and Cybernetics — Part C: Applications And Reviews.* vol. 37, No. 6, pp 1152-1162, November 2007.

[2] (2011, May 14) *Voice over Internet Protocol*, Wikipedia, the free encyclopedia, [Online]. Available: http://en.wikipedia.org/wiki/VoIP

[3] Juniper Networks, Inc., White Paper – Voice over IP 101, "Understanding the basic networking functions, components and signaling protocols in VoIP network", (2007, May 21). Available: http://www.juniper.net/solutions/literature/white_papers/200087.pdf

[4] Bill Hayes. (2003, May 26) "*Conducting a Security Audit: An Introductory Overview*", [Online]. Available: http://www.symantec.com/connect/articles/conducting-security-audit-introductory-overview

[5] D. Richard Kuhn, Thomas J. Walsh, Steffen Fries. (January 2005) "*Security Considerations for Voice over IP Systems. Recommendations of the National Institute of Standards and Technology*". Available: http://csrc.nist.gov/publications/nistpubs/800-58/SP800-58-final.pdf

[6] White Paper – "Protocols involved in transporting voice over an IP based network". (2007, October 12). Available: http://www.voip-calculator.com/protocols.html

[7] White Paper – "Understanding SIP – Today's hottest communication protocol comes of age". Ubiquity. (2010, May 12). Available: http://www.sipcenter.com/sip.nsf/html/WEBB5YNVK8/$FILE/Ubiquity_SIP_Overview.pdf

[8] (2011, Jan 07). SIP Signaling. SIP Center. [Online]. Available: http://www.sipcenter.com/sip.nsf/html /SIP+Signaling

[9] H. Schulzrinne - Columbia University, G. Camarillo - Ericsson, A. Johnston - WorldCom, J. Peterson - Neustar, R. Sparks - dynamicsoft, M. Handley - ICIR, E. Schooler - AT&T. *"SIP : Session Initiation Protocol* – RFC 3261". June 2002.

[10] (2010, May 23) What is SIP – Introduction? SIP Center. [Online]. Available: http://www.sipcenter.com/sip.nsf/html/What+Is+SIP+Introduction

[11] M. Handley, V. Jacobson. *"SDP: Session Description Protocol – RFC2327"*. April 1998.

[12] M. Handley – UCL, V. Jacobson - Packet Design, C. Perkins - University of Glasgow. "*SDP: Session Description Protocol* – RFC4566". July 2006.

[13] (2010, May 23) "*SDP: Session Description Protocol*" – From Javvin. [Online]. Available: http://www.javvin.com /protocolSDP.html

[14] (2010, May 23) "*Session Description Protocol*" – From Wikipedia. [Online]. Available: http://www.en.wikipedia.org /wiki/Session_Description_Protocol

[15] (2010, May 23) "*Real-time Transport Protocol (RTP)*" – From Javvin. [Online]. Available: http://www.javvin.com/protocolRTP.html

[16] (2010, May 23) Wikipedia – "*Real-time Transport Protocol*". [Online]. Available: http://en.wikipedia.org/wiki/Real_time_Transport_protocol

[17] Ville Hallivuori. (2010, May 23) "Real-time Transport Protocol (RTP) security". [Online] Helsinki University of Technology. Year 2000. Available: http://www.securityvibes.com/docs/DOC-1274

[18] M. Handley – ACIRI, C. Perkins - USC/ISI, E. Whelan - UCL. "*SAP: Session Announcement Protocol* – RFC2974". October 2000.

[19] H. Schulzrinne - Columbia University, S. Casner - Packet Design, R. Frederick - Blue Coat Systems Inc., V. Jacobson - Packet Design. "*RTCP: Real-time Transport Control Protocol*- RFC3550", July 2003.

[20] (2010, January 12) "*H.323 Protocols Suite*". [Online]. Available: http://www.protocols.com/pbook/h323.htm

[21] Dimitris Geneiatakis, Tasos Dagiuklas, Georgios Kambourakis, Costas Lambrinoudakis, And Stefanos Gritzalis, University Of The Aegean, Karlovassi Sven Ehlert And Dorgham Sisalem, Fraunhofer Fokus Institute. "Survey of Security Vulnerabilities in Session Initiation Protocol". IEEE Communications Surveys & Tutorials. The Electronic Magazine of Original Peer-Reviewing Survey Articles. Vol. 8, No. 3. pp 68-81. 3rd Quarter 2006.

[22] Dimitris Geneiatakis, Georgios Kambourakis, Tasos Dagiuklas, Costas Lambrinoudakis and Stefanos Gritzalis. "SIP Security Mechanisms: A state-of-the-art review", in *Proc. 5th International Network Conference*. Vol. 20, pp 147-155. Year 2005.

[23] Ruishan Zhang, Xinyuan Wang, Xiaohui Yang, Xuxian Jiang. "Billing Attacks on SIP-Based VoIP Systems", in *Proceedings of the first USENIX workshop on Offensive Technologies.* Article No. 4. Year 2007.

[24] Florent Carli. SANS GSEC Practical Assignment. "Security Issues with DNS". SANS Institute Infosec Reading Room. Version 1.4b. Year 2003.

[25] Ville Hallivuori. (2010, May 23) "Real-time Transport Protocol (RTP) security". [Online] Helsinki University of Technology. Year 2000. Available: http://www.securityvibes.com/docs/DOC-1274

[26] David Endler, Dipak Ghosal, Reza Jafari, Akbal Karlcut, Marc Kolenko, Nhut Nguyen, Wil Walkoe, Jonathan Zar. "VOIPSA VoIP Security and Privacy Threat Taxonomy". Public Release 1.0. October 24, 2005.

[27] S. Niccolini NEC. “VoIP Intrusion Detection and Prevention: Recommendations & Prototype” A Presentation – NEC Europe Ltd., Network Laboratories. August 29, 2006

[28] Mohamed El Baker Nassar, Radu State, Olivier Festor. INRIA Lorraine – LORIA. “Intrusion Detection Mechanisms for VoIP Applications”, in *Dans Third annual VoIP security workshop (VSW'06)*. 18 Oct 2006.

[29] Dorgham Sisalem and Jiri Kuthan, Tekelec Sven Ehlert, Fraunhofer Fokus. “Denial of Services Attacks Targeting a SIP VoIP Infrastructure: Attack Scenarios and Prevention Mechanisms”, in *IEEE Network. International Conference on Principles, Systems and Applications of IP telecommunications.* Vol 20, No. 5, pp 26-31. October 2006.

[30] (2010, April 26) U.S. National Information Assurance Glossary. Committee on National Security Systems. [Online]. Available: www.cnss.gov/Assets/pdf/cnssi_4009.pdf

[31] Mohamed Nassar, “VoIP Networks Monitoring and Intrusion Detection”. Ph.D. Dissertation, Dept. Computer Science, The Henri Poincare University – Nancy, France. March 2009.

[32] J. Franks - Northwestern University, P. Hallam-Baker - Verisign, Inc., J. Hostetler - AbiSource, Inc., S. Lawrence - Agranat Systems, Inc., P. Leach - Microsoft Corporation, A. Luotonen - Netscape Communications Corporation, L. Stewart - Open Market, Inc. “HTTP Authentication: Basic and Digest Access Authentication”. Internet Engineering Task Force (IETF). RFC 2617. June 1999.

[33] Tasos Dagiuklas, Dimitris Geneiatakis, George Kambourakis - (University of Aegean), Dorgham Sisalem, Sven Ehlert, Jens Fiedler - (Fraunhofer Fokus), Jiří Markl, Michal Rokos - (Nextsoft), Olivier Botron (Telip), Jesus Rodriguez (VozTelecom), Juntong Liu (Embiron). “SNOCER - Low Cost Tools for Secure and Highly Available VoIP Communication Services”. Year 2006.

[34] Mark Collier. SecureLogix Corporation. “Basic Vulnerability Issues for SIP Security”. Secure Logix Corporation. March 2005.

[35] Y. Rebahi, D. Sislem. "SIP Services Providers and Spam Problem", in *2nd Workshop on Securing Voice over IP*. June 2005.

[36] Shawn McGann and Douglas C. Sicker. "An Analysis of Security Threats and Tools in SIP-Based VoIP Systems". MS Thesis, Dept. Interdisciplinary Telecommunications, University of Colorado, Boulder, U.S. June 2005.

[37] Forwarded by Stephen Northcutt (Director of Training & Certification, The SANS Institute). "Snort 2.1 – Intrusion Detection". 2nd Edition. Syngress Publishing Inc, Rockland, MA, 2004. ISBN: 1-931836-04-3

[38] (2010, January 12) *Intrusion Detection.* IT Security. [Online]. Available: http://www.itsecurity.com/features/intrusion-detection-030807

[39] Peng Ning (North Carolina State University), Sushil Jajodia (George Mason University). "Intrusion Detection Techniques". December 2003.

[40] Theodoros Lappas and Konstantinos Pelechrinis. Department of Computer Science and Engineering, UC Riverside. "*Data Mining Techniques for (Network) Intrusion Detection Systems*". Year 2007.

[41] Fengmin Gong, Katerina Goseva-Postojanova, Feiyi Wang, Rong Wang, Kalyanaraman Vaidyanathan, Kishor Trivedi, Balamurugan Muthusamy. "Characterizing Intrusion Tolerant Systems Using A State Transition Model". Issue Date 2001, Vol. 2, pp 211-221. August 07, 2002.

[42] Zhou Lianying and Liu Fengyu. "A Swarm-Intelligence-Based Intrusion Detection Technique". *IJCSNS International Journal of Computer Science and Network Security*, VOL.6 No.7B. July 2006.

[43] Stefano Zanero. "360 Anomaly Based Unsupervised Intrusion Detection". Politecnico di Milano, Dept. Elettronica e Informazione, Milano, Italy. Black Hat Briefings – Washington DC. March 2007.

[44] Niccolini, S.; Garroppo, R.G.; Giordano, S.; Risi, G.; Ventura, S.; NEC Europe Ltd., Heidelberg, Germany. "Sip Intrusion Detection And Prevention: Recommendations And Prototype Implementation", in *1st IEEE Workshop on VoIP Management and Security*, pp 47-52, 2006.

[45] Yu-Sung Wu, Saurabh Bagchi (School of Electrical & Computer Eng., Purdue University), Sachin Garg, Navjot Singh, Tim Tsai (Avaya Labs). "SCIDIVE: A Stateful and Cross Protocol Intrusion Detection Architecture for Voise-over-IP Environments", in *International Conference on Dependable Systems and Networks,* pp 433 – 442. 2004

[46] Dorothy E. Denning. "An Intrusion Detection Model. SRI International", in *IEEE Transactions on Software Engineering - Special issue on computer security and privacy.* Vol. 30, No. 2, pp 222-232, Year 1987.

[47] Evgeniya Petrova Nikolova, Veselina Gospodinova Jecheva (Burgas Free University). "Anomaly Based Intrusion Detection Based on the Junction Tree Algorithm", in *Journal of Information Assurnce and Security.* Vol. 2, No. 3, pp. 184-188, Year 2007

[48] Dorothy E. Denning (SRI International). "An Intrusion Detection Model", in *IEEE Symposium on Security and Privacy*, pp 118, Year 1986.

[49] Yacine Bouzida, Christophe Mangin. "A framework for detecting anomalies in VoIP networks", in *proceedings IEEE Computer Society the Third International Conference on Availability, Reliability and Security*, pp 204-2011, Year 2008

[50] A. Johnston, S. Donovan, R. Sparks, C. Cunningham, K. Summers - Network Working Group. *"Session Initiation Protocol* (SIP) Basic Call Flow Examples – RFC 3665". December 2003.

[51] B. Campbell, J. Rosenberg, H. Schulzrinne, C. Huitema, D. Gurle – Network Working Group. "*Session Initiation Protocol (SIP) Extension for Instant Messaging – RFC 3482*". December 2002.

[52] B. Campbell, J. Rosenberg. "SIP Instant Message Sessions, draft-ietf-simple-im-session-00". IETF Internet – Draft. January 11, 2002.