Enabling Secure IP Telephony in Enterprise Networks

By

BRENNEN EMERICK REYNOLDS
B.S. (University of California, Davis) 2001

THESIS

Submitted in partial satisfaction of the requirements for the degree of

MASTERS OF SCIENCE

in

Computer Engineering

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

_____

_____

_____

Committee in charge

2002

Enabling Secure IP Telephony in Enterprise Networks

Dedicated to Pop for his unselfish gift of knowledge and wisdom.

Acknowledgments

I would like to thank the members of my thesis committee, Dr. Matt Bishop, Dr. Chen-Nee Chuah, and especially Dr. Dipak Ghosal, for their invaluable guidance and support throughout the entire process of researching and writing this thesis. I would also like to thank Dr. S. Felix Wu for his comments on early drafts of the material.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 A New Paradigm of Services & Applications

Over the past two decades, network traffic has evolved from short text messages to high volume multimedia driven content. With the rapid introduction of new applications and services into the global networks, there are new requirements in the operation of the network. For example, new real-time applications demand a much higher quality of service (QoS) than existing applications such as transferring electronic mail.

A key outcome of this is an increase in the complexity of the network infrastructure. However, the fundamental operation of these devices has not changed. Enterprise networks of today are connected together through a large number of static devices which include firewalls, intrusion detection systems, and network address translation devices. These devices are collectively referred to as middleboxes. Middleboxes were originally designed to operate with traffic that can be specified using a simple static set of rules. They are capable of handling the applications currently deployed within enterprises, but that is changing.

New applications are being introduced in corporate networks. The functionality of these new applications is much more dynamic than existing services. The most widely deployed dynamic applications thus far have been Internet Protocol (IP) Telephony and video con-

ferencing. While these new applications offer services not previously available over data networks, problems have stemmed from the dynamic nature of the underlying protocols. Dynamic applications do not operate using the well-known port structure that traditional client-server services have used. [1] Each session of a dynamic application consists of multiple data streams with only the first control stream connecting to a well-known port on the remote terminal. The additional data streams, used for audio and video data in IP telephony and video conferencing, are sent between arbitrary ports negotiated by the endpoints using the initial connection.

Creating rule sets to allow traffic from dynamic applications to pass through static network middleboxes is a significant problem. Generalizing from a single session to an enterprise level only increases the complexity of rule creation. Because each session can theoretically use any of the 64k available ports, the only type of traditional rules that can be used for dynamic applications is to allow all traffic to pass. Implementing this type of rule set renders filtering devices including firewalls useless. Thus, this is not a practical solution for enterprises wishing to deploy dynamic applications.

1.2 IP Telephony in Converged Networks

An important dynamic application that is rapidly gaining adoption in the industry is IP telephony. A driving force behind the success of IP telephony is it allows enterprises to offer services and levels of integration not possible with standard telephony systems. A simple example is the addition of Click-to-Call services on product support pages to allow customers to automatically contact an enterprise support staff [1]. Another benefit of IP telephony is the reduction in overall operating expenses. With a complete deployment of IP telephony, only one enterprise wide network must be provisioned and maintained. The maintenance costs of a data network are much less than a circuit switched network because

[1] Services such as email, file transfer and the World Wide Web all operate at a standard port number or well-known port number.

1) statistical multiplexing gain in the data network and 2) less complex than circuit based networks. This, coupled with the improved bandwidth efficiency of the voice encoding algorithms used in IP telephony, dramatically reduces the per call cost across the enterprise.

IP telephony not only removes the need for two networks within an enterprise, it also allows the interconnection of two global networks. A full IP telephony deployment within an enterprise includes connections to the Internet and the Public Switched Telephone Network (PSTN). As a result, the enterprise's network acts as a bridge and control point between the two networks. The interconnection of these two networks will eventually allow terminals on either network to utilize and access resources and information contained in both.

IP telephony services can be deployed using one of two different protocols. The Internet Engineering Task Force (IETF) has developed a plain text protocol called Session Initiation Protocol (SIP) [23] based on the structure of the Hypertext Transfer Protocol (HTTP) [18]. The International Telecommunication Union (ITU) developed H.323 [31] which inherited the structure and basic functionality from the Signaling System 7 (SS7) [47] protocol used within the PSTN. The two protocols are not directly compatible but provide comparable features. H.323 was developed first and currently has a larger piece of the market share but recently SIP has been gaining momentum [58].

As with any new technology there are many different issues that must be addressed. Enterprises deploying IP telephony must tackle the management of new network resources, guarantee a high level of quality of service (QoS) within the network, and ensure that all IP telephony devices are interoperable. Each of these issues must be properly addressed for an IP telephony deployment to be successful.

1.3 Security Issues in a Converged Network

The convergence of the Internet and the PSTN is either an exciting or a very scary proposition depending on one's perspective. From an information accessibility and inte-

grated services viewpoint, the possibilities created by the interconnection is almost unlimited. However, the security implications surrounding the convergence are extremely significant. Both networks have unique vulnerabilities and threat models. By interconnecting the two, new threat models are created which not only encompass the two individual models but also new cross network vulnerabilities and attacks.

The security ramifications of two global networks converging is not the only element of IP telephony where security must be considered. Both IP telephony protocols are relatively new and still undergoing development. As a result, they have not been completely scrutinized and examined for vulnerabilities. Furthermore, implementations using the protocols are also very new and have not undergone exhaustive testing.

One significant difference between traditional telephony service in the PSTN and IP telephony is the manner in which data and control information are transported. In the PSTN the control and data are separated by different logical channels. This results in end terminals being unable to access and manipulate the control information. IP telephony removes the separation and incorporates both intelligence and control mechanisms in the end terminals. Furthermore, both the control and data information are now transmitted over a network maintained by numerous independent and sometimes competing network and service providers. The open architecture of the Internet greatly increases the ability of unauthorized parties to collect and/or modify information streams.

Ensuring the integrity of the call is a major concern that must be addressed before IP telephony can be widely deployed. In addition to the use of a single transport medium, use of the Internet Protocol [27] as the underlying transport protocol results in a larger number of vulnerabilities and weaknesses.

1.4 Key Contributions

The overall objective of this thesis is to allow dynamic applications, specifically IP telephony, to be deployed securely within an enterprise. To achieve this objective, three different issues are addressed in this thesis.

• The Secure Telephony Enabled Middlebox (STEM) architecture provides the foundation for securely deploying IP telephony services. It includes a security enforcement entity and an intelligent firewall capable of handling dynamic application sessions.

• A classification of vulnerabilities within STEM and the IP telephony protocols is carried out. The list categorizes the attacks by type and analyzes their impact.

• The Simple IP Telephony using Multi-layered Protection (STUMP) framework includes a property-oriented vulnerability analysis for IP telephony. Four different sensors detect and control most flood based DoS attacks targeting an IP telephony enabled enterprise network.

1.5 Thesis Organization

The remainder of this thesis is organized as follows. Chapter 2 expands on the interworkings of the layout of converged networks, the SIP protocol, and typical call setup scenarios. Chapter 3 describes a modified converged network architecture called Secure Telephony Enabled Middlebox (STEM). Details of the architectural components and device interaction are discussed in this chapter. The chapter also examines how several typical call scenarios are handled by the new architecture. The chapter concludes with a discussion of an effort to implement a prototype of the STEM architecture. Chapter 4 enumerates the major network vulnerabilities within a converged network and in IP telephony services. Chapter 5 introduces the Secure IP Telephony using Multi-layered Protection (STUMP) framework. The STUMP framework reduces the impact of a large number of the vulnerabilities outlined in Chapter 4 using both new and existing protection mechanisms. The

chapter also includes quantitative analysis of an attack sensors developed to handle flood based denial of service (DoS) attacks. A summary of this work and concluding remarks are given in Chapter 6.

Chapter 2

IP Telephony Over Converged Networks

2.1 Converged Network Architecture

A typical enterprise network consists of two sections: 1) the internal network and 2) the de-militarized zone (DMZ). The DMZ is connected to the public Internet through an external firewall and contains various servers that need to be accessed from external locations. This includes web, mail, and domain name service (DNS) servers. The internal network is connected to the DMZ by another firewall. In some architectures, the two firewalls are replaced by a single firewall with three network interfaces [12].

Deploying complex new services can require both additional devices to be added and

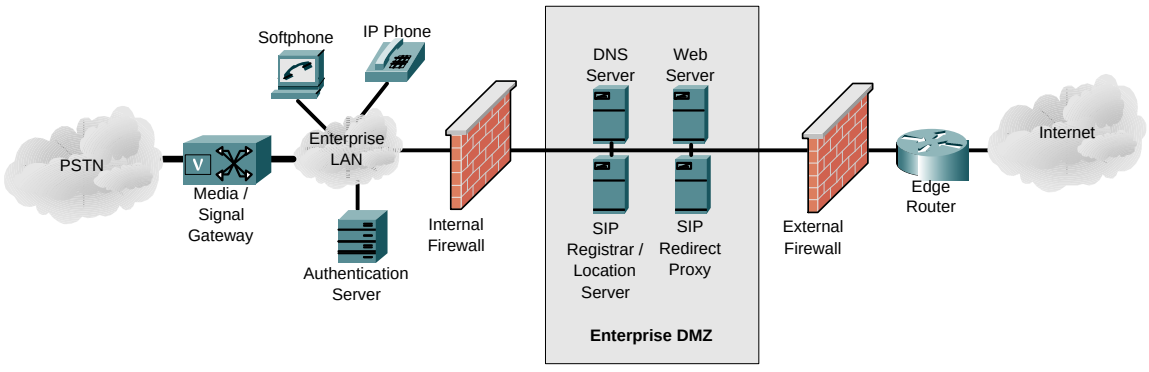


Figure 2.1: IP Telephony Enabled Enterprise Network

existing elements to be modified. IP telephony is no exception. An example IP telephony enabled enterprise network is shown in Figure 2.1. Additional components that are required include a SIP Proxy and a Registrar/Location Server, and a Media/Signal Gateway to connect to the PSTN.

The SIP Proxy server [46] (or H.323 Gatekeeper [21]) is placed in the enterprise DMZ. All IP telephony signaling and control traffic is routed through the proxy while the media streams bypass the proxy and are exchanged directly between the end terminals. The proxy server can support additional features such as lists of addresses used for Spam. The Spam lists could include both individual client lists as well as enterprise wide lists. The SIP Registrar/Location Server (RLS) is also located within the enterprise DMZ. The RLS's main function is maintaining the location (IP address) of end users within the enterprise. The RLS must also communicate with other RLSs to determine optimal call path selection for telephony routing over IP (TRIP) [45].

The Media/Signal Gateway (MSG) is a complete network stack proxy that connects the internal LAN to the PSTN. The MSG consists of voice ports bound to voice trunks on the PSTN side and an Ethernet connection to the internal LAN. Additionally, it may include a pair of SS7 signaling links to a Signal Transfer Points (STPs). The gateway provides control and data message conversion between the two networks.

In addition to the introduction of new devices in the enterprise network, certain existing network elements must be modified. The static firewalls must be replaced with new dynamic firewalls capable of parsing all layers of the network stack or application layer proxies to handle the dynamic applications traffic.

To enable cross network calls (PSTN-to-Net and Net-to-PSTN), the DNS service must be extended. Each telephony terminal must be assigned an E.164 number, a.k.a phone number, in a similar fashion to PSTN terminals. The DNS servers must then implement the ENUM protocol [17]. ENUM uses the NAPTR DNS Resource Record [14] type to store a mapping of E.164 number to a globally unique DNS name. All ENUM names belong to

the e164.arpa domain. While ENUM is required for PSTN-to-Net calls, it can also be used for Net-to-Net calls.

2.2 IP Telephony Protocols

Over the past four years, two protocols have been developed to support IP telephony; Session Initiation Protocol (SIP) and H.323. Both protocols provide a similar set of services but are very different architecturally. SIP, developed by the IETF, is based on the Hypertext Transfer Protocol (HTTP) and is a text based which uses the Session Description Protocol (SDP) [22], another text based protocol, for channel establishment and configuration. H.323, developed by the ITU, is an umbrella name given to a large suite of protocols. Within the H.323 protocol suite [21] are H.245 for call control, H.225.0 for connection establishment, H.235 for security and H.332 for conference calls. All the protocols in the suite utilize ASN.1 and packed encoding rules for generating messages. To manipulate these messages, special code-generators must be used. Both H.323 and SIP utilize the Real-Time Protocol (RTP) [51] to transfer the multimedia data. This means the choice of call control protocol does not influence the quality of service of a call [53]. Given that H.323 is a more complex protocol to decode and examine, this work is based on SIP. However, the solutions outlined are applicable to a H.323 environment.

2.2.1 Session Initiation Protocol

The Session Initiation Protocol [23] is an ASCII based client-server protocol (server side binds to port 5060) that uses either the Transmission Control Protocol (TCP) [40] or the User Datagram Protocol (UDP) [39] as a transport. The following quote from the RFC gives a detailed description of SIP's capabilities:

> "The Session Initiation Protocol (SIP) is an application-layer control (signaling) protocol for creating, modifying and terminating sessions with one or more participants. These sessions include Internet multimedia conferences,

> Internet telephone calls and multimedia distribution. Members in a session can communicate via multicast or via a mesh of unicast relations, or a combination of these. SIP invitations used to create sessions carry session descriptions which allow participants to agree on a set of compatible media types. SIP supports user mobility by proxying and redirecting requests to the user's current location. Users can register their current location. SIP is not tied to any particular conference control protocol. SIP is designed to be independent of the lower-layer transport protocol and can be extended with additional capabilities." [23]

SIP Network Elements

Network elements required to support SIP are shown in Figure 2.1. The SIP RFC [23] dictates that a SIP server can operate in two functional modes: proxy or redirect. In proxy mode, a server acts as a liaison and forward received messages toward their destination. Redirect servers do not pass messages on toward their final destination. Instead, they inform the calling terminal of the current location of the called terminal so the two terminals can contact each other directly. The Registrar/Location Server (RLS) must provide one basic function; a table mapping SIP URIs to an IP address for each user. The RLS functionality can be incorporated into a SIP server, but for enterprise level scenarios it is not recommended. Separating the two devices requires a directory service such as the Lightweight Directory Access Protocol (LDAP) [69] or a multicast-based protocol [49] to be used for queries between the SIP server and RLS. The SIP RFC does not directly include provisions for a Media/Signal Gateway (MSG), but it is needed to connect the enterprise to the PSTN. The translation between the SIP or H.323 signaling facilities and the Signaling System 7 (SS7) used to control PSTN calls are beyond the scope of this work and readers should refer to [47, 48, 65]. The user agents specifications state that both client and server capabilities must be included in an end terminal. This is required because the terminal must take on the role of a client when initiating a call and that of a server when receiving a call.

Message Routing

The design of SIP allows a call to be routed through numerous servers between end terminals. To ensure that all request and response messages follow the correct path a tracking mechanism is included with the protocol. The **Via** header in both the request and response messages is used to track the SIP servers the message has passed through. For request messages each SIP server must add its address to the end of the **Via** field. Thus, when the end terminal receives the request it has the exact path the message took with the server closes to it at the end of the list. The response message generated by the terminal will contain a copy of the **Via** field with the last entry removed. The terminal will use the address it removed from the end of the **Via** list as the destination for the response message. When a server receives a response message it removes the last entry in the **Via** field and uses it as the destination address the message of the next hop. This process continues until the response message arrives at the original terminal with the **Via** field empty. The ability to track the application-level path of the messages also allows SIP servers to detect loops that the underlying protocols may be unaware of.

SIP Message Header Overview

The address scheme used in SIP is similar to electronic mail. Each URI is comprised of a name and a host or domain separated by an @ sign (*name@domain*, *name@host*). However, since there is the possibility that a SIP terminal will be calling a terminal attached to the PSTN and not the Internet the SIP URI has an additional form. To reach PSTN terminals, the URI contains the phone number and the address of the MSG to route the call through (*phone-number@gateway*).

The six signaling methods included in the SIP RFC are outlined Table 2.1.

In addition to the request methods, SIP includes a fixed set of response messages. The response codes are based upon the ones used in HTTP. Table 2.2 lists the six categories and a broad description of the categories. For a detailed description of individual responses,

Table 2.1: SIP Request Methods

INVITE	This is the message that initiates the call. It includes information about the calling party, call-ID, call sequence number, called party and usually an SDP description of call parameters. It can also be used to modify the calls operating state while the call is taking place.
ACK	The calling agent responds with ACK only to INVITE requests that have been successfully accepted with a 200 code. The ACK can also contain the SDP description of the media capability of the called party.
BYE	A client sends this message when a call is to be released.
OPTIONS	This message is sent to query the capabilities of a call agent.
CANCEL	If a request is in progress this message will cause it to be canceled. The CANCEL message must include the call-ID, call sequence number and the source and destination the original request contained. It has no effect on an established call.
REGISTER	Clients use the REGISTER method to register the their address with a SIP server.

refer to the RFC [23].

Table 2.2: SIP Response Code Categories

1xx - Informational	Proceeding with the execution of the request.
2xx - Success	The request was successfully parsed and executed by the called party.
3xx - Redirection	The call needs more processing before it can be determined if it can be completed.
4xx - Request Failure	The request cannot be parsed by the server or cannot be serviced.
5xx - Server Failure	The request may be valid, but the server cannot execute it.
6xx - Global Failure	The user request cannot be serviced by any server.

An example INVITE request is show in Figure 2.2. The second part of the request is the SDP header and will be discussed in the next section.

2.2.2 Call Control Using Session Description Protocol

While SIP is used to initiate a call, it does not include the capability to configure the parameters that will govern the call including audio and video codecs. The Session De-

```
INVITE sip:ghosal@cs.ucdavis.edu SIP/2.0
via: SIP/2.0/UDP 192.168.0.80:5062
from: sip:bereynolds@ece.ucdavis.edu
to: sip:ghosal@cs.ucdavis.edu
call-ID: 84532652@192.168.0.80
CSeq: 1 INVITE
content-type: application/sdp
content-length: 250

v=0
o=bereynolds 84532652 84532652 IN IP4 192.168.0.80
s=Feedback on M.S. Thesis
c=IN IP4 192.168.0.80
e=bereynolds@ece.ucdavis.edu
t=2873397496 2873404696
m=audio 4657 RTP/AVP 0
```

Figure 2.2: Sample SIP INVITE Message

scription Protocol (SDP) [22] is used to describe a multimedia session. The SDP header serves three purposes: the type of information to be exchanged (audio, video or both), how the sender should encode the information and where to send the information. For unicast calls the sender of the SDP description is requesting that the receiver use the configuration specified in the description but for multicast calls the SDP message indicates the configuration the sender will use to transmit.

A sample SDP description is included at the bottom of Figure 2.2. Many fields defined by the RFC are not included in this basic example. The RFC [22] contains a complete list of fields and their full definitions. The following list provides a brief description of the fields used in the example in Figure 2.2.

- v = protocol version (only version 0 has been defined by the RFC)
- o = originator of the session and session identifier
- s = session name
- c = connection data

- e = email address of originator
- t = time session is active
- m = media announcement including media type, destination port number, transport level application and media format (codec)

2.2.3 Media Transport Using Real-Time Protocol

With the call setup complete the transmission of media is done using the Real-Time Protocol (RTP) [50]. RTP consists of two parts; RTP itself for transmitting the data and RTCP, Real-Time Control Protocol, for real-time channel control and quality of service. To achieve as close to real-time deliver as possible RTP runs over the UDP but is capable of using any transport layer protocol. The RTP streams use the destination ports included in the SDP header of the call. RTP always uses an even numbered port and the corresponding RTCP stream uses the next odd numbered port. Unlike SIP and SDP, RTP is a binary protocol. Figure 2.3 shows the complete layout of a RTP packet.

2.2.4 SIP Call Models

The SIP protocol defines four basic call models. The direct call model shown in Figure 2.4 is the most simplistic of the four models. It represents a direct call between two terminals without the involvement of a SIP server. The first step is to setup a TCP connection for the control stream of the call. Using the TCP connection, the calling terminal sends an INVITE request directly to the called terminal. This results in the called terminal generating several responses to inform the calling terminal of the status of the call; trying and ringing. A success response is sent after the called terminal answers. A final acknowledgment from the calling terminal completes the call setup. As well a establishing the call, these initial request-response messages configure how the media flows will operate using SDP. Upon call setup completion, the media flow using RTP begins. At the end of the call, the terminals exchange BYE messages and close all open connections.

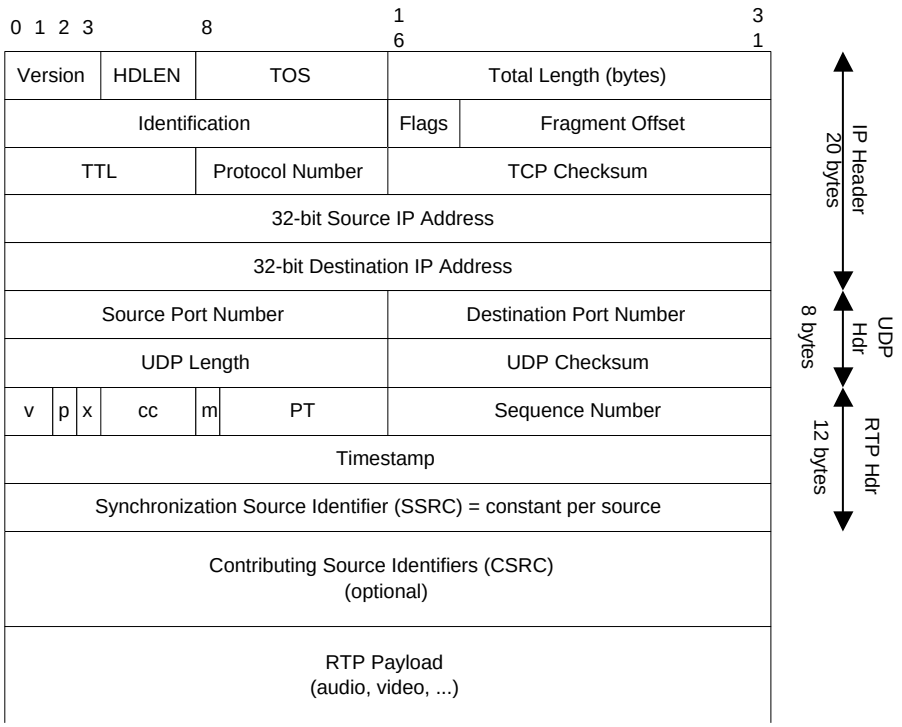


Figure 2.3: RTP Packet Layout

The redirect call model closely resembles the direct call model with the exception of the initial message sequence. As Figure 2.5 shows, the calling terminal sends an INVITE request to the redirect server to obtain the current address of the other terminal. Upon receiving the current address, another INVITE request is sent. After this, the call proceeds as in the direct model.

The proxy call model in Figure 2.6 can be viewed as two direct calls bridged together. Each terminal communicates with one or more intermediate proxy servers. The end terminals create TCP connections with the proxy. The proxy then forwards messages between the two terminals. The RTP media streams do not pass through the proxy, but are sent directly between the two end terminals.

The PSTN call model differs from the others in one regard: the call spans both the data network and the PSTN. Figure 2.7 shows how the MSG translates the various messages between the two networks. The sequence of messages follows very closely to the proxy

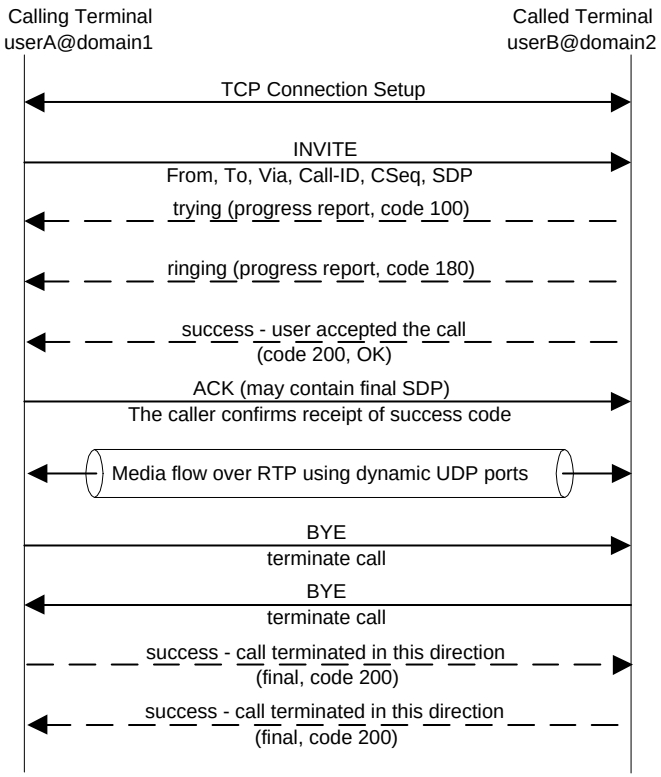


Figure 2.4: SIP Direct Call Model

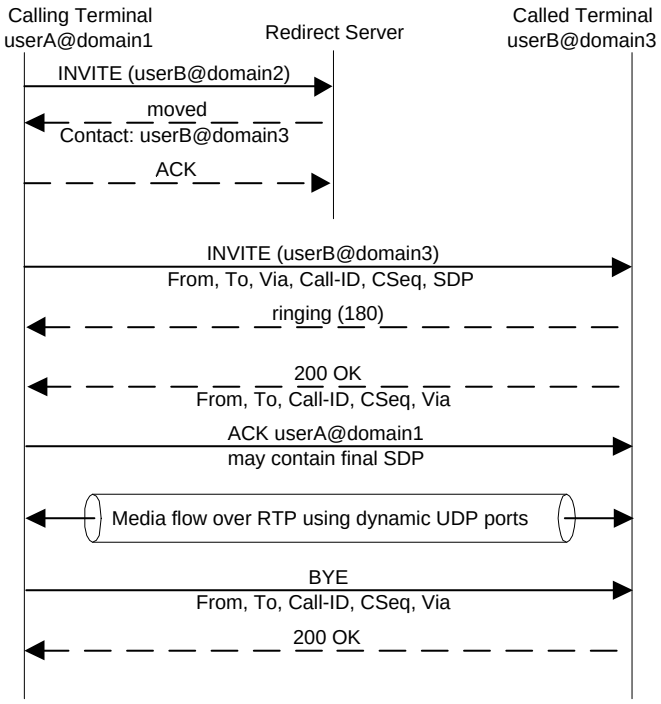


Figure 2.5: SIP Redirect Call Model

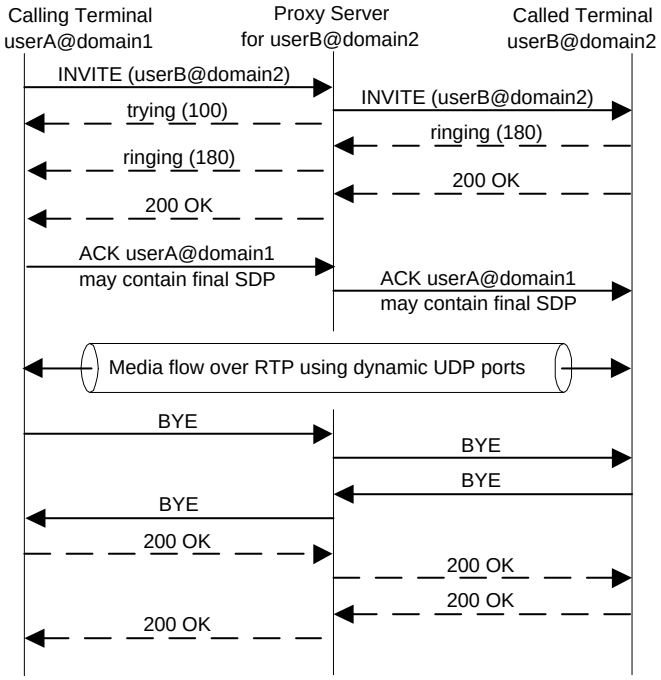


Figure 2.6: SIP Proxy Call Model

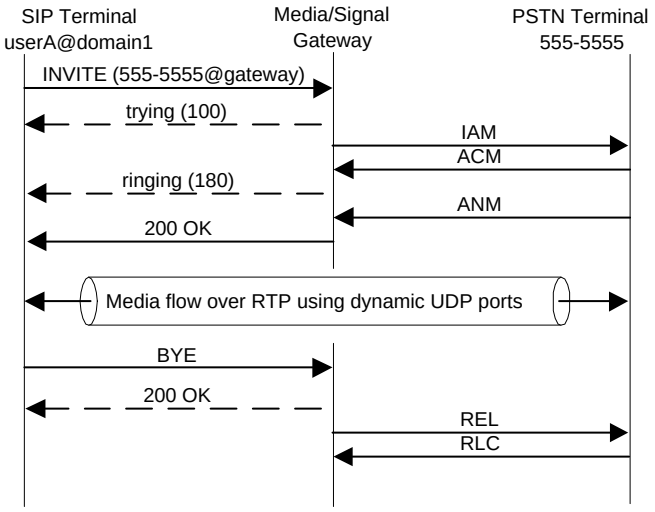


Figure 2.7: SIP PSTN Call Model

call model.

In addition to the basic call models, SIP is also capable of providing multiparty conferences. SIP allows conference calls to be constructed via three different modes: network-level multicast, dedicated bridges (also known as multipoint control units) or a full-mesh of unicast connections. Each mode fits certain conference settings better than others. For large conferences multicast or bridges are more function, but for three-way calling or small groups a full-mesh of unicast connections is a more appropriate configuration. For additional information on the conference calling capabilities of SIP refer to [54, 55].

Chapter 3

STEM - Secure Telephony Enabled Middlebox

To securely provide IP telephony services within an enterprise, security must be a major consideration from the beginning. This chapter first outlines work that has been done to address the middlebox box issue. However, as was shown in the previous chapter, IP telephony is a very complex application. Therefore, the entire network architecture must be considered if the service is to be deployed securely. This chapter introduces the Secure Telephony Enabled Middlebox (STEM) architecture. The STEM architecture provides a solid foundation to securely deploy IP telephony and other dynamic applications within an enterprise environment. While a STEM enabled enterprise network differs from traditional enterprise network and basic IP telephony networks, it does not require any modification to the IP telephony protocols stack.

The remainder of the chapter is comprised of four sections discussing different aspects of the STEM architecture. The first section covers the required network components and their functionalities with respect to the SIP protocol. (Note: the H.323 protocol works very similarly.) Next, the protocols used in communication between the components are outlined. It is important that the IP telephony protocols created by the IETF and ITU

operate normally within a STEM network. The third section examines in detail a number of call setup scenarios. The final section discusses some work that was done to create a prototype of the STEM architecture.

3.1 Previous Work

Solutions have been presented during the past two years that attempt to solve some of the problems caused by deploying dynamic applications in static environments. All of the solutions modify the middleboxes on the perimeter of an enterprise's network, namely the firewall. This section includes an overview of the previous solutions to the static middlebox problem. The reader should reference the cited works if a detailed description of the work is required.

3.1.1 Application Layer Proxies

A solution proposed in [33] by Jiri Kuthan examines the creation of a protocol to allow manipulation of the firewall configuration from an external device. In their topology, Figure 3.1, a new device was added for handling only IP telephony traffic [33]. This new device sits adjacent to the firewall and all IP telephony control traffic is routed through it instead of the firewall. The IP telephony proxy may be required to perform network address translation of the data streams if the internal network uses private addresses.

The IP telephony proxy and the firewall communicate via a protocol created by Kuthan called Firewall Control Protocol (FCP). FCP allows the proxy to instruct the firewall to pass only the RTP packets associated with a IP telephony call. The proxy extracts the ports that will be used by the RTP streams during the call setup and instructs the firewall to allow the appropriate RTP streams. The FCP messages allow the proxy to specify the exact tuple that will match the media stream. The fine granularity of the stream specifications is to to ensure that only legitimate traffic traverse the firewall.

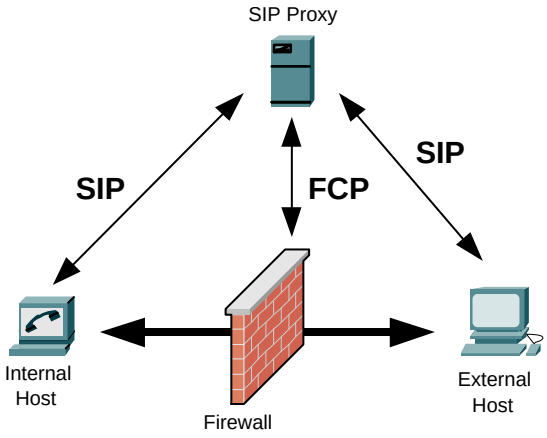


Figure 3.1: Decomposed Firewall Architecture

This solution off-loads the task of determining the port numbers used by the media flow from the firewall. Additionally, by not incorporating the proxy into the firewall, an enterprise is not dependent on updates by firewall vendors to be able to deploy IP telephony. This solution allows a large volume of calls to be handled without affecting the performance of the firewall because the proxy handles the bulk of the work. However, the FCP messages can cause a bottleneck in the system. With a large call volume, the overhead of adding and removing the rules via a user-space daemon on the firewall would be significant. There are also major security issues surrounding Kuthan's solution that are not addressed. He does comment that a real world deployment must consider authenticating the FCP messages when deploying this solution in a hostile network. Suggestions are given to run FCP through another protocol such as IPSec [63] or Transport Layer Security (TLS) [15]. A cautionary message that firewalls should check all FCP messages against a basic set of access controls before committing the changes is also included.

A drawback to this solution is that additional devices must be added for each dynamic application deployed. This will ultimately reduce the overall protection provided by the firewall because various traffic will never be inspected by it. In [33] it is assumed that each proxy will use their detailed knowledge of the protocol to ensure that only legitimate traffic

passes through them, but he does not make any assumptions about the security of the system the proxy resides on. These additional systems are completely exposed to the Internet and could become additional targets for attackers trying to gain access to the internal network.

3.1.2 Layered Firewall Architectures

A new internal architecture for enterprise firewalls was presented in [43]. Within the firewall are layers of abstraction are placed between the internal components as shown in Figure 3.2. These layers, called adaptation layers, exist between the firewall, internal operating system, IP telephony parser, and external components. The layers provide a common API between each component. This allows modules to be interchanged without any inconsistency assuming the module's APIs are consistent.

The adaptation layer between the IP telephony parser and the external interface allows communication with other devices attached to the network. This communication channel is by external devices to query the operation of the firewall and to allow the behavioral modifications to be made remotely. The specifics of the protocol to be used are not addressed in [43] nor is the issue of message authentication. By allowing external devices to modify the internal components of the firewall, a potential hostile target might be able to cause the firewall to operate in an undesirable manner. This firewall architecture allows enterprise to use IP telephony, but the lack of authentication mechanisms ensure that this solution needs additional work before it can be deployed.

3.1.3 IETF Initiatives

Academic institutions are not alone in investigating the problem of supporting dynamic applications through static devices. The Internet Engineering Task Force has created several working groups (WG) to examine different aspects of this area.

The Midcom WG [24] was formed to design a protocol similar to the Firewall Control

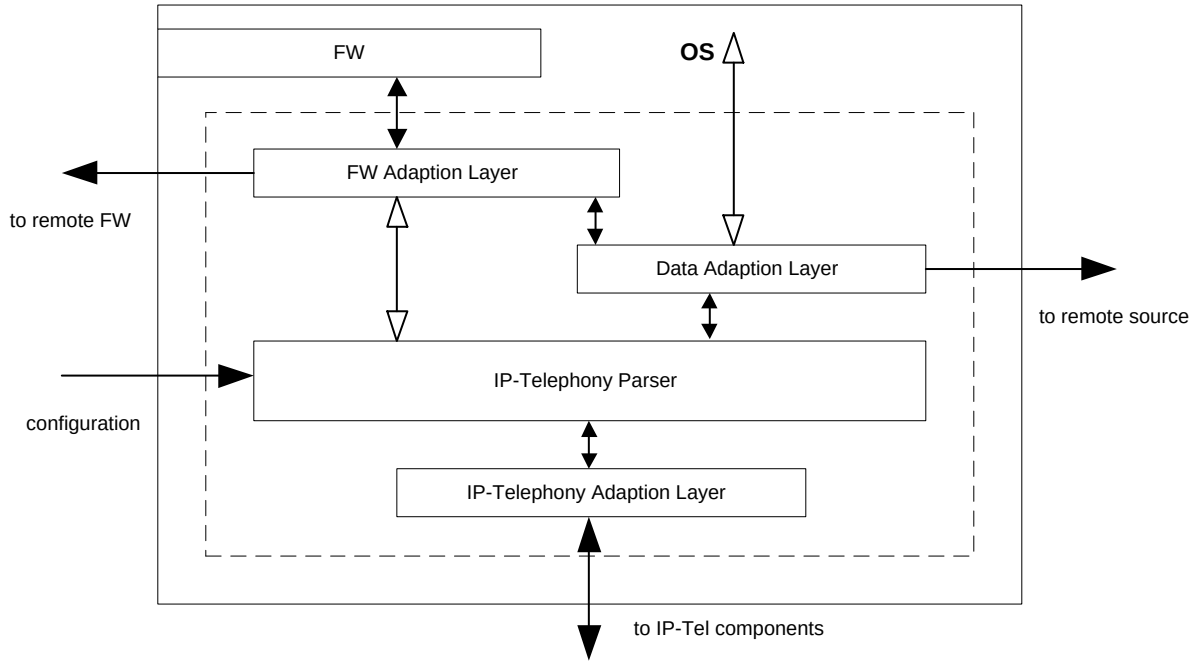


Figure 3.2: Adaption Layered Firewall Architecture

Protocol (FCP) previously discussed. The WG has published several preliminary documents describing the functional requirements the protocol must fulfill [57, 61, 5]. The protocol is being designed to allow it to be used for communication with a wide array of network middleboxes.

There are several working groups dedicated to the protocols used by dynamic applications. The SIP WG [26] and SIPPING WG [25] are tasked with the creation, extension and maintenance of the SIP protocol [46, 44, 30, 38]. The MMusic WG has a similar responsibility for the SDP protocol [22, 52]. Finally, the RTP protocol is assigned to the Audio/Visual working group [50, 7].

3.2 Architecture Components

In designing the STEM architecture, one overall goal was to leverage existing enterprise infrastructure. The STEM network layout closely mirrors a typical converged enterprise

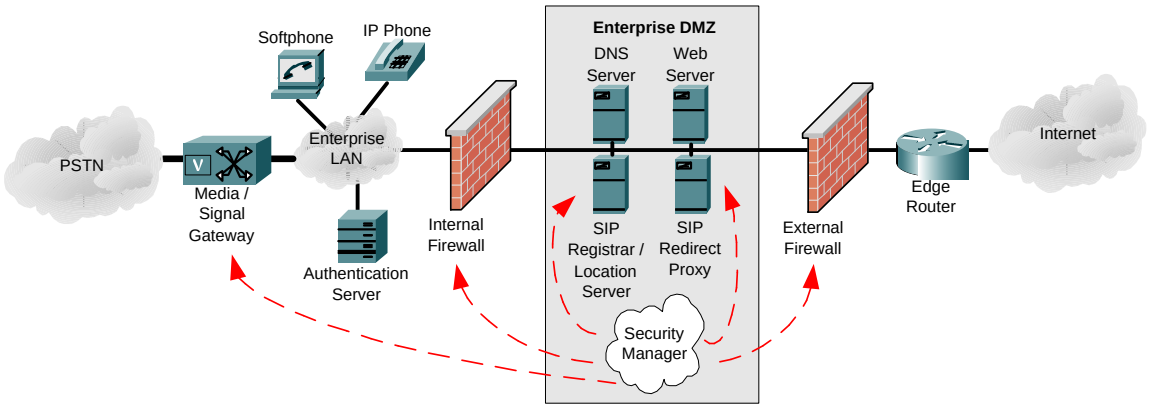


Figure 3.3: Secure Telephony Enabled Middlebox Architecture

network structure. Figure 3.3 is an example of a STEM enabled enterprise network. The enterprise network is partitioned into two sections, the internal LAN and the DMZ, in addition to being connected to the Internet and PSTN.

A comparison of Figures 2.1 and 3.3 shows that new components are required by the STEM architecture. However, several of the components within the STEM architecture include enhanced functionality and features not found in traditional converged networks. The remainder of this section discusses the operation of the new components and the enhancements made to the existing components.

3.2.1 Security Manager

The overall security in STEM enabled enterprise is centrally managed by the Security Manager (SM). While the SM operates as the overseer of all operations within the network, it is not a physical entity. The functional units of the SM are incorporated into different network elements. The SM entity shown in Figure 3.3 simply functions as a central portal into the distributed components. The four functional units of the SM are a) URI mappings, b) URI preferences, c) SPAM lists, and d) user authentication.

Each end user or terminal will have a unique SIP URI. Within the proposed IP telephony protocols, the URIs are not bound to either machine addresses or physical locations.

The implication of this is that employees may roam within an enterprise and be able to receive calls to a single URI. Therefore, a mapping between the URI and the network address of the machine the employee is currently at is very important. The Security Manager is responsible for maintaining the database correlating user addresses (SIP URIs) to network addresses (IP addresses). Both IP telephony protocols include provisions for this type of functionality. In SIP, the Registrar / Location Server (RLS) is responsible for this. In addition to the basic URI to IP mapping, information must also be included that determines if the employee is connected to the enterprise locally or remotely. In the case of an employee connecting remotely, all incoming calls must be first routed through the enterprise's Virtual Private Network (VPN) concentrator.

In addition to providing the URI mapping, the RLS is also responsible for the second functional unit of the SM: the URI preferences. Each URI has a set of preferences associated with it. This includes, but is not limited to: automatically accept new calls, forwarding calls to another user or service (e.g. voice mail), automatically dropping calls or querying the user when a call arrives. The configuration of these preferences determines how the SIP Proxy will route incoming calls from both the Internet and the PSTN.

The third component of the SM is the Spam address lists. This consists of two different list types. One is an enterprise wide list that resides on the SIP Proxy. Any incoming call from an address on this list is automatically rejected. The severity of this is large enough that the list should only be utilized in extreme cases and after all other avenue of resolution have been exhausted. The second list type is kept on a per URI basis. Again, this information is stored by the RLS. Each employee is able to manager their own list of offending addresses. Upon reception of a new incoming call, the SIP Proxy queries the list stored on the RLS.

The final element of the SM is a user authentication enforcement mechanism. This is implemented in both the SIP Proxy and the Media/Signal Gateway (MSG). To ensure that only legitimate users are allowed to make outgoing calls, both the SIP Proxy and the

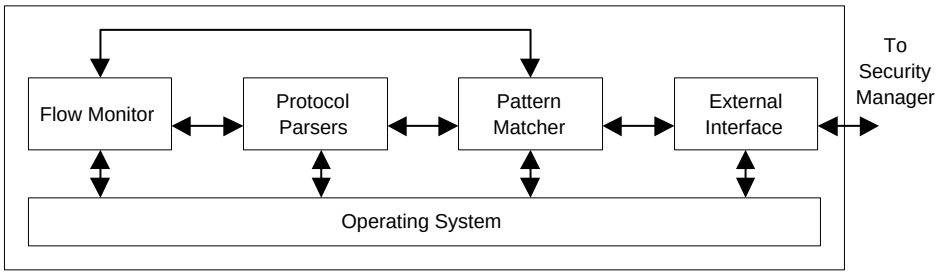


Figure 3.4: Internal Layout of STEM Firewall

MSG require users to be authenticated. Almost all enterprises have an existing centralized authentication infrastructure, therefore, the SIP Proxy and MSG simply request the credentials provided to the end users by the existing infrastructure. This can include, but is not limited to: active directory, Kerberos [32], and Radius [42].

3.2.2 Firewall

The firewalls within the STEM architecture fill an equally important as the Security Manager. They ensure that traffic flows follow a specific path and provide boundaries between the different network segments. To accomplish this goal, the functional behavior of the firewalls in the STEM architecture are much different than traditional firewalls. The firewalls are capable of parsing traffic from dynamic applications as well as providing traditional static filtering.

The internal design of the firewalls in the STEM architecture is an enhancement upon existing dynamic designs. The basic internal layout was adopted from [43]. The interaction of the discrete internal components is shown in Figure 3.4. By designing each component as a self-contained entity, it is possible to dynamically load and unload them without impacting the other functions of the device. Each class of components have a common interface to allow them to easily be interchanged and upgraded.

• **Protocol Parser** The most important block in the firewall architecture is the Protocol Parser. This component is comprised of multiple parsers. Each parser is designed to understand the operation of a single complex protocol.

The SIP parser includes a call monitor component that is responsible for ensuring that each call follows the protocol specified state transitions. The dynamic port numbers are extracted from the call setup and passed to the Pattern Matcher which opens the appropriate pinholes. There is the possibility of two calls selecting the same port numbers. Therefore, the SIP parser must monitor both the port and internal IP address associated with a data stream. It will instruct the Pattern Matcher to completely close a port only after all streams have terminated. Additionally, the parser also extracts the media codecs advertised by the terminals during call setup. This information is passed to the Flow Monitor to detect malicious streams.

• **Flow Monitor** The Flow Monitor is designed to handle malicious data streams. It monitors the data rate of the call streams and if they exceed the threshold set by the bandwidth requirements advertised during the call setup the firewall can respond. The triggered response can be set by the individual network administrators and could include dropping packets of the malicious stream or applying a traffic throttling algorithm.

• **Pattern Matcher** The Pattern Matcher is the most basic component in the firewall and all packet filter firewalls include this component. It allows configuration of static rule-sets using machine addresses, transport protocols, and port number specifications [59]. Each rule-set has an action assigned to it that is executed when the rule is triggered.

• **External Interface** The External Interface component allows the firewall to communicate with other devices. It is responsible for parsing incoming messages from other components and generating appropriate response messages.

3.2.3 Media/Signal Gateway, SIP Proxy & SIP Registrar

All three of these components have been modified slightly from their default behavior in a converged network. The Media/Signal Gateway is the least modified component. The only modifications is the validation engine used to allow outgoing calls to be completed.

The SIP Proxy has been modified to query the Registrar/Location Server for multiple pieces of information including the IP address the called URI is currently registered at and how to direct the call or if it should be rejected. It also must be able to validate user credentials when an outgoing call is placed.

The RLS now must store the additional information outlined previously as well as allow users to make modifications to this information. To ensure that users can only manipulate their own information, the RLS must also include the ability to validate user credentials.

3.3 Inter-Device Communication

There are numerous communication sessions required to have a STEM network operate smoothly. However, a large number of those can be handled by protocols already being used within the network. All of the user authentication is handled via the pre-existing infrastructure. The queries between the SIP Proxy and RLS can be done using a structured database language such as SQL or via services such as LDAP. The preference and Spam list configuration sessions between the end users and RLS can be done using the SIP protocol.

The only communication sessions that require a new protocol are those between the Security Manager console and the firewalls. A configuration protocol to allow a middleboxes internal state to be modified remotely is currently under development by the IETF Midcom Working Group. A final version has not been released yet, but a protocol requirements [60] and architectural framework [56] have been released detailing the overall functionality of the final protocol.

3.4 Example Call Scenarios

The topology of a converged network and the flexibility of the IP telephony protocols results in a large number of call setup scenarios. Therefore, this section only addresses the more common types of calls within an enterprise setting. These scenarios can be partitioned into four categories: Incoming Net-to-Net, Outgoing Net-to-Net, PSTN-to-Net and Net-to-PSTN. For a comprehensive listing of call scenarios please refer to [30].

3.4.1 Incoming Net-to-Net Calls

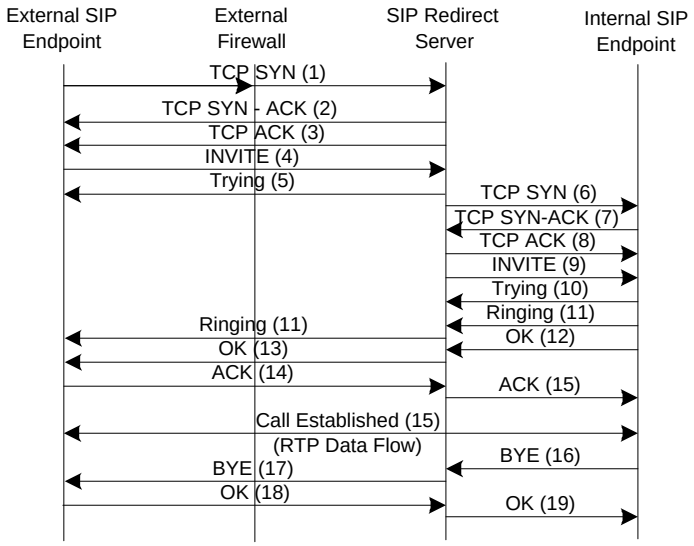
All incoming Net-to-Net calls follow the same call setup. The external calling terminal initiates a TCP connection to port 5060[1] of the destination terminal. When the initial TCP SYN packet arrives at the external firewall, the SIP Protocol Parser identifies it as a new call. The parser routes the SYN packet to the enterprise's SIP Proxy regardless of the IP address the packet was initially destined for.

The SIP Proxy completes the TCP handshake with the calling terminal. This TCP connection will be used to transmit all of the call setup information. The calling terminal sends a SIP INVITE request over the TCP connection to the proxy. When the INVITE passes through the external firewall, the Protocol Parser extracts the ports the calling terminal will use to transmit their media stream when the call begins. The SIP Proxy queries the RLS to determine how to handle the new call request. If the request is accepted, the RLS provides the SIP Proxy the IP address the request is to be sent to.

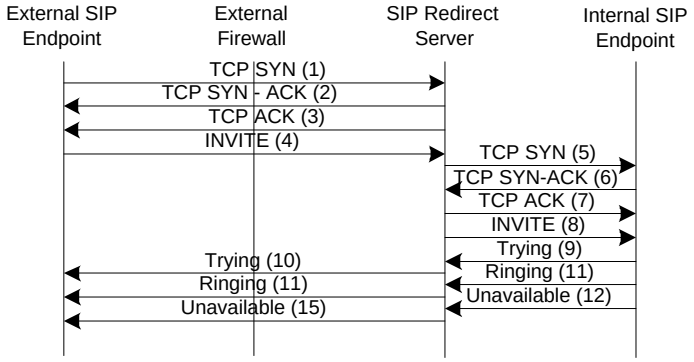
Before the INVITE can be forwarded to the called terminal, a TCP connection must be established between the proxy and called terminal. After the call request has been relayed to the called terminal, the proxy acts as a bridge between the two control channel TCP connections.

In the case where the called terminal accepts the calls, Figure 3.5a, the SIP *OK* response

[1] SIP servers listen on the well-known port 5060 as specified in the RFC [46].

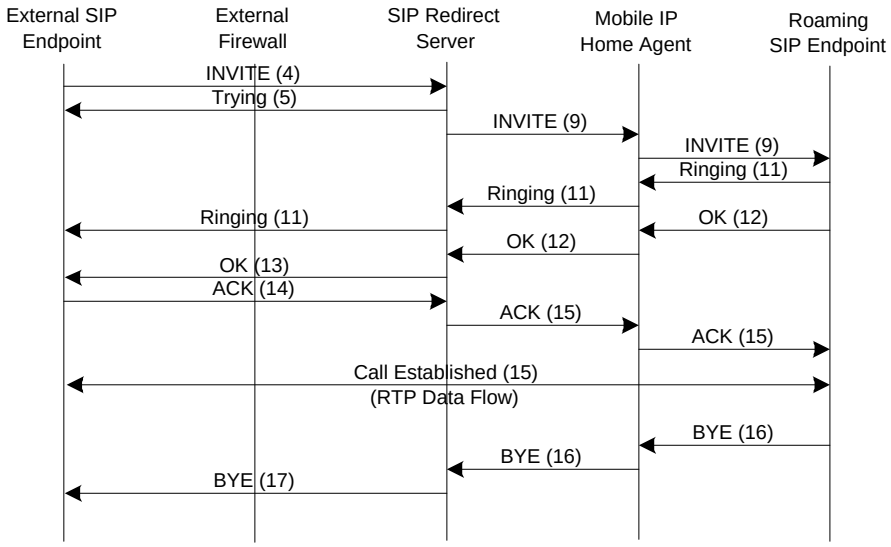


(a) Successful



(b) Called Party is Busy

Figure 3.5: Incoming Net-to-Net Call Flows - Part 1



(c) Called Party is Roaming

Figure 3.6: Incoming Net-to-Net Call Flows - Part 2

will contain the SDP fields specifying the ports the called terminal will used to transmit their media stream. This information will be extracted by the Protocol Parsers in both firewalls and the appropriate pinholes will be opened. This message concludes the call setup phase and the two terminals begin exchanging RTP media streams directly.

If the called terminal does not accept the call or is busy, Figure 3.5b, the *Busy Here* or *Temporarily Unavailable* response will be generated by the called terminal. The Protocol Parsers will remove a call from their watch tables after seeing this response because no additional call traffic exists.

It is also possible for the called user to not be connected to the enterprise LAN locally. A user must inform the RLS if they are tunneled into the enterprise network via a VPN or some other means. When an incoming call arrives at the proxy for a user connected via a VPN, the call is typically routed to either the VPN concentrator or the called terminal's Home Agent [37]. Figure 3.6c shows an example where a call is routed through the Home Agent of an employee using Mobile IP.

3.4.2 Outgoing Net-to-Net Calls

The structure of outgoing Net-to-Net calls follow a format similar to incoming calls. All calls must be routed through the SIP Proxy located in the DMZ. To enforce this, the two firewalls have static filters in place to only allow SIP traffic from 1) the internal network to the Proxy and 2) the Proxy to the Internet. The filters, coupled with the authentication required to make outgoing phone calls, ensure that unauthorized users are not able to make calls to the terminals on the Internet.

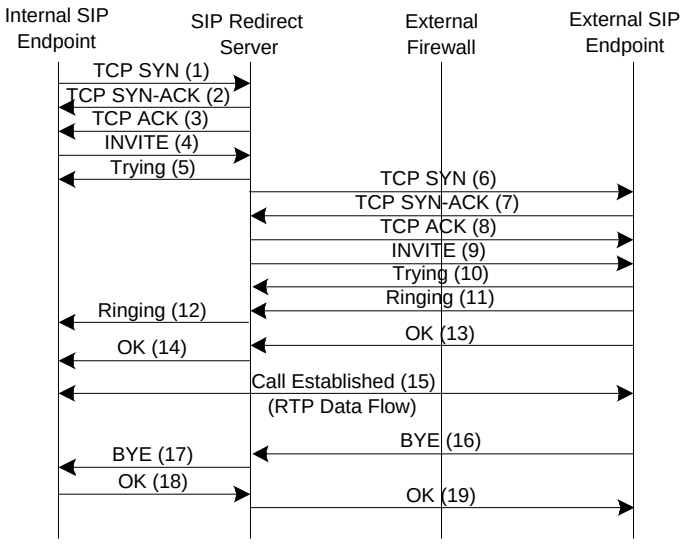
After establishing the TCP connection with the proxy and authenticating, the calling terminal sends the initial INVITE request. The message flow after this is influenced by the call being accepted or rejected. Figure 3.7a shows an example where the call is accepted while Figure 3.7b shows what happens when the called terminal does not answer and the call is canceled. As in the case of incoming calls, the firewalls do not open the pinholes for a call until the call acceptance message is sent.

3.4.3 PSTN-to-Net Calls

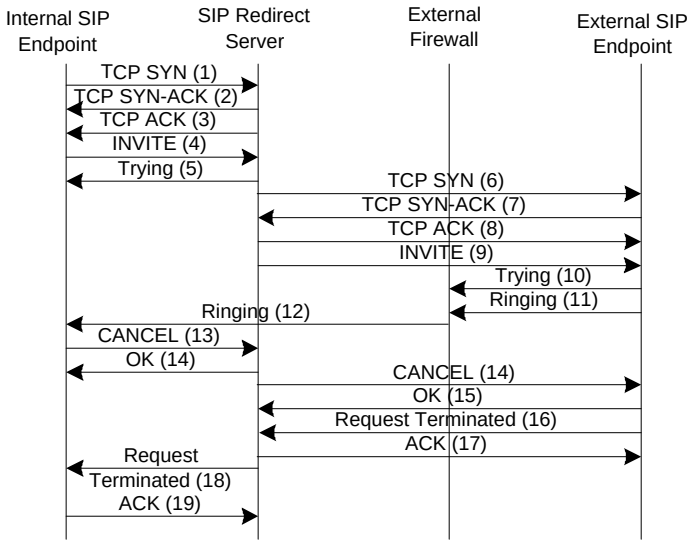
To allow calls to be placed between PSTN terminals and terminals on an IP network, each terminal in the IP network must be assigned an address that is capable of being specified by terminals attached to the PSTN, e.g., a phone number (or E.164 number). To allow E.164 numbers to be assigned to IP terminals, the DNS service within the enterprise must include the ENUM extension [17]. The result of this global naming scheme is that PSTN terminals are unaware they are communicating with terminal on a different network. The translation between the two network protocol stacks is performed by the Media/Signal Gateway.

Figure 3.8a shows the message sequence required to connect a terminal on the PSTN to one on an enterprise network. The SS7 network routes the dialed number to the enterprise MSG[2]. A voice port on the gateway is allocated for the incoming call. The MSG translates

[2] Details of SS7 routing are beyond the scope of this work, refer to [47].

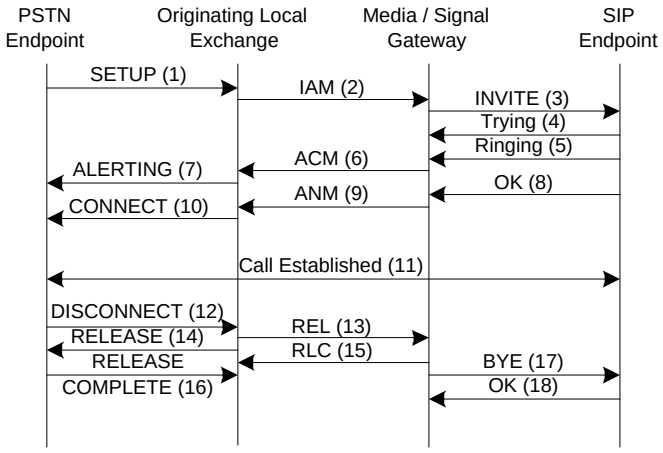


(a) Successful

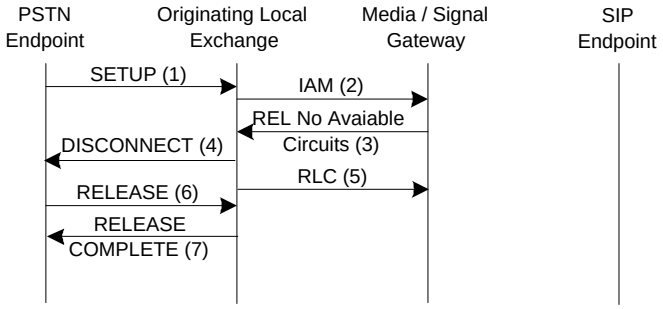


(b) Called Party Does not Answer

Figure 3.7: Outgoing Net-to-Net Call Flows



(a) Successful



(b) Voice Ports are Filled

Figure 3.8: PSTN-to-Net Call Flows

the E.164 number to an IP address using ENUM. After the destination address has been resolved, the gateway establishes a IP telephony (e.g., SIP) connection with the called terminal. In this scenario, the called terminal accepts the call and the message is relayed through the gateway back to the calling terminal. When the call terminates, the appropriate tear down messages are transmitted, the circuits are released, and the voice port in the gateway is freed.

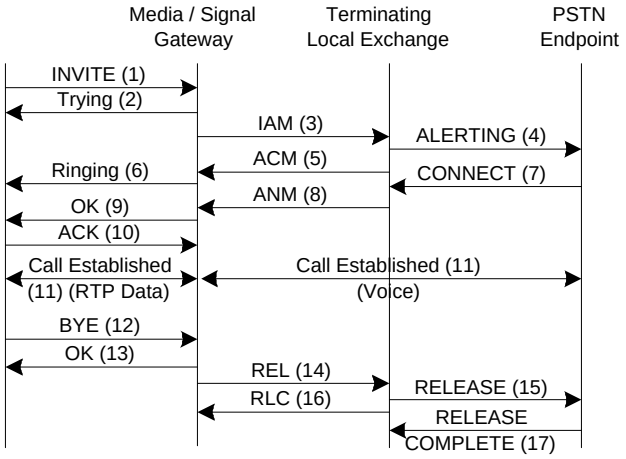
It is also very possible if there are a large number of cross network calls, that when a new call reaches the MSG there are no available voice ports. When this happens, a busy message is returned to the calling terminal as shown in Figure 3.8b.

3.4.4 Net-to-PSTN Calls

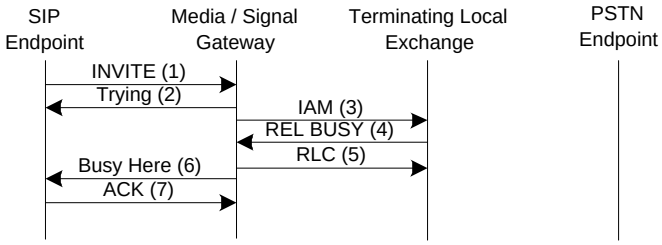
Calls originating from the enterprise's LAN for terminals on the PSTN are structured in a similar manner to those destined for terminals on the Internet. Instead of using the SIP Proxy as a bridge between two TCP connections, the MSG is used. The user must first authenticate to the MSG before sending an INVITE request and having a voice port allocated. For calls destined for the PSTN, the URI within the INVITE request must use the alternate form. The destination terminals E.164 number is used as the user field and the MSG's IP address is the host field when constructing the URI.

The MSG translates the INVITE message into the SS7 IAM message. The IAM message is routed via the SS7 signaling interconnects to the Terminating Local Exchange (TLE). For successful calls, such as the one shown in Figure 3.9a, the TLE alerts the end terminal and the call continues.

For calls where the called PSTN terminal is busy the TLE returns a Release (REL) message with the busy flag set. Figure 3.9b shows that the MSG translates the REL message into a SIP *Busy Here* response which is returned to the calling terminal.



(a) Successful



(b) Called Party is Busy

Figure 3.9: Net-to-PSTN Call Flows

3.5 Implementing a Prototype of STEM

A rapid prototype implementation of the STEM architecture was attempted. The various components were to be built upon a modified version of the Linux operating system [2]. However, it was discovered as work progressed, that the current state and facilities provided by the 2.4 series kernel was not sufficient. This section discusses the approach that was adopted to implement STEM.

3.5.1 Building a Dynamic Firewall

The implementation of the dynamic firewall needed in the STEM architecture was the biggest challenge. The 2.4 series Linux kernel incorporates the Netfilter firewalling subsystem [3]. As shown in Figure 3.10, there are several kernel hooks within Netfilter that allow modules to gain access to packets at different processing stages. In addition to providing well defined points in the routing paths, the framework also allows for the creation of Protocol Helper modules to allow the subsystem to parse complex protocols. These two properties were the deciding factors to use Linux as the base for implementing our dynamic firewalls.

The static filter capabilities are well defined and implemented in the existing Netfilter code as well as the ability to do stateful tracking of connections. Each Protocol Parser in the new dynamic firewall could be written as a Protocol Helper module and dynamically loaded and unloaded within the firewall as needed. Because this was only going to be a prototype, only a SIP Protocol Parser was to be built.

The Flow Monitor component in the new firewall was designed to be built upon the Traffic Control (TC) framework available in the 2.4 series kernels.

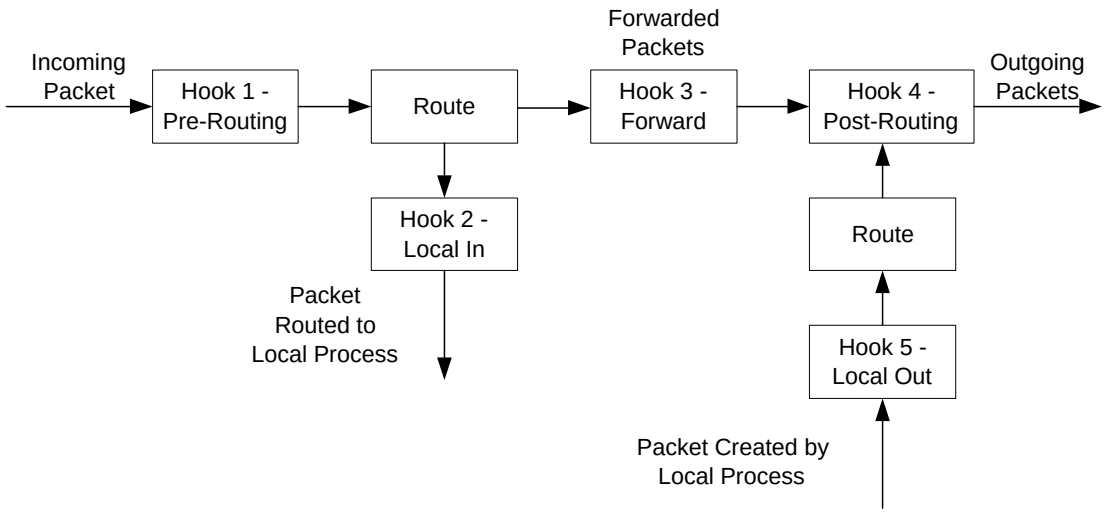


Figure 3.10: Netfilter Hook Layout

SIP Protocol Parser

Incoming Connection Mangling All incoming SIP calls to the enterprise are routed to a central SIP server. This requires the IP address to be modified for all SIP traffic. The Destination Network Address Translation (DNAT) hook in Netfilter allows a module to change the destination IP address of an incoming packet. By creating a tuple and mask to match any incoming traffic to port 5060, all incoming calls and related call control information can be re-routed to any destination. The Source NAT (SNAT) hook allows the reverse operation to be done. This allows the external hosts to believe they are taking directly with the called host but are actually being relayed through the proxy.

The RTP streams are not proxied through the central server, but are constructed between the two end terminals directly and therefore, it is not necessary to mangle the IP address of the packets. (Note: see section on NAT for special case).

Tracking SIP Call Control Flows The SIP protocol provides a call control framework to setup and manipulate IP telephony calls. It can operate over any transport layer protocol, but most implementations use either UDP or TCP. Each end terminal must provide both a

client interface and a server interface. Therefore, unlike most transport level connections, two uni-directional streams are associated with each call control. This situation is further complicated because some soft-phone implementations send each control message over a different socket. As a result, a large number of data streams are logically linked to a single call.

The connection tracking capability of Netfilter allows for helper modules to be written for protocols which do not operate in a standard fashion. However, the current design of the connection tracking element does not handle multiple related streams easily. Additionally, because each stream only has traffic flowing in one direction, it becomes difficult to associate the different streams with a single logical connection.

Ensuring Validity of Control Message Content The ability to add intelligence to a firewall is a big improvement. However, to ensure that the traffic is legitimate requires a large amount of expensive content checking. A connection tracking helper module can easily include basic sanity checks for different protocols. For the SIP helper, it is easy to verify that the first line of the message conforms to the protocol specifications. However, any checking beyond that adds a great deal more complexity. SIP allows for host names to be used within the protocol messages as well as IP addresses. However, the use of host names creates significant problems. There is currently no way to resolve a DNS name from within the Linux kernel. To verify that an embedded domain name is legitimate would require the kernel to interact with some form of asynchronous DNS resolver operating in user-space. Implementing the resolver and verifying that the kernel would not block while the name is being resolved is a difficult task.

The SIP call signaling is very complex. The small number of request message primitives means that they are used in numerous different manners to provide the features SIP offers. The INVITE request can be used for initiating a call, placing a call on hold, three way calling or multi-party conference calls. Therefore, ensuring that each message is syn-

tactically correct and logically valid is very difficult. The finite state diagram for all call message flows is very large. Incorporating in a helper module the capability to parse the entire SIP message and also maintaining state for a large volume of calls would require a huge amount of processing power.

Multiple INVITEs per Call The SIP call model allows for multiple INVITE messages to be generated for each call session. Features such as call hold, call forward and call park are all implemented using additional INVITE messages with different header configurations. The use of multiple INVITE messages is also a technique to perform a denial of service (DoS) on an end terminal. It can be difficult to differentiate between multiple legitimate INVITE messages and a DoS attack. Chapter 5 fully addresses the issue of detecting and responding to DoS attacks in the STEM architecture.

Incorporating NAT Functionality At the IP layer, providing Network Address Translation (NAT) functionality is not difficult. However, doing the same at the application layer is not straight forward. If the enterprise's internal network uses private addresses, all communication with external parties must utilize network address translation. There are two significant problems with providing NAT for SIP.

First, the variable nature of the header fields makes it hard to determine what is and is not applicable for being NATed. Assuming the parser could guarantee that all fields are detected, a second problem presents itself. Certain fields can either use IP addresses of DNS names to specify hosts. If private DNS names are utilized, they must first be resolved into IP addresses before NAT can occur. As mentioned earlier, it is currently not possible to do address resolution from within kernel space. Using the asynchronous resolver workaround introduces a number of timing issues.

Another issue related to NAT is encountered with the RTP streams. Two consecutive ports are required for each RTP session for the control and data streams. The existing NAT behavior does not guarantee that after the ports have been NATed they will still be

consecutive. The current Netfilter framework does not include a reservation scheme to ensure that after being NATed, the ports are still consecutive.

Dynamically Adding and Removing Pinholes The Netfilter framework provides several user-space tools to allow the static filter sets to be manipulated. However, doing these manipulations from other section of kernel space is not permitted. Therefore, adding and removing the pinholes required to allow the multiple data streams for each call would require the kernel level Protocol Parser to invoke a user-space application each time. The overhead of transferring information between user and kernel space as well as the loading and execution of a user-space application would dramatically reduce the performance of the firewall.

Does Packet Encryption Break Everything? All of the problems discussed thus far have been related to implementation details of the Protocol Parser. SIP allows for almost all of the packet contents to be encrypted including many of the header fields. The only fields that must remain in clear text are those used to route the control message to the next hop. This means the entire SDP header can be encrypted including the ports to be used by the RTP streams. Requiring the firewall to decrypt each SIP message adds a tremendous amount of overhead in addition to the cryptographic problems of key distribution and trust. There are several potential workarounds to the problem including using encapsulation or redundant headers but these are not true solutions to the problem.

Performance & Scalability

The Protocol Parsers need to be highly efficient pieces of code. However, incorporating many of the workarounds and fixes previously discussed, result in code that is anything but that. The modification of the static filters from user-space is a slow and expensive operation. In addition, if some form of aggregation of rules is not done, the rule sets will grow linearly with the number of calls and system degradation will occur at high call volumes.

The Flow Monitor must operate in real-time, which is a very daunting task given the amount of traffic it must analyze. Therefore, some form of packet sampling must be employed. The performance of the monitor will greatly depend on the speed and accuracy of the sampling algorithm.

3.5.2 Incorporating the Security Manager

The Vovida Open Communication Application Library (VOCAL) [4] was chosen to provide the SIP Proxy and Registrar / Location Server. The VOCAL suite provided an open source implementation of both servers. The VOCAL source is well laid out and commented to allow extensions to be easily incorporated. The four functional units of the Security Manager fit extremely well into the existing code base. The work on adding all the extensions necessary to implement the SM was not completed once it was determined it was not possible to build the firewall.

Chapter 4

Vulnerabilities in Converged Networks

The more complicated a system becomes, the more likely there will be flaws and vulnerabilities within it. Converged networks are no exceptions. In a standard deployment, a large number of potential vulnerabilities exist. This chapter provides an enumeration of several major classes of vulnerabilities present in a IP telephony enabled network and gives several possible attacks for each.

4.1 Information Gathering

The collection of information can be a very powerful tool. Often times an attacker will attempt to gain a large amount of information about network topologys, device configuration and traffic behavior patterns before attacking a target. The SIP protocol includes several potential avenues to allow attacks to gain valuable information about a target without introducing abnormal behavior into the system.

The SIP OPTIONS request allows an individual to query the capabilities and resources available on a remote terminal before establishing a call. In an attempt to discover a network's topology layout, the **Max-Forward** header field can be used to determine how many proxies there are between two terminals. This technique is similar to that used by the traceroute utility. A third technique can be used to blindly scan for IP telephony enabled

terminals. An IP telephony terminal receiving a CANCEL request for a session that does not exist will automatically generate a *Transaction Does Not Exist* response. By sending CANCEL requests to a large number of hosts, an attacker can determine if they are IP telephony terminals or traditional computers.

4.2 Eavesdropping

Third party data capture is a serious problem in any shared network. In converged networks, it is potentially easier for an attacker to capture their desired information. Without preserving the integrity of the control messages, it is possible for a third party to influence the path a call follows. In addition, proxies can be added to the route path to ensure all control messages are passed through a host controlled by the attacker.

In addition to having the ability to re-route calls, the information transmitted over telephone calls has the potential to contain very valuable information. Utilities exist to decode the RTP media streams in real-time and play voice back as a wave file on a third parties computer [41]. Additionally, eavesdropping can collect Dual Tone Multi-Frequency information transmitted during the call including voice mail passwords, calling card numbers, bank accounts or credit card numbers.

4.3 Connection Hijacking

The next step beyond eavesdropping on calls is taking control over the call or hijacking it. Session hijacking has been done at the transport layer in the past. Using SIP it is possible to perform session hijacking at the application level. An attacker who sends falsified response messages include *Moved Permanently*, *Moved Temporarily* or *Use Proxy* to an end terminal will be able to redirect calls to a different terminal than intended. An attacker could also issue a new INVITE request with a different call configuration or modify the

values of the SIP headers as they are in route between the two endpoints.

The control stream of a call is not the only part susceptible to hijacking. An attacker could generate a RTP media stream destined to a particular victim and use the Synchronization Source (SSRC) identifier of the target. By ensuring the sequence number and time stamps are higher and newer than the target, the victim will hear the attackers media stream unknowingly.

4.4 Denial of Service

The final category of attacks is the most destructive. Denial of service (DoS) attacks have the ability to completely remove the usefulness of a device. Almost every element in a network as well as each layer of the network stack can be the victim of a DoS attack. This section will focus on a small subset of all possible attacks. The four attacks discussed are thought to be the most destructive flood based DoS attacks in a converged network. The first three focus on attacks primarily launched from the Internet while the final one is a PSTN based attack.

• **TCP SYN Flood** A possible DoS target is the network stack on the victim machine. The TCP SYN flood is an attack which attempts to overwhelm the victims' network stack by sending a huge number of TCP SYN packets to the victim [11]. Doing this causes the victims computer to be unable to service legitimate incoming SYN packets and therefore service is denied to those trying to connect to the victim. This attack is most effective when it is directed at a machine that provides services to a large volume of clients. The SIP Proxy and web servers within an enterprise DMZ are the more likely targets.

• **SIP INVITE Flood** Converged networks add a level of targets that are not usually considered in traditional data networks: humans. A SIP INVITE flood is a DoS attack against a human. The attack is launched by establishing multiple TCP connections with

an end terminal and then generating a large number of INVITE messages. Each INVITE message must be dealt with individually by the targeted human. Generating enough of these request will overwhelm the target and either consume all of their time answering the request or they will simply ignore the telephony terminal and potentially miss legitimate calls.

A variant to this is generating a moderate number of requests to a large number of targets. This will not result in a DoS at the human level, but rather the SIP Proxy could become overwhelmed. This has the potential of causing the proxy to drop or miss request and response messages from existing calls. While the SIP protocol has re-transmission properties, the call setup time is greatly increased if they are used.

• **Malicious RTP Streams** The link connecting the enterprise network and the Internet is another potential target for attack. By increasing the volume of traffic beyond the capacity of the link, the quality of service for all data traffic is degraded. IP telephony allows for a new type of bandwidth consuming flood to be launched. The RTP packets that carry the encoded voice data are typically very small but are generated in large numbers. An attacker using a modified SIP soft-phone could create unusually large RTP packets. The extra information within the packets, if constructed properly, would either be discarded at the receiving terminal or converted into frequencies outside the audible range. The operator of the receiving terminal would not even be aware they are under attack. If enough RTP streams going to a target network were of this malicious nature, it could saturate the enterprise's Internet connection.

• **Voice Port Exhaustion** A converged network also allows for new types of DoS attacks to be executed that impact both the data and phone networks. The Media/Signal Gateway contains a finite number of voice ports. Should all of the ports be allocated, no new calls can

be initiated between the PSTN and the enterprise's LAN[1]. A new DoS attack exploits this by attempting to establish and maintain enough calls between the two networks to ensure all other call attempts are denied. This attack can come from either the internal LAN or from the PSTN if enough phones are marshaled. With the limited IP telephony deployment today, the need to communicate with PSTN terminals is vital for any enterprise deploying IP telephony. Therefore, from both a financial and productivity impact perspectives this is the most costly form of DoS attack.

[1]This can happen during normal operating conditions if the number of voice ports to handle the expected outgoing call volume has not been correctly determined

Chapter 5

Secure IP Telephony using Multi-layered Protection Framework

To ensure the architecture outlined in Chapter 3 is protected, a multi-layered framework must be developed. This chapter begins with an overview of related work done on handling and detecting denial of service attacks in traditional data and PSTN networks. A property-oriented vulnerability analysis for IP telephony is given. To ensure resource fairness, four different types of sensors to detect and control flood based DoS attacks that make up the Secure IP Telephony using Multi-layered Protection (STUMP) framework are introduced. The final section discusses some quantitative results of one sensor.

5.1 Previous Work

Protection against DoS/DDoS attacks has been a popular topic in recent years. The trend has been to focus on intrusion detection for DoS, attack response (e.g., reducing the impact of a DDoS attack instance), and, source tracing and identification. Since the scope of this work is not on attack source tracing, only related works in detection and response of DoS attacks will be described here.

Accurate, efficient, and fast detection of DoS attack instances is the first and critical step in successfully defending the target systems against various forms of DoS attacks. Wang et al [67] introduced a simplistic, yet powerful, algorithm that exploits the normal behavior of TCP traffic to detect the presence of a SYN flood attack. Their algorithm was used as a basis for the algorithms presented in this work. Also, in [35], the method of EWMA (Exponentially Weighted Moving Average) is used to detect DoS attacks against QoS in a DiffServ networking environment.

Yau et al [68] developed a scheme to include throttles on the network routes that use a leaky-bucket approach to reduce the incoming rate of traffic to targeted servers. Another approach to countering DoS attacks at the network infrastructure is the use of Pushback and Aggregate Congestion Control [36, 19, 28]. The work on DoS attacks is also not limited to only IP based networks. In [10], the authors examine media stimulated focused overloads in the PSTN.

Some existing solutions have been developed to handle specific DoS attacks on the targeted terminals such as TCP synflood. For instance, both SYN cookies [9] and SYN cache [34] are extensions to the network protocol stack in an attempt to reduce the resource consumption of each incoming SYN packet.

Yet another approach to reducing the impact of a DoS attack is from a quality of service (QoS) point of view. By limiting the amount of resources each type or aggregate of traffic can consume, the extent of a DoS attack can be severely limited. In [20], Garg and Reddy present a prototype system capable of enforcing QoS restrictions on various resources including network bandwidth, protocol state memory buffers and CPU cycles.

5.2 Property-Oriented Vulnerability Analysis for IP Telephony

To protect a complex network system, such as an IP telephony enabled enterprise network, the first step is to comprehensively analyze the potential security vulnerabilities. We classify possible attacks according to the security properties (or requirements) that these attacks are trying to violate. For IP telephony, the three main desirable security properties are: a) access control to use the IP telephony service, b) integrity and authenticity of the IP telephony signaling messages, and c) resource availability and fairness in providing IP telephony service. Other properties such as confidentiality and accountability, due to the space limitation, are out of the scope of this paper. This section contains a brief explanation of each of the properties, and illustrate possible attack scenarios against these properties. We will also describe various mechanisms to mitigate these attacks.

5.2.1 Access Control to Use the IP Telephony Services

With the deployment of wireless networks within enterprises, the probability that an unauthorized user will be able to connect to the internal LAN has greatly increased. Therefore, it is entirely feasible to assume that an attacker will be able to make telephony calls once attached to the network by any medium. To ensure that this is not possible, all outgoing calls must be made by authenticated users. Most enterprises have some form of central authentication server in their organization such as active directory, Kerberos [32], or Radius [42]. To prevent unauthorized outgoing calls, devices within the control path must be able to query the authentication server to ensure the identity of the caller.

5.2.2 Integrity and Authenticity of the IP Telephony Signaling Messages

Signaling messages in a IP telephony system should NOT be tampered and repudiated in any way, and violating this integrity and authenticity property will cause potentially serious service disruption. For instance, an attacker can maliciously impersonate others to send SIP CANCEL request messages to drop all incoming calls to a particular terminal or to cancel all outgoing calls made by an individual. Another technique is to send a BYE request message to all the terminals involved in an already established call. This results in the call being dropped and the terminals having to reestablish it. A third type of attacks on signaling messages is caused by an attacker generating illegitimate SIP response messages informing the calling terminal that the called address is no longer available.

Yet another class of attack is based on call redirection. By injecting malicious SIP response messages into an existing call control stream, an attacker can alter the servers the control stream is routed through. If desired, they can force the call to be routed through a compromised proxy. Other responses can be generated to cause the calling terminal to believe the called party has either changed locations or address. An attacker re-registering with a Registrar/Location Server by sending a SIP REGISTER request with a new URI for the target party is another attack in this class. The result is that all future incoming calls will be routed to the new URI allowing the attacker to impersonate the target.

All the above examples are related to maliciously modify or insert signaling messages, while the attacker can be either an insider or an outsider. Only being concerned about outsider attacks allows simple symmetric authentication protocols on the control channels are sufficient to eliminate all the vulnerabilities related to the properties of authentication and integrity. On the other hand, for insider threats, the situation is significantly more complicated as some of the threats can not be handled even with the most advanced public key protocols alone. In such cases, both prevention and intrusion detection mechanisms might

need to be integrated. For instance, applying the concept of "property-oriented" detection [66] allows the further separation of a main property into a hierarchy of sub-properties, and then develop sensors to detect the violations against any of the sub-properties.

5.2.3 Resource Fairness and Availability in Providing IP Telephony Services

In a IP telephony system, resources such as bandwidth must be allocated efficiently and fairly to accommodate the maximum number of callers. This property can be violated by attackers who aggressively and abusively (but maybe legally as well) obtain unnecessarily large amount of resources. In other words, the resources have been unfairly distributed among all the callers. Alternatively, the attacker can simply flood the network with large number of packets/messages such that the resources are unavailable to all the callers, including the attacker himself/herself.

Similar to the ideas of FRED (Flow-based Random Early Dropping) or ACC (Aggregate-based Congestion Control), a network device can monitor the resource usage for each data stream (a micro flow) or a small aggregate of data streams (a macro flow). By using sampling schemes [16, 13, 29], this device will be able to track the number of packets sent per flow and also monitor the size of the packets. If a flow is determined to be abusive, the device can either notify an administrator or activate response mechanisms like per aggregate rate limiting.

Dealing with flood based DoS attacks is much more complicated, especially when the floods can happen in three different levels in IP telephony: users, contact points such as SIP proxy, and network links. To further complicate the situation, the attacks can come from either the Internet or the PSTN. While these are two very different networks both topology and functional wise, there are striking similarities.

The packet switching nature of data networks allows multiple connections to share the same physical channel. Therefore, unlike in circuit switched networks, an IP telephone

terminal can receive and potentially participate in multiple calls at once. An attacker can easily overwhelm a single terminal by sending several call INVITE requests in a short period of time.

The second level of the flooding target is the internal contact points in the enterprise. For Net-to-PSTN and Net-to-Net calls, this is the SIP Proxy, and, for PSTN based calls, it is the Media/Signal Gateway (MSG). Each of these devices has a finite amount of resources which every call requires. The MSG contains a fixed number of voice ports. Every call occupies a single port for the entire duration of the call. For calls being sent through the proxy, the resource limit is not as finite as the PSTN equivalent. However, the server acting as the proxy does have a limit amount of memory and processing ability, thus sets a limit on the number of simultaneous calls it can handle. A large volume of calls could result in these resources being completely consumed and denying any further calls. It should be noted that this condition could occur under normal operation.

The third level of attack targets includes network links connecting the enterprise network to the global network. For access to the PSTN network, this is the SS7 signaling link between the MSG and the STP. These links are typically 56 kbps. The other network links connect the corporate LAN to the Internet. If enough traffic is sent into or out of the enterprise, these links can become saturated. Since the network link between the corporate LAN and the Internet carries traffic other than just IP telephony, saturating this connection will result in all services being disrupted.

5.3 Sensors for Detecting DoS Attacks

This section describes a multi-layer architecture that provides protection against flood based DoS attacks. The architecture employs four types of sensors to detect and control a) application layer flood attacks using SIP messages, b) transport layer flood attacks, c) PSTN originated flood attacks, and d) flood attacks using RTP streams. The following subsections

describe the functionality of these sensors and potential responses after a certain type of attacks is detected. The details of the detection and response algorithms are included in Section 5.4.

5.3.1 Application Layer Attack Sensor (ALAS)

The key objective of an ALAS is to detect and control DoS attacks that exploit application layer signaling and control messages, i.e., SIP messages. The target of the attack could be an individual user and/or the SIP proxy server. To detect such attacks, the ALAS must monitor each URI independently. During an observation period, the URI is extracted from INVITE and OK messages and is stored in a tracking table within the sensor. Each URI entry has a counter to track the number of INVITEs and OKs observed. At the end of the sampling period, ALAS checks if any of the URI counter exceeds a chosen threshold to decide whether an attack is detected.

Upon detecting an attack targeting a specific URI, ALAS notifies the SIP Proxy via a control message that contains a severity indicator. The SIP proxy will then initiate an "attack response" by returning *Temporarily Unavailable* or *Busy Here* messages to a fraction of incoming calls to the corresponding URI. The severity indicator in the control message determines the probability that a new incoming call will be allowed to pass through the proxy. In the worst case scenario, all calls to the URI will be blocked by the proxy. The call restrictions are only removed when ALAS instructs the proxy to do so.

Instead of placing the ALAS in front the SIP Proxy, it is possible to place it behind as shown in Figure 5.1. However, this will change the traffic characteristics seen by the sensor. During an attack, the response mechanism in the SIP proxy may be activated to reject a fraction of incoming calls. As a result, the sensor will fail to see all incoming calls to update the tracking table correctly. It is possible to inform the sensor which calls are blocked at the proxy by modifying the interaction between the proxy and sensor, but this is the scope of future work.

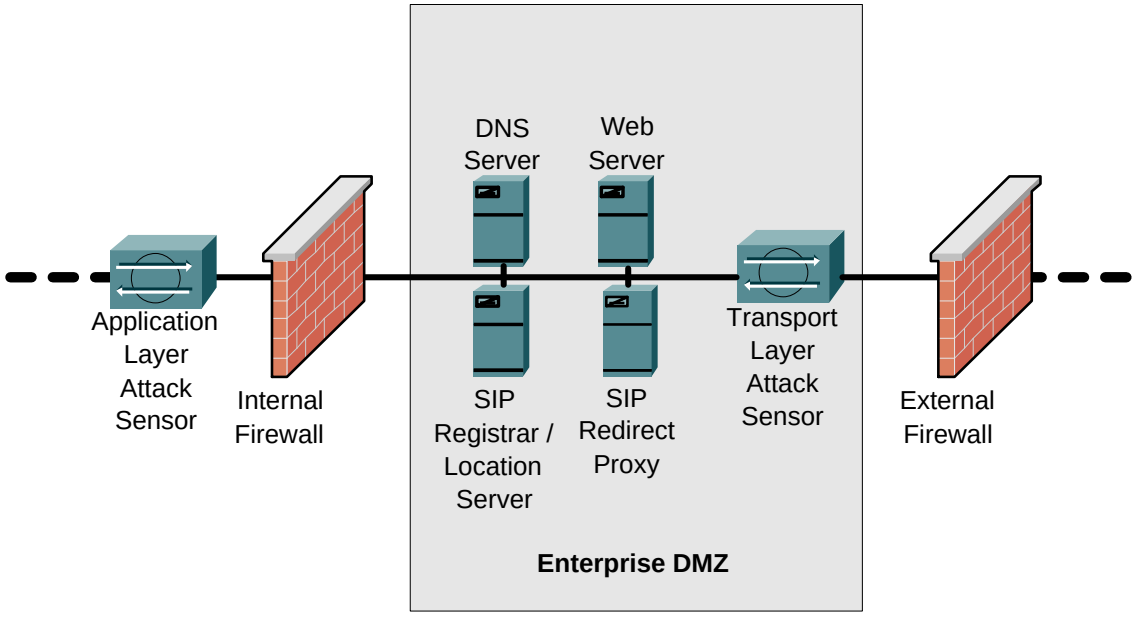


Figure 5.1: ALAS placed behind the SIP Proxy.

The overhead incurred by monitoring individual URIs is justified because it allows the response mechanism to target only those URIs affected by the attack. Using an aggregate based approach, e.g., keeping one aggregate counter for all URIs, would result in all end terminals being affected by the response mechanisms if an attack was detected.

5.3.2 Transport Layer Attack Sensor (TLAS)

A TLAS is used to monitor TCP SYN and ACK packets to identify attacks targeted at the transport/network layer. The pair of SYN and ACK packets can be used to detect an attack because of several reasons. First, the external firewall is a stateful device and will not allow ACK packets not associated with an existing connection to pass. The result of this is that an attacker cannot flood a target with a mixture of both SYN and ACK packets in an attempt to hide the attack from the TLAS since the ACKs will not traverse the firewall. Secondly, it is relatively difficult for an attacker to spoof the source address of a SYN packet and then generate a correct ACK packet because the SYN-ACK packet generated by the targeted enterprise server will be sent to the spoofed address. The attacker might be

able to view the SYN-ACK packet if they were located on the data path between the target and the spoofed address, but this situation is rare.

Using the SYN and ACK pair also allows for a short observation period. The time between the two packets is equal to the round trip time between the enterprise server and the initiating machine. In the worst case, this value would be on the order of several seconds. This close time proximity between packets allows the sensor to use a short observation period and detect an attack quickly.

The location of the TLAS within the network can be leveraged to provide protection to all machines in the DMZ. Tracking the related SYN and ACK packets at an individual connection level or end terminal is not appropriate because of the extremely large volume of connections and the lack of trustworthiness of source addresses. Furthermore, DoS attacks targeting the network layer of a device usually require a large volume of traffic. Therefore, monitoring at an aggregate level is sufficient to detect anomaly during the presence of a network attack.

The response mechanisms for a transport layer attack can be classified into three categories: end server response, firewall response and router response. At the end server SYN cache [34] or SYN cookies [9] can be used to reduce the amount of resources consumed by an incoming SYN packet. Rate limiting at the firewall can be activated to decrease the frequency of incoming SYN packets to the servers. Finally, Pushback and Aggregate Congestion Control [36, 19] can be used by upstream providers to drop offending flows before they reach an enterprise network's border.

5.3.3 ALAS for PSTN Originated Attacks

In a converged network, it is possible to launch an attack from the PSTN. It is, however, more difficult than attacks launched from the Internet, because a large number of individual phones must be marshaled and the attack must be coordinated between a large number of individuals. Nevertheless, provision must be made to detect and control such flood at-

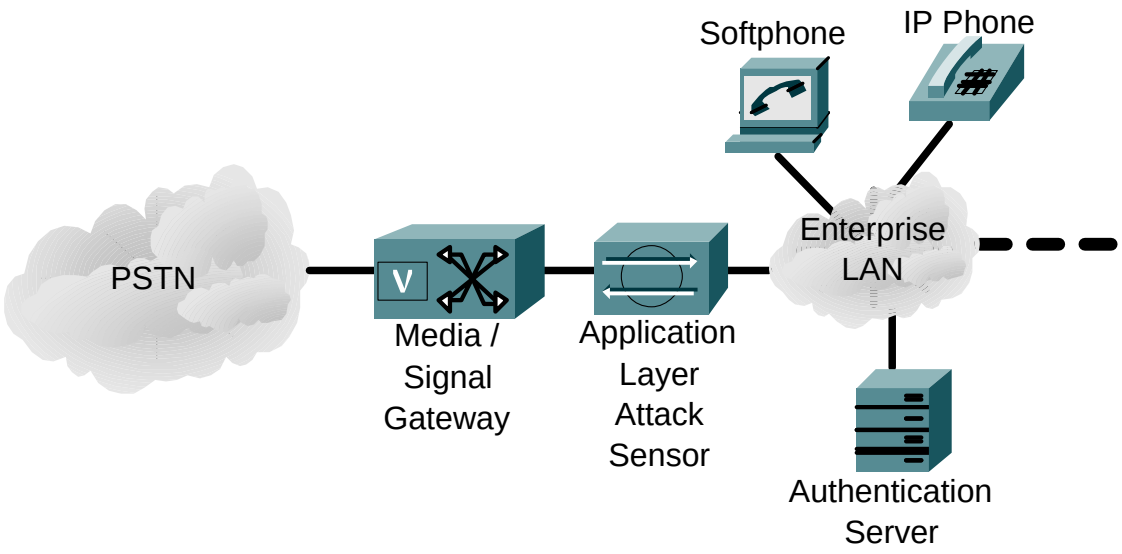


Figure 5.2: Detecting PSTN Based DoS Attacks

tacks. To detect and control such attacks, an ALAS can be deployed in series with the MSG (Figure 5.2). A sensor placed here would operate almost identically to one placed behind the SIP Proxy 5.3.1. The difference between the two deployment locations is the response mechanisms that are utilized. For PSTN based attacks, the MSG must generate Transfer Controlled (TFC) messages or Release Busy messages for the targeted E.164 numbers depending on the severity of the attack [47].

5.3.4 RTP Stream Attack Sensor (RSAS)

A RSAS is designed to detect attacks using malicious RTP streams to saturate the enterprise network. After successfully initiating a call to a victim, the attacker can either send very large RTP packets or increase the transmission rate to hoard the link capacity. RSAS can either police individual stream flow or leverage statistical techniques to detect large flows. In the first approach, a protocol parser can extract codecs and bandwidth requirements of each call upon call setup. This information is provided to RSAS. RSAS monitors the data rate for each individual RTP stream, and identify the malicious flows that

exceed its stipulated limit. Since an RTP stream in IP Telephony is typically very small (8-64 kbps), statistical marking techniques used in [16] to identify big RTP streams can be leveraged. Once a malicious stream is detected, the packets can be dropped by the firewall immediately to prevent resource depletion of legitimate calls.

However, neither of the two solutions can prevent the saturation of the link outside the firewall, and one has to rely on the upstream router to install similar RSAS and response mechanism.

5.4 Design of ALAS

Studies of TCP traffic suggests that the average session length is between 12 and 19 seconds [64]. Enterprise telephony traffic, however, lasts much longer and is typically in the order of minutes. Studies reported in [62] show that up to 10% of the call have duration over 10 minutes. This difference in session lengths imposes restrictions on the sampling schemes that can be used to detect an impending attack. Under normal IP telephony operation, the number of initiated handshakes should be very close to the number of complete handshakes within a fixed observation period. A key characteristic of application layer DoS attack is that the handshaking process will not be completed. Therefore, if the difference between the number of initiated and completed handshakes suddenly becomes very large, it is strong indication that the system is under attack. An additional benefit of using the handshakes to detect attacks is the temporal proximity of the messages. This allows for shorter sampling periods and hence lower detection time.

5.4.1 Detection Algorithm

The algorithm used in detecting the presence of an attack is based on the work presented in [67]. The correlation between the number of connection establishment attempts and the completed handshakes is similar to the relationship between connection setup and

tear-down. The difference can be modeled as a stationary, random process. The sensor is an implementation of Sequential Change Point Detection [6] scheme. In particular, the detection of an attack is accomplished by normalizing the difference with the average number of connections and applying the non-parametric cumulative sum method [8].

At the end of each observation period t_0, Δ_n is calculated to be the number of establishment attempts ($EA(n)$) minus the number of completed handshakes ($HS(n)$). To remove the dependency between the mean of Δ_n and the sample size, a normalized value X_n is calculate based on $\Delta_n/\bar{C}$ where $\bar{C}$ is the average number of connections during the observation period t_0. $\bar{C}$ is defined as:

$$\bar{C}(n) = \alpha\bar{C}(n-1) + (1-\alpha)HS(n) \tag{5.1}$$

The detection of an attack within a single observation period is based upon the expected value of X_n. Under normal operation, $E(X_n) = d \ll 1$. To make detection easy, a value o is chosen such that $o > d$ and $\bar{X}_n = X_n - o$. By shifting X_n, whenever $\bar{X}_n$ is positive it indicates the presence of an attack.

To ensure that short high volume attacks as well as longer low volume attacks are detected by the sensors, the algorithm includes a cumulative sum component. We define y_n as

$$y_n = \begin{cases} y_{n-1} + \bar{X}_n, & \text{if } (y_{n-1} + \bar{X}_n) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{5.2}$$

If y_n this value exceeds a pre-defined threshold value, T, the system is considered to be under attack.

5.4.2 Recovery Algorithm

The impact of an attack can be amplified if it takes a long time to resume normal operation once the attack has ceased. Three different recovery algorithms were considered in this study.

Linear Recovery: This algorithm is the default behavior of the detection algorithm once the attack has stopped. The value of $\bar{X}_n$ is close to $-o$ and thus y_n linearly decays to 0. If the value of y_n is large when the attack ceases and the offset, o, is small, it will require a long time for y_n to drop below the threshold T. This results in the response mechanisms to remain activated for $\frac{y_n}{o}$ minutes after the attack is over. Note that the underlying assumption is that, the sensor observes unfiltered data, i.e., the sensor is place between the SIP proxy server and the external firewall.

Exponential Recovery: In this recovery algorithm, y_n is decremented using a multiplicative factor once $\bar{X}_n < 0$. The value of y_n is calculated by:

$$y_n = \begin{cases} y_{n-1} + \bar{X}_n, & \text{if } \bar{X}_n > 0 \\ y_{n-1} - o^i, & \text{otherwise} \end{cases}$$

If $\bar{X}_n \leq 0$, the value of i is incremented after y_n is calculated. Once y_n returns to 0 or begins to increase, the value of i is reset to 1. Using this approach, the time for which the attack response mechanism remains active after the attack has ceased is $\log_o(y_n)$ minutes.

Reset after Timeout: This scheme is an extension of the linear recovery algorithm. When the value of y_n begins to drop, a timer, E, is started. The value of y_n is allowed to decay linearly until the timer expires. At the expiration of the timer, if the value of y_n is still above the threshold T, it is reset to 0. Unlike the other two approaches, by using discrete timeouts it is possible to place a fixed upper bound, E, on the time the response mechanisms will be in place after the attack has stopped.

5.5 Quantitative Analysis of ALAS

To ensure the application layer sensor operated correctly, one was placed in the front of the DMZ. This guaranteed that all traffic coming into and out of the enterprise's network would pass through the sensor. As shown in Figure 5.3, it is also possible to place a NLAS sensor in series with the ALAS.

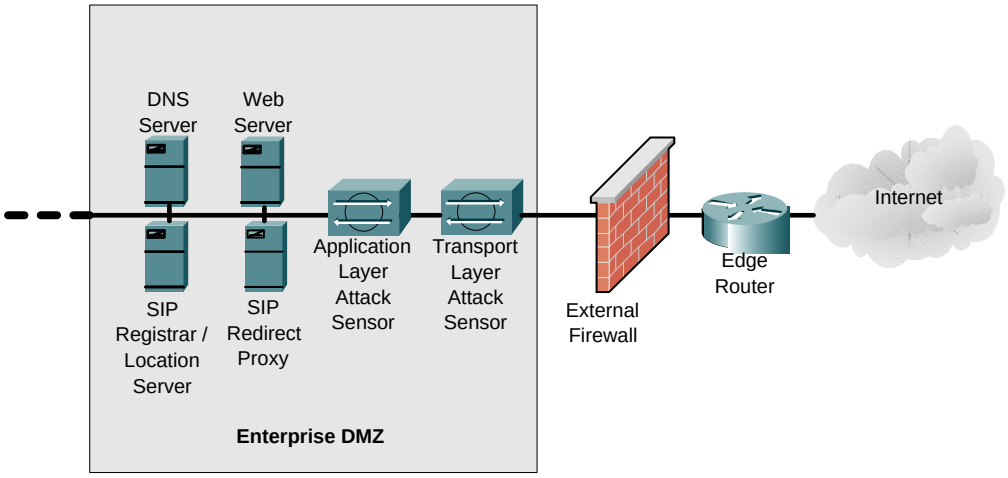


Figure 5.3: ALAS Placed within the DMZ

5.5.1 DoS Attack Models

To evaluate the performance of ALAS, the following three different DoS scenarios were considered.

Limited DoS Attack: This attack involves a single URI being targeted by one or more attackers. The volume of incoming attack calls is from a low annoyance level (of one hostile call per minute) to an overwhelming level (of 10 or more hostile calls per minute). This attack does not degrade the level of service throughout the enterprise.

Stealth DoS Attack: This attack involves one or more attackers targeting a large number of URIs. Each URI receives a very low volume of calls (e.g., one per minute or less). However, in the aggregate this results in a high usage of network resources.

Aggressive DoS Attack: This attack can be viewed as a combination of the two previous cases. In this attack one or more attackers initiate a low volume of calls to a moderate number of URIs. The impact of the attack is two fold: 1) the targeted end users were successfully disrupted from their normal operations and 2) a large amount of network resources were consumed causing other services to suffer.

5.5.2 Enterprise User Model

The user model defining the distribution of calls to different URIs is shown in Table 5.1. The majority of URIs received a very low volume of calls during the simulation period. However, there are certain addresses within an enterprise (e.g., help desk, front office, etc.) that receive a much higher volume of calls. To determine if the volume of legitimate calls affected the performance of the sensors, both high and low volume users were included in the model.

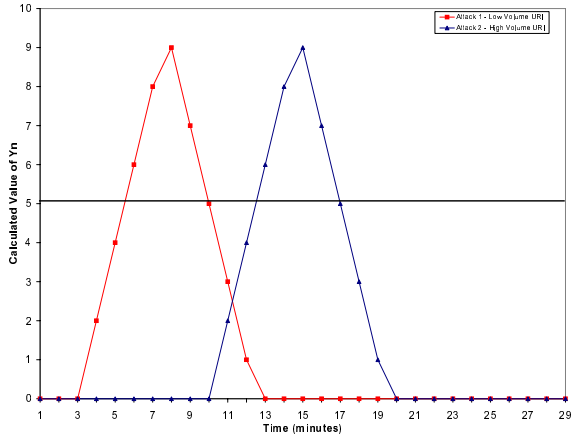
Table 5.1: Enterprise Call Distribution

Class	Calls Received During Simulation	Number of URIs
A	1	500
B	2 - 5	400
C	6 - 10	80
D	11 - 20	20

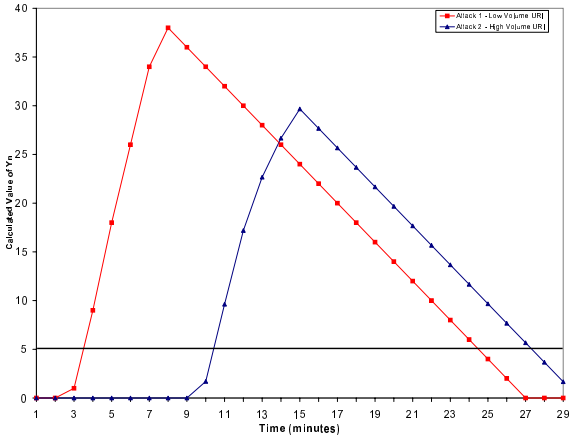
5.5.3 Simulation Parameters

Each simulation run lasted thirty minutes with the detection algorithm sampling the traffic and calculating statistics each minute. For each recovery algorithm, two simulation experiments were conducted with limited DoS attacks using 4 and 10 hostile calls per minute to a single URI. To ensure that ALAS would detect the attack regardless of the volume of legitimate calls, two URIs were targeted during each simulation. One of the URIs received 2 to 5 calls during the simulation period and the other received up to 20 calls. The attacks were each 5 minutes in length and started on the second and seventh minute of the simulation.

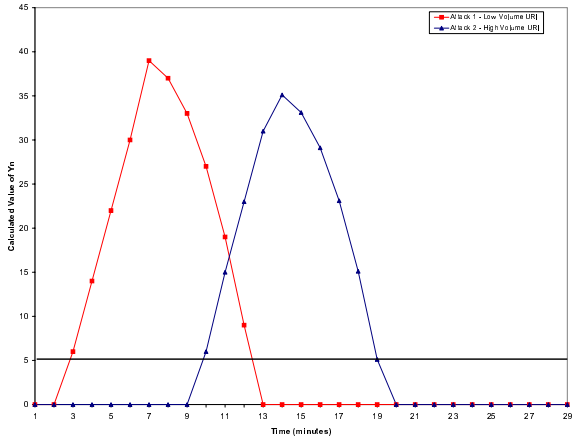
Two additional simulations used a stealth DoS attack and an aggressive DoS attack, respectively. The stealth attack targeted 200 unique URIs out of the 1000 URIs within the enterprise and generated one call a minute to each URI. The aggressive attack only used 50 unique URIs, with 3 calls minute to each target. The attacks lasted 10 minutes and began



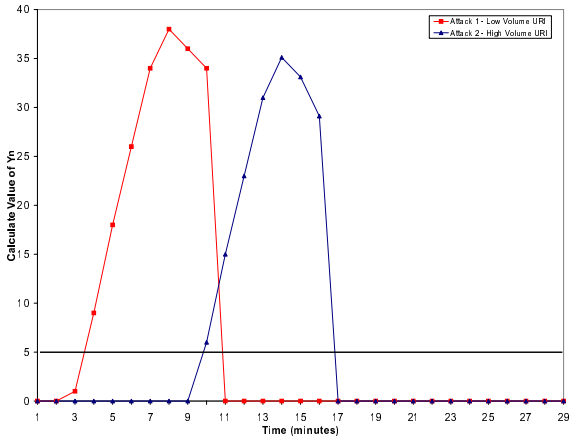
(a) Linear Recovery with 4 attack calls per minute



(b) Linear Recovery with 10 attack calls per minute



(c) Exponential Recovery with 10 attack calls per minute



(d) Reset after Timeout with 10 attack calls per minute

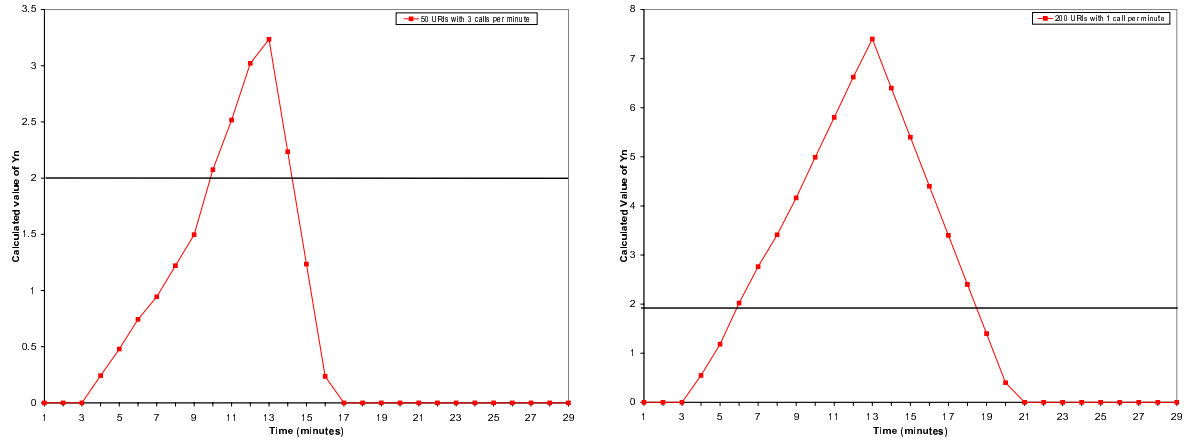
Figure 5.4: Individual URI Detection using Limited DoS ($o_{uri} = 2$ and $T_{uri} = 5$)

on the second minute of the simulation.

The offset values, o_{uri} and o_{agg}, were set to 2 and 1, respectively. The attack thresholds, T_{uri} and T_{agg}, were set to 5 and 2. The value E for the discrete timeout algorithm was set to 2.

5.5.4 Experimental Results

Two key metrics were used to determine its performance: attack detection time and system recovery time. Figure 5.4 shows the sensor's detection of a limited DoS attack



(a) Aggressive DoS Attack (b) Stealth DoS Attack

Figure 5.5: Aggregate Level Detection ($o_{agg} = 1$ and $T_{agg} = 2$)

at the individual URI level. Figure 5.5 shows the detection plots at the aggregate level for the aggressive and stealth DoS attacks. By choosing the offset and threshold values appropriately, the false alarm rate was reduced to zero for all simulations. Lowering the values would allow for more stealthier attacks to be detected, but would also increased the false alarm rate.

The attack detection times for the four DoS attacks types are shown in Table 5.2. The results in Figures 5.4 and 5.5 show that the larger the volume of attack calls, the shorter the detection time. The one result that might seem surprising is the stealth attack was detected in less time than the aggressive attack. This is because the overall call volume was greater for the particular stealth and aggressive attacks used in this study. The aggressive attack generated 150 attack calls per minute (three calls to 50 different URIs) while the stealth generated 200 attack calls per minute (one call to 200 different URIs).

Table 5.2: DoS Attack Detection Time

Attack Type	Detection Time
4 calls/min Limited DoS	4 minutes (URI level)
10 calls/min Limited DoS	2 minutes (URI level)
50 URI Aggressive DoS	6 minutes (URI level) 8 minutes (aggregate level)
200 URI Stealth DoS	4 minutes (aggregate level)

To evaluate the performance and impact of the different recovery algorithms, a limited DoS targeting a low volume URI was used. Table 5.3 shows the amount of time required from the end of the attack until the levels in the sensor dropped below the threshold. Figures 5.4b, 5.4c and 5.4d provide a graphical representation of the recovery algorithms operation. As expected, the linear recovery algorithm performance was substantially lower than the other two. For real world deployments, the increase in sensor complexity to use the exponential or reset after timeout algorithms is acceptable because of the great increase in performance. The cost of using a poor performing recovery algorithm is substantial if the response mechanisms remain activated much beyond the end of the attack.

Table 5.3: Recovery Time for Limited DoS on Low Volume URI

Attack Volume - Recovery Alg.	Recovery Time
4 calls/min - Linear Recovery	3 minutes
10 calls/min - Linear Recovery	17 minutes
10 calls/min - Exponential Recovery	6 minutes
10 calls/min - Reset after Timeout	3 minutes

To ensure that the detection algorithm works independent of the volume of legitimate traffic received by any URI, two limited attacks targeting two URIs from different user categories in Table 5.1 were considered. For users with a high volume of legitimate traffic, the value $\bar{C}(n)$ in Equation 5.1 is large. This impacts the normalization of the difference between connection attempts and establishments. The larger the value of $\bar{C}(n)$, the greater in reduction of X_n because $X_n = \Delta_n/\bar{C}(n)$. Figure 5.4b shows the impact of this normalization. The peak value of the attack on the high volume URI is 25% less than the low volume URI target.

Chapter 6

Summary

The deployment of IP telephony and other dynamic applications will allow enterprises to become more cost effective and offer a higher level of integration. However, before wide spread adoption can place several major obstacles must be overcome. It was shown that dynamic applications cannot function properly when operating over static network devices. Previous work has provided limited solutions that only address single issues. The Secure Telephony Enabled Middlebox (STEM) architecture presented in this work is a comprehensive solution to the problem.

The STEM architecture was designed to be technologically neutral, thereby allowing it to work in a diverse range of network environments. Ensuring the security of the network was a top priority in the STEM architecture. Major network-based vulnerabilities present in an IP telephony deployment were enumerated and the impacts were discussed. A complete protection framework was outlined to address all of the major classes of vulnerabilities. The Secure IP Telephony using Multi-layered Protection (STUMP) framework uses a combination of existing and new technology to ensure overall network security.

A detailed examination of DoS attacks against IP telephony enabled enterprise networks showed that a large class of attacks can only be handled by implementing dedicated sensors in enterprise network. The operation and implementation of sensors at the trans-

port and application layers were described in detail. Each of these sensors exploited a non-parametric cumulative sum algorithm to detect the presence of an attack. In addition to attack detection, the impact and performance of three different recovery algorithms were examined. A quantitative analysis using a simulated enterprise environment showed that the detection algorithm correctly identified three different types of DoS attacks and that there was differences between the different recovery algorithms.

6.1 Future Work

With the design of the architecture complete, the creation of a complete prototype network using the STEM architecture with the STUMP overlay framework overlaid is the next step. A prototype would allow experimental validation of the security and functionality of the architecture. Further simulation configurations of the two DoS sensors developed is also needed. The impact of the various parameters and their interaction with the performance of the sensors. Finally, the flow monitor concept needs further refinement before it can be implemented and tested.

Appendix A

List of Publications & Presentations

A.1 Publications

• B. Reynolds and D. Ghosal, ”STEM: Secure Telephony Enabled Middlebox”, in *IEEE Communications*, vol. 40, no. 10, Oct. 2002.

• B. Reynolds and D. Ghosal, ”Defending Against Attack in Converged Networks”, in *Proceedings of the 2002 UC Davis Student Workshop on Computing*, Davis, Oct. 2002.

• B. Reynolds and D. Ghosal, ”Secure IP Telephony using Multi-layered Protection”, to appear in *Proceeding of Network and Distributed System Security Symposium* (NDSS), San Diego, Feb. 2003.

• B. Reynolds, D. Ghosal, C. Chuah and S. F. Wu, ”Vulnerability Analysis and A Security Architecture for IP Telephony”, submitted to *Open Architectures and Network Programming* (OPENARCH), San Francisco, Apr. 2003.

A.2 Presentations

• ”Challenges and Rewards in Enterprise Deployment of IP Telephony”, Network Lab Seminar, University of California at Davis (Mar. 2002).

- ”Deploying IP Telephony in an Enterprise and the Vulnerabilities that Come With It”, Security Lab Seminar, University of California at Davis (July 2002).

Bibliography

[1] estara - push to talk. World Wide Web, `http://www.estara.com/`, 2002.

[2] The linux documentation project. World Wide Web, `http://www.tldp.org/`, 2002.

[3] Netfilter - firewalling, NAT, and packet mangling for Linux 2.4. World Wide Web, `http://www.netfilter.org/`, 2002.

[4] Vovida.org - your source for open source communication. World Wide Web, `http://www.vovida.org/`, 2002.

[5] M. Barnes. Internet draft: MIDCOM protocol evaluation, May 2002. Work in Progress.

[6] M. Basseville and I. V. Nikiforov. *Detection of Abrupt Changes: Theory and Application*. Prentice Hall, 1993.

[7] Baugher, McGrew, Oran, Blom, Carrara, Naslund, and Norrman. Internet draft: The secure real time transport protocol, May 2002. Work in Progress.

[8] B. E. Brodsky and B. S. Darkhovsky. *Nonparametric Methods in Changepoint Problems*. Kluwer Academic Publishers, 1993.

[9] Bronzesoft.org. SYN cookies firewall. World Wide Web, `http://www.bronzesoft.org/projects/scfw`, 2002.

[10] J. Burns and D. Ghosal. Design and analysis of a new algorithm for automatic detection and control of media stimulated focused overloads. In *Proceedings of International Teletraffic Congress*, Edinburgh, June 1997.

[11] CERT Coordination Center. CERT Advisory CA-1996-21: TCP SYN flooding and IP spoofing attacks, November 2000.

[12] William Cheswick and Steven Bellovin. *Firewalls and Internet Security*. Addison Wesley Longman, Inc., New York, NY, 1st edition, 1994.

[13] K. Claffy, G. Polyzos, and H. Braun. Application of sampling methodologies to network traffic characterization. In *Proceedings of ACM SIGCOMM*, San Francisco, September 1993.

[14] R. Daniel and M. Mealling. RFC 2168: Resolution of Uniform Resource Identifiers using the Domain Name System, June 1997.

[15] T. Dierks and C. Allen. RFC 2246: The TLS Protocol, January 1999.

[16] C. Estan and G. Varghese. New directions in traffic measurement and accounting. In *Proceedings of ACM SIGCOMM*, Pittsburg, August 2002.

[17] P. Faltstrom. RFC 2916: E.164 number and dns, September 2000.

[18] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners Lee. RFC 2068: Hypertext Transfer Protocol – HTTP/1.1, January 1997.

[19] Sally Floyd and Van Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, August 1993.

[20] A. Garg and A. Reddy. Mitigation of dos attacks through qos regulation. In *Proceedings of IEEE International Workshop on Quality of Service (IWQoS)*, Miami Beach, May 2002.

[21] ITU-T Recommendation H.323. Visual telephone systems and equipment for local area networks which provide a non-guaranteed quality of service, May 1996.

[22] M. Handley and V. Jacobson. RFC 2327: SDP: Session Description Protocol, April 1998.

[23] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg. RFC 2543: SIP: Session Initiation Protocol, March 1999.

[24] IEEE. Middlebox communication (midcom) charter. World Wide Web, `http://www.ietf.org/html.charters/midcom-charter.html`, 2002.

[25] IEEE. Session initiation protocol investigation (sipping) charter. World Wide Web, `http://www.ietf.org/html.charters/sipping-charter.html`, 2002.

[26] IEEE. Session initiation protocol (sip) charter. World Wide Web, `http://www.ietf.org/html.charters/sip-charter.html`, 2002.

[27] University of Southern California Information Sciences Institute. RFC 791: Internet Protocol, September 1981.

[28] J. Ioannidis and S. Bellovin. Implementing pushback: Router-based defense against DDoS attacks. In *Proceedings of Network and Distributed System Security Symposium*, San Diego, February 2002.

[29] J. Jedwab, P. Phall, and B. Pinna. Traffic estimation for the largest sources on a network, using packet sampling with limited storage. Technical Report 35, Hewlett Packard Labs, March 1992.

[30] A. Johnston, S. Donovan, R. Sparks, C. Cunningham, D. Willis, J. Rosenberg, K. Summers, and H. Schulzrinne. Internet draft: SIP call flow examples, April 2002. Work in Progress.

[31] A. Karim. H.323 and associated protocols, 2000.

[32] J. Kohl and C. Neuman. RFC 1510: The Kerberos Network Authentication Service (V5), September 1993.

[33] J. Kuthan. Internet telephony traversal across decomposed firewalls and NATs. In *Proceedings of the 2nd IP Telephony Workshop*, New York, April 2001.

[34] J. Lemon. Resisting SYN flood DoS attacks with a SYN cache. In *Proceedings of USENIX BSDCon 2002*, San Francisco, February 2002.

[35] V. Mahadig. Detection of denial-of-QoS attacks based on chi square statistic and ewma control charts. world Wide Web, `http://arqos.csc.ncsu.edu/papers/2002-02-usenixsec-diffservattack.pdf`, February 2002.

[36] R. Mahajan, S. Bellovin, S. Floyd, J. Vern, and P. Scott. Controlling high bandwidth aggregates in the network. Technical Report 1, University of California, Berkeley - International Computer Science Institute, February 2001.

[37] C. Perkins. RFC 2002: IP Mobility Support, October 1999.

[38] J. Peterson. Internet draft: A privacy mechanism for the session initiation protocol, May 2002. Work in Progress.

[39] J. Postel. RFC 768: User Datagram Protocol, August 1980.

[40] J. Postel. RFC 793: Transmission Control Protocol, September 1981.

[41] N. Provos. Vomit - voice over misconfigured internet telephones. World Wide Web, `http://vomit.xtdnet.nl/`, 2001.

[42] C. Rigney, S. Willens, A. Rubens, and W. Simpson. RFC 2865: Remote Authentication Dial in User Service (RADIUS), June 2000.

[43] U. Roedig, R. Ackermann, and R. Steinmetz. Evaluating and improving firewalls for IP-telephony environments. In *Proceedings of the 1st IP Telephony Workshop*, Berlin, April 2000.

[44] J. Rosenber, J. Weinberger, and H. Schulzrinne. Internet draft: SIP extensions for NAT traversal, November 2001. Work in Progress.

[45] J. Rosenberg and H. Schulzrinne. RFC 2871: A Framework for Telephony Routing over IP, June 2000. Status: Informational.

[46] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. Internet draft: SIP: Session initiation protocol, February 2002. Work in Progress.

[47] Travis Russell. *Signaling System 7*. McGraw-Hill, New York, NY, 3rd edition, 2000.

[48] Travis Russell. *Telecommunications Protocols*. McGraw-Hill, New York, NY, 2nd edition, 2000.

[49] E. Schooler. A multicast user directory service for synchronous rendezvous. Master's thesis, Department of Computer Science, California Institute of Techonology, August 1996.

[50] H. Schulzrinne, S. Casnet, R. Frederick, and V. Jacobson. RFC 1889: RTP: A Transport Protocol for Real-Time Applications, January 1996.

[51] H. Schulzrinne and S. Petrack. RFC 2833: RTP payload for DTMF digits, telephony tones and telephony signals, May 2000.

[52] H. Schulzrinne, A. Rao, and R. Lanphier. Internet draft: Real time streaming protocol, February 2002. Work in Progress.

[53] Henning Schulzrinne and Jonathan Rosenberg. A Comparison of SIP and H.323 for Internet Telephony.

[54] Henning Schulzrinne and Jonathan Rosenberg. Signaling for Internet Telephony, 1998.

[55] Henning Schulzrinne and Jonathan Rosenberg. Internet Telephony: Architecture and Protocols - an IETF Perspective. *Computer Networks*, 31(3):237–255, 1999.

[56] P. Srisuresh, J. Kuthan, J. Rosenberg, A. Molitor, and A. Rayan. Internet Draft: Middlebox communication architecture and framework, December 2001. Work in Progress.

[57] P. Srisuresh, J. Kuthan, J. Rosenberg, A. Molitor, and A. Rayhan. Internet draft: Middlebox communicationarchitecture and framework, February 2002. Work in Progress.

[58] A. Stephens and P. Cordell. SIP and H.323 - interworking VoIP networks, 2001.

[59] Richard Stevens. *TCP/IP Illustrated Volume 1: The Protocols*, volume 1. Addison Wesley Longman, Inc., Reading, MS, 1st edition, 1994.

[60] R. Swale, P. Mart, P. Sijben, S. Brim, and M. Shore. Internet Draft: Middlebox communciations protocol requirements, November 2001. Work in Progress.

[61] R. Swale, P. Mart, P. Sijben, S. Brim, and M. Shore. Internet draft: Middlebox communications protocol requirements, November 2001. Work in Progress.

[62] Telecost. Enterprise call durations distributions. World Wide Web, `http://www.telecost.co.uk/pages/OnCallDurations.htm`, 2002.

[63] R. Thayer, N. Doraswamy, and R. Glenn. RFC 2411: IP Security Document Roadmap, November 1998.

[64] K. Thompson, G. J. Miller, and R. Wilder. Wide-area internet traffic patterns and characteristics. *IEEE Network*, 11(6), December 1997.

[65] John van Bosse. *Signaling in Telecommunication Networks*. John Wiley & Sons, New York, NY, 1st edition, 1998.

[66] F. Wang, F. Gong, and F. S. Wu. Property oriented fault detection for link state based routing protocols. In *Proceedings of IEEE International Conference on Compter Communications and Networks*, Las Vegas, October 2000.

[67] H. Wang, D. Zhang, and K. Shin. Detecting SYN flooding attacks. In *Proceedings of IEEE INFOCOM 2002*, New York, June 2002.

[68] D. Yau, J. Lui, and F. Liang. Defending against distributed denial-of-service attacks with max-min fair server-centric router throttles. In *Proceedings of IEEE International Workshop on Quality of Service (IWQoS)*, Miami Beach, May 2002.

[69] W. Yeong, T. Howes, and S. Kille. RFC 1777: Lightweight Directory Access Protocol, March 1995.