Global Information Assurance Certification Paper

Bill Olson

GSEC Practical Version 1.4b

September 30, 2003

SIP – What is it and what its deployment could mean to your current infrastructure.

Abstract:

SIP (Session Initiation Protocol) is everywhere! Well maybe not everywhere, but its deployment is gaining in the market. With senior management asking IT staffs to lower costs by leveraging their current internet connections, add more content types and the need to maintain an always available status, people will be likely to turn to SIP to provide a solution.

This paper will explain what SIP is, how it works, review Network Address Translation and how SIP reacts to being NAT'ed. Lastly, it will explain what deployment options exist and what they mean to overall security of your organization.

What is SIP*?*

Session Initiation Protocol (SIP) is a text-based protocol, similar to HTTP and SMTP, for initiating interactive communication sessions between users. Such sessions include voice, video, chat, interactive games, and virtual reality. It's was designed to leverage other protocols like RTP (RFC 1889), SDP (RFC 2327) RTSP (2326) and RTP/AVP (RFC 1980). Simply put SIP is way to initiate communication between two or more parties.

The communication capabilities of SIP are almost endless, using the current technology it possible for Alice to converse with Bob regardless of his location. Using a SIP URL (Uniform Resource locator) Alice can "dial" Bob" most anywhere. This can be from most any device to most any device, meaning Bob can be in his office using a laptop while Alice is in her car and using her PDA. Their call can start with only voice to voice then add to 2-way video and even add in a third party to play an Internet enabled game. Using the Internet to complete all of the above eliminates the need for a separate voice or PSTN line.

SIP does not assure Quality of Service (QOS). SIP is not designed to handle large data file transfers and does not negotiate what protocols will be used or detail how the communication will take place. SIP simply makes a best effort attempt to locate the requested party and delivers an additional protocol SDP (Session Description Protocol). SDP is the "how" protocol and carries the details of the communication request.

As mentioned above these requests can come from many different types of SIP enabled devices. This can be a SIP enabled hardware phone such as Pingtel's Expresa phone (http://www.pingtel.com/pr_xpressa.jsp) or a soft phone (http://www.xten.com/). The latest MSN messenger is using SIP to deliver pieces of communication. However this paper would be incomplete if it did not mention that each of these solutions may (and do) differ in their delivery and format communication requests. It is beyond the scope of this paper to detail these differences. This paper will focus on several major vendors' attempts to deliver the communications with Pingtel being the main focus.

Figure 1.1 details the how the connection is made between the Alice and Bob (From RFC 3261).

The original SIP RFC was 2543 and was published in March of 1999. Since that time a new RFC 3261 has superseded it. In addition, there are at the time of this writing, there are some thirty RFC's and scores more drafts relating to the SIP protocol. Each of these RFC's builds on or leverages the current SIP implementation to deliver richer and more dynamic services.

There is even talk of SIP enabling household appliances. That would be a hacker's dream to be able to mess with someone's microwave so the popcorn was either always under-popped or over-popped or always melt ice cream. Where this could get very interesting might be a refrigerator that sends a message when you are out of milk.

How does SIP work

The SIP protocol is responsible for session establishment, modification and termination. Calls can be created only when the receiving party is willing to accept the call and if they both can agree on how the call will be established. SIP also can use either UDP or TCP to send its messages. With UDP currently being the most popular.

Note: There are many books and whitepapers that detail exactly all of the types of connections and the interaction with PBXs and regular PSTN lines. Since this paper is focusing on the interaction of NATing firewalls and SIP the focus will remain on the initiation of a call, as this that is where a significant amount of problems occur. See the references page for a list of books.

To understand SIP, we need to review a few terms that will help illustrate how communication is established and where problem points will exist.

The first term is **User Agent** (UA). A UA can be many different things; simply put a UA is what interacts with the user. When a user wants to make a call they launch a program that has a UA in it. The user then interacts with that program to format/dial a call. The receiver of the call also has UA when its UA receives a properly formatted message it may pop up a window indicating a request for communication. One thing to remember is a UA can be autonomous, like a SIP answering machine. Calls received by this type of UA would not need a user to react in order to complete a call. One other point about UAs, they could be highly

aware of their surroundings. For example, when a voice call is received by a UA it is intelligent enough to forward the media stream (via the SDP information) to an application that handles voice. The same could hold true with video games and streaming video.

The second term is **Redirect Server**. These servers help locate SIP users by providing alternative addresses where a user might be. A big feature of SIP is user mobility. For example, since Bob is an outside salesman he is rarely in the same place twice. A redirect server knows this can inform the call initiator of Bob's current location. Redirect servers also can handle group addressing; an example would be an 800 number that links up a call center staff. A Redirect Server can send a broadcast and or a directed message to several different UAs indicating an incoming call.

The third term is **Proxy Server**. There are several types of proxies, but in each instance the proxy is handling the communication on behalf of the users UA. This means that when locating someone, the user only needs to know a single address, and the proxy server can forward these request between servers and even domains.

A possible advantage would be in a corporation with several organizational units. When Alice tries to call Bob@company.com, a proxy server handles request and pass it on to Bob's real address, bob@sales.company.com. Again the key here is a single address is all that needs to be published.

Another use for some proxies is SIP security. There are several built-in components, including encryption and authentication. The SIP messages can be encrypted to prevent information leakage, such as user location or detailed information about the internal network. Authentication is a mechanism that can ensure those only known/trusted users can gain to access the system.

The fourth term is **Registrar Servers**. These servers handle the incoming registrations of the UA that it services. Registering with a Registrar Server is how someone can locate other users. When a user boots their phone it's programmed with information about what Registrar Server it should send its info. This registration process is what allows for people to roam and yet still be located.

Many of these components can live on one server or they can be spread across several servers. The main thing to remember is that the call initiator does not need to know much more than a given users SIP address in order to locate them. The system has a built-in process that can locate any registered user.

SIP addresses are URLs that resemble email addresses. Alice's address could look something like SIP:alice@company.com. DNS entries handle how the caller is located with additional records. These SRV records are based on the RFC2052. These DNS requests are based on protocol and resolve address much like MX records. Below is an exam of what the records look like;

```
_sip._udp IN SRV 1 3 5060  sipxchange
_sip._tcp  IN SRV  1  3 5060 sipxchange
```

Using this information, a UA or proxy server can locate Alice and forward her the call INVITE message.

The core of the SIP specification defines six types of SIP requests; each with a different purpose. The first line of a SIP request contains the call method. This information is what denotes the purpose of the message.

INVITES are a request/response type message, similar to an HTTP request. They use the well-known "three-way handshake" to create a session between two parties. However there is a slight twist, in that there are four steps to this handshake.

Alice wants to call Bob. She dials Bob URL thus sending an INVITE message to Bob. Bob is located (how is not critical just yet) and his UA begins to process the message. Bob's UA sends back a message indicating his has received the request. This message indicates to Alice that Bob has been located and his phone is ringing.

This process is analogous to picking up a regular phone, hearing a dial tone and dialing someone's phone number. Once the call is processed by the central switch, it is passed to the receiving party's phone, which begins to ring; the call initiator also hears a ring indicating the call is going through. *Note*: that either in the traditional call or the SIP example, the call has not been established as of yet, the parties have only been located.

In the case of the SIP call, Bob's has the caller ID feature in his SIP phone and decides that he will accept Alice's call. Bob lifts the handset on his SIP enabled phone which sends a second message to Alice. This second message confirms that he is willing to accept the call and has agreed upon the proposed communication method described in the INVITE.

The call is not complete as Alice's UA needs to send a final packet called and **ACK**. It is this final packet that really makes SIP a powerful protocol.

Consider a call coming into a Redirect Server, the server would receive a call and confirm that fact to caller by beginning to ring. However the Redirect Server is broadcasting this message to 100 different users in a large call center. When one of the users answers the call, that individual UA will send the ACK. It is this ACK that completes a call and carries the final information to call initiator.

Since SIP is a clear text protocol it is easy to peer inside to see exactly what is going on. Below is a detailed description of call being completed:

Below is an INVITE message. All of the information is in the clear. This allows multiple vendors and different versions of UAs to read messages and begin processing them without the need for special software or user controls.

Note: All messages in the paper have been sanitized and in some case modified for effect.

INVITE sip:Bob@company.com SIP/2.0
UDP SIP User Agent sent message:

----Remote Host:sipxchange.company.com---- Port: 5060----
From: "Alice” <sip:200@company.com>;tag=1c23354
To: “Bob” <Bob@company.com>
Call-Id: call-1062902085-96@192.168.100.112
Cseq: 1 INVITE
Contact: "Alice"<sip:200@192.168.100.112>
Content-Type: application/sdp
Content-Length: 312
Accept-Language: en
Allow: INVITE, ACK, CANCEL, BYE, REFER, OPTIONS, NOTIFY, REGISTER, SUBSCRIBE
Supported: sip-cc, sip-cc-01, timer, replaces
User-Agent: Pingtel/2.1.10 (VxWorks)
Date: Sun, 07 Sep 2003 02:34:45 GMT
Via: SIP/2.0/UDP 192.168.100.112
v=0
o=Pingtel 5 5 IN IP4 192.168.100.112
s=phone-call
c=IN IP4 192.168.100.112
t=0 0
m=audio 8776 RTP/AVP 96 97 0 8 18 98
a=rtpmap:96 eg711u/8000/1
a=rtpmap:97 eg711a/8000/1
a=rtpmap:0 pcmu/8000/1
a=rtpmap:8 pcma/8000/1
a=rtpmap:18 g729/8000/1
a=fmtp:18 annexb=no
a=rtpmap:98 telephone-event/8000/1
----------END----------

The **To:** and **From:** headers indicate this a call from Alice to Bob. These can indicate the fully qualified name of each user or be formatted with IP addresses (e.g. Bob@10.10.10.1). You can also see that this is an UDP packet and will be using port 5060 to establish this call. In the SDP description it also indicates the port number that it will be using during the call. For a more detailed description of the headers see RFC 3621.

The next packet comes from Bob, who’s receiving the call or in this case the INVITE. His response to Alice comes directly from his UA. It reads the message and responds back to Alice with the following:

SIP/2.0 **180 Ringing**
UDP SIP User Agent sent message:
----Remote Host:sipxchange.company.com---- Port: 5060----
From: "Alice” <sip:200@company.com>;tag=1c23354
To: “Bob” <Bob@company.com>
Call-Id: call-1062902085-96@192.168.100.112
Cseq: 1 INVITE

Via: SIP/2.0/UDP
xxx.xxx.120.210:5060;branch=z9hG4bKe91bd066b1837c79acc840e865
372a94
Via: SIP/2.0/UDP
xxx.xxx.120.210;branch=z9hG4bK9817359afa4d98ddbba36b2ce229aff2
Via: SIP/2.0/UDP 192.168.100.112
Contact: sip:Bob@192.168.100.54
User-Agent: Pingtel/2.1.10 (VxWorks)
Date: Sun, 07 Sep 2003 02:34:45 GMT
Content-Length: 0
----------END----------

This packet is routed back to Alice following the **VIA** headers from the INVITE. This message, as do all SIP messages have a message number, with ringing being **180**.

This is the first place things can go very wrong. What if the VIA headers only contained Alice's private address, being non-routable, Alice would never receive any indication that Bob has received the call. For our example, let's assume that is not an issue and proceed with the call initiation.

Once Alice's UA receives the 180 ringing message her phone displays a message indicating Bob's phone is ringing.

Once Bob decides he want to speak with Alice and lifts his handset his UA sends a second packet. This packet is message type 200 and indicates that Bob is willing to accept the call and agreed on using the following media methods to establish communication.

SIP/2.0 200 OK
UDP SIP User Agent sent message:
----Remote Host:sipxchange.company.com---- Port: 5060----
From: "Alice" <sip:200@company.com>;tag=1c23354
To: "Bob" <Bob@company.com>
Call-Id: call-1062902085-96@192.168.100.112
Cseq: 1 INVITE
Content-Type: application/sdp
Content-Length: 308
Via: SIP/2.0/UDP
x.x.120.210:5060;branch=z9hG4bKe91bd066b1837c79acc840e865372
a94
Via: SIP/2.0/UDP
x.x.120.210;branch=z9hG4bK9817359afa4d98ddbba36b2ce229aff2
Via: SIP/2.0/UDP 192.168.100.112
Record-Route:
<sip:xxx.xxx.120.210:5060;lr;a;t=1c23354;s=daa7370949ca39f006ee33c62e8ea
4e7>

```
Contact: sip:Bob@192.168.100.54
Allow: INVITE, ACK, CANCEL, BYE, REFER, OPTIONS, NOTIFY, REGISTER,
SUBSCRIBE
User-Agent: Pingtel/2.1.10 (VxWorks)
Date: Sun, 07 Sep 2003 02:34:50 GMT

v=0
o=Pingtel 5 5 IN IP4 192.168.100.54
s=phone-call
c=IN IP4 192.168.100.54
t=0 0
m=audio 8770 RTP/AVP 96 97 0 8 18 98
a=rtpmap:96 eg711u/8000/1
a=rtpmap:97 eg711a/8000/1
a=rtpmap:0 pcmu/8000/1
a=rtpmap:8 pcma/8000/1
a=rtpmap:18 g729/8000/1
a=fmtp:18 annexb=no
a=rtpmap:98 telephone-event/8000/1
```

Again this could be a problem spot. For example, if Alice's address is un-reachable this response will go nowhere and eventually time out. Assuming that the address is reachable, Alice's UA will see message type 200 coming in and respond with a final ACK. Unless the properties/features of this conversation change, this will be the final SIP packet processed as the conversation will continue over the agreed upon methods setup by this SIP exchange.

There are several other SIP methods that are used to during connections; these include OPTIONS, REGISTER, CANCEL and BYE. The **OPTIONS** method is a way of requesting information from a server and or UA. Letting the user know what features the server or the user can provide or accept. **REGISTER** method is used by a UA to register itself with a server. Calls can be ended before they begin with the **CANCEL** method and when both parties are finished the **BYE** method terminates a call.

The above explanation assumes that security controls like firewalls and NATing devices were not present or that all communication was happening on the private side. Since that is not a real world scenario, the question is how these messages would react to being behind a firewall and being NAT'ed.

NAT

In order to understand the problems with SIP and NAT, we need to review the implications that NAT has on all end-to-end communications.

The Network Address Translator (NAT) was born out of the need to resolve IP address depletion and provide scaling in routing. At the time, there was a clear

indication that there would not be enough IP addresses. It was designed only to be a short-term solution and to supplement the technology Classless InterDomain Routing -CIDR RFCs 1716. So NAT, the concept of re-using addresses was developed.

The good news is hundreds of people can now access devices on the Internet while using a single address. The bad news is hiding or translating the original and possible source address, removed the direct end-to-end communication between the two devices. This complicates and in some cases makes impossible several types of communication.

In order to fully understand NAT and the effects it has on end-to-end communication we need a basic understanding how IP connections work.

There are 65535 ports that are used for communication, the first 1023 are predefined or privileged ports and are used for well-known services like SMTP and FTP. When you type ftp.company.com, your operating system picks a random port above 1023 and assigns it to this session. While packet leaves the OS on a random port it is going the defined port for FTP. When the server ftp.company.com replies it does so on source port used by the client. This allows the FTP server to keep track of all the sessions it has going on, since no two sessions can use the source IP and the same source port.

As mentioned above there is a shortage of IP addresses. Using RFC 1918 as guide, many organizations have chosen to use private address for their internal networks. These internal addresses are not routable on the Internet, meaning if a packet were to somehow end up on the Internet it would die, never reaching its destination. Since this would make reaching any outside site impossible a border router or NATing firewall steps in and modifies the packet, allowing the user to reach the destination.

The firewall or router modifies or rewrites the packet by striping the source IP and the source port and insert a routable source IP and assigns port from a connection table. The packet is then forwarded on to original source IP address and port. When the device replies it sends the packet back to routable IP address of the NATing device. The device checks its connection table for a match, re-writes the packet again this time striping out the routable IP of the NAT device and inserting the original private IP and port. The internal device processes the packet as if it was talking directly ftp.server.com.

There are many issues with NAT and port translation, in fact an entire RFC was written that details out most of these issues. RFC 2993, the Architectural Implications of NAT. Additionally there are four different types of NAT, full cone, port restricted cone, restricted cone and symmetric. The exact differences of each are beyond the scope of this paper, however big thing to remember is the original packet is modified in someway.

Let's focus on how SIP works with NAT

Take this scenario; an organization has a firewall with three interfaces. The interfaces are private, public and a DMZ. The SMTP gateway is in the DMZ has a public IP address; the PC/Servers are on the private side and use RFC1918 addresses. Users and servers requiring access the Internet use hide-NAT. The firewall rules only allow incoming mail and define a specific set of services allowed outbound.

The organization would like to add VoIP using SIP. Most enterprise SIP deployments will include at least one SIP proxy. The proxy will most likely act as the call initiator and like an email server it will handle all the routing of all incoming calls. Reviewing the establishment of a SIP based call there are two parts of a SIP call the SIP requests and responses (the ringing) and the RTP (the voice or media stream).

The first question should be architecture, where will each of the components fit in the above scenario. For people to be able to reach your users the SIP proxy will need to be accessible from anywhere. This is much like a SMTP server. A special "srv" type entry will need to be made in the organizations DNS records. These entries will allow for the resolution of the following address "sip:someone@xyzcompnay.com".

You have three choices, locate the SIP proxy in the DMZ, using a public address, locate it on the private segment, using Static NAT to publish to the outside and final option is to locate the proxy outside the firewall forgoing NAT and a lot of security controls. The phones could also go in the DMZ, however the level of practicality must be examined. In a large deployment the possibility of having 100 or more addresses available is generally not realistic.

One final architectural decision to consider is what protocol to use UDP or TCP. This decision will be based on the hardware/software vendor you are working with and their compatibility with UDP and TCP. While some vendors are just adopting TCP most vendors have a more robust service offering based on UDP. Our scenario will use the UDP protocol.

In addition to getting past the firewall and NAT problems we need to ensure the solution focuses on several fundamental security issues, Call Hijacking, Denial of Service and authentication and integrity.

In our scenario, we will locate the SIP proxy in the DMZ using a public IP and locate each of the phones on the private segment. We will then modify the DNS records internally and externally, adding the "srv" records that enable the resolution of the domain.

Locating the SIP server behind the firewall will also the use of current security controls to monitor and control traffic. Our firewall and network controls will ensure the proper flow of traffic preventing DOS. The SIP proxy and phones will also be set up base on the vendor specifications controlling authentication and user integrity.

The firewall rules will also require some modifications; the SIP protocol will need to be allowed into the SIP proxy. SIP traffic to the DMZ from the private side will need to be allowed as will the reverse. Where things get really sticky is dealing with RTP traffic.

Once a call is been negotiated and accepted the media stream (the voice) will need to pass directly between the two end points. This could mean opening lots of ports on the firewall, allowing traffic to pass directly into private network where the phones are located.

We will focus on a user on the inside deciding to call someone on the outside using SIP. The UAs (User Agent) are programmed to forward all calls to the SIP proxy for routing. With the external address as the destination the request and the SIP port of 5060, the UA sends the packet to the SIP proxy. The SIP proxy uses pre-programmed rules to process the call. Since this is not destined for the internal network, the proxy uses DNS to resolve the domain of the person being called. The SIP server also modifies the packets source address to use its own.

This modification is required in our scenario for two reasons; the phone's IP address is not routable and therefore the response would never be received. Secondly modifying the source IP will ensure that any response from the called party can be tracked through the proxy.

When the callee answers the phone, the RTP portion of the call is directed directly to the call initiator taking the proxy out of the loop.

The SIP messages will generally pass through NAT to a proxy without too much trouble, the key being the messages are returned to the call initiator using the same source port the SIP server received the request on. For example if the UA sent a packet with the source port of 17000 and a destination port or 5060 to the SIP server, the server will need to reply to 17000 not to the SIP port of 5060. The IP address should be a constant at this point since the firewall is connected to all three networks routing and NAT are generally not problematic.

The RTP media steam packets do not have a chance. The SIP message body contains the information about the two end points. The end point clients fill in the information based on what they know about themselves. The client sitting behind a NAT device only knows its internal IP address and port. That is added to the SIP message body under the SDP heading. When the destination end point replies it will use the SDP information. Since the IP address of the originating endpoint is RFC1918 the packet will be dropped at the router.

Take RTP out of the equation for a second, he two end points are still not talking directly to each other while negotiating the call. At least one proxy is inline and if the communication is going on between two large companies, there very well might be a proxy on the far end. To negotiate these proxies the SIP messages have built-in clues needed for each SIP server along the way to forward the messages along. The **VIA:** and **record-route** headers will allow either endpoint to trace their way back and forth even if there are several proxies in the path.

Examining a SIP Invite message we can see the VIA header includes the address of the SIP proxy that is handling the initiation of this call. This is indicated in green. The problem is noted in red, the RTP packets only have the address on the initiating end point. In this call Bob is calling Alice.

UDP SIP User Agent sent message:
----Remote Host:sipxchange.company.com---- Port: 5060----
INVITE sip:Alice@xxx.xxx.208.116 SIP/2.0 *(modified to protect innocent)*
From: “Bob”<sip:200@company.com>;tag=1c23354
To: sip:4444@xxx.xxx.208.116 *(modified to protect innocent)*
Call-Id: call-1062902085-96@192.168.100.112
Cseq: 1 INVITE
Contact: “Bob”<sip:200@192.168.100.112>
Content-Type: application/sdp
Content-Length: 312
Accept-Language: en
Allow: INVITE, ACK, CANCEL, BYE, REFER, OPTIONS, NOTIFY, REGISTER, SUBSCRIBE Supported: sip-cc, sip-cc-01, timer, replaces User-Agent: Pingtel/2.1.10 (VxWorks)
Date: Sun, 07 Sep 2003 02:34:45 GMT
Via: SIP/2.0/UDP xxx.xxx.65.12
Via: SIP/2.0/UDP 192.168.100.112

v=0
o=Pingtel 5 5 IN IP4 192.168.100.112
s=phone-call
c=IN IP4 192.168.100.112
t=0 0
m=audio 8776 RTP/AVP 96 97 0 8 18 98
a=rtpmap:96 eg711u/8000/1
a=rtpmap:97 eg711a/8000/1
a=rtpmap:0 pcmu/8000/1
a=rtpmap:8 pcma/8000/1
a=rtpmap:18 g729/8000/1
a=fmtp:18 annexb=no
a=rtpmap:98 telephone-event/8000/1
----------END----------

Below is an example of the 200 call accepted message. The VIA headers and record-route information indicate the route the packet will follow. If this packet can make it through the firewall to the SIP proxy, the SIP proxy will forward it to the initiating end point. Once again a message type 200 is equivalent to picking up a ringing phone. However two-way communication was not the result of this call. Since the firewall rule set does not allow RTP outbound and all traffic not specifically allowed is being dropped, this call is unsuccessful as neither party can hear one another.

UDP SIP User Agent sent message:

----Remote Host:xxx.xxx.120.210---- Port: 5060----
SIP/2.0 200 OK
From: “Bill Olson”<sip:200@company.com>;tag=1c23354
To: sip:Alice@xxx.xxx.208.116;tag=565
Call-Id: call-1062902085-96@192.168.100.112
Cseq: 1 INVITE
Content-Type: application/sdp
Content-Length: 308
Via: SIP/2.0/UDP
xxx.xxx.120.210:5060;branch=z9hG4bKe91bd066b1837c79acc840e865372a94
Via: SIP/2.0/UDP 192.168.100.112
Record-Route:
<sip:xxx.xxx.120.210:5060;lr;a;t=1c23354;s=daa7370949ca39f006ee33c62e8ea
4e7>
Contact: sip:Bob@192.168.100.112
Allow: INVITE, ACK, CANCEL, BYE, REFER, OPTIONS, NOTIFY, REGISTER,
SUBSCRIBE
User-Agent: Pingtel/2.1.10 (VxWorks)
Date: Sun, 07 Sep 2003 02:34:50 GMT

v=0
o=Pingtel 5 5 IN IP4 192.168.100.112
s=phone-call
c=IN IP4 192.168.100.112
t=0 0
m=audio 8770 RTP/AVP 96 97 0 8 18 98
a=rtpmap:96 eg711u/8000/1
a=rtpmap:97 eg711a/8000/1
a=rtpmap:0 pcmu/8000/1
a=rtpmap:8 pcma/8000/1
a=rtpmap:18 g729/8000/1
a=fmtp:18 annexb=no
a=rtpmap:98 telephone-event/8000/1
----------END----------

Possible Solutions

The above example is not all that complex and yet you can see there are several issues. The problems are increased when you begin to consider other call features, e.g. hold, three-way calling and voicemail. Additionally it does not consider a call coming from the outside in

To resolve these problems, several solutions have been designed and proposed. This section will outline some o them and highlight the pros and cons and any security concerns.

STUN/TURN

The first solution is something called Simple Traversal of UDP through Network Address Translation, more commonly known as STUN, RFC3489. This is a protocol that is designed to let the SIP UA discover if it is behind NAT and determine the type of NAT. One issue with this solution is that it may not work with the most popular type of NAT – symmetric.

To implement this solution, it requires the addition of a STUN server to be located outside the firewall and all of the UAs must be STUN compatible. STUN capable UAs send out exploratory messages, to determine the receive ports to use. The STUN server examines these messages and informs the UA which IP and port were used during the exploratory request. The UA receives this information and formats all SIP messages with this information. Since the STUN server does not sit in line, the SIP messages will not "pass-through" it.

However there is a problem with this solution, since most NAT implementations use symmetric NAT. This means the NATing device keeps a mapping based on source IP:port and destination IP:port. Since the destination IP is different than the STUN server there is a strong possibility the port will also be different. If this happens the call will fail. Incoming calls will also fail since the port is different.

STUN relies on the fact that once the outgoing port has been mapped for the STUN server, all traffic regardless of where is it appearing will be allowed to pass, using the reverse mapping. NATs that work this way are very susceptible to port scanning and generally create other serious security concerns.

An additional solution proposed by IETF, Traversal Using Relay NAT or TURN, adds the ability for the TURN server to replace the STUN server. Similar to STUN the client sends a request for an IP and port. The TURN server replies, sending the IP and port of the TURN server. All replies are then received by the TURN server and forwarded to NAT which then re-addresses it and sends it through. These are designed to also solve the symmetric NAT problems.

As you can see, this complicates thing even more and may require upgrades to current VoIP hardware and reconfiguration of the current security architecture. In fact the TURN IETF draft says TURN should only be used as a last resort.

Pros:

- No need to modify current firewall or other network architecture
- Works with most residential NATs
- Adds additional functionality to games and other applications

Cons:

- Server with potential vulnerabilities unprotected
- Many VoIP vendors have not and will not make STUN compatible hardware
- Need keep alive packets to ensure media flows through firewall

- For symmetric NAT implementations may require an additional server – which is also located outside the firewall
- Servers vulnerable to attack

UPnP

While we all ran to patch and turn off UPnP from our Windows XP installations, we may have over looked the original purpose of the protocol. Targeted at SOHO users it is designed to solve a number of issues including the simple configuration of small networks. This solution also requires all UAs to be specially enabled, in this case for UPnP. There is also a significant security issue with this solution; the client controls the opening and close of the firewall ports. This is a major change in philosophy and the potential to exploit via worm or virus is high.

Pros:

- Simple configurations for users
- It is automatic

Cons:

- Will not work with cascading NATs – this would rule deployment to a lot of SOHO locations.
- Most of the enterprise Firewall deployments are not compatible with UPnP

Application Layer Gateway (ALG)

This solution is the most exciting and maybe the simplest to deploy while being extremely complex for the vendors to develop. An ALG understands all of the packets passing through it up to the application layer. The ALG reads through the SIP messages and can modify the messages to reflect the public IP address and port of the ALG .

Since all the traffic is controlled by the ALG traffic can be restricted and verified. ALG can help to ensure callers are who they say they are by examining the source and destination address of the packet this would help prevent call Hijacking. In addition the can ALG examine the contents of the packets passing through every allowed port to ensure they contain the properly formed messages. Full stateful inspection of the SIP commands ensures the SIP packets are structurally valid and arrive in the appropriate sequence. ALG vendors need to walk a fine line to ensure the added controls do not slow communication or add latency that might lower voice quality.

Pros:

- Highest security, packet inspection can verify SIP data before it enters the network
- No change in VoIP equipment

Cons:

- Requires firewall upgrade or replacement
- Possible loss of voice quality with added latency

Currently true ALGs are few are far between, but the wide range of feature possibilities promised will make this solution one of the first considerations for most organizations.

Manual Configuration

This solution requires that each UA be configured with the details of the public IP address and ports to be used during call setup. The NAT device is also configured with static mappings for each client.

Pros:

- Allows traffic to pass
- No change in VoIP equipment

Cons:

- Need for IP addresses to match the number of UAs
- Ports are always open to the internet to allow incoming calls

This solution is one of the least popular solutions for the obvious security reasons.

Tunneling Techniques

This solution attempts to resolve the problem by locating a server inside and a server outside of the firewall, tunneling the both the SIP signals and media information through the current NAT device.

The SIP proxy server is programmed to forward all call information to the tunnel origination server. This is configured send all information the tunnel termination server located outside the firewall. This data passes through the firewall usually unencrypted. Since the two end points know about one another it is easy to deal with NAT and ports since each can be set statically. With small changes in the firewall rule set allowing traffic to pass through the firewall calls from the outside can be routed inside to the SIP proxy.

While this solution requires only minimal changes to the firewall policy it present significant security risks since the tunnel termination server is unprotected and if compromised could allow traffic to pass through the firewall

Pros:

- Allows traffic to pass
- No change in VoIP equipment
- Simple firewall rule changes

Cons:

- Security risks – both known and unknown
- Additional hardware for servers – single point failure

Other solutions

One solution that is not a solution just yet is something called Interactive Connectivity Establishment (ICE). Submitted to IEFT in March of 2003 by Jonathan Rosenberg, this proposal attempts to make use of TURN, STUN and other developing solutions, by having the UAs figure it out on their own. While this looks promising there are still issues with overall security and with the complexity of setup.

There are also several carrier class products that solve these problems and lots of security and performance features. These are very impressive but their current cost will prohibit them from being deployed to all but the largest of enterprises.

Additionally the IEFT group Middlebox Communication (MIDCOM) has submitted two other RFCs 3303 and 3304. These are designed to solve a lot problems NAT and firewalls present and will play some role in a total solution for SIP and other VoIP problems.

One other group that is actively perusing a complete solution is Session Initiation Proposal Investigation (SIPPING). Their charter is an ambitious one; in that they are trying to tie a lot of the loose ends together to develop detailed and complete requirements and extensions to the SIP protocol.

Conclusion

SIP is not necessarily a new protocol however its deployment is not currently widespread. This lack of exposure has limited the risk to date. With each new positive development comes new risk. These new risks while currently undefined are not completely indefensible. If SIP deployments follow basic security guidelines, like IDS (Host and Network), patch management, and secure designs we can reduce and control the risks.

While SIP has the promise of spectacular features and benefits, there are still many questions and issues with interoperability. There are new security concerns with most solutions. Adding SIP to your organization will require exhaustive research and testing before a successful deployment. Additionally the security policies of your organization will dictate a significant portion of the architectural decisions. Working with an experienced vendor and integrator will help to ensure you meet your company's security and performance goals.

References

[1] Stahl, Karl Enabling NATs And Firewalls With SIP http://www.tmcnet.com/it/0503/0503SIP.htm

[2] Sip Center IETF SIP RFC's, Drafts and Announcements, http://www.sipcenter.com/news/drafts/ietfsipdrafts.html 2003

[3] Kuthan, Jiri and Sisalem, Dorgham Session Initiation Tutorial http://www.iptel.org/sip/ 2000-2002

[4] Sinnreich, Henry and Johnston, Alan B. Internet Communications Using SIP John Wiley & Sons, 2001

[5] Camarillo, Gonzalo and Rosenberg, Jonathan. SIP Demystified McGraw-Hill Professional, 2001

[6] Rosenberg, Jonathan General SIP information http://www.jdrosen.net/ 2003

[7] Pingtel. SIPExchange an IP PBX http://www.pingtel.com/pr_sipx_overview.jsp 2003

[8] Intertex. SIP Firewalls http://www.intertexdata.com/products/list.asp?iMenuID=110&t=20 2003

[9] Dynamicsoft - Rosenberg, Jonathan SIP and NAT http://www.dynamicsoft.com/news/presentations/SIPnNAT.pdf May 2002

[10] Dynamicsoft, Rosenberg, Jonathan IETF Update http://www.dynamicsoft.com/news/presentations/VON-Q2-2003_IETF_Update.pdf 2003

[11] Columbia University, Schulzrinne, Henning SIP FAQ, http://www.cs.columbia.edu/sip/faq/ September 2003

[12] DeltaThree. Sterman, Baruch and Schwartz, David NAT traversal in SIP http://corp.deltathree.com/technology/nattraversalinsip.pdf date not available