             Requirements from Session Initiation Protocol (SIP)
                   Session Border Control (SBC) Deployments

Abstract

   This document describes functions implemented in Session Initiation
   Protocol (SIP) intermediaries known as Session Border Controllers
   (SBCs). The goal of this document is to describe the commonly
   provided functions of SBCs. A special focus is given to those
   practices that are viewed to be in conflict with SIP architectural
   principles. This document also explores the underlying requirements
   of network operators that have led to the use of these functions and
   practices in order to identify protocol requirements and determine
   whether those requirements are satisfied by existing specifications
   or if additional standards work is required.

Status of This Memo

Table of Contents

1. Introduction

   In the past few years, there has been a rapid adoption of the Session
   Initiation Protocol (SIP) [1] and deployment of SIP-based
   communications networks. This has often outpaced the development and
   implementation of protocol specifications to meet network operator
   requirements. This has led to the development of proprietary
   solutions. Often, these proprietary solutions are implemented in
   network intermediaries known in the marketplace as Session Border
   Controllers (SBCs) because they typically are deployed at the border
   between two networks. The reason for this is that network policies
   are typically enforced at the edge of the network.

   Even though many SBCs currently behave in ways that can break end-to-
   end security and impact feature negotiations, there is clearly a
   market for them. Network operators need many of the features current
   SBCs provide, and often there are no standard mechanisms available to
   provide them.

   The purpose of this document is to describe functions implemented in
   SBCs. A special focus is given to those practices that conflict with
   SIP architectural principles in some way. The document also explores
   the underlying requirements of network operators that have led to the
   use of these functions and practices in order to identify protocol
   requirements and determine whether those requirements are satisfied
   by existing specifications or if additional standards work is
   required.

2. Background on SBCs

   The term SBC is relatively non-specific, since it is not standardized
   or defined anywhere. Nodes that may be referred to as SBCs but do
   not implement SIP are outside the scope of this document.

   SBCs usually sit between two service provider networks in a peering
   environment, or between an access network and a backbone network to
   provide service to residential and/or enterprise customers. They
   provide a variety of functions to enable or enhance session-based
   multi-media services (e.g., Voice over IP). These functions include:
   a) perimeter defense (access control, topology hiding, and denial-of-
   service prevention and detection); b) functionality not available in
   the endpoints (NAT traversal, protocol interworking or repair); and
   c) traffic management (media monitoring and Quality of Service
   (QoS)). Some of these functions may also get integrated into other
   SIP elements (like pre-paid platforms, Third Generation Partnership
   Project (3GPP) Proxy CSCF (P-CSCF) [6], 3GPP I-CSCF, etc.).

SIP-based SBCs typically handle both signaling and media and can
implement behavior that is equivalent to a "privacy service" (as
described in [2]) performing both Header Privacy and Session
Privacy). SBCs often modify certain SIP headers and message bodies
that proxies are not allowed to modify. Consequently, they are, by
definition, B2BUAs (Back-to-Back User Agents). The transparency of
these B2BUAs varies depending on the functions they perform. For
example, some SBCs modify the session description carried in the
message and insert a Record-Route entry. Other SBCs replace the
value of the Contact header field with the SBCs' address and generate
a new Call-ID and new To and From tags.

```
                        +-----------------+
                        |      SBC        |
          [signaling]   |  +-----------+  |
          <------------|->| signaling |<-|---------->
            outer       |  +-----------+  |  inner
            network     |       |         |  network
                        |  +-----------+  |
          <------------|->|   media   |<-|---------->
            [media]     |  +-----------+  |
                        +-----------------+
```
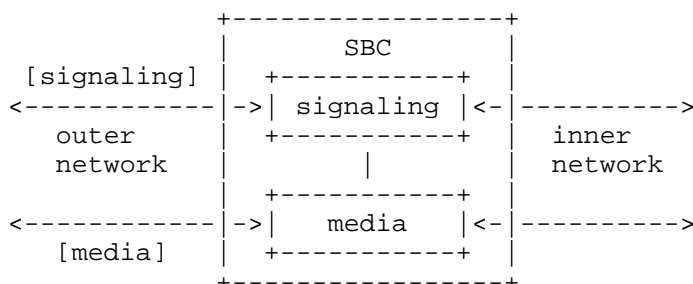
                   Figure 1: SBC Architecture

Figure 1 shows the logical architecture of an SBC, which includes a
signaling and a media component. In this document, the terms outer
and inner network are used for describing these two networks. An SBC
is logically associated with the inner network, and it typically
provides functions such as controlling and protecting access to the
inner network from the outer network. The SBC itself is configured
and managed by the organization operating the inner network.

In some scenarios, SBCs operate with users' (implicit or explicit)
consent; while in others, they operate without users' consent (this
latter case can potentially cause problems). For example, if an SBC
in the same administrative domain as a set of enterprise users
performs topology hiding (see Section 3.1), the enterprise users can
choose to route their SIP messages through it. If they choose to
route through the SBC, then the SBC can be seen as having the users'
implicit consent. Another example is a scenario where a service
provider has broken gateways and it deploys an SBC in front of them
for protocol repair reasons (see Section 3.6). Users can choose to
configure the SBC as their gateway and, so, the SBC can be seen as
having the users' implicit consent.

2.1. Peering Scenario

   A typical peering scenario involves two network operators who
   exchange traffic with each other. An example peering scenario is
   illustrated in Figure 2. An originating gateway (GW-A1) in Operator
   A's network sends an INVITE that is routed to the SBC in Operator B's
   network. Then, the SBC forward it to the softswitch (SS-B). The
   softswitch responds with a redirect (3xx) message back to the SBC
   that points to the appropriate terminating gateway (GW-B1) in
   Operator B's network. If Operator B does not have an SBC, the
   redirect message would go to the Operator A's originating gateway.
   After receiving the redirect message, the SBC sends the INVITE to the
   terminating gateway.

```
        Operator A              .            Operator B
                                .
                                .            2) INVITE
    +-----+                     .        /--------------->+-----+
    |SS-A |                     .       /  3) 3xx (redir.) |SS-B |
    +-----+                     .      /  /---------------+-----+
                                .     /  /
    +-----+  1) INVITE      +-----+--/  /              +-----+
    |GW-A1|--------------->| SBC |<---/     4) INVITE   |GW-B1|
    +-----+                 +-----+--------------------->+-----+
                                .
    +-----+                     .                       +-----+
    |GW-A2|                     .                       |GW-B2|
    +-----+                     .                       +-----+
```
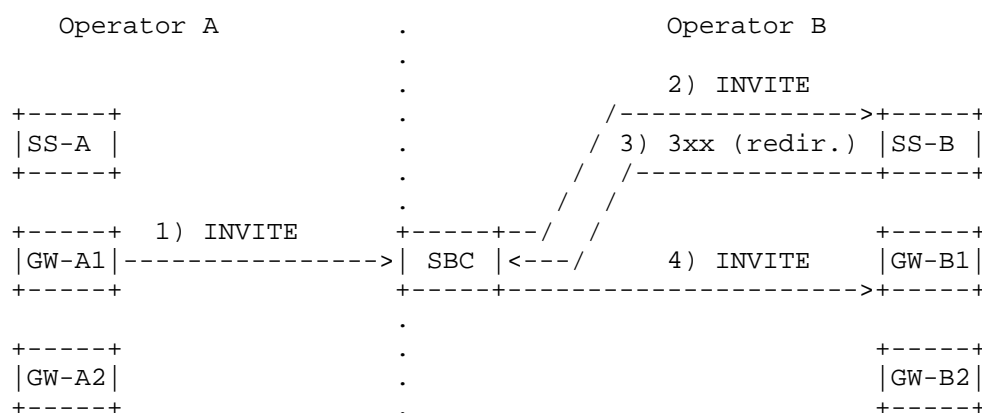
                      Figure 2: Peering with SBC

   From the SBC's perspective the Operator A is the outer network, and
   Operator B is the inner network. Operator B can use the SBC, for
   example, to control access to its network, protect its gateways and
   softswitches from unauthorized use and denial-of-service (DoS)
   attacks, and monitor the signaling and media traffic. It also
   simplifies network management by minimizing the number of ACL (Access
   Control List) entries in the gateways. The gateways do not need to
   be exposed to the peer network, and they can restrict access (both
   media and signaling) to the SBCs. The SBC helps ensure that only
   media from sessions the SBC authorizes will reach the gateway.

2.2. Access Scenario

   In an access scenario, presented in Figure 3, the SBC is placed at
   the border between the access network (outer network) and the
   operator's network (inner network) to control access to the
   operator's network, protect its components (media servers,

application servers, gateways, etc.) from unauthorized use and DoS
attacks, and monitor the signaling and media traffic. Also, since
the SBC is call stateful, it may provide access control functions to
prevent over-subscription of the access links. Endpoints are
configured with the SBC as their outbound proxy address. The SBC
routes requests to one or more proxies in the operator network.

```
        Access Network                    Operator Network


   +-----+
   | UA1 |<---------\
   +-----+           \
                      \
   +-----+             \------->+-----+        +-------+
   | UA2 |<-------------------->| SBC |<----->| proxy |<-- -
   +-----+                /---->+-----+        +-------+
                         /
   +-----+   +-----+    /
   | UA3 +---+ NAT |<---/
   +-----+   +-----+
```
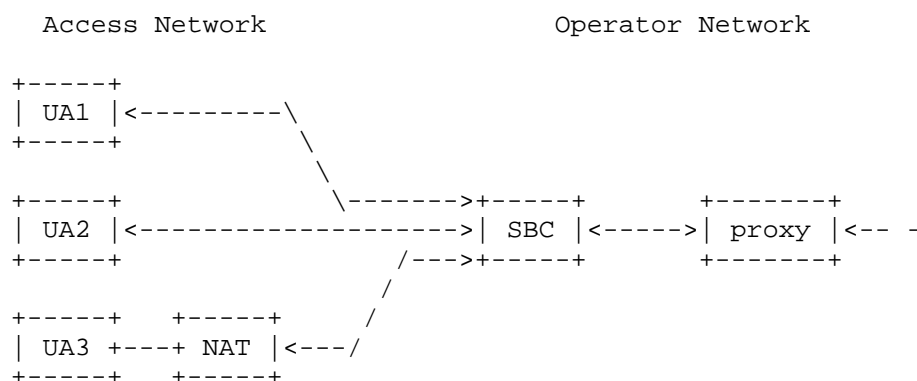
                 Figure 3: Access Scenario with SBC

The SBC may be hosted in the access network (e.g., this is common
when the access network is an enterprise network), or in the operator
network (e.g., this is common when the access network is a
residential or small business network). Despite where the SBC is
hosted, it is managed by the organization maintaining the operator
network.

Some endpoints may be behind enterprise or residential NATs. In
cases where the access network is a private network, the SBC is a NAT
for all traffic. It is noteworthy that SIP traffic may have to
traverse more than one NAT. The proxy usually does authentication
and/or authorization for registrations and outbound calls. The SBC
modifies the REGISTER request so that subsequent requests to the
registered address-of-record are routed to the SBC. This is done
either with a Path header field [3] or by modifying the Contact to
point at the SBC.

The scenario presented in this section is a general one, and it
applies also to other similar settings. One example from a similar
setting is the one where an access network is the open internet, and
the operator network is the network of a SIP service provider.

3. Functions of SBCs

   This section lists those functions that are used in SBC deployments
   in current communication networks. Each subsection describes a
   particular function or feature, the operators' requirements for
   having it, explanation of any impact to the end-to-end SIP
   architecture, and a concrete implementation example. Each section
   also discusses potential concerns specific to that particular
   implementation technique. Suggestions for alternative implementation
   techniques that may be more architecturally compatible with SIP are
   outside the scope of this document.

   All the examples given in this section are simplified; only the
   relevant header lines from SIP and SDP (Session Description Protocol)
   [7] messages are displayed.

3.1. Topology Hiding

3.1.1. General Information and Requirements

   Topology hiding consists of limiting the amount of topology
   information given to external parties. Operators have a requirement
   for this functionality because they do not want the IP addresses of
   their equipment (proxies, gateways, application servers, etc.) to be
   exposed to outside parties. This may be because they do not want to
   expose their equipment to DoS attacks, they may use other carriers
   for certain traffic and do not want their customers to be aware of
   it, or they may want to hide their internal network architecture from
   competitors or partners. In some environments, the operator's
   customers may wish to hide the addresses of their equipment or the
   SIP messages may contain private, non-routable addresses.

   The most common form of topology hiding is the application of header
   privacy (see Section 5.1 of [2]), which involves stripping Via and
   Record-Route headers, replacing the Contact header, and even changing
   Call-IDs. However, in deployments that use IP addresses instead of
   domain names in headers that cannot be removed (e.g., From and To
   headers), the SBC may replace these IP addresses with its own IP
   address or domain name.

   For a reference, there are also other ways of hiding topology
   information than inserting an intermediary, like an SBC, to the
   signaling path. One of the ways is the UA-driven privacy mechanism
   [8], where the UA can facilitate the concealment of topology
   information.

3.1.2. Architectural Issues

   Performing topology hiding, as described above, by SBCs that do not
   have the users' consent presents some issues. This functionality is
   based on a hop-by-hop trust model as opposed to an end-to-end trust
   model. The messages are modified without the subscriber's consent
   and could potentially modify or remove information about the user's
   privacy, security requirements, and higher-layer applications that
   are communicated end-to-end using SIP. Neither user agent in an end-
   to-end call has any way to distinguish the SBC actions from a man-in-
   the-middle (MITM) attack.

   The topology hiding function does not work well with Authenticated
   Identity Management [4] in scenarios where the SBC does not have any
   kind of consent from the users. The Authenticated Identity
   Management mechanism is based on a hash value that is calculated from
   parts of From, To, Call-ID, CSeq, Date, and Contact header fields
   plus from the whole message body. If the authentication service is
   not provided by the SBC itself, the modification of the
   aforementioned header fields and the message body is in violation of
   [4]. Some forms of topology hiding are in violation, because they
   are, e.g., replacing the Contact header of a SIP message.

3.1.3. Example

   The current way of implementing topology hiding consists of having an
   SBC act as a B2BUA (Back-to-Back User Agent) and remove all traces of
   topology information (e.g., Via and Record-Route entries) from
   outgoing messages.

   Imagine the following example scenario: the SBC
   (p4.domain.example.com) receives an INVITE request from the inner
   network, which in this case is an operator network. The received SIP
   message is shown in Figure 4.

```
   INVITE sip:callee@u2.domain.example.com SIP/2.0
   Via: SIP/2.0/UDP p3.middle.example.com;branch=z9hG4bK48jq9w174131.1
   Via: SIP/2.0/UDP p2.example.com;branch=z9hG4bK18an6i9234172.1
   Via: SIP/2.0/UDP p1.example.com;branch=z9hG4bK39bn2e5239289.1
   Via: SIP/2.0/UDP u1.example.com;branch=z9hG4bK92fj4u7283927.1
   Contact: sip:caller@u1.example.com
   Record-Route: <sip:p3.middle.example.com;lr>
   Record-Route: <sip:p2.example.com;lr>
   Record-Route: <sip:p1.example.com;lr>
```

            Figure 4: INVITE Request Prior to Topology Hiding

   Then, the SBC performs a topology hiding function. In this scenario,
   the SBC removes and stores all existing Via and Record-Route headers,
   and then inserts Via and Record-Route header fields with its own SIP
   URI. After the topology hiding function, the message could appear as
   shown in Figure 5.

```
   INVITE sip:callee@u2.domain.example.com SIP/2.0
   Via: SIP/2.0/UDP p4.domain.example.com;branch=z9hG4bK92es3w230129.1
   Contact: sip:caller@u1.example.com
   Record-Route: <sip:p4.domain.example.com;lr>
```

             Figure 5: INVITE Request after Topology Hiding

   Like a regular proxy server that inserts a Record-Route entry, the
   SBC handles every single message of a given SIP dialog. If the SBC
   loses state (e.g., SBC restarts for some reason), it may not be able
   to route messages properly (note: some SBCs preserve the state
   information also on restart). For example, if the SBC removes Via
   entries from a request and then restarts, thus losing state; the SBC
   may not be able to route responses to that request, depending on the
   information that was lost when the SBC restarted.

   This is only one example of topology hiding. Besides topology hiding
   (i.e., information related to the network elements is being hidden),
   SBCs may also do identity hiding (i.e., information related to
   identity of subscribers is being hidden). While performing identity
   hiding, SBCs may modify Contact header field values and other header
   fields containing identity information. The header fields containing
   identity information is listed in Section 4.1 of [2]. Since the
   publication of [2], the following header fields containing identity
   information have been defined: "P-Asserted-Identity", "Referred-By",
   "Identity", and "Identity-Info".

3.2. Media Traffic Management

3.2.1. General Information and Requirements

   Media traffic management is the function of controlling media
   traffic. Network operators may require this functionality in order
   to control the traffic being carried on their network on behalf of
   their subscribers. Traffic management helps the creation of
   different kinds of billing models (e.g., video telephony can be
   priced differently than voice-only calls) and it also makes it
   possible for operators to enforce the usage of selected codecs.

   One of the use cases for media traffic management is the
   implementation of intercept capabilities that are required to support
   audit or legal obligations. It is noteworthy that the legal
   obligations mainly apply to operators providing voice services, and
   those operators typically have infrastructure (e.g., SIP proxies
   acting as B2BUAs) for providing intercept capabilities even without
   SBCs.

   Since the media path is independent of the signaling path, the media
   may not traverse through the operator's network unless the SBC
   modifies the session description. By modifying the session
   description, the SBC can force the media to be sent through a media
   relay which may be co-located with the SBC. This kind of traffic
   management can be done, for example, to ensure a certain QoS level,
   or to ensure that subscribers are using only allowed codecs. It is
   noteworthy that the SBCs do not have direct ties to routing topology
   and they do not, for example, change bandwidth reservations on
   Traffic Engineering (TE) tunnels, nor do they have direct interaction
   with routing protocol.

   Some operators do not want to manage the traffic, but only to monitor
   it to collect statistics and make sure that they are able to meet any
   business service level agreements with their subscribers and/or
   partners. The protocol techniques, from the SBC's viewpoint, needed
   for monitoring media traffic are the same as for managing media
   traffic.

   SBCs on the media path are also capable of dealing with the "lost
   BYE" issue if either endpoint dies in the middle of the session. The
   SBC can detect that the media has stopped flowing and issue a BYE to
   both sides to clean up any state in other intermediate elements and
   the endpoints.

   One possible form of media traffic management is that SBCs terminate
   media streams and SIP dialogs by generating BYE requests. This kind
   of procedure can take place, for example, in a situation where the

   subscriber runs out of credits. Media management is needed to ensure
   that the subscriber cannot just ignore the BYE request generated by
   the SBC and continue its media sessions.

3.2.2. Architectural Issues

   Implementing traffic management in this manner requires the SBC to
   access and modify the session descriptions (i.e., offers and answers)
   exchanged between the user agents. Consequently, this approach does
   not work if user agents encrypt or integrity-protect their message
   bodies end-to-end. Again, messages are modified without subscriber
   consent, and user agents do not have any way to distinguish the SBC
   actions from an attack by a MITM. Furthermore, this is in violation
   of Authenticated Identity Management [4], see Section 3.1.2.

   The insertion of a media relay can prevent "non-media" uses of the
   media path, for example, the media path key agreement. Sometimes
   this type of prevention is intentional, but it is not always
   necessary. For example, if an SBC is used just for enabling media
   monitoring, but not for interception.

   There are some possible issues related to the media relaying. If the
   media relaying is not done in the correct manner, it may break
   functions like Explicit Congestion Notification (ECN) and Path MTU
   Discovery (PMTUD), for example. The media relays easily break such
   IP and transport layer functionalities that rely on the correct
   handling of the protocol fields. Some especially sensitive fields
   are, for example, ECN and Type of Service (ToS) fields and the Don't
   Fragment (DF) bit.

   The way in which media traffic management functions impedes
   innovation. The reason for the impediment is that in many cases,
   SBCs need to be able to support new forms of communication (e.g.,
   extensions to the SDP protocol) before new services can be put into
   use, which slows the adoption of new innovations.

   If an SBC directs many media streams through a central point in the
   network, it is likely to cause a significant amount of additional
   traffic to a path to that central point. This might create possible
   bottleneck in the path.

   In this application, the SBC may originate messages that the user may
   not be able to authenticate as coming from the dialog peer or the SIP
   Registrar/Proxy.

3.2.3. Example

   Traffic management may be performed in the following way: The SBC
   behaves as a B2BUA and inserts itself, or some other entity under the
   operator's control, in the media path. In practice, the SBC modifies
   the session descriptions carried in the SIP messages. As a result,
   the SBC receives media from one user agent and relays it to the other
   user agent and performs the identical operation with media traveling
   in the reverse direction.

   As mentioned in Section 3.2.1, codec restriction is a form of traffic
   management. The SBC restricts the codec set negotiated in the offer/
   answer exchange [5] between the user agents. After modifying the
   session descriptions, the SBC can check whether or not the media
   stream corresponds to what was negotiated in the offer/answer
   exchange. If it differs, the SBC has the ability to terminate the
   media stream or take other appropriate (configured) actions (e.g.,
   raise an alarm).

   Consider the following example scenario: the SBC receives an INVITE
   request from the outer network, which in this case is an access
   network. The received SIP message contains the SDP session
   descriptor shown in Figure 6.

      v=0
      o=owner 2890844526 2890842807 IN IP4 192.0.2.4
      c=IN IP4 192.0.2.4
      m=audio 49230 RTP/AVP 96 98
      a=rtpmap:96 L8/8000
      a=rtpmap:98 L16/16000/2

              Figure 6: Request Prior to Media Management

   In this example, the SBC performs the media traffic management
   function by rewriting the "m=" line, and removing one "a=" line
   according to some (external) policy. Figure 7 shows the session
   description after the traffic management function.

      v=0
      o=owner 2890844526 2890842807 IN IP4 192.0.2.4
      c=IN IP4 192.0.2.4
      m=audio 49230 RTP/AVP 96
      a=rtpmap:96 L8/8000

              Figure 7: Request Body After Media Management

Media traffic management has a problem where the SBC needs to
understand the session description protocol and all extensions used
by the user agents. This means that in order to use a new extension
(e.g., an extension to implement a new service) or a new session
description protocol, SBCs in the network may need to be upgraded in
conjunction with the endpoints. It is noteworthy that a similar
problem, but with header fields, applies to, for example, topology
hiding function, see Section 3.1. Certain extensions that do not
require active manipulation of the session descriptors to facilitate
traffic management will be able to be deployed without upgrading
existing SBCs, depending on the degree of transparency the SBC
implementation affords. In cases requiring an SBC modification to
support the new protocol features, the rate of service deployment may
be affected.

3.3. Fixing Capability Mismatches

3.3.1. General Information and Requirements

SBCs fixing capability mismatches enable communications between user
agents with different capabilities or extensions. For example, an
SBC can enable a plain SIP [1] user agent to connect to a 3GPP
network, or enable a connection between user agents that support
different IP versions, different codecs, or that are in different
address realms. Operators have a requirement and a strong motivation
for performing capability mismatch fixing, so that they can provide
transparent communication across different domains. In some cases,
different SIP extensions or methods to implement the same SIP
application (like monitoring session liveness, call history/diversion
etc.) may also be interworked through the SBC.

3.3.2. Architectural Issues

SBCs that are fixing capability mismatches do it by inserting a media
element into the media path using the procedures described in
Section 3.2. Therefore, these SBCs have the same concerns as SBCs
performing traffic management: the SBC may modify SIP messages
without consent from any of the user agents. This may break end-to-
end security and application extensions negotiation.

The capability mismatch fixing is a fragile function in the long
term. The number of incompatibilities built into various network
elements is increasing the fragility and complexity over time. This
might lead to a situation where SBCs need to be able to handle a
large number of capability mismatches in parallel.

3.3.3. Example

   Consider the following example scenario where the inner network is an
   access network using IPv4 and the outer network is using IPv6. The
   SBC receives an INVITE request with a session description from the
   access network:

      INVITE sip:callee@ipv6.domain.example.com SIP/2.0
      Via: SIP/2.0/UDP 192.0.2.4
      Contact: sip:caller@u1.example.com

      v=0
      o=owner 2890844526 2890842807 IN IP4 192.0.2.4
      c=IN IP4 192.0.2.4
      m=audio 49230 RTP/AVP 96
      a=rtpmap:96 L8/8000

             Figure 8: Request Prior to Capabilities Match

   Then, the SBC performs a capability mismatch fixing function. In
   this scenario, the SBC inserts Record-Route and Via headers and
   rewrites the "c=" line from the sessions descriptor. Figure 9 shows
   the request after the capability mismatch adjustment.

      INVITE sip:callee@ipv6.domain.com SIP/2.0
      Record-Route: <sip:[2001:DB8::801:201:2ff:fe94:8e10];lr>
      Via: SIP/2.0/UDP sip:[2001:DB8::801:201:2ff:fe94:8e10]
      Via: SIP/2.0/UDP 192.0.2.4
      Contact: sip:caller@u1.example.com

      v=0
      o=owner 2890844526 2890842807 IN IP4 192.0.2.4
      c=IN IP6 2001:DB8::801:201:2ff:fe94:8e10
      m=audio 49230 RTP/AVP 96
      a=rtpmap:96 L8/8000

                Figure 9: Request after Capability Match

   This message is then sent by the SBC to the onward IPv6 network.

3.4. Maintaining SIP-Related NAT Bindings

3.4.1. General Information and Requirements

   NAT traversal in this instance refers to the specific message
   modifications required to assist a user agent in maintaining SIP and
   media connectivity when there is a NAT device located between a user
   agent and a proxy/registrar and, possibly, any other user agent. The

primary purpose of NAT traversal function is to keep up a control
connection to user agents behind NATs. This can, for example, be
achieved by generating periodic network traffic that keeps bindings
in NATs alive. SBCs' NAT traversal function is required in scenarios
where the NAT is outside the SBC (i.e., not in cases where SBC itself
acts as a NAT).

An SBC performing a NAT (Network Address Translator) traversal
function for a user agent behind a NAT sits between the user agent
and the registrar of the domain. NATs are widely deployed in various
access networks today, so operators have a requirement to support it.
When the registrar receives a REGISTER request from the user agent
and responds with a 200 (OK) response, the SBC modifies such a
response decreasing the validity of the registration (i.e., the
registration expires sooner). This forces the user agent to send a
new REGISTER to refresh the registration sooner that it would have
done on receiving the original response from the registrar. The
REGISTER requests sent by the user agent refresh the binding of the
NAT before the binding expires.

Note that the SBC does not need to relay all the REGISTER requests
received from the user agent to the registrar. The SBC can generate
responses to REGISTER requests received before the registration is
about to expire at the registrar. Moreover, the SBC needs to
deregister the user agent if this fails to refresh its registration
in time, even if the registration at the registrar would still be
valid.

SBCs can also force traffic to go through a media relay for NAT
traversal purposes (more about media traffic management in
Section 3.2). A typical call has media streams to two directions.
Even though SBCs can force media streams from both directions to go
through a media relay, in some cases, it is enough to relay only the
media from one direction (e.g., in a scenario where only the other
endpoint is behind a NAT).

3.4.2. Architectural Issues

This approach to NAT traversal does not work if end-to-end
confidentiality or integrity-protection mechanisms are used (e.g.,
Secure/Multipurpose Internet Mail Extensions (S/MIME)). The SBC
would be seen as a MITM modifying the messages between the user agent
and the registrar.

There is also a problem related to the method of how SBCs choose the
value for the validity of a registration period. This value should
be as high as possible, but it still needs to be low enough to
maintain the NAT binding. Some SBCs do not have any deterministic

method for choosing a suitable value. However, SBCs can just use a
sub-optimal, relatively small value that usually works. An example
from such value is 15 seconds (see [9]).

NAT traversal for media using SBCs poses few issues as well. For
example, an SBC normally guesses the recipient's public IP address on
one of the media streams relayed by the SBC by snooping on the source
IP address of another media stream relayed by the same SBC. This
causes security and interoperability issues since the SBC can end up
associating wrong destination IP addresses on media streams it is
relaying. For example, an attacker may snoop on the local IP address
and ports used by the SBC for media relaying the streams and send a
few packets from a malicious IP address to these destinations. In
most cases, this can cause media streams in the opposite directions
to divert traffic to the attacker resulting in a successful MITM or
DoS attack. A similar example of an interoperability issue is caused
when an endpoint behind a NAT attempts to switch the IP address of
the media streams by using a re-INVITE. If any media packets are re-
ordered or delayed in the network, they can cause the SBC to block
the switch from happening even if the re-INVITE successfully goes
through.

3.4.3. Example

Consider the following example scenario: The SBC resides between the
UA and Registrar. Previously, the UA has sent a REGISTER request to
the Registrar, and the SBC receives the registration response shown
in Figure 10.

```
   SIP/2.0 200 OK
   From: Bob <sip:bob@biloxi.example.com>;tag=a73kszlfl
   To: Bob <sip:bob@biloxi.example.com>;tag=34095828jh
   CSeq: 1 REGISTER
   Contact: <sips:bob@client.biloxi.example.com>;expires=3600
```

        Figure 10: Response Prior to NAT Maintenance Function

When performing the NAT traversal function, the SBC may rewrite the
expiry time to coax the UA to re-register prior to the intermediating
NAT deciding to close the pinhole. Figure 11 shows a possible
modification of the response from Figure 10.

```
SIP/2.0 200 OK
From: Bob <sip:bob@biloxi.example.com>;tag=a73kszlfl
To: Bob <sip:bob@biloxi.example.com>;tag=34095828jh
CSeq: 1 REGISTER
Contact: <sips:bob@client.biloxi.example.com>;expires=60
```

           Figure 11: Manipulated Response for NAT Traversal

   Naturally, other measures could be taken in order to enable the NAT
   traversal (e.g., non-SIP keepalive messages), but this example
   illustrates only one mechanism for preserving the SIP-related NAT
   bindings.

3.5. Access Control

3.5.1. General Information and Requirements

   Network operators may wish to control what kind of signaling and
   media traffic their network carries. There is strong motivation and
   a requirement to do access control on the edge of an operator's
   network. Access control can be based on, for example, link-layer
   identifiers, IP addresses or SIP identities.

   This function can be implemented by protecting the inner network with
   firewalls and configuring them so that they only accept SIP traffic
   from the SBC. This way, all the SIP traffic entering the inner
   network needs to be routed though the SBC, which only routes messages
   from authorized parties or traffic that meets a specific policy that
   is expressed in the SBC administratively.

   Access control can be applied to either only the signaling or both
   the signaling and media. If it is applied only to the signaling,
   then the SBC might behave as a proxy server. If access control is
   applied to both the signaling and media, then the SBC behaves in a
   similar manner as explained in Section 3.2. A key part of media-
   layer access control is that only media for authorized sessions is
   allowed to pass through the SBC and/or associated media relay
   devices.

   Operators implement some functionalities, like NAT traversal for
   example, in an SBC instead of other elements in the inner network for
   several reasons: (i) preventing packets from unregistered users to
   prevent chances of DoS attack, (ii) prioritization and/or re-routing
   of traffic (based on user or service, like E911) as it enters the
   network, and (iii) performing a load balancing function or reducing
   the load on other network equipment.

In environments where there is limited bandwidth on the access links,
the SBC can compute the potential bandwidth use by examining the
codecs present in SDP offers and answers. With this information, the
SBC can reject sessions before the available bandwidth is exhausted
to allow existing sessions to maintain acceptable quality of service.
Otherwise, the link could become over-subscribed and all sessions
would experience a deterioration in quality of service. SBCs may
contact a policy server to determine whether sufficient bandwidth is
available on a per-session basis.

3.5.2. Architectural Issues

Since the SBC needs to handle all SIP messages, this function has
scalability implications. In addition, the SBC is a single point of
failure from an architectural point of view. Although, in practice,
many current SBCs have the capability to support redundant
configuration, which prevents the loss of calls and/or sessions in
the event of a failure on a single node.

If access control is performed only on behalf of signaling, then the
SBC is compatible with general SIP architectural principles, but if
it is performed for signaling and for media, then there are similar
problems as described in Section 3.2.2.

3.5.3. Example

Figure 12 shows a callflow where the SBC is providing both signaling
and media access control (ACKs omitted for brevity).

```
        caller                    SBC                    callee
         |                         |                       |
         |  Identify the caller    |                       |
         |<- - - - - - - - - - - >|                       |
         |                         |                       |
         |     INVITE + SDP        |                       |
         |----------------------->|                       |
         |              [Modify the SDP]                   |
         |                         | INVITE + modified SDP |
         |                         |---------------------->|
         |                         |                       |
         |                         |     200 OK + SDP      |
         |                         |<----------------------|
         |              [Modify the SDP]                   |
         |                         |                       |
         | 200 OK + modified SDP   |                       |
         |<----------------------|                       |
         |                         |                       |
         |     Media    [Media inspection]    Media        |
         |<=====================>|<=====================>|
         |                         |                       |
```
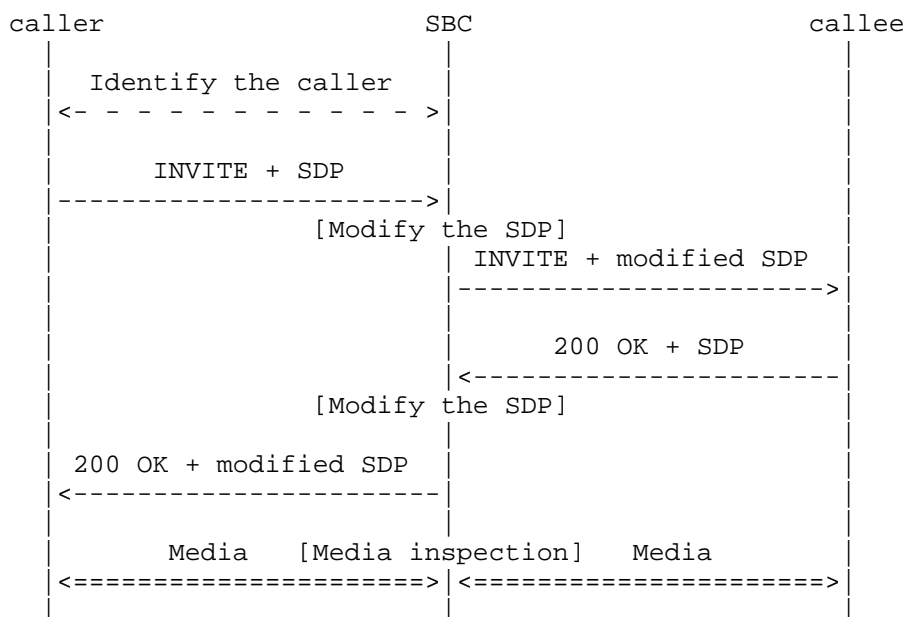
                    Figure 12: Example Access Callflow

   In this scenario, the SBC first identifies the caller, so it can
   determine whether or not to give signaling access to the caller.
   This might be achieved using information gathered during
   registration, or by other means. Some SBCs may rely on the proxy to
   authenticate the user agent placing the call. After identification,
   the SBC modifies the session descriptors in INVITE and 200 OK
   messages in a way so that the media is going to flow through the SBC
   itself. When the media starts flowing, the SBC can inspect whether
   the callee and caller use the codec(s) upon which they had previously
   agreed.

3.6. Protocol Repair

3.6.1. General Information and Requirements

   SBCs are also used to repair protocol messages generated by not-
   fully-standard-compliant or badly implemented clients. Operators may
   wish to support protocol repair, if they want to support as many
   clients as possible. It is noteworthy that this function affects
   only the signaling component of an SBC, and that the protocol repair
   function is not the same as protocol conversion (i.e., making
   translation between two completely different protocols).

3.6.2. Architectural Issues

   In many cases, doing protocol repair for SIP header fields can be
   seen as being compatible with SIP architectural principles, and it
   does not violate the end-to-end model of SIP. The SBC repairing
   protocol messages behaves as a proxy server that is liberal in what
   it accepts and strict in what it sends.

   However, protocol repair may break security mechanism that do
   cryptographical computations on SIP header values. Attempting
   protocol repair for SIP message bodies (SDP) is incompatible with
   Authenticated Identity Management [4] and end-to-end security
   mechanisms such as S/MIME.

   A similar problem related to increasing complexity, as explained in
   Section 3.3.2, also affects protocol repair function.

3.6.3. Examples

   The SBC can, for example, receive an INVITE message from a relatively
   new SIP UA as illustrated in Figure 13.

      INVITE sip:callee@sbchost.example.com
      Via: SIP/2.0/UDP u1.example.com:5060;lr
      From: Caller <sip:caller@one.example.com>
      To: Callee   <sip:callee@two.example.com>
      Call-ID: 18293281@u1.example.com
      CSeq: 1   INVITE
      Contact: sip:caller@u1.example.com

              Figure 13: Request from a Relatively New Client

   If the SBC does protocol repair, it can rewrite the 'lr' parameter on
   the Via header field into the form 'lr=true' in order to support some
   older, badly implemented SIP stacks. It could also remove excess
   white spaces to make the SIP message more human readable.

3.7. Media Encryption

3.7.1. General Information and Requirements

   SBCs are used to perform media encryption/decryption at the edge of
   the network. This is the case when media encryption (e.g., Secure
   Real-time Transport Protocol (SRTP)) is used only on the access
   network (outer network) side and the media is carried unencrypted in
   the inner network. Some operators provide the ability to do legal

      interception while still giving their customers the ability to
      encrypt media in the access network. One possible way to do this is
      to perform media encryption function.

3.7.2. Architectural Issues

      While performing a media encryption function, SBCs need to be able to
      inject either themselves, or some other entity to the media path. It
      must be noted that this kind of behavior is the same as a classical
      MITM attack. Due to this, the SBCs have the same architectural
      issues as explained in Section 3.2.

3.7.3. Example

      Figure 14 shows an example where the SBC is performing media-
      encryption-related functions (ACKs omitted for brevity).

```
      caller                SBC#1              SBC#2              callee
        |                     |                  |                  |
        |    INVITE + SDP     |                  |                  |
        |-------------------->|                  |                  |
        |             [Modify the SDP]           |                  |
        |                     |                  |                  |
        |                     | INVITE + mod. SDP|                  |
        |                     |----------------->|                  |
        |                     |           [Modify the SDP]          |
        |                     |                  |                  |
        |                     |                  | INVITE + mod. SDP|
        |                     |                  |----------------->|
        |                     |                  |                  |
        |                     |                  |   200 OK + SDP   |
        |                     |                  |<-----------------|
        |                     |           [Modify the SDP]          |
        |                     |                  |                  |
        |                     | 200 OK + mod. SDP|                  |
        |                     |<-----------------|                  |
        |             [Modify the SDP]           |                  |
        |                     |                  |                  |
        |  200 OK + mod. SDP  |                  |                  |
        |<--------------------|                  |                  |
        |                     |                  |                  |
        |     Encrypted       |      Plain       |     Encrypted    |
        |       media     [enc./dec.]  media   [enc./dec.]  media   |
        |<==================>|<- - - - - - -  ->|<==================>|
        |                     |                  |                  |
```
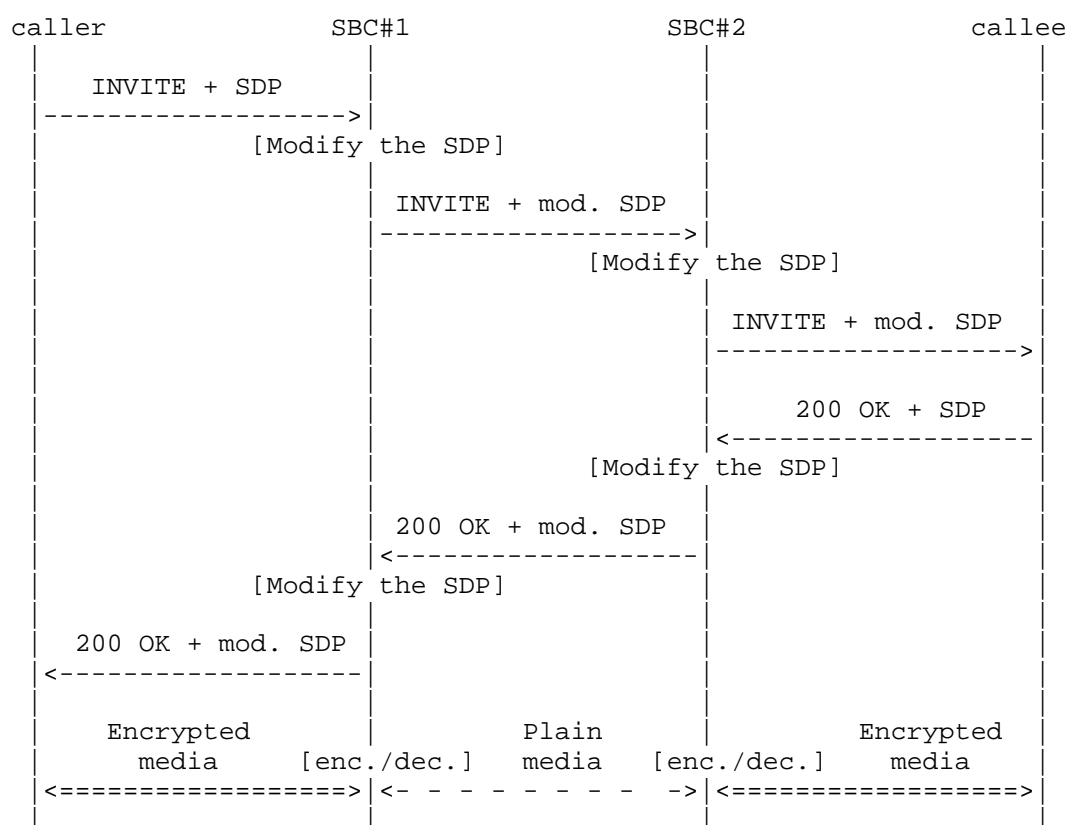
                  Figure 14: Media Encryption Example

First, the UAC sends an INVITE request, and the first SBC modifies
the session descriptor in a way that it injects itself to the media
path. The same happens in the second SBC. Then, the User Agent
Server (UAS) replies with a 200 OK response and the SBCs inject
themselves in the returning media path. After signaling, the media
starts flowing, and both SBCs perform media encryption and
decryption.

4. Derived Requirements for Future SIP Standardization Work

   Some of the functions listed in this document are more SIP-unfriendly
   than others. This list of requirements is derived from the functions
   that break the principles of SIP in one way or another when performed
   by SBCs that do not have the users' consent. The derived
   requirements are:

   Req-1: There should be a SIP-friendly way to hide network topology
           information. Currently, this is done by stripping and
           replacing header fields, which is against the principles of
           SIP on behalf of some header fields (see Section 3.1).

   Req-2: There should be a SIP-friendly way to direct media traffic
           through intermediaries. Currently, this is done by modifying
           session descriptors, which is against the principles of SIP
           (see Sections 3.2, 3.4, 3.5, and 3.7).

   Req-3: There should be a SIP-friendly way to fix capability
           mismatches in SIP messages. This requirement is harder to
           fulfill on complex mismatch cases, like the 3GPP/SIP [1]
           network mismatch. Currently, this is done by modifying SIP
           messages, which may violate end-to-end security (see Sections
           3.3 and 3.6), on behalf of some header fields.

   Req-1 and Req-3 do not have an existing, standardized solution today.
   There is ongoing work in the IETF for addressing Req-2, such as SIP
   session policies [10], Traversal Using Relays around NAT (TURN) [11],
   and Interactive Connectivity Establishment (ICE) [12]. Nonetheless,
   future work is needed in order to develop solutions to these
   requirements.

5. Security Considerations

   Many of the functions this document describes have important security
   and privacy implications. One major security problem is that many
   functions implemented by SBCs (e.g., topology hiding and media
   traffic management) modify SIP messages and their bodies without the
   user agents' consent. The result is that the user agents may

interpret the actions taken by an SBC as a MITM attack. SBCs modify
SIP messages because it allows them to, for example, protect elements
in the inner network from direct attacks.

SBCs that place themselves (or another entity) on the media path can
be used to eavesdrop on conversations. Since, often, user agents
cannot distinguish between the actions of an attacker and those of an
SBC, users cannot know whether they are being eavesdropped on or if
an SBC on the path is performing some other function. SBCs place
themselves on the media path because it allows them to, for example,
perform legal interception.

On a general level, SBCs prevent the use of end-to-end
authentication. This is because SBCs need to be able to perform
actions that look like MITM attacks, and in order for user agents to
communicate, they must allow those type of attacks. It other words,
user agents cannot use end-to-end security. This is especially
harmful because other network elements, besides SBCs, are then able
to do similar attacks. However, in some cases, user agents can
establish encrypted media connections between one another. One
example is a scenario where SBC is used for enabling media monitoring
but not for interception.

An SBC is a single point of failure from the architectural point of
view. This makes it an attractive target for DoS attacks. The fact
that some functions of SBCs require those SBCs to maintain session-
specific information makes the situation even worse. If the SBC
crashes (or is brought down by an attacker), ongoing sessions
experience undetermined behavior.

If the IETF decides to develop standard mechanisms to address the
requirements presented in Section 4, the security and privacy-related
aspects of those mechanisms will, of course, need to be taken into
consideration.

6. Acknowledgements

The ad hoc meeting about SBCs, held on November 9, 2004 in Washington
DC during the 61st IETF meeting, provided valuable input to this
document. The authors would also like to thank Sridhar Ramachandran,
Gaurav Kulshreshtha, and Rakendu Devdhar. Reviewers Spencer Dawkins
and Francois Audet also deserve special thanks.

7. References

7.1. Normative References

   [1]   Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A.,
          Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP:
          Session Initiation Protocol", RFC 3261, June 2002.

   [2]   Peterson, J., "A Privacy Mechanism for the Session Initiation
          Protocol (SIP)", RFC 3323, November 2002.

   [3]   Willis, D. and B. Hoeneisen, "Session Initiation Protocol (SIP)
          Extension Header Field for Registering Non-Adjacent Contacts",
          RFC 3327, December 2002.

   [4]   Peterson, J. and C. Jennings, "Enhancements for Authenticated
          Identity Management in the Session Initiation Protocol (SIP)",
          RFC 4474, August 2006.

   [5]   Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with
          Session Description Protocol (SDP)", RFC 3264, June 2002.

7.2. Informative References

   [6]   3GPP, "IP Multimedia Subsystem (IMS); Stage 2", 3GPP TS 23.228
          10.0.0, March 2010.

   [7]   Handley, M., Jacobson, V., and C. Perkins, "SDP: Session
          Description Protocol", RFC 4566, July 2006.

   [8]   Munakata, M., Schubert, S., and T. Ohba, "User-Agent-Driven
          Privacy Mechanism for SIP", RFC 5767, April 2010.

   [9]   Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines for
          Application Designers", BCP 145, RFC 5405, November 2008.

   [10]   Hilt, V., Camarillo, G., and J. Rosenberg, "A Framework for
          Session Initiation Protocol (SIP) Session Policies", Work
          in Progress, February 2010.

   [11]   Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using
          Relays around NAT (TURN): Relay Extensions to Session Traversal
          Utilities for NAT (STUN)", RFC 5766, April 2010.

   [12]   Rosenberg, J., "Interactive Connectivity Establishment (ICE): A
          Protocol for Network Address Translator (NAT) Traversal for
          Offer/Answer Protocols", RFC 5245, MonthTBD 2010.

Authors' Addresses

    Jani Hautakorpi (editor)
    Ericsson
    Hirsalantie 11
    Jorvas 02420
    Finland

    EMail: Jani.Hautakorpi@ericsson.com

    Gonzalo Camarillo
    Ericsson
    Hirsalantie 11
    Jorvas 02420
    Finland

    EMail: Gonzalo.Camarillo@ericsson.com

    Robert F. Penfield
    Acme Packet
    71 Third Avenue
    Burlington, MA 01803
    US

    EMail: bpenfield@acmepacket.com

    Alan Hawrylyshen
    Skype, Inc.
    2055 E. Hamilton Ave
    San Jose, CA 95125
    US

    EMail: alan.ietf@polyphase.ca

    Medhavi Bhatia
    3CLogic
    9700 Great Seneca Hwy.
    Rockville, MD 20850
    US

    EMail: mbhatia@3clogic.com