           Connection Reuse in the Session Initiation Protocol (SIP)

Abstract

   This document enables a pair of communicating proxies to reuse a
   congestion-controlled connection between themselves for sending
   requests in the forwards and backwards direction. Because the
   connection is essentially aliased for requests going in the backwards
   direction, reuse is predicated upon both the communicating endpoints
   authenticating themselves using X.509 certificates through Transport
   Layer Security (TLS). For this reason, we only consider connection
   reuse for TLS over TCP and TLS over Stream Control Transmission
   Protocol (SCTP). This document also provides guidelines on
   connection reuse and virtual SIP servers and the interaction of
   connection reuse and DNS SRV lookups in SIP.

Status of This Memo

   This is an Internet Standards Track document.

   This document is a product of the Internet Engineering Task Force
   (IETF). It represents the consensus of the IETF community. It has
   received public review and has been approved for publication by the
   Internet Engineering Steering Group (IESG). Further information on
   Internet Standards is available in Section 2 of RFC 5741.

   Information about the current status of this document, any errata,
   and how to provide feedback on it may be obtained at
   http://www.rfc-editor.org/info/rfc5923.

Copyright Notice

Table of Contents

1. Introduction

   SIP entities can communicate using either unreliable/connectionless
   (e.g., UDP) or reliable/connection-oriented (e.g., TCP, SCTP
   [RFC4960]) transport protocols. When SIP entities use a connection-
   oriented protocol (such as TCP or SCTP) to send a request, they
   typically originate their connections from an ephemeral port.

   In the following example, A listens for SIP requests over TLS on TCP
   port 5061 (the default port for SIP over TLS over TCP), but uses an
   ephemeral port (port 49160) for a new connection to B. These
   entities could be SIP user agents or SIP proxy servers.

```
        +-----------+ 49160 (UAC)     5061 (UAS) +-----------+
        |           |--------------------------->|           |
        |  Entity   |                             |  Entity   |
        |    A      |                             |    B      |
        |           | 5061 (UAS)                  |           |
        +-----------+                             +-----------+
```
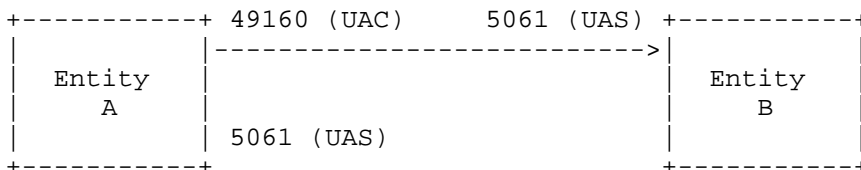
        Figure 1: Uni-directional connection for requests from A to B

   The SIP protocol includes the notion of a persistent connection
   (defined in Section 2), which is a mechanisms to insure that
   responses to a request reuse the existing connection that is
   typically still available, as well as reusing the existing
   connections for other requests sent by the originator of the
   connection. However, new requests sent in the backwards direction --
   in the example above, requests from B destined to A -- are unlikely
   to reuse the existing connection. This frequently causes a pair of
   SIP entities to use one connection for requests sent in each
   direction, as shown below.

```
        +-----------+ 49160             5061 +-----------+
        |           |......................>|           |
        |  Entity   |                        |  Entity   |
        |    A      | 5061             49170 |    B      |
        |           |<---------------------- |           |
        +-----------+                        +-----------+
```

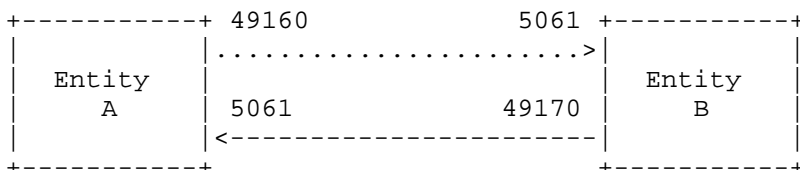   Figure 2: Two connections for requests between A and B

   Unlike TCP, TLS connections can be reused to send requests in the
   backwards direction since each end can be authenticated when the
   connection is initially set up. Once the authentication step has
   been performed, the situation can thought to resemble the picture in
   Figure 1 except that A and B both use a single shared connection, for

example, between port 49160 on A and port 5061 on B. When A wants to
send a request to B, it will reuse this connection, and when B wants
to send a request to A, it will reuse the same connection.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
"SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
document are to be interpreted as described in RFC 2119 [RFC2119].

Additional terminology used in this document:

Advertised address: The address that occurs in the Via header
   field's sent-by production rule, including the port number and
   transport.

Alias: Reusing an existing connection to send requests in the
   backwards direction; i.e., A opens a connection to B to send a
   request, and B uses that connection to send requests in the
   backwards direction to A.

Connection reuse: See "Alias".

Persistent connection: The process of sending multiple, possibly
   unrelated requests on the same connection, and receiving responses
   on that connection as well. More succinctly, A opens a connection
   to B to send a request, and later reuses the same connection to
   send other requests, possibly unrelated to the dialog established
   by the first request. Responses will arrive over the same
   connection. Persistent connection behavior is specified in
   Section 18 of RFC 3261 [RFC3261]. Persistent connections do not
   imply connection reuse.

Resolved address: The network identifiers (IP address, port,
   transport) associated with a user agent as a result of executing
   RFC 3263 [RFC3263] on a Uniform Resource Identifier (URI).

Shared connection: See "Persistent connection".

3. Applicability Statement

   The applicability of the mechanism described in this document is for
   two adjacent SIP entities to reuse connections when they are agnostic
   about the direction of the connection, i.e., either end can initiate
   the connection. SIP entities that can only open a connection in a
   specific direction -- perhaps because of Network Address Translation
   (NAT) and firewalls -- reuse their connections using the mechanism
   described in the outbound document [RFC5626].

   This memo concerns connection reuse, not persistent connections (see
   definitions of these in Section 2). Behavior for persistent
   connections is specified in Section 18 of RFC 3261 [RFC3261] and is
   not altered by this memo.

   This memo documents that it is good practice to only reuse those
   connections where the identity of the sender can be verified by the
   receiver. Thus, TLS (RFC 5246 [RFC5246]) connections (over any
   connection-oriented transport) formed by exchanging X.509
   certificates can be reused because they authoritatively establish
   identities of the communicating parties (see Section 5).

4. Benefits of TLS Connection Reuse

   Opening an extra connection where an existing one is sufficient can
   result in potential scaling and performance problems. Each new
   connection using TLS requires a TCP three-way handshake, a handful of
   round trips to establish TLS, typically expensive asymmetric
   authentication and key generation algorithms, and certificate
   verification. This can lead to a build up of considerable queues as
   the server CPU saturates by the TLS handshakes it is already
   performing (Section 6.19 of Rescorla [Book-Rescorla-TLS]).

   Consider the call flow shown below where Proxy A and Proxy B use the
   Record-Route mechanism to stay involved in a dialog. Proxy B will
   establish a new TLS connection just to send a BYE request.

```
                      Proxy A     Proxy B
                         |        |
    Create connection 1 +---INV--->|
                         |        |
                         |<---200---+ Response over connection 1
                         |        |
    Reuse connection 1  +---ACK--->|
                         |        |
                         =        =
                         |        |
                         |<---BYE---+ Create connection 2
                         |        |
         Response over  +---200--->|
         connection 2
```
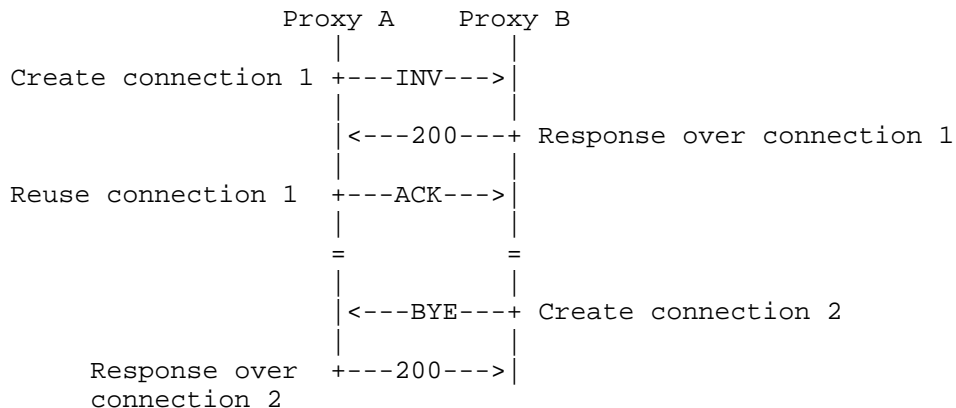
                Figure 3: Multiple connections for requests

   Setting up a second connection (from B to A above) for subsequent
   requests, even requests in the context of an existing dialog (e.g.,
   re-INVITE request or BYE request after an initial INVITE request, or
   a NOTIFY request after a SUBSCRIBE request or a REFER request), can
   also cause excessive delay (especially in networks with long round-
   trip times). Thus, it is advantageous to reuse connections whenever
   possible.

   From the user expectation point of view, it is advantageous if the
   re-INVITE requests or UPDATE requests are handled automatically and
   rapidly in order to avoid media and session state from being out of
   step. If a re-INVITE request requires a new TLS connection, the re-
   INVITE request could be delayed by several extra round-trip times.
   Depending on the round-trip time, this combined delay could be
   perceptible or even annoying to a human user. This is especially
   problematic for some common SIP call flows (for example, the
   recommended example flow in Figure 4 in RFC 3725 [RFC3725] uses many
   re-INVITE requests).

   The mechanism described in this document can mitigate the delays
   associated with subsequent requests.

5. Overview of Operation

   This section is tutorial in nature, and does not specify any
   normative behavior.

We now explain this working in more detail in the context of
communication between two adjacent proxies. Without any loss of
generality, the same technique can be used for connection reuse
between a User Agent Client (UAC) and an edge proxy, or between an
edge proxy and a UAS, or between an UAC and an UAS.

P1 and P2 are proxies responsible for routing SIP requests to user
agents that use them as edge proxies (see Figure 4).

```
              P1 <===================> P2
          p1.example.com            p2.example.net
            (192.0.2.1)               (192.0.2.128)


     +---+                                     +---+
     |   |    0---0              0---0    |   |
     |___|    /-\                /-\      |___|
    /    /   +---+              +---+    /    /
   +----+                              +----+
     User Agents                        User Agents
     example.com domain                 example.net domain
```
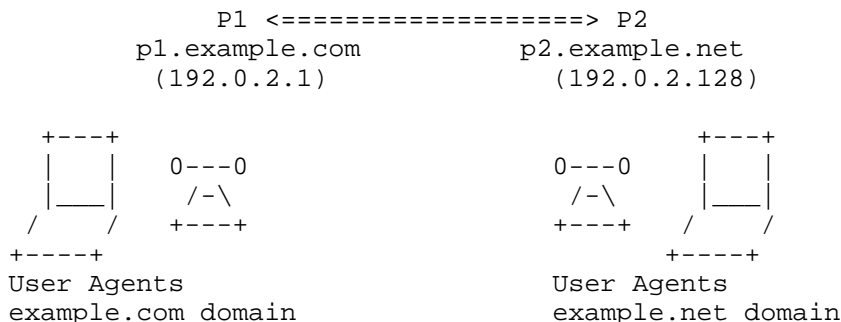
Figure 4: Proxy setup

For illustration purpose the discussion below uses TCP as a transport
for TLS operations. Another streaming transport -- such as SCTP --
can be used as well.

The act of reusing a connection is initiated by P1 when it adds an
"alias" header field parameter (defined later) to the Via header
field. When P2 receives the request, it examines the topmost Via
header field. If the Via header contained an "alias" header field
parameter, P2 establishes a binding such that subsequent requests
going to P1 will reuse the connection; i.e., requests are sent over
the established connection.

With reference to Figure 4, in order for P2 to reuse a connection for
requests in the backwards direction, it is important that the
validation model for requests sent in this direction (i.e., P2 to P1)
is equivalent to the normal "connection in each direction" model,
wherein P2 acting as client would open up a new connection in the
backwards direction and validate the connection by examining the
X.509 certificate presented. The act of reusing a connection needs
the desired property that requests get delivered in the backwards
direction only if they would have been delivered to the same
destination had connection reuse not been employed. To guarantee
this property, the X.509 certificate presented by P1 to P2 when a TLS
connection is first authenticated are cached for later use.

To aid the discussion of connection reuse, this document defines a
data structure called the connection alias table (or simply, alias
table), which is used to store aliased addresses and is used by user
agents to search for an existing connection before a new one is
opened up to a destination. It is not the intent of this memo to
standardize the implementation of an alias table; rather, we use it
as a convenience to aid subsequent discussions.

P1 gets a request from one of its upstream user agents, and after
performing RFC3263 [RFC3263] server selection, arrives at a resolved
address of P2. P1 maintains an alias table, and it populates the
alias table with the IP address, port number, and transport of P2 as
determined through RFC3263 server selection. P1 adds an "alias"
header field parameter to the topmost Via header field (inserted by
it) before sending the request to P2. The value in the sent-by
production rule of the Via header field (including the port number),
and the transport over which the request was sent becomes the
advertised address of P1:

Via: SIP/2.0/TLS p1.example.com;branch=z9hG4bKa7c8dze;alias

Assuming that P1 does not already have an existing aliased connection
with P2, P1 now opens a connection with P2. P2 presents its X.509
certificate to P1 for validation (see Section 9.1). Upon connection
authentication and acceptance, P1 adds P2 to its alias table. P1's
alias table now looks like:

Destination IP Address	Destination Port	Destination Transport	Destination Identity	Alias Descriptor
...				
192.0.2.128	5061	TLS	sip:example.net sip:p2.example.net	25

Subsequent requests that traverse from P1 to P2 will reuse this
connection; i.e., the requests will be sent over the descriptor 25.

The following columns in the alias table created at the client
warrant an explanation:

1. The IP address, port, and transport are a result of executing the
    RFC3263 server resolution process on a next-hop URI.

2. The entries in the fourth column consists of the identities of
    the server as asserted in the X.509 certificate presented by the
    server. These identities are cached by the client after the
    server has been duly authenticated (see Section 9.1).

3. The entry in the last column is the socket descriptor over which
    P1, acting as a client, actively opened a TLS connection. At
    some later time, when P1 gets a request from one of the user
    agents in its domain, it will reuse the aliased connection
    accessible through socket descriptor 25 if and only if all of the
    following conditions hold:

    A. P1 determines through the RFC3263 server resolution process
        that the {transport, IP-address, port} tuple of P2 to be
        {TLS, 192.0.2.128, 5061}, and

    B. The URI used for the RFC3263 server resolution matches one of
        the identities stored in the cached certificate (fourth
        column).

When P2 receives the request, it examines the topmost Via header
field to determine whether P1 is willing to use this connection as an
aliased connection (i.e., accept requests from P2 towards P1). The
Via header field at P2 now looks like the following (the "received"
header field parameter is added by P2):

Via: SIP/2.0/TLS p1.example.com;branch=z9hG4bKa7c8dze;alias;
  received=192.0.2.1

The presence of the "alias" Via header field parameter indicates that
P1 supports aliasing on this connection. P2 now authenticates the
connection (see Section 9.2) and if the authentication was
successful, P2 creates an alias to P1 using the advertised address in
the topmost Via header field. P2's alias table looks like the
following:

Destination IP Address	Destination Port	Destination Transport	Destination Identity	Alias Descriptor
...				
192.0.2.1	5061	TLS	sip:example.com sip:p1.example.com	18

There are a few items of interest here:

1. The IP address field is populated with the source address of the
    client.

2. The port field is populated from the advertised address (topmost
    Via header field), if a port is present in it, or 5061 if it is
    not.

3. The transport field is populated from the advertised address
   (topmost Via header field).

4. The entries in the fourth column consist of the identities of the
   client as asserted in the X.509 certificate presented by the
   client. These identities are cached by the server after the
   client has been duly authenticated (see Section 9.2).

5. The entry in the last column is the socket descriptor over which
   the connection was passively accepted. At some later time, when
   P2 gets a request from one of the user agents in its domain, it
   will reuse the aliased connection accessible through socket
   descriptor 18 if and only if all of the following conditions
   hold:

   A. P2 determines through RFC3263 server resolution process that
      the {transport, IP-address, port} tuple of P1 to be {TLS,
      192.0.2.1, 5061}, and

   B. The URI used for RFC3263 server resolution matches one of the
      identities stored in the cached certificate (fourth column).

6. The network address inserted in the "Destination IP Address"
   column is the source address as seen by P2 (i.e., the "received"
   header field parameter). It could be the case that the host name
   of P1 resolves to different IP addresses due to round-robin DNS.
   However, the aliased connection is to be established with the
   original sender of the request.

6. Requirements

   The following are the requirements that motivated this specification:

   1. A connection sharing mechanism should allow SIP entities to reuse
      existing connections for requests and responses originated from
      either peer in the connection.

   2. A connection sharing mechanism must not require clients to send
      all traffic from well-know SIP ports.

   3. A connection sharing mechanism must not require configuring
      ephemeral port numbers in DNS.

   4. A connection sharing mechanism must prevent unauthorized
      hijacking of other connections.

   5. Connection sharing should persist across SIP transactions and
      dialogs.

   6. Connection sharing must work across name-based virtual SIP
       servers.

   7. There is no requirement to share a complete path for ordinary
       connection reuse. Hop-by-hop connection sharing is more
       appropriate.

7. Formal Syntax

   The following syntax specification uses the augmented Backus-Naur
   Form (BNF) as described in RFC 5234 [RFC5234]. This document extends
   the via-params to include a new via-alias defined below.

      via-params =/ via-alias
      via-alias = "alias"

8. Normative Behavior

8.1. Client Behavior

   Clients SHOULD keep connections up as long as they are needed.
   Connection reuse works best when the client and the server maintain
   their connections for long periods of time. Clients, therefore,
   SHOULD NOT automatically drop connections on completion of a
   transaction or termination of a dialog.

   The mechanism for connection reuse uses a new Via header field
   parameter. The "alias" header field parameter is included in a Via
   header field value to indicate that the client wants to create a
   transport layer alias. The client places its advertised address in
   the Via header field value (in the sent-by production).

   If the client places an "alias" header field parameter in the topmost
   Via header of the request, the client SHOULD keep the connection open
   for as long as the resources on the host operating system allow it
   to, and that the client MUST accept requests over this connection --
   in addition to the default listening port -- from its downstream
   peer. And furthermore, the client SHOULD reuse the connection when
   subsequent requests in the same or different transactions are
   destined to the same resolved address.

      Note that RFC 3261 states that a response arrives over the same
      connection that was opened for a request.

Whether or not to allow an aliased connection ultimately depends on
the recipient of the request; i.e., the client does not get any
confirmation that its downstream peer created the alias, or indeed
that it even supports this specification. Thus, clients MUST NOT
assume that the acceptance of a request by a server automatically
enables connection aliasing. Clients MUST continue receiving
requests on their default port.

Clients MUST authenticate the connection before forming an alias;
Section 9.1 discusses the authentication steps in more detail. Once
the server has been authenticated, the client MUST cache, in the
alias table, the identity (or identities) of the server as determined
in Section 7.1 of RFC 5922 [RFC5922]. The client MUST also populate
the destination IP address, port, and transport of the server in the
alias table; these fields are retrieved from executing RFC3263 server
resolution process on the next-hop URI. And finally, the client MUST
populate the alias descriptor field with the connection handle (or
identifier) used to connect to the server.

Once the alias table has been updated with a resolved address, and
the client wants to send a new request in the direction of the
server, the client reuses the connection only if all of the following
conditions hold:

1. The client uses the RFC3263 resolution on a URI and arrives at a
    resolved address contained in the alias table, and

2. The URI used for RFC3263 server resolution matches one of the
    identities stored in the alias table row corresponding to that
    resolved address.

Clients MUST be prepared for the case that the connection no longer
exists when they are ready to send a subsequent request over it. In
such a case, a new connection MUST be opened to the resolved address
and the alias table updated accordingly.

This behavior has an adverse side effect when a CANCEL request or an
ACK request for a non-2xx response is sent downstream. Normally,
these would be sent over the same connection over which the INVITE
request was sent. However, if between the sending of the INVITE
request and subsequent sending of the CANCEL request or ACK request
to a non-2xx response, the connection was closed, then the client
SHOULD open a new connection to the resolved address and send the
CANCEL request or ACK request there instead. The client MAY insert
the newly opened connection into the alias table.

8.2. Server Behavior

   Servers SHOULD keep connections up unless they need to reclaim
   resources. Connection reuse works best when the client and the
   server maintain their connections for long periods of time. Servers,
   therefore, SHOULD NOT automatically drop connections on completion of
   a transaction or termination of a dialog.

   When a server receives a request over TLS whose topmost Via header
   field contains an "alias" header field parameter, it signifies that
   the upstream client will leave the connection open beyond the
   transaction and dialog lifetime, and that subsequent transactions and
   dialogs that are destined to a resolved address that matches the
   identifiers in the advertised address in the topmost Via header field
   can reuse this connection.

   Whether or not to use in the reverse direction a connection marked
   with the "alias" Via header field parameter ultimately depends on the
   policies of the server. It can choose to honor it, and thereby send
   subsequent requests over the aliased connection. If the server
   chooses not to honor an aliased connection, the server MUST allow the
   request to proceed as though the "alias" header field parameter was
   not present in the topmost Via header.

      This assures interoperability with RFC3261 server behavior.
      Clients can include the "alias" header field parameter without
      fear that the server will reject the SIP request because of its
      presence.

   Servers MUST be prepared to deal with the case that the aliased
   connection no longer exist when they are ready to send a subsequent
   request over it. This can happen if the peer ran out of operating
   system resources and had to close the connection. In such a case,
   the server MUST open a new connection to the resolved address and the
   alias table updated accordingly.

   If the sent-by production of the Via header field contains a port,
   the server MUST use it as a destination port. Otherwise, the default
   port is the destination port.

   Servers MUST follow the authentication steps outlined in Section 9.2
   to authenticate the connection before forming an alias.

   The server, if it decides to reuse the connection, MUST cache in the
   alias table the identity (or identities) of the client as they appear
   in the X.509 certificate subjectAlternativeName extension field. The
   server also populates the destination IP address, port, and transport
   in the alias table from the topmost Via header field (using the

";received" parameter for the destination IP address). If the port
number is omitted, a default port number of 5061 is to be used. And
finally, the server populates the alias descriptor field with the
connection handle (or identifier) used to accept the connection from
the client (see Section 5 for the contents of the alias table).

Once the alias table has been updated, and the server wants to send a
request in the direction of the client, it reuses the connection only
if all of the following conditions hold:

1. The server, which acts as a client for this transaction, uses the
    RFC3263 resolution process on a URI and arrives at a resolved
    address contained in the alias table, and

2. The URI used for RFC3263 server resolution matches one of the
    identities stored in the alias table row corresponding to that
    resolved address.

8.3. Closing a TLS connection

Either the client or the server may terminate a TLS session by
sending a TLS closure alert. Before closing a TLS connection, the
initiator of the closure MUST either wait for any outstanding SIP
transactions to complete, or explicitly abandon them.

After the initiator of the close has sent a closure alert, it MUST
discard any TLS messages until it has received a similar alert from
its peer. The receiver of the closure alert MUST NOT start any new
SIP transactions after the receipt of the closure alert.

9. Security Considerations

This document presents requirements and a mechanism for reusing
existing connections easily. Unauthenticated connection reuse would
present many opportunities for rampant abuse and hijacking.
Authenticating connection aliases is essential to prevent connection
hijacking. For example, a program run by a malicious user of a
multiuser system could attempt to hijack SIP requests destined for
the well-known SIP port from a large relay proxy.

9.1. Authenticating TLS Connections: Client View

When a TLS client establishes a connection with a server, it is
presented with the server's X.509 certificate. Authentication
proceeds as described in Section 7.3 ("Client behavior") of RFC 5922
[RFC5922].

9.2. Authenticating TLS Connections: Server View

   A TLS server conformant to this specification MUST ask for a client
   certificate; if the client possesses a certificate, it will be
   presented to the server for mutual authentication, and authentication
   proceeds as described in Section 7.4 ("Server behavior") of RFC 5922
   [RFC5922].

   If the client does not present a certificate, the server MUST proceed
   as if the "alias" header field parameter was not present in the
   topmost Via header. In this case, the server MUST NOT update the
   alias table.

9.3. Connection Reuse and Virtual Servers

   Virtual servers present special considerations for connection reuse.
   Under the name-based virtual server scheme, one SIP proxy can host
   many virtual domains using one IP address and port number. If
   adequate defenses are not put in place, a connection opened to a
   downstream server on behalf of one domain can be reused to send
   requests in the backwards direction to a different domain. The
   "Destination Identity" column in the alias table has been added to
   aid in such defenses.

   Virtual servers MUST only perform connection reuse for TLS
   connections; virtual servers MUST NOT perform connection reuse for
   other connection-oriented transports. To understand why this is the
   case, note that the alias table caches not only which connections go
   to which destination addresses, but also which connections have
   authenticated themselves as responsible for which domains. If a
   message is to be sent in the backwards direction to a new SIP domain
   that resolves to an address with a cached connection, the cached
   connection cannot be used because it is not authenticated for the new
   domain.

   As an example, consider a proxy P1 that hosts two virtual domains --
   example.com and example.net -- on the same IP address and port.
   RFC3263 server resolution is set up such that a DNS lookup of
   example.com and example.net both resolve to an {IP-address, port,
   transport} tuple of {192.0.2.1, 5061, TLS}. A user agent in the
   example.com domain sends a request to P1 causing it to make a
   downstream connection to its peering proxy, P2, and authenticating
   itself as a proxy in the example.com domain by sending it a X.509
   certificate asserting such an identity. P2's alias table now looks
   like the following:

Destination IP Address	Destination Port	Destination Transport	Destination Identity	Alias Descriptor
...				
192.0.2.1	5061	TLS	sip:example.com	18

At some later point in time, a user agent in P2's domain wants to
send a request to a user agent in the example.net domain. P2
performs an RFC3263 server resolution process on sips:example.net to
derive a resolved address tuple {192.0.2.1, 5061, TLS}. It appears
that a connection to this network address is already cached in the
alias table; however, P2 cannot reuse this connection because the
destination identity (sip:example.com) does not match the server
identity used for RFC3261 resolution (sips:example.net). Hence, P2
will open up a new connection to the example.net virtual domain
hosted on P1. P2's alias table will now look like:

Destination IP Address	Destination Port	Destination Transport	Destination Identity	Alias Descriptor
...				
192.0.2.1	5061	TLS	sip:example.com	18
192.0.2.1	5061	TLS	sip:example.net	54

The identities conveyed in an X.509 certificate are associated with a
specific TLS connection. Absent such a guarantee of an identity tied
to a specific connection, a normal TCP or SCTP connection cannot be
used to send requests in the backwards direction without a
significant risk of inadvertent (or otherwise) connection hijacking.

The above discussion details the impact on P2 when connection reuse
is desired for virtual servers. There is a subtle, but important
impact on P1 as well.

P1 should keep separate alias tables for the requests served from the
UAs in the example.com domain from those served by the UAs in the
example.net domain. This is so that the boundary between the two
domains is preserved; P1 MUST NOT open a connection on behalf of one
domain and reuse it to send a new request on behalf of another
domain.

10. Connection Reuse and SRV Interaction

   Connection reuse has an interaction with the DNS SRV load balancing
   mechanism. To understand the interaction, consider the following
   figure:

```
          /+---- S1
+-------+/
| Proxy |------- S2
+-------+\
          \+---- S3
```
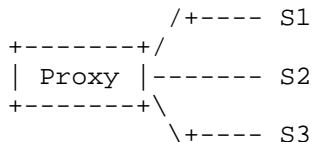
                       Figure 5: Load balancing

   Here, the proxy uses the DNS SRV to load balance across the three
   servers, S1, S2, and S3. Using the connect reuse mechanism specified
   in this document, over time the proxy will maintain a distinct
   aliased connection to each of the servers. However, once this is
   done, subsequent traffic is load balanced across the three downstream
   servers in the normal manner.

11. IANA Considerations

   This specification defines a new Via header field parameter called
   "alias" in the "Header Field Parameters and Parameter Values" sub-
   registry as per the registry created by RFC 3968 [RFC3968]. The
   required information is:

Header Field	Parameter Name	Predefined Values	Reference
Via	alias	No	RFC5923

12. Acknowledgments

   Thanks to Jon Peterson for helpful answers about certificate behavior
   with SIP, Jonathan Rosenberg for his initial support of this concept,
   and Cullen Jennings for providing a sounding board for this idea.
   Other members of the SIP WG that contributed to this document include
   Jeroen van Bemmel, Keith Drage, Matthew Gardiner, Rajnish Jain, Benny
   Prijono, and Rocky Wang.

   Dale Worley and Hadriel Kaplan graciously performed a WGLC review of
   the document. The resulting revision has benefited tremendously from
   their feedback.

13. References

13.1. Normative References

   [RFC3261] 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.

   [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
              Requirement Levels", BCP 14, RFC 2119, March 1997.

   [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security
              (TLS) Protocol Version 1.2", RFC 5246, August 2008.

   [RFC3263] Rosenberg, J. and H. Schulzrinne, "Session Initiation
              Protocol (SIP): Locating SIP Servers", RFC 3263,
              June 2002.

   [RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax
              Specifications: ABNF", RFC 5234, January 2008.

   [RFC5922] Gurbani, V., Lawrence, S., and B. Laboratories, "Domain
              Certificates in the Session Initiation Protocol (SIP)",
              RFC 5922, June 2010.

13.2. Informative References

   [RFC3968] Camarillo, G., "The Internet Assigned Number Authority
              (IANA) Header Field Parameter Registry for the Session
              Initiation Protocol (SIP)", BCP 98, RFC 3968,
              December 2004.

   [RFC5626] Jennings, C., Mahy, R., and F. Audet, "Managing Client-
              Initiated Connections in the Session Initiation Protocol
              (SIP)", RFC 5626, October 2009.

   [Book-Rescorla-TLS]
              Rescorla, E., "SSL and TLS: Designing and Building Secure
              Systems", Addison-Wesley Publishing, 2001.

   [RFC3725] Rosenberg, J., Peterson, J., Schulzrinne, H., and G.
              Camarillo, "Best Current Practices for Third Party Call
              Control (3pcc) in the Session Initiation Protocol (SIP)",
              BCP 85, RFC 3725, April 2004.

   [RFC4960] Stewart, R., "Stream Control Transmission Protocol",
              RFC 4960, September 2007.

Authors' Addresses

   Vijay K. Gurbani (editor)
   Bell Laboratories, Alcatel-Lucent

   EMail: vkg@alcatel-lucent.com

   Rohan Mahy
   Unaffiliated

   EMail: rohan@ekabal.com

   Brett Tate
   BroadSoft

   EMail: brett@broadsoft.com