SIP WG
Internet-Draft
Updates: 3261 (if approved)
Intended status: Standards Track
Expires: April 18, 2008

R. Mahy
Plantronics
V. Gurbani, Ed.
Bell Laboratories, Alcatel-Lucent
B. Tate
BroadSoft

October 16, 2007

Connection Reuse in the Session Initiation Protocol (SIP)
draft-ietf-sip-connect-reuse-08

Status of this Memo

Copyright Notice

Abstract

This document enables a pair of communicating proxies to reuse a congestion-controlled connection between themselves for sending requests in the forward and backwards direction. Because the connection is essentially aliased for requests going in the backwards direction, reuse should be predicated upon both the communicating endpoints authenticating themselves using X.509 certificates through TLS. For this reason, we only consider connection reuse for TLS over TCP and TLS over SCTP. A single connection cannot be reused for the TCP or SCTP transport between two peers, and this document provides insight into why this is the case. As a remedy, it suggests using two TCP connections (or two SCTP associations), each opened pro-actively towards the recipient by the sender. Finally, this document also provides guidelines on connection reuse and virtual SIP servers and the interaction of connection reuse and DNS SRV lookups in SIP.

Table of Contents

1. Terminology 3
2. Applicability Statement 3
3. Introduction 3
4. Benefits of TLS Connection Reuse 4
5. Overview of Operation 5
  5.1. TLS Operations 6
  5.2. TCP Operations 9
  5.3. SCTP Operations 10
6. Requirements 10
7. Formal Syntax 10
8. Normative Behavior 11
  8.1. Client Behavior 11
  8.2. Server Behavior 13
9. Security Considerations 14
  9.1. Authenticating TLS Connections: Client View 14
  9.2. Authenticating TLS Connections: Server View 14
  9.3. Security Considerations for the TCP Transport 15
10. Support for Virtual Servers 16
  10.1. Virtual Servers and TLS Connections 16
11. Connection Reuse and SRV Interaction 18
12. IANA Considerations 18
13. Acknowledgments 18
14. References 19
  14.1. Normative References 19
  14.2. Informational References 19
Authors' Addresses 20
Intellectual Property and Copyright Statements 21

1. 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 [2].

Additional terminology used in this document:

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

Alias: A transport layer connection associated with a resolved address.

Resolved address: The network identifiers (IP address, port, transport) associated with a user agent as a result of executing RFC3263 [4] on a Uniform Resource Identifier (URI).

2. 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 firewall reasons -- reuse their connections using the mechanism described in [9].

3. Introduction

SIP [1] entities can communicate using either unreliable/ connectionless (e.g., UDP) or reliable/connection-oriented (e.g., TCP, SCTP [15]) 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, Entity A listens for SIP requests over TLS [3] on TCP port 5061 (the default port for SIP over TLS over TCP), but uses an ephemeral port (port 8293) for a new connection to Entity B. These entities could be SIP User Agents or SIP Proxy Servers.

```
+-----------+ 8293 (UAC)      5061 (UAS) +-----------+
|           |--------------------------->|           |
|  Entity   |                            |  Entity   |
|     A     |                            |     B     |
|           | 5061 (UAS)                 |           |
+-----------+                            +-----------+
```

Figure 1: Uni-directional connection for requests from A to B.

The SIP protocol includes mechanisms which insure that responses to a request reuse the existing connection which is typically still available, and also includes provisions for reusing existing connections for other requests sent by the originator of the connection. However, new requests sent in the opposite 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.

```
+-----------+ 8293              5061 +-----------+
|           |.......................>|           |
|  Entity   |                        |  Entity   |
|     A     | 5061              9741 |     B     |
|           |<-----------------------|           |
+-----------+                        +-----------+
```

Figure 2: Two connections for requests between A and B.

While this is adequate for TCP, and indeed is the only way to securely do connection reuse over that transport (see Section 9.3), TLS connections can be reused 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 the connection opened from Entity A to Entity B is shared. When Entity A wants to send a request to Entity B, it will reuse this connection, and when Entity B wants to send a request to Entity A, it will reuse the same connection.

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 3-way handshake, a handful of round-trips to establish TLS, typically expensive asymmetric authentication and key generation algorithms, and certificate verification. This may 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 [10]).

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
                         |          |
     Re-use 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 or BYE after an initial INVITE, or a NOTIFY after a SUBSCRIBE [14] or a REFER [13]), 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-INVITEs or UPDATE [11] requests are handled automatically and rapidly in order to avoid media and session state from being out of step. If a re-INVITE requires a new TLS connection, the reINVITE 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 number 4 in RFC3725 [12] use many reINVITEs).

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, it should be clear that the same technique can be used for connection reuse between a 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 through 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.

5.1. TLS Operations

For illustration purpose, the discussion below uses TCP as a transport for TLS operations. Another streaming transport -- such as SCTP [15] -- can be used as well.

The act of reusing a connection is initiated by an user agent client (UAC) when it adds an "alias" parameter (defined later) to the Via header. When a user agent server (UAS) receives the request, it examines the topmost Via header. If the header contained an "alias" parameter, the UAS establishes a binding such that subsequent requests going to the UAC 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 opposite direction, it is important to note that the validation model for requests sent in this direction (i.e., P2 to P1) should be 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 must have the desired property that requests get delivered in the reverse 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 must be 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. User agents can consult the alias table for an existing connection before opening up

a new one.

P1 gets a request from one of its upstream user agents, and after performing 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" parameter to the topmost Via header (inserted by it) before sending the request to P2. The value in the sent-by production rule of the Via header (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 P2s to its alias table. P1's alias table now looks like:

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

Figure 5: Alias table at the client.

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

There are three items of interest in the alias table created at the client:

1. The IP address, port and transport are a result of executing 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 RFC3263 server resolution process that the request should be sent to P2 on port 5061 using TLS, and

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

When the server, P2, receives the request, it examines the topmost Via to determine whether P1 supports aliased connections. The Via at P2 now looks like (the "received" 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" parameter indicates that P1 does support aliasing. 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. P2's alias table looks like:

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

Figure 6: Alias table at the server.

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), 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).
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 request should be sent to P1 on port 5061 using TLS, 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 should be the source address as seen by P2 (i.e., the "received" 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.

5.2. TCP Operations

Connection reuse on the TCP transport is done differently from the TLS case. This is to prevent a service hijacking security attack outlined in Section 9.3.

In TCP, connection reuse is accomplished by each host pro-actively opening up a TCP connection towards its neighbor. Thus, two TCP connections will be needed between an adjacent pair of hosts, as depicted in Figure 2.

The presence of the "alias" parameter in the topmost Via for a TCP transport is not required.

From an operations point of view, the same data structure used to maintain TLS connections can be used for TCP connections as well. For TCP connections, the contents of this table will be slightly different in two ways: first, the "Destination Transport" will be "TCP", and second, the "Destination Identity" is null, or empty.

With reference to Figure 4, P1 gets a request from one of its upstream user agents, and after performing RFC3263 server selection, arrives at the resolved address of P2. P1 populates the alias table with the IP address, port number, and transport of P2 as determined through RFC3263 server selection. The value of the sent-by production rule of the Via header (including the port number), and the transport over which the request was sent becomes the advertised address of P1:

```
Via: SIP/2.0/TCP p1.example.com;branch=z9hG4bKa7c8dze
```

Assuming that P1 does not already have an existing entry with P2's resolved address in the alias table, P1 now opens up a new TCP connection with P2. When P2 accepts the connection, 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	5060	TCP	-	32

Figure 7: Alias table at the client for TCP transport.

Because this same TCP connection cannot be used to send requests from

P2 to P1, P2 will not update its alias table. Instead, at a later time, if a request from P2 is destined towards P1, P2 will actively open up a new TCP connection towards P1, and update its alias table accordingly. Once this is done, P1 and P2 will reuse the same connections that they established proactively for subsequent requests.

5.3. SCTP Operations

Operations on SCTP associations that are not protected by TLS are susceptible to the same session hijacking scenario outlined in Section 9.3. Thus, SCTP association reuse on associations not protected by TLS must mirror how connection reuse is done for TCP connections (see Section 5.2).

Operations on SCTP associations that are protected by TLS (i.e., the topmost Via header at the receiving host has a sent-by transport value of "SCTP-TLS" (see [7]) can be reused freely in the manner depicted in Section 5.1. This is because the SCTP association is authenticated at both ends, thus allowing it to be reused.

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 4234 [5]. This document extends the

via-params to include a new via-alias defined below.

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

8. Normative Behavior

This document specifies how to reuse connections. It is RECOMMENDED that servers keep connections up unless they need to reclaim resources, and that clients 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. SIP entities therefore SHOULD NOT automatically drop connections on completion of a transaction or termination of a dialog.

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.

> Note that 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 that the INVITE request was sent over. However, if between the sending of the INVITE and subsequent sending of the CANCEL or ACK to a non-2xx response, the connection was reclaimed, then the client SHOULD open a new connection to the resolved address and send the CANCEL or ACK there instead. The newly opened connection MAY be inserted into the alias table.

8.1. Client Behavior

For the TCP and SCTP transport, when the client executes the RFC3263 server selection mechanism to arrive at an IP address, port, and transport tuple to send the request to, it updates the alias table with this information. Subsequent requests that resolve to the same IP address, port, and transport tuple MUST reuse the same connection. The client must keep the connection open for as long as the resources on the operating system allow it to. It MUST only accept responses over this connection and MUST NOT accept any requests over this connection.

For TCP and SCTP transports, the client MUST NOT insert the "alias" parameter in the topmost Via header..

The rest of the discussion below applies to only the TLS transport over TCP or SCTP.

For TLS transports, the proposed mechanism uses a new Via header field parameter. The "alias" 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).

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

> Note that RFC3261 states that a response should arrive 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. They MUST continue receiving requests on their default port.

For TLS connections, 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 they appear in the X.509 certificate subjectAlternativeName extension field. 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 socket descriptor 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, it should reuse 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.

8.2. Server Behavior

A TCP connection, or a SCTP association accepted at the server is used by the server to only send responses upstream. It MUST NOT be used to send requests. Furthermore, if the topmost Via header of a request that arrived had the "alias" parameter in it, the server MUST NOT accord any semantics to this parameter and must behave as if the parameter was not present.

The rest of the discussion below applies to only the TLS transport.

When a server receives a request over TLS whose topmost Via header contains an "alias" 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 can reuse this connection.

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

> This assures interoperability with RFC3261 server behavior. Clients should feel comfortable including the "alias" 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 may happen if the peer ran out of operating system resources and had to close the connection. In such a case, a new connection MUST be opened to the resolved address and the alias table updated accordingly.

If the Via sent-by contains a port, it MUST be used as a destination port. Otherwise the default port is the destination port.

Servers must authenticate the connection before forming an alias. Section 9.2 discusses the authentication steps in more detail.

The server, if it decides to accept 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 must also populate the destination IP address,

port and transport in the alias table from the topmost Via header (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 must populate the alias descriptor field with the socket descriptor 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 should reuse 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.

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 5 of [8].

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 6 of [8].

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

9.3. Security Considerations for the TCP Transport

The mechanism for reusing TLS connections MUST NOT be used to reuse TCP connections or SCTP associations because there isn't any way to perform the authentication step. Instead, it is RECOMMENDED that TCP and SCTP peers who want to avail of connection reuse do so such that each peer actively opens up a connection in the direction of the other (as depicted in Figure 2). This manner of opening connections, while still not secure, is at least much more apparent and direct than using the connection reuse mechanism over TCP or SCTP in an unauthenticated fashion.

Connection reuse over TCP or SCTP is inherently insecure. Because the nature of the aliasing mechanism is such that it redirects requests destined for one port at a host to another port, service hi-jacking can result if adequate care is not taken to ensure that the redirected port is indeed authorized to receive the requests that would normally have gone to another, authorized port. Consider the following scenario to understand the service hi-jacking attack that can be mounted when using connection reuse over TCP. The scenarios depicts the attack using TCP as a transport, but the same result is acheived over SCTP as well.

A TCP server receives a request with the topmost Via header as follows (the "received" parameter is added by the server after getting the request):

```
Via: SIP/2.0/TCP uac.example.com;branch=z9hG4bKa7c8dze;
   received=192.0.4.33
```

If we were to allow TCP connection reuse in the same manner as TLS connection reuse, then the server would update its alias table such that whenever a request is destined to 192.0.4.33, port 5060, it will instead be sent to the peer at the end of the aliased connection. A security attack can now be mounted as follows: assume a malware program is running on a multi-user computer. The malware program knows that a user on the computer runs a SIP user agent, but the SIP user agent is currently not active (possibly by scanning ports on the local machine to seek a busy port 5060). Note that the malware program does not need to wait until the legitimate user agent was not running, however, doing so increases the chances that the server will not reject the malware program's request. Once the malware program decides that a legitimate user agent is not running, it sends a request to the server with an "alias" parameter. The server believes it is accepting a request from a legitimate user agent and sends subsequent requests to the aliased connection. The SIP service on the computer has now effectively been hi-jacked for the default port. The malware program does not need administrative privileges to

execute, and in fact, can masquerade as any user (legitimate or not) of the computer.

Later on, when the legitimate user agent is started, it may also send a request with an "alias" parameter to the server, which may detect that it now has two aliased connections. Making matters much worse, it cannot determine which of the two is the legitimate one and may well reject the request from the legitimate user.

In another form of this attack, the legitimate user agent may not support connection aliasing, but the malware program may use the mechanism to usurp the SIP service on the computer.

In yet another form of an attack, the malware program uses the aliasing mechanism to shortcut registering with a proxy to receive requests. In this case, it sends a request to the edge proxy (who may also substitute as the inbound proxy with access to a location service for that domain). In the request is a bogus request URI that will cause the edge proxy to fail the request, however, the edge proxy keeps the connection open and any subsequent requests destined to that host on the default port are instead sent to the malware program. Registration is thus not needed in order to receive incoming requests.

HTTP Digest is useful to mitigate only a subset of these attacks over TCP. For instance, HTTP Digest helps in authenticating a user agent to a proxy server before the alias table is updated. However, HTTP Digest is of no help when one proxy desires to enter an aliasing agreement with another downstream proxy.

10. Support for Virtual Servers

Virtual servers present special considerations for connection reuse. Under the name-based virtual server scheme, one SIP proxy may host many virtual domains. If adequate defenses are not put in place, a connection opened to a downstream server on behalf of one domain can be usurped by a malicious user in another domain. The Destination Identity column in the alias table has been added to aid in such defenses. If an implementation does not support virtual SIP servers, it MAY omit caching the identities in the alias table; however, if an implementation supports virtual SIP servers, then it MUST cache the identities in the alias table.

10.1. Virtual Servers and TLS Connections

To understand the specific problem associated with hijacking a TLS connection when virtual servers are used, consider a proxy P1 that

hosts two domains: atlanta.example.com and chicago.example.org. Also assume that the physical IP address of P1 is 192.168.0.1. Incoming requests to all the domains that P1 hosts arrive on port 5061.

A user, bob@atlanta.example.com, sends an instant message to a user Alice in a domain not hosted by P1. Alice's proxy establishes an alias to P1, thereby resulting in the following alias table (note: to illustrate the connection hijacking problem associated with virtual servers, the alias table below does not contain the Destination Identity column).

Destination IP Address	Destination Port	Destination Transport	Alias Descriptor
...			
192.168.0.1	5061	TLS	25

Figure 8: Alias table at Alice's Proxy.

At some later time, a user hosted by another virtual domain in P1, bob@chicago.example.org, sends an instant message to Alice. Alice's proxy will get the network identifiers from the topmost Via, and note that they are already in the alias table. Thinking that the newly arrived request is intended to replace the old (possibly stale) alias, it may update its alias table with the new descriptor.

Some time after that, Alice wants to send an instant message to sips:bob@atlanta.example.com. When RFC3263 resolution is done on sips:atlanta.example.com, the resolved address will match an entry in the alias table. But that entry is now aliased to a connection with bob@chicago.example.org. The end result of all this is that an instant message intended for bob@atlanta.example.com ends up in the inbox of bob@chicago.example.org.

It is to alleviate this very problem that the identities from the X.509 certificates are stored in the alias table and used to determine whether or not to reuse a connection. Saving the identities in the alias table mitigates this problem because Alice's proxy will actually form two aliased connections to P1: one row in the table will contain the resolved address of P1 but with an identity corresponding to atlanta.example.com and a second row will contain the same resolved address but with an identity corresponding to chicago.example.org. Now, when Alice's proxy wants to send a request in the backwards direction, it will match the URI used to do RFC3263 resolution to the appropriate identity before reusing the connection.

11. 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 9: Load balancing.

Here, the proxy uses 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.

12. 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 [6]. The required information is:

Header Field	Parameter Name	Predefined Values	Reference
Via	alias	No	RFCXXXX

RFC XXXX [NOTE TO RFC-EDITOR: Please replace with final RFC number of this specification.]

13. 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.

14. References

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

[3] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

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

[5] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, October 2005.

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

[7] Rosenberg, J., Schulzrinne, H., and G. Camarillo, "The Stream Control Transmission Protocol (SCTP) as a Transport for the Session Initiation Protocol (SIP)", RFC 4168, October 2005.

[8] Gurbani, V., Lawrence, S., and A. Jeffrey, "Domain Certificates in the Session Initiation Protocol (SIP)", draft-gurbani-sip-domain-certs-06 (work in progress), July 2007.

14.2. Informational References

[9] Jennings, C. and R. Mahy, "Managing Client Initiated Connections in the Session Initiation Protocol (SIP)", draft-ietf-sip-outbound-04.txt (work in progress), June 2006.

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

[11] Rosenberg, J., "The Session Initiation Protocol (SIP) UPDATE Method", RFC 3311, September 2002.

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

[13] Sparks, R., "The Session Initiation Protocol (SIP) Refer Method", RFC 3515, April 2003.

[14] Roach, A., "The Session Initiation Protocol (SIP)-Specific Event Notification", RFC 3265, June 2002.

[15] Stewart, R., Xie, Q., Morneault, K., Sharp, C., Schwarzbauer, H., Taylor, T., Rytina, I., Kalla, M., Zhang, L., and V. Paxson, "The Session Initiation Protocol (SIP)-Specific Event Notification", RFC 2960, October 2000.

Authors' Addresses

Rohan Mahy
Plantronics

Email: rohan@ekabal.com

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

Email: vkg at acm dot org

Brett Tate
BroadSoft

Email: brett@broadsoft.com

Full Copyright Statement

Intellectual Property

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).