        Comparison of Different NAT Traversal Techniques
   for Media Controlled by the Real-Time Streaming Protocol (RTSP)

Abstract

   This document describes several Network Address Translator (NAT)
   traversal techniques that were considered to be used for establishing
   the RTP media flows controlled by the Real-Time Streaming Protocol
   (RTSP). Each technique includes a description of how it would be
   used, the security implications of using it, and any other deployment
   considerations it has. There are also discussions on how NAT
   traversal techniques relate to firewalls and how each technique can
   be applied in different use cases. These findings were used when
   selecting the NAT traversal for RTSP 2.0, which is specified in a
   separate document.

Copyright Notice

Table of Contents

1. Introduction

   Today there is a proliferating deployment of different types of
   Network Address Translator (NAT) boxes that in many cases only
   loosely follow standards [RFC3022] [RFC2663] [RFC3424] [RFC4787]
   [RFC5382]. NATs cause discontinuity in address realms [RFC3424];
   therefore, an application protocol, such as the Real-Time Streaming
   Protocol (RTSP) [RFC2326] [RTSP], needs to deal with such
   discontinuities caused by NATs. The problem is that, being a media
   control protocol managing one or more media streams, RTSP carries
   network address and port information within its protocol messages.
   Because of this, even if RTSP itself, when carried over the
   Transmission Control Protocol (TCP) [RFC793], for example, is not
   blocked by NATs, its media streams may be blocked by NATs. This will
   occur unless special protocol provisions are added to support NAT
   traversal.

   Like NATs, firewalls are also middleboxes that need to be considered.
   Firewalls help prevent unwanted traffic from getting in or out of the
   protected network. RTSP is designed such that a firewall can be
   configured to let RTSP-controlled media streams go through with
   limited implementation effort. The effort needed is to implement an
   Application Level Gateway (ALG) to interpret RTSP parameters. There
   is also a large class of firewalls, commonly home firewalls, that use
   a filtering behavior that appears to be the same as what NATs have.
   This type of firewall will be successfully traversed using the same
   solution as employed for NAT traversal, instead of relying on an RTSP
   ALG. Therefore, firewalls will also be discussed and some important
   differences highlighted.

   This document describes several NAT traversal mechanisms for RTSP-
   controlled media streaming. Many of these NAT solutions fall into
   the category of "UNilateral Self-Address Fixing (UNSAF)" as defined
   in [RFC3424] and quoted below:

      [UNSAF] is a process whereby some originating process attempts to
      determine or fix the address (and port) by which it is known -
      e.g. to be able to use address data in the protocol exchange, or
      to advertise a public address from which it will receive
      connections.

   Following the guidelines spelled out in RFC 3424, we describe the
   required RTSP extensions for each method, transition strategies, and
   security concerns. The transition strategies are a discussion of how
   and if the method encourages a move towards not having any NATs on
   the path.

This document is capturing the evaluation done in the process to
recommend firewall/NAT traversal methods for RTSP streaming servers
based on [RFC2326] as well as the RTSP 2.0 core specification [RTSP].
The evaluation is focused on NAT traversal for the media streams
carried over the User Datagram Protocol (UDP) [RFC768] with RTP
[RFC3550] over UDP being the main case for such usage. The findings
should be applicable to other protocols as long as they have similar
properties.

At the time when the bulk of work on this document was done, a single
level of NAT was the dominant deployment for NATs, and multiple
levels of NATs, including Carrier-Grade NATs (CGNs), were not
considered. Thus, any characterizations or findings may not be
applicable in such scenarios, unless CGN or multiple levels of NATs
are explicitly noted.

An RTSP NAT traversal mechanism based on Interactive Connectivity
Establishment (ICE) is specified in "A Network Address Translator
(NAT) Traversal Mechanism for Media Controlled by Real-Time Streaming
Protocol (RTSP)" [RTSP-NAT].

1.1. Network Address Translators

We begin by reviewing two quotes from Section 3 in "Network Address
Translation (NAT) Behavioral Requirements for Unicast UDP" [RFC4787]
concerning NATs and their terminology:

   Readers are urged to refer to [RFC2663] for information on NAT
   taxonomy and terminology. Traditional NAT is the most common type
   of NAT device deployed. Readers may refer to [RFC3022] for
   detailed information on traditional NAT. Traditional NAT has two
   main varieties -- Basic NAT and Network Address/Port Translator
   (NAPT).

   NAPT is by far the most commonly deployed NAT device. NAPT allows
   multiple internal hosts to share a single public IP address
   simultaneously. When an internal host opens an outgoing TCP or
   UDP session through a NAPT, the NAPT assigns the session a public
   IP address and port number, so that subsequent response packets
   from the external endpoint can be received by the NAPT,
   translated, and forwarded to the internal host. The effect is
   that the NAPT establishes a NAT session to translate the (private
   IP address, private port number) tuple to a (public IP address,
   public port number) tuple, and vice versa, for the duration of the
   session. An issue of relevance to peer-to-peer applications is
   how the NAT behaves when an internal host initiates multiple
   simultaneous sessions from a single (private IP, private port)
   endpoint to multiple distinct endpoints on the external network.

      In this specification, the term "NAT" refers to both "Basic NAT"
      and "Network Address/Port Translator (NAPT)".

      This document uses the term "Address and Port Mapping" as the
      translation between an external address and port and an internal
      address and port. Note that this is not the same as an "address
      binding" as defined in RFC 2663.

      Note: In the above text, it would be more correct to use an
      external IP address instead of a public IP address. The external
      IP address is commonly a public one, but it might be of another
      type if the NAT's external side is in a private address domain.

   In addition to the above quote, there exists a number of address and
   port mapping behaviors described in more detail in Section 4.1 of
   [RFC4787] that are highly relevant to the discussion in this
   document.

   NATs also have a filtering behavior on traffic arriving on the
   external side. Such behavior affects how well different methods for
   NAT traversal works through these NATs. See Section 5 of [RFC4787]
   for more information on the different types of filtering that have
   been identified.

1.2. Firewalls

   A firewall is a security gateway that enforces certain access control
   policies between two network administrative domains: a private domain
   (intranet) and an external domain, e.g., the Internet. Many
   organizations use firewalls to prevent intrusions and malicious
   attacks on computing resources in the private intranet [RFC2588].

   A comparison between NAT and a firewall is given below:

   1. A firewall sits at security enforcement/protection points, while
       NAT sits at borders between two address domains.

   2. NAT does not in itself provide security, although some access
       control policies can be implemented using address translation
       schemes. The inherent filtering behaviors are commonly mistaken
       for real security policies.

   It should be noted that many NAT devices intended for Residential or
   Small Office, Home Office (SOHO) use include both NATs and firewall
   functionality.

1.3. Glossary

   Address-Dependent Mapping: The NAT reuses the port mapping for
        subsequent packets sent from the same internal IP address and
        port to the same external IP address, regardless of the
        external port; see [RFC4787].

   Address and Port-Dependent Mapping: The NAT reuses the port mapping
        for subsequent packets sent from the same internal IP address
        and port to the same external IP address and port while the
        mapping is still active; see [RFC4787].

   ALG: Application Level Gateway is an entity that can be embedded in
        a NAT or other middlebox to perform the application layer
        functions required for a particular protocol to traverse the
        NAT/middlebox.

   Endpoint-Independent Mapping: The NAT reuses the port mapping for
        subsequent packets sent from the same internal IP address and
        port to any external IP address and port; see [RFC4787].

   ICE: Interactive Connectivity Establishment; see [RFC5245].

   DNS: Domain Name Service

   DoS: Denial of Service

   DDoS: Distributed Denial of Service

   NAT: Network Address Translator; see [RFC3022].

   NAPT: Network Address/Port Translator; see [RFC3022].

   RTP: Real-Time Transport Protocol; see [RFC3550].

   RTSP: Real-Time Streaming Protocol; see [RFC2326] and [RTSP].

   RTT: Round Trip Times

   SDP: Session Description Protocol; see [RFC4566].

   SSRC: Synchronization source in RTP; see [RFC3550].

2. Detecting the Loss of NAT Mappings

   Several NAT traversal techniques in the next chapter make use of the
   fact that the NAT UDP mapping's external address and port can be
   discovered. This information is then utilized to traverse the NAT
   box. However, any such information is only good while the mapping is
   still valid. As the IAB's UNSAF document [RFC3424] points out, the
   mapping can either timeout or change its properties. It is therefore
   important for the NAT traversal solutions to handle the loss or
   change of NAT mappings, according to RFC 3424.

   First, since NATs may also dynamically reclaim or readjust address/
   port translations, "keep-alive" and periodic repolling may be
   required according to RFC 3424. Second, it is possible to detect and
   recover from the situation where the mapping has been changed or
   removed. The loss of a mapping can be detected when no traffic
   arrives for a while. Below we will give some recommendations on how
   to detect the loss of NAT mappings when using RTP/RTCP under RTSP
   control.

   An RTP session normally has both RTP and RTCP streams. The loss of
   an RTP mapping can only be detected when expected traffic does not
   arrive. If a client does not receive media data within a few seconds
   after having received the "200 OK" response to an RTSP PLAY request
   that starts the media delivery, it may be the result of a middlebox
   blocking the traffic. However, for a receiver to be more certain to
   detect the case where no RTP traffic was delivered due to NAT
   trouble, one should monitor the RTCP Sender reports if they are
   received and not also blocked. The sender report carries a field
   telling how many packets the server has sent. If that has increased
   and no RTP packets have arrived for a few seconds, it is likely the
   mapping for the RTP stream has been removed.

   The loss of mapping for RTCP is simpler to detect. RTCP is normally
   sent periodically in each direction, even during the RTSP ready
   state. If RTCP packets are missing for several RTCP intervals, the
   mapping is likely lost. Note that if neither RTCP packets nor RTSP
   messages are received by the RTSP server for a while (default 60
   seconds), the RTSP server has the option to delete the corresponding
   RTP session, SSRC and RTSP session ID, because either the client can
   not get through a middlebox NAT/firewall, or the client is
   malfunctioning.

3. Requirements on Solutions

   This section considers the set of requirements for the evaluation of
   RTSP NAT traversal solutions.

   RTSP is a client-server protocol. Typically, service providers
   deploy RTSP servers on the Internet or otherwise reachable address
   realm. However, there are use cases where the reverse is true: RTSP
   clients are connecting from any address realm to RTSP servers behind
   NATs, e.g., in a home. This is the case, for instance, when home
   surveillance cameras running as RTSP servers intend to stream video
   to cell phone users in the public address realm through a home NAT.
   In terms of requirements, the primary issue to solve is the RTSP NAT
   traversal problem for RTSP servers deployed in a network where the
   server is on the external side of any NAT, i.e., the server is not
   behind a NAT. The server behind a NAT is desirable but of much lower
   priority.

   Important considerations for any NAT traversal technique are whether
   any protocol modifications are needed and where the implementation
   burden resides (e.g., server, client, or middlebox). If the
   incentive to get RTSP to work over a NAT is sufficient, it will
   motivate the owner of the server, client, or middlebox to update,
   configure, or otherwise perform changes to the device and its
   software in order to support NAT traversal. Thus, the questions of
   who this burden falls on and how big it is are highly relevant.

   The list of feature requirements for RTSP NAT solutions are given
   below:

   1. Must work for all flavors of NATs, including NATs with Address
       and Port-Dependent Filtering.

   2. Must work for firewalls (subject to pertinent firewall
       administrative policies), including those with ALGs.

   3. Should have minimal impact on clients not behind NATs and that
       are not dual hosted. RTSP dual hosting means that the RTSP
       signaling protocol and the media protocol (e.g., RTP) are
       implemented on different computers with different IP addresses.

       * For instance, no extra protocol RTT before arrival of media.

   4. Should be simple to use/implement/administer so people actually
       turn them on.

       * Discovery of the address(es) assigned by NAT should happen
          automatically, if possible.

    5. Should authenticate dual-hosted client's media transport receiver
        to prevent usage of RTSP servers for DDoS attacks.

    The last requirement addresses the Distributed Denial-of-Service
    (DDoS) threat, which relates to NAT traversal as explained below.

    During NAT traversal, when the RTSP server determines the media
    destination (address and port) for the client, the result may be that
    the IP address of the RTP receiver host is different than the IP
    address of the RTSP client host. This poses a DDoS threat that has
    significant amplification potentials because the RTP media streams in
    general consist of a large number of IP packets. DDoS attacks can
    occur if the attacker can fake the messages in the NAT traversal
    mechanism to trick the RTSP server into believing that the client's
    RTP receiver is located on a host to be attacked. For example, user
    A may use his RTSP client to direct the RTSP server to send video RTP
    streams to target.example.com in order to degrade the services
    provided by target.example.com.

    Note that a simple mitigation is for the RTSP server to disallow the
    cases where the client's RTP receiver has a different IP address than
    that of the RTSP client. This is recommended behavior in RTSP 2.0
    unless other solutions to prevent this attack are present; see
    Section 21.2.1 in [RTSP]. With the increased deployment of NAT
    middleboxes by operators, i.e., CGN, the reuse of an IP address on
    the NAT's external side by many customers reduces the protection
    provided. Also in some applications (e.g., centralized
    conferencing), dual-hosted RTSP/RTP clients have valid use cases.
    The key is how to authenticate the messages exchanged during the NAT
    traversal process.

4. NAT-Traversal Techniques

    There exists a number of potential NAT traversal techniques that can
    be used to allow RTSP to traverse NATs. They have different features
    and are applicable to different topologies; their costs are also
    different. They also vary in security levels. In the following
    sections, each technique is outlined with discussions on the
    corresponding advantages and disadvantages.

    The survey of traversal techniques was done prior to 2007 and is
    based on what was available then. This section includes NAT
    traversal techniques that have not been formally specified anywhere
    else. This document may be the only publicly available specification
    of some of the NAT traversal techniques. However, that is not a real
    barrier against doing an evaluation of the NAT traversal techniques.
    Some techniques used as part of some of the traversal solutions have
    been recommended against or are no longer possible due to the outcome

of standardization work or their failure to progress within IETF
after the initial evaluation in this document. For example, RTP
No-Op [RTP-NO-OP] was a proposed RTP payload format that failed to be
specified; thus, it is not available for use today. In each such
case, the missing parts will be noted and some basic reasons will be
given.

4.1. Stand-Alone STUN

4.1.1. Introduction

Session Traversal Utilities for NAT (STUN) [RFC5389] is a
standardized protocol that allows a client to use secure means to
discover the presence of a NAT between itself and the STUN server.
The client uses the STUN server to discover the address and port
mappings assigned by the NAT. Then using the knowledge of these NAT
mappings, it uses the external addresses to directly connect to the
independent RTSP server. However, this is only possible if the NAT
address and port mapping behavior is such that the STUN server and
RTSP server will see the same external address and port for the same
internal address and port.

STUN is a client-server protocol. The STUN client sends a request to
a STUN server and the server returns a response. There are two types
of STUN messages -- Binding Requests and Indications. Binding
Requests are used when determining a client's external address and
soliciting a response from the STUN server with the seen address.
Indications are used by the client for keep-alive messages towards
the server and requires no response from the server.

The first version of STUN [RFC3489] included categorization and
parameterization of NATs. This was abandoned in the updated version
[RFC5389] due to it being unreliable and brittle. This particular
traversal method uses the removed functionality described in RFC 3489
to detect the NAT type to give an early failure indication when the
NAT is showing the behavior that this method can't support. This
method also suggests using the RTP No-Op payload format [RTP-NO-OP]
for keep-alives of the RTP traffic in the client-to-server direction.
This can be replaced with another form of UDP packet as will be
further discussed below.

4.1.2. Using STUN to Traverse NAT without Server Modifications

This section describes how a client can use STUN to traverse NATs to
RTSP servers without requiring server modifications. Note that this
method has limited applicability and requires the server to be
available in the external/public address realm in regards to the
client located behind a NAT(s).

Limitations:

o The server must be located in either a public address realm or the
   next-hop external address realm in regards to the client.

o The client may only be located behind NATs that perform Endpoint-
   Independent or Address-Dependent Mappings (the STUN server and
   RTSP server on the same IP address). Clients behind NATs that do
   Address and Port-Dependent Mappings cannot use this method. See
   [RFC4787] for the full definition of these terms.

o Based on the discontinued middlebox classification of the replaced
   STUN specification [RFC3489]; thus, it is brittle and unreliable.

Method:

An RTSP client using RTP transport over UDP can use STUN to traverse
a NAT(s) in the following way:

1. Use STUN to try to discover the type of NAT and the timeout
    period for any UDP mapping on the NAT. This is recommended to be
    performed in the background as soon as IP connectivity is
    established. If this is performed prior to establishing a
    streaming session, the delays in the session establishment will
    be reduced. If no NAT is detected, normal SETUP should be used.

2. The RTSP client determines the number of UDP ports needed by
    counting the number of needed media transport protocols sessions
    in the multimedia presentation. This information is available in
    the media description protocol, e.g., SDP [RFC4566]. For
    example, each RTP session will in general require two UDP ports:
    one for RTP, and one for RTCP.

3. For each UDP port required, establish a mapping and discover the
    public/external IP address and port number with the help of the
    STUN server. A successful mapping looks like: client's local
    address/port <-> public address/port.

4. Perform the RTSP SETUP for each media. In the Transport header,
    the following parameter should be included with the given values:
    "dest_addr" [RTSP] or "destination" + "client_port" [RFC2326]
    with the public/external IP address and port pair for both RTP
    and RTCP. To be certain that this works, servers must allow a
    client to set up the RTP stream on any port, not only even ports
    and with non-contiguous port numbers for RTP and RTCP. This
    requires the new feature provided in RTSP 2.0 [RTSP]. The server
    should respond with a Transport header containing an "src_addr"

        or "source" + "server_port" parameters with the RTP and RTCP
        source IP address and port of the media stream.

    5. To keep the mappings alive, the client should periodically send
        UDP traffic over all mappings needed for the session. For the
        mapping carrying RTCP traffic, the periodic RTCP traffic is
        likely enough. For mappings carrying RTP traffic and for
        mappings carrying RTCP packets at too low of a frequency, keep-
        alive messages should be sent.

    If a UDP mapping is lost, the above discovery process must be
    repeated. The media stream also needs to be SETUP again to change
    the transport parameters to the new ones. This will cause a glitch
    in media playback.

    To allow UDP packets to arrive from the server to a client behind an
    Address-Dependent or Address and Port-Dependent Filtering NAT, the
    client must first send a UDP packet to establish the filtering state
    in the NAT. The client, before sending an RTSP PLAY request, must
    send a so-called hole-punching packet on each mapping to the IP
    address and port given as the server's source address and port. For
    a NAT that only is Address-Dependent Filtering, the hole-punching
    packet could be sent to the server's discard port (port number 9).
    For Address and Port-Dependent Filtering NATs, the hole-punching
    packet must go to the port used for sending UDP packets to the
    client. To be able to do that, the server needs to include the
    "src_addr" in the Transport header (which is the "source" transport
    parameter in RFC2326). Since UDP packets are inherently unreliable,
    to ensure that at least one UDP message passes the NAT, hole-punching
    packets should be retransmitted a reasonable number of times.

    One could have used RTP No-Op packets [RTP-NO-OP] as hole-punching
    and keep-alive messages had they been defined. That would have
    ensured that the traffic would look like RTP and thus would likely
    have the least risk of being dropped by any firewall. The drawback
    of using RTP No-Op is that the payload type number must be
    dynamically assigned through RTSP first. Other options are STUN, an
    RTP packet without any payload, or a UDP packet without any payload.
    For RTCP it is most suitable to use correctly generated RTCP packets.
    In general, sending unsolicited traffic to the RTSP server may
    trigger security functions resulting in the blocking of the keep-
    alive messages or termination of the RTSP session itself.

    This method is further brittle as it doesn't support Address and
    Port-Dependent Mappings. Thus, it proposes to use the old STUN
    methods to classify the NAT behavior, thus enabling early error
    indication. This is strictly not required but will lead to failures
    during setup when the NAT has the wrong behavior. This failure can

also occur if the NAT changes the properties of the existing mapping
and filtering state or between the classification message exchange
and the actual RTSP session setup, for example, due to load.

4.1.3. ALG Considerations

If a NAT supports RTSP ALG (Application Level Gateway) and is not
aware of the STUN traversal option, service failure may happen,
because a client discovers its NAT external IP address and port
numbers and inserts them in its SETUP requests. When the RTSP ALG
processes the SETUP request, it may change the destination and port
number, resulting in unpredictable behavior. An ALG should not
update address fields that contain addresses other than the NAT's
internal address domain. In cases where the ALG modifies fields
unnecessarily, two alternatives exist:

1. Use Transport Layer Security (TLS) to encrypt the data over the
    RTSP TCP connection to prevent the ALG from reading and modifying
    the RTSP messages.

2. Turn off the STUN-based NAT traversal mechanism.

As it may be difficult to determine why the failure occurs, the usage
of TLS-protected RTSP message exchange at all times would avoid this
issue.

4.1.4. Deployment Considerations

For the stand-alone usage of STUN, the following applies:

Advantages:

o STUN is a solution first used by applications based on SIP
   [RFC3261] (see Sections 1 and 2 of [RFC5389]). As shown above,
   with little or no changes, the RTSP application can reuse STUN as
   a NAT traversal solution, avoiding the pitfall of solving a
   problem twice.

o Using STUN does not require RTSP server modifications, assuming it
   is a server that is compliant with RTSP 2.0; it only affects the
   client implementation.

Disadvantages:

o Requires a STUN server deployed in the same address domain as the
   server.

o   Only works with NATs that perform Endpoint-Independent and
    Address-Dependent Mappings. Address and Port-Dependent Filtering
    NATs create some issues.

o   Brittle to NATs changing the properties of the NAT mapping and
    filtering.

o   Does not work with Address and Port-Dependent Mapping NATs without
    server modifications.

o   Will not work if a NAT uses multiple IP addresses, since RTSP
    servers generally require all media streams to use the same IP as
    used in the RTSP connection to prevent becoming a DDoS tool.

o   Interaction problems exist when an RTSP-aware ALG interferes with
    the use of STUN for NAT traversal unless TLS-secured RTSP message
    exchange is used.

o   Using STUN requires that RTSP servers and clients support the
    updated RTSP specification [RTSP], because it is no longer
    possible to guarantee that RTP and RTCP ports are adjacent to each
    other, as required by the "client_port" and "server_port"
    parameters in RFC 2326.

   Transition:

   The usage of STUN can be phased out gradually as the first step of a
   STUN-capable server or client should be to check the presence of
   NATs. The removal of STUN capability in the client implementations
   will have to wait until there is absolutely no need to use STUN.

4.1.5. Security Considerations

   To prevent the RTSP server from being used as Denial-of-Service (DoS)
   attack tools, the RTSP Transport header parameters "destination" and
   "dest_addr" are generally not allowed to point to any IP address
   other than the one the RTSP message originates from. The RTSP server
   is only prepared to make an exception to this rule when the client is
   trusted (e.g., through the use of a secure authentication process or
   through some secure method of challenging the destination to verify
   its willingness to accept the RTP traffic). Such a restriction means
   that STUN in general does not work for use cases where RTSP and media
   transport go to different addresses.

   STUN combined with RTSP that is restricted by destination address has
   the same security properties as the core RTSP. It is protected from
   being used as a DoS attack tool unless the attacker has the ability
   to spoof the TCP connection carrying RTSP messages.

Using STUN's support for message authentication and the secure
transport of RTSP messages, attackers cannot modify STUN responses or
RTSP messages (TLS) to change the media destination. This protects
against hijacking; however, as a client can be the initiator of an
attack, these mechanisms cannot securely prevent RTSP servers from
being used as DoS attack tools.

4.2. Server Embedded STUN

4.2.1. Introduction

This section describes an alternative to the stand-alone STUN usage
in the previous section that has quite significantly different
behavior.

4.2.2. Embedding STUN in RTSP

This section outlines the adaptation and embedding of STUN within
RTSP. This enables STUN to be used to traverse any type of NAT,
including Address and Port-Dependent Mapping NATs. This would
require RTSP-level protocol changes.

This NAT traversal solution has limitations:

1. It does not work if both the RTSP client and RTSP server are
    behind separate NATs.

2. The RTSP server may, for security reasons, refuse to send media
    streams to an IP that is different from the IP in the client RTSP
    requests.

Deviations from STUN as defined in RFC 5389:

1. The RTSP application must provision the client with an identity
    and shared secret to use in the STUN authentication;

2. We require the STUN server to be co-located on the RTSP server's
    media source ports.

If the STUN server is co-located with the RTSP server's media source
port, an RTSP client using RTP transport over UDP can use STUN to
traverse ALL types of NATs. In the case of Address and Port-
Dependent Mapping NATs, the party on the inside of the NAT must
initiate UDP traffic. The STUN Binding Request, being a UDP packet
itself, can serve as the traffic initiating packet. Subsequently,
both the STUN Binding Response packets and the RTP/RTCP packets can
traverse the NAT, regardless of whether the RTSP server or the RTSP
client is behind NAT (however, only one of them can be behind a NAT).

Likewise, if an RTSP server is behind a NAT, then an embedded STUN
server must be co-located on the RTSP client's RTCP port. Also, it
will become the client that needs to disclose his destination address
rather than the server, so the server can correctly determine its NAT
external source address for the media streams. In this case, we
assume that the client has some means of establishing a TCP
connection to the RTSP server behind NAT so as to exchange RTSP
messages with the RTSP server, potentially using a proxy or static
rules.

To minimize delay, we require that the RTSP server supporting this
option must inform the client about the RTP and RTCP ports from where
the server will send out RTP and RTCP packets, respectively. This
can be done by using the "server_port" parameter in RFC 2326 and the
"src_addr" parameter in [RTSP]. Both are in the RTSP Transport
header. But in general, this strategy will require that one first
does one SETUP request per media to learn the server ports, then
perform the STUN checks, followed by a subsequent SETUP to change the
client port and destination address to what was learned during the
STUN checks.

To be certain that RTCP works correctly, the RTSP endpoint (server or
client) will be required to send and receive RTCP packets from the
same port.

4.2.3. Discussion on Co-located STUN Server

In order to use STUN to traverse Address and Port-Dependent Filtering
or Mapping NATs, the STUN server needs to be co-located with the
streaming server media output ports. This creates a demultiplexing
problem: we must be able to differentiate a STUN packet from a media
packet. This will be done based on heuristics. The existing STUN
heuristics is the first byte in the packet and the Magic Cookie field
(added in RFC 5389), which works fine between STUN and RTP or RTCP
where the first byte happens to be different. Thanks to the Magic
Cookie field, it is unlikely that other protocols would be mistaken
for a STUN packet, but this is not assured. For more discussion of
this, please see Section 5.1.2 of [RFC5764].

4.2.4. ALG Considerations

The same ALG traversal considerations as for stand-alone STUN applies
(Section 4.1.3).

4.2.5. Deployment Considerations

   For the "Embedded STUN" method the following applies:

   Advantages:

   o STUN is a solution first used by SIP applications. As shown
      above, with little or no changes, the RTSP application can reuse
      STUN as a NAT traversal solution, avoiding the pitfall of solving
      a problem twice.

   o STUN has built-in message authentication features, which makes it
      more secure against hijacking attacks. See the next section for
      an in-depth security discussion.

   o This solution works as long as there is only one RTSP endpoint in
      the private address realm, regardless of the NAT's type. There
      may even be multiple NATs (see Figure 1 in [RFC5389]).

   o Compared to other UDP-based NAT traversal methods in this
      document, STUN requires little new protocol development (since
      STUN is already an IETF standard), and most likely less
      implementation effort, since open source STUN server and client
      implementations are available [STUN-IMPL] [PJNATH].

   Disadvantages:

   o Some extensions to the RTSP core protocol, likely signaled by RTSP
      feature tags, must be introduced.

   o Requires an embedded STUN server to be co-located on each of the
      RTSP server's media protocol's ports (e.g., RTP and RTCP ports),
      which means more processing is required to demultiplex STUN
      packets from media packets. For example, the demultiplexer must
      be able to differentiate an RTCP RR packet from a STUN packet and
      forward the former to the streaming server and the latter to the
      STUN server.

   o Does not support use cases that require the RTSP connection and
      the media reception to happen at different addresses, unless the
      server's security policy is relaxed.

   o Interaction problems exist when an RTSP ALG is not aware of STUN
      unless TLS is used to protect the RTSP messages.

   o Using STUN requires that RTSP servers and clients support the
      updated RTSP specification [RTSP], and they both agree to support
      the NAT traversal feature.

o Increases the setup delay with at least the amount of time it
   takes to perform STUN message exchanges. Most likely an extra
   SETUP sequence will be required.

Transition:

The usage of STUN can be phased out gradually as the first step of a
STUN-capable machine can be used to check the presence of NATs for
the presently used network connection. The removal of STUN
capability in the client implementations will have to wait until
there is absolutely no need to use STUN, i.e., no NATs or firewalls.

4.2.6. Security Considerations

See Stand-Alone STUN (Section 4.1.5).

4.3. ICE

4.3.1. Introduction

Interactive Connectivity Establishment (ICE) [RFC5245] is a
methodology for NAT traversal that has been developed for SIP using
SDP offer/answer. The basic idea is to try, in a staggered parallel
fashion, all possible connection addresses in which an endpoint may
be reached. This allows the endpoint to use the best available UDP
"connection" (meaning two UDP endpoints capable of reaching each
other). The methodology has very nice properties in that basically
all NAT topologies are possible to traverse.

Here is how ICE works at a high level. Endpoint A collects all
possible addresses that can be used, including local IP addresses,
STUN-derived addresses, Traversal Using Relay NAT (TURN) addresses,
etc. On each local port that any of these address and port pairs
lead to, a STUN server is installed. This STUN server only accepts
STUN requests using the correct authentication through the use of a
username and password.

Endpoint A then sends a request to establish connectivity with
endpoint B, which includes all possible "destinations" [RFC5245] to
get the media through to A. Note that each of A's local address/port
pairs (host candidates and server reflexive base) has a co-located
STUN server. B in turn provides A with all its possible destinations
for the different media streams. A and B then uses a STUN client to
try to reach all the address and port pairs specified by A from its
corresponding destination ports. The destinations for which the STUN
requests successfully complete are then indicated and one is
selected.

If B fails to get any STUN response from A, all hope is not lost.
Certain NAT topologies require multiple tries from both ends before
successful connectivity is accomplished; therefore, requests are
retransmitted multiple times. The STUN requests may also result in
more connectivity alternatives (destinations) being discovered and
conveyed in the STUN responses.

4.3.2. Using ICE in RTSP

The usage of ICE for RTSP requires that both client and server be
updated to include the ICE functionality. If both parties implement
the necessary functionality, the following steps could provide ICE
support for RTSP.

This assumes that it is possible to establish a TCP connection for
the RTSP messages between the client and the server. This is not
trivial in scenarios where the server is located behind a NAT, and
may require some TCP ports be opened, or proxies are deployed, etc.

The negotiation of ICE in RTSP of necessity will work different than
in SIP with SDP offer/answer. The protocol interactions are
different, and thus the possibilities for transfer of states are also
somewhat different. The goal is also to avoid introducing extra
delay in the setup process at least for when the server is not behind
a NAT in regards to the client, and the client is either having an
address in the same address domain or is behind the NAT(s), which can
address the address domain of the server. This process is only
intended to support PLAY mode, i.e., media traffic flows from server
to client.

1. ICE usage begins in the SDP. The SDP for the service indicates
    that ICE is supported at the server. No candidates can be given
    here as that would not work with on demand, DNS load balancing,
    etc., which have the SDP indicate a resource on a server park
    rather than a specific machine.

2. The client gathers addresses and puts together its candidates for
    each media stream indicated in the session description.

3. In each SETUP request, the client includes its candidates in an
    ICE-specific transport specification. For the server, this
    indicates the ICE support by the client. One candidate is the
    most prioritized candidate and here the prioritization for this
    address should be somewhat different compared to SIP. High-
    performance candidates are recommended rather than candidates
    with the highest likelihood of success, as it is more likely that
    a server is not behind a NAT compared to a SIP user agent.

   4. The server responds to the SETUP (200 OK) for each media stream
       with its candidates. A server not behind a NAT usually only
       provides a single ICE candidate. Also, here one candidate is the
       server primary address.

   5. The connectivity checks are performed. For the server, the
       connectivity checks from the server to the clients have an
       additional usage. They verify that there is someone willing to
       receive the media, thus preventing the server from unknowingly
       performing a DoS attack.

   6. Connectivity checks from the client promoting a candidate pair
       were successful. Thus, no further SETUP requests are necessary
       and processing can proceed with step 7. If an address other than
       the primary has been verified by the client to work, that address
       may then be promoted for usage in a SETUP request (go to step 7).
       If the checks for the available candidates failed and if further
       candidates have been derived during the connectivity checks, then
       those can be signaled in new candidate lines in a SETUP request
       updating the list (go to step 5).

   7. Client issues the PLAY request. If the server also has completed
       its connectivity checks for the promoted candidate pair (based on
       the username as it may be derived addresses if the client was
       behind NAT), then it can directly answer 200 OK (go to step 8).
       If the connectivity check has not yet completed, it responds with
       a 1xx code to indicate that it is verifying the connectivity. If
       that fails within the set timeout, an error is reported back.
       The client needs to go back to step 6.

   8. Process completed and media can be delivered. ICE candidates not
       used may be released.

   To keep media paths alive, the client needs to periodically send data
   to the server. This will be realized with STUN. RTCP sent by the
   client should be able to keep RTCP open, but STUN will also be used
   for SIP based on the same motivations as for ICE.

4.3.3. Implementation Burden of ICE

   The usage of ICE will require that a number of new protocols and new
   RTSP/SDP features be implemented. This makes ICE the solution that
   has the largest impact on client and server implementations among all
   the NAT/firewall traversal methods in this document.

   RTSP server implementation requirements are:

   o STUN server features

   o Limited STUN client features

   o SDP generation with more parameters

   o RTSP error code for ICE extension

   RTSP client implementation requirements are:

   o Limited STUN server features

   o Limited STUN client features

   o RTSP error code and ICE extension

4.3.4. ALG Considerations

   If there is an RTSP ALG that doesn't support the NAT traversal
   method, it may interfere with the NAT traversal. As the usage of ICE
   for the traversal manifests itself in the RTSP message primarily as a
   new transport specification, an ALG that passes through unknown will
   not prevent the traversal. An ALG that discards unknown
   specifications will, however, prevent the NAT traversal. These
   issues can be avoided by preventing the ALG to interfere with the
   signaling by using TLS for the RTSP message transport.

   An ALG that supports this traversal method can, on the most basic
   level, just pass the transport specifications through. ALGs in NATs
   and firewalls could use the ICE candidates to establish a filtering
   state that would allow incoming STUN messages prior to any outgoing
   hole-punching packets, and in that way it could speed up the
   connectivity checks and reduce the risk of failures.

4.3.5. Deployment Considerations

   Advantages:

   o Solves NAT connectivity discovery for basically all cases as long
      as a TCP connection between the client and server can be
      established. This includes servers behind NATs. (Note that a
      proxy between address domains may be required to get TCP through.)

   o Improves defenses against DDoS attacks, since a media-receiving
      client requires authentications via STUN on its media reception
      ports.

Disadvantages:

o Increases the setup delay with at least the amount of time it
   takes for the server to perform its STUN requests.

o Assumes that it is possible to demultiplex between the packets of
   the media protocol and STUN packets. This is possible for RTP as
   discussed, for example, in Section 5.1.2 of [RFC5764].

o Has a fairly high implementation burden put on both the RTSP
   server and client. However, several open source ICE
   implementations do exist, such as [NICE] and [PJNATH].

4.3.6. Security Considerations

   One should review the Security Considerations section of ICE and STUN
   to understand that ICE contains some potential issues. However,
   these can be avoided by correctly using ICE in RTSP. An important
   factor is to secure the signaling, i.e., use TLS between the RTSP
   client and server. In fact ICE does help avoid the DDoS attack issue
   with RTSP substantially as it reduces the possibility for a DDoS
   using RTSP servers on attackers that are on path between the RTSP
   server and the target and capable of intercepting the STUN
   connectivity check packets and correctly sending a response to the
   server. The ICE connectivity checks with their random transaction
   IDs from the server to the client serves as a return-routability
   check and prevents off-path attackers to succeed with address
   spoofing. This is similar to Mobile IPv6's return routability
   procedure (Section 5.2.5 of [RFC6275]).

4.4. Latching

4.4.1. Introduction

   Latching is a NAT traversal solution that is based on requiring RTSP
   clients to send UDP packets to the server's media output ports.
   Conventionally, RTSP servers send RTP packets in one direction: from
   server to client. Latching is similar to connection-oriented
   traffic, where one side (e.g., the RTSP client) first "connects" by
   sending an RTP packet to the other side's RTP port; the recipient
   then replies to the originating IP and Port. This method is also
   referred to as "late binding". It requires that all RTP/RTCP
   transport be done symmetrically. This in effect requires Symmetric
   RTP [RFC4961]. Refer to [RFC7362] for a description of the Latching
   of SIP-negotiated media streams in Session Border Controllers.

   Specifically, when the RTSP server receives the Latching packet
   (a.k.a. hole-punching packet, since it is used to punch a hole in the

firewall/NAT) from its client, it copies the source IP and Port
number and uses them as the delivery address for media packets. By
having the server send media traffic back the same way as the
client's packets are sent to the server, address and port mappings
will be honored. Therefore, this technique works for all types of
NATs, given that the server is not behind a NAT. However, it does
require server modifications. The format of the Latching packet will
have to be defined.

Latching is very vulnerable to both hijacking and becoming a tool in
DDoS attacks (see Security Considerations in [RFC7362]) because
attackers can simply forge the source IP and Port of the Latching
packet. The rule for restricting IP addresses to one of the
signaling connections will need to be applied here also. However,
that does not protect against hijacking from another client behind
the same NAT. This can become a serious issue in deployments with
CGNs.

4.4.2. Necessary RTSP Extensions

To support Latching, RTSP signaling must be extended to allow the
RTSP client to indicate that it will use Latching. The client also
needs to be able to signal its RTP SSRC to the server in its SETUP
request. The RTP SSRC is used to establish some basic level of
security against hijacking attacks or simply to avoid mis-association
when multiple clients are behind the same NAT. Care must be taken in
choosing clients' RTP SSRC. First, it must be unique within all the
RTP sessions belonging to the same RTSP session. Second, if the RTSP
server is sending out media packets to multiple clients from the same
send port, the RTP SSRC needs to be unique among those clients' RTP
sessions. Recognizing that there is a potential that RTP SSRC
collisions may occur, the RTSP server must be able to signal to a
client that a collision has occurred and that it wants the client to
use a different RTP SSRC carried in the SETUP response or use unique
ports per RTSP session. Using unique ports limits an RTSP server in
the number of sessions it can simultaneously handle per interface IP
addresses.

The Latching packet as discussed above should have a field that can
contain a client and RTP session identifier to correctly associate
the Latching packet with the correct context. If an RTP packet is to
be used, there would be a benefit to using a well-defined RTP payload
format for this purpose as the No-Op payload format proposed
[RTP-NO-OP]. However, in the absence of such a specification, an RTP
packet without a payload could be used. Using SSRC is beneficial
because RTP and RTCP both would work as is. However, other packet
formats could be used that carry the necessary identification of the
context, and such a solution is discussed in Section 4.5.

4.4.3. ALG Considerations

   An RTSP ALG not supporting this method could interfere with the
   methods used to indicate that Latching is to be done, as well as the
   SSRC signaling, thus preventing the method from working. However, if
   the RTSP ALG instead opens the corresponding pinholes and creates the
   necessary mapping in the NAT, traversal will still work. Securing
   the RTSP message transport using TLS will avoid this issue.

   An RTSP ALG that supports this traversal method can for basic
   functionality simply pass the related signaling parameters
   transparently. Due to the security considerations for Latching,
   there might exist a benefit for an RTSP ALG that will enable NAT
   traversal to negotiate with the path and turn off the Latching
   procedures when the ALG handles this. However, this opens up to
   failure modes when there are multiple levels of NAT and only one
   supports an RTSP ALG.

4.4.4. Deployment Considerations

   Advantages:

   o Works for all types of client-facing NATs (requirement 1 in
      Section 3).

   o Has little interaction problems with any RTSP ALG changing the
      client's information in the Transport header.

   Disadvantages:

   o Requires modifications to both the RTSP server and client.

   o Limited to working with servers that are not behind a NAT.

   o The format of the packet for "connection setup" (a.k.a Latching
      packet) is not defined.

   o SSRC management if RTP is used for Latching due to risk for mis-
      association of clients to RTSP sessions at the server if SSRC
      collision occurs.

   o Has significant security considerations (See Section 4.4.5), due
      to the lack of a strong authentication mechanism and will need to
      use address restrictions.

4.4.5. Security Considerations

   Latching's major security issue is that RTP streams can be hijacked
   and directed towards any target that the attacker desires unless
   address restrictions are used. In the case of NATs with multiple
   clients on the inside of them, hijacking can still occur. This
   becomes a significant threat in the context of CGNs.

   The most serious security problem is the deliberate attack with the
   use of an RTSP client and Latching. The attacker uses RTSP to set up
   a media session. Then it uses Latching with a spoofed source address
   of the intended target of the attack. There is no defense against
   this attack other than restricting the possible address a Latching
   packet can come from to the same address as the RTSP TCP connection
   is from. This prevents Latching to be used in use cases that require
   different addresses for media destination and signaling. Even
   allowing only a limited address range containing the signaling
   address from where Latching is allowed opens up a significant
   vulnerability as it is difficult to determine the address usage for
   the network the client connects from.

   A hijack attack can also be performed in various ways. The basic
   attack is based on the ability to read the RTSP signaling packets in
   order to learn the address and port the server will send from and
   also the SSRC the client will use. Having this information, the
   attacker can send its own Latching packets containing the correct RTP
   SSRC to the correct address and port on the server. The RTSP server
   will then use the source IP and Port from the Latching packet as the
   destination for the media packets it sends.

   Another variation of this attack is for a man in the middle to modify
   the RTP Latching packet being sent by a client to the server by
   simply changing the source IP and Port to the target one desires to
   attack.

   One can fend off the snooping-based attack by applying encryption to
   the RTSP signaling transport. However, if the attacker is a man in
   the middle modifying Latching packets, the attack is impossible to
   defend against other than through address restrictions. As a NAT
   rewrites the source IP and (possibly) port, this cannot be
   authenticated, but authentication is required in order to protect
   against this type of DoS attack.

   Yet another issue is that these attacks also can be used to deny the
   client the service it desires from the RTSP server completely. The
   attacker modifies or originates its own Latching packets with a port

other than what the legit Latching packets use, which results in the
media server sending the RTP/RTCP traffic to ports the client isn't
listening for RTP/RTCP on.

The amount of random non-guessable material in the Latching packet
determines how well Latching can fend off stream hijacking performed
by parties that are off the client-to-server network path, i.e., it
lacks the capability to see the client's Latching packets. The
proposal above uses the 32-bit RTP SSRC field to this effect.
Therefore, it is important that this field is derived with a non-
predictable random number generator. It should not be possible by
knowing the algorithm used and a couple of basic facts to derive what
random number a certain client will use.

An attacker not knowing the SSRC but aware of which port numbers that
a server sends from can deploy a brute-force attack on the server by
testing a lot of different SSRCs until it finds a matching one.
Therefore, a server could implement functionality that blocks packets
to ports or from sources that receive or send multiple Latching
packets with different invalid SSRCs, especially when they are coming
from the same IP and Port. Note that this mitigation in itself opens
up a new venue for DoS attacks against legit users trying to latch.

To improve the security against attackers, the amount of random
material could be increased. To achieve a longer random tag while
still using RTP and RTCP, it will be necessary to develop RTP and
RTCP payload formats for carrying the random material.

4.5. A Variation to Latching

4.5.1. Introduction

   Latching as described above requires the usage of a valid RTP format
   as the Latching packet, i.e., the first packet that the client sends
   to the server to establish a bidirectional transport flow for RTP
   streams. There is currently no appropriate RTP packet format for
   this purpose, although the RTP No-Op format was a proposal to fix the
   problem [RTP-NO-OP]; however, that work was abandoned. [RFC6263]
   discusses the implication of different types of packets as keep-
   alives for RTP, and its findings are very relevant to the format of
   the Latching packet.

   Meanwhile, there have been NAT/firewall traversal techniques deployed
   in the wireless streaming market place that use non-RTP messages as
   Latching packets. This section describes a variant based on a subset
   of those solutions that alters the previously described Latching
   solution.

4.5.2. Necessary RTSP Extensions

   In this variation of Latching, the Latching packet is a small UDP
   packet that does not contain an RTP header. In response to the
   client's Latching packet, the RTSP server sends back a similar
   Latching packet as a confirmation so the client can stop the so-
   called "connection phase" of this NAT traversal technique.
   Afterwards, the client only has to periodically send Latching packets
   as keep-alive messages for the NAT mappings.

   The server listens on its RTP-media output port and tries to decode
   any received UDP packet as the Latching packet. This is valid since
   an RTSP server is not expecting RTP traffic from the RTSP client.
   Then, it can correlate the Latching packet with the RTSP client's
   session ID or the client's SSRC and record the NAT bindings
   accordingly. The server then sends a Latching packet as the response
   to the client.

   The Latching packet can contain the SSRC to identify the RTP stream,
   and care must be taken if the packet is bigger than 12 bytes,
   ensuring that it is distinctively different from RTP packets, whose
   header size is 12 bytes.

   RTSP signaling can be added to do the following:

   1. Enable or disable such Latching message exchanges. When the
       firewall/NAT has an RTSP-aware ALG, it is possible to disable
       Latching message exchange and let the ALG work out the address
       and port mappings.

   2. Configure the number of retries and the retry interval of the
       Latching message exchanges.

4.5.3. ALG Considerations

   See Latching ALG considerations in Section 4.4.3.

4.5.4. Deployment Considerations

   This approach has the following advantages when compared with the
   Latching approach (Section 4.4):

   1. There is no need to define an RTP payload format for firewall
       traversal; therefore, it is more simple to use, implement, and
       administer (requirement 4 in Section 3) than a Latching protocol,
       which must be defined.

   2. When properly defined, this kind of Latching packet exchange can
       also authenticate RTP receivers, to prevent hijacking attacks.

   This approach has the following disadvantage when compared with the
   Latching approach:

   1. The server's sender SSRC for the RTP stream or other session
       Identity information must be signaled in the RTSP's SETUP
       response, in the Transport header of the RTSP SETUP response.

4.5.5. Security Considerations

   Compared to the security properties of Latching, this variant is
   slightly improved. First of all it allows for a larger random field
   in the Latching packets, which makes it more unlikely for an off-path
   attacker to succeed in a hijack attack. Second, the confirmation
   allows the client to know when Latching works and when it doesn't and
   thus when to restart the Latching process by updating the SSRC.

   Still, the main security issue remaining is that the RTSP server
   can't know that the source address in the Latching packet was coming
   from an RTSP client wanting to receive media and not from one that
   likes to direct the media traffic to a DoS target.

4.6. Three-Way Latching

4.6.1. Introduction

   Three-Way Latching is an attempt to try to resolve the most
   significant security issues for both previously discussed variants of
   Latching. By adding a server request response exchange directly
   after the initial Latching, the server can verify that the target
   address present in the Latching packet is an active listener and
   confirm its desire to establish a media flow.

4.6.2. Necessary RTSP Extensions

   Uses the same RTSP extensions as the Alternative Latching method
   (Section 4.5) uses. The extensions for this variant are only in the
   format and transmission of the Latching packets.

   The client-to-server Latching packet is similar to the Alternative
   Latching (Section 4.5), i.e., a UDP packet with some session
   identifiers and a random value. When the server responds to the
   Latching packet with a Latching confirmation, it includes a random
   value (nonce) of its own in addition to echoing back the one the
   client sent. Then a third message is added to the exchange. The
   client acknowledges the reception of the Latching confirmation

message and echoes back the server's nonce, thus confirming that the
Latched address goes to an RTSP client that initiated the Latching
and is actually present at that address. The RTSP server will refuse
to send any media until the Latching Acknowledgement has been
received with a valid nonce.

4.6.3. ALG Considerations

   See Latching ALG considerations in Section 4.4.3.

4.6.4. Deployment Considerations

   A solution with a three-way handshake and its own Latching packets
   can be compared with the ICE-based solution (Section 4.3) and have
   the following differences:

   o Only works for servers that are not behind a NAT.

   o May be simpler to implement due to the avoidance of the ICE
      prioritization and check-board mechanisms.

   However, a Three-Way Latching protocol is very similar to using STUN
   in both directions as a Latching and verification protocol. Using
   STUN would remove the need for implementing a new protocol.

4.6.5. Security Considerations

   Three-Way Latching is significantly more secure than its simpler
   versions discussed above. The client-to-server nonce, which is
   included in signaling and also can be bigger than the 32 bits of
   random data that the SSRC field supports, makes it very difficult for
   an off-path attacker to perform a DoS attack by diverting the media.

   The client-to-server nonce and its echoing back does not protect
   against on-path attackers, including malicious clients. However, the
   server-to-client nonce and its echoing back prevents malicious
   clients to divert the media stream by spoofing the source address and
   port, as it can't echo back the nonce in these cases. This is
   similar to the Mobile IPv6 return routability procedure
   (Section 5.2.5 of [RFC6275]).

   Three-Way Latching is really only vulnerable to an on-path attacker
   that is quite capable. First, the attacker can learn the client-
   to-server nonce either by intercepting the signaling or by modifying
   the source information (target destination) of a client's Latching
   packet. Second, it is also on-path between the server and target
   destination and can generate a response using the server's nonce. An

adversary that has these capabilities is commonly capable of causing
significantly worse damage than this using other methods.

Three-Way Latching results in the server-to-client packet being
bigger than the client-to-server packet, due to the inclusion of the
server-to-client nonce in addition to the client-to-server nonce.
Thus, an amplification effect does exist; however, to achieve this
amplification effect, the attacker has to create a session state on
the RTSP server. The RTSP server can also limit the number of
responses it will generate before considering the Latching to be
failed.

4.7. Application Level Gateways

4.7.1. Introduction

An ALG reads the application level messages and performs necessary
changes to allow the protocol to work through the middlebox.
However, this behavior has some problems in regards to RTSP:

1. It does not work when RTSP is used with end-to-end security. As
    the ALG can't inspect and change the application level messages,
    the protocol will fail due to the middlebox.

2. ALGs need to be updated if extensions to the protocol are added.
    Due to deployment issues with changing ALGs, this may also break
    the end-to-end functionality of RTSP.

Due to the above reasons, it is not recommended to use an RTSP ALG in
NATs. This is especially important for NATs targeted to home users
and small office environments, since it is very hard to upgrade NATs
deployed in SOHO environments.

4.7.2. Outline on How ALGs for RTSP Work

In this section, we provide a step-by-step outline on how one could
go about writing an ALG to enable RTSP to traverse a NAT.

1. Detect any SETUP request.

2. Try to detect the usage of any of the NAT traversal methods that
    replace the address and port of the Transport header parameters
    "destination" or "dest_addr". If any of these methods are used,
    then the ALG should not change the address. Ways to detect that
    these methods are used are:

* For embedded STUN, it would be to watch for a feature tag,
  like "nat.stun", and to see if any of those exist in the
  "supported", "proxy-require", or "require" headers of the RTSP
  exchange.

* For stand-alone STUN and TURN-based solutions: This can be
  detected by inspecting the "destination" or "dest_addr"
  parameter. If it contains either one of the NAT's external IP
  addresses or a public IP address, then such a solution is in
  use. However, if multiple NATs are used, this detection may
  fail. Remapping should only be done for addresses belonging
  to the NAT's own private address space.

Otherwise, continue to the next step.

3. Create UDP mappings (client given IP and Port <-> external IP and
    Port) where needed for all possible transport specifications in
    the Transport header of the request found in (step 1). Enter the
    external address and port(s) of these mappings in the Transport
    header. Mappings shall be created with consecutive external port
    numbers starting on an even number for RTP for each media stream.
    Mappings should also be given a long timeout period, at least 5
    minutes.

4. When the SETUP response is received from the server, the ALG may
    remove the unused UDP mappings, i.e., the ones not present in the
    Transport header. The session ID should also be bound to the UDP
    mappings part of that session.

5. If the SETUP response settles on RTP over TCP or RTP over RTSP as
    lower transport, do nothing: let TCP tunneling take care of NAT
    traversal. Otherwise, go to the next step.

6. The ALG should keep the UDP mappings belonging to the RTSP
    session as long as: an RTSP message with the session's ID has
    been sent in the last timeout interval, or a UDP message has been
    sent on any of the UDP mappings during the last timeout interval.

7. The ALG may remove a mapping as soon as a TEARDOWN response has
    been received for that media stream.

4.7.3. Deployment Considerations

   Advantages:

   o No impact on either client or server.

   o Can work for any type of NATs.

Disadvantages:

o When deployed, they are hard to update to reflect protocol
   modifications and extensions. If not updated, they will break the
   functionality.

o When end-to-end security is used, the ALG functionality will fail.

o Can interfere with other types of traversal mechanisms, such as
   STUN.

Transition:

An RTSP ALG will not be phased out in any automatic way. It must be
removed, probably through the removal or update of the NAT it is
associated with.

4.7.4. Security Considerations

An ALG will not work with deployment of end-to-end RTSP signaling
security; however, it will work with the hop-by-hop security method
defined in Section 19.3 of RTSP 2.0 [RTSP]. Therefore, deployment of
ALG may result in clients located behind NATs not using end-to-end
security, or more likely the selection of a NAT traversal solution
that allows for security.

The creation of a UDP mapping based on the signaling message has some
potential security implications. First of all, if the RTSP client
releases its ports and another application is assigned these instead,
it could receive RTP media as long as the mappings exist and the RTSP
server has failed to be signaled or notice the lack of client
response.

A NAT with RTSP ALG that assigns mappings based on SETUP requests
could potentially become the victim of a resource exhaustion attack.
If an attacker creates a lot of RTSP sessions, even without starting
media transmission, this could exhaust the pool of available UDP
ports on the NAT. Thus, only a limited number of UDP mappings should
be allowed to be created by the RTSP ALG.

4.8. TCP Tunneling

4.8.1. Introduction

Using a TCP connection that is established from the client to the
server ensures that the server can send data to the client. The
connection opened from the private domain ensures that the server can
send data back to the client. To send data originally intended to be

transported over UDP requires the TCP connection to support some type
of framing of the media data packets. Using TCP also results in the
client having to accept that real-time performance can be impacted.
TCP's problem of ensuring timely delivery was one of the reasons why
RTP was developed. Problems that arise with TCP are: head-of-line
blocking, delay introduced by retransmissions, and a highly varying
rate due to the congestion control algorithm. If a sufficient amount
of buffering (several seconds) in the receiving client can be
tolerated, then TCP will clearly work.

4.8.2. Usage of TCP Tunneling in RTSP

   The RTSP core specification [RTSP] supports interleaving of media
   data on the TCP connection that carries RTSP signaling. See
   Section 14 in [RTSP] for how to perform this type of TCP tunneling.
   There also exists another way of transporting RTP over TCP, which is
   defined in Appendix C.2 in [RTSP]. For signaling and rules on how to
   establish the TCP connection in lieu of UDP, see Appendix C.2 in
   [RTSP]. This is based on the framing of RTP over the TCP connection
   as described in [RFC4571].

4.8.3. ALG Considerations

   An RTSP ALG will face a different issue with TCP tunneling, at least
   the interleaved version. Now the full data stream can end up flowing
   through the ALG implementation. Thus, it is important that the ALG
   is efficient in dealing with the interleaved media data frames to
   avoid consuming to many resources and thus creating performance
   issues.

   The RTSP ALG can also affect the transport specifications that
   indicate that TCP tunneling can be done and its prioritization,
   including removing the transport specification, thus preventing TCP
   tunneling.

4.8.4. Deployment Considerations

   Advantage:

   o Works through all types of NATs where the RTSP server is not NATed
      or is at least reachable like it was not.

   Disadvantages:

   o Functionality needs to be implemented on both server and client.

   o Will not always meet multimedia stream's real-time requirements.

Transition:

   The tunneling over RTSP's TCP connection is not planned to be phased
   out. It is intended to be a fallback mechanism and for usage when
   total media reliability is desired, even at the potential price of
   loss of real-time properties.

4.8.5. Security Considerations

   The TCP tunneling of RTP has no known significant security problems
   besides those already presented in the RTSP specification. It is
   difficult to get any amplification effect for DoS attacks due to
   TCP's flow control. The RTSP server's TCP socket, if independently
   used for media tunneling or only RTSP messages, can be used for a
   redirected syn attack. By spoofing the source address of any TCP
   init packets, the TCP SYNs from the server can be directed towards a
   target.

   A possible security consideration, when session media data is
   interleaved with RTSP, would be the performance bottleneck when RTSP
   encryption is applied, since all session media data also needs to be
   encrypted.

4.9. Traversal Using Relays around NAT (TURN)

4.9.1. Introduction

   TURN [RFC5766] is a protocol for setting up traffic relays that allow
   clients behind NATs and firewalls to receive incoming traffic for
   both UDP and TCP. These relays are controlled and have limited
   resources. They need to be allocated before usage. TURN allows a
   client to temporarily bind an address/port pair on the relay (TURN
   server) to its local source address/port pair, which is used to
   contact the TURN server. The TURN server will then forward packets
   between the two sides of the relay.

   To prevent DoS attacks on either recipient, the packets forwarded are
   restricted to the specific source address. On the client side, it is
   restricted to the source setting up the allocation. On the external
   side, it is limited to the source address/port pair that have been
   given permission by the TURN client creating the allocation. Packets
   from any other source on this address will be discarded.

   Using a TURN server makes it possible for an RTSP client to receive
   media streams from even an unmodified RTSP server. However, the
   problem is those RTSP servers most likely restrict media destinations
   to no other IP address than the one the RTSP message arrives from.
   This means that TURN could only be used if the server knows and

accepts that the IP belongs to a TURN server, and the TURN server
can't be targeted at an unknown address. Alternatively, both the
RTSP TCP connection as well as the RTP media is relayed through the
same TURN server.

4.9.2. Usage of TURN with RTSP

   To use a TURN server for NAT traversal, the following steps should be
   performed.

   1. The RTSP client connects with the RTSP server. The client
       retrieves the session description to determine the number of
       media streams. To avoid the issue of having the RTSP connection
       and media traffic from different addresses, the TCP connection
       must also be done through the same TURN server as the one in the
       next step. This will require the usage of TURN for TCP
       [RFC6062].

   2. The client establishes the necessary bindings on the TURN server.
       It must choose the local RTP and RTCP ports that it desires to
       receive media packets. TURN supports requesting bindings of even
       port numbers and contiguous ranges.

   3. The RTSP client uses the acquired address and port allocations in
       the RTSP SETUP request using the destination header.

   4. The RTSP server sends the SETUP reply, which must include the
       Transport header's "src_addr" parameter (source and port in RTSP
       1.0). Note that the server is required to have a mechanism to
       verify that it is allowed to send media traffic to the given
       address unless TCP relaying of the RTSP messages also is
       performed.

   5. The RTSP client uses the RTSP server's response to create TURN
       permissions for the server's media traffic.

   6. The client requests that the server starts playing. The server
       starts sending media packets to the given destination address and
       ports.

   7. Media packets arrive at the TURN server on the external port; if
       the packets match an established permission, the TURN server
       forwards the media packets to the RTSP client.

   8. If the client pauses and media is not sent for about 75% of the
       mapping timeout, the client should use TURN to refresh the
       bindings.

4.9.3. ALG Considerations

   As the RTSP client inserts the address information of the TURN
   relay's external allocations in the SETUP messages, the ALG that
   replaces the address, without considering that the address does not
   belong to the internal address realm of the NAT, will prevent this
   mechanism from working. This can be prevented by securing the RTSP
   signaling.

4.9.4. Deployment Considerations

   Advantages:

   o Does not require any server modifications given that the server
      includes the "src_addr" header in the SETUP response.

   o Works for any type of NAT as long as the RTSP server has a
      reachable IP address that is not behind a NAT.

   Disadvantages:

   o Requires another network element, namely the TURN server.

   o A TURN server for RTSP may not scale since the number of sessions
      it must forward is proportional to the number of client media
      sessions.

   o The TURN server becomes a single point of failure.

   o Since TURN forwards media packets, as a necessity it introduces
      delay.

   o An RTSP ALG may change the necessary destinations parameter. This
      will cause the media traffic to be sent to the wrong address.

   Transition:

   TURN is not intended to be phased out completely; see Section 19 of
   [RFC5766]. However, the usage of TURN could be reduced when the
   demand for having NAT traversal is reduced.

4.9.5. Security Considerations

   The TURN server can become part of a DoS attack towards any victim.
   To perform this attack, the attacker must be able to eavesdrop on the
   packets from the TURN server towards a target for the DoS attack.
   The attacker uses the TURN server to set up an RTSP session with
   media flows going through the TURN server. The attacker is in fact

creating TURN mappings towards a target by spoofing the source
address of TURN requests. As the attacker will need the address of
these mappings, he must be able to eavesdrop or intercept the TURN
responses going from the TURN server to the target. Having these
addresses, he can set up an RTSP session and start delivery of the
media. The attacker must be able to create these mappings. The
attacker in this case may be traced by the TURN username in the
mapping requests.

This attack requires that the attacker has access to a user account
on the TURN server to be able to set up the TURN mappings. To
prevent this attack, the RTSP server needs to verify that the
ultimate target destination accepts this media stream, which would
require something like ICE's connectivity checks being run between
the RTSP server and the RTSP client.

5. Firewalls

Firewalls exist for the purpose of protecting a network from traffic
not desired by the firewall owner. Therefore, it is a policy
decision if a firewall will let RTSP and its media streams through or
not. RTSP is designed to be firewall friendly in that it should be
easy to design firewall policies to permit passage of RTSP traffic
and its media streams.

The firewall will need to allow the media streams associated with an
RTSP session to pass through it. Therefore, the firewall will need
an ALG that reads RTSP SETUP and TEARDOWN messages. By reading the
SETUP message, the firewall can determine what type of transport and
from where the media stream packets will be sent. Commonly, there
will be the need to open UDP ports for RTP/RTCP. By looking at the
source and destination addresses and ports, the opening in the
firewall can be minimized to the least necessary. The opening in the
firewall can be closed after a TEARDOWN message for that session or
the session itself times out.

The above possibilities for firewalls to inspect and respond to the
signaling are prevented if end-to-end confidentiality protection is
used for the RTSP signaling, e.g., using the specified RTSP over TLS.
As a result, firewalls can't be actively opening pinholes for the
media streams based on the signaling. To enable an RTSP ALG in the
firewall to correctly function, the hop-by-hop signaling security in
RTSP 2.0 can be used (see Section 19.3 of [RTSP]). If not, other
methods have to be used to enable the transport flows for the media.

Simpler firewalls do allow a client to receive media as long as it
has sent packets to the target. Depending on the security level,
this can have the same behavior as a NAT. The only difference is

that no address translation is done. To use such a firewall, a
client would need to implement one of the above described NAT
traversal methods that include sending packets to the server to
create the necessary filtering state.

6. Comparison of NAT Traversal Techniques

   This section evaluates the techniques described above against the
   requirements listed in Section 3.

   In the following table, the columns correspond to the numbered
   requirements. For instance, the column under R1 corresponds to the
   first requirement in Section 3: must work for all flavors of NATs.
   The rows represent the different NAT/firewall traversal techniques.
   Latch is short for Latching, "V. Latch" is short for "variation of
   Latching" as described in Section 4.5, and "3-W Latch" is short for
   the Three-Way Latching described in Section 4.6.

   A summary of the requirements are:

   R1: Work for all flavors of NATs

   R2: Must work with firewalls, including those with ALGs

   R3: Should have minimal impact on clients not behind NATs, counted in
       minimal number of additional RTTs

   R4: Should be simple to use, implement, and administer

   R5: Should provide mitigation against DDoS attacks

   The following considerations are also added to the requirements:

   C1: Will the solution support both clients and servers behind NAT?

   C2: Is the solution robust as NAT behaviors change?

```
-----------+------+------+------+------+------+------+------+
           | R1   | R2   | R3   | R4   | R5   | C1   | C2   |
-----------+------+------+------+------+------+------+------+
 STUN      | No   | Yes  | 1    | Maybe| No   | No   | No   |
-----------+------+------+------+------+------+------+------+
 Emb. STUN | Yes  | Yes  | 2    | Maybe| No   | No   | Yes  |
-----------+------+------+------+------+------+------+------+
 ICE       | Yes  | Yes  | 2.5  | No   | Yes  | Yes  | Yes  |
-----------+------+------+------+------+------+------+------+
 Latch     | Yes  | Yes  | 1    | Maybe| No   | No   | Yes  |
-----------+------+------+------+------+------+------+------+
 V. Latch  | Yes  | Yes  | 1    | Yes  | No   | No   | Yes  |
-----------+------+------+------+------+------+------+------+
 3-W Latch | Yes  | Yes  | 1.5  | Maybe| Yes  | No   | Yes  |
-----------+------+------+------+------+------+------+------+
 ALG       |(Yes) | Yes  | 0    | No   | Yes  | No   | Yes  |
-----------+------+------+------+------+------+------+------+
 TCP Tunnel| Yes  | Yes  | 1.5  | Yes  | Yes  | No   | Yes  |
-----------+------+------+------+------+------+------+------+
 TURN      | Yes  | Yes  | 1    | No   | Yes  |(Yes) | Yes  |
-----------+------+------+------+------+------+------+------+
```

                 Figure 1: Comparison of Fulfillment of Requirements

Looking at Figure 1, one would draw the conclusion that using TCP
Tunneling or Three-Way Latching are the solutions that best fulfill
the requirements. The different techniques were discussed in the
MMUSIC WG. It was established that the WG would pursue an ICE-based
solution due to its generality and capability of also handling
servers delivering media from behind NATs. TCP Tunneling is likely
to be available as an alternative, due to its specification in the
main RTSP specification. Thus, it can be used if desired, and the
potential downsides of using TCP is acceptable in particular
deployments. When it comes to Three-Way Latching, it is a very
competitive technique given that you don't need support for RTSP
servers behind NATs. There was some discussion in the WG about if
the increased implementation burden of ICE is sufficiently motivated
compared to a the Three-Way Latching solution for this generality.
In the end, the authors believed that the reuse of ICE, greater
flexibility, and any way needed to deploy a new solution were the
decisive factors.

The ICE-based RTSP NAT traversal solution is specified in "A Network
Address Translator (NAT) Traversal mechanism for media controlled by
Real-Time Streaming Protocol (RTSP)" [RTSP-NAT].

7. Security Considerations

   In the preceding sections, we have discussed security merits of the
   different NAT/firewall traversal methods for RTSP. In summary, the
   presence of NAT(s) is a security risk, as a client cannot perform
   source authentication of its IP address. This prevents the
   deployment of any future RTSP extensions providing security against
   the hijacking of sessions by a man in the middle.

   Each of the proposed solutions has security implications. Using STUN
   will provide the same level of security as RTSP without transport-
   level security and source authentications, as long as the server does
   not allow media to be sent to a different IP address than the RTSP
   client request was sent from.

   Using Latching will have a higher risk of session hijacking or DoS
   than normal RTSP. The reason is that there exists a probability that
   an attacker is able to guess the random bits that the client uses to
   prove its identity when creating the address bindings. This can be
   solved in the variation of Latching (Section 4.5) with authentication
   features. Still, both those variants of Latching are vulnerable
   against a deliberate attack from the RTSP client to redirect the
   media stream requested to any target assuming it can spoof the source
   address. This security vulnerability is solved by performing a
   Three-way Latching procedure as discussed in Section 4.6.

   ICE resolves the binding vulnerability of Latching by using signed
   STUN messages, as well as requiring that both sides perform
   connectivity checks to verify that the target IP address in the
   candidate pair is both reachable and willing to respond. ICE can,
   however, create a significant amount of traffic if the number of
   candidate pairs are large. Thus, pacing is required and
   implementations should attempt to limit their number of candidates to
   reduce the number of packets.

   If the signaling between the ICE peers (RTSP client and server) is
   not confidentiality and integrity protected, ICE is vulnerable to
   attacks where the candidate list is manipulated. The lack of
   signaling security will also simplify spoofing of STUN binding
   messages by revealing the secret used in signing.

   The usage of an RTSP ALG does not in itself increase the risk for
   session hijacking. However, the deployment of ALGs as the sole
   mechanism for RTSP NAT traversal will prevent deployment of end-
   to-end encrypted RTSP signaling.

The usage of TCP tunneling has no known security problems. However,
it might provide a bottleneck when it comes to end-to-end RTSP
signaling security if TCP tunneling is used on an interleaved RTSP
signaling connection.

The usage of TURN has severe risk of DoS attacks against a client.
The TURN server can also be used as a redirect point in a DDoS attack
unless the server has strict enough rules for who may create
bindings.

Since Latching and the variants of Latching have such big security
issues, they should not be used at all. Three-Way Latching as well
as ICE mitigates these security issues and performs the important
return-routability checks that prevent spoofed source addresses, and
they should be recommended for that reason. RTP ALGs are a security
risk as they can create an incitement against using secure RTSP
signaling. That can be avoided as ALGs require trust in the
middlebox, and that trust becomes explicit if one uses the hop-by-hop
security solution as specified in Section 19.3 of RTSP 2.0. [RTSP].
The remaining methods can be considered safe enough, assuming that
the appropriate security mechanisms are used and not ignored.

8. Informative References

   [NICE]   Libnice, "The GLib ICE implementation", June 2015,
                <http://nice.freedesktop.org/wiki/>.

   [PJNATH]   "PJNATH - Open Source ICE, STUN, and TURN Library", May
                2013, <http://www.pjsip.org/pjnath/docs/html/>.

   [RFC768]   Postel, J., "User Datagram Protocol", STD 6, RFC 768,
                DOI 10.17487/RFC0768, August 1980,
                <http://www.rfc-editor.org/info/rfc768>.

   [RFC793]   Postel, J., "Transmission Control Protocol", STD 7,
                RFC 793, DOI 10.17487/RFC0793, September 1981,
                <http://www.rfc-editor.org/info/rfc793>.

   [RFC2326]   Schulzrinne, H., Rao, A., and R. Lanphier, "Real Time
                Streaming Protocol (RTSP)", RFC 2326,
                DOI 10.17487/RFC2326, April 1998,
                <http://www.rfc-editor.org/info/rfc2326>.

   [RFC2588]   Finlayson, R., "IP Multicast and Firewalls", RFC 2588,
                DOI 10.17487/RFC2588, May 1999,
                <http://www.rfc-editor.org/info/rfc2588>.

   [RFC2663]   Srisuresh, P. and M. Holdrege, "IP Network Address
               Translator (NAT) Terminology and Considerations",
               RFC 2663, DOI 10.17487/RFC2663, August 1999,
               <http://www.rfc-editor.org/info/rfc2663>.

   [RFC3022]   Srisuresh, P. and K. Egevang, "Traditional IP Network
               Address Translator (Traditional NAT)", RFC 3022,
               DOI 10.17487/RFC3022, January 2001,
               <http://www.rfc-editor.org/info/rfc3022>.

   [RFC3261]   Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston,
               A., Peterson, J., Sparks, R., Handley, M., and
               E. Schooler, "SIP: Session Initiation Protocol",
               RFC 3261, DOI 10.17487/RFC3261, June 2002,
               <http://www.rfc-editor.org/info/rfc3261>.

   [RFC3424]   Daigle, L., Ed. and IAB, "IAB Considerations for
               UNilateral Self-Address Fixing (UNSAF) Across Network
               Address Translation", RFC 3424, DOI 10.17487/RFC3424,
               November 2002, <http://www.rfc-editor.org/info/rfc3424>.

   [RFC3489]   Rosenberg, J., Weinberger, J., Huitema, C., and R. Mahy,
               "STUN - Simple Traversal of User Datagram Protocol (UDP)
               Through Network Address Translators (NATs)", RFC 3489,
               DOI 10.17487/RFC3489, March 2003,
               <http://www.rfc-editor.org/info/rfc3489>.

   [RFC3550]   Schulzrinne, H., Casner, S., Frederick, R., and V.
               Jacobson, "RTP: A Transport Protocol for Real-Time
               Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550,
               July 2003, <http://www.rfc-editor.org/info/rfc3550>.

   [RFC4566]   Handley, M., Jacobson, V., and C. Perkins, "SDP: Session
               Description Protocol", RFC 4566, DOI 10.17487/RFC4566,
               July 2006, <http://www.rfc-editor.org/info/rfc4566>.

   [RFC4571]   Lazzaro, J., "Framing Real-time Transport Protocol (RTP)
               and RTP Control Protocol (RTCP) Packets over Connection-
               Oriented Transport", RFC 4571, DOI 10.17487/RFC4571, July
               2006, <http://www.rfc-editor.org/info/rfc4571>.

   [RFC4787]   Audet, F., Ed. and C. Jennings, "Network Address
               Translation (NAT) Behavioral Requirements for Unicast
               UDP", BCP 127, RFC 4787, DOI 10.17487/RFC4787, January
               2007, <http://www.rfc-editor.org/info/rfc4787>.

   [RFC4961]   Wing, D., "Symmetric RTP / RTP Control Protocol (RTCP)",
               BCP 131, RFC 4961, DOI 10.17487/RFC4961, July 2007,
               <http://www.rfc-editor.org/info/rfc4961>.

   [RFC5245]   Rosenberg, J., "Interactive Connectivity Establishment
               (ICE): A Protocol for Network Address Translator (NAT)
               Traversal for Offer/Answer Protocols", RFC 5245,
               DOI 10.17487/RFC5245, April 2010,
               <http://www.rfc-editor.org/info/rfc5245>.

   [RFC5382]   Guha, S., Ed., Biswas, K., Ford, B., Sivakumar, S., and
               P. Srisuresh, "NAT Behavioral Requirements for TCP",
               BCP 142, RFC 5382, DOI 10.17487/RFC5382, October 2008,
               <http://www.rfc-editor.org/info/rfc5382>.

   [RFC5389]   Rosenberg, J., Mahy, R., Matthews, P., and D. Wing,
               "Session Traversal Utilities for NAT (STUN)", RFC 5389,
               DOI 10.17487/RFC5389, October 2008,
               <http://www.rfc-editor.org/info/rfc5389>.

   [RFC5764]   McGrew, D. and E. Rescorla, "Datagram Transport Layer
               Security (DTLS) Extension to Establish Keys for the
               Secure Real-time Transport Protocol (SRTP)", RFC 5764,
               DOI 10.17487/RFC5764, May 2010,
               <http://www.rfc-editor.org/info/rfc5764>.

   [RFC5766]   Mahy, R., Matthews, P., and J. Rosenberg, "Traversal
               Using Relays around NAT (TURN): Relay Extensions to
               Session Traversal Utilities for NAT (STUN)", RFC 5766,
               DOI 10.17487/RFC5766, April 2010,
               <http://www.rfc-editor.org/info/rfc5766>.

   [RFC6062]   Perreault, S., Ed. and J. Rosenberg, "Traversal Using
               Relays around NAT (TURN) Extensions for TCP Allocations",
               RFC 6062, DOI 10.17487/RFC6062, November 2010,
               <http://www.rfc-editor.org/info/rfc6062>.

   [RFC6263]   Marjou, X. and A. Sollaud, "Application Mechanism for
               Keeping Alive the NAT Mappings Associated with RTP / RTP
               Control Protocol (RTCP) Flows", RFC 6263,
               DOI 10.17487/RFC6263, June 2011,
               <http://www.rfc-editor.org/info/rfc6263>.

   [RFC6275]   Perkins, C., Ed., Johnson, D., and J. Arkko, "Mobility
               Support in IPv6", RFC 6275, DOI 10.17487/RFC6275, July
               2011, <http://www.rfc-editor.org/info/rfc6275>.

   [RFC7362]   Ivov, E., Kaplan, H., and D. Wing, "Latching: Hosted NAT
               Traversal (HNT) for Media in Real-Time Communication",
               RFC 7362, DOI 10.17487/RFC7362, September 2014,
               <http://www.rfc-editor.org/info/rfc7362>.

   [RTP-NO-OP] Andreasen, F., "A No-Op Payload Format for RTP", Work in
               Progress, draft-ietf-avt-rtp-no-op-04, May 2007.

   [RTSP]   Schulzrinne, H., Rao, A., Lanphier, R., Westerlund, M.,
               and M. Stiemerling, "Real Time Streaming Protocol 2.0
               (RTSP)", Work in Progress,
               draft-ietf-mmusic-rfc2326bis-40, February 2014.

   [RTSP-NAT] Goldberg, J., Westerlund, M., and T. Zeng, "A Network
               Address Translator (NAT) Traversal Mechanism for Media
               Controlled by Real-Time Streaming Protocol (RTSP)", Work
               in Progress, draft-ietf-mmusic-rtsp-nat-22, July 2014.

   [STUN-IMPL] "Open Source STUN Client and Server", May 2013,
               <http://sourceforge.net/projects/stun/>.

Acknowledgements

Authors' Addresses

    Magnus Westerlund
    Ericsson
    Farogatan 6
    Stockholm SE-164 80
    Sweden

    Phone: +46 8 719 0000
    Email: magnus.westerlund@ericsson.com

    Thomas Zeng
    PacketVideo Corp

    Email: thomas.zeng@gmail.com