Tunneling RTSP in HTTP

1. Introduction

There is a need for tunneling RTP streams inside RTSP in HTTP because in some cases users are located behind a firewall that is configured to only let HTTP through.

After discussing the context of the problem we will give the requirements for a solution and outline what a solution could be.

2. Motivation

RTSP [1, 10.12] clearly defines how RTP stream data can be embedded with RTSP methods. It also recommends that RTSP should be transported using TCP.

Note that tunneling RTSP (only) in HTTP would not be useful since the RTP data transported using UDP would be blocked.
configured to exclude UDP traffic can be traversed, which is already very useful.

However for many end users the situation is even worse; indeed some ISPs and many corporate Internet access are protected by strict firewalls. Typically these firewalls are configured to exclude all traffic except HTTP. Thus there is need to transport media through HTTP.

Although it is recognized that doing so is not optimal (see [1] and [2] for a discussion on why RTP/UDP is a better idea) it should be obvious that the core reason why tunneling streaming in HTTP in not a good idea is due to the use of TCP for transport, which is not an issue we need to discuss here since transporting RTP inside RTSP on TCP (as described in [1]) suffers from the same problem.

In fact the need for such a solution is so strong that most media streaming products implement (pseudo) streaming through HTTP in one way or another.

It is also obvious that although in many cases the reason why a corporate or ISP firewall is configured for HTTP only is pure paranoia, there are cases when suppressing media streaming is a genuine concern for an IT administrator. In this last case providing a standard technology that makes it possible to filter RTSP in HTTP would be a very good idea.

It is however of fundamental importance to stress that one of the contexts where such tunneling is needed is in environments where IT management is not leading edge. Indeed paranoia very often goes together with ignorance and/or lack of capability to trustfully communicate between decision makers and knowledgeable technical people. In such environments firewalls are set for maximum security i.e. HTTP-only and solutions that are known to work will be kept as long as possible. Therefore the solution we seek MUST work with all deployed firewalls otherwise it will be a very little value since we cannot expect this key target population to upgrade their firewalls if this is what is required to enable RTSP tunneling in HTTP.

On the other hand the situation is very different for more forward-looking organizations. We have two cases then. In places where IT administrators opened the required UDP ports in their firewalls so as to enable streaming, new solutions i.e. upgrading the firewall—will be easily adopted. There are also places where IT administrator did not open UDP ports for streaming upon explicit instructions from their management to stop media streaming. In such places surely the emergence of a standard technology enabling to deploy firewalls that can also block tunneling of media in HTTP will be well received. Therefore the solution we seek should also provide the ability to configure filtering mechanisms so as to give full control on what can and what cannot traverse the firewall.

3. Requirements

The requirements for tunneling RTSP in HTTP are therefore the
following:
Requirement 1: to traverse existing (deployed) HTTP-only firewalls
Requirement 2: to allow the development of new HTTP-level firewalls
where RTSP tunneling can be detected and eventually filtered.

4. Solutions

One solution is described in [3]. We will not discuss this solution in full detail but instead we will focus on what seems to be the most controversial issue.

5. Discussion

As described in [3] there is a need to prevent deployed HTTP proxy agents from trying to parse the RTSP syntax that lies after the HTTP header. Indeed HTTP-level firewalls are there to do exactly that: check that TCP connections carry HTTP data and nothing else. Unfortunately it seems that some deployed implementations will try to check the correctness of the HTTP syntax and in doing so stumble upon the RTSP syntax, causing the service to be denied. The solution proposed in [3] is therefore to hide RTSP syntax by trans-coding it in base64.

One obvious problem with this solution is that it may be seen as some kind of a cheat. We pretend as discussed in section 2 that this is not a true concern. Actually, as mentioned above, tunneling streaming media in HTTP is already performed on very large scales by a number of proprietary solutions and firewall administrators are actually lacking standard-based solutions to recover control upon such bandwidth-intensive traffic.

On the other hand, if a solution such as the one described in [3] was to become an IETF standard, proxy agents could detect this scenario by looking for an Accept or Content-Type header containing "application/x-rtsp-tunnelled". Classical filtering techniques could then be applied.

Alternatively other marking schemes could be designed to allow detection of RTSP tunneling into HTTP.

6. Security Considerations

Tunneling RTSP in HTTP does not have different security considerations than RTSP on TCP (covered by [1]) nor HTTP.

7. Acknowledgements

Gentric, Jones Expires January 2002 3
Tunneling RTSP in HTTP July 2001

This work has been started after a discussion in the Internet Streaming Media Alliance forum; authors wish to thank the people of this forum for raising interesting points.

8. References
9. Authors’ Addresses

Philippe Gentric
Philips
51 rue Carnot
92156 Suresnes
France
e-mail: philippe.gentric@philips.com

anne jones
Apple
1 Infinite Loop
Cupertino, CA 95014
e-mail: astoria@apple.com