SIPPING
Internet-Draft
Expires: August 16, 2004

J. Rosenberg
dynamicsoft

February 16, 2004

Requirements for Session Policy for the Session Initiation Protocol (SIP)
draft-ietf-sipping-session-policy-req-01

Status of this Memo

Copyright Notice

Abstract

The proxy server plays a central role as an intermediary in the establishment of sessions in the Session Initiation Protocol (SIP). In that role, they can define and impact policies on call routing, rendezvous, and other call features. However, there is no standard means by which proxies can have any influence on session policies, such as the codecs that are to be used. As such, ad-hoc and non-conformant techniques have been deployed to allow for such policy mechanisms. There is a need for a standards-based and complete mechanism for session policies. This document defines a set of requirements for such a mechanism.

Table of Contents

1. Introduction

The Session Initiation Protocol [2] enables the setup and management of interactive multimedia sessions on IP networks. A central element in SIP is the proxy server. Proxies are responsible for request routing, rendezvous, authentication and authorization, mobility, and other signaling services. However, proxies are divorced from the actual sessions - audio, video, and messaging - that SIP establishes. Details of the sessions are carried in the payload of SIP messages, and are usually described with the Session Description Protocol (SDP) [1]. Indeed, SIP provides end-to-end encryption features using S/MIME, so that all information about the sessions can be hidden from eavesdroppers and proxies alike.

However, experience has shown that there is a need for SIP intermediaries to impact aspects of the session. One aspect is the path that the media streams will take. Frequently, a SIP provider will need or want the media to traverse some kind of intermediary, such as a NAT. Indeed, the central concept of the midcom framework [4] is to define a model of how this can be done. In this model, a midcom agent, typically a proxy server, interacts with the middlebox to open and close media pinholes, obtain NAT bindings, and so on. In this role as a midcom agent, the proxy will need to examine and possibly modify the session description in the body of the SIP message. This modification is to achieve a specific policy objective: to force the media to route through an intermediary.

In another application, SIP is used in a wireless network. The network provider has limited resources for media traffic. During periods of high activity, the provider would like to restrict codec usage on the network to lower rate codecs.

In yet a third application, SIP is used in a network that has gateways which support a single codec type (say, G.729). When communicating with a partner network that uses gateways with a different codec (say, G.723), the network modifies the SDP to route the session through a converter that changes the G.729 to G.723.

The desire to impact aspects of the session inevitably occurs in domains where the administrator of the SIP domain is also the owner and administrator of an IP network over which it is known that the sessions will traverse. This includes enterprises, Internet access providers, and in some cases, backbone providers.

Since SIP is the protocol by which the details of these sessions are negotiated, it is natural for providers to wish to impose their session policies through some kind of SIP means. To date, this has been accomplished through SDP editing, a process where proxies dig

into the bodies of SIP messages, and modify them in order to impose their policies. However, this SIP editing technique has many drawbacks.

2. Problems with Existing Situation

RFC 3261 explicitly disallows proxy servers from manipulating the content of bodies. This is at odds with the common industry practice of extensive manipulation of bodies by proxies. Although a common practice, it is at odds with the SIP specification for many reasons:

End-to-End Encryption: SIP uses S/MIME to support end-to-end security security features. Authentication, message integrity, and encryption are provided. The encryption capabilities are important for end-to-end privacy services, for example. The end-to-end message integrity and authentication are important for preventing numerous attacks, including theft of calls, eavesdropping attacks, and so on. If end-to-end authentication is used, any manipulation of the body will cause the message integrity check to fail. If end-to-end encryption is used, the proxy won't even be able to look at the SDP to modify it. In this case, media may not function, and the call will fail.

Require Processing: A UA may require that an extension be applied to the SDP body. This is accomplished by including a Require header in the SIP message. Proxies do not look at such headers. If the proxy processes the SDP without understanding the extension, it may improperly modify the SDP, resulting in a call failure.

Consent: Ultimately, end users need to be in control of the media they send. If a user makes a call through a SIP network, they have the expectation that their media is delivered to the recipient. By having proxies modify the SDP in some way, they act in ways outside of expected behavior of the system.

Future Proofing: One of the benefits of the SIP architecture is that only the endpoints need to understand sessions, session descriptions, bodies, and so on. This facilitates the use of proxy networks to provide communications services for future session types, such as games and messaging. However, if proxies require an understanding of session types and session descriptions, the SIP network becomes locked in to providing features for a particular set of session types. If a new session description protocol, such as SDPng [10], were introduced, calls would not function even though the endpoints support SDPng. Furthermore, it would be hard to determine why it did not function, since the failure would occur transparently in some proxy in the middle of the network.

Robustness: Having a proxy manipulate the body introduces a host of new failure modes into the network. Firstly, the proxy itself will need to have state in some form in order to properly manipulate the SDP. This means that, should the proxy fail, the

call may not be able to continue. Secondly, proxies typically won't enforce the media policy. Rather, they leave that to some media middlebox somewhere on the media path. This media middlebox may fail as well. Since the user does not know of its existence, they may not be able to detect this failure or retry the media path around it.

Scalability: One of the reasons SIP scales so well is that proxies don't have to be aware of the details of the sessions being established through them. If a proxy needs to examine and/or manipulate session descriptions, this could require many additional processing steps. The proxy may need to traverse a multi-part body to find the SDP, in the case of SIP-T [5]. The proxy will need to parse, modify, and possibly re-serialize the session description. All of this requires additional processing that worsens the performance of the proxies.

We note that many of these problems are similar to those pointed out by the IAB regarding Open Pluggable Exchange Services (OPES) [6]. Indeed, the problems are similar. Both have to do with the involvement of intermediaries in manipulation of end-to-end content. Here, the content is not in the body itself, but is a session described by the body.

We believe a better solution is needed.

3. Requirements for a Solution

In order to prevent the continuing usage of SDP editing to achieve session policies, we believe explicit protocol support is needed to provide a mechanism that can overcome the limitations above. As per the IETF SIP change process [7], the first step in any such activity is to specify requirements for the solution. This section is an enumeration of those requirements.

3.1 General Requirements

REQ-GEN-1: The solution should work even with SIP end-to-end encryption and end-to-end authentication enabled.

REQ-GEN-2: The solution should not force a proxy to violate the SIP specification or any defined extensions.

REQ-GEN-3: The solution should not require substantial processing burden on the proxies.

REQ-GEN-4: The solution should not require proxies to understand a specific type of session description (i.e., SDP or SDPng).

REQ-GEN-5: The solution should have a minimal impact on call setup delays, and ideally, have no impact on call setup delays.

REQ-GEN-6: The solution should require minimal overhead, since it is anticipated to receive wide use in wireless networks.

REQ-GEN-7: The solution should be extensible, supporting new session policy types in the future.

REQ-GEN-8: The solution must not require that the proxies be in the same administrative domain as the media intermediaries.

3.2 Policy Requirements

REQ-POL-1: The solution should allow specification of independent policies by each proxy along the call setup path, without any coordination between proxies.

REQ-POL-2: The solution should allow a proxy to specify media policies on a stream-by-stream basis.

REQ-POL-3: When used in conjunction with the offer/answer model [3], the solution should allow a proxy to specify independent policies for the media streams in each direction.

REQ-POL-4: The mechanism must provide the ability to inform the UA about the set of session-independent session policies when the device starts up. These are session policies that do not depend on a particular session.

REQ-POL-5: The mechanism must allow the provider to change the session-independent policies at least a few times a day.

REQ-POL-6: The mechanism must allow the session independent policies to vary on a user by user basis.

REQ-POL-7 The mechanism must provide a way to inform the client about changes in session independent session policies when they occur.

3.3 Policy Types

REQ-POL-4: The solution should allow a proxy to request media sessions to traverse through one or more intermediaries.

REQ-POL-5: The solution should allow a proxy to request a specific source routing mechanism to be used (when applicable) in order to traverse those intermediaries. The source routing technique may be media-specific, or a generic technique, such as IP-in-IP [8]

REQ-POL-6: Intermediaries must be identifiable using either an IP address or an FQDN, in order to support DNS-based load balancing and failover techniques.

REQ-POL-7: The solution should allow a proxy to inspect the addresses for the media sessions, so that it can set policies in intervening firewalls.

REQ-POL-8: The solution should allow proxies to request that a particular media stream not be used (video, for example).

REQ-POL-9: The solution should allow proxies to request that a particular codec not be used.

REQ-POL-10: The solution should allow proxies to express preferences for the use of particular codecs.

REQ-POL-11: The solution should allow proxies to request that Quality of Service (QoS) should be requested for a stream.

REQ-POL-12: The solution should allow proxies to ask endpoints to use specific parameters in their QoS reservations.

REQ-POL-13: The solution should allow proxies to ask endpoints to provide a specific credential in their QoS requests. This requirement covers the functionality currently described in [9].

3.4 Consent Requirements

Consent plays a critical role for this problem. End users must be allowed control over how they communicate with each other. Indeed, with end-to-end IP connectivity, there is frequently little the provider can do to force users to communicate one way or another. Ultimately, any means a provider comes up with can be circumvented by some creative engineering in the clients. As such, policy requests by proxies are just that - requests, and are ultimately honored at the discretion of the end users. The mechanism needs to recognize this, and be engineered to work within this model, rather than try to work around it.

REQ-CON-1: The mechanism should allow the UAC to know the set of policies requested by the proxies along the call path. [[OPEN ISSUE: Is it more important for the UAC to know about changes requested for media in one direction or the other?]]

REQ-CON-2: The mechanism should allow the UAS to know the set of policies requested by the proxies along the call path.

REQ-CON-3: The mechanism should allow the UAC to reject any policy requests made by proxies.

REQ-CON-4: The mechanism should allow the UAS to reject any policy requests made by proxies.

REQ-CON-5: The mechanism should allow the proxies to know whether or not the UAC has accepted its policy requests.

REQ-CON-6: The mechanism should allow the proxies to know whether or not the UAS has accepted its policy requests.

REQ-CON-7: The mechanism should allow the proxies to inform the UAC and UAS of the consequences of non-compliance to the policies. Potential consequences include call rejection, degraded media quality, lack of connectivity for a media stream, and so on.

3.5 Security Requirements

REQ-SEC-1: The mechanism should allow user agents to verify the identity of the providers requesting the session policies.

REQ-SEC-2: The mechanism should allow user agents to verify the integrity of the session policies.

REQ-SEC-3: The mechanism must provide assurances to the UAC and UAS that only proxies on the actual SIP signaling path have requested session policies.

REQ-SEC-4: The mechanism should allow proxies to ensure the confidentiality of the session policies, so that no one but the UAC or UAS can observe them. [[OPEN ISSUE: Is this really a requirement?]]

REQ-SEC-5: The mechanism must not enable any new denial-of-service attacks to be launched. [[OPEN ISSUE: This is motherhood and apple pie - does it need to be here?]]

REQ-SEC-6: The mechanism shall still allow for media security through Secure RTP [11]. In the case of intermediaries which process the RTP in some way that would invalidate any signatures, the UAs must be aware of the presence of the intermediary, and perform key exchanges with it. [[OPEN ISSUE: This may be an impossible requirement to meet without using a B2BUA.]]

4. Security Considerations

Requirements related to security are considered in Section 3.5.

5. Acknowledgements

I would like to thank Volker Hilt, Gonzalo Camarillo, Miguel Garcia and Kumiko Ono for their input.

Informative References

[1] Handley, M. and V. Jacobson, "SDP: Session Description Protocol", RFC 2327, April 1998.

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

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

[4] Srisuresh, P., Kuthan, J., Rosenberg, J., Molitor, A. and A. Rayhan, "Middlebox communication architecture and framework", RFC 3303, August 2002.

[5] Vemuri, A. and J. Peterson, "Session Initiation Protocol for Telephones (SIP-T): Context and Architectures", BCP 63, RFC 3372, September 2002.

[6] Floyd, S. and L. Daigle, "IAB Architectural and Policy Considerations for Open Pluggable Edge Services", RFC 3238, January 2002.

[7] Mankin, A., Bradner, S., Mahy, R., Willis, D., Ott, J. and B. Rosen, "Change Process for the Session Initiation Protocol (SIP)", BCP 67, RFC 3427, December 2002.

[8] Perkins, C., "IP Encapsulation within IP", RFC 2003, October 1996.

[9] Marshall, W., "Private Session Initiation Protocol (SIP) Extensions for Media Authorization", RFC 3313, January 2003.

[10] Kutscher, D., Ott, J. and C. Bormann, "Session Description and Capability Negotiation", draft-ietf-mmusic-sdpng-07 (work in progress), October 2003.

[11] Baugher, M., "The Secure Real-time Transport Protocol", draft-ietf-avt-srtp-09 (work in progress), July 2003.

Author's Address

Jonathan Rosenberg
dynamicsoft
600 Lanidex Plaza
Parsippany, NJ 07054
US

Phone: +1 973 952-5000
EMail: jdrosen@dynamicsoft.com
URI: http://www.jdrosen.net

Intellectual Property Statement

Full Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.