SIP Core
Internet-Draft
Updates: 3261 (if approved)
Intended status: Standards Track
Expires: May 23, 2020

R. Shekh-Yusef
Avaya
C. Holmberg
Ericsson
V. Pascual
webrtchacks

November 20, 2019

Third-Party Token-based Authentication and Authorization for Session Initiation Protocol (SIP)
draft-ietf-sipcore-sip-token-authnz-06

Abstract

This document updates RFC 3261 and defines a mechanism for SIP, that is based on the OAuth 2.0 and OpenID Connect Core 1.0 specifications, to enable the delegation of the user authentication and SIP registration authorization to a dedicated third-party entity that is separate from the SIP network elements that provide the SIP service.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 23, 2020.

Table of Contents

1. Introduction

The SIP protocol [RFC3261] uses the framework used by the HTTP protocol [RFC7230] for authenticating users, which is a simple challenge- response authentication mechanism that allows a server to challenge a client request and allows a client to provide authentication information in response to that challenge.

OAuth 2.0 [RFC6749] defines a token based authorization framework to allow clients to access resources on behalf of their user.

The OpenID Connect 1.0 [OPENID] specifications defines a simple identity layer on top of the OAuth 2.0 protocol, which enables clients to verify the identity of the user based on the authentication performed by a dedicated authorization server, as well as to obtain basic profile information about the user.

This document updates [RFC3261], by defining the UAC procedures if it receives a 401/407 response with multiple WWW-Authenticate/Proxy-Authenticate header fields, providing challenges using different authentication schemes for the same realm.

This document defines an mechanism for SIP, that is based on the OAuth 2.0 and OpenID Connect Core 1.0 specifications, to enable the delegation of the user authentication and SIP registration authorization to a dedicated third-party entity that is separate from the SIP network elements that provide the SIP service.

1.1. Terminology

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

1.2. SIP User Agent Types

[RFC6749] defines two types of clients, confidential and public, that apply to the SIP User Agents.

- Confidential User Agent: is a SIP UA that is capable of maintaining the confidentiality of the user credentials and any tokens obtained using these user credentials.

- Public User Agent: is a SIP UA that is incapable of maintainings the confidentiality of the user credentials and any obtained tokens.

2. SIP Procedures

Section 22 of [RFC3261] defines the SIP procedures for the Digest authentication mechanism procedures. The same procedures apply to the Bearer authentication mechanism, with the changes described in this section.

2.1. UAC Behavior

2.1.1. Obtaining Tokens

When a UAC sends a request without credentials (or with credentials that are no longer valid), and receives a 401 (Unauthorized) or a 407 (Proxy Authentication Required) response that contains a WWW-Authenticate header field (in case of a 401 response) or a Proxy-Authenticate header field (in case of a 407 response) that indicates "Bearer" scheme authentication and contains an address to an Authorization Server, the UAC contacts the Authorization Server in order to obtain tokens, and includes the requested scopes based on a local configuration. The tokens returned to the UA depend on the type of server: with an OAuth AS, the tokens provided are the access token and refresh token. The access token will be sent to the SIP servers to authorize UAC's access to the service. The refresh token will only be used with the AS to get new access token and refresh token, before the expiry of the current access token. With an OpenID Connect server, an additional ID-Token is returned, which contains the SIP URI and other user specific details, and will be consumed by

the UAC.

The method used to authenticate the user and obtain these tokens is out of scope for this document, with one potential method is the Native App mechanism defined in [RFC8252]. The advantages of using the mechanism defined in [RFC8252] is that the user will be directed to use a browser to interact with the authorization server. This allows the authorization server to prompt the user for multi-factor authentication, redirect the user to third-party identity providers, and the use of single-sign-on sessions.

If the UAC receives a 401/407 response with multiple WWW-Authenticate/Proxy-Authenticate header fields, providing challenges using different authentication schemes for the same realm, the UAC provides credentials for one or more of the schemes that it supports, based on local policy.

NOTE: The address of the Authorization Server might be known to the UAC e.g., using means of configuration, in which case the UAC can

contact the Authorization Server in order to obtain the access token before it sends SIP request without credentials.

2.1.2. Access Token Claims

The type of services that an access token grants access to can be determined using different methods. Which methods are used is based on local policy. If an access token is encoded as a JWT, it might contain a list of claims [RFC7519], some registered and some are application specific claims. The REGISTRAR can grant access to services either based on such claims, using some other mechanism, or a combination of claims and some other mechanism. If an access token is a reference token, the REGISTRAR will grant access based on some other mechanism. Examples of such other mechanisms are introspection [RFC7662], user profile lookups, etc.

2.1.3. Protecting the Access Token

[RFC6749] mandates that Access Tokens are protected with TLS when in transit. However, TLS only guarantees hop-to-hop protection when

used to protect SIP signaling. Therefore the Access Token MUST be protected in a way so that only authorized SIP servers will have access to it. Endpoints that support this specifications MUST support encrypted JSON Web Tokens (JWT) [RFC7519] for encoding and protecting Access Token when included in SIP requests, unless some other mechanism is used to guarantee that only authorized SIP endpoints have access to the Access Token.

2.1.4. REGISTER Request

The procedures in this section assumes that the UAC has obtained a token as specified in section Section 2.1.1

When a UAC sends a REGISTER request in order to create a binding, it MUST include an Authorization headerf field with a Bearer scheme, carrying the access token, in the request, as specified in [RFC6750]. Based on local policy, the UAC MAY include an access token that has been used for another binding associated with the same AOR in the request.

When the UAC sends a binding refresh REGISTER request, it SHOULD include an Authorization header field with either the access token previously used for the binding, or a new access token (obtained using the refresh token) if the previous one has expired.

If the access token included in a REGISTER request is not accepted, and the UAC receives a 401 response or a 407 response, the UAC follows the procedures in Section 2.1.1.

2.1.5. Non-REGISTER Request

The procedures in this section assumes that the UAC has obtained a token as specified in section Section 2.1.1

When a UAC sends a request in order to initiate a SIP dialog, or sends a stand-alone request, the UAC MUST include an Authorization header field with a Bearer scheme, carrying the access token, in the request, as specified in [RFC6750]. Based on local policy, the UAC MAY include an access token that has been used for another dialog, or

for another stand-alone request, if the target of the new request is the same.

When the UAC sends a mid-dialog request, the UAC SHOULD include an Authorization header field with either the access token previously used within the dialog, or with a new access token if the previous one has expired or the UAC refreshed the access token before its expiry time.

If the access token included in a request is not accepted, and the UAC receives a 401 response or a 407 response, the UAC follows the procedures in Section 2.1.1.

2.2. UAS and Registrar Behavior

When a UAS or a Registrar receives a SIP request that does not contain an Authorization header field with a valid access token, and the UAS/Proxy decides to challenge the originator of the request, the proxy MUST challenge the request and send a 401 (Unauthorized) response. The UAS/Proxy MUST include a Proxy-Authentication header field in the response, indicate "Bearer" scheme and include an address to an Authorization Server from there the originator can obtain an access token.

When a UAS/Registrar receives a SIP request that contains an Authorization header field with an access token, the UAS/Registrar MUST validate the access token, using the procedures associated with the type of access token used. If the validation is successful the UAS/Registrar can continue to process the request using normal SIP procedures. If the validation fails, the UAS/Registrar MUST reject the request.

2.3. Proxy Behavior

When a proxy receives a SIP request that does not contain a Proxy-Authorization header field with a valid access token, and the proxy decides to challenge the originator of the request, the proxy MUST challenge the request and send a 407 (Proxy Authentication Required) response. The proxy MUST include a Proxy-Authentication header field in the response, indicate "Bearer" scheme and include an address to

an Authorization Server from there the originator can obtain an access token.

When a proxy receives a SIP request that contains an Proxy-Authorization header field with an access token, and the proxy has previously challenged the originator of the request, the proxy MUST validate the access token, using the procedures associated with the type of access token used. If the validation is successful the proxy can continue to process the request using normal SIP procedure. If the validation fails, the UAS/Registrar MUST reject the request.

3. WWW-Authenticate Response Header Field

This section describes the syntax of the WWW-Authenticate Response Header Field when used with the Bearer scheme to challenge the UA for credentials.

```
challenge  =/  ("Bearer" LWS bearer-cln *(COMMA bearer-cln))
bearer-cln = realm / scope / authz-server / error /
             auth-param
authz-server = "authz_server" EQUAL authz-server-value
authz-server-value = https-URI
```

The authz-server parameters contains the HTTPS URI, as defined in [RFC7230], of the authorization server.

The realm and auth-param parameters are defined in [RFC3261].

As per [RFC3261], the realm string alone defines the protection domain. [RFC3261] states that the realm string must be globally unique and recommends that the realm string contains a hostname or domain name. It also states that the realm string should be human-readable identifier that can be rendered to the user.

The scope and error parameters are defined in [RFC6749].

The scope parameter could be used by the registrar/proxy to indicate to the UAC the minimum scope that must be associated with the access

token to be able to get service. As defined in [RFC6749], the value

of the scope parameter is expressed as a list of space-delimited, case-sensitive strings. The strings are defined by the authorization server. The values of the scope parameter is out of scope for this document.

The error parameter could be used by the registrar/proxy to indicate to the UAC the reason for the error, with possible values of "invalid_token" or "invalid_scope".

4. 'sip.oauth2' Media Feature Tag

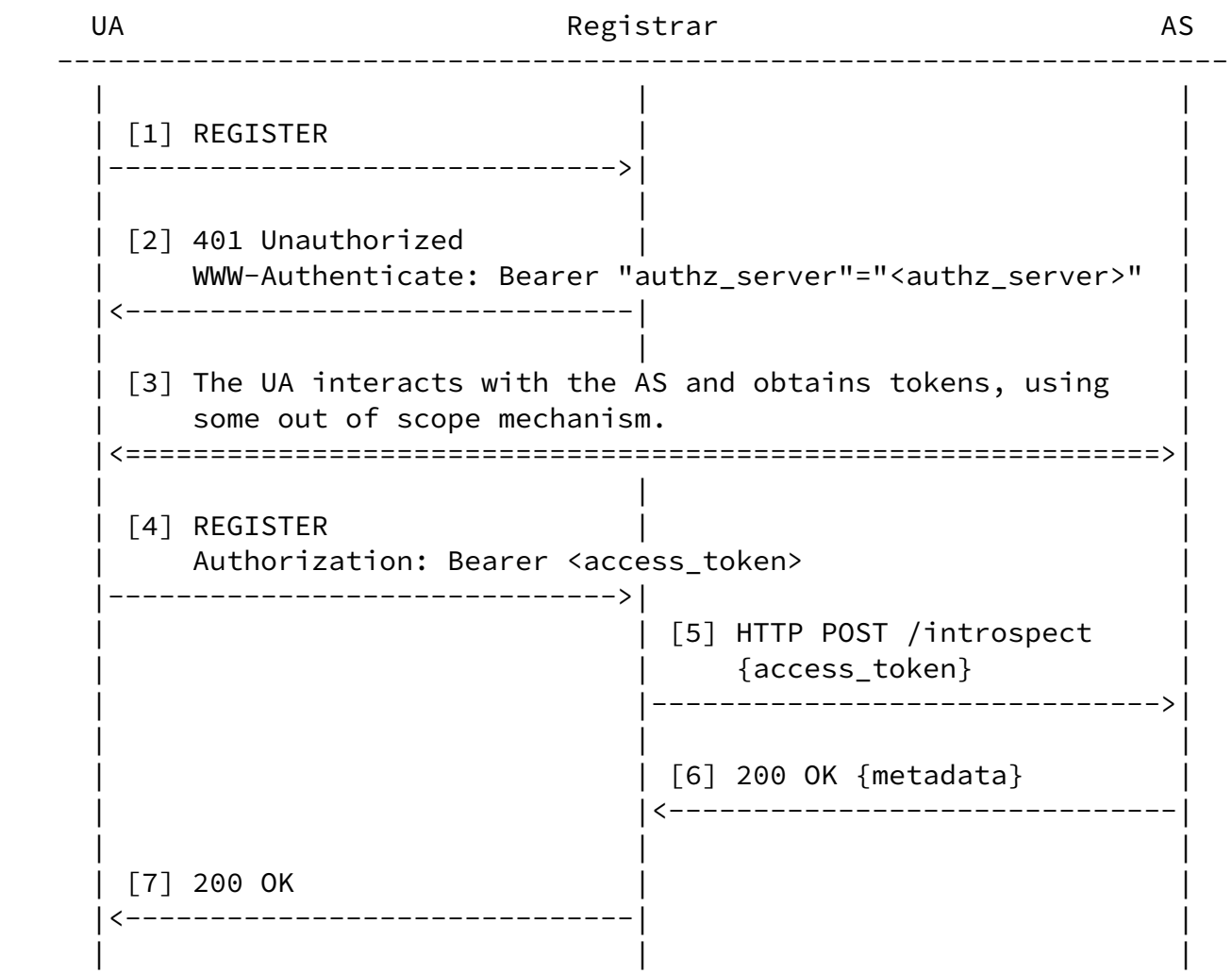
The sip.oauth2 media feature tag, when inserted in the Contact header field of a SIP REGISTER request, conveys that the SIP UA associated with the tag supports a token based authentication mechanism, where the user authentication and SIP registration authorization is performed by a third party. The media feature tag has no values.

token-mt = "+sip.oauth2"

5. Example Flows

5.1. Registration

The figure belows show an example of a SIP registration, where the UA is informed about the Authorization Server (AS) from where to obtain an access token by the registratar in a 401 response to the REGISTER request.



In step [1], the UA starts the registration process by sending a SIP REGISTER request to the registrar without any credentials. The REGISTER request includes an indication that the UA supports token-based autentication, using a sip.oauth2 media feature tag.

In step [2], the registrar challenges the UA, by sending a SIP 401 (Unauthorized) response to the REGISTER request. In the response the

registrar includes information about the AS to contact in order to obtain a token.

In step [3], the UA interacts with the AS, potentially using the OAuth Native App mechanism defined in [RFC8252], authenticates the user and obtains the tokens needed to access the SIP service.

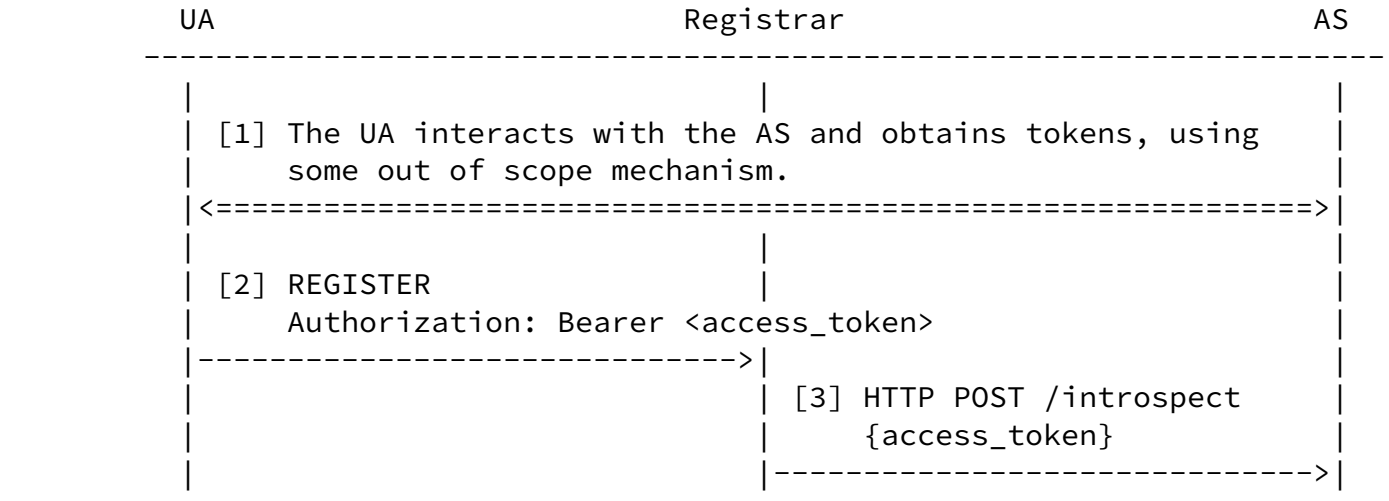
In step [4], the UA retries the registration process by sending a new SIP REGISTER request that includes the access token that the UA obtrained previously.

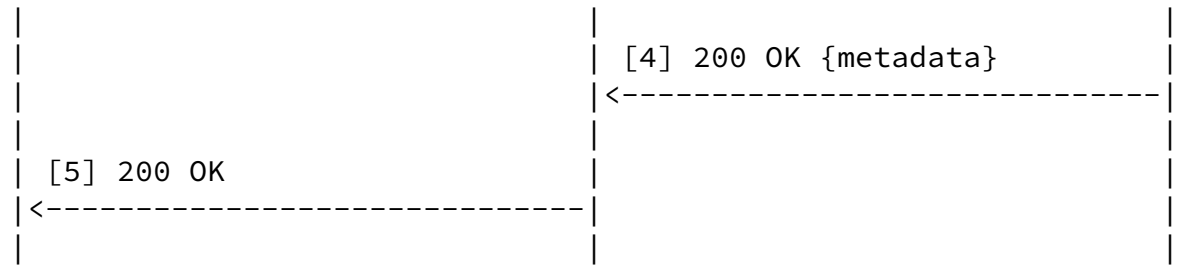
The registrar validates the access token. If the access token is a reference token, the registrar MAY perform an introspection, as in steps [5] and [6], in order to obtain more information about the access token and its scope, as per [RFC7662]. Otherwise, after the registrar validates the token to make sure it was signed by a trusted entity, it inspects its claims and act upon it.

In step [7], once the registrar has succesfully verified and accepted the access token, it sends a 200 (OK) response to the REGISTER request.

5.2. Registration with Pre-Configured AS

The figure belows show an example of a SIP registration, where the UA has pre-configured information about the Authorization Server (AS) from where to obtain the access token.





In step [1], the UA interacts with the AS, potentially using the OAuth Native App mechanism defined in [RFC8252], authenticates the user and obtains the tokens needed to access the SIP service.

In step [2], the UA retries the registration process by sending a new SIP REGISTER request that includes the access token that the UA obtrained previously.

The registrar validates the access token. If the access token is a reference token, the registrar MAY perform an introspection, as in steps [3] and [4], in order to obtain more information about the access token and its scope, as per [RFC7662]. Otherwise, after the registrar validates the token to make sure it was signed by a trusted entity, it inspects its claims and act upon it.

In step [5], once the registrar has succesfully verified and accepted the access token, it sends a 200 (OK) response to the REGISTER request.

6. Security Considerations

The security considerations for OAuth are defined in [RFC6749]. The security considerations for bearer tokens are defined in [RFC6750]. The security considerations for JSON Web Tokens (JWT) are defined in [RFC7519]. These security considerations also apply to SIP usage of access token as defined in this document.

[RFC6749] mandates that Access Tokens are protected with TLS. However, TLS only guarantees hop-to-hop protection when used to protect SIP signaling. Therefore the Access Token MUST be protected in a way so that only authorized SIP endpoints will have access to

it. Endpoints that support this specifications MUST support encrypted JSON Web Tokens (JWT) [RFC7519] for encoding and protecting Access Token when included in SIP requests, unless some other mechanism is used to guarantee that only authorized SIP endpoints have access to the Access Token.

7. IANA Considerations

7.1. SIP Media Feaure Tag

7.1.1. sip.oauth2

This section defines a new media feature tag that extends the "SIP Media Feature Tag Registration Tree" subregistry [RFC3840] under the "Media Feature Tags" registry (https://www.iana.org/assignments/media-feature-tags).

Media feature tag name: sip.oauth2

Summary of the media feature indicated by this feature tag: This media feature tag, when inserted in the Contact header field of a SIP REGISTER request, conveys that the SIP UA associated with the tag supports a token based authentication mechanism, where the user authentication and SIP registration authorization is performed by a third party.

Values appropriate for use with this feature tag: none

Related standards or documents: RFC XXXX

Security considerations: This media feature tag does not introduce

new security considerations, as it simply indicates support for a basic SIP feature. However, if an attacker manages to remove the media feature tag from a SIP REGISTER request, the SIP UA that inserted it might not be able to authenticate itself with the SIP registrar to which the SIP request is addressed, as the SIP registrar might not be aware that the SIP UA supports the feature associated with the media feature tag.

Contact: IESG (iesg@ietf.org)

8. Acknowledgments

The authors would also like to thank the following for their review and feedback on this document:

Paul Kyzivat, Olle Johansson, Roman Shpount, and Dale Worley.

The authors would also like to thank the following for their review and feedback of the original document that was replaced with this document:

Andrew Allen, Martin Dolly, Keith Drage, Paul Kyzivat, Jon Peterson, Michael Procter, Roy Radhika, Matt Ryan, Ivo Sedlacek, Roman Shpount, Robert Sparks, Asveren Tolga, and Dale Worley.

9. Normative References

[OPENID] Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., and C. Mortimore, "OpenID Connect Core 1.0", February 2014.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[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, <https://www.rfc-editor.org/info/rfc3261>.

[RFC3840] Rosenberg, J., Schulzrinne, H., and P. Kyzivat, "Indicating User Agent Capabilities in the Session Initiation Protocol (SIP)", RFC 3840, DOI 10.17487/RFC3840, August 2004, <https://www.rfc-editor.org/info/rfc3840>.

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/info/rfc6749>.

[RFC6750] Jones, M. and D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage", RFC 6750, DOI 10.17487/RFC6750, October 2012, <https://www.rfc-editor.org/info/rfc6750>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/info/rfc7230>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/info/rfc7519>.

[RFC7662] Richer, J., Ed., "OAuth 2.0 Token Introspection", RFC 7662, DOI 10.17487/RFC7662, October 2015, <https://www.rfc-editor.org/info/rfc7662>.

[RFC8252] Denniss, W. and J. Bradley, "OAuth 2.0 for Native Apps", BCP 212, RFC 8252, DOI 10.17487/RFC8252, October 2017, <https://www.rfc-editor.org/info/rfc8252>.

Authors' Addresses

Rifaat Shekh-Yusef
Avaya
425 Legget Drive
Ottawa, Ontario
Canada

Phone: +1-613-595-9106
EMail: rifaat.ietf@gmail.com

Christer Holmberg
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

EMail: christer.holmberg@ericsson.com

Victor Pascual
webrtchacks
Spain

EMail: victor.pascual.avila@gmail.com