Asterisk and sipml5 interoperability

At the moment Asterisk has limited functionality to communicate with clients that use WebRTC, like sipml5. Asterisk understands the offered media profile but it still has some issues with setting up the ICE connections. So the signaling works (setting up a call) but setting up the media streams fails. Also Asterisk can't do videocalls with standard WebRTC clients because WebRTC uses VP8 as its video codec and Asterisk has no support for VP8.

Enter webrtc2sip. webrtc2sip is a media gateway developed by Doubango that can connect any SIP client, regardless of the underlying framework. webrtc2sip effectively takes care of all the communication between legacy SIP clients and WebRTC clients, it not only handles media profile and codec conversion but also all ICE connections.

Setting up webrtc2sip

From the webrtc2sip website:

> *webrtc2sip is a smart and powerful gateway using RTCWeb and SIP to turn your browser into a phone with audio, video and SMS capabilities. The gateway allows your web browser to make and receive calls from/to any SIP-legacy network or PSTN.*

The contents of this tutorial are loosely based on the webrtc2sip Google Code project Wiki entry Building_Source_v2_0.

Prerequisites

This tutorial assumes basic knowledge of administering a Linux system, VoIP/SIP and building software from source. You will need a basic Ubuntu 12.04 install with the necessary packages to build webrtc2sip. To install those packages the following command will do:

```
sudo apt-get install build-essential libtool automake subversion git-core
libsrtp0-dev \
libssl-dev speex libspeexdsp-dev yasm libvpx-dev libgsm1-dev libxml2-dev \
libx264-dev screen pkg-config
```

Before you can start building webrtc2sip itself you will first need to build ffmpeg and doubango from source. doubango is the framework on which relies webrtc2sip and for doubango to be able to do the necessary conversion you will need some libraries from the ffmpeg project.

ffmpeg

You will need to build from a recent ffmpeg git clone because with the ffmpeg packages available for Ubuntu the doubango framework on which webrtc2sip relies will fail to build.

Build and install ffmpeg with:

```
cd /usr/local/src
git clone --depth 1 git://source.ffmpeg.org/ffmpeg.git ffmpeg
cd ffmpeg
./configure --extra-cflags="-fPIC" --extra-ldflags="-lpthread" --enable-
pic \
--enable-memalign-hack --enable-shared --disable-static --disable-network \
--disable-protocols --disable-pthreads --disable-devices --disable-filters \
```

```
--disable-bsfs --disable-muxers --disable-demuxers --disable-parsers \
--disable-hwaccels --disable-ffmpeg --disable-ffplay --disable-ffserver \
--disable-encoders --disable-decoders --disable-zlib --enable-gpl --disable-
debug \
--enable-encoder=h263 --enable-encoder=h263p --enable-decoder=h263 \
--enable-encoder=mpeg4 --enable-decoder=mpeg4 --enable-libx264 \
--enable-encoder=libx264 --enable-decoder=h264
make -j `getconf _NPROCESSORS_ONLN`
make install
ldconfig
```

doubango

Now you can build doubango from source. Use svn to pull in a recent checkout.

```
cd /usr/local/src
svn co http://doubango.googlecode.com/svn/branches/2.0/doubango doubango
cd doubango
./autogen.sh
./configure --with-ssl --with-srtp --with-vpx --with-speex --with-speexdsp \
--enable-speexresampler --enable-speexjb --enable-speexdenoiser --with-gsm \
--with-ffmpeg --with-h264 --prefix=/usr/local
make -j `getconf _NPROCESSORS_ONLN`
make install
ldconfig
```

If you want to use Linphone clients with VP8 you will have to disable VP8 extensions otherwise Linphone won't show any video:

```
sed -i \
's/TDAV_VP8_DISABLE_EXTENSION       0/TDAV_VP8_DISABLE_EXTENSION       1/' \
/usr/local/src/doubango/tinyDAV/src/codecs/vpx/tdav_codec_vp8.c
```

webrtc2sip

With doubango successfully installed you can now pull in the source of webrtc2sip from svn and build it from source.

```
cd /usr/local/src
svn co http://webrtc2sip.googlecode.com/svn/trunk/ webrtc2sip
cd webrtc2sip
mp_mediaproxy.cc
./autogen.sh
./configure --with-doubango=/usr/local --prefix=/usr/local
make -j `getconf _NPROCESSORS_ONLN`
make install
mkdir -p /usr/local/etc/webrtc2sip
cp config.xml /usr/local/sbin/
```

If you want webrtc2sip to look for `/usr/local/etc/webrtc2sip/config.xml` on startup you can modify `mp_mediaproxy.cc` before building webrtc2sip:

```
sed -i '1,/NULL/s/NULL/"\/usr\/local\/etc\/webrtc2sip\/config.xml"/' \
mp_mediaproxy.cc
```

After that you can build webrtc2sip as described above. This step is optional as webrtc2sip now has the `--config=/path/to/config.xml` command line option that allows you to have webrtc2sip

look for the designated `config.xml` file on startup.

Configuration

Configuration of webrtc2sip is done in `/usr/local/etc/webrtc2sip/config.xml`. Edit it to your needs with the help of the webrtc2sip Technical Guide. A sample configuration could look like this:

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Please check the technical guide
(http://webrtc2sip.googlecode.com/svn/trunk/documentation/technical-guide-
1.0.pdf)
for more information on how to adjust this file -->
<config>

  <debug-level>INFO</debug-level>

  <transport>udp;*;10060</transport>
  <transport>ws;*;10060</transport>
  <transport>wss;*;10062</transport>

  <enable-rtp-symetric>yes</enable-rtp-symetric>
  <enable-100rel>no</enable-100rel>
  <enable-media-coder>yes</enable-media-coder>
  <enable-videojb>yes</enable-videojb>
  <video-size-pref>vga</video-size-pref>
  <rtp-buffsize>65535</rtp-buffsize>
  <avpf-tail-length>100;400</avpf-tail-length>
  <srtp-mode>optional</srtp-mode>
  <srtp-type>sdes;dtls</srtp-type>

  <codecs>pcma;pcmu;gsm;vp8;h264-bp;h264-mp;h263;h263+;mp4v-es</codecs>

  <!--nameserver>66.66.66.6</nameserver-->

  <!--ssl-certificates>
    C:/Projects/ssl/priv.pem;
    C:/Projects/ssl/pub.pem;
    C:/Projects/ssl/ca-cert.pem;
    no
  </ssl-certificates-->

</config>
```

Run it!

```
screen -dmS webrtc2sip webrtc2sip \
--config=/usr/local/etc/webrtc2sip/config.xml
screen -r webrtc2sip
```

Setting up Asterisk

Installation

```
sudo apt-get install libncurses5-dev libnewt-dev libsqlite3-dev
wget http://downloads.asterisk.org/pub/telephony/asterisk/asterisk-11\
-current.tar.gz
```

```
tar zxvf asterisk-11-current.tar.gz
cd asterisk-11.1.2
./configure
make menuselect # optional step
make -j `getconf _NPROCESSORS_ONLN`
make install
make config
make samples
```

VP8 Passthrough Support

If you want to connect SIP clients to Asterisk using VP8 then you can patch Asterisk to add VP8 passthrough support. Note that this patch is experimental so use at your own risk.

Before running `./configure` download the patch an apply it:

```
wget https://raw.github.com/AutoStatic/asterisk-vp8/master\
/asterisk_11_vp8_passthrough_support.patch
patch -p1 -i asterisk_11_vp8_passthrough_support.patch
```

After that you can continue with compiling and installing.

Configuration

Create md5 hashes for your users. Even though md5 is far from being secure it is always better to have hashed passwords then plain-text passwords in your `sip.conf`.

```
echo -n WebRTCClient:mydomain.tld:mystrongpassword | md5sum \
| cut -d " " -f 1
echo -n LegacySIPClient:mydomain.tld:mystrongpassword | md5sum \
| cut -d " " -f 1
```

`/etc/asterisk/sip.conf`

```
[general]
context = public
allowoverlap = no
realm = mydomain.tld
udpbindaddr = 111.222.333.444
allow = !all,alaw,ulaw,gsm
useragent = Name of your PBX
nat = auto_force_rport,auto_comedia
videosupport = yes

[webrtc-template](!)
type = friend
context = webrtc
host = dynamic
allow = h264,h263p,mpeg4

[WebRTCClient](webrtc-template)
callerid = “WebRTCClient” <100>
md5secret = md5hashofuser
directmedia = outgoing

[LegacySIPClient](webrtc-template)
callerid = “LegacySIPClient” <200>
md5secret = md5hashofuser
```

```
directmedia = no
```

/etc/asterisk/extensions.conf

```
[public]
exten => s,1,NoOp()
 same => n,Hangup()

[webrtc]
exten => 100,1,Dial(SIP/WebRTCClient)
 same => n,Hangup()

exten => 200,1,Dial(SIP/LegacySIPClient)
 same => n,Hangup()
```

If you have compiled Asterisk with VP8 passthrough support you can simply add `vp8` to the list of allowed videocodecs in your webrtc template.

Setting up sipml5

Prerequisites

Download Google Chrome if you haven't done so yet. If Google Chrome is already installed make sure you're using the latest stable version (as of writing version 23.0.1271.97). Start Chrome and browse to http://sipml5.org/call.htm

sipml5: Registration

Display Name: *WebRTCClient*
Private Identity: *WebRTCClient*
Public Identity: *sip:WebRTCClient@mydomain.tld*
Password: *mystrongpassword*
Realm: *mydomain.tld*

sipml5: Expert settings

Disable Video: *unticked*
Enable RTCWeb Breaker: *ticked*
WebSocket Server URL: *ws://mydomain.tld:10060*
SIP outbound Proxy: *N/A*

Make a call!

In sipml5 you should now be connected. You can make a call by entering `200` in the Call control field. The client that you used with the `LegacySIPClient` account should now start ringing. Once the call gets accepted audio and video should start flowing between both endpoints.