

Contents

- 1 FreeSWITCH Config
 - ◆ 1.1 Forcing SAVP in the SDP
- 2 Device-specific Bugs / Configuration
 - ◆ 2.1 Polycom
 - ◆ 2.2 Linksys 922/942/962
- 3 Tests / Debug
- 4 Overhead

FreeSWITCH Config

FreeSWITCH supports SRTP via SDES ([1]). No configuration is necessary (?if using default config?) on FreeSWITCH side, although your phone may require setup. Note that unless you enable TLS also, the key for the SRTP goes in the clear. For fully secure connection between your phone and the FS, use TLS + SRTP, this prevents any snooping. For end to end security, such that even FS cannot listen to your conversations, use ZRTP on both ends. FreeSWITCH allows ZRTP through it, as well as can act as ZRTP end point. But for end to end security to the called party, do not enroll into the FS ZRTP.

See Secure RTP for information on placing an SRTP call.

WARNING:

This information is regarded as dated, and there will be many/much improvements forthcoming as I learn more about this fantastical feature of FreeSWITCH to support a variety of transports including TLS for SIP, and SRTP for secure media. Testing is commencing and I'd expect that I'll have something documented the last weekend in Feb 2009.

The variable `${sip_secure_media_confirmed}` will not be set until the first `execute_extension` or `transfer` is placed. Therefore, when the call initially comes into the switch, this will fail if the call has not yet been routed even if the call has everything properly set to be deemed as secure.

Use the `${sip_secure_media_confirmed}` variable to verify SRTP is being used for a call.

Example:

```
<extension name="is_secure">
  <condition field="${sip_secure_media_confirmed}" expression="^true$">
    <action application="sleep" data="1000"/>
    <action application="gentones" data="${bong-ring}"/>
  </condition>
</extension>
```

Ideally you want the leg b (i.e. call / RTP stream from FS to other party) to also be secure. For this uncomment the line that exports `sip_secure_media=true` below. This change is to be done in the dialplan (`conf/dialplan/default.xml`)

SRTP

```
<condition field="${sip_has_crypto}" expression="^(AES_CM_128_HMAC_SHA1_32|AES_CM_128_HMAC_S
  <action application="set" data="sip_secure_media=true"/>
  <action application="export" data="sip_secure_media=true"/>
</condition>
```

In order for this to be effective, late coded negotiation must NOT to be turned on. So make sure if you have this line, you comment it : `<param name="inbound-late-negotiation" value="true"/>`

Forcing SAVP in the SDP

By default FreeSWITCH will offer both AVP and SAVP in the SDP. To force the SDP to contain only the SAVP line use the `sdp_secure_savp_only` channel variable.

To enable this for inbound calls on a gateway, when 3pcc is enabled and the FreeSWITCH is offering sdp, add an option like this:

```
<gateways>
  <X-PRE-PROCESS cmd="set" data="sdp_secure_savp_only=true"/>
</gateways>
```

For outbound routes, add an option like this to your dial plan:

```
<condition field="destination_number" expression="^(72\d{2})$" >
  <action application="export" data="sip_secure_media=true" />
  <action application="export" data="sdp_secure_savp_only=true" />
</condition>
```

Device-specific Bugs / Configuration

Polycom

(Tested on SoundPointIP 501- should work with others, requires SIP 2.X Firmware) Polycom bug workaround- `sec.srtp.offer.HMAC_SHA1_80="0"` (phones won't negotiate SDES properly otherwise) The phone will display a lock next to the line key icon after the call is successfully connected via SRTP.

```
<security>
  <SRTP sec.srtp.enable="1" sec.srtp.leg.enable="1" sec.srtp.offer="1" sec.srtp.require="1"
    sec.srtp.offer.HMAC_SHA1_80="0" sec.srtp.offer.HMAC_SHA1_32="1" sec.srtp.key.lifetime=
    sec.srtp.mki.enabled="" sec.srtp.sessionParams.noAuth.offer="" sec.srtp.sessionParams.
    sec.srtp.sessionParams.noEncrypRTP.offer="" sec.srtp.sessionParams.noEncrypRTP.require
    sec.srtp.sessionParams.noEncrypRTCP.offer="" sec.srtp.sessionParams.noEncrypRTCP.requi
    sec.srtp.sessionParams.leg.noAuth.offer="" sec.srtp.sessionParams.leg.noAuth.require=""
    sec.srtp.sessionParams.leg.noEncrypRTP.offer="" sec.srtp.sessionParams.leg.noEncrypRTP
    sec.srtp.sessionParams.leg.noEncrypRTCP.offer="" sec.srtp.sessionParams.leg.noEncrypRT
    sec.srtp.sessionParams.IP_4000.noAuth.offer="" sec.srtp.sessionParams.IP_4000.noAuth.r
    sec.srtp.sessionParams.IP_4000.noEncrypRTP.offer="" sec.srtp.sessionParams.IP_4000.noE
    sec.srtp.sessionParams.IP_4000.noEncrypRTCP.offer="" sec.srtp.sessionParams.IP_4000.no
    sec.srtp.leg.allowLocalConf="" />
</security>
```

This can be added to a custom config file, or changed from defaults in the sip.cfg file distributed w/ firmware.

Linksys 922/942/962

All Linksys 9x2 model phones can do proper ([RFC 3711](#)) SRTP negotiation since firmware version 6.1.3a. Earlier versions support only Linksys old-style proprietary protocol. Linksys 921 and 941 models cannot be upgraded to firmware versions later than 5.1.8 where TCP, TLS and SRTP is not supported at all. UPDATE: It is observed that after a certain period of time we are hitting a bug where SRTP key is being renewed, but the server is not notified properly.

Under the SIP settings tab, there is a SRTP method drop-down menu with **x-sipura** and **s-descriptor**. The latter is set for standardized SRTP to work with FreeSWITCH. Further more, by turning **Secure Call Serv option** to **Yes**, one can easily use service activation code for the secure call (defaults are ***16** and ***17** for turning on and off the secure call option in every ***18** and ***19** just for next one call).

Tests / Debug

You can disable TLS and run a sip trace to see if the a=crypto line is there in the SIP messages. Alternatively you can do a "show channels" on the console.

For a really secure setup, you want both leg a and leg b of the calls to be SRTP (and TLS of course). For TLS support see the wiki on TLS. To check that leg b is secure, use your phone, to Wireshark, use sip trace or check console log if "sip_secure_media" is being exported.

Overhead

SRTP adds extra 4 bytes to the packets. At 20ms this means $50 * 4 * 8 = 1600$ bit/s overhead.