SIP Testing w/ FreeSWITCH

ClueCon, August 2013

Moisés Silva <moy@sangoma.com>
Manager, Software Engineering

SANGOMA

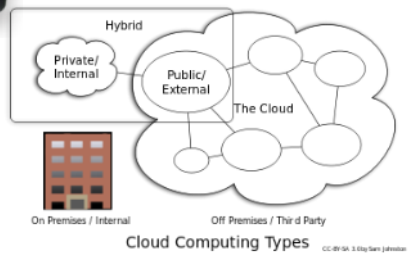
CONNECT WITH SANGOMA

About Sangoma

- Industry pioneer with over 25 years of experience in communications hardware and software
- Publicly traded company since 2000
  - TSXV: STC
- One of the most financially healthy companies in our industry
  - Growing, Profitable, Cash on the Balance Sheet, No Debt
- Mid-market sized firm with just under 100 staff in all global territories
  - Offices in Canada (Toronto), US (CA, NJ), EU (UK & Holland), APAC (India), CALA (Miami)
- World wide customer base
  - Selling direct to carriers and OEMs
  - Selling to the enterprise through a network of distribution partners

Broad Line of Great Products

- Voice Telephony Boards
  - Analog/digital/hybrid, WAN, ADSL
- Session border controllers
- Microsoft Lync
- VoIP Gateways
  - NetBorder SIP to TDM
  - SS7 to SIP
- Software Applications
  - NetBorder Express, Call Progress Analyzer…
- Transcoding (boards/appliances)
- Fiber connectivity (STM1)
- Wireless products (GSM)



Agenda

- Testing Overview
- Functionality Tests
- Load Tests
- Security Tests

Overview

- I know, SIP testing can be scary

Overview

- Testing complex systems requires detailed engineering and deep knowledge of OSes, wide range of protocols, hardware, etc
- Not everyone likes doing it, it is not glamorous work …
- But … It's developer's responsibility to test, not customer's … shocking!

Overview

- Lots of open source tools out there that can be used for testing:
  - Sipp
  - Sipsak
  - Sipvicious
  - Voiper
  - FreeSWITCH
  - Asterisk

Overview

- Commercial tools as well
  - IXLoad from Ixia
  - SIP Hammer from Empirix

Overview

- FreeSWITCH can be used to test other systems
  - Generate calls with full RTP wide array of codecs
  - Support for IPv4/IPv6, TLS, SRTP, STUN, ICE etc
  - Flexible programmable logic via XML, Python etc
  - Originate/terminate T.38 faxing
  - Originate/terminate SIP/TDM calls (and others)
  - Easy to hook up modules to test media or signaling:
    - Example: tone_detect, mod_bert, fs_test

Functionality Tests

Functionality Tests

Functionality Tests

- Verify expected SIP behaviors
  - REFER actually places a new call to given destination
  - 183 with SDP actually bridges media
  - 4/5XX responses hang up or retry a call
  - REGISTER creates an AOR in your DB
  - … And you can go crazy with Presence tests …

Functionality Tests

- Identify your most important functionality
- Execute manual tests, take traces (pcap/ wireshark)
- Write test scenarios for them
- Automate them! (Python/Ruby/PERL scripting)

SIPp

SIPp

- FreeSWITCH Wiki SIPP Quote

“IF YOU DO NOT UNDERSTAND HOW TO STRESS TEST PROPERLY THEN DON'T BOTHER

Using SIPp is part dark art, part voodoo, part Santeria.

YOU HAVE BEEN WARNED”

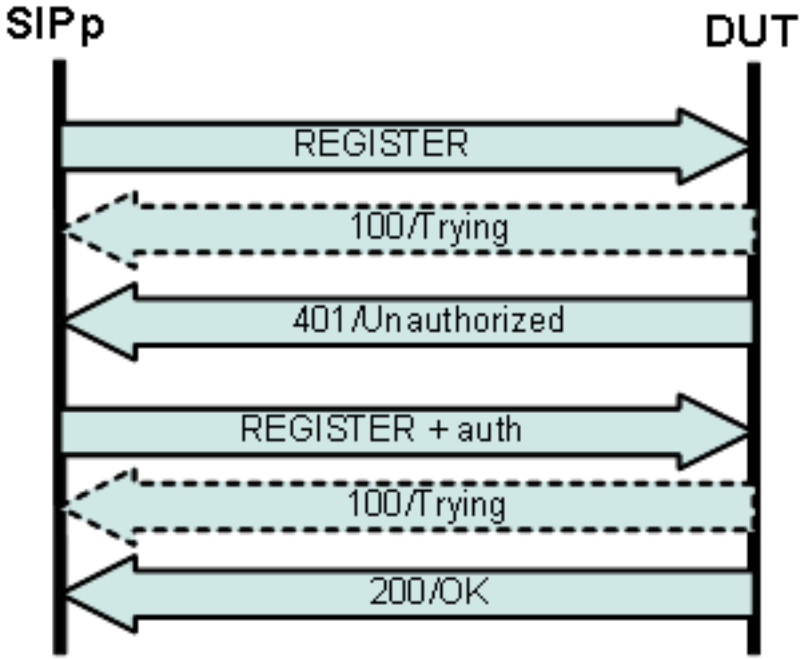
SIPp

- Low-level SIP functionality & performance test tool
- Not super user-friendly, errors can go unnoticed
- Requires a firm grasp on SIP (requests, responses, transactions, dialogs)
- Flow logic is XML-based

SIPp

```
<scenario name="Basic Sipstone UAC">
  <!-- In client mode (sipp placing calls), the Call-ID MUST be         -->
  <!-- generated by sipp. To do so, use [call_id] keyword.               -->
  <send retrans="500">
    <![CDATA[

      INVITE sip:3333@sip2sip.info SIP/2.0
      Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
      From: "Anonymous"<sip:Anonymous@[local_ip]:[local_port]>;tag=[pid]SIPpTag00[call_number]
      To: sut <sip:[service]@[remote_ip]:[remote_port]>
      Call-ID: [call_id]
      CSeq: 1 INVITE
      Contact: sip:sipp@[local_ip]:[local_port]
      Max-Forwards: 70
      Subject: Performance Test
      Content-Type: application/sdp
      Content-Length: [len]

      v=0
      o=user1 53655765 2353687637 IN IP[local_ip_type] [local_ip]
      s=-
      c=IN IP[media_ip_type] [media_ip]
      t=0 0
      m=audio [media_port] RTP/AVP 0
      a=rtpmap:0 PCMU/8000

    ]]>
  </send>

  <recv response="100"
        optional="true">
  </recv>
```

SIPp


SIPp
DUT
REGISTER
100/Trying
401/Unauthorized
REGISTER + auth
100/Trying
200/OK

SIPp

- <send>, <recv>, <pause>, <exec>, rinse & repeat
- <send> sends raw SIP messages
- <recv> indicates you are expecting a SIP response or request
- <pause> waits some milliseconds
- <exec> Can be used to play a pcap (and other stuff)

SIPp

- <send> takes care of re-transmissions if "retrans" attribute is used
- <recv> blocks if non-optional
- <exec> playing a file is non-blocking (surprising if you know FreeSWITCH/Asterisk playback)

SIPp

- More complex scenarios can be created with conditional branching
- Use statistical branching to add some variety to your scenarios
- <pause> can be done using different distribution models such as normal, exponential, pareto, etc

SIPp

- Subtle mistakes can go unnoticed (no media)

```
<send>
  <![CDATA[

    SIP/2.0 183 Session Progress
    [last_Via:]
    [last_From:]
    [last_To:];tag=[pid]SIPpTag01[call_number]
    [last_Call-ID:]
    [last_CSeq:]
    Contact: <sip:[local_ip]:[local_port];transport=[transport]>
    Content-Type: application/sdp
    Content-Length: [len]

    v=0
    o=user1 53655765 2353687637 IN IP[local_ip_type] [local_ip]
    s=-
    c=IN IP[media_ip_type] [media_ip]
    t=0 0
    m=audio 20000 RTP/AVP 0
    a=rtpmap:0 PCMU/8000

  ]]>
</send>

<!-- Play a pre-recorded PCAP file (RTP stream)                    -->
<nop>
  <action>
    <exec play_pcap_audio="g711a.pcap"/>
  </action>
</nop>
<pause milliseconds="2000"/>
```

SIPp

- Use [media_port] tag, do not hard-code ports in the SDP

```
<send>
  <![CDATA[

    SIP/2.0 183 Session Progress
    [last_Via:]
    [last_From:]
    [last_To:];tag=[pid]SIPpTag01[call_number]
    [last_Call-ID:]
    [last_CSeq:]
    Contact: <sip:[local_ip]:[local_port];transport=[transport]>
    Content-Type: application/sdp
    Content-Length: [len]

    v=0
    o=user1 53655765 2353687637 IN IP[local_ip_type] [local_ip]
    s=-
    c=IN IP[media_ip_type] [media_ip]
    t=0 0
    m=audio [media_port] RTP/AVP 0
    a=rtpmap:0 PCMU/8000

  ]]>
</send>

<!-- Play a pre-recorded PCAP file (RTP stream)                    -->
<nop>
  <action>
    <exec play_pcap_audio="g711a.pcap"/>
  </action>
</nop>
<pause milliseconds="2000"/>
```

SIPp

- Make sure you use –rtp_echo
- Make sure you insert a <pause> after playing a pcap and make sure the pcap is long enough
- For load tests raise your process limits (ulimit –a for details)

SIPp

- Automating creation of SIPp scenarios out of pcap captures:
  - Sippie
    - http://sourceforge.net/projects/sippie/
  - Sniff2sipp
    - http://svnview.digium.com/svn/sniff2sipp/

SIPSak

- Mostly useful for flood tests
- Much simpler/smaller than sipp, but less control
- Easily used for RFC4475 testing (SIP Torture)

FreeSWITCH

- You can create SIP flows indirectly using FreeSWITCH applications
- No direct/raw SIP access, but possible through FreeSWITCH channel variables
- Logic programmable in XML, Python, LUA etc

FreeSWITCH

- Use ESL originate to send INVITEs
- fs_test Python script mimics some SIPp options
  - https://github.com/moises-silva/fs_test
- Control INVITE SIP headers through “sip_h_” originate variables
- Send REFER with “deflect” application

FreeSWITCH

- Send 180 with “ring_ready”
- Send 183 with “pre_answer”
- Send 200 with “answer”
- Send 3XX with “redirect”
- Send 4XX/5XX/6XX with “respond”
- Send BYE with “hangup”

FreeSWITCH

- G.711 media test / checking can be accomplished using mod_bert or tone_detect
  - https://github.com/moises-silva/freeswitch/tree/mod_bert

- Calls failing the media test are hung up with MEDIA_TIMEOUT reason

Load Tests

Load Tests

- Load testing can be a fine art

- Be careful and define testing scope
  - OS (Linux, Windows, 64/32 bit, OS packages versions)
  - Media features (RTP/SRTP, UDPTL, Codec)
  - Signaling Features (TLS, PRACK, Presence, T.38)
  - Hardware environment (CPU, Memory, PCI/PCIx, HD)
  - Network environment (TCP/UDP/Ethernet settings)

Load Tests

- Performance can vary widely when changing just a few environment characteristics, be sure to test after each change
- Record your findings (ie: use Cacti)
- Do no underestimate non-call-related load
  - Registrations, Presence, MWI, etc

Load Tests

- Measure your network performance / throughput
- Use good cat6 ethernet cables!
- Use Iperf
  - https://code.google.com/p/iperf/

Load Tests

- Launching iperf server

```
root@sigchld ~
# iperf3 -s --bind 192.167.167.100
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Accepted connection from 192.166.166.20, port 56457
[  5] local 192.167.167.100 port 5201 connected to 192.166.166.20 port 56458
[  7] local 192.167.167.100 port 5201 connected to 192.166.166.20 port 56459
[  9] local 192.167.167.100 port 5201 connected to 192.166.166.20 port 56460
[ 11] local 192.167.167.100 port 5201 connected to 192.166.166.20 port 56461
[ 13] local 192.167.167.100 port 5201 connected to 192.166.166.20 port 56462
[ 15] local 192.167.167.100 port 5201 connected to 192.166.166.20 port 56463
[ 17] local 192.167.167.100 port 5201 connected to 192.166.166.20 port 56464
[ ID] Interval           Transfer     Bandwidth
[  5]   0.00-120.05 sec  1.55 GBytes   111 Mbits/sec
[  7]   0.00-120.05 sec  1.55 GBytes   111 Mbits/sec
[  9]   0.00-120.05 sec  1.55 GBytes   111 Mbits/sec
[ 11]   0.00-120.05 sec  1.54 GBytes   110 Mbits/sec
[ 13]   0.00-120.05 sec  1.55 GBytes   111 Mbits/sec
[ 15]   0.00-120.05 sec  1.54 GBytes   110 Mbits/sec
[ 17]   0.00-120.05 sec  1.55 GBytes   111 Mbits/sec
[SUM]   0.00-120.05 sec  10.8 GBytes   774 Mbits/sec
```

Load Tests

- Launching iperf client

```
root@moy-nsc-2u src
# iperf3 -c 192.167.167.100 -b 1G -t 120 -P 7
Connecting to host 192.167.167.100, port 5201
[  4] local 192.166.166.20 port 56458 connected to 192.167.167.100 port 5201
[  6] local 192.166.166.20 port 56459 connected to 192.167.167.100 port 5201
[  8] local 192.166.166.20 port 56460 connected to 192.167.167.100 port 5201
[ 10] local 192.166.166.20 port 56461 connected to 192.167.167.100 port 5201
[ 12] local 192.166.166.20 port 56462 connected to 192.167.167.100 port 5201
[ 14] local 192.166.166.20 port 56463 connected to 192.167.167.100 port 5201
[ 16] local 192.166.166.20 port 56464 connected to 192.167.167.100 port 5201
[ ID] Interval           Transfer     Bandwidth
[  4]   0.00-120.01 sec  1.55 GBytes   111 Mbits/sec
[  6]   0.00-120.01 sec  1.55 GBytes   111 Mbits/sec
[  8]   0.00-120.01 sec  1.55 GBytes   111 Mbits/sec
[ 10]   0.00-120.01 sec  1.55 GBytes   111 Mbits/sec
[ 12]   0.00-120.01 sec  1.55 GBytes   111 Mbits/sec
[ 14]   0.00-120.01 sec  1.55 GBytes   111 Mbits/sec
[ 16]   0.00-120.01 sec  1.55 GBytes   111 Mbits/sec
[SUM]   0.00-120.01 sec  10.8 GBytes   776 Mbits/sec
```

Load Tests

- Do not forget to verify with bwm-ng

```
bwm-ng v0.5 (probing every 1.000s), press 'h' for help
input: /proc/net/dev type: rate
|       iface                   Rx                   Tx                Total
==============================================================================
          lo:           0.00  b/s            0.00  b/s            0.00  b/s
        eth1:         783.93 Mb/s            1.53 Mb/s          785.47 Mb/s
------------------------------------------------------------------------------
       total:         783.93 Mb/s            1.53 Mb/s          785.47 Mb/s
```

Iperf server bandwidth

```
bwm-ng v0.6 (probing every 1.000s), press 'h' for help
input: /proc/net/dev type: rate
/         iface                   Rx                   Tx                Total
==============================================================================
          lo:           0.00  b/s            0.00  b/s            0.00  b/s
         eth0:          1.25 Mb/s          786.43 Mb/s          787.68 Mb/s
------------------------------------------------------------------------------
       total:           1.25 Mb/s          786.43 Mb/s          787.68 Mb/s
```

Iperf client bandwidth

Load Tests

- Slight payload change (iperf –l 172) causes significant performance difference

```
bwm-ng v0.5 (probing every 1.000s), press 'h' for help
input: /proc/net/dev type: rate
|       iface                   Rx                   Tx                Total
==============================================================================
          lo:           0.00  b/s            0.00  b/s            0.00  b/s
        eth1:         633.87 Mb/s          11.65 Mb/s          645.53 Mb/s
------------------------------------------------------------------------------
       total:         633.87 Mb/s          11.65 Mb/s          645.53 Mb/s
```

Iperf server bandwidth

```
bwm-ng v0.6 (probing every 1.000s), press 'h' for help
input: /proc/net/dev type: rate
|        iface                   Rx                   Tx                Total
==============================================================================
            lo:           0.00  b/s            0.00  b/s            0.00  b/s
          eth0:          11.67 Mb/s         633.49 Mb/s          645.16 Mb/s
------------------------------------------------------------------------------
         total:          11.67 Mb/s         633.49 Mb/s          645.16 Mb/s
```

Iperf client bandwidth

Security Tests

Security Tests

Security Tests

- Sipvicious
- Voiper

SipVicious

- Sipvicious is handy to test your fail2ban rules
- Use svwar.py and svcrack.py to trigger your fail2ban
- Verify the host was blocked

Voiper

- Voiper is handy for fuzzy/vulnerability testing
  - http://voiper.sourceforge.net/

- Whatever you do, do not click on the last link at that page (UnprotectedHex)

Voiper

- python fuzzer.py -f SIPInviteCommonFuzzer -i 192.168.168.1 -p 5060 -a sessions/scen1 -c 0

- Tons of messages like this on FreeSWITCH:

```
nta: received INVITE sip:101@192.168.168.1 SIP/2.0 (CSeq 2147483645)
nta: INVITE has bad Contact header
nta: Via check: extra received=aaaa:aaaa:aaaa:aaaa:aaaa:aaaa:aaaa:aaaa from 192.168.168.20:56581
nta: Via check: received=192.168.168.20
nta: INVITE (2147483645) is Bad Contact Header
tport_vsend(0x852bae8): Address family not supported by protocol with (s=33 UDP/aaaa:aaaa:aaaa:aaaa:aaaa:aaaa:aaaa:aaaa:5060)
mreply: send fails
```

Voiper

- Note fail2ban can hardly help here (if at all)
- Solution is report malformed packets via events and possibly block hosts sending excess of malformed traffic

QUESTIONS

Contact Us

- Sangoma Technologies

  100 Renfrew Drive, Suite 100
  Markham, Ontario L3R 9R6
  Canada

- Website

  http://www.sangoma.com/

- Telephone

  +1 905 474 1990 x2 (for Sales)

- Email

  sales@sangoma.com

/Sangoma

/Sangoma

/SangomaTechnologies

blog.sangoma.com

THANK YOU

SANGOMA

CONNECT WITH SANGOMA