

SIP Stack Fingerprinting and Stack Difference Attacks

Hendrik Scholz

hscholz@raisdorf.net

<http://www.wormulon.net/>

Black Hat Briefings 2006, Las Vegas



Agenda

- VoIP Introduction
- SIP Fingerprinting
 - Locating Devices
 - RNG Analysis
- Stacks and Parsers
- Stack Desynchronization
- Conclusion

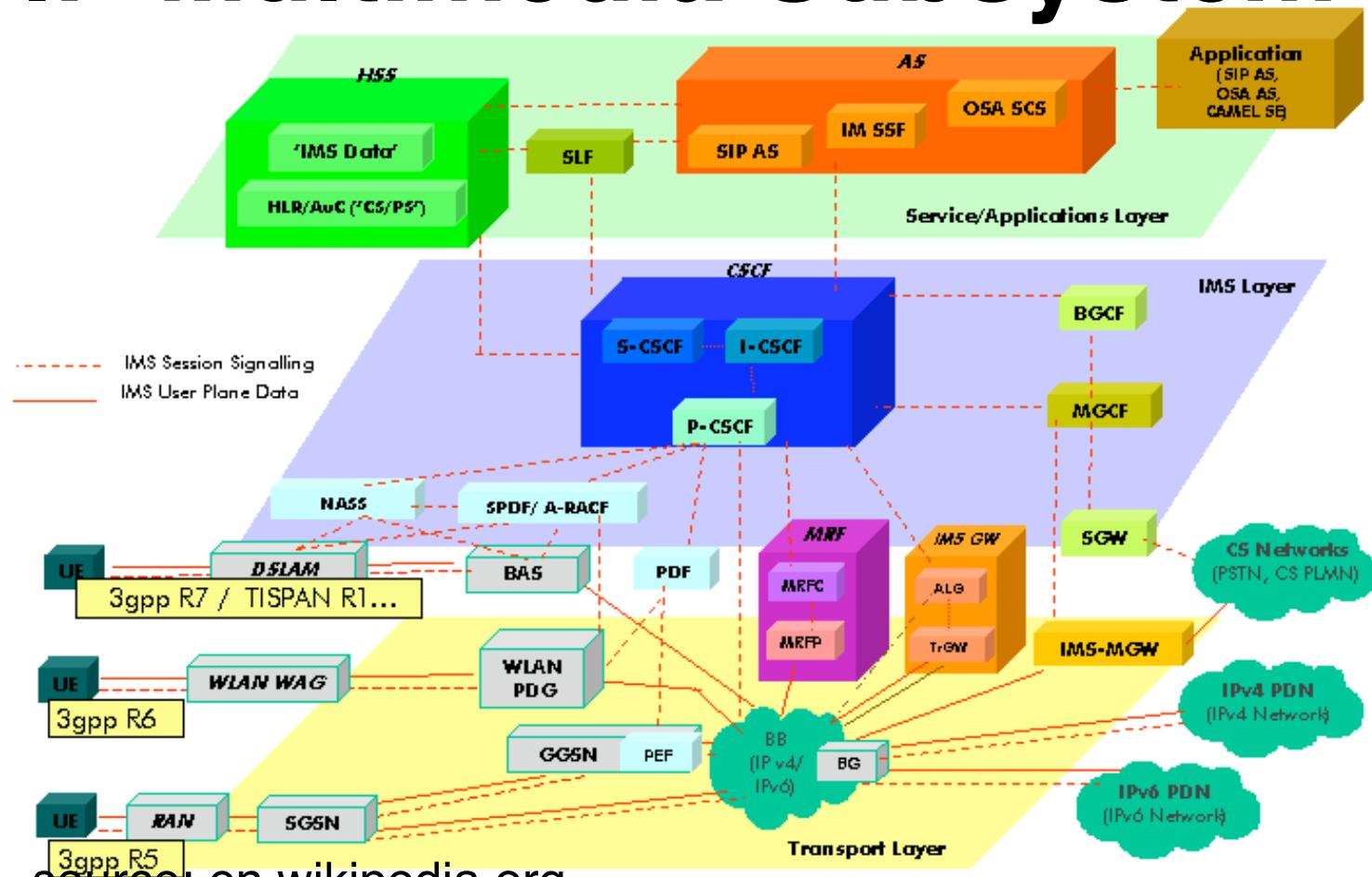


What is VoIP?

- VoIP = Voice over IP
- aims to be PSTN replacement
- traditional PSTN equipment IP enabled
- in production use today
- undergoing explosive growth



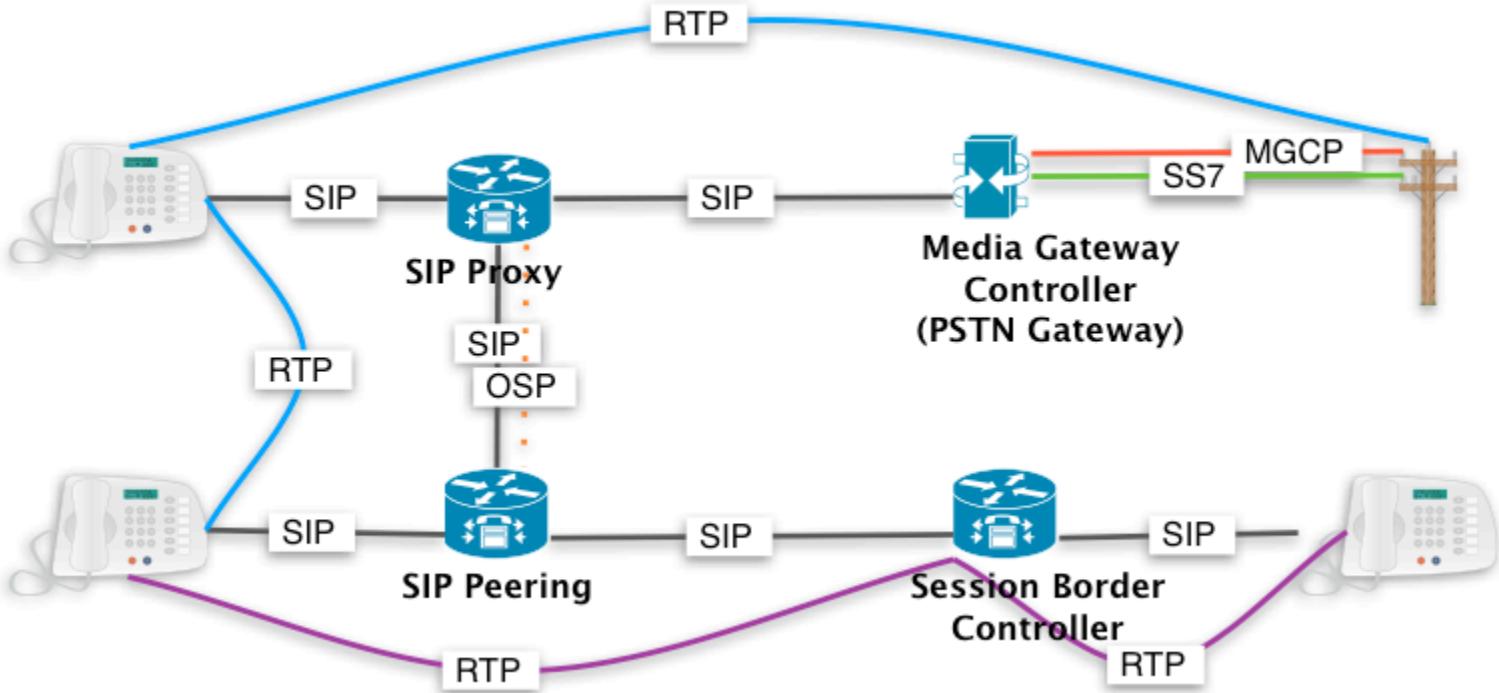
IP Multimedia Subsystem



source: en.wikipedia.org



Network Diagram



Session Initiation Protocol (SIP)



SIP RFCs – Feel Lost?

- 1847, 2045, 2046, 2047, 2048, 2198, 2327, 2543,
2616, 2617, 2633, 2733, 2791, 2833, 2848, 2959,
2976, 3087, 3050, **3204**, 3219, 3261, 3262, 3263,
3264, 3265, 3266, 3310, 3311, 3312, 3313, 3319,
3320, 3321, 3322, **3323**, 3324, 3325, 3326, 3327,
3329, 3361, 3351, 3372, 3388, 3389, 3398, 3407,
3420, 3428, 3455, 3468, 3485, **3515**, 3550, 3551,
3555, 3556, 3605, 3606, 3608, 3611, 3702, 3711,
3725, 3764, 3824, 3840, **3842**, 3856, 3857, 3890,
3891, 3903, 3911, 3959, 3960, 3968, 3969, 3976,
4028, 4077, **4083**, 4091, 4092, 4117, 4123, 4145,
4168, 4189, 4235, 4240, 4244, 4245, 4317, 4320,
4321, 4353, 4354, 4411, 4412



SIP Standards

- <http://www.packetizer.com/voip/sip/standards.html>
- 'some' additional drafts
- new RFCs/drafts on a weekly basis



SIP: Protocol Design

- plain text, http like
- Requests
 - INVITE, REGISTER, SUBSCRIBE, BYE
- Responses
 - 200 OK, 404 Not Found, 500 Server Error
- complex state engine
- supports UDP, TCP, TLS transport



Supplementary Services

- Implementation of PSTN features
- post SIP Standardization
 - not available on all devices
 - new headers
 - new methods
- multiple Implementations
 - i.e. Call Hold



Attacking VoIP Networks



Black Hat Briefings

Security Threats

- Interception & Modification
 - RTP/media attacks
 - re-routing
- Eavesdropping
 - call pattern tracking
 - number harvesting
 - communication reconstruction



Security Threats

- • Social Threads
 - Theft of Services
 - Unwanted Contact
 - Misrepresentation (Identity Theft)
- • Denial of Service
 - Flooding
 - Malformed messages
- • Combinations
 - Spoofed identity and RTP replay



Objective

- How to conduct an attack?
- Stack Desynchronization
 - multiple devices always involved
- use legitimate-looking traffic
 - circumvent IDS/IPS



How to Attack?

- Locate
 - more than just an ICMP PING
- Identify / Fingerprint
 - which stack is running?
 - Configuration
- Exploit



Locating Devices



Black Hat Briefings

Locating Devices

- SIP layer PING
 - OPTIONS request
 - INVITE, CANCEL
 - random garbage
- SIP based response is enough
 - 404 Not Found
 - 400 Bad Request (parser error)



Implementation

- mashup of sipsak and nmap
- utilizes SIP OPTIONS request
 - custom requests via CLI
- basic banner grabbing



smap Output

```
$ smap -O -t 200 89.53.10.0/24

scanning 89.53.10.0... timeout
scanning 89.53.10.1... timeout
...
scanning 89.53.10.8... up
User-Agent: AVM FRITZ!Box Fon WLAN 7050 14.04.01 (Jan 25 2006)
scanning 89.53.10.9... up
User-Agent: AVM FRITZ!Box Fon WLAN 7050 14.04.01 (Jan 25 2006)
scanning 89.53.10.10... up
User-Agent: AVM FRITZ!Box Fon WLAN 7050 14.04.01 (Jan 25 2006)
...

256 hosts scanned, 114 up, 142 down, 0 errors
$ nmap -sP 89.53.10.0/24
...
Nmap run completed -- 256 IP addresses (138 hosts up) scanned
in 5.400 seconds
$
```



Fingerprinting



Black Hat Briefings

Active Fingerprinting

- Strategy
 - craft requests
 - interpret responses
- Operating System Fingerprinting
 - nmap
 - *ICMP Usage in Scanning* by Ofir Arkin



Active Fingerprinting (cont)

- Example:
 - Send ICMP Netmask request
 - Got a response? Might be Solaris
- Pro
 - on demand, can trigger bugs
- Contra
 - noisy, detectable



Passive Fingerprinting

- Strategy
 - sniff existing traffic
 - identify based on oddities
- Pro
 - undetectable
- Contra
 - hard to differ between minor versions



SIP Fingerprinting



Black Hat Briefings

Whitehat Rationale

- Tracking down interworking issues
- Identification of malicious devices
- Prevention/detection of attacks
 - drop INVITEs from non-interoperable devices
 - lower impact of faulty clients
- SPIT bots will be small, not feature-blown



Blackhat Rationale

- Identify and locate specific devices
- Identify exploitworthy boxes
 - 4 T1 lines vs. 2 analogue lines
- Disguise program as being legit
 - honeynet nmap feature



Requirements

- • Blackhat
 - locate devices
 - do it fast (low VoIP per IP ratio)
 - fingerprint devices (actively)
- • Whitehat
 - passive Fingerprinting
 - IDS/IPS functionality
 - resource conservative



passive Fingerprinting

- order/existance of headers
 - i.e. Accept header set?
- order/formatting inside headers
 - brackets
 - displayname
 - order of tags
- interpretation of RFCs
 - Max-Forwards set to !70



active Fingerprinting

- test implemented methods
- response to unsupported messages
- response to fuzzed lines
- response on busy
 - timing
- response to unsupported media
 - 415, 486, 603



Sample PDU

```
OPTIONS sip:freenet.de SIP/2.0
Via: SIP/2.0/UDP
    192.168.178.22:64401;branch=z9hG4bK.3704f405;rport;alias
From: sip:sipsak@192.168.178.22:64401;tag=5463c52e
To: sip:freenet.de
Call-ID: 1415824686@192.168.178.22
CSeq: 1 OPTIONS
Contact: sip:sipsak@192.168.178.22:64401
Content-Length: 0
Max-Forwards: 70
User-Agent: sipsak 0.9.6
Accept: text/plain
```



Randomness

- unique per-session strings used to match messages
 - Call-ID
 - To/From tags
 - Call Sequence (CSeq)
 - Via branch
- issues
 - predictable
 - information leakage



Call-ID Implementations

- Analysis of
 - sipsak
 - sipp
 - opal
 - Asterisk
 - Teles iSwitch
 - Cisco PGW
- Newport SBC (Via branch)



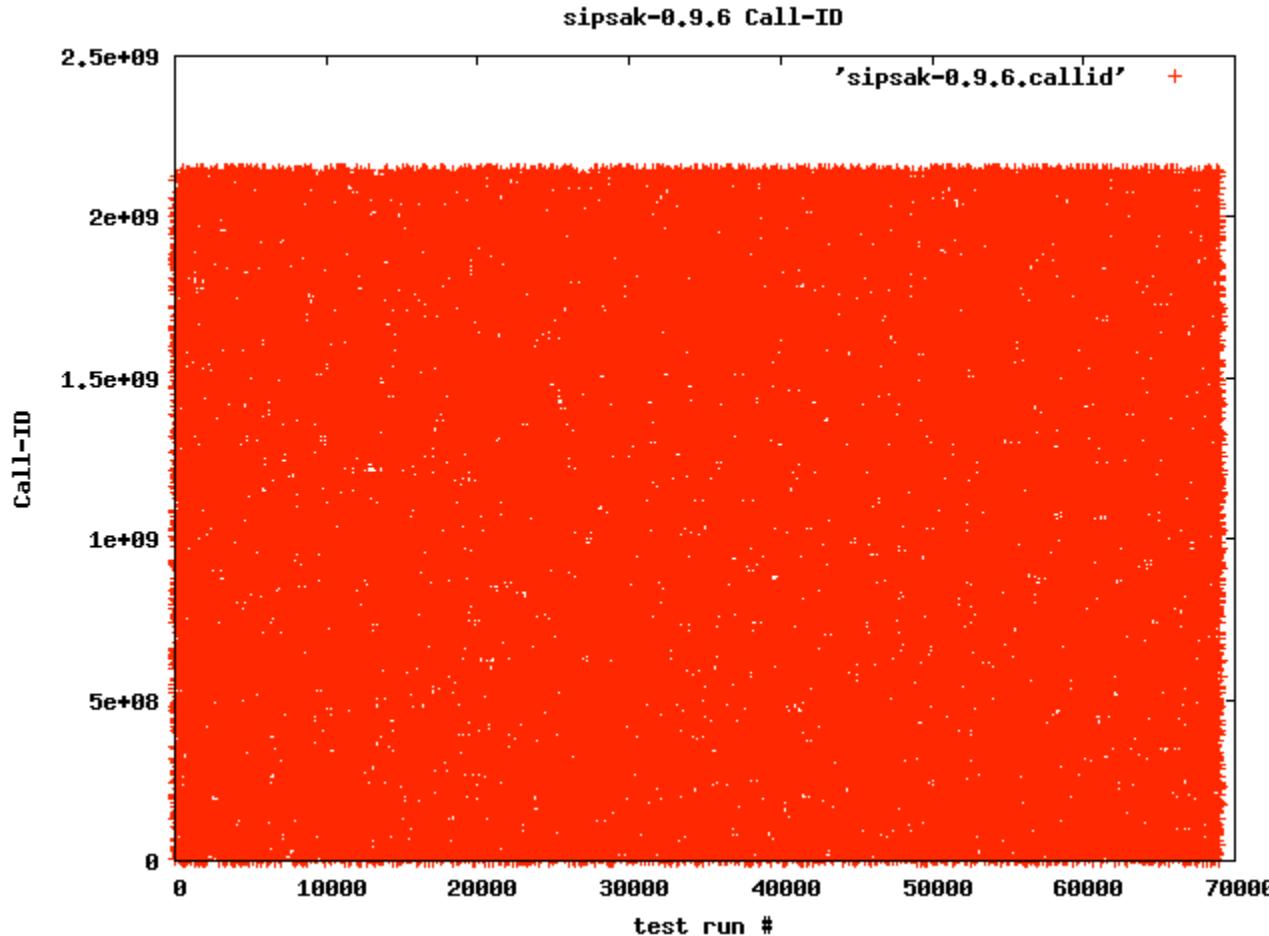
Call-ID: sipsak

- <http://sipsak.org/>
- stateless test tool
- Call-ID generator:

```
srand(time(0) ^ getpid());  
c = (unsigned int) rand();  
c+= lport; /* local UDP port */
```
- just works



Call-ID: sipsak

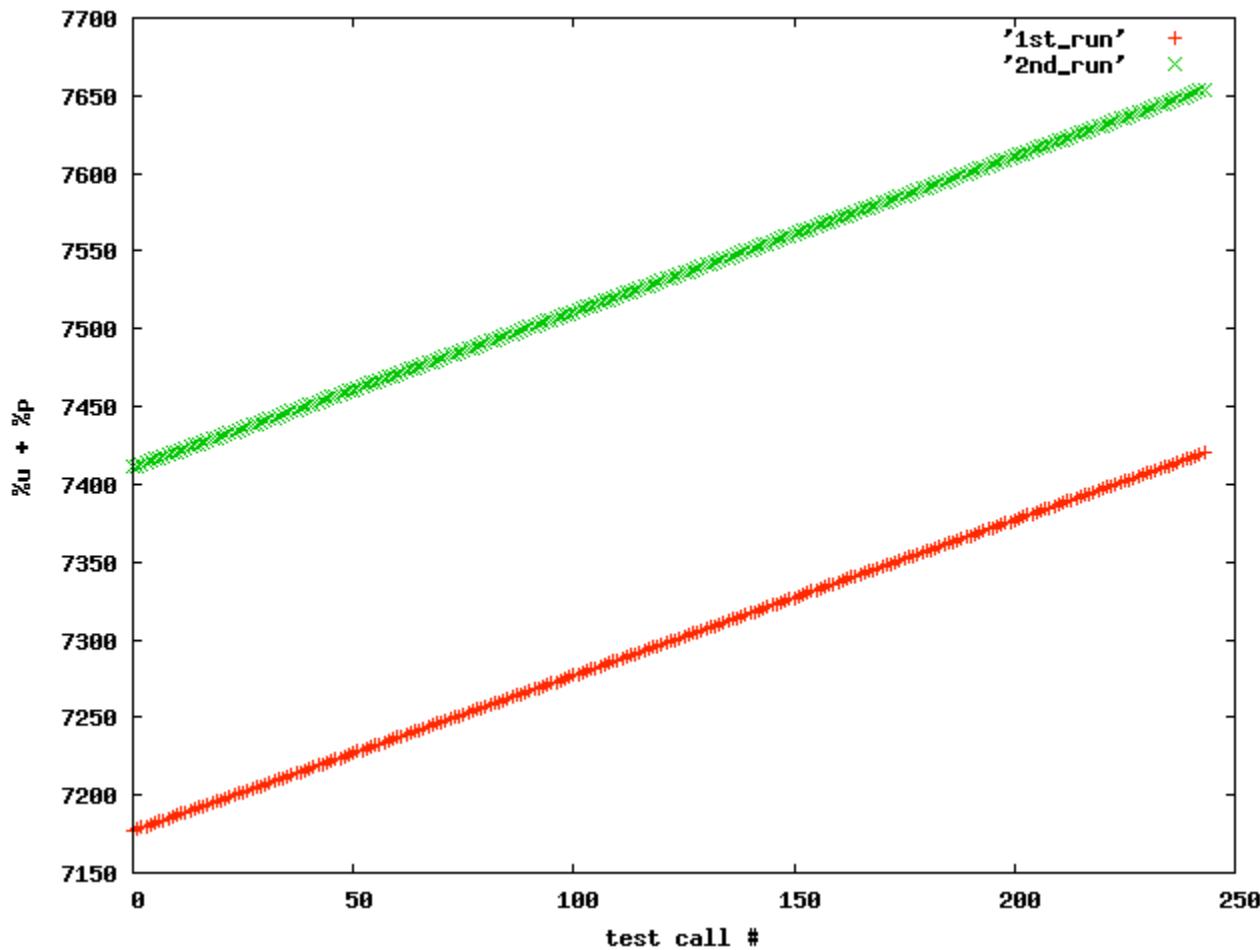


Call-ID: sipp

- <http://sipp.sf.net/>
 - CLI
 - call generator
 - performance tests
- Call-ID
 - %u-%p@%s
 - <unsigned int> - <PID> @ <local IP>



Call-ID: sipp



Call-ID: opal

- <http://openh323.org/>
 - Open Phone Abstraction Library
 - OpenH323 successor
 - foundation for Ekiga
- Call-ID opal/guid.hxx:

```
PString id =
    OpalGloballyUniqueID().AsString() +
    „@“ + PIPSocket::GetHostName();
```



Call-ID: opal (cont)

```
■ OpalGloballyUniqueID::OpalGloballyUniqueID()
  : PBYTEArray(GUID_SIZE)
{
    // Want time of UTC in 0.1 microseconds since 15 Oct 1582.

    PInt64 timestamp;
    static PInt64 deltaTime = PInt64(10000000)*24*60*60*
        ( 16                      // Days from 15th October
          + 31                     // Days in December 1583
          + 30                     // Days in November 1583

■ #ifdef _WIN32
        + (1601-1583)*365      // Whole years
        + (1601-1583)/4);     // Leap days

■ // Get nanoseconds since 1601
```



Call-ID: opal (cont)

```
theArray[0] = (BYTE)(timestamp&0xff);
theArray[1] = (BYTE)((timestamp>>8)&0xff);
theArray[2] = (BYTE)((timestamp>>16)&0xff);
theArray[3] = (BYTE)((timestamp>>24)&0xff);
theArray[4] = (BYTE)((timestamp>>32)&0xff);
theArray[5] = (BYTE)((timestamp>>40)&0xff);
theArray[6] = (BYTE)((timestamp>>48)&0xff);
theArray[7] = (BYTE)((((timestamp>>56)&0x0f) + 0x10); // Version
                     number is 1

theArray[8] = (BYTE)((((clockSequence>>8)&0x1f) | 0x80); // DCE
                     compatible GUID
theArray[9] = (BYTE)clockSequence;

memcpy(theArray+10, macAddress.b, 6);
```



Call-ID: opal (cont)

- MAC address part of unique IDs
 - everything that uses `OpalGloballyUniqueID()`
- unique identification of clients
 - one client using multiple accounts
 - one client registered at multiple registrars
 - SPIT bot initiating calls



Call-ID: Asterisk

- chan_sip.c: build_callid()
- Asterisk 1.0.0 – 1.1.x

```
val = rand();  
snprintf(callid, len, "%08x", val);
```
- Asterisk 1.2.0 – 1.2.9.1

```
val = thread_safe_rand();  
snprintf(callid, len, "%08x", val);
```
- Call-ID collisions on pre 1.2.0
 - issue #5712



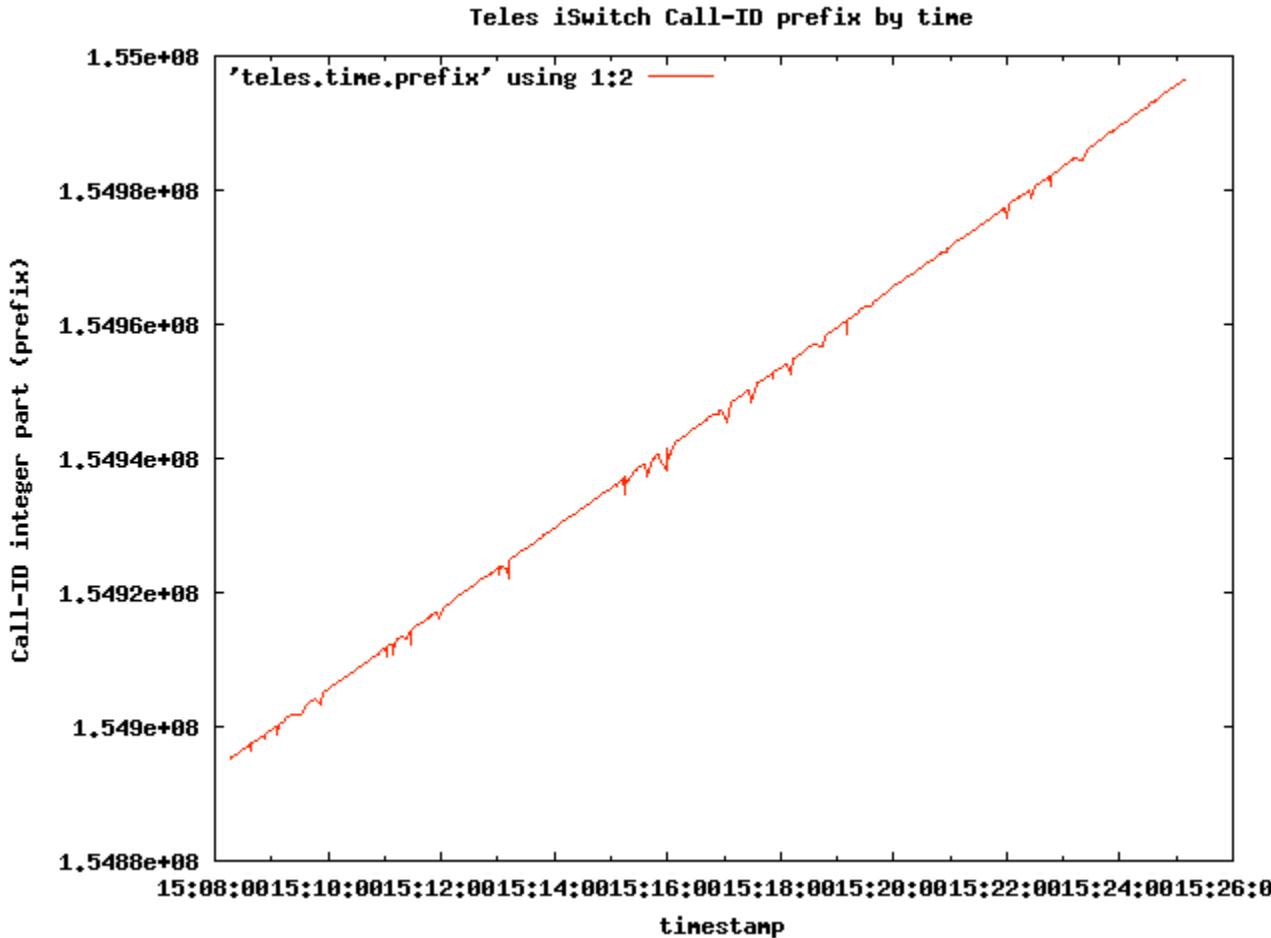
Call-ID: Teles iSwitch

- Call-ID contains MAC address
 - identification of physical hardware
 - randomness limited to few Bytes
- Call-ID prefix recycled in
 - branch
 - To/From tag

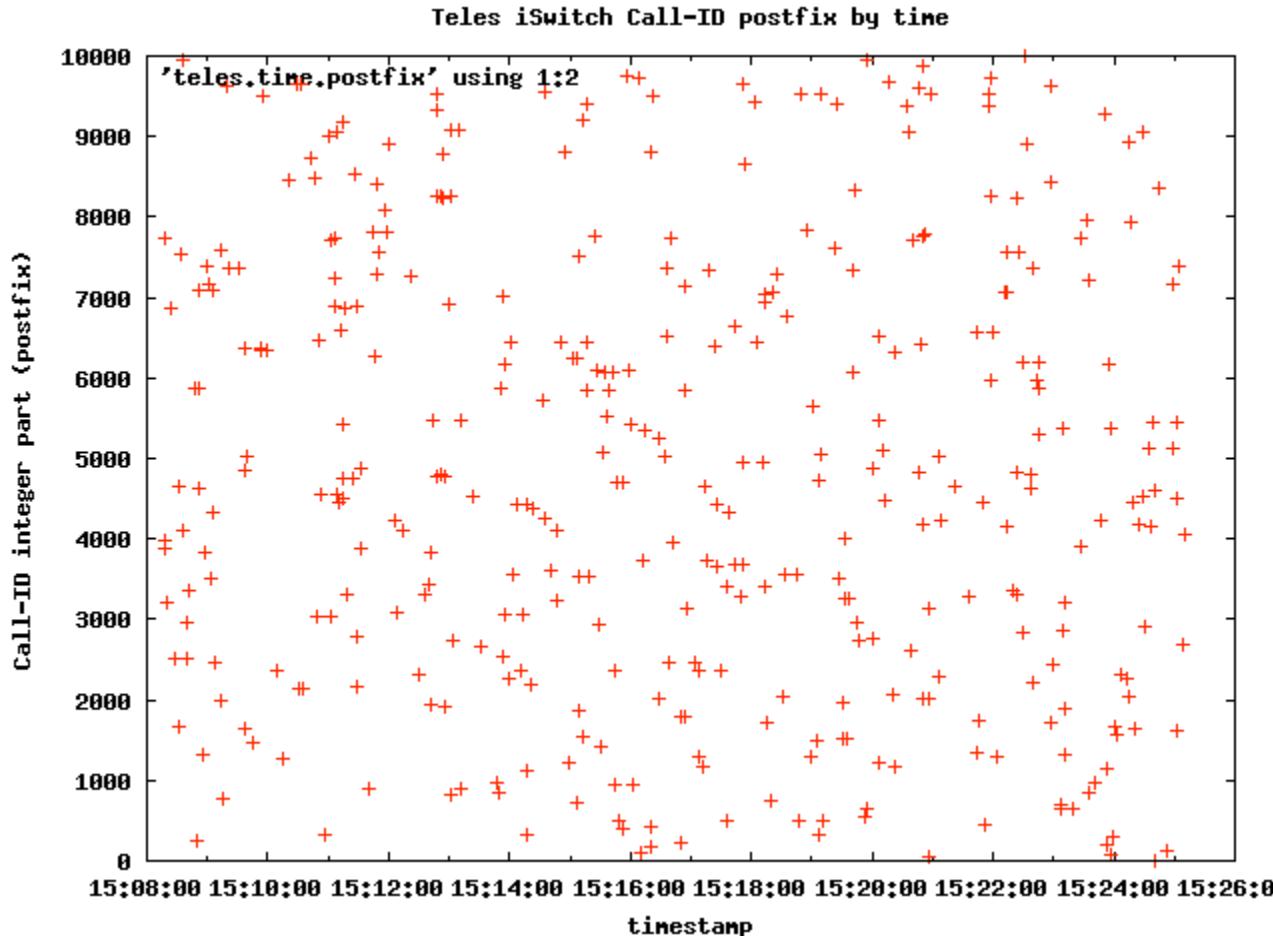
008082384A39**093B8B14**000026E4@10.1.1.1



Call-ID: Teles iSwitch 'prefix'



Call-ID: Teles iSwitch 'postfix'



Newport SBC

- branch leaks information

Via: SIP/2.0/UDP 10.1.1.66:5060;branch=z9hG4bKterm-1845faf-4931082470-493130162115.

Via: SIP/2.0/UDP 10.1.1.66:5060;branch=z9hG4bKterm-1845fb0-49310995520-49310995108.

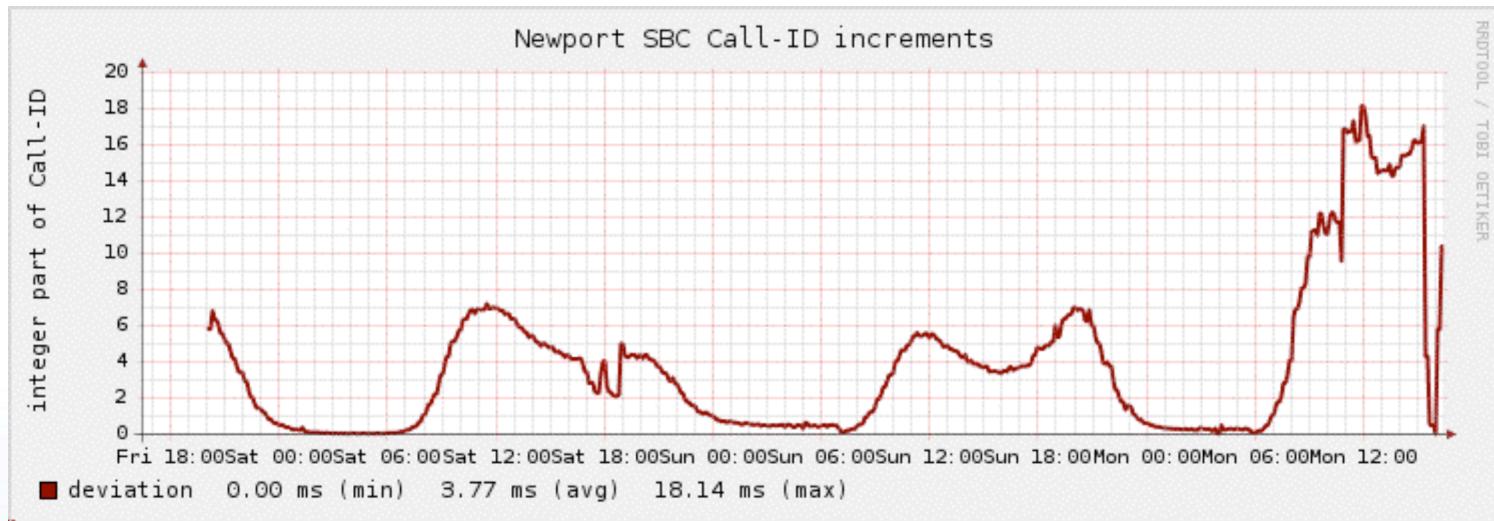
Via: SIP/2.0/UDP 10.1.1.66:5060;branch=z9hG4bKterm-1845fb1-493142973448-4931422104.

- contains A and B phone #
 - even with set CLIR
- incrementing counter
 - Calls/sec



Newport SBC: Call-ID

- obtain calls per seconds
 - even if not all INVITEs are visible



Related Fingerprinting Work

- *Incorporate Active Fingerprinting into SPIT Prevention Systems*
 - by Zon-Yin Shae at 3rd VoIP Security WS
 - Analysis of SIP header order/existance



Stacks and Parsers



Black Hat Briefings

Stack Torture Tests

- SIP torture tests
 - PROTOS test suite
 - RFC 4475 Torture Test Messages
- limited to one Stack/parser



Comparing SIP parsers

- throw traffic at stacks
- compare parsed results
- Stacks
 - SER, OpenSER
 - libosip2
 - sofia
 - SBC, IP PBX, end user devices



Iptel SER vs. libosip2

- Implementation:
 - pcap/libnids interface to read traffic
 - throw packet at both libraries
 - parse message
 - fill meta structure
 - compare meta structure content



stackcmp test (1)

- individual parser fails

```
$ LD_LIBRARY_PATH=~/CVS/sip_router/lib/cds ./stackcmp -r  
~/dump/sip.cap  
DEBUG: osip_parsebuf() failed  
...  
From: <sip:abc,scholz@freenet.de>;tag=1223992913  
...  
DEBUG: ser_parsebuf() failed  
...  
To: „Leitung 2“ <sip:abc bar@10.184.138.82:5060>  
$
```



stackcmp test (2)

- 'successful' parsing
- **comparison fails**

```
LD_LIBRARY_PATH=~/CVS/sip_router/lib/cds ./stackcmp -r  
~/dump/sip-fe.cap41032
```

...

```
OSIP To: uri='sip:claus.bachmann@freenet.de',  
display='', tag=''
```

```
SER To: uri='sip:claus.%20bachmann@freenet.de',  
display='', tag=''
```



Stack Comparison Results

- Iptel SER
 - designed to ignore + fix bugs
 - hardly ever fails
- libosip2
 - 4-5x slower than SER
 - fails on various messages



SER/OpenSER Results

- 'same' parser
- OpenSER faster (mem. management)
- accepts invalid traffic
 - unescaped % (should read %25)
 - lowercase methods



libosip2 Results

- accepts spaces in URIs
 - doesn't make any sense
 - could trigger error in application
- comma not accepted in displayname

From: „Scholz, H.“ <sip:hs@123.org>



Exploitation of Stack Differences

Stack Desynchronization



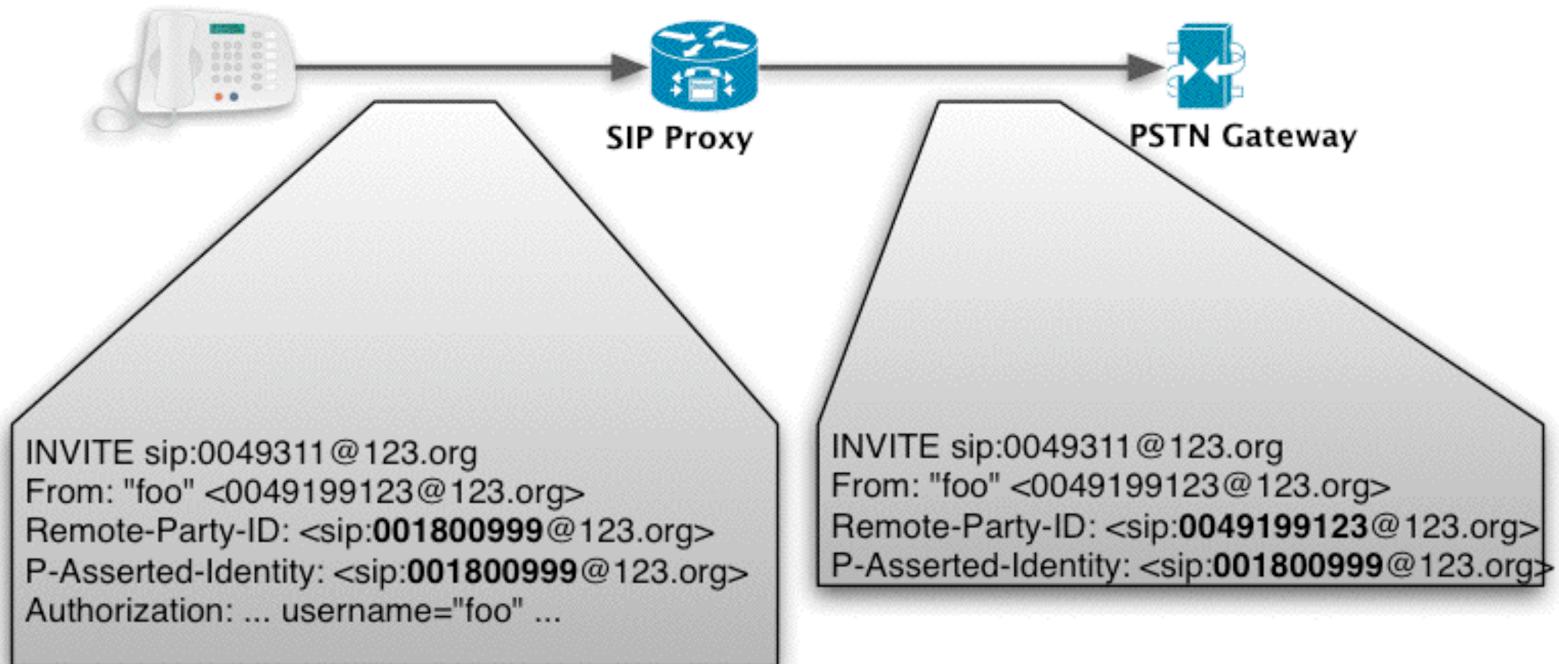
Caller-ID spoofing

- Implementations
 - To/From fields
 - Remote-Party-ID
 - RFC3323/RFC3325
- Privacy is post-RFC3261
 - devices might not support it
 - network elements might not filter it



Caller-ID spoofing

- Authentication/Authorization by ID
 - calling your own cell phone mailbox



Resources on CD

- smap
 - locating devices
- parser_test
 - find messages SER couldn't parse
- stackcmp
 - stack comparison tool
- sipfp
 - passive SIP fingerprinting tool



Conclusions

- • Passive Fingerprinting
 - IDS as second line of Defense possible
 - SPIT detection/countermeasures
 - (still) sufficient in most cases
- • Active Fingerprinting
 - possible
 - probably doesn't scale



Thanks for your time!
Questions?

