OpenSIPS 2.3
Capturing beyond SIP

Ionuţ Ioniţă

- 3 May 2017 -

AMSTERDAM OPENSIPS SUMMIT 2017

Outline

- Introduction
- Previous work
- Limitations
- Extended Tracing
- Conclusions

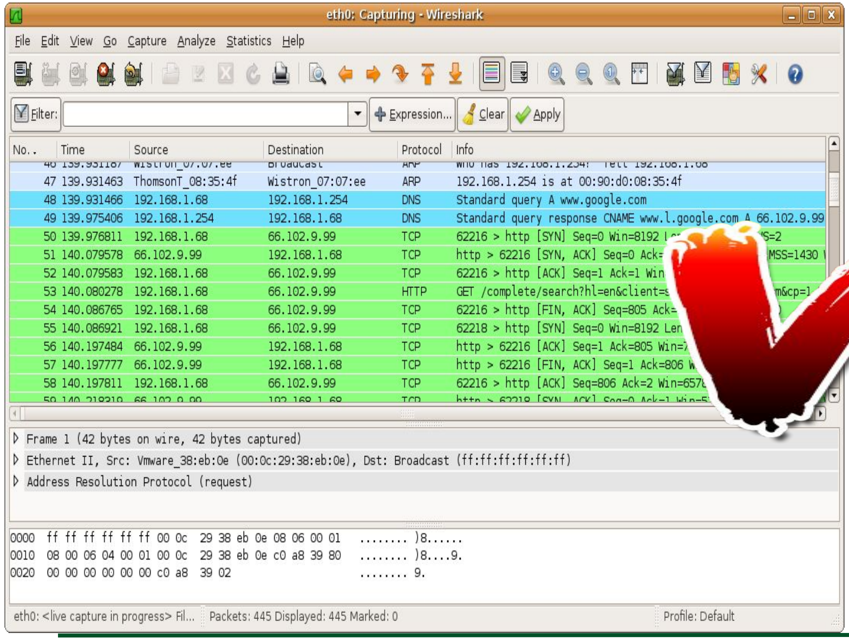
Introduction

Why trace?

- save traffic for
  - visualising statistics
  - searching through captured traffic
  - digging in for problems
- in case something goes wrong, it’s much easier to
  - inspect traffic
  - detect problems related to authentication, call failures, undesired SIP flow

Why trace?

- why not wireshark/tcpdump?

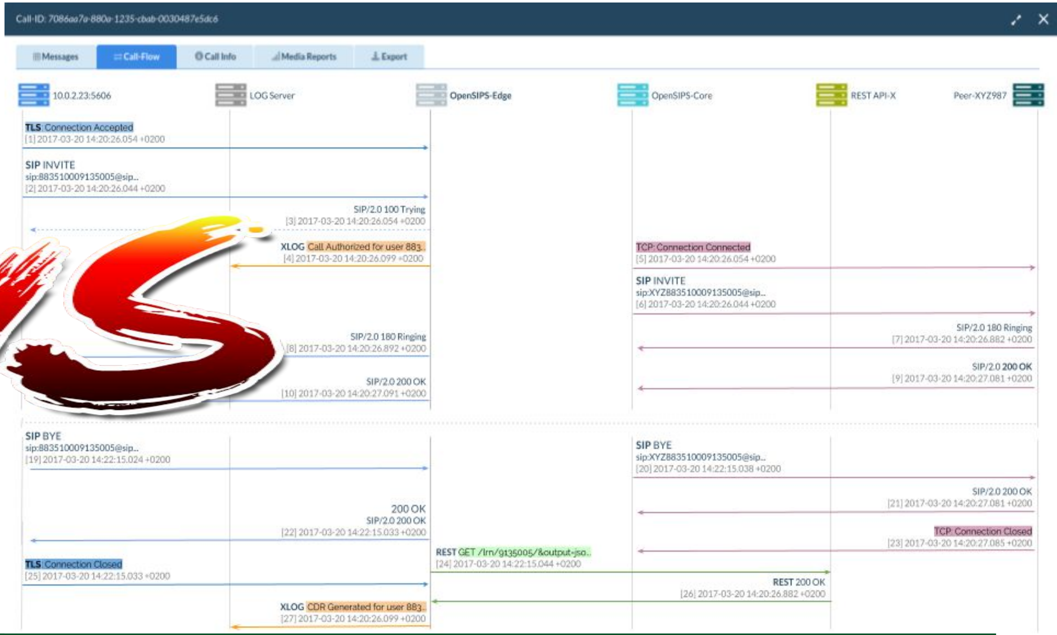

eth0: Capturing - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression...
Clear
Apply

```
No. .  Time        Source           Destination      Protocol  Info
   46 139.931187  Wistron_07:07:ee Broadcast        ARP       Who has 192.168.1.254?  Tell 192.168.1.68
   47 139.931463  ThomsonT_08:35:4f Wistron_07:07:ee ARP      192.168.1.254 is at 00:90:d0:08:35:4f
   48 139.931466  192.168.1.68     192.168.1.254    DNS       Standard query A www.google.com
   49 139.975406  192.168.1.254    192.168.1.68     DNS       Standard query response CNAME www.l.google.com A 66.102.9.99
   50 139.976811  192.168.1.68     66.102.9.99      TCP       62216 > http [SYN] Seq=0 Win=8192 L          S=2
   51 140.079578  66.102.9.99      192.168.1.68     TCP       http > 62216 [SYN, ACK] Seq=0 Ack=           MSS=1430
   52 140.079583  192.168.1.68     66.102.9.99      TCP       62216 > http [ACK] Seq=1 Ack=1 Win
   53 140.080278  192.168.1.68     66.102.9.99      HTTP      GET /complete/search?hl=en&client=s          m&cp=1
   54 140.086765  192.168.1.68     66.102.9.99      TCP       62216 > http [FIN, ACK] Seq=805 Ack=
   55 140.086921  192.168.1.68     66.102.9.99      TCP       62218 > http [SYN] Seq=0 Win=8192 Len
   56 140.197484  66.102.9.99      192.168.1.68     TCP       http > 62216 [ACK] Seq=1 Ack=805 Win=
   57 140.197777  66.102.9.99      192.168.1.68     TCP       http > 62216 [FIN, ACK] Seq=1 Ack=806 W
   58 140.197811  192.168.1.68     66.102.9.99      TCP       62216 > http [ACK] Seq=806 Ack=2 Win=657
   59 140.218319  66.102.9.99      192.168.1.68     TCP       http > 62218 [SYN, ACK] Seq=0 Ack=1 Win=5

▷ Frame 1 (42 bytes on wire, 42 bytes captured)
▷ Ethernet II, Src: Vmware_38:eb:0e (00:0c:29:38:eb:0e), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
▷ Address Resolution Protocol (request)

0000  ff ff ff ff ff ff 00 0c  29 38 eb 0e 08 06 00 01   ........ )8......
0010  08 00 06 04 00 01 00 0c  29 38 eb 0e c0 a8 39 80   ........ )8....9.
0020  00 00 00 00 00 00 c0 a8  39 02                     ........ 9.

eth0: <live capture in progress> Fil...  Packets: 445 Displayed: 445 Marked: 0   Profile: Default
```



```
TLS Connection Accepted
[1] 2017-03-20 14:20:26.054 +0200
SIP INVITE
sip:883510009135005@sip...
[2] 2017-03-20 14:20:26.044 +0200
SIP/2.0 100 Trying
[3] 2017-03-20 14:20:26.054 +0200
XLOG Call Authorized for user 883...
[4] 2017-03-20 14:20:26.099 +0200
TCP: Connection Connected
[5] 2017-03-20 14:20:26.054 +0200
SIP INVITE
sip:XYZ883510009135005@sip...
[6] 2017-03-20 14:20:26.044 +0200
SIP/2.0 180 Ringing
[7] 2017-03-20 14:20:26.882 +0200
SIP/2.0 180 Ringing
[8] 2017-03-20 14:20:26.892 +0200
SIP/2.0 200 OK
[9] 2017-03-20 14:20:27.081 +0200
SIP/2.0 200 OK
[10] 2017-03-20 14:20:27.091 +0200
SIP BYE
sip:883510009135005@sip...
[19] 2017-03-20 14:22:15.024 +0200
SIP BYE
sip:XYZ883510009135005@sip...
[20] 2017-03-20 14:22:15.038 +0200
SIP/2.0 200 OK
[21] 2017-03-20 14:20:27.081 +0200
200 OK
SIP/2.0 200 OK
[22] 2017-03-20 14:22:15.033 +0200
TCP: Connection Closed
[23] 2017-03-20 14:20:27.085 +0200
REST GET /lrn/9135005/&output-jso...
[24] 2017-03-20 14:22:15.044 +0200
TLS Connection Closed
[25] 2017-03-20 14:22:15.033 +0200
REST 200 OK
[26] 2017-03-20 14:20:26.882 +0200
XLOG CDR Generated for user 883...
[27] 2017-03-20 14:20:26.099 +0200
```

A story in two chapters

- Part 1 - 2.2
  - **proto_hep** module - handling **HEP messages** network logic
  - HEP oriented **sipcapture** module - capture all types of **HEP** messages
- Part 2 - 2.3
  - switch focus to **siptrace -** capture as much events as possible (xlogs, rest queries, network events, mi commands)

Previous work - 2.2

PROTO_HEP

- network level module(client and server)
- define **HEP** destinations and listeners

```
    listen=hep_tcp:10.0.0.1:6001 #server
    modparam("proto_hep", "hep_id",
"[hep_dst] 10.0.0.1:6001;version=3;transport=tcp") #client
```

- **TCP(HEP3)** and **UDP(HEP1 and HEP2)**

SIPCAPTURE

- process captured **HEP** messages
- **hep_route** - route for processing **HEP**
- **report_capture** - save **HEP** message to **DB**
- hep message setter/getter functions for **HEP** chunks

Limitations

- very powerful capturing node but…
- SIP-centric =>
  - very hard to (only via logs)
    - detect script bugs
    - debug failed **REST** queries
    - debug network failures
      - TLS or WS failed handshakes
      - reason for closed connections
  - no ways to
    - trace MI command status

Goals

- capture as much as possible

- correlate data

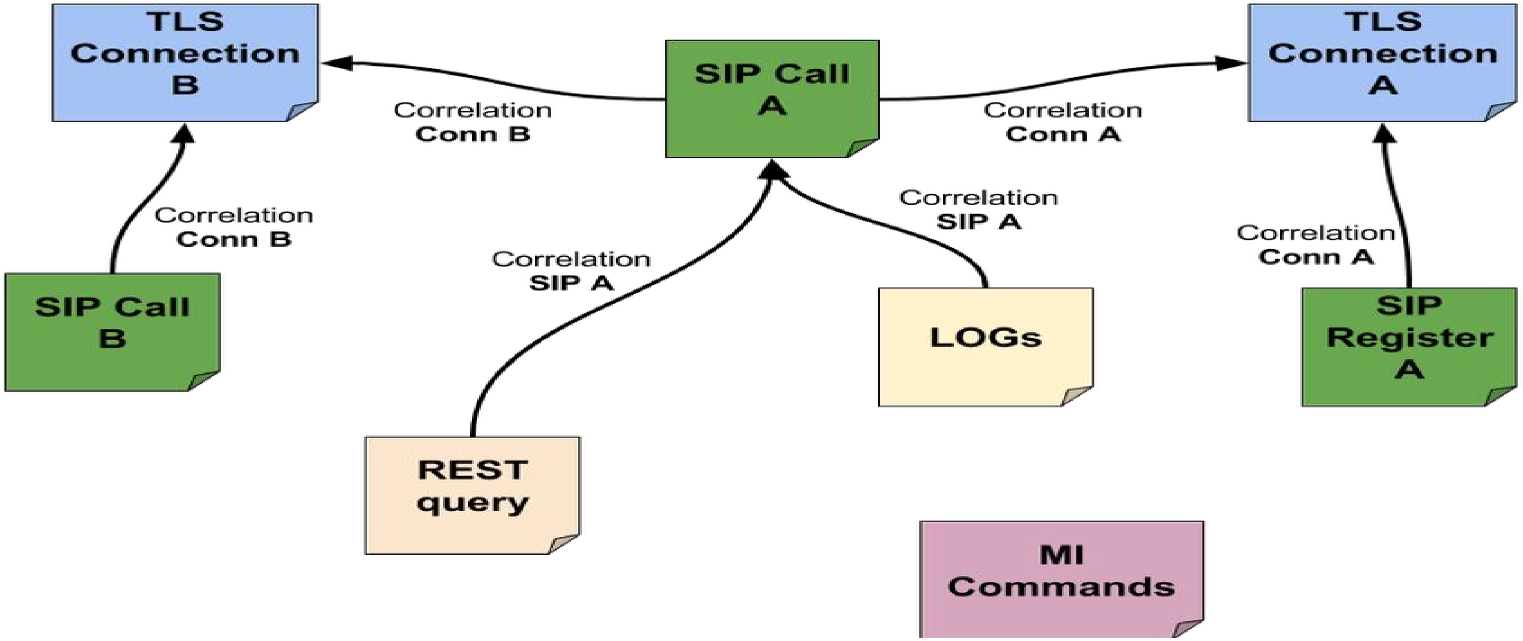
Capturing more than SIP - 2.3

PART 1 - DATA TYPES

Apart from SIP the following data types are traced:

- sip context
  - xlog messages
  - REST queries
- network level
  - All protocols that use TCP
- MI commands

PART 2 - DATA CORRELATION (1)



PART 2 - DATA CORRELATION (2)

- link HEP packets
  - **external**
  - **internal**
- points of correlation
  - **external** correlation
    - SIP => xlog, REST - callId
    - network => SIP - unique internal connection id
  - **internal** correlation
    - SIP - callId
    - REST, MI - guid generated by OpenSIPS
    - network - unique internal connection id

PART 2 - DATA CORRELATION (3)

Storing correlation data in HEP packet

- **internal** correlation
  - HEP chunk id 0x11(17 decimal)
  - plain text
  - only one possible
- **external** correlation
  - HEP chunk id **101** decimal (not standard)
  - JSON payload
  - linked proto identified by JSON key

```
{
    "net": "123aaabbbccc",
    "sip": "abdef12345"
}
```

SIP CONTEXT TRACING

- events because of SIP
- sip context
    - message
    - transaction
    - dialog
- correlation
    - **internal**
    - **external**
- controlled via **sip_trace** function

sip_trace(“trace_id”, “scope”, “type”)

SIP CONTEXT TRACING - SIP

- sip_trace(“hep_id”, “scope”, **“sip”**)
- correlation
  - **internal** - SIP callId

```
10f3e104-9158-458a-a341-ee4e281a74ee
```

  - **external** - **net** messages via unique connection id

```
{
    “net” : “1159999397753232466”
}
```

SIP CONTEXT TRACING - XLOG(1)

- sip_trace("hep_id", "scope", "**xlog**")
- correlation
  - **internal** - SIP CallId

```
5da03998-3819-46d1-84c2-aafaf92266ab
```

  - **external** - to **SIP** via the **callId**

```
{
      "sip":  "5da03998-3819-46d1-84c2-aafaf92266ab"
}
```

SIP CONTEXT TRACING - XLOG(2)

- HEP payload information
  - log level **Event**
  - xlog **text**

```
{
      "Event":         "INFO",
      "text": "SCRIPT:AUTH:DBG: authorize ret code is 1"
}
```

SIP CONTEXT TRACING - REST(1)

- sip_trace(“hep_id”, “scope”, “**rest**”)
- correlation
  - **internal** - request to reply(unique internal ID)

```
RESTCORR7y0AAJSUvligJ8dwAAAAAHWX+zk=
```

  - **external** - to **SIP** via **callId**

```
{
        "sip":  "10f3e104-9158-458a-a341-ee4e281a74ee"
}
```

SIP CONTEXT TRACING - REST(2)

- HEP payload information
  - request
    - **HTTP** first line of request
    - **payload**(optional) if the request has a payload

```
{ "first_line":   "GET /lrn/18329008433 HTTP/1.1" }
```

  - reply
    - **HTTP** first line of reply
    - **payload** of the reply

```
{ "first_line":   "HTTP/1.1 200 OK",
  "payload":      "{"rn":"2819549999","data_points":2..." }
```

SIP CONTEXT TRACING - CONTROL VIA MI

- **sip_trace** MI function
  - state of trace ids
    - **on** tracing active
    - **off** tracing disabled
  - control global tracing
    - enable/disable all trace ids
  - control tracing per trace id
    - enable/disable tracing for one trace_id

```
sip_trace [trace_id]* [state]*
```

*** - optional**

NETWORK TRACING

- events determining SIP events
- supported protocols
  - TCP
  - TLS
  - WS
  - WSS
- correlation
  - **internal** - unique OpenSIPS id for each connection

NETWORK TRACING(2)

- enable via **trace_on** and **trace_destination**

```
modparam(“proto_X”, “trace_on”, 1)
modparam(“proto_X”, “trace_destination”, “hep_dest”) #proto_hep defined
```

- control at runtime via MI
  - **X_trace_on** MI command where **X** is the proto
    - if no parameter show state
    - **on/off** parameter to enable/disable

tcp_trace_on [state]*

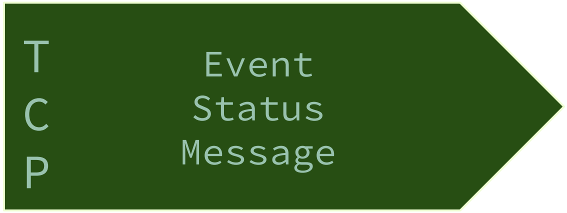
NETWORK TRACING(3)

- control traced connections
  - **trace_filter_route** transport modules parameter
  - filtering based on
    - local interface of the connection **$Ri $Rp**
    - remote interface of the connection **$si $sp**
  - exiting from the route with **drop** will cause packet not traced

```
modparam("proto_X", "trace_filter_route", "net_filter")
route[net_filter] {
    if ( check_source_address("10") )
        exit; #trace this connection
    drop; #don't trace this connection
    ....
}
```

NETWORK TRACING - TCP

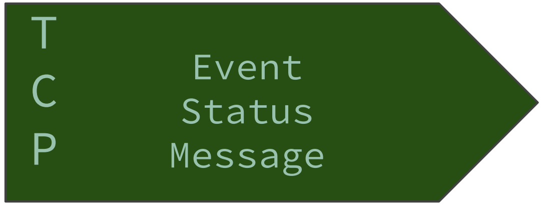
- initial event – traced information



```
{"Status": "SUCCESS", “Event”:”ACCEPTED”, "Message": "Connection accepted..."}
```

NETWORK TRACING - TCP(2)

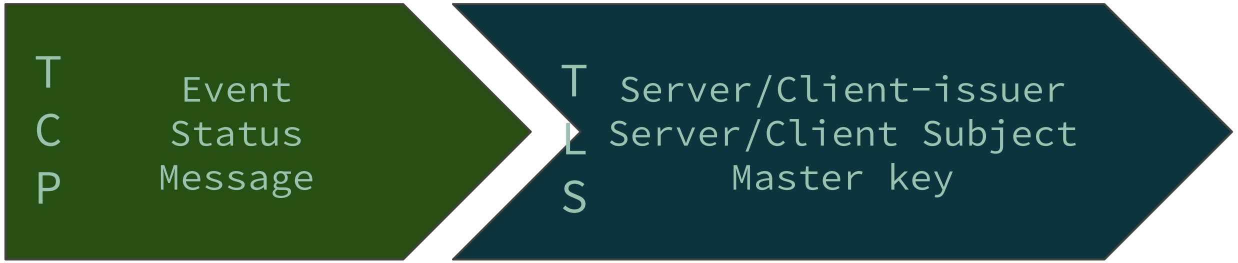
- **termination** event(connection closed) - same for all protos



```
{"Status": "SUCCESS", “Event”:”Closed”, "Message": "Timeout on no traffic"}
```

NETWORK TRACING - TLS

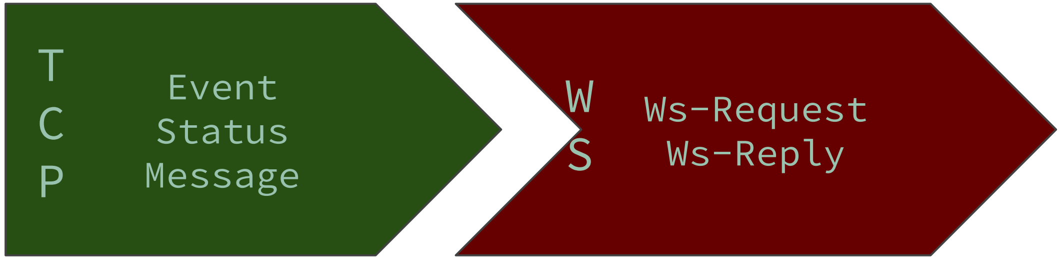
- initial event - traced information



```
{ "server-subject":"/CN=opensips.org/ST=RO...",
"server-issuer": "/CN=opensips.org/ST=RO...",
"master-key": "dc1d6f8a...",
"Status": "SUCCESS", "Message": "Connection accepted..." }
```

NETWORK TRACING - WS

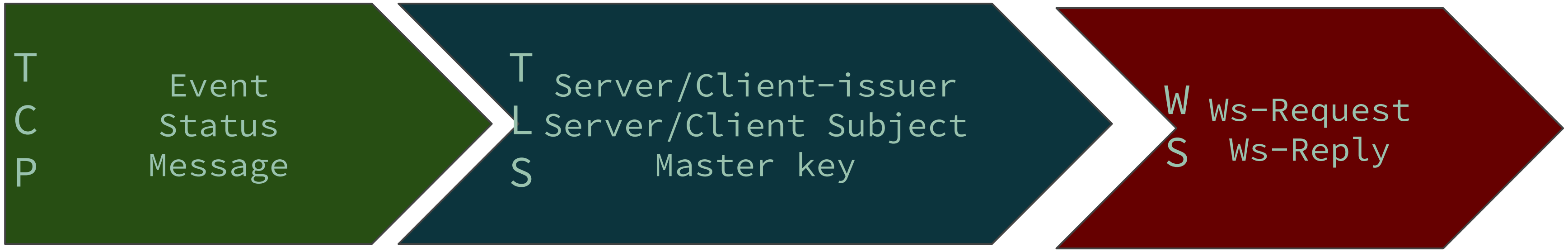
- initial event - traced information



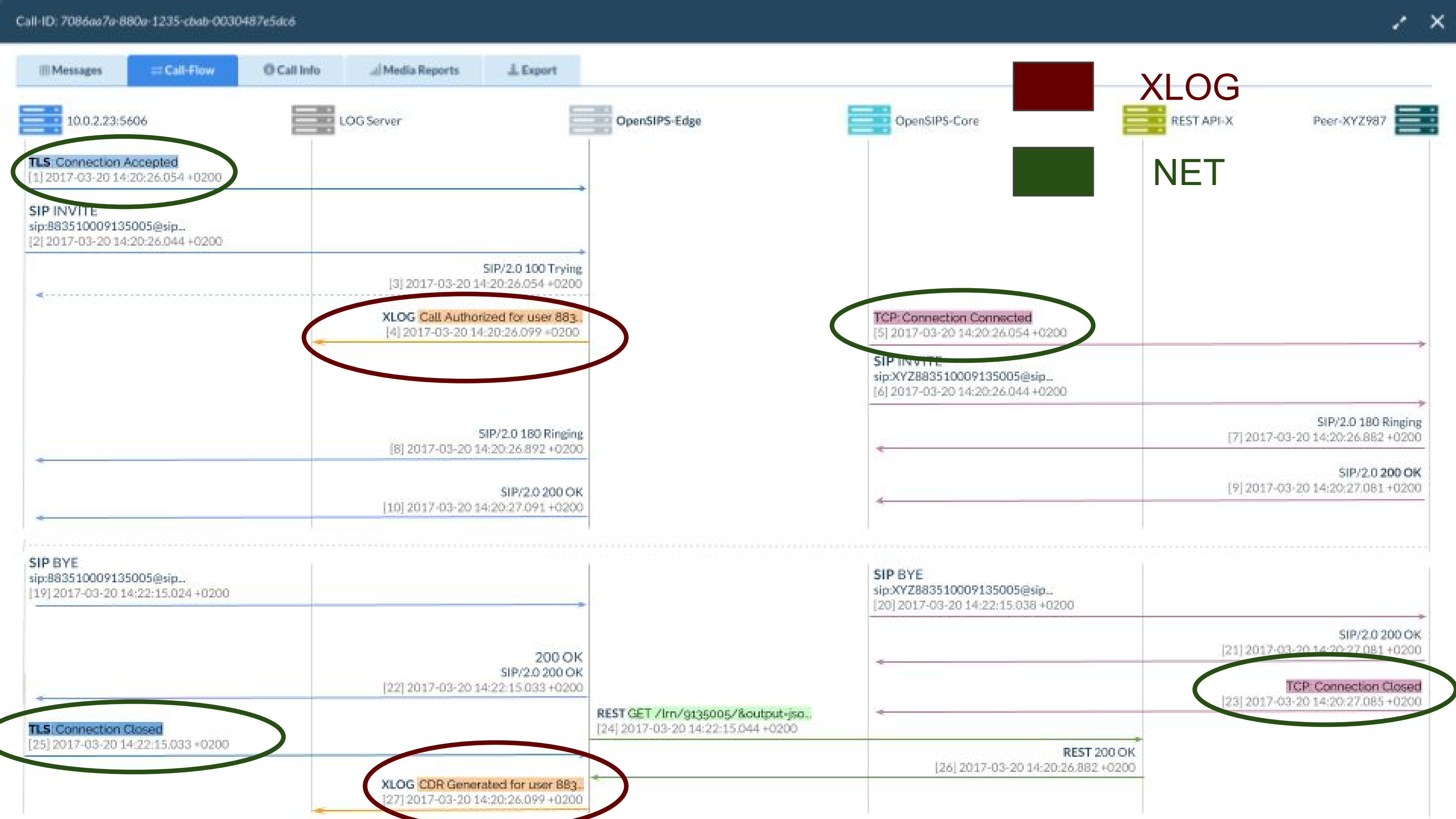
```
{ "Status": "SUCCESS", "Message": "Connection accepted...", "Ws-Request": "GET
/ HTTP/1.1...", "Ws-Reply": "HTTP/1.1 101 Switching Protocols..." }
```

NETWORK TRACING - WSS

- initial event - traced information



```
{ "server-subject":"/CN=opensips.org/ST=RO...", "server-issuer":
"/CN=opensips.org/ST=RO...", "master-key": "dc1d6f8a...",
"Status": "SUCCESS", "Message": "Connection accepted...",
"Ws-Request": "GET / HTTP/1.1...", "Ws-Reply": "HTTP/1.1 101 Switching
Protocols..." }
```



🟥 XLOG

🟩 NET

TLS: Connection Accepted
[1] 2017-03-20 14:20:26.054 +0200

SIP INVITE
sip:883510009135005@sip...
[2] 2017-03-20 14:20:26.044 +0200

SIP/2.0 100 Trying
[3] 2017-03-20 14:20:26.054 +0200

XLOG Call Authorized for user 883...
[4] 2017-03-20 14:20:26.099 +0200

TCP: Connection Connected
[5] 2017-03-20 14:20:26.054 +0200

SIP INVITE
sip:XYZ883510009135005@sip...
[6] 2017-03-20 14:20:26.044 +0200

SIP/2.0 180 Ringing
[7] 2017-03-20 14:20:26.882 +0200

SIP/2.0 180 Ringing
[8] 2017-03-20 14:20:26.892 +0200

SIP/2.0 200 OK
[9] 2017-03-20 14:20:27.081 +0200

SIP/2.0 200 OK
[10] 2017-03-20 14:20:27.091 +0200

SIP BYE
sip:883510009135005@sip...
[19] 2017-03-20 14:22:15.024 +0200

SIP BYE
sip:XYZ883510009135005@sip...
[20] 2017-03-20 14:22:15.038 +0200

SIP/2.0 200 OK
[21] 2017-03-20 14:20:27.081 +0200

200 OK
SIP/2.0 200 OK
[22] 2017-03-20 14:22:15.033 +0200

TCP: Connection Closed
[23] 2017-03-20 14:20:27.085 +0200

REST GET /lrn/9135005/&output-jso...
[24] 2017-03-20 14:22:15.044 +0200

TLS: Connection Closed
[25] 2017-03-20 14:22:15.033 +0200

REST 200 OK
[26] 2017-03-20 14:20:26.882 +0200

XLOG CDR Generated for user 883...
[27] 2017-03-20 14:20:26.099 +0200

MI TRACING

- no connection to SIP

- support in all MI modules (**mi_json**, **mi_xmlrpc**, **mi_fifo**, **mi_http**, **mi_datagram**)

MI TRACING(2)

- set **trace_destination** to enable

modparam(“mi_json”,”trace_destination”, “hep_id”) ##from **proto_hep**

- decide traced mi commands via blacklists/whitlists

```
modparam(“mi_json”,”trace_bwlist”, “w: ps”)
modparam(“mi_json”, “trace_bwlist”, “b: get_statistics”)
```

MI TRACING - HEP PACKET

- HEP payload information
    - request
        - MI **command**
        - **backend**(module) that generated the command
        - MI command **parameters**

```
{  "command":      "get_statistics",  "backend":      "json",  "parameters":
"rcv_requests,..." }
```

MI TRACING - HEP PACKET(2)

- HEP payload information
    - reply
        - **code** and **reason** of the reply
        - backend **reply for the command**

```
{"code": "404", "reason":"Statistics Not Found", "reply":"{"error..."}}"
```

Conclusions

Summing up

- version 2.2 opened new possibilities
- capture everything
  - no more **SIP-centricity**
- Extended tracing
  - logs
  - rest
  - network
  - mi

Future Work

- TCP statistics
  - **getsockopt** - TCP_INFO
- B2B sessions
  - correlate dialogs with B2B **external** correlation
- trace more data
  - accounting
  - sql queries
  - developers check **trace_api.h**

- Ionuţ Ioniţă
    - OpenSIPS Project: www.opensips.org
    - Email: ionutionita@opensips.org