Analysis of attacks / vulnerabilities SS7 / Sigtran using Wireshark (and / or tshark) and Snort

Madrid, March 2018.

By: Alejandro Corletti Estrada (acorletti@darFe.es - acorletti@hotmail.com)

INDEX

1. Purpose of this document.
2. Presentation.
3. Introduction to SS7 / Sigtran.
   - 3.1. SS7 signaling.
   - 3.2. Sigtran
4. Presentation of the different types of attacks.
   - 4.1. Analysis of IR 82 GSM (Security SS7 implementation on SS7 network.
   - 4.2. Classification of different types of attacks.
   - 4.3. Detail analysis.
5. Traffic capture analysis: patterns to look for, employment of Wireshark.
6. Traffic capture analysis using Snort.
   - 6.1. The configuration file.
   - 6.2. Outputs.
   - 6.3. The SS7.rules.
7. Other additional tools.
   - 7.1. MATE (Meta Analysis and Tracing Engine) for Wireshark.
   - 7.2. Tshark.
   - 7.3. Unify frames (mergecap).
   - 7.4. Capture information (capinfos).
8. Conclusions.

REFERENCES

1. Purpose of this document.

To present an eminently technical work methodology that allows: detecting, identifying and evaluating attacks on **SS7 / Sigtran** networks through "traffic analysis" based on the "**Wireshark"** (and / or tshark) and "**Snort**" tools.

2. Presentation.

For some years now (we could say that around 2010), we have begun to hear that this signaling system (*the true **heart** of the entire world voice network and certain types of data*) presents **serious security problems**. The exploitation of them opens a range to all types of attacks, at present they are already running in several telephone operators, stealing money from bank accounts, intercepting phone calls, locating the position of mobile phones, performing different types of fraud in voice and navigation, executing service denials, etc.

In these lines, we will not develop the SS7 (Signaling System 7), nor Sigtran (Transport Signaling), we will only make a very brief presentation of them to be able to understand the problem.

It is worth mentioning that the "**traffic analysis**" is the ONLY methodology we have to understand and evaluate this type of anomalies in our signaling flows. We dare to make this statement, based on a series of documents and standards that we will present in this document.

We are facing a **serious problem worldwide** and that will necessarily be extended for at least the next ten years, because this signaling will only be replaced when all the world's trunks use SIP and / or DIAMETER, which are voice over IP protocols. It to say, when the end-to-end connectivity for all voice services is packetized by TCP / IP stack.

3. Introduction to SS7 / Sigtran.

3.1. Signaling.

The basic purpose of signaling is to establish a language for the exchange of control information that allows two telephone lines located in any part of the telephone network to communicate with each other.

Specifically covers all aspects related to:

- **Establishment.**
- **Maintenance**
- **Closing**

Of a communication (*nowadays, be it voice or data*).

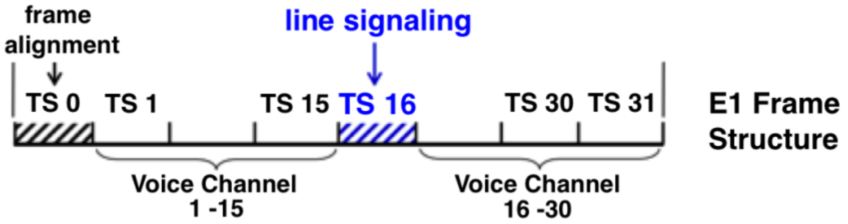
SS7 enters into production in the 1980s as a "**closed**" network of telephone operators, defined as Signaling by common channel. Establishes the procedures and protocols for information exchange between the resident entities of a signaling network {*fixed telephony (PSTN) - packet network (PDTN) - mobile telephony (PLMN)*} for supervision, control, access, management and routing of services of voice or data Transmitted in the digital channels of the **PCM** links (*Pulse Code Modulation*).

If we want to go into a little more detail, there are two types of Signage:

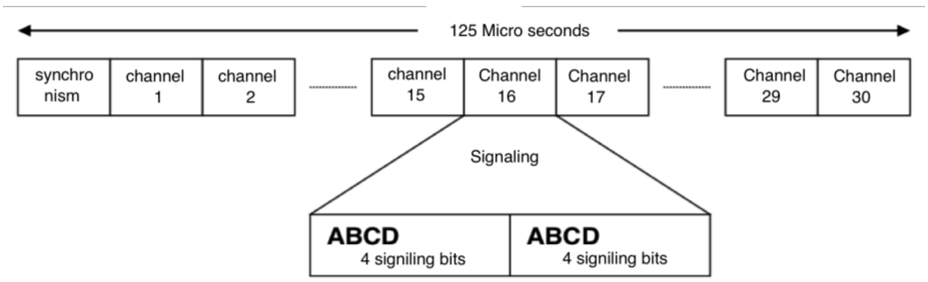
- **Access** (or subscriber)
  - DSS (digital subscriber signaling) → data (ISDN D channel)
  - PSTN (analog subscriber) by independent frequencies
- **Trunk**
  - CAS (Signaling by associated channel)
  - **CCS** (Signaling by common channel) ← **This is SS7**

The basis of this system, as we mentioned PCM, was based on the first steps on digitalization of the "analog voice", where in our part of the planet was adopted as "acceptable bandwidth" for a vocal category the value of 4000 Hertz and according to the sampling theorem, their bandwidth was taken twice in "samples", that is, 8000 samples/second, which were finally "coded" with 8 bits, obtaining what was called "basic digitized voice channel". "= 64,000 bits / sec = **64 Kbps**. (In another countries it use 7 bit, then the result is = 56 Kbps)

This basic channel, was integrated into what they call "digital hierarchies", of the cells the first of them (in its Plesiochronous version or PDH) was the famous **E1** frame(*I reiterate that for our part of the Western world, there are others, for example: T1=56 Kbps*) This frame, is the sum of 32 channels of 64 Kbps grouped in "Time Slots". Of these 32 slot machines, 30 are channels where low "voice", the first is for "synchronism", and in the case of SS7 only the" **Time Slot 16** "is used in this channel travels **all** the SS7 signaling through the use of two groups of 4 bits (called ABCD) this "bits" only signaling two voice channels per frame, of the 30 of each E1.. Since, once the frame E1 is "injected" into the trunk network, it has a total duration of 125 **µ**s (Micro seconds), every 8 frames passes 1 millisecond, therefore, every 2 milliseconds pass 16 frames with the eyes re-signaling the two channels of the first frame E1.The following is an example of this format:



Basic format of E1 frame

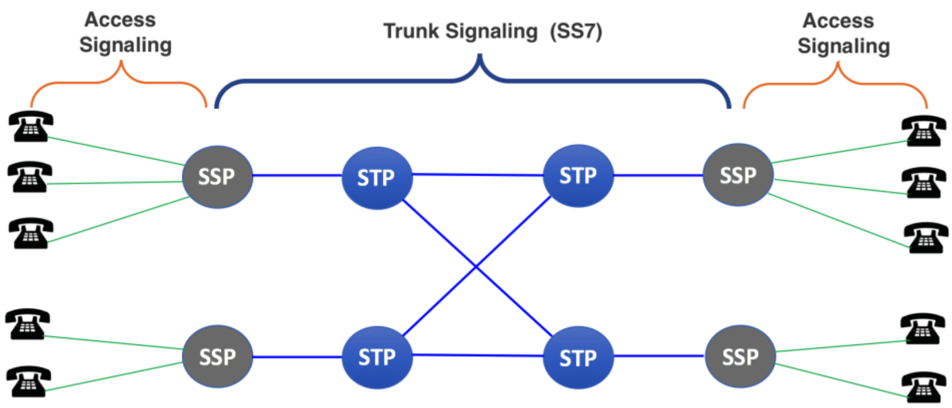


Basic format of 16 Time Slot

The SS7 network is based on a 7-level protocol stack that responds to the ISO/OSI model (**not accessible** from the TCP/IP stack). This layer model allows to move information through three types of nodes, called:

SP (Signaling Point).

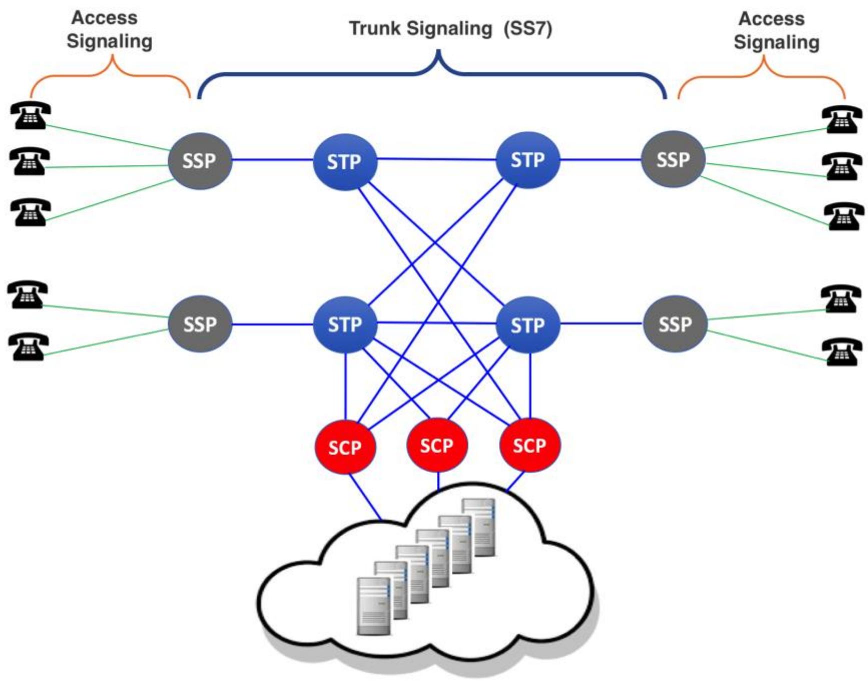
- **SSP** (Signaling Switching Point).
- **STP** (Signaling Transfer Point) Router or GW, does not generate messages, only routes, makes transfer measurements.
- **SCP** (Signaling Control Point) provides access to Applications (eg DDBB, etc).



SS7 Network

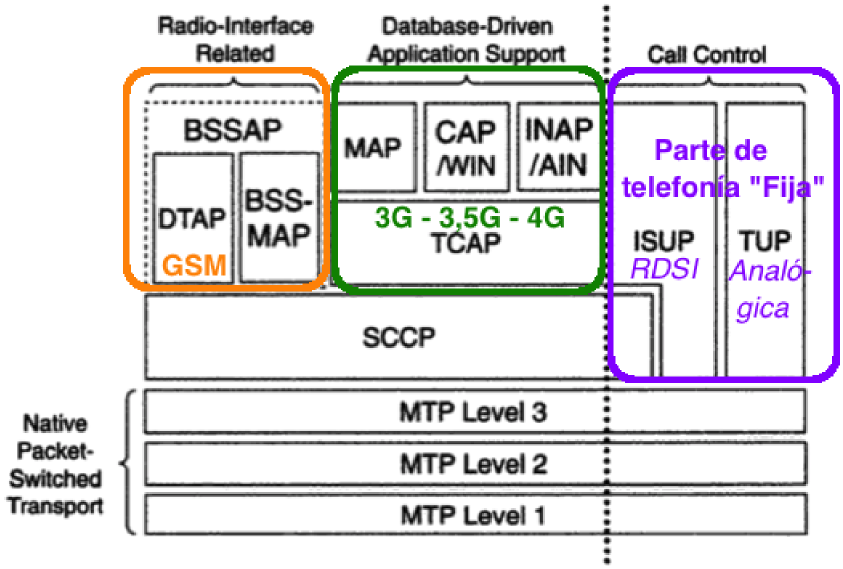
SS7 cone mentioned, born in the 80`s within a "closed" frame, only accessible by telephone operators .... The problem is born with the "**Intelligence**" that we started to put to our networks (**NGIN**: Next Generation Intelligent Network).

Specifically, this "intelligence" in a summarized form, perhaps begins with the first **ISDN** networks (*Integrated Services Digital Network*) and a protocol called **ISUP** (*which will be presented later*) is implanted in the SS7 stack, when the first mobile networks begin, it incorporates a new layer in the form of **BSSAP** (*which we will also present below*) and finally the **MAP** *(Mobile Application Part)* protocol for all aspects of profiles, messaging, dual authentication systems, mobility, roaming, unstructured services, etc. Below is an image where these new aspects appear that are ultimately offered through Servers or software applications (something new in the SS7 hierarchy).

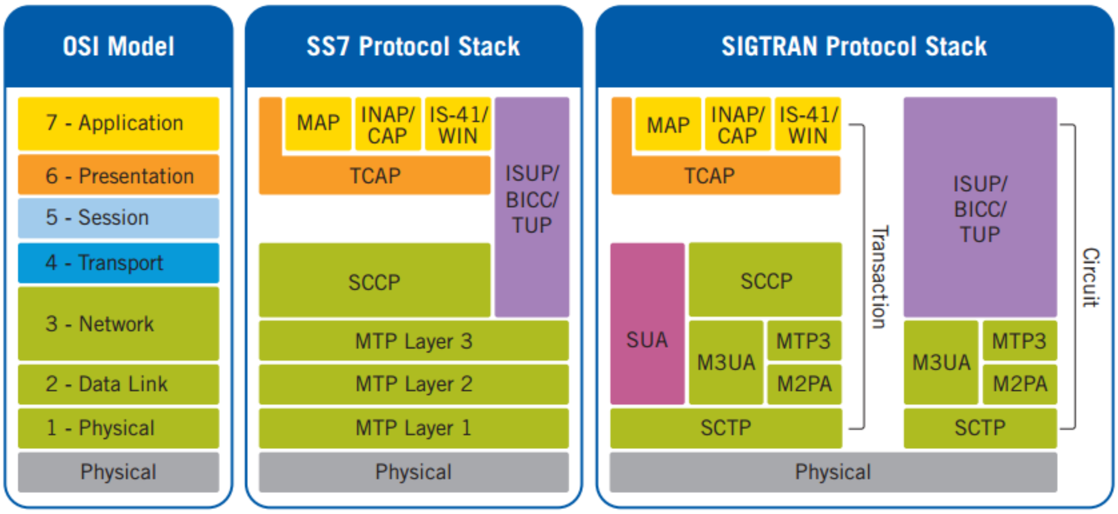


SS7 & NGIN network

As these new protocols are incorporated, the SS7 infrastructure under the seven-layer ISO/OSI model begins to become inoperable, and in this way "Sigtran" is born, which incorporates the TCP/IP stack below this SS7 family ... .. and we enter the IP world (*with the good ... and also the bad ...*) **Here our problems and vulnerabilities potentially accessible from anywhere in the world through IP routing begin**. Here we present a couple of images of the layers models.



SS7 Stack



OSI stack - SS7 stack – Sigtran stack

As we mentioned at the beginning, this is not a text about SS7/Sitgtran, but a brief presentation of both, therefore the only aspects that we wish to highlight are:

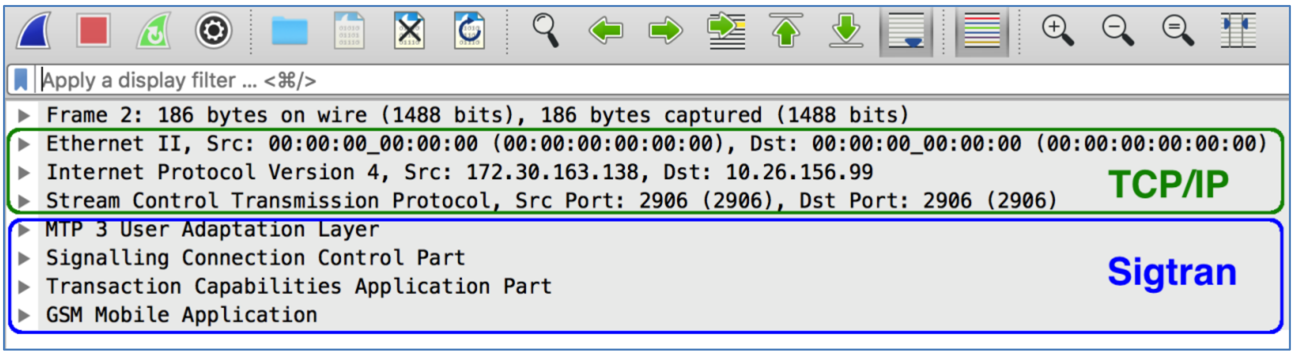
- In the **SS7** stack (*central*) we can see a model that responds to the seven layers of the **OSI** stack (*left*). We reiterate that it has NO communication with the TCP/IP world. The protocols that most interest us in this model are: SCCP, TCAP, MAP, CAP (Camel) and ISUP.
- In the **Sigtran** stack (*right*) we must highlight SCTP which is the protocol that replaces TCP or UDP at the transport level incorporating the advantages of both (*It should be noted that nowadays it is also used as a transport for other TCP stack application protocols/IP, for example http*)

- The "**Phisical**" level of the Sigtran stack is not more than levels 1,2 and 3 of the TCP / IP stack.

Only for descriptive purposes, we present below these protocols.

- MTP Layer 1: (Message Transfer Part nivel 1)
- MTP Layer 2: (Message Transfer Part nivel 2)
- MTP Layer 3: (Message Transfer Part nivel 3)
- SCCP: (Signaling Connection Control Part)
- ISUP: (ISDN User Part) ISDN signaling messages (D channel)
- TUP: (Telephone User Part) telephone signaling messages
- TCAP: (Transaction Capability Application Part)
  - MAP: (Mobile Application Part) Employed by MSC, SGSN y GGSN
  - INAP: (Intelligent Network Application Protocol)
  - AIN: (Advanced Intelligent Network)
  - CAP: (/CAMEL Application Protocol *[Customizable Applications for Mobile Enhanced Logic]*) Roaming.
  - WIN: (Wireless Intelligent Networking)
- BSSAP: (Base Station System Application Protocol) Employ native GSM systems with MSC and BSS, provide two kinds of functions:
  - DTAP: (Direct Transfer Application Part) call management and mobility management.
  - BSS-MAP: Dialogue between MSC-BSS and Handover.
- IS-41 WIN: (ANSI-41) Mobility management in mobile telephony (ANSI/TIA/EIA-41.5-D, Wireless Intelligent Networking (WIN) extensions ANSI/TIA/EIA-751, ANSI/TIA/EIA-764, ANSI/TIA/EIA-771, ANSI/TIA/EIA-826 [Prepaid])

Let's see a first traffic capture on the Sigtran stack so that we begin to understand this system of "packaging".



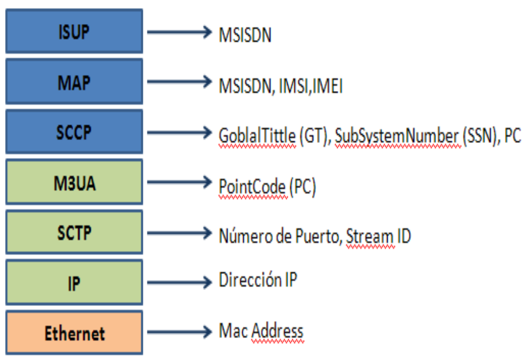
*Traffic capture (in **green** TCP / IP stack protocols, in **blue** Sigtran stack protocols)*

Finally, for our analysis work, we can not ignore at least the basic addressing schemes used by these protocols.

- **MSISDN:** (Mobile Station Integrated Services Digital Network) composed of the country code (Spain is 34) and the subscriber's telephone number (National Destination Code plus Subscriber Number).
- **IMSI**: (International Mobile Subscriber Identity), the unique identifier of each SIM card in the mobile network (formed by the Mobile Country Code, the Mobile Network Code and the Mobile Subscription Identification Number).
- **IMEI**: (International Mobile Equipment Identity) the identifier of each mobile terminal (you can check the mobile by dialing * # 06 # in the dialer-dialer).
- **GoblalTitlle**: It is the SCCP address of each node in the SS7 network, using the same format as the telephone numbers of the subscribers. but in this case they represent nodes of the network, not people.
- **SubSystemNumber** (SSN): indicates to each node of the network with what other type of node will establish link / communication. Each type of node has its own number: 6 HLR (MAP), 7 VLR (MAP), 8 MSC (MAP) ....
- **PointCode**: is the identifier of layer 3 of MTP that is assigned to each node of the network.

All of them are found in the following link of the 3GPP "**TS 23.003: Numbering, addressing and identification**", with a better description and the format of each one.

To clarify a little more the relationship of each of them with its corresponding protocol, we present an association of them by means of an image.



4. Presentation of the different types of attacks.

To start this section, we will do so through a document presented by the GSMA (GSM Association) for treating the issue as a whole, although we must take into account that it only applies to the mobile network. Later in our document we will also address the fixed network.

4.1. Analysis of IR 82 GSM (Security SS7 implementation on SS7 network guidelines - Version 3.0 21 - March 2016).

The attacks can be executed mainly by:

- Handling of **SCCP**
- **MAP** alterations

NOTE: Bear in mind that because it is a document published by the GSMA, it does not deal with ISUP or TUP (fixed network)

This document, identify 55 risk operations, and classify them into 5 categories:

- **Category 1**: Messages that should only happen in the "Home Net"
- **Category 2**: Messages that are **NOT** from the "Home Net"
- **Category 3**: Messages that should normally be received from a subscriber that is in an"External Net"and exclusively from that "External Net"
- **Category 4**: Operations with SMS
- **Category 5**: CAMEL

It document presents a table associating these categories with its possible solutions:

Domain	Cat1	Cat2	Cat3	Cat4	Cat5
Roaming out	X		x	x	
Roaming in	X	X			x
Home	X	X		x	x

Table 1: Category vs. Solution table

Category 1 messages can be filtered by relatively simple techniques at the edge of the network.

This can be done by evaluating the type of message and verifying whether the message has been sent or not from an "Extenal Net".

Category 2 messages can not be filtered at the edge of the network.
An operator must correlate the subscriber's statuses and verify if the subscriber is in an "External Net" or not before it can be blocked.

Category 3 messages should use more sophisticated approaches. These are messages that have a legitimate use in the network and simply can not be filtered. A protection system needs to analyze the flow of messages from the network and be able to look for changes in the behavior of network elements and subscribers. For example, looking at the previous location of the subscribers.

4.2. Classification of different types of attacks.

For the analysis of these attacks, we will draw on the different references that exist on the Internet:

Engel, T[i]

Langlois, P. [ii]

Nohl, K. [iii]

Vauboin, P.-O. [iv]

According to these references the attacker must be:

1) **Connected to the SS7 network in some way.**
2) **Ability to generate arbitrary SS7 messages at will, and**
3) **Capable of imitating a node in the SS7 network providing SS7 capabilities.**

They can be grouped into four categories:

1) Filtered or poorly secured information (information leaks).
2) Protocol Fuzzing (D.o.S, Resource Exhaustion, etc.).
3) Recognition and enumeration of the network (mapping and scan of nodes, ports, etc.).
4) Injection of packages (SendRoutingInfo, ProvideSubscriberLocation, etc).

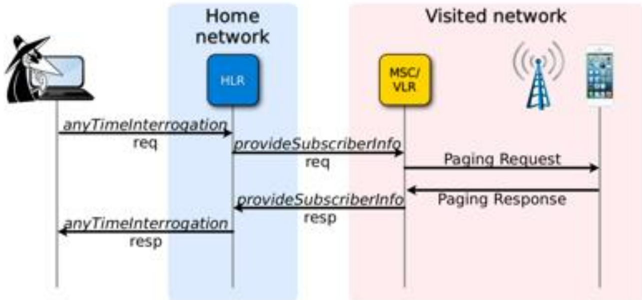
4.3. Detail analysis.

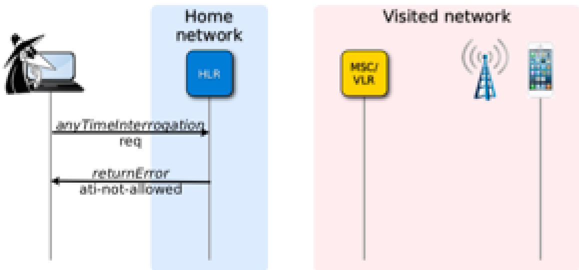
Below we present fifteen different types of attacks that we have classified in this way in order to "segregate" as much as possible the traffic patterns and the origins and destinations of them. This classification does not try to be exhaustive and of course it can be debated and even refuted thinking that it is possible to group some of them or even further break them down.

Our healthy intention to present it in this way, is only the one mentioned: to be able to evaluate different flows and parameters, but from now on we accept any kind of criticism to the aspect, it is just a fucking sight more.

1. Information search on cells-HLR-VLR / MSC

a. The MAP service. **anyTimeInterrogation** (**ATI**) can consult the subscriber's HLR for its Cell-Id and IMEI (phone serial number, can be used to look up phone type) (*Textual in the document: "**31c3-ss7-locate-track-manipulate.pdf**" , page 13 of **Tobias Engel***) from HOME NET to VISITED NET

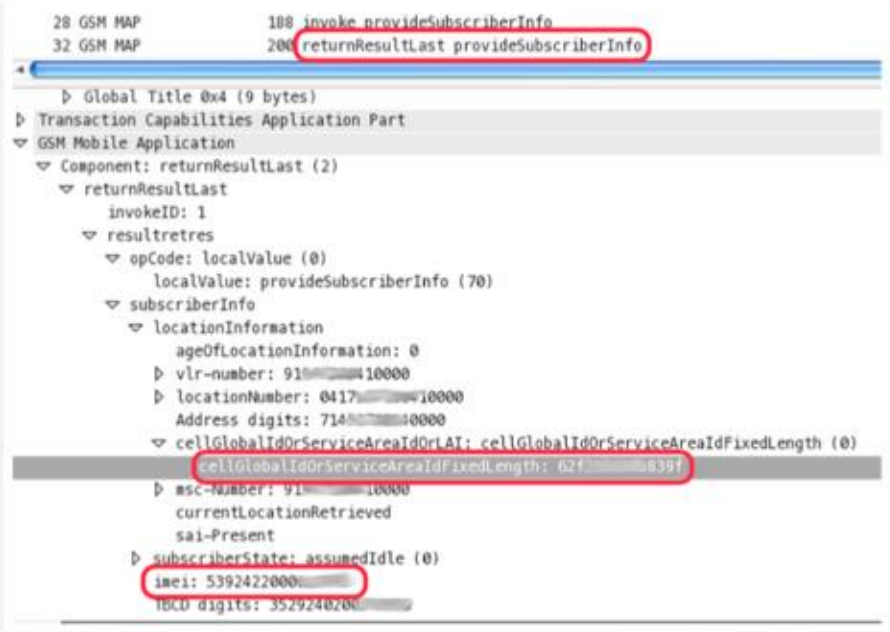




← Some networks currently block it.

b. Instead, query the MSC / VLR directly (that is, direct query to the MSC / VLR instead of the HLR) Within the same HOME NET (*page 16*)

c. Once the IMSI of the subscriber is known, the intruder can consult directly by the "Cell ID" of the same, in this case the MAP parameter is: "**provideSuscriberInfo Request**" and if the MSC / VLR responds, it will do so with "**provideSuscriberInfo Response**" (*see traffic capture on page 18 of the aforementioned document*).



The parameter: **returnResultLast** (2) **anyTimeInterrogation** (**ATI**) should not respond to anyone who interrogates it, if it does, it offers the GT, the VLR-number, the locationNumber and the Address digits (all MAP fields). (*we can see it in the doc: "**31c3-ss7-locate-track-manipulate.pdf**", page 14).*

SOLUTION: Analyze the possibility of implementing blocking ATI to IPs or SCTP or improper TCAP).

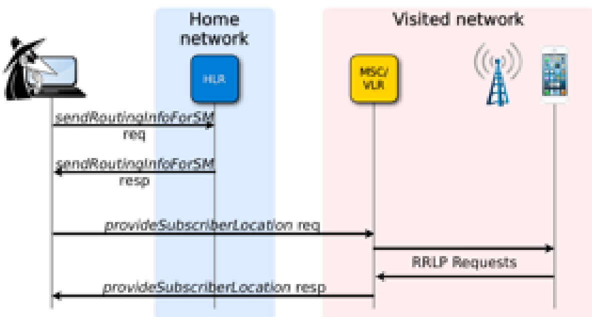
SOLUTION in a German Operator:

- The Operator started filtering all network-internal messages at the network's borders
- This (combined with SMS home routing) basically eliminated the simple form of tracking as seen before
- Attack traffic dropped more than 80%:

2. Location Services (LCS) (use of Emergency Location)

Again on MAP, two steps are carried out:

a. The intruder sends **sendRoutinginfoForSM request** (to the HLR), which responds with **sendRoutinginfoForSM response**



b. second send: **provideSuscriberLocation request** (now to the MSC / VLR, the one that consults the antenna), which responds with

provideSuscriberLocation response (*see it on page 25 of the aforementioned document*).

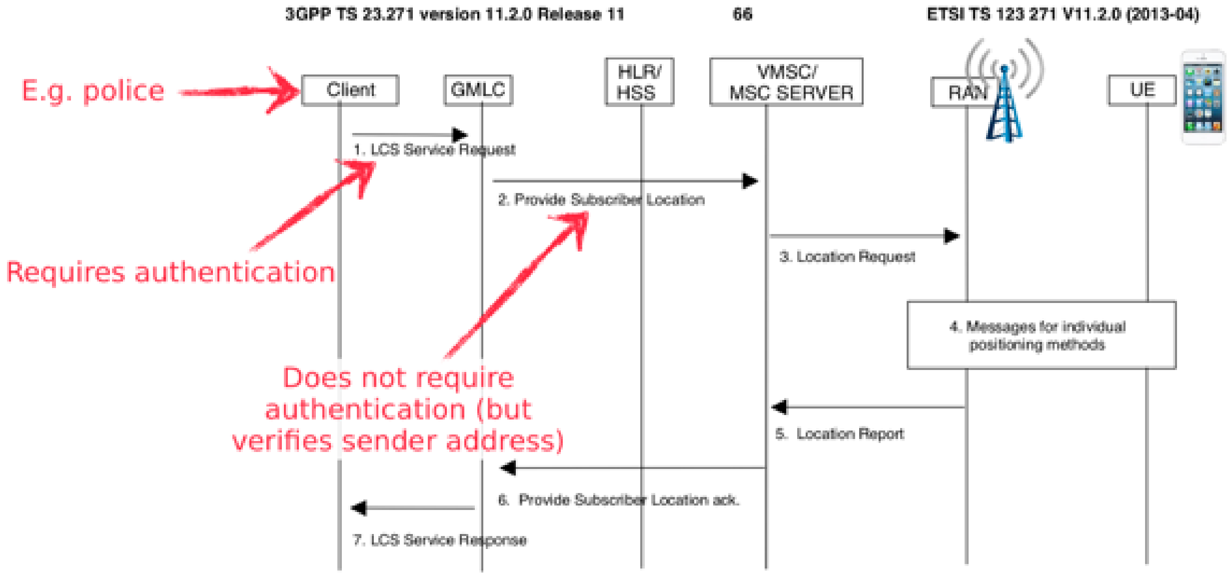
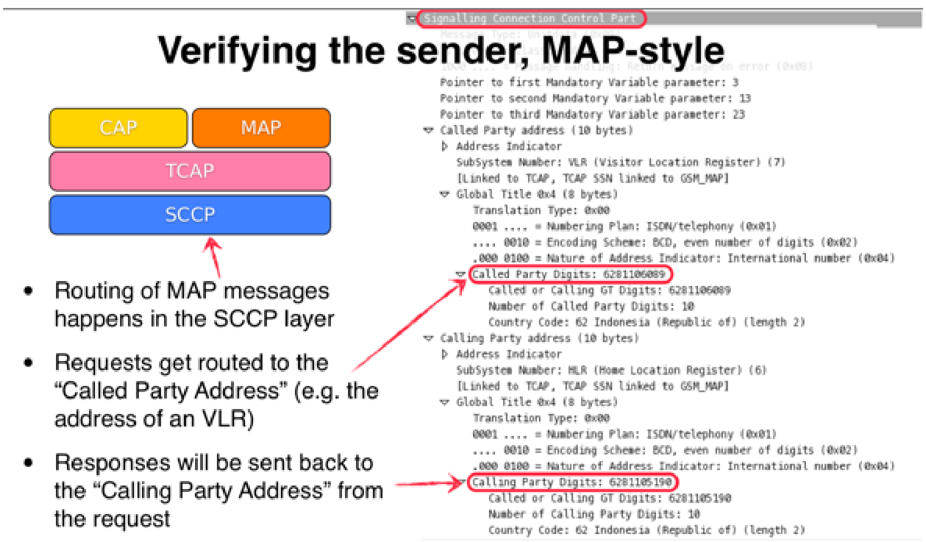


Figure 9.3: Positioning for a Emergency Services MT-LR without HLR Query

Routing of MAP messages occurs in the SCCP layer (this is very important !!! ANALYZE SCCP !!!! in the **Called Party Digits** and **Calling Party Digits** fields)

The requests are directed to the "Address of the called party" (*for example, the address of a VLR*). Answers will be sent back to the "Calling party address" of the request.



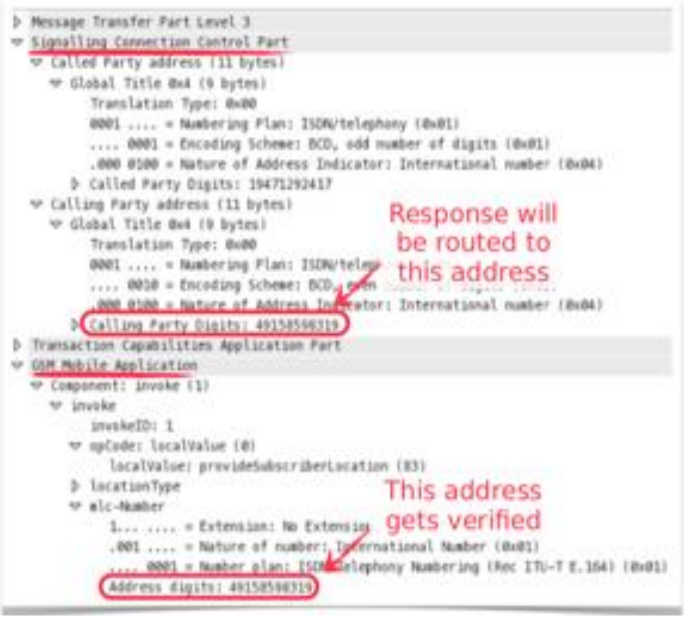
(See capture traffic on page 26 of the aforementioned document)

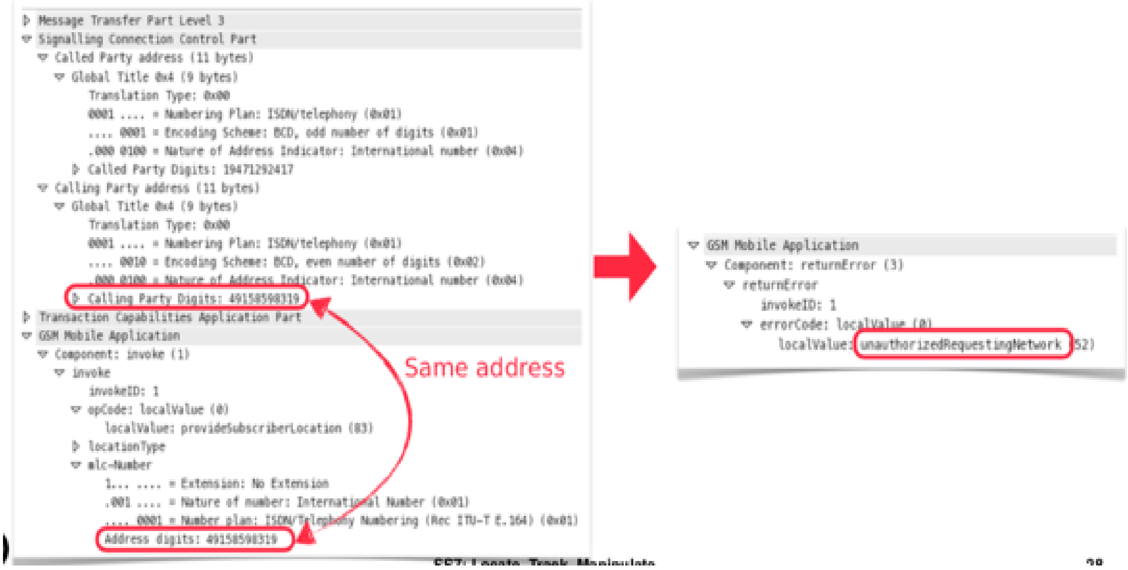
Problem: **SCCP** does not know anything about MAP or what entities should be able to use what MAP services

SOLUTION:

Make the sender **Put another copy** of your "Calling party address" in an additional field in the MAP layer, so that it can be verified (This is very good, since you can verify this address from MAP, verify that it is correct :

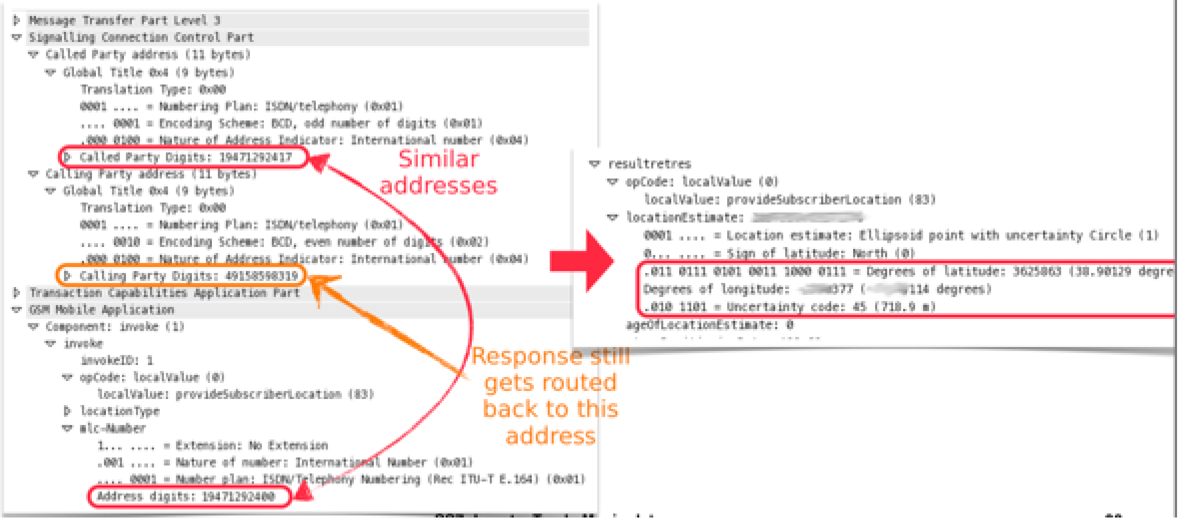
- If it is not, it generates an error
- If it is, go ahead and send the answer with the correct field in SCCP (Called Party Digits and Calling Party Digits)





The routing will continue to occur at the network layer addresses (*See page 27 of the aforementioned document*).

- If we enter an address from the same network that we sent the request to:

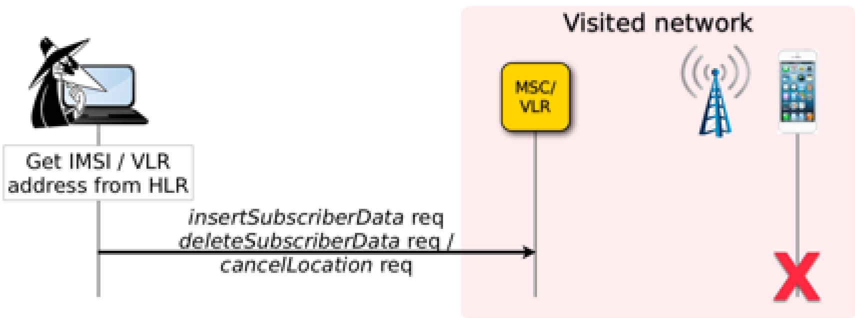


3. Denial of Service

Not only is it possible to read subscriber data, but it can also be modified, since most of the VLR / MSC in the network do not perform plausibility checks.

Once the intruder knows the address of the MSC/VLR, he can send the following parameters via MAP:

- **insertSubscriberData req**
- **deleteSubscriberData req**
- **cancelLocation req**



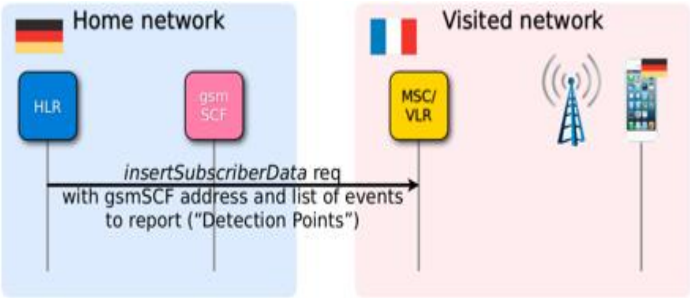
SOLUTION:
Control every aspect of what a subscriber is allowed to do:

- enable or disable calls/SMS or incoming and/or outgoing data
- eliminate the subscriber of the VLR as a whole.

4. CAMEL "Customised Applications for Mobile networks Enhanced Logic"

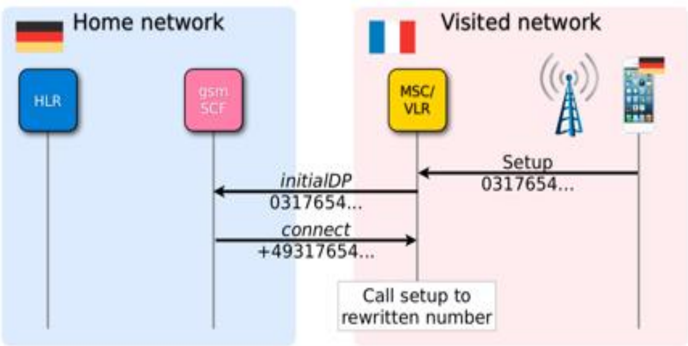
Specified in **3GPP TS 23.078**, it is like an overlay on the usual MAP logic. Defines a set of events for which the VLR must contact the CAMEL entity in the subscriber's home network (**gsmSCF** = "GSM Service Control Function")

The gsmSCF then decides if the desired action can continue without being modified or modified or will be aborted. Example:



The German subscriber is roaming in France.

German HLR tells French VLR "notify my gsmSCF at the address +4917, when the subscriber wants to make a call".

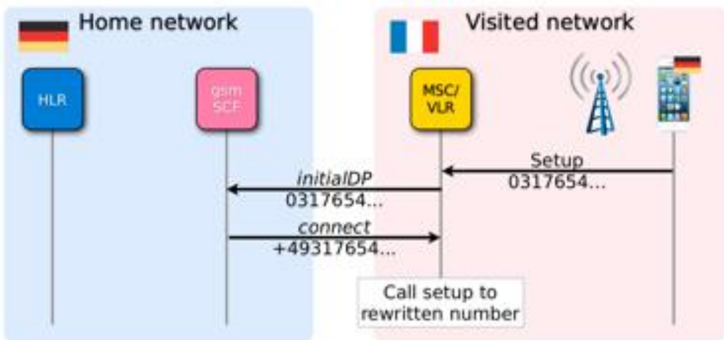


The subscriber wants to make a phone call, but he dialed the number in German national format (0317654 ...)
MSC asks gsmSCF in the home network what to do with the call, gsmSCF rewrites the number to international format (+49317654 ..) and tells MSC to continue with the new number.

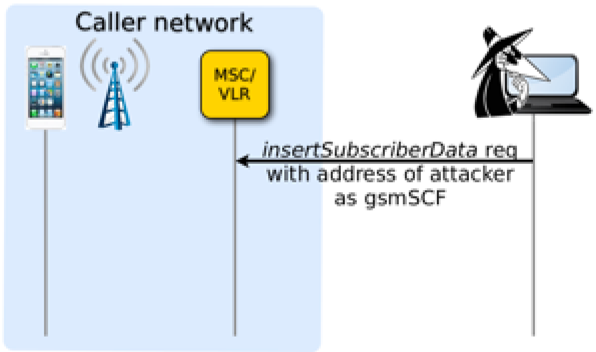
Intercepting calls with CAMEL

A basic function of CAMEL is when a subscriber of network A (Germany), visits network B (Belgium), let's analyze it:

a. the subscriber being in B, calls a number in network B (but without putting the international code in front of him, he is calling his "own network".



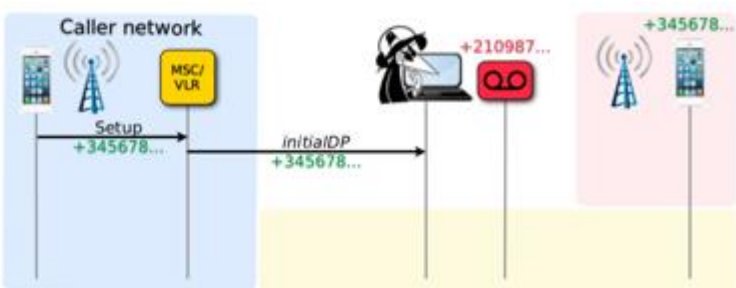
b. the MSC / VLR (of the network in which it is located, in this case network B) consults the gsmSCF (network A) and rewrites it in its international format (in this case it would add +49) and tells the MSC to continue with the call.

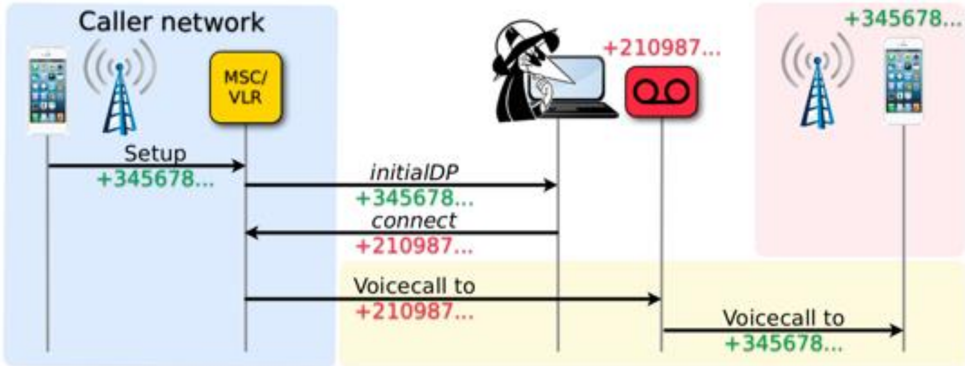


c. For the interception of this call, first, the intruder "overwrites" the address of the gsmSCF with its "false gsmSCF". This is done with the parameter: insertSubscriberData req (from MAP).

d. the MSC (in this case again from network A) will respond to the "false gsmSCF", the parameter is "initialDP".

e. The intruder over writes the number now, for example +210987 ..., registering it as its own proxy (e.g. an Asterisk PBX).





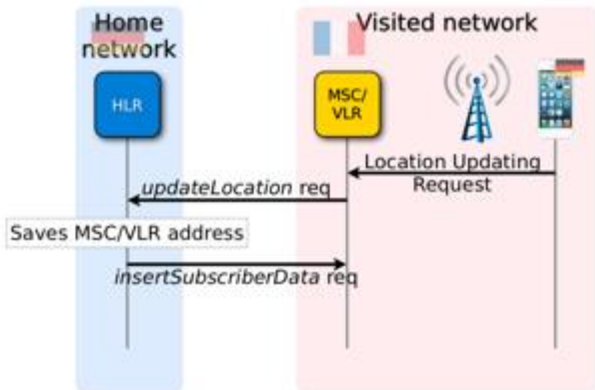
f. MSC will configure the call to +210987 ..., leaving a MitM to the original phone (being able to record the whole conversation).

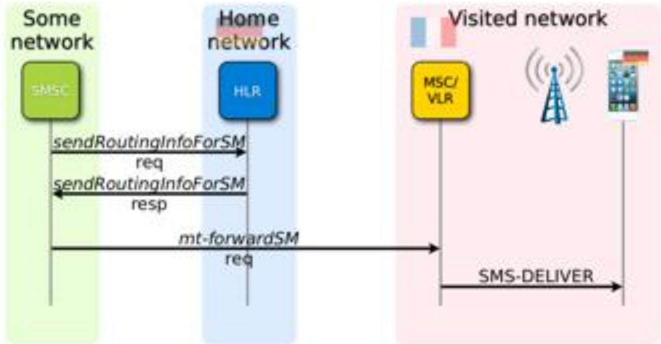
(All this is shown on pages 31 to 37 of the aforementioned document)

5. HLR Location Update

When a subscriber travels to another region or country, the VLR / MSC sends a MAP update request to the subscriber's HLR (the parameter is: **updateLocation req**).

The HLR sends a copy of the subscriber's data to the VLR / MSC and stores the address of the VLR / MSC (the parameter is: **insertSubscirberData req**).



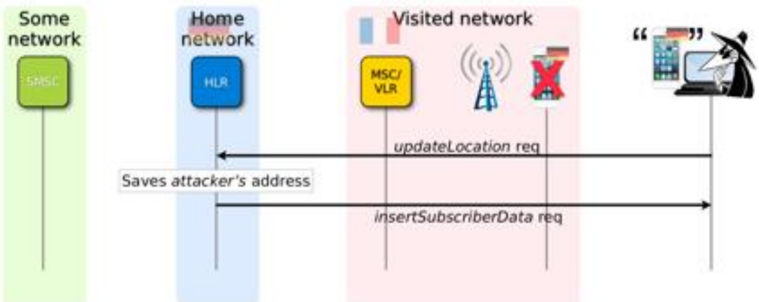


Now, when someone wants to call or send a text message to the subscriber from any network, the routing information is requested from the HLR (from the originating network, for example the SMSC sends the HLR **sendRoutingInfoForSM req** and the HLR responds with: **sendRoutingInfoForSM resp**) and gives you the address of the VLR / MSC.
Finally, if the call is sent from that network or the SMS will send it directly to the MSC / VLR that has just been indicated by the HLR through the parameter: **mt-forwardSM req**.

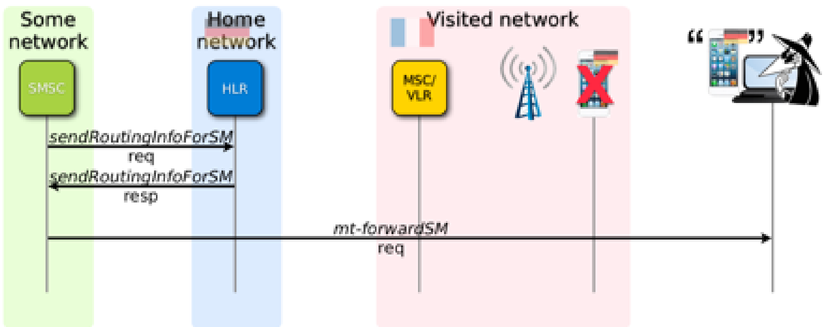
<u>HLR: Stealing Subscribers</u>

The **updateLocation** procedure is also not authenticated.

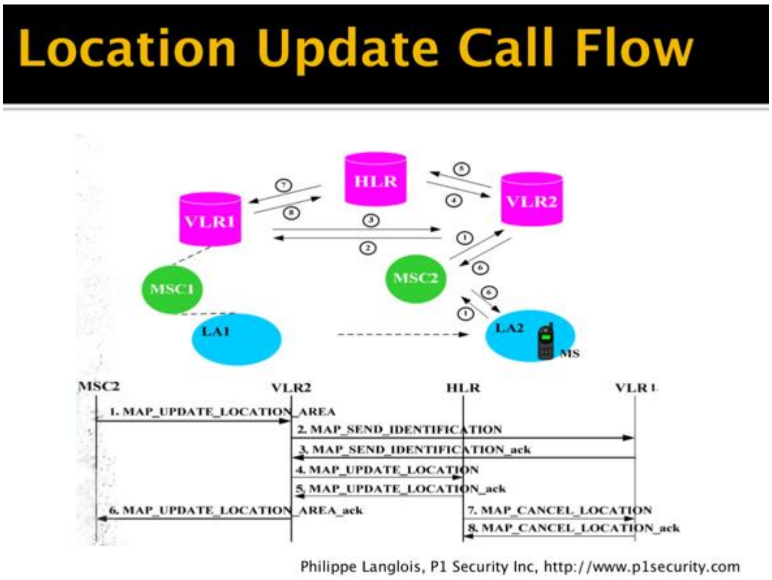
An attacker can simply simulate that a subscriber is in his "network" by sending the **updateLocation** with his Global Title to the HLR of the subscriber (The parameters are: **updateLocation req**, to which the HLR will respond with **insertSubscriberData req** and remember that saving this address in the HLR).



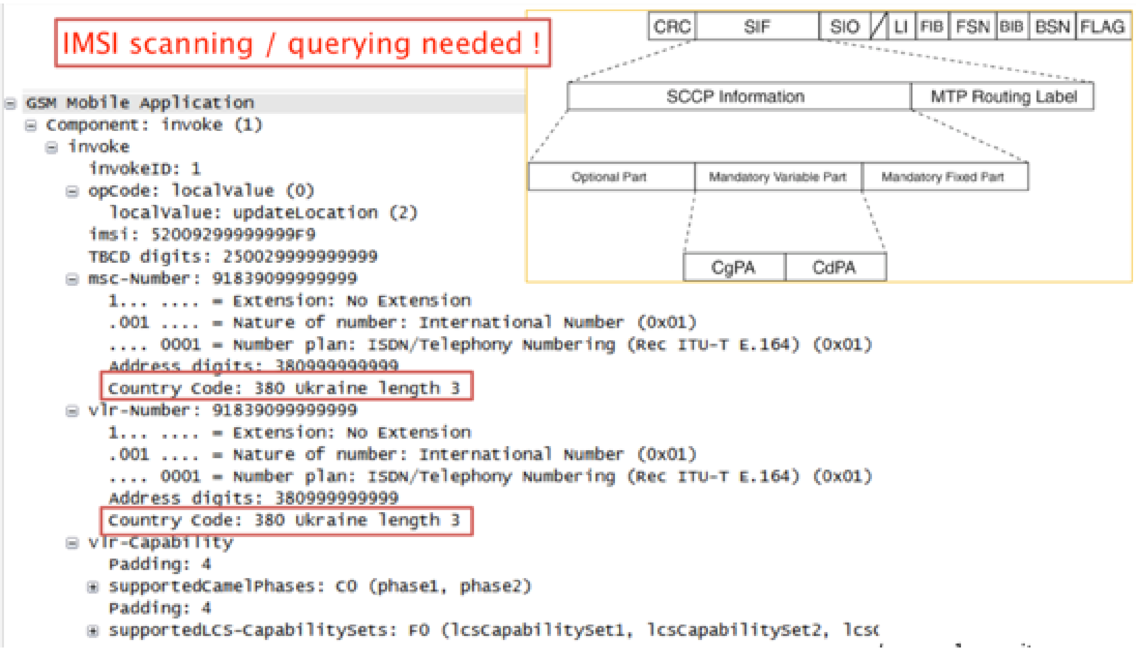
Now, the calls and SMS for that subscriber are routed to the attacker. Example: Subscriber bank sends text with **mTAN**. The attacker intercepts the message and transfers money to his own account.

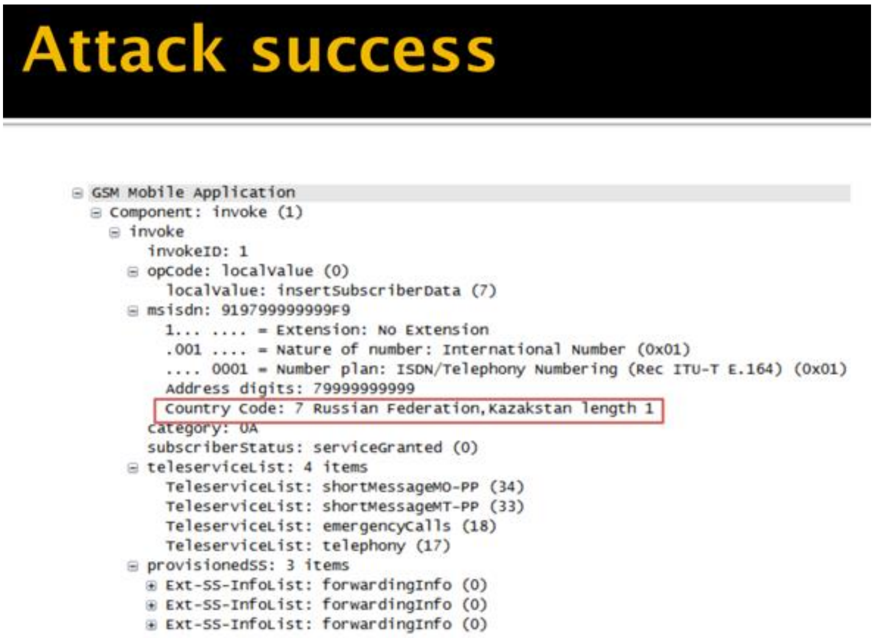


(All this appears on pages 38 to 42 of the aforementioned document)



We can also analyze it from the aforementioned document by **Philip Langlois**:

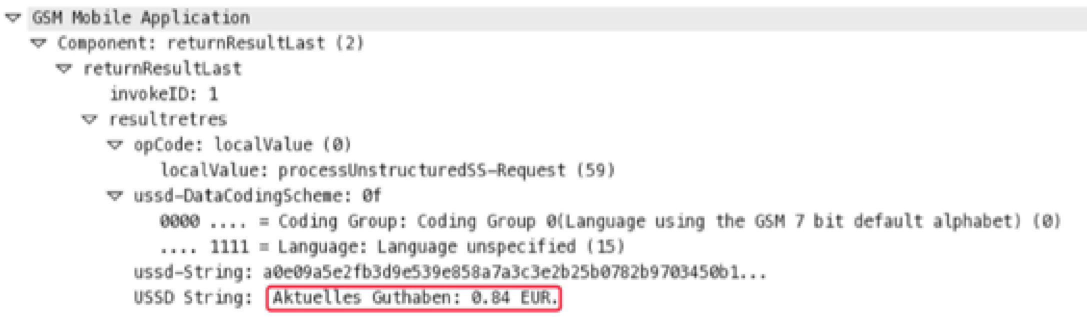




6. HLR Supplemantary Services.

The **USSD** (Unstructured Supplementary Service Data) codes can be executed by other subscribers. Some operators offer transfer or prepayments through credit cards.
Call forwarding can be configured / deleted. An attacker could forward calls from a subscriber to a premium rate number controlled by him and then call the subscriber's number, billing all premium rate calls to the subscriber

Active SIM switch in case of Multi-SIM. Requests can be sent even without a previous **updateLocation**, because the HLR does not verify if the subscriber is in the network that is sending the request.



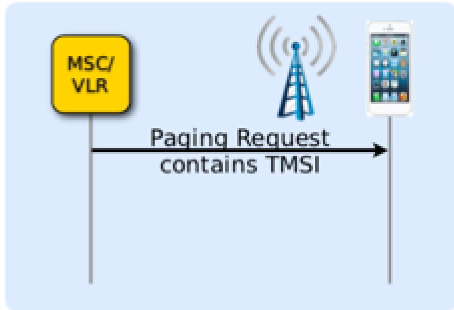
All these parameters are also part of MAP and the field is USSD String)

(*All this appears on pages 43 and 44 of the aforementioned Tobias Engel document*)

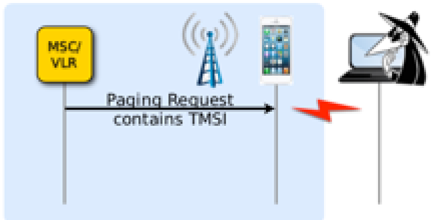
7. Hybrid Attacks: TMSI De-anonymization

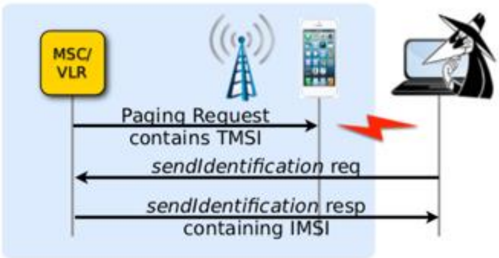
An attacker can find out the telephone numbers of the subscribers around him:

- The paging of subscribers (for example, to notify them of an incoming call) happens unencrypted.
- **TMSI** (Temporary Mobile Subscriber Identifier) is normally used for paging so that the subscriber's real identity (IMSI) does not have to be sent by the unencrypted air interface.
  (The parameter sent from the MSC / VLR to the ME is: **PagingRequest** and contains the TMSI).



The attacker captures TMSI in the air (For example with OsmocomBB)

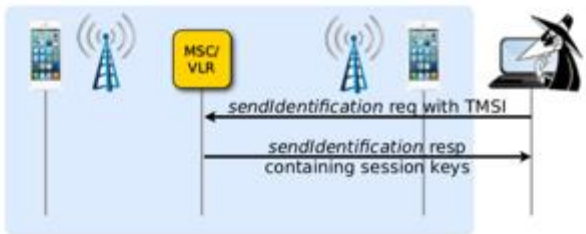


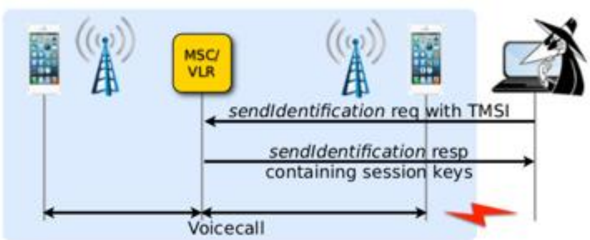


The MSC may be asked to send the IMSI if the TMSI is known (the parameter is: **sendIdentification req**, to which the MSC / VLR will respond with **sendIdentification resp**, containing the IMSI). With **updateLocation**, the attacker can discover the **MSISDN** that belongs to the IMSI.

8. Hybrid Attacks: Intercept Calls

The subscriber's session key can also be requested from the MSC (in this case the intruder sends the parameter: **sendIdentification req** with the TMSI to the MSC / VLR, before which the latter responds with: **sendIdentification resp** containing the session keys).





If the attacker captures an encrypted GSM or UMTS call, he can decrypt it using the session key.

Pay attention that this attack can be classified as "passive" because you do not need to use or request the IMSI (as in the previous case).

9. LTE (Long term evolution)

LTE uses the **Diameter** protocol in the network core

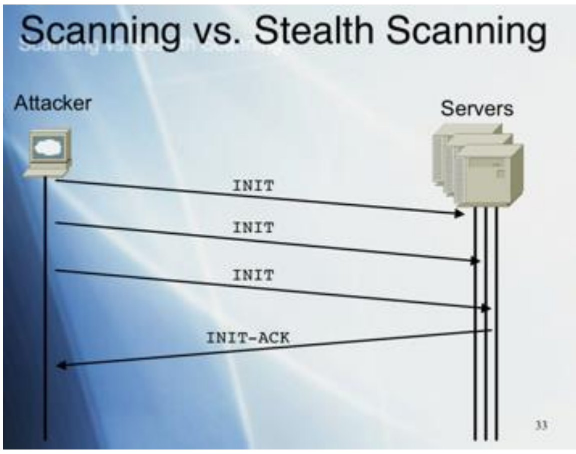
SS7 is becoming an inherited protocol, but:

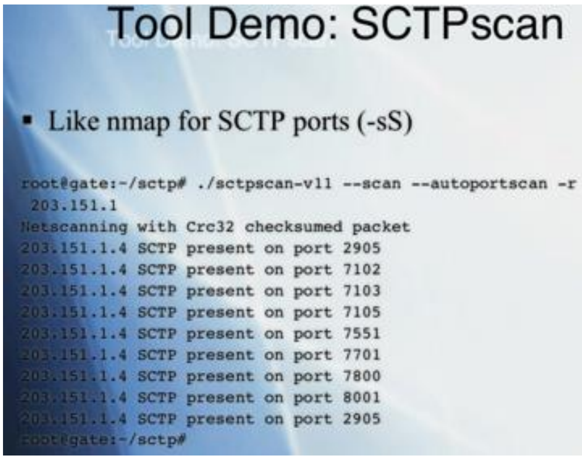
- A large part of the SS7 design has been ported to Diameter, including its defects.
- For example, there is still no end-to-end authentication for subscribers
- GSM / UMTS (and with them SS7) will be available for a long time (probably around 20 years)

In order to have GSM / UMTS connections to LTE, there are interfaces that map most of the functionality of SS7 (including its flaws) in Diameter.

<u>10. Attacks via SCTP protocol</u> (source: "**bh-eu-07-langlois-ppt-apr19.pdf**"[v])

What we have been able to find out about it is mainly based on SCTP scans.





<u>11. Attacks in combination with DIAMETER</u>. (Source: **diameter_research.pdf**[vi])

<u>NOTE</u>: This document "**diameter_research.pdf**" must also be taken into account to evaluate **IMS** and **VoLTE** since it is fundamentally focused on this network.

Many of the current networks and functions of **FTTH** and **VoLTE** that could work basically with **Diameter** (*without the need for SS7*) still need to live together and dialogue with SS7 for inherited aspects, and it is likely that we will continue for quite some time.

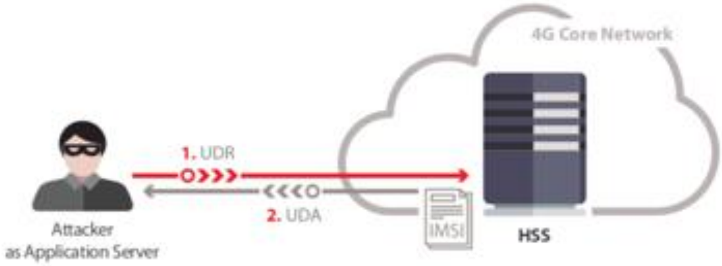
For this reason, this potential attack scenario must be considered.

There are also ways to obtain IMSI from a subscriber through a Diameter network. This requires the mobile subscriber number (**MSISDN**) and the address of an edge node in the Diameter signaling network.

An attack scenario that uses a known vulnerability is as follows. An attacker, acting as SMS Center (**SMS-C**), sends an especially designed SSR (**Send-Routing-Info-for-**

SM-Request) message to the Home Subsciber Server (**HSS**). If successful, the attacker receives the IMSI from the relevant user in response.

In a second scenario, the attacker can pose as an application server and send a specially designed UDR message (**User-Data-Request**) to the HSS. The data received in response to the HSS will contain the user's IMSI.

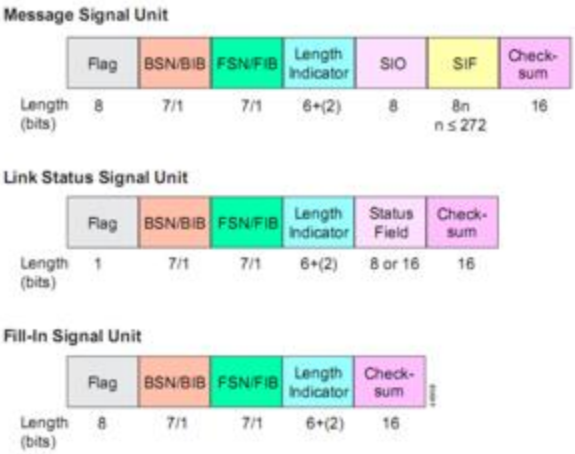


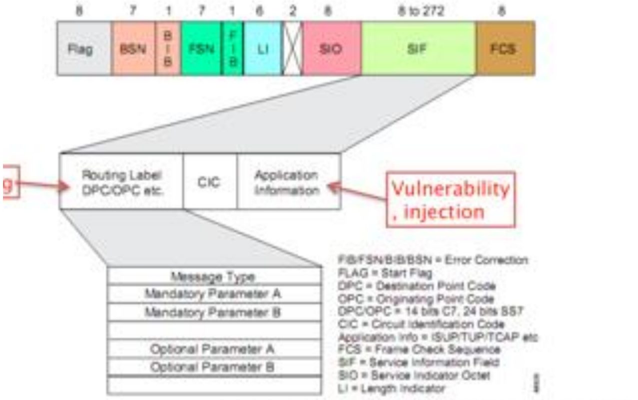
Another way to force the disclosure of IMSI is to attack the **IWF** (Interworking Function) node responsible for the compatibility between the Diameter network and the networks of previous generations. In this case, an SRI4SM MAP MAP SS7 request is translated (or moved) to the equivalent Diameter SRR request. In response, the attacker receives the requested IMSI.

Once the attacker obtains the IMSI addresses more than one subscriber from the nodes of the mobile network that serve the subscriber, he has the information he needs to launch other attacks.

12. ISUP Attacks (Source: **FRHACK2009_Attacking-SS7_Langlois.pdf**[vii])

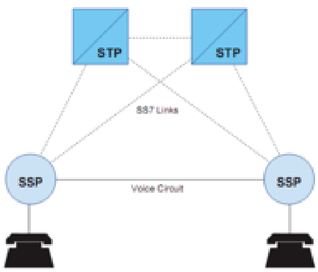
Recall that this protocol (**ISUP**) is the one that uses the SS7 battery for **ISDN** networks.

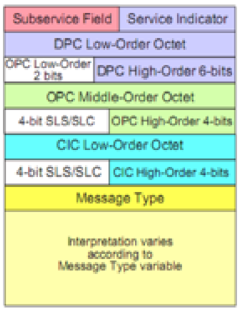




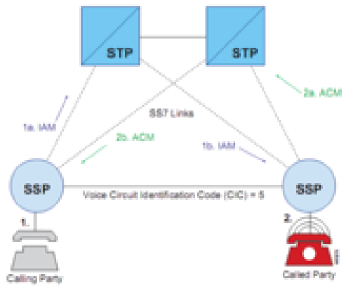
Philippe Langlois, P1 Security Inc, http://www.p1security.com

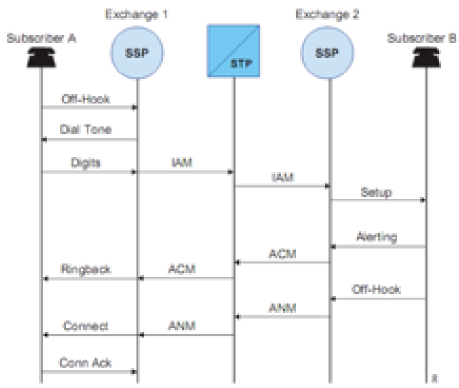
ISUP message (ITU-T)

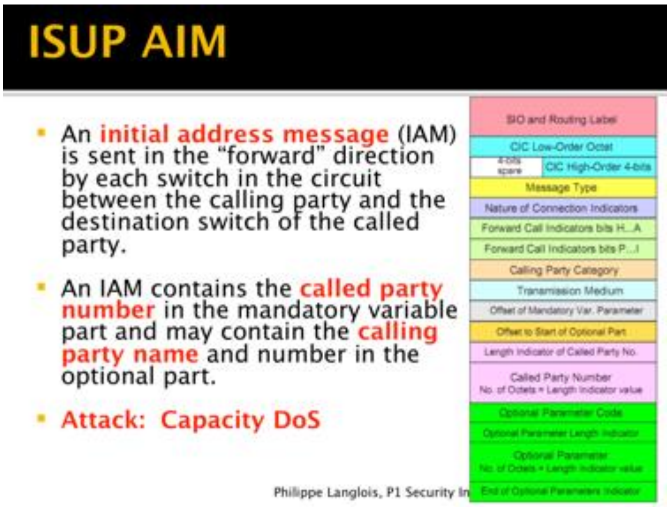


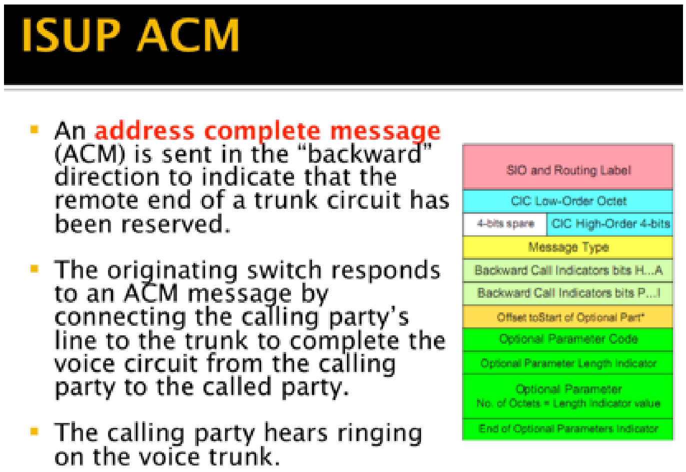


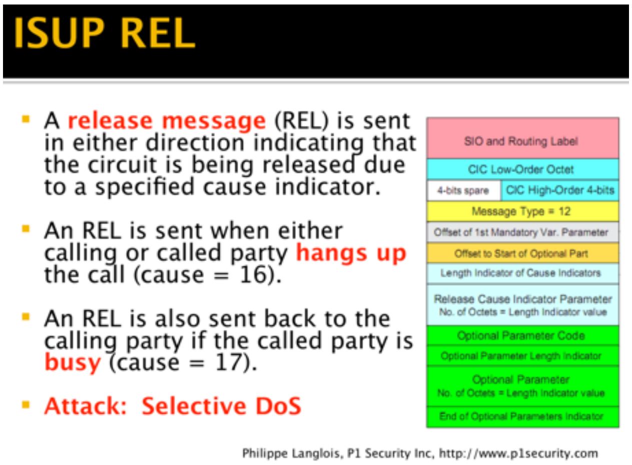
ISUP call initiation flow:

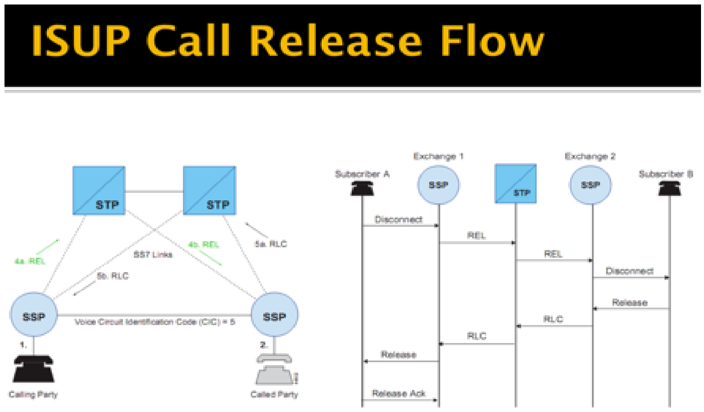


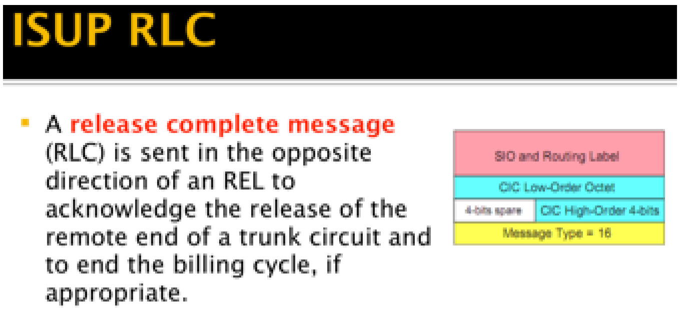


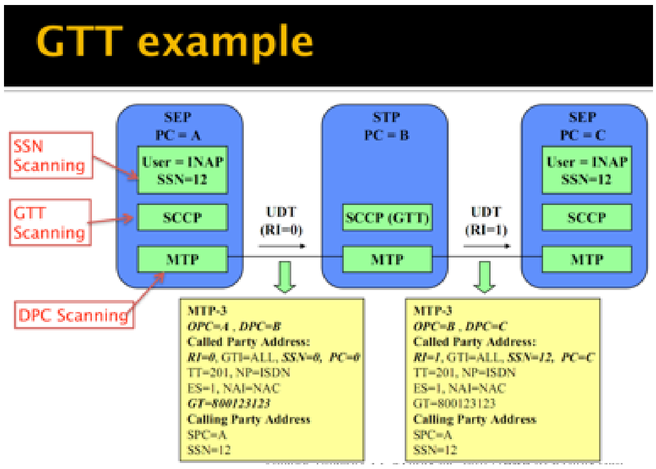












<u>13. Filtered or poorly secured information</u> (information leaks) (Source: **Final Research Report.pdf**[viii])

We have all ever heard or received a work session on the importance of safeguarding the sensitive information of our company, this becomes critical when we talk about the document **IR 21.** This document collects the "technical

specifications" of each operator and is delivered to each operator with which it establishes an interconnection agreement. Gather all sensitive information about the network architecture, network type, protocol versions, IP addresses of the nodes, global title of the nodes, etc.
Just try searching in your favorite search engine "**IR21 filetype: pdf**" or similar searches, you will find more than one document!

RAEX IR. 21 template ver. 8.2

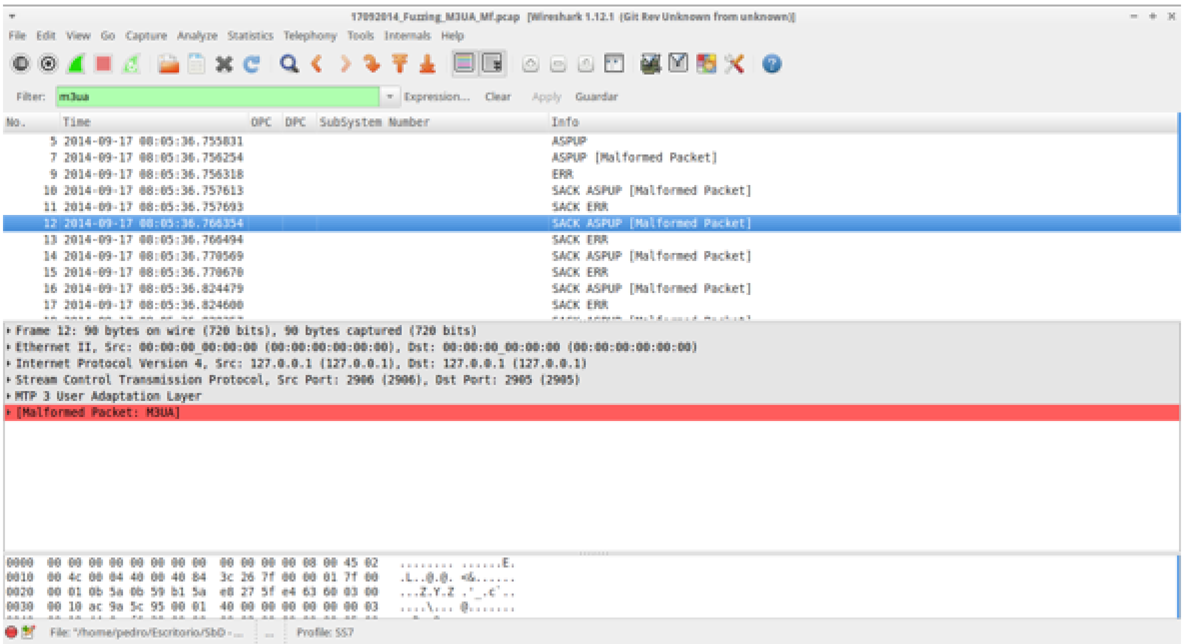
Node type	Node ID	GT address / Address range	IP address / Address range	Vendor info	SW / HW version	Dual access	Location	UTC offset	DST start	DST end
MSC/ VLR-2G+3G	MSSMG01	[redacted]		Ericsson		0	Belo Horizonte	-03:00		
MSC/ VLR-2G+3G	MSSMG01	[redacted]		Ericsson		0	Belo Horizonte	-03:00		
MSC/ VLR-2G+3G	MSSMG02	[redacted]		Ericsson		0	Belo Horizonte	-03:00		
MSC/ VLR-2G+3G	MSSMG02	[redacted]		Ericsson		0	Belo Horizonte	-03:00		

As you can see in the image (*fragment of an IR21*), we can not only see the manufacturer of the nodes and what nodes they are (Ericsson MSCs / VLRS 2G and 3G), their Global Title And also their location.

14. Protocols Fuzzing (Source: Hacking en redes SS7 - Security By Default.pdf[ix])

The Fuzzing is demonstrating lately the great amount of vulnerabilities and programming defects that can be found automatically and the potential of the tools (PROTOS, codenomicon, scapy, etc.) that use this method to study the security or robustness of the software.

In the case of SS7, we can start playing with two tools; scapy and zzuf. Clearly when launching these tools against our SS7 stacks, we can see how the application becomes heavy as well as erroneous messages sent to the server. We can focus on the protocol that interests us to investigate (SCTP, M3UA, SCCP, etc.) and once the message is isolated, forward it to our other machine to check our success:



Using these two tools, it is advisable to adapt or develop a specific monitoring for the application that is responsible for starting the SS7 protocol stack, since it is very possible that at any moment something happens to the unexpected application and we will have to study which message or what situation has been the cause.

15. Internal attacks to SS7 (Source: **Hacking en redes SS7 ~ Security By Default.pdf,** *idem before reference*)

In the aforementioned report we present a series of possibilities that can be executed from the network segments that have visibility with the Sigtran / SS7 infrastructure, it is worth considering it as an attack vector because it is able to perform any of the above.

Final reference of this section.

A document that is also worth considering is the Thesis of [x] Jensen, K. that presents a very useful table about several techniques that have been recommended to provide some mitigations to the vulnerabilities of SS7. These techniques are not specifically designed to stop attacks, but they provide another layer of security.

This table refers to parameters of the MAP protocol associated with the first three categories that we have just presented.

Procedure	Service	Communicating nodes
Location Update	Mobility	MSC,VLR,HLR
Purge MS	Mobility	HLR,VLR/SGSN
Delete Subscriber Data	Mobility	HLR,VLR
Any Time Interrogation	Mobility	gsmSCF,HLR
Short Message Mobile Originated	SMS	MSC,SMSC,HLR
Short Message Mobile Terminated	SMS	MSC,SMSC,HLR
Short Message Alert	SMS	MSC,SMSC,VLR
Retrieve Routing Info	SMS	MSC,HLR,VLR
Send Routing Info For GPRS	PDP	SGSN,HLR
Activate Trace Mode	Oam	HLR,VLR
Send IMSI	Oam	HLR,VLR
Registration Procedure	Supplementary	MSC,VLR,HLR
Erasure Procedure	Supplementary	MSC,VLR,HLR

Table 2: Implemented normal MAP procedures in the simulator.
Image extracted from the document referenced "Jensen.K"

Category 1 messages can be filtered by relatively simple techniques at the edge of the network.
This can be done by evaluating the type of message and verifying whether the message has been sent or not from an "Extenal Net".

Category 2 messages can not be filtered at the edge of the network.
An operator must correlate the subscriber's statuses and verify if the subscriber is in an "External Net" or not before it can be blocked.

Category 3 messages should use more sophisticated approaches. These are messages that have a legitimate use in the network and simply can not be filtered. A protection system needs to analyze the flow of messages from the network and be able to look for changes in the behavior of network elements and subscribers. For example, looking at the previous location of the subscribers.

3	ForwardSM (MO) [2]	SMSC	N	Compare current VLR and Cg SCCP (note 1)		Compare current VLR and Cg SCCP (note 1) (note 2)
3	UpdateLocation UpdateGPRSLocation	HLR	Y	Check Location		Check Location
3	SendAuthenticationInfo	HLR	Y	Check Location		Check Location

5. Traffic capture analysis: patterns to look for, employment of Wireshark.

In these paragraphs we assume that the reader already has previous work experience with "traffic captures" and in particular also in the use of the "**Wireshark**" tool.

On these issues, we have already developed other publications and videos that are at your entire disposal for download and study in the following locations:

Traffic analysis course:

Section: "**Downloads**" → "**Information Technologies**" → "**Networks and Communications**" on our Web: www.darFe.es

Or directly at the following URL:

http://www.darfe.es/joomla/index.php/descargas/summary/4-redes-y-comunicaciones/39-curso-de-analisis-de-trafico

We also have a sequence of "**six videos**" on the theme of "Traffic Analysis" using Wireshark on our "Youtube channel":

https://www.youtube.com/user/infoDarfe/videos

Also, if you want to practice more, we have several examples of "**traffic captures**" made and classified by protocols, which you can also download for free at:

https://www.darfe.es/joomla/index.php/capturas

In short, we invite you if you have not yet started your work on "traffic analysis" you can refer to the addresses and publications mentioned, and reiterate, in the following paragraphs we take these basic aspects as known.

Next, we will develop the state in which we are on analysis of SS7 / Sigtran to begin to "raise awareness" about the importance of being able to evaluate or analyze these flows from the point of view of the safety of a signaling network . There is an important document that we must keep in mind for this evaluation:

FS.11 - SS7 Interconnect Security Monitoring Guidelines[xi].

Recall some paragraphs:

First, the operator must:

- ✓ Understand that SS7 is no longer secure and should be separated from other SS7 networks to protect its own network and its subscribers.
- ✓ Have control over their own SS7 elements, which means that an operator can separate its internal network, or home network, from all other external networks.
- ✓ Secondly, the operator must be able to **capture the traffic** that enters the defined edge of the network, making it possible to determine from where a message originated, externally or internally.

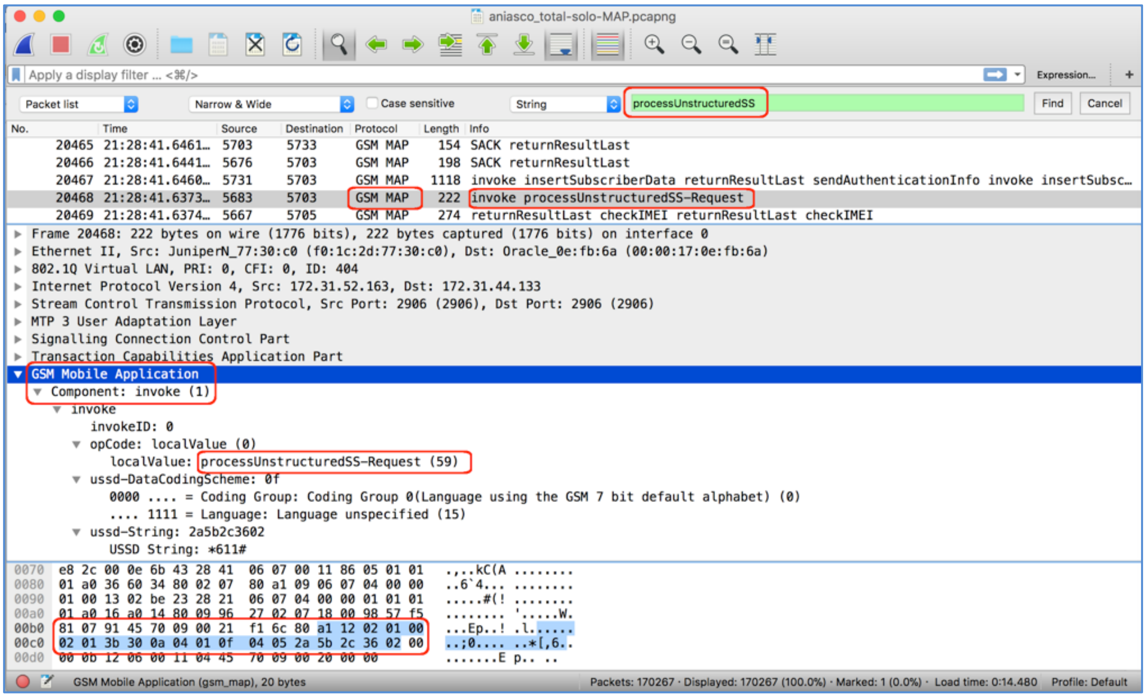
The document: "**FS.11 - SS7 Interconnect Security Monitoring Guidelines - Version 1.0**" (19 November 2015). In Point 2.2. How to monitor:

The goal of monitoring is **to assess whether suspicious / malicious SS7 activity is occurring**. How to achieve this, will vary between the operators and the capabilities of each operator, as well as their objectives. The monitoring effort may vary from:

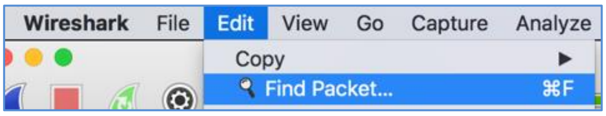
- ✓ Sampling a portion of the interconnection traffic for a limited period of time, looking for known issues, to determine if the problem is occurring, or
- ✓ Monitoring all interconnection traffic continuously, both incoming and outgoing, to determine the maximum scope of the problem and looking for possible new attacks.

When we work with **SS7 Sigtran** traffic captures, we can concretely evaluate this type of traffic patterns that we developed in the previous sections. In our case we will work with "**Wireshark**" that we invite you to install in a virtual machine, if it is a Linux distribution, Debian, "**Kali**" better, that better, because it will also facilitate the work with several additional tools that already comes preinstalled.

Let's start our work on "traffic captures".



As you can see in the previous image, we have started to "search" different types of "occurrences" within the global capture. In this particular case, we have selected the option "find Packet" from the "Edit" menu.



Once this option is selected, the "**Find**" bar will appear in the upper part of Wireshark, in which we can see in the previous image that it has been decided to look for the "**ProcessUnstructuredSS**" parameter, which is one of the "MAP" protocols. , in turn we have decided to look for it as "**String**" and within the "**Packet list**" window (*It could have been also in the "Packet details" or "Packet Bytes" windows*).

In the previous capture, we have highlighted how we can identify certain parameters that can help us identify several things:

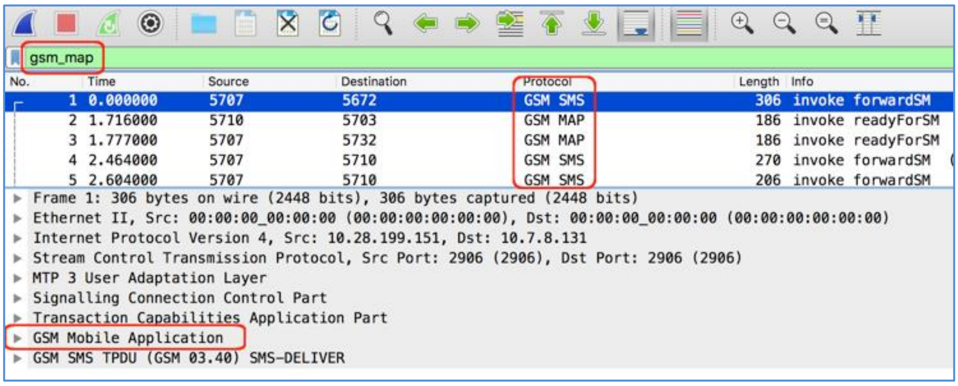
- Protocols that are being used (Ex: GSM MAP).
- Parameters used in that frame (**ProcessUnstructuredSS**).
- Source addresses, destination at any level (IP, SCCP, IMSI, TMSI, MSISDN, etc).
- Requests and answers (Invoke = Request).
- Hexadecimal sequence that circulated through the network.
- Etc.

Based on the catches made, with "**tcpdump**" or "**Wireshark**" we can go "exporting" the types of captures that we want, and go crumbling a high volume of traffic until we reach the flows that we want.

The other advantage of working with this tool is to be able to identify the traffic patterns that we can consider suspicious (as indicated by "FS.11 - SS7: **if suspicious / malicious SS7 activity is occurring.**").

As we appreciate in the previous image, we have all the information of the patterns that make reference to all the security documents SS7 / Sigtran that we have presented, both in text and in hexadecimal.

We will progress little by little with the identification of these parameters, but to go ahead a little in the subject, and as we have just seen in the previous image, first of all if we want to start studying the attacks in the order that we present our classification of fifteen of them, for example we can initially focus on the frames containing the "MAP" protocol, for this we simply place in the visualization filter: "**gsm_map**".

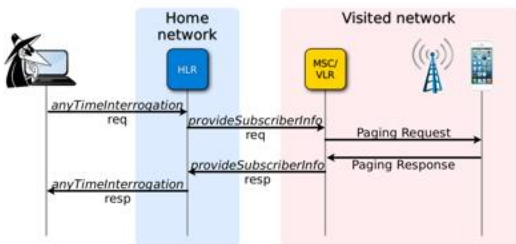


As we can see, we have filtered all the frames that contain this protocol.

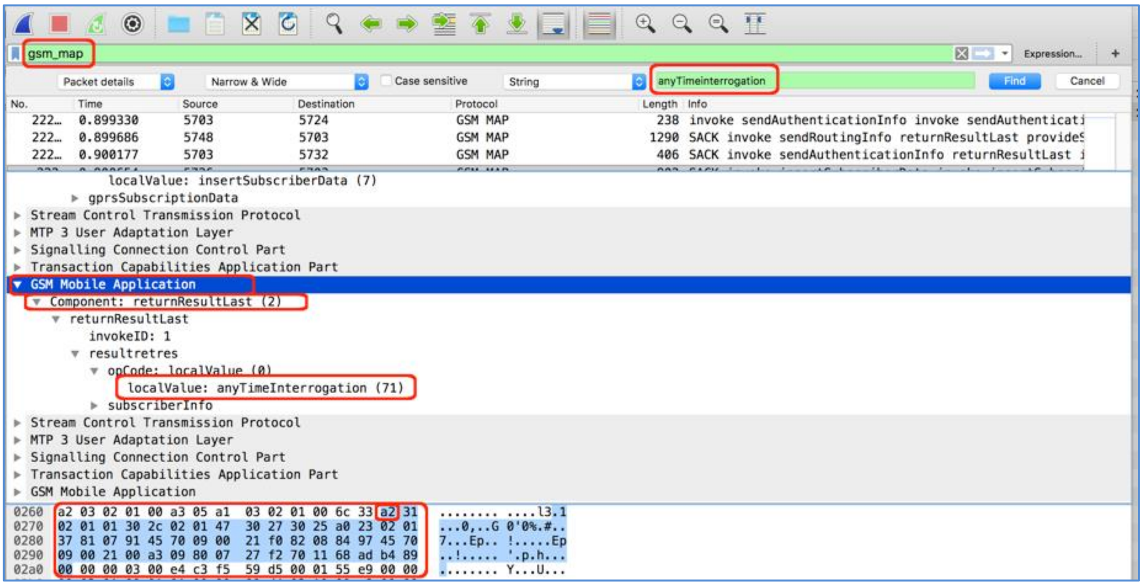
Let's begin to apply these concepts to the specific cases that concern us, for example, let's return to our **attack No. 1** on the list of our classification (*of the 15 of them we have presented*).

This attack: **1. Information search on cells-HLR-VLR / MSC**

Recall that in this case, the initial analysis of the attack should focus on finding within the MAP protocol the parameter: **anyTimeInterrogation (ATI)**, but "only when the HLR, sends its response to" outside the HOME_NET "(*do not forget that the SOLUTION, just block this response out of the HOME_NET*).



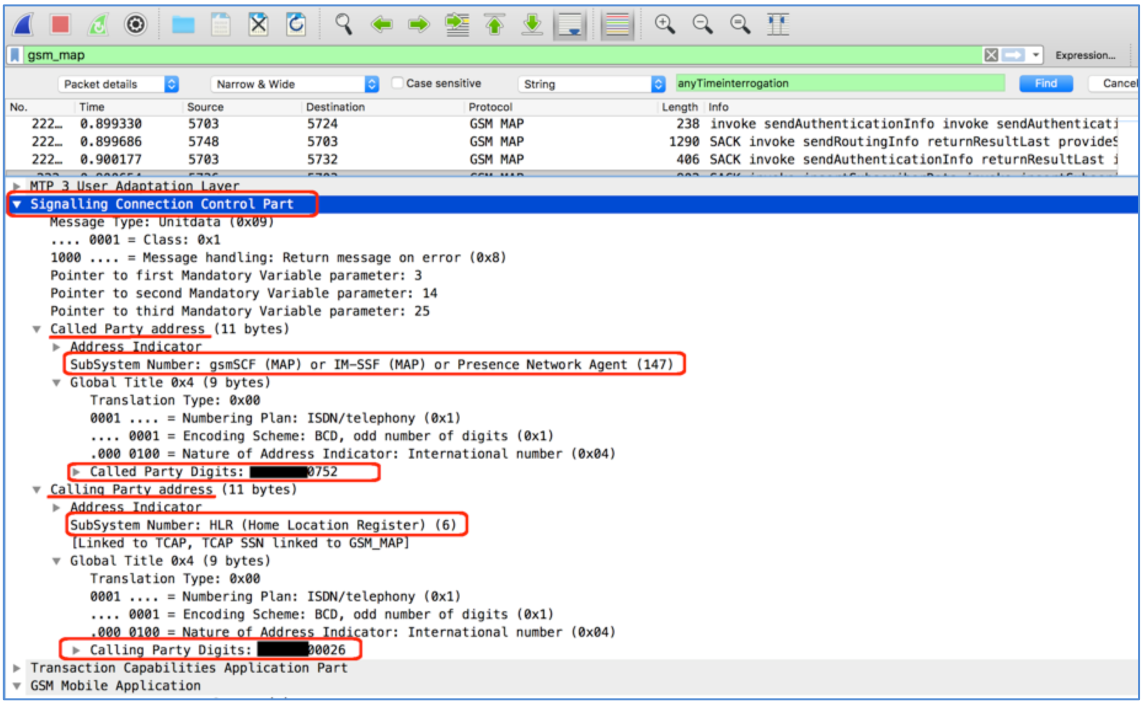
Therefore, the beginning of our first analysis could be exactly what is presented in the image that follows.



As we can see in the previous image:

- We have placed a "visualization filter" so that it only shows us "**gsm_map**" protocol.
- We have done a search, so that we only present those "MAP" frames that contain the "**anyTimeInterrogation**" parameter.
- In this particular case it is a response, which we can see in the Component field: "**returnResultLast**".
- This last parameter allows us to move forward and begin to present what we will soon execute with the IDS "**Snort**". When we started to pay attention to the third window of Wireshark, this is the one that offers us the information in "hexadecimal", that is the primary translation of the "bits" that actually circulated through the communication channel. In the case of the MAP protocol, we can identify that it is a response (that is to say, the parameter we are looking for "**anyTimeInterrogation_resp**"), because as we have highlighted in **red**, this value in hexadecimal is identified by the hexadecimal value "**a2**". When dealing with a Request (or request) in MAP, this field has hexadecimal value "**a1**". These values in hexa, we will see later that are fundamental if you want to work with "**Snort**".

But keep in mind that this can only be classified as "anomalous" if and only if the "**HLR, sends its response out of the HOME NET**", therefore these filters that we are using are not enough, because we do not see this in the protocol "MAP", but we must go down to a protocol of lower level in our stack. In this specific case we have it relatively easy because we have the **SCCP** protocol whose addressing scheme we presented in previous sections and it is precisely who can tell us with total clarity from whom and to whom will direct that traffic. This detail is presented in the following capture.



In this capture we have deleted the addresses (or telephones) because it is real traffic of a telephone operator. again, we have highlighted in red the parameters that offer us information to evaluate this potential attack, which in this case are:

- Deployment of the fields of the SCCP protocol (which is pure SS7).
- **Called Party Address**: who is requesting this information.
- **SubSystem Number** (SSN): presented in previous sections, which clearly indicates which element is involved. In this case we can see that it is a **gsmSCF**.
- **Calling Party Address**: that is, who you want to communicate with.
- **SubSystem Number** (SSN): presented in previous sections, which clearly indicates which element is involved. In this case we can see that the destination is an **HLR**.

Returning to the analysis of our **attack No. 1**, remember that the "sense" of these parameters is "only when the **HLR**, sends its response to" outside the HOME_NET", therefore, it is clear that in the capture of traffic from the previous image, it is strictly the other way around (it is an SSN = **gsmSCF** towards an SSN = **HLR**), but here we have some information that will be very useful for us:

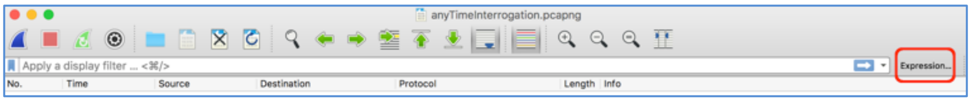
We can apply a visualization filter, just on **SCCP** protocol

and include the "**SSN**" fields.

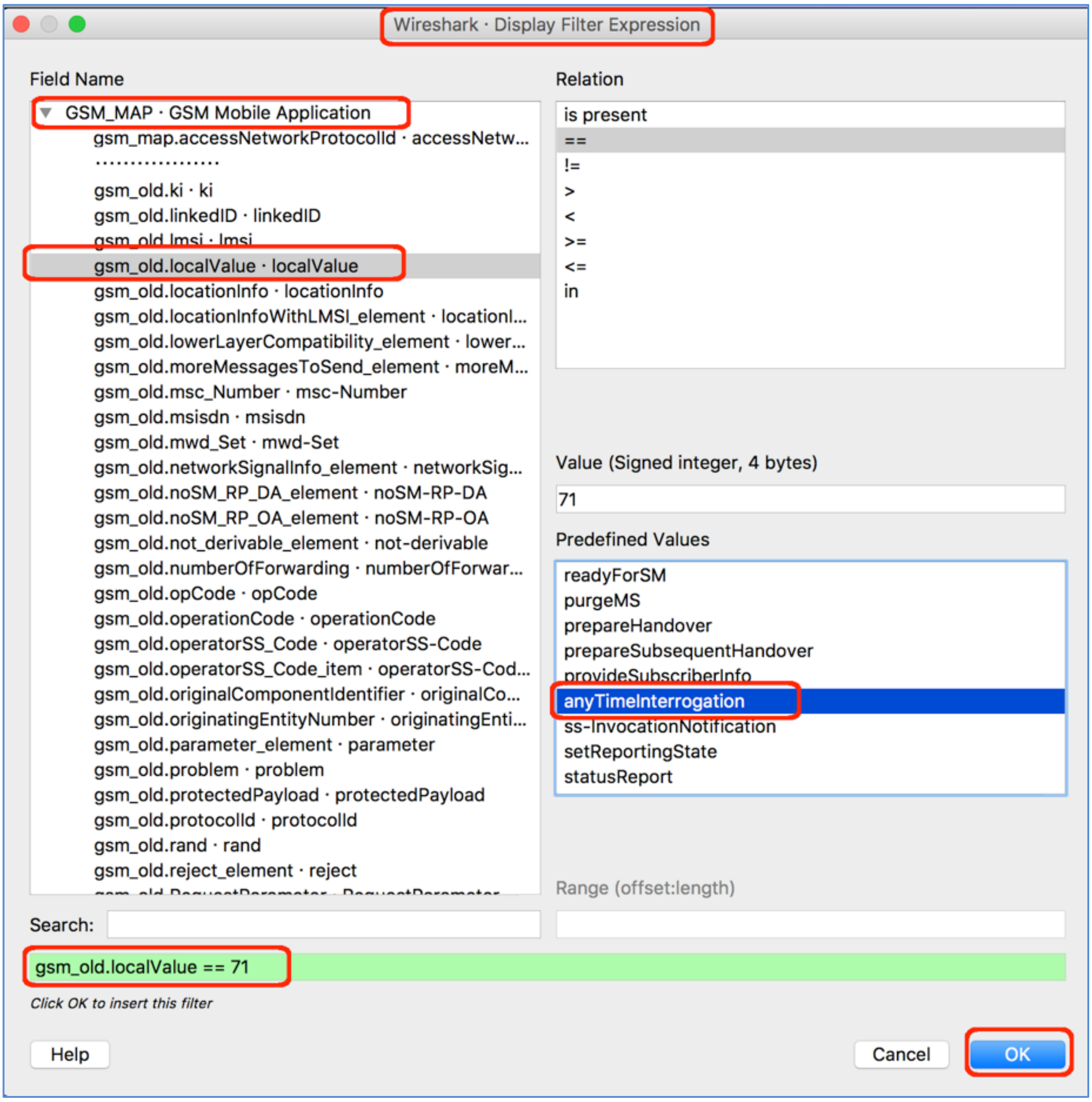
Let's see how to do it.

STEP 1: First, let's start with the data that we are interested in processing and discard what, in this specific case (**attack No. 1**) does not interest us. The parameter that we need to study without any doubt is: "**anyTimeInterrogation**", so we can start by applying a "visualization filter", so that it only shows us the frames that contain this parameter. Wireshark has pre-loaded hundreds of communication protocols, and for each of them, most of the parameters it supports. In our case study, the protocol "**gsm_map**" has nothing more and nothing less than 2,287 parameters, and each of them in turn admits "n" number of values.

Here is an image of how to configure it.



As we can see in the previous image, in the upper part we have this bar that is just to apply the "visualization filters" (*inside the white window it reads: "**Apply a display filter**"*). If we select "**Expression**" (as shown in the box "red"), the window whose image we present below is displayed, which in our case was downloaded by the different protocol families that Wireshark offers us until we reach "**gsm_map**".



As you can see in the previous image, of the 2,287 parameters, in our case we are looking for "**anyTimeInterrogation**", which is one of the values of the true MAP parameter itself that is called: **gsm_old.Value**, and that for the value "**anyTimeInterrogation**", corresponds to the number"**71**".

Important note: Remember that MAP is one of our main protocols when it comes to vulnerabilities "SS7 / Sigtran", therefore moving within it will be fundamental for our analysis. In this particular case, the parameter "**gsm_old.Value**" offers us a lot of information, for example, in advance of other attack patterns, within this field, we also have:

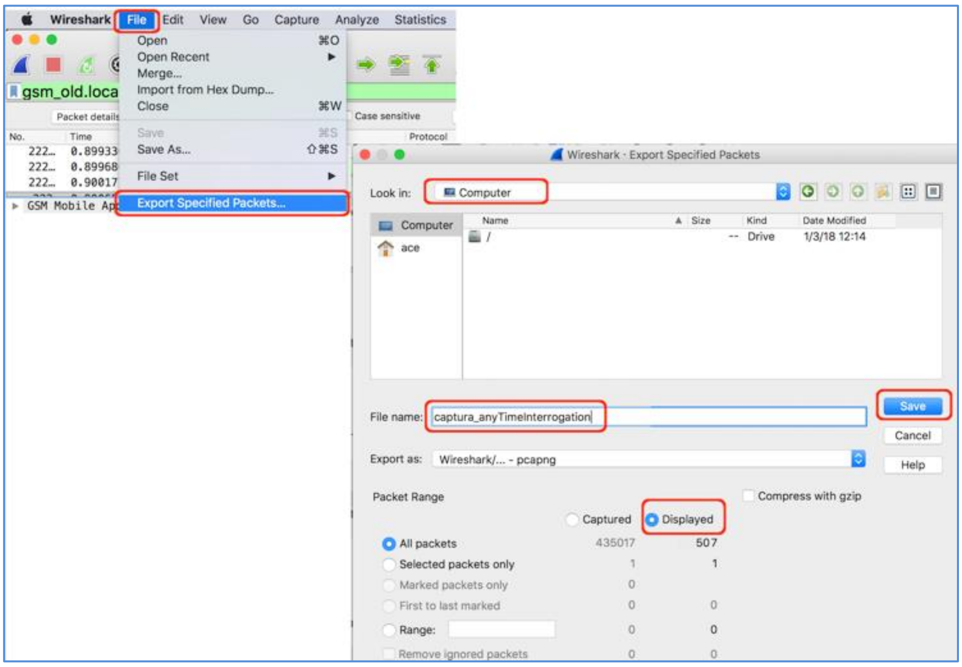
gsm_old.localValue == 2 -----------> **updateLocation**
gsm_old.localValue == 3 -----------> **cancelLocation**
gsm_old.localValue == 7 -----------> **insertSubscriberData**
gsm_old.localValue == 8 -----------> **deleteSubscriberData**
gsm_old.localValue == 19 -----------> **ProcessUnstructuredSS-Data**

gsm_old.localValue == 22 -----------> **sendRoutingInfo**
gsm_old.localValue == 45 -----------> **sendRoutingInfoForSM**
gsm_old.localValue == 60 -----------> **ProcessUnstructuredSS-Request**
gsm_old.localValue == 70 -----------> **provideSubscriberInfo**
gsm_old.localValue == 71 -----------> **anyTimeInterrogation**
gsm_old.localValue == 83 -----------> **provideSuscriberLocation**

With these values for **gsm_map**, we are practically covering all the patterns of attacks that we present in this text for mobile networks.

So, until now we have managed to apply a visualization filter so that Wireshark shows us only the frames that contain the parameter "**anyTimeInterrogation**", now the most practical way to keep moving forward is to "save" this selection in which we know we can continue analyzing specifically this value.

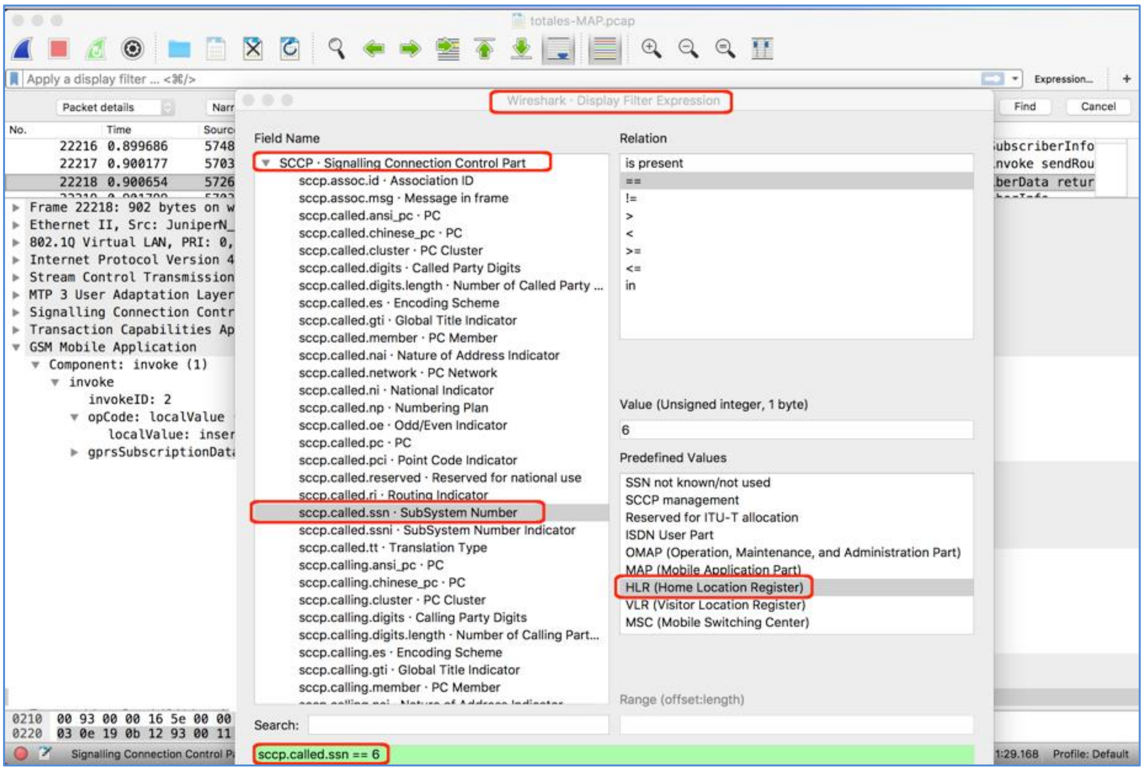
To do so and only the frames that contain this parameter remain, we can do it as it is represented in the image that follows.



As you can see in the previous image, this option is "**File**" → "**Export Specified Packets ...**" and from there we select the desired directory and name (in our case "**captura_anyTimeInetrrogation**"), and it is very important that we select within "**packet Range**" → "**Displayed**", so that we only remain with the packages that contain this parameter, discarding the rest (*we can see in the image that only **507** packages of the original **435017** will be saved*).

<u>STEP 2:</u> Now, working with this new saved capture, we will continue filtering the search so that it does not only present the SCCP frames that have an **<u>HLR origin</u>** (*as indicated by **attack No 1***).

To do this, in the same way that we work with the visualization filter for "**gsm_map**", the "**SCCP**" protocol is also preloaded and has another number of configuration options for filtering, as we can see in the following image.



Once again, we have highlighted in **red** the parameters that interest us to continue evaluating **attack No. 1**. In this case, as we have been commenting, we are interested in identifying concretely when "from" an HLR this parameter was sent, therefore , as shown in the previous image, we must select "**sccp.called.ssn == 6**" which is the SubSystem Number (from SCCP) that identifies an HLR.

<u>Another important NOTE</u>: Like **MAP**, in this case it is very likely that in our subsequent analyzes we will also have to identify another type of elements or protocols of the SS7 / Sigtran network. This is where we should look for them and then we present a list of the main ones for our work (*the values that follow are for "**sccp.calling.ssn**" but they are identical if we wish to apply them for "**sccp.called.ssn**"*).

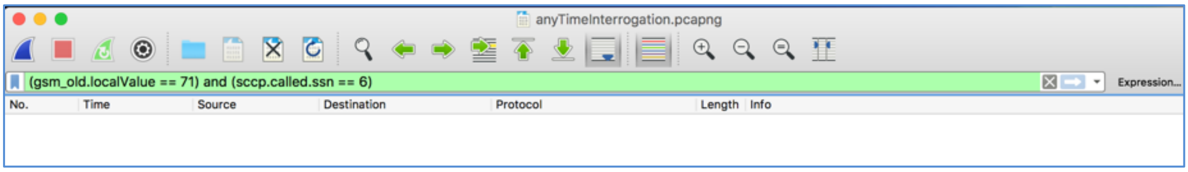
<u>Sccp</u> <u>filters</u>:

sccp.calling.nai == 0x4 (**International Number**)
sccp.calling.ssn == 3 (**ISUP**)
sccp.calling.ssn == 5 (**MAP**)
sccp.calling.ssn == 6 (**HLR**)
sccp.calling.ssn == 7 (**VLR**)
sccp.calling.ssn == 8 (**MSC**)
sccp.calling.ssn == 9 (**EIC/EIR**)
sccp.calling.ssn == 10 (**AuC**)
sccp.calling.ssn == 145 (**GMLC MAP**)
sccp.calling.ssn == 146 (**CAP**)
sccp.calling.ssn == 147 (**gsmSCF** (MAP) or IM-SSF (MAP) or Presence Network Agent)
sccp.calling.ssn == 149 **SGSN** (MAP)
sccp.calling.ssn == 150 **GGSN** (MAP)
sccp.calling.ssn == 250 **BSC** (BSSAP-LE)
sccp.calling.ssn == 251 **MSC** (BSSAP-LE)

In short, the filter that we would be interested in applying would be a "concatenation" of what we have been presenting, which would concretely look like:

(gsm_old.localValue == 71) and (sccp.calling.ssn == 6)

In the image that follows we can see it graphically.



As we can see, after applying this filter we have not been shown ANY frame, therefore this implies that the "sample" of evaluated frames HAS NOT EXISTED **attack No. 1**, since no HLR has sent the parameter "**anyTimeInterrogation**", in our case towards any type of destination.

More detail about SCCP.

As we were presenting, another protocol that controls Wireshark is "**SCCP**" that, for us, as we mentioned at the beginning of everything, is very important, because for example as we have just seen, it allows us to identify the "**calling** and **called part**" that are The true origins and destinations of calls in pure SS7. In this way we can identify calls that come from outside, for example with the following display filter:

sccp.calling.digits matches 34 and not sccp.called.digits matches 34

In the previous filter, we would be specifically telling Wireshark to show us all the frames whose origin is outside of Spain (34) and whose destination was in Spain, but the important thing is that we say it at the SCCP level, that is, below Sigtran, therefore, will be the true devices of the pure SS7 network that establish this dialogue.

We wanted to emphasize this parameter, as we have already noticed, we have been emphasizing the concept of "**Home_Net**" and "**External_Net**". These two concepts are fundamental for any area that operates the "nodes" of the SS7 network, because as anyone can easily deduce, if you do not know which frame corresponds to an origin and/or destination of "my" SS7 architecture and which one "alien" to it, because we can not evaluate the occurrence of an attack or not.

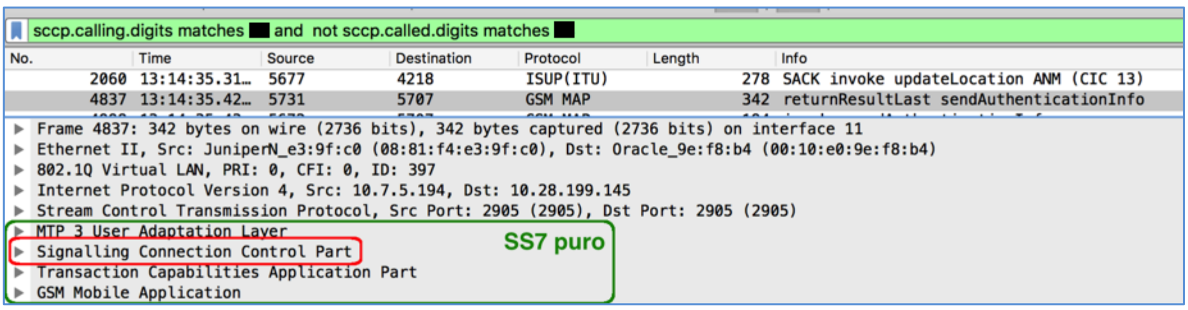
This seems excessively trivial, in the reality of day to day is not so, well let's not forget that these architectures were born in the 80' as something closed and thus grew under these concepts. The operators of these networks are usually people who have been in the area for many years and it is very difficult to break this "inertia of thought". With much more frequency than we believe, we will find, that there are no network maps, no clear inventories, no locations, no unambiguous IP addressing schemes, etc. With this, we would have enough problem, but in turn there is another worse.

In many of the SS7 architectures, **NAT** (*Network Address Translation*) is used, therefore, if we want to look for "internal" (*Home_Net*) and/or "external" (*External_Net*) traffic patterns through IP addressing, in these cases ALL the frames will have a private IP address (source or destination) within this range, making it practically impossible to know at the network level, ie by its IP address, if that frame comes or goes outside our architecture (*or Home_Net*).

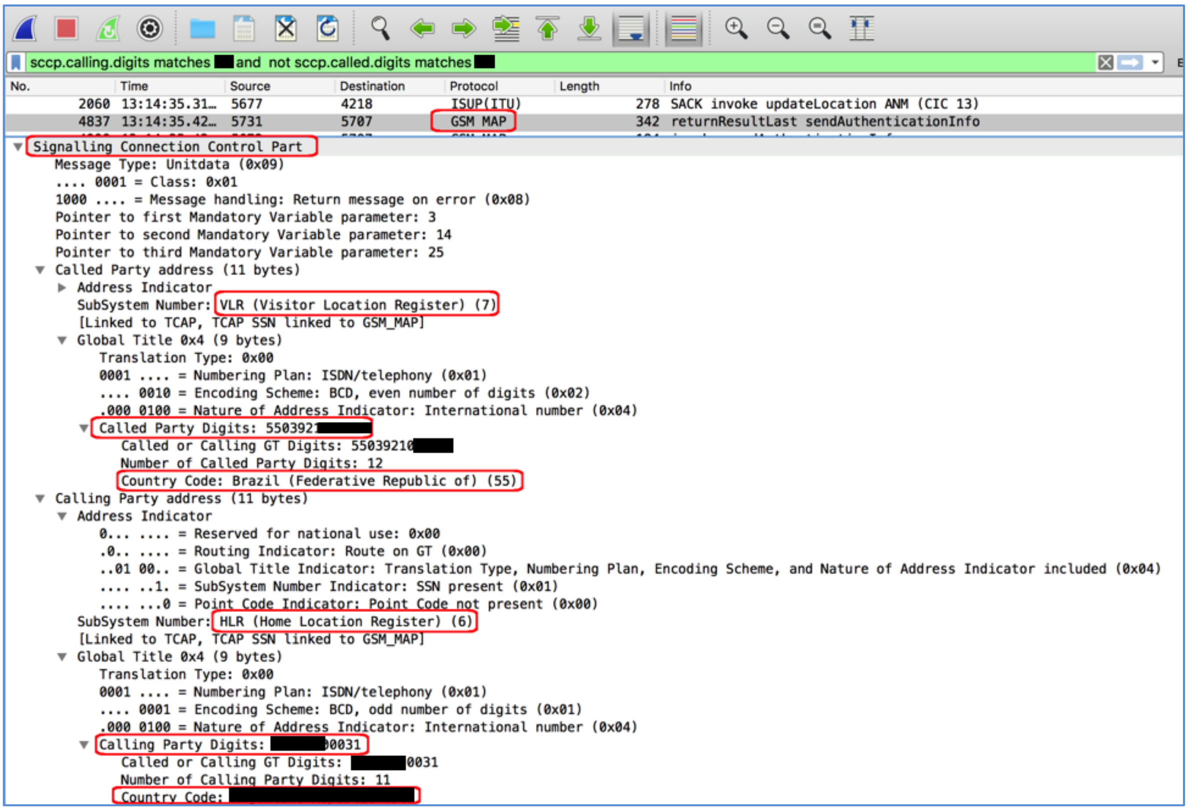
In some cases (*we could almost call "privileged"*), the "entry point" to the Home_Net is only one (for example an STP), before which, it can be inferred that if a frame comes from outside (External_Net) it will from that only IP address, but reiterate, this should be a NORMAL situation from the point of view of the security of a network SS7/Sigran, it is a situation almost "privileged" because it is not normal that this happens.

In any of the situations described, we have the SCCP protocols as an ally, through which we can find a satisfactory solution to the analysis of these frames.

In the images that follow, we can appreciate by means of the analysis and filtering of these parameters. First let's look at the SS7 package in Sigtran and pay attention to "**SCCP**" (*Signaling Connection Control Part*).



In the image that follows, we display the fields of the calling and called party through a frame that comes from a **VLR** of Brazil (Called part), towards a **HLR** (calling part) of another Country (that we will hide intentionally).



All these fields and nodes origin and destination, we can filter them perfectly with "Wireshark" through the visualization filters that we presented recently in the:

Another important NOTE: ....

Sccp filters:

sccp.calling.nai == 0x4 (**International Number**)
sccp.calling.ssn == 3 (**ISUP**)
………

Summary of this section.

We have initially presented the importance of the fact of being able to "analyze traffic" SS7/Sigtran, since it is about the information flows that circulate through our signaling network, therefore, necessarily, there will happen, or not, the attacks on it

We started working with the tool "Wireshark" for this type of analysis, but always taking as reference the patterns of a real attack, which in our case we have done on the basis of this own classification through which we had presented as "**attack No. 1** "

Sequentially approach each of the actions and steps we can take to "crumble", filter and obtain concrete results on these parameters and attack flows, until we verify that this first attack was NOT in our "sample" of traffic.

This last conclusion, we can not ignore it. Firstly because it is irrefutable evidence, but secondly because we can not trust it, because it is only a "sample", which should lead us to the conclusion that we have another additional task that is to refine, adjust, maximize or optimize the samples that we take, in terms of the listening segments, the "**capture filters**" (*which we have not developed here but that are quite simple to apply, both with wireshark, tshark or tcpdump*), the schedules that we do, the addresses IP, etc ... this is a very good challenge to face, but it depends on each particular signaling network.

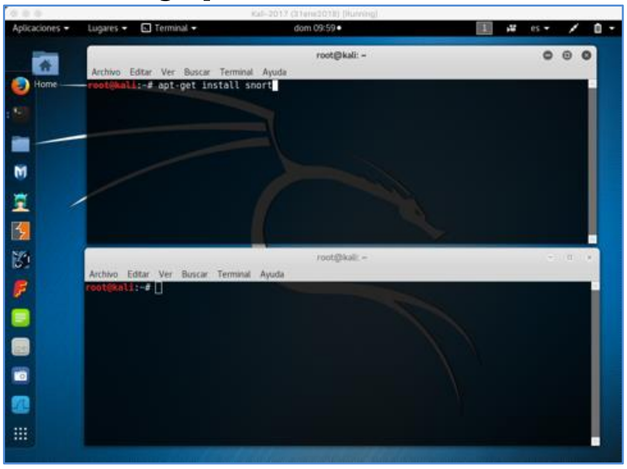
Finally, leave the message and teaching that this same procedure, we can apply in the same way to the rest of the patterns and / or attack parameters that we have been presenting. In this section we end with this first attack, because as it is commonly in Spain said "for sample, just one button is enough".

6. Traffic capture analysis using Snort.

Once presented the initial work that we can be doing with Wireshark, let's move on to the use of the second tool "**Snort**".

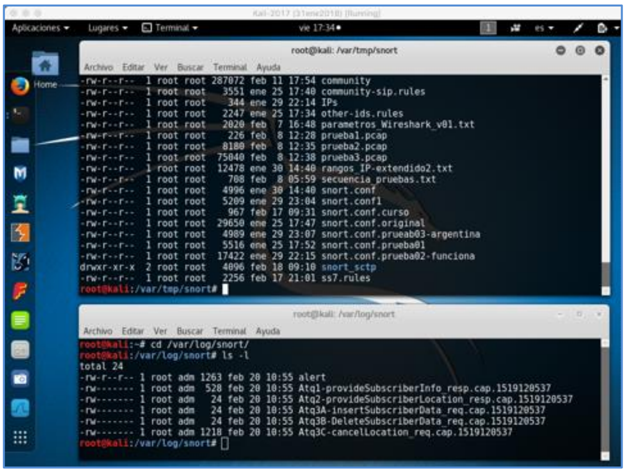
Our advice, as expressed in section 5. of this document, is that you work with a "virtual machine" and install a Linux operating system, if possible with the "Debian" distribution of "**Kali**".

Once installed the graphical interface is the one that we are seeing on the left.



For our part, in the case of working with Snort, it is practical for us to have two consoles open. Another aspect is that "Snort" is not installed in "Kali" so we must install it with the command "**apt-get install snort**", as you can see in this image.

The work proposal with two consoles, as we will see later, makes it easier for us to be able to execute "Snort" and see their results simultaneously. For this, it is convenient to be positioned in the upper window on the route where we have the configurations and rules and in the lower window, the route to where we decide to send the "outputs", in our case, as shown in this image, the working directory will be: "**/var/tmp/snort**" and the one of the outputs: "**/var/log/snort**", as we can see in this image.



In our case, we have been working with this **spectacular IDS** for twenty years and, due to the enormous power it offers, we recommend that if the reader decides to use it, he does so consciously and for that, he resorts to the best source of information that is sufficiently complete and updated on its website of origin and in particular in its manual "**Snort Users Manual**", which can be downloaded at: https://www.snort.org/documents

This **IDS/IPS** (*Intrusion Detection/Prevention System*) open source, offers us the possibility of working "**On Line**" and also "**Off Line**", therefore, we can choose

the methodology that best fits our operation.
From the point of view of simplicity in our case, maybe the best option is to work "Off Line" (*of course, if we have the possibility of installing it as a probe in some segment of the network SS7/Sigtran and "mirror" traffic to it, or place it with a "Splitter" is another possibility "On Line", and even much better option*). In the case of operating "Off Line", we can request the different types of "traffic captures" that we need, for example:

- By zones.
- By IP addresses.
- By type of element (STP, MSC, HLR, etc.).
- By interface (external, internal, service, etc.).

Always considering that we must be strict in showing that this activity is basic and fundamental in an SS7 network (and every operator of these nodes "SHOULD" be able to make these captures, as indicated by international standards).

To move forward with this text, we will present a way of working "Off Line" on the basis of traffic captures taken in a segment of SS7/Sigtran.

We are not going to develop a Snort course, we will only present the basic concepts to understand this work proposal. We take it for granted that we already have our virtual machine "**Kali**" running, therefore, we will focus on three concepts:

- **Configuration file.**
- **SS7.rules**
- **Outputs**

6.1. The configuration file.

Among the many options offered by Snort, one of them is precisely to be able to use it as "**IDS**", for this, the first step is to be able to execute it by calling its configuration file, as we will see in this point, concretely this is done with the "-c" option indicating where this file is located.

The configuration file, when Snort is installed, already brings us a pre-configured model (**snort.conf**) that we can use almost immediately with some small adjustment. In our case of the many options that it offers, as we will see immediately, we are only interested in very basic aspects. This configuration file consists of four basic components:

- The Sniffer.
- The preprocessors.
- The detection engine.
- Outputs.

As we already mentioned, we will not go into the detail of this file, because it is not a Snort course here, we will only focus on the details that we need specifically for our work.

If you wish to deepen a little more methodologically about this software, we have an article published on our Web (www.darFe.es), in the Section:

"Downloads" → "Information Technologies" → "Security"

That we can access through the following URL:

http://www.darfe.es/joomla/index.php/descargas/viewdownload/5-seguridad/45-metodologia-nessus-snort

Thinking about analyzing SS7/Sigtran traffic anomalies, we must keep in mind again our classification of the "15 types of attacks". Remember that in all of them it was necessary to identify with complete certainty the origin and destination of the frames, because we do not forget that any parameter, for example: **anyTimeInterrogation (ATI),** we can classify it as a potential attack "only when the HLR, sends its response towards "outside the HOME_NET".

Under this idea then, a starting point for the configuration of our IDS, is to be able to indicate as accurately as possible, all the elements that are "HLRs", "MSCs", "SMS-Cs", etc.

This configuration is part of the first section of "**snort.conf**" and it is done by defining what are the "variables" that we are going to use. In our case, they are the IP addresses of each of the devices that make up our SS7 / Sigtan architecture.

Below we present a series of examples of how this section of our "**snort.conf**" file could be.

Setup the network addresses you are protecting

ipvar HOME_NET
10.2.16.64/26,10.2.17.128/25,10.2.19.192/29,10.2.19.200/29,10.2.19.64/29,172.30.16.128/28,
172.30.16.160/28,172.31.10.128/28,172.31.4.0/24,172.31.22.160/28

ipvar **EXTERNAL_NET** any (*or we can also put* **!HOME_NET**)

#var SS7 (*it's just our comment to clarify that it's about SS7)*

ipvar MSC 10.3.1.0/27,10.4.1.0/27,10.5.1.0/27

ipvar HLR-HSS 10.30.1.0/27,10.31.1.0/26

ipvar SMS-C 172.17.31.10/32,172.17.31.11/32,172.17.31.12/32

ipvar GW 172.17.33.10/32,172.33.12/32

portvar STP_ports 2905:2913

var **RULE_PATH** /var/tmp/snort (*is the PATH that we have chosen for the rules* *that we will design on SS7)*

6.2. Outputs

The second section of the "**snort.conf**" file that we want to define properly is "**Outputs**". Below we present a series of examples of how this section could be, since Snort supports several types of them.

output alert_csv: alert.csv *(If we want to obtain outputs that later facilitate their exploitation, for example, by means of calculation templates).*

output log_null *(Standard output in "Log" format, or not)*

output log_tcpdump: SMS.cap *(Standard output in "tcpdump" format)*

```
# Output in "pcap" format - Attact 1 detected
```

(We can define our own Output format based on a certain rule, it will be seen more clearly after we present the "Snort rules").

```
ruletype provideSubscriberInfo_resp {
    type alert
    output log_tcpdump: Atq1-provideSubscriberInfo_resp.cap
    }

# Output in "pcap" format - Attact 2 detected
ruletype provideSubscriberLocation_resp {
    type alert
    output log_tcpdump: Atq2-provideSubscriberLocation_resp.cap
    }

# Output in "pcap" format - Attact 3A detected
ruletype insertSubscriberData_req {
    type alert
    output log_tcpdump: Atq3A-insertSubscriberData_req.cap
    }

# Output in "pcap" format - Attact 3B detected
ruletype DeleteSubscriberData_req {
    type alert
    output log_tcpdump: Atq3B-DeleteSubscriberData_req.cap
```

```
    }

# Output in "pcap" format - Attact 3C detected
ruletype cancelLocation_req {
    type alert
    output log_tcpdump: Atq3C-cancelLocation_req.cap
    }
```

Finally, we must indicate where you should go to look for the rules that we define for SS7 (*which will be developed in the following point*).

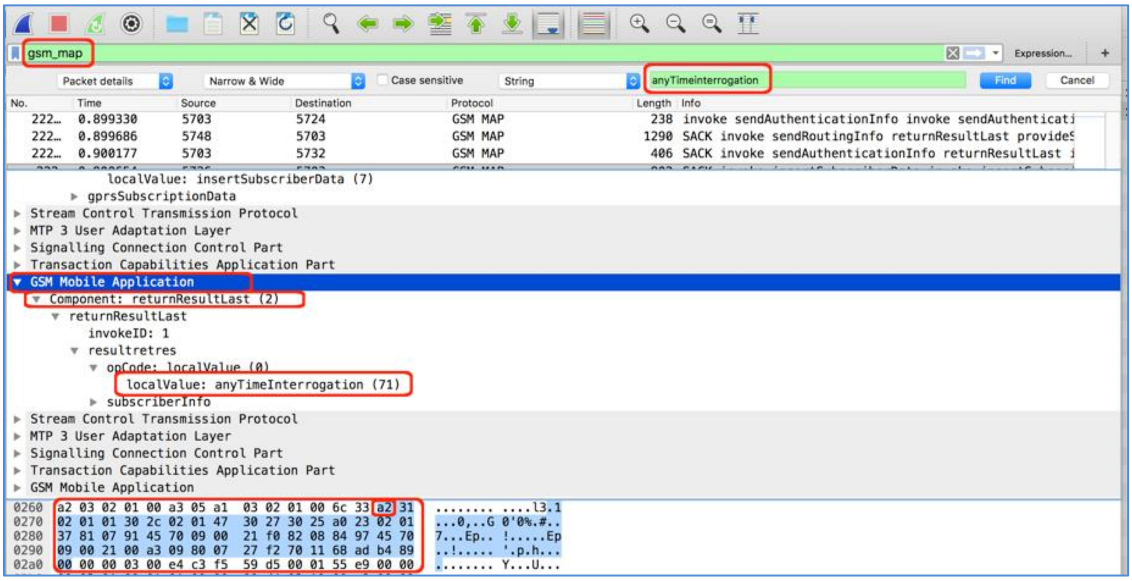
```
include $RULE_PATH/ss7.rules
```

6.3. The SS7.rules.

When installing Snort, bring hundreds (or thousands) of rules classified by families, which in the standard use of this tool, it is always advised to keep up to date, as new rules are published and adjusted every day.

In our case, the task is different, since we are not interested at all in analyzing the rules for "http", "telnet", "ssh", etc. We must focus specifically on SS7/Sigtran. For this Snort, with its powerful flexibility and parameterization, offers us the possibility of creating our own personalized rules to what we want and in our opinion, this may be one of the greatest qualities that this software has. In this section we especially recommend a detailed study of the "**Snort Users Manual**" because it is almost infinite the amount of possibilities that it offers us.

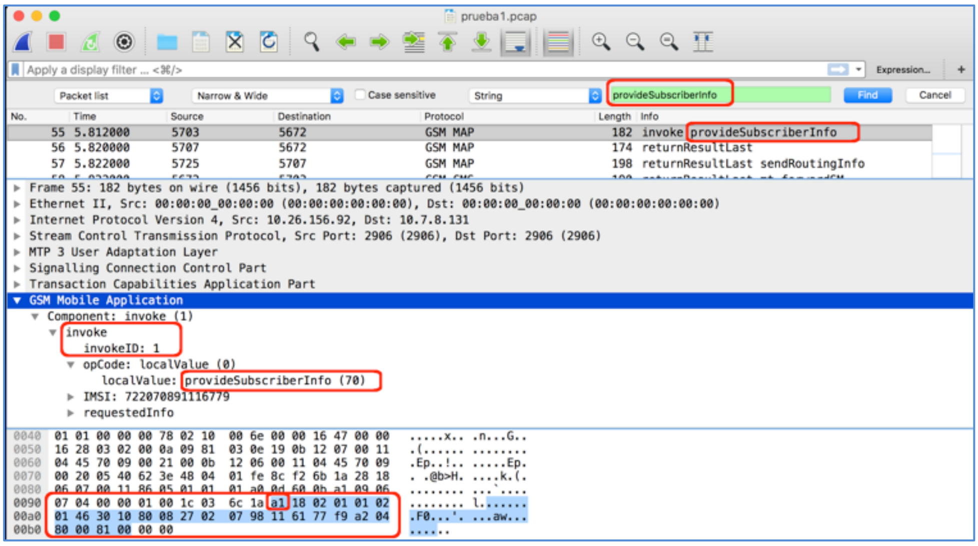
Before entering fully into these rules, it is important that we go back a bit, on the issues already seen with Wireshark and, maintaining our coherence with respect to the sequence of analysis, let's return the "**attack No. 1**" on the parameter "**antTimeInterrogation (ATI)**".



When we start with this parameter, we just present the previous image and we emphasized that hexadecimal value "**a2**" that is highlighted in **red** in the lower part of the capture (*that is, in the lower window, of "hexadecimal" values of Wireshark*). On that occasion we mentioned that when the frame at the "**gsm_map**" level is a "request" (*invoke*) this value corresponds to "**a1**" and when it is a "response" (*returnResultLast*), it corresponds to "**a2**" (*as in the image*).

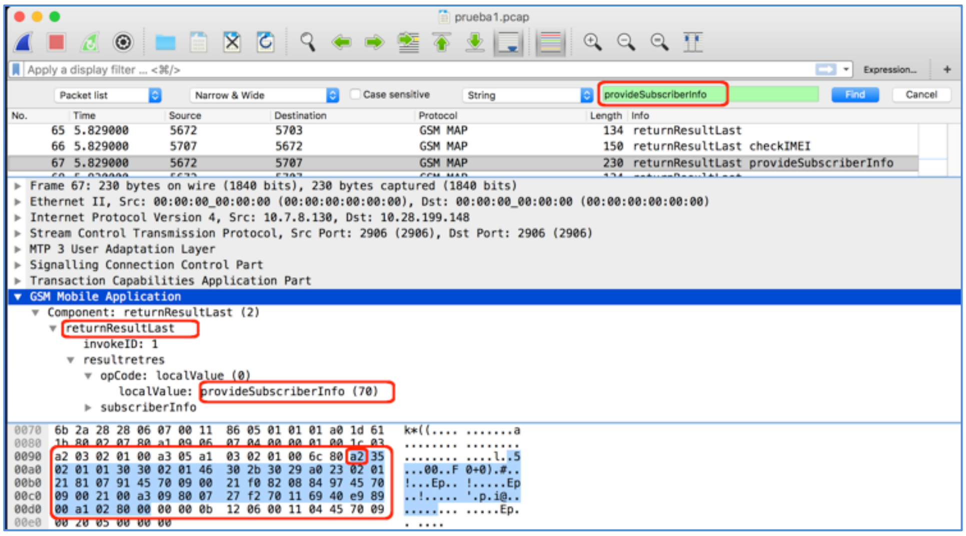
In turn, also in the previous image, we can see that we have also highlighted in **red**, and in the same lower window, ALL values in hexadecimal. This we have done, because by studying them will be our starting point for the design of our own SS7 rules, which is the next thing to try.

Here are some more examples, to develop where we want to go.



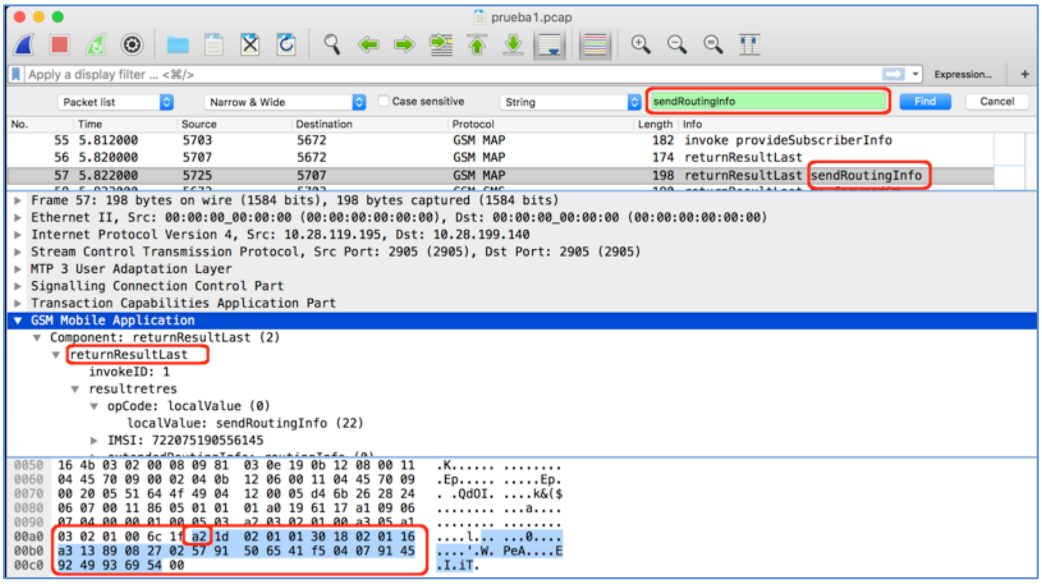
Following with the parameters that are reasons for attacks, as we presented in point 5. In the previous image, we can see that we are applying a "visualization filter" to only present the MAP parameter: "**providerSubscriberInfo**", as we mentioned in the previous paragraph, in this case it is an "**Invoke**", that is to say "**request**" and therefore its initial value is "**a2**" (we have highlighted all this in **red**).

If we want to analyze the same, but for a "**providerSubscriberInfo response** "we can appreciate it in the following image.



Regardless of this first "**a1**" or "**a2**" value of the "**gsm_map**" protocol, let's advance a bit more by paying attention to the rest of the hexadecimal values (*which we reiterate, are the closest and lowest level representation of the "**bits**" that actually circulated through that network infrastructure*).

To reaffirm this last concept we are going to present one more image, but this time on another of the parameters motives of our list of attacks, in the following image, we can see a frame that contains "**sendRoutingInfo response**".



Again on the previous image, pay attention to the hexadecimal values of the lower window that we have highlighted in **red**.

If we analyze in this way each of the parameters that we want to analyze, we can create the "**hexadecimal traffic patterns**" that uniquely identify the occurrence of this parameter. That is, we will be working at the lowest level of the layer model, with which there is NOTHING that can be ignored, because any other type of analysis through the different protocols will always be "packaged" at levels higher than This we are "looking" at us.

The domain of information that circulates at the level of "bits" opens the game to any type of "**detection**", and now we can start to create the rules that we want with Snort.

Next, we present some of the parameters that we have come to identify with a high degree of certainty.

NOTE: These associations are the initial focus of several hours of study, but they do not claim to be 100% accurate, let alone complete (*hundreds of hours of work are still needed*).

One of the future lines of research that we would love readers to join, is this:

- **of traffic patterns**
- **Creation and adjustment of new rules ss7/Sigtran (*ss7.rules*)**

From the work done so far, we can present the following "**hexadecimal traffic patterns**":

a2 ** 02 01 01 30 2c 02 01 47 30 27 (MAP anyTimeinterrogation)

-----> Component: returnResultLast

(MAP anyTimeinterrogation) NO tenemos capturas

a2 ** 02 01 01 30 ** 02 01 46 30 (MAP provideSubscriberInfo)

-----> Component: returnResultLast

a1 ** 02 01 01 02 01 46 30 10 80 (MAP provideSubscriberInfo)

-----> Component: invoke

02 01 01 02 01 2e 30 48 84 (MAP mo-forwardSM)

a1 ** 8b 02 01 00 02 01 2c (MAP mt-forwardSM)

a1 6a 02 01 01 02 01 16 30 62 (MAP sendRoutingInfo)

a1 ** 02 01 01 02 01 2d 30 15 (MAP sendRoutingInfoForSM)

-----> Component: invoke

a2 ** 02 01 01 30 1a 02 01 2d 30 15 (MAP sendRoutingInfoForSM)

-----> Component: returnResultLast

a2 ** 02 01 01 30 0e 02 01 02 30 09 04 07 (MAP updateLocation)

-----> Component: returnResultLast

a1 ** 02 01 01 02 01 02 30 26 04 08 (MAP updateLocation)

-----> Component: invoke (invokeID: 1)

a1 ** 02 01 01 02 01 07 30 (MAP InsertSubscriberData)

-----> Component: invoke (invokeID: 1)

a1 ** 02 01 02 02 01 07 30 (MAP InsertSubscriberData)

-----> Component: invoke (invokeID: 2)

a1 ** 02 01 03 02 01 07 30 (MAP InsertSubscriberData)

-----> Component: invoke (invokeID: 3)

a2 ** 02 01 01 30 25 02 01 07 30 (MAP InsertSubscriberData)

-----> Component: returnResultLast (InvokeID: 1)

a2 ** 02 01 02 30 09 02 01 07 30 (MAP InsertSubscriberData)
-----> Component: returnResultLast (InvokeID: 2)

a2 0e 02 01 05 30 09 02 01 07 30 (MAP InsertSubscriberData)
-----> Component: returnResultLast (InvokeID:5)

a1 ** 02 01 01 02 01 08 30 (MAP deleteSubscriberData)
-----> Component: invoke (invokeID: 1)

a1 ** 02 01 01 02 01 03 a3 0d (MAP cancelLocation)

-----> Component: invoke (invokeID: 1) (cancelLocation)

a1 .{2,4} 02 01 00 02 01 00 30 (CAMEL-v2 o MAP initialDP)
-----> Component: invoke (invokeID: 1) (cancelLocation (3))

With the recognition of these hexadecimal values, we can start with the creation of our own detection rules for Snort, which we will call "**ss7.rules**" (*as we have presented in our configuration file model: "**snort.conf**")* .

We will not go into explanations of what Snort's rules are like (*again, we recommend reading the "**Snort Users Manual**"*), we only mention that a Snort rule must have a "header" (*previous to parentheses*) and a "body" (*enclosed in parentheses*).

Let's start to "create" new rules.

Example 1:

We are going to apply one of the hexadecimal patterns that we have just presented, for example:

a1 6a 02 01 01 02 01 16 30 62 (MAP sendRoutingInfo)

If we wanted to start with a rule that can detect the occurrence of these patterns in hexadecimal, we could start with:

```
alert ip any any -> any any (msg:"MAP - sendRoutingInfo";
content:"|a16a020101020116|"; priority:1; sid:1000001; rev:1;)
```

What are we saying here?

1) That generates an alert: alert

2) For everything that follows the protocol ip: ip

3) Whose source is any ip/network: any

4) Whose origin is any port: any

5) That only goes in a sense "**->**" (*although in this case it does not have logic, then all "any"*)

6) Whose destination is any ip/network: any

7) Whose destination is any port: any

8) That if there were any occurrence, that is, if this rule were applied to any frame, it would generate a message with the following content: MAP - sendRoutingInfo

9) That the rule "skip" only when it finds this sequence in hexadecimal: content: "| a16a020101020116 |".

10) What gives top priority: priority: 1

11) That the identifier of this rule is: sid: 1000001 (*Snort advises that*

when creating local rules, use an ID greater than 1,000,000)

12) What is the first revision of it: rev: 1

Of course we are presenting an extremely basic rule to get started.

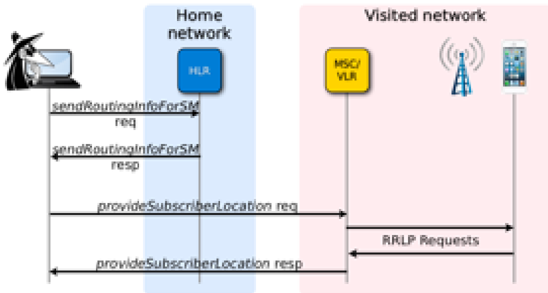
Example 2:

To improve it a bit, we could start to adjust its design, for example with one of its "any".

If we return to the topic of our attacks, for example the **attack No. 2.** Location Services (LCS) (use of emergency location).

Again on MAP, two steps are carried out:

a. The intruder sends **sendRoutinginfoForSM** request (to the HLR), which responds with **sendRoutinginfoForSM response.**



In this particular case, we see that this parameter can begin to analyze it in its "**request**" and its "**response**". A proposal can be to evaluate if we have in our traffic screens some response frame, initially containing "**sendRoutinginfo**".

Given this hypothesis, it is clear that the **HLR** must send it to an "**External Net**", to adjust this configuration on the previous rule, we can then do it in the following way:

alert ip **$HLR-HSS** any -> **!$HOME_NET** any (msg:"MAP - sendRoutingInfo"; content:"|a16a020101020116|"; priority:1; sid:1000001; rev:1;)

What are we saying now?

1) That this rule is only activated when the IP / network coincides with the variable that we have defined in the previous point (*within our example of the "**snort.conf**" file*) as **var HLR-HSS**, and that is why we invoke it with the sign "$" ahead: **$ HLR-HSS**
2) That only this rule is activated when the IP/network, does NOT match ("!") With the variable that we have defined in the previous point (*within our example of the file "**snort.conf**"*) as **var HOME_NET**: **!$ HOME_NET**

Para mejorarla un poco, podríamos empezar a ajustar su diseño, por ejemplo con alguno de sus "any".

Si retomamos el tema de nuestros ataques, por ejemplo el **ataque Nro 2.** Location Services (LCS) (empleo de la Localización de emergencia).
Nuevamente sobre MAP, se realizan dos pasos:

a. El intruso envía **sendRoutinginfoForSM request** (al HLR), el cual responde con **sendRoutinginfoForSM response**

Example 3:

If we continue paying attention to the **attack No 2** presented in the previous example, we will see that in reality, we are not looking for the parameter "**sendRoutinginfo**", but we must be even more precise and look for "**sendRoutingInfoForSM**" (***SM*** *comes for "Short Messages*"), if we go back to the "hexadecimal patterns" that we presented earlier we can see the following (*for request and response*):

a1 ** 02 01 01 02 01 2d 30 15 (MAP sendRoutingInfoForSM)

-----> Component: invoke

a2 ** 02 01 01 30 1a 02 01 2d 30 15 (MAP sendRoutingInfoForSM)

-----> Component: returnResultLast

When we represent (*in our text*) a hexadecimal value with "**" we try to reflect that this hexadecimal pair can adopt any value, since we have proven this in the study of several occurrences of this parameter.

In the body of a Snort rule, if we work with "pure" hexadecimal values, they must be enclosed between vertical bars "| ... .. |" and they **DO NOT** support the "wild card" "**".

Fortunately this wonderful tool offers us a solution, the use of expressions "p**cre**" (*Perl Compatible Regular Expressions*).

```
alert ip $HLR-HSS any -> !$HOME_NET any (msg:"MAP -
sendRoutingInfo";
pcre:"/\xa2.{1}\x02\x01\x01\x30\x1a\x02\x01\x2d/";
priority:1; sid:1000001; rev:1;)
```

What are we saying now?

That instead of analyzing a specific "**content**", apply a "**pcre**" expression with hexadecimal values: "**\x**" and that, after the value "**a2**" consider one and only one "ASCII character" represented by a pair of hexa numbers ". {1}" (*if we had wanted exactly two, we should have set* ***". {2}"****, if it was any value between 1 to 5 characters:* ***". {1-5}"*** *etc).*

Example 4:

In order not to further extend each of the aspects to be considered in relation to the creation of Snort rules (*an issue that we insist should be seriously studied from the "****Snort Users Manual****"),* let us analyze in more depth the real rules on which we are working on SS7 / Sigtran traffic.

Attack Category 1: an HLR respond to EXT_NET with "**provideSubscriberInfo resp**"

The first commented rules allows to verify that there is or not traffic "**provideSubscriberInfo_resp**"

If you need to verify it, you should remove the comment "#"

The second rule (without comment) is the one that detects the occurrence of this attack 1

```
#provideSubscriberInfo_resp ip any any -> any any (msg: "MAP - provideSubscriberInfo_resp -  Attack No 1""; pcre:"/\xa2.{1}\x02\x01\x01\x30.{1}\x02\x01\x46\x30/"; sid:1000010;)
```

```
provideSubscriberInfo_resp ip $HLR-HSS any -> !$HOME_NET any (msg: "MAP - provideSubscriberInfo_resp -  Attack No 1"; pcre:"/\xa2.{1}\x02\x01\x01\x30.{1}\x02\x01\x46\x30/"; sid:1000011;)
```

What are we saying now?

As we can see, we are already creating rules that are related to our study and presentation of the fifteen attack categories. In this case we are generating a clear message: **msg: "MAP - provideSubscriberInfo_resp - Attack No. 1".**

But the aspect that we want to emphasize is that this new rule no longer starts with "**alert**" like the previous examples, this time its first word is "**provideSubscriberInfo_resp**".

If we go back, when we present the configuration file "**snort.conf**", in the "**outputs**" section, at the end of it, we comment that they can be "customized" and specifically in our example "**snort.conf**" file We exposed the following output:

```
# Output in "pcap" format for Attack 1 detected
```

(We can define our own Output format based on a certain rule, it will be seen more clearly after we present the "Snort rules").

```
ruletype provideSubscriberInfo_resp {
    type alert
    output log_tcpdump: Atq1-provideSubscriberInfo_resp.cap
    }
```

In this Example 4, we are precisely, relating this rule with a specific output that we have created ourselves and called "provideSubscriberInfo_resp", we want it to behave like "**alert type**": type alert and that its output is in "log_tcpdump" format "(*ie ".cap")* and with the name: Atq1-provideSubscriberInfo_resp.cap.

Example 5:

We present below other rules that respond to the same previous format and are the ones **we are using in SS7/Sigtran networks in production with very good results.**

Attack Category 2: an MSC to respond an EXT_NET with "provideSubscriberLocation resp"

For this attack we have not yet obtained any traffic capture containing this parameter

Attack Category 3 (DoS): from EXT_NET to MSC with any of the following parameters "insertSubscriberData req", "DeleteSubscriberData req" or "cancelLocation req"

In all cases, the first commented rule "#" allows to verify that there is traffic "provideSubscriberInfo resp"

If you need to verify it, you should remove the comment "#"

The second rule (without comment) is the one that detects the occurrence of this attack 1

```
# insertSubscriberData_req ip any any -> any any (msg: "MAP - insertSubscriberData_req - Attack No 3A"; pcre:"/\xa1.{1}\x02\x01.{1}\x02\x01\x70\x30/"; sid:1000031;)

insertSubscriberData_req ip !$HOME_NET any -> $MSC  any (msg: "MAP - insertSubscriberData_req - Attack No 3A"; pcre:"/\xa1.{1}\x02\x01.{1}\x02\x01\x70\x30/"; sid:1000032;)

# DeleteSubscriberData_req ip any any -> any any (msg: "MAP - DeleteSubscriberData_req - Attack No 3B"; pcre:"/\xa1.{1}\x02\x01\x01\x02\x01\x08\x30/"; sid:1000033;)

DeleteSubscriberData_req ip !$HOME_NET any -> $MSC any (msg: "MAP - DeleteSubscriberData_req - Attack No 3B"; pcre:"/\xa1.{1}\x02\x01\x01\x02\x01\x08\x30/"; sid:1000034;)

#cancelLocation_req ip any any -> any any (msg: "MAP -
```

cancelLocation_req - Attack No 3C";
pcre:"/\xa1.{1}\x02\x01\x01\x02\x01\x03\xa3/";
sid:1000035;)

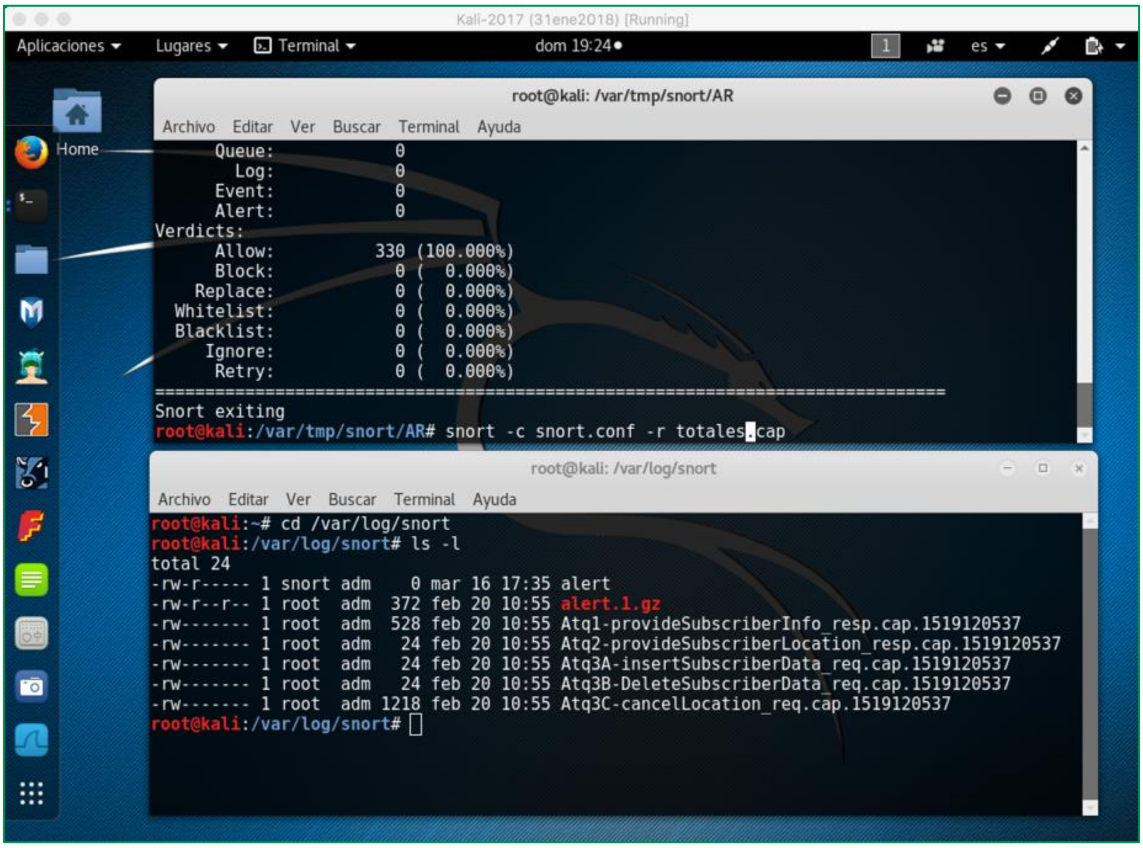
cancelLocation_req ip !$HOME_NET any -> $MSC any (msg: "**MAP - cancelLocation_req - Attack No 3C**";
pcre:"/\xa1.{1}\x02\x01\x01\x02\x01\x03\xa3/";
sid:1000035;)

Finally, to not dwell on it, we present how Snort can be launched to make use of this configuration example "**snort.conf**" and the "**ss7.rules**" that we have just presented.

The format to launch it from a console would be:

 # snort -c snort.conf -r captur_to_employ.pcap

Here is a screenshot of our virtual machine "**Kali**" where you can see the execution and the results of it.



As we can see in the previous image, in the upper console of the same one, the concrete command that has just been executed is presented, and in the lower window the results of the outputs with the format that we have decided to assign to it.

In this lower window, it can be clearly seen that the empty "alert" file has been generated (*by our **log_null** configuration*), and then, each of the outputs in the "**.cap**" format of the **three attacks** whose rules we have just presented.

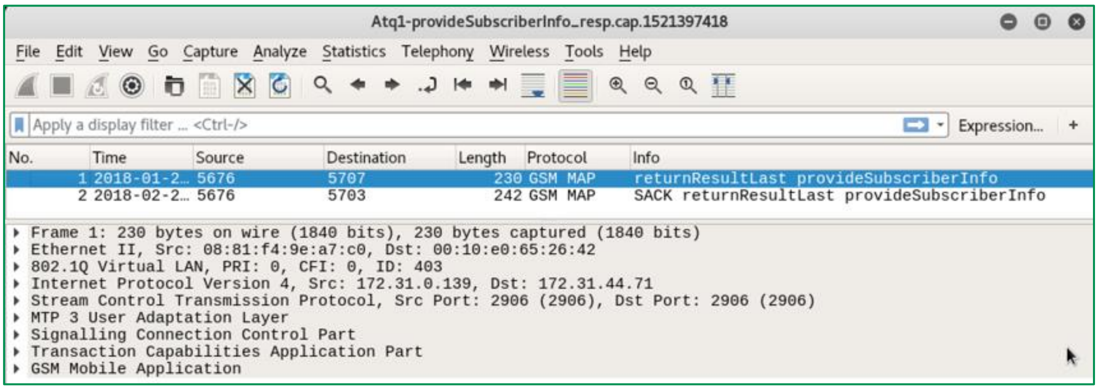
Those that have a size of 24 bytes only contain the title, that is, they are empty (*this attack pattern has not been found*), but specifically the files:

- **Atq1-provideSubscriberInfo_resp.cap**
- **Atq3C-canelLocation_req.cap**

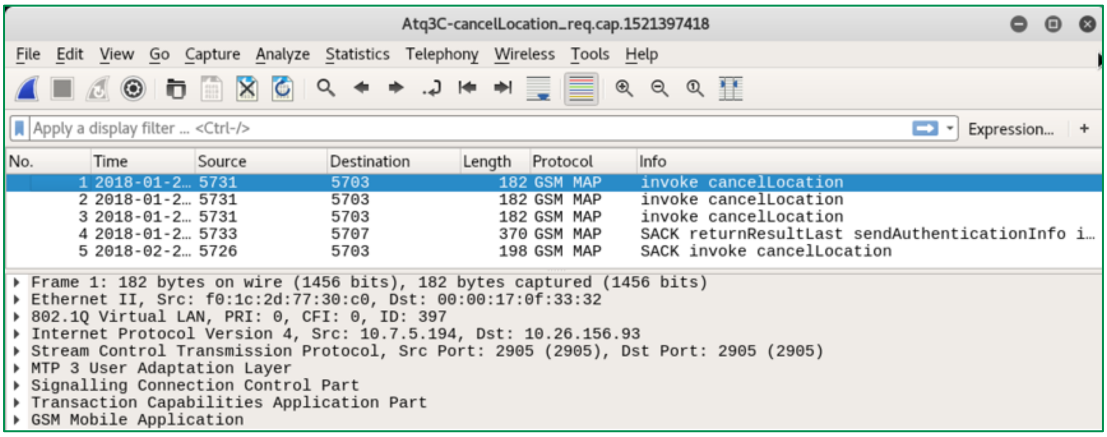
These two **YES** do have frames that have been stored because they comply with our search pattern.

We can not say that there are no false positives, but it is true that we now have a minimum series of frames of **435,017** of our initial work file on which we assemble all the "small initial captures" (**captur_complete.pcap**) of 161.7 MB in size

Specifically, if we open with Wireshark, both results of the launch of Snort in the virtual machine, the frames generated after applying our **ss7.rules** are:



As the title of the previous image indicates, it is the final result on **Attack No. 1**, and contains only two frames.



As the title of the previous image indicates, it is the final result on **Attack No.**

3C, and contains only five frames.

7. Other additional tools.

Below we present some tools that have been useful in this work.

7.1. MATE (*Meta Analysis and Tracing Engine*) for Wireshark.

MATE is a Wireshark plugin that extracts data from the tree of frames and then, using that information, tries to group them according to how this module has been configured. The language used is the "canonical" layer model, calling "**PDU**" (Protocol Data Unit) to the information sets of each level to be treated, generating a "tree" of PDUs with the fields that we have filtered, offering many options that can be useful.

https://wiki.wireshark.org/Mate

Specifically MATE, is based on a file in which we can configure in a simple way the different fields that we are interested in grouping of the frames of any type of captures.
In the URL that appears above, within the documentation that it offers us, it presents an example file called "tcp.mate", which we can see below.

```
Pdu tcp_pdu Proto tcp Transport ip {
      Extract addr From ip.addr;
      Extract port From tcp.port;
                    Extract tcp_start From tcp.flags.syn;
                    Extract tcp_stop From tcp.flags.reset;
                    Extract tcp_stop From tcp.flags.fin;
};

Gop tcp_ses On tcp_pdu Match (addr, addr, port, port) {
   Start (tcp_start=1);
            Stop (tcp_stop=1);
};

Done;
```

In it, a new "**PDU**" whose name is "**tcp_pdu**" is being generated that will work on the "**TCP**" protocol and interpret everything that is "transported" over the "**IP**" protocol, from there it requests to "extract" the fields "**ip-addr**", "**tcp.port**", "**tcp.flags.syn**", etc. And then "group" them by means of a "Pdu Group (Gop)" called "**tcp.ses**". Again, it is not our intention to develop a course on MATE (*please, if you wish to deepen it, you have all the documentation in the URL that was indicated at the beginning*). In these lines, we only wish to present the same, to give some example of how we have used it, and above all to awaken the reader's interest in it.

In our case, for example, we want to work with the protocol "**SCCP**" and "**GSM_MAP**", for that, then we must generate our own configuration file with the fields that we want to group, for this we can do it initially as presented below.

```
Pdu ip_pdu Proto ip Transport ip {
    Extract ip_fuente From ip.src;
        Extract ip_destino From ip.dst;
            };

Pdu sccp_pdu Proto sccp Transport ip {
        Extract sccp_clg_dig From sccp.calling.digits;
        Extract sccp_clg_ssn From sccp.calling.ssn;
        Extract sccp_cld_dig From sccp.called.digits;
        Extract sccp_cld_ssn From sccp.called.ssn;
        Extract gsm_map_UpdateLocation From gsm_old.localValue;
        Extract gsm_map_msisdn  From gsm_map.ch.msisdn;
        Extract gsm_map_locationnumber.digits
                    From gsm_map.locationnumber.digits;
        };
Done;
```

The first thing we want to highlight, is that this plugin uses the same format as the "visualization filter" of Wireshark, therefore, any parameter that we want to configure, we can look for it in "**Expression**" from the "Wireshark" filters and copy its Format.

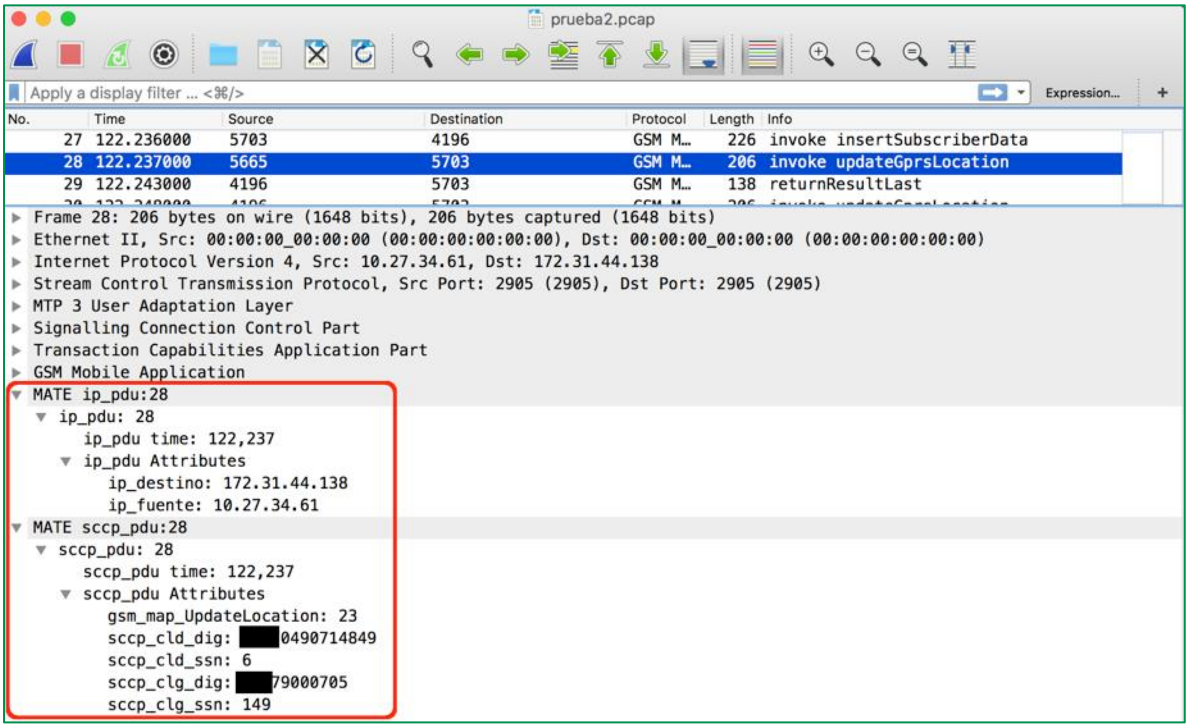
Beyond re-explaining what we are doing, let's see the results that we would obtain with this configuration file, which we have called "**sccp.mate**".

To launch this plugin, we can do it by command line, or from the same graphical interface of Wireshark, let's see the first case.

```
#wireshark -o "mate.config: sccp.mate" -r  test2.pcap
```

With the above command, we tell you to run Wireshark, overwriting its default configuration **("-o")** using the matte configuration contained in the "**sccp.mate**" file and this is done by reading **("-r")** the capture "**test2.cap**".

Once this command line has been executed, the graphical interface of Wireshark will open and it will offer us new fields, as we can see in the following image.



As you can see in the previous image highlighted in red, this new group of parameters now appears, for each frame that applies, in which it presents the information that we have just requested in our file "**sccp.mate**". We have deleted the digits of the Country.

All the fields presented in this image have already been developed in previous points, we invite the reader to review these concepts within this text.

It is not worthwhile to continue developing more concepts and possibilities that MATE offers us, because the idea that we try to present in these brief final lines is to also consider other tools that together offer us more filtering, visualization, selection, etc. It is the reader who has the freedom of action to take advantage of them in the best way he considers appropriate, integrate them into what has already been seen, take advantage of them to have another point of view or strengthen them by programming different actions to improve this work.

7.1. Tshark.

In cases where it may be useful NOT to use the graphical interface of Wireshark, and in particular because of the power offered by the command line to execute simple programs, this software (Wireshark) also offers this command "**tshark**" that operates under the same logic and allows us to make use of almost all Wireshark options and filters.

There is a lot of information about it on the Internet, as are your options clear enough if we write "#**tshark -help**"

Without this being a "tshark" course, in these lines we only wish to highlight the importance of taking this command into account, because as we will see below it offers us an almost unlimited possibility of analysis. If this capacity is added to simple "**scripts**", for example in "**bash**" programming, the results can be excellent and the only frontier will be the creativity and imagination of the person who develops them. We present below some simple examples, where we can appreciate the application of the same filters that we have used in the "Wireshark" section of this same document.

```
sh-3.2#  tshark -r  test1.pcap
   1  0.000000     5707 → 5672    GSM SMS 306 invoke forwardSM
   2  1.716000     5710 → 5703    GSM MAP 186 invoke readyForSM
   3  1.777000     5707 → 5732    GSM MAP 186 invoke readyForSM
   4  2.464000     5707 → 5710    GSM SMS 270 invoke forwardSM  (Short Message fragment
4 of 4)
   5  2.604000     5707 → 5710    GSM SMS 206 invoke forwardSM
   6  2.683000     5703 → 5710    GSM SMS 306 invoke forwardSM
   7  2.829000     5707 → 5710    GSM SMS 322 invoke forwardSM  (Short Message fragment
2 of 4)
   8  3.592000     5707 → 5710    GSM SMS 218 invoke forwardSM
   9  3.617000     5707 → 5710    GSM SMS 298 invoke forwardSM
  10   3.681000     5703 → 5710    GSM SMS 322 invoke forwardSM  (Short Message fragment
1 of 4)
  11……………( continues until the last frame)
```

As we can see in the previous command "<u>read</u>" the capture "**test1.pcap**".

Next, we present the application of the same visualization filter that we use in the "Wireshark" section, whose result presents us with total clarity, which are the frames that include this search (in this case we are filtering. **"gsm_old.localValue=="** that implies "updateLocation").

```
#tshark -Y gsm_old.localValue==2 -r  test1.pcap
  18  5.218000     5748 → 5707     GSM MAP 210 invoke updateLocation
  70  5.832000     5707 → 5748     GSM MAP 150 returnResultLast updateLocation
  79  5.933000     5703 → 5732     GSM MAP 206 invoke updateLocation
  83  6.151000     5710 → 5703     GSM MAP 210 invoke updateLocation
  90  6.225000     5710 → 5703     GSM MAP 210 invoke updateLocation
  92  6.229000     5707 → 5732     GSM MAP 210 invoke updateLocation
  93  6.242000     5710 → 5703     GSM MAP 210 invoke updateLocation
  95  6.253000     5703 → 5732     GSM MAP 210 invoke updateLocation
  99  6.306000     5703 → 5732     GSM MAP 206 invoke updateLocation
 100  6.313000     5703 → 5732     GSM MAP 210 invoke updateLocation
```

Next, we can see another example where we "<u>concatenate</u>" more than one visualization filter and whose result is a single occurrence of it.

```
#tshark -Y "(gsm_old.localValue==2) and (sccp.calling.ssn==6)" -r  prueba1.pcap
  70  5.832000     5707 → 5748     GSM MAP 150 returnResultLast updateLocation
```

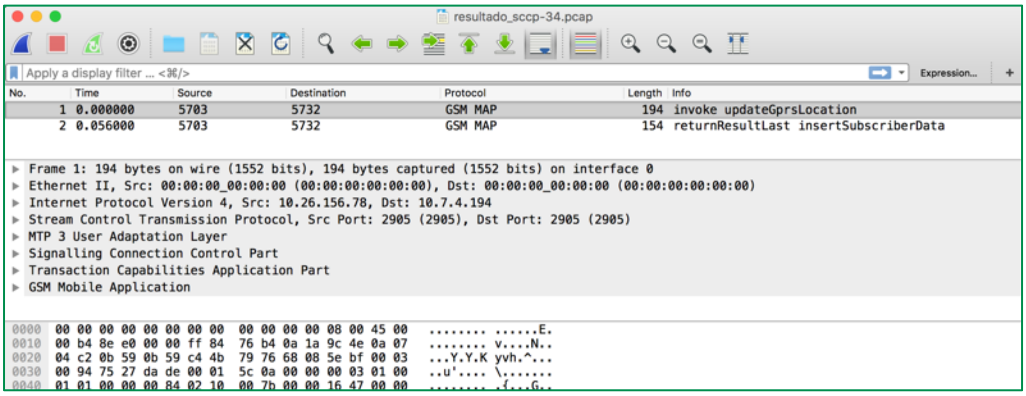
Next, we repeat the same example that we put in "Wireshark" to identify frames that come from any "External Net" outside of Spain, analyzing its "**SCCP**" address.

```
#tshark -Y "sccp.calling.digits matches 34 and  not sccp.called.digits matches 34" -r prueba1.pcap
   84   6.151000        5703 → 5732       GSM MAP 194 invoke updateGprsLocation
   89   6.207000        5703 → 5732       GSM MAP 154 returnResultLast insertSubscriberData
```

Finally, we can see that the previous filter, as with Wireshark, we can send it to a "**-w**" (write) output, in our case we have called it "**result_sccp-34.pcap**".

```
#tshark -Y "sccp.calling.digits matches 34 and not sccp.called.digits matches 34" -r test1.pcap -w result_sccp-34.pcap
```

If we open it with Wireshark we can verify that it has been saved with this format and that of course they are perfectly compatible.



7.2. Unify frames (mergecap).

An important command that we have already mentioned and we emphasize here is, if several "**mergecap**" captures are received, with this, they can be unified to treat all of them as one. The basic format that we can apply to join several captures "pcap" is:

#mergecap *.pcap -w nombre_salida.pcap

7.3. Capture information (**capinfos**).

This command, which is usually integrated in the different Linux distributions, and ultimately is part of the family "**tcpdump**" or "**libpcap**", as the name implies, gives us information of the files in "**.cap** " format. (and its variants: .pcap, .pcapng, etc.).

In our case it is quite practical for us to make a first glance at any capture we have, and in particular, to know what type of distribution of protocols corresponds to it.

Let's see some simple examples.

#capinfos -u prueba1.pcap

File name: prueba1.pcap

Capture duration: 6,313000 seconds

It quickly indicates the duration of a complete capture.

#capinfos -i prueba1.pcap

File name: prueba1.pcap

Data bit rate: 29 kbps

It tells us quickly the speed with which the data was captured.

#capinfos -c totales-MAP.pcap

File name: totales-MAP.pcap

Number of packets: 435 k

It tells us quickly how many packages of that capture.

For more detail of all the options offered by this command, you can write it without options and all of them will be displayed (**#capinfos**)

8. Conclusions

Throughout this text, we have tried to present the problem that we currently have with SS7/Sigtran in all the telephony operators in the world.

We analyze the texts and research on the different types of attacks that already exist in practice and that can cause a high impact on these architectures.

Let us make it clear that the only way to deal with these vulnerabilities is to understand and analyze the flows of "bits" that circulate through these signaling networks, without this work, we would be operating blindly.

We were developing a work methodology that allows us to identify and detect the potential occurrences of these "traffic patterns", and then be able to adopt the measures that best fit our architecture.

We present different techniques and tools to carry out this work and we advance with concrete examples on real traffic SS7/Sigtran.

We offered solutions to this problem in an understandable way and above all by using ALL tools under "**Open Source**" licenses, so that we do not have to resort to applications or payment software.

We wanted to disseminate the work in the state it is in, knowing perfectly that it is only a starting point, robust, but in an initial state. We take this decision because we are aware that as in all development based on "Open Source", the sum of efforts is what really enhances it, so we prefer to share it now as an invitation to new developers and contributions that allow us to mature and lead to production this way of working.

As a closing of this document, we want to express that for us the greatest satisfaction would be to find an echo of this methodology and be able to see it "grow" with the contributions of anyone who wants to get on this car.

Madrid, march 2018

REFERENCES

[i] Engel, T. Locating Mobile Phones using Signaling System #7. [Online]. Available: https://events.ccc.de/, Accessed 07.11.2015

Engel, T. December 2014. SS7: Locate. Track. Manipulate. [Online]. Available: https://media.ccc.de, Accessed 22.10.2015.

31c3-ss7-locate-track-manipulate.pdf - SS7: Locate. Track. Manipulate. Tobias Engel <tobias@ccc.de>

[ii] Langlois, P. 2010. Getting in the SS7 kingdom: hard technology and and disturbingly easy hacks to ge (Engel)t entry points in the walled garden. [Online]. Available: http://www.hackitoergosum.org/2010/ HES2010-planglois-Attacking-SS7.pdf, Accessed: 25.11.2015.

[iii] Nohl, K. December 2014. Mobile self-defence. [Online]. Available: https: //media.ccc.de, Accessed 22.10.2015.

1783_101228.27C3.GSM-Sniffing.Nohl_Munaut.pdf - GSM Sniffing - Karsten Nohl

[iv] Vauboin, P.-O. & Oliveira, A. D. April 2014. Worldwide attacks on SS7 network.

[v] Langlois, P. - bh-eu-07-langlois-ppt-apr19.pdf - SCTPscan - Finding entry points to SS7 Networks & Telecommunication Backbones

[vi] diameter_research.pdf - NEXT GENERATION NETWORKS, NEXT LEVEL CYBERSECURITY PROBLEMS (2017).

[vii] FRHACK2009_Attacking-SS7_Langlois.pdf

[viii] Kamwendo, B. - Research Report.pdf - University of the Withwatersrand – Johannesburg

[ix] Hacking en redes SS7 ~ Security By Default.pdf - www.Securitybydefault.com

[x] Jensen, K. Junio 2016. Improving SS7 Security Using Machine Learning Techniques. Master's Thesis. Master of Science in Information Security (KJensen_2016.pdf).

[xi] GSMA - FS.11 - SS7 Interconnect Security Monitoring Guidelines - Version 1.0 (19 November 2015).

Guide to 3G security v130 Draft-Changed to cover MAP-SEC

3GPP TS 29.002 V15.2.0 (2017-12) 3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Mobile Application Part (MAP) specification (Release 15)

ETSI TS 100 974 V7.15.0 (2004-03) - Digital cellular telecommunications system (Phase 2+); Mobile Application Part (MAP) specification (3GPP TS 09.02 version 7.15.0 Release 1998)

ETSI TS 129 002 V14.4.0 (2018-01) - Digital cellular telecommunications system (Phase 2+) (GSM); Universal Mobile Telecommunications System (UMTS); Mobile Application Part (MAP) specification (3GPP TS 29.002 version 14.4.0 Release 14)

GSMA - RIFS62_03 CR1005 to FS11 v3_2 DRAFT v0_2.docx - SS7 Interconnect Security Monitoring and Firewall Guidelines CR1005 to Version 3.2 DRAFT v0.2