Security analysis of the LTE-Uu interface using open source software

Domonkos P. Tomcsányi

Pázmány Péter Catholic University
Faculty of Information Technology and Bionics
Computer Engineering BSc

Supervisor: Dr. Márton Csapodi
Advisor: László Szűcs

Budapest, 2016

Pázmány Péter Katolikus Egyetem
Információs Technológiai és Bionikai Kar

Szakdolgozat témabejelentő nyilatkozat

Hallgató

Név: Tomcsányi Domonkos Pál	**Neptun kód:** G1RGYR
Szak: Mérnökinformatikus BSc	

Témavezető

Név: Szűcs László
Intézet, cég: Nokia
Beosztás, tudományos fokozat: Senior Security Specialist, MSc

Belső konzulens neve: Dr. Csapodi Márton

Dolgozat

A dolgozat címe (magyarul): Az LTE-Uu interfész biztonsági vizsgálata nyílt forráskódú szoftverek felhasználásával
A dolgozat címe (angolul): Security analysis of the LTE-Uu interface using open source software

A dolgozat témája

LTE is currently the most popular choice among operators to provide mobile broadband services to customers. Its security is based on concepts first introduced in UMTS and currently considered strong, because it addresses all the main issues GSM had: integrity protection against fake base stations and stronger encryption against over-the-air eavesdropping. However, to encourage a more rapid deployment of 4G implementing most of these security features were made optional.

Ultimately this led to baseband software versions that don't even implement these optional features making it harder for operators to strengthen the security of their radio infrastructure later. Moreover, some fail to correctly implement even the mandatory, reduced set of security protocols making them and the device they are built-in vulnerable to a wide variety of attacks.

The goal of this research is to create a framework for testing such implementation errors using (if possible) open source components. Areas of research should include messages with invalid formatting, reserved or not used values and response codes, and also full scenarios (complete flows of messages) that should be considered invalid. All uncovered vulnerabilities shall be reported responsibly to the manufacturers to allow creation and deployment of patches.

A hallgató feladatai:

- Summarize the history of mobile networks and the development of LTE
- Describe an LTE network (components, protocols, roles)
- Analyze LTE - Layer 3 specifications, especially NAS (3GPP - TS 24.301)
- Describe security test methods, decide which ones to use and create test cases
- Create an overview about the currently available open source tools in this topic
- Design and implement the framework necessary for doing the security tests (hardware, software, equipments to be tested)
- Carry out the planned security tests
- Evaluate results and responsibly report any findings to vendors or 3GPP if necessary

A témavezetést vállalom.

A témavezető aláírása

A hallgató témavezetését belső konzulensként vállalom.

Belső konzulens aláírása

Kérem a szakdolgozat témájának jóváhagyását.

Budapest, ...

A hallgató aláírása

A szakdolgozat témáját az Információs Technológiai és Bionikai Kar jóváhagyta.

Budapest, ...

Dr. Szolgay Péter

dékán

A hallgató a konzultációkon részt vett és a kiírásban foglalt feladatokat teljesítette.

Budapest, ...

A témavezető aláírása

A dolgozat a TVSZ 3. sz. mellékletében foglalt tartalmi és formai követelményeknek megfelel.

Budapest, ...

Belső konzulens aláírása

Alulírott Tomcsányi Domonkos Pál, a Pázmány Péter Katolikus Egyetem Információs Technológiai és Bionikai Karának hallgatója kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, és a szakdolgozatban csak a megadott forrásokat használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen a forrás megadásával megjelöltem. Ezt a Szakdolgozatot más szakon még nem nyújtottam be.

....
Tomcsányi Domonkos Pál

Table of Contents

Kivonat

Az LTE egy biztonságosnak és modernnek tartott távközlési rendszer, amely mobil szélessávú szolgáltatásokat nyújt felhasználók millióinak szerte a világban. Ebben a kutatásban megkérdőjelezem a fenti állításokat egy LTE bázis állomás, valamint egy saját minimalista gerinchálózat építésével. Ezekhez csak az Internetről szabadon elérhető nyílt forráskódú programokat és könnyen elérhető, kereskedelmi forgalomban is kapható hardvereszközöket használtam fel. Az elkészült tesztkörnyezet többek között alkalmas az LTE-Uu interface tesztelésére mind biztonsági, mind pedig helyesség szempontjából. A rendszer életképességét bizonyította, hogy több, korábban már más kutatók által felfedezett hibát sikerült vele reprodukálni az elvárt eredménnyel.

A rendszert egyéb célokra is felhasználtam a kutatás során, ezeket az alábbiakban részletezem:

- fuzzer szoftver felhasználásával generált teszt esetek injektálása és a hatásuk megfigyelése a célpont baseband implementációra
- a vészhelyzeti értesítő és broadcast protokoll (CBS) megismerése, kipróbálása, biztonsági vizsgálata

A felhasznált egyedi, saját készítésű szoftver immáron szabad szoftverként elérhető, hogy elősegítse és megkönnyítse az összes olyan jövőbeni kutatást, amely ezzel a területtel kíván foglalkozni.

Bár az LTE biztonsági szintje sokat javult az abszolút elődként ismert GSM-hez képest, mégis szükséges, hogy lássuk a hibáit és hiányosságait, mert csak így van lehetőségünk a jövő szabványait még biztonságosabbá tenni.

Abstract

LTE is considered to be a secure, modern ecosystem providing mobile broadband services to millions of users around the globe. In this research I challenge this by building an LTE base-station and minimalistic core network using open-source software and off the shelf hardware. I use this test environment to test the security of LTE's air interface (known as the Uu interface). I was able to prove how powerful such a simple system could be by reproducing bugs found by other researchers, then creating a proof of concept fuzzer and in the end trying out emergency broadcast alerts. The custom software created during this research is released as open-source to encourage and inspire the research of this field. LTE's security improved a lot since GSM or even 3G, but it is not a fault-free system and we need to be fully aware of its shortcomings to see the complete picture of it.

1. Introduction

LTE is currently the most popular choice among operators to provide mobile broadband services to customers. Its security is based on concepts first introduced in UMTS and currently considered strong, because it addresses all the main issues GSM had: integrity protection against fake base stations and stronger encryption against over-the-air eavesdropping. However, to encourage a more rapid deployment of 4G implementing most of these security features were made optional.

Ultimately this led to baseband software versions that don't even implement these optional features making it harder for operators to strengthen the security of their radio infrastructure later. Moreover, some fail to correctly implement even the mandatory, reduced set of security protocols making them and the device they are built-in vulnerable to a wide variety of attacks.

The goal of this research was to create a framework for testing such implementation errors using (if possible) open source components. Areas of research included messages with invalid formatting, reserved or not used values and response codes.

During my analysis I accomplished the following:

- I acquired a general good understanding of the state of air interface security from GSM to LTE
- I gathered and analyzed multiple publications describing and challenging LTE's security
- I evaluated software solutions designed to emulate LTE network components using off the shelf hardware
- I developed a test environment that consists of an open-source LTE base-station software combined with my own minimalistic core-network emulator
- I was able to verify the functionality of my environment via reproducing previously known bugs
- I extended my core-network emulator to make it capable of performing fuzzing based testing of baseband implementations
- I carried out an analysis (according my knowledge the first ever) of LTE's Cell Broadcast Services, found security weaknesses which I was able to demonstrate using my environment
- All tests I performed were verified using a commercial base-station to prove their feasibility and the interoperability of my core-network emulator

This thesis is divided into three parts: in the first chapters I describe the history of mobile broadband technologies, what motivated and shaped LTE to be the way it is today. The second part describes my test environment and testing methodology. Last but not least I present my results, describing what could be done in LTE security research using only open-source software and off the shelf hardware. I also briefly describe some ideas for mitigation and then finish with my conclusion and ideas for future work.

2. History of mobile broadband technologies

Since the beginning of life almost immediately there was a natural need, a desire to communicate with one and other, share feelings and information with our fellows. As human development carried on new technologies emerged with the purpose of aiding people satisfying this need. Information becoming more and more important is a trend we are currently experiencing and have been for the last 200 years, hence exchanging information is one of the greatest challenges of our times.

The whole landscape of communication changed many times during this period, but without doubt one of the greatest changes of all was the invention of the Internet (beginning with ARPANET in 1969). Suddenly one was able to connect with someone else residing anywhere on the planet, opening a whole new level of interaction that wasn't possible before.

In parallel with the Internet another industry was born out of necessity: the mobile industry. Mobility is defined as the ability to move from one place to another and the mobile industry's main goal revolves around this definition: giving people the possibility to access services, for example communicate while being on the move. Interestingly the Internet and the mobile industry didn't merge for quite a long time, instead they both developed individually. One of the causes for this was that a lot of technology required for such a merger simply wasn't invented yet – computing power for example wasn't adequate and it was only available in huge mainframe computers that are the exact opposite of 'mobile'. Another reason for this phenomenon was tradition – telecommunication was developed earlier than computers, therefore connecting mobile communication that has its roots in classic, "landline" telephony was not seen as a necessity for long time.

However, there was one thing common in both industries – they both were based on digital architectures. A digital architecture is a system that uses binary signals to represent data. For communication that is an optimal choice since only two different symbols need to be transmitted which leads to a more noise-resistant system. In the case of the Internet no other option came into consideration since computers had been using binary representation of data therefore the network which was designed to connect them together needed to be digital. On the other hand, telecommunication technologies were based mainly on analogue lines and systems, because their main task was to carry voice communication. When there wasn't any hardware commonly available that can do analogue to digital conversion of voice data in real time going fully analogue was the only option. However, in the 1980s the rapid development of computers and especially chips created a new situation in which digital telephony became feasible –also for mobile devices! A new standard in Europe called GSM was born – a digital system for mobile communication.

Now finally the Internet and mobile industry shared a common digital architecture, so the possibility for interconnection became feasible. Not long after GSM's introduction the standard was extended with the support for packet-switched data (non-voice binary data) *GSM 97* – and finally in 1993 with the standardization of GPRS (General Packet Radio Services) mobile broadband was born.

2.1 GPRS/EDGE

To better understand the way GPRS is constructed first basic concepts of GSM needs to be introduced. GSM was designed with the following requirements in mind: it needs to be as efficient as possible (because handheld devices have limited computing power and limited battery capacity), it needs to transmit mainly voice data and it should be robust to still provide reliable transmissions using simple hardware. These were all taken into consideration when GSM's coding scheme was chosen, or when the protocols used for transmissions were designed. The result of this work is the most widespread, most commonly deployed mobile communication system of all time with the largest subscriber base [1].

GSM uses two different techniques to provide multiple access: frequency division and time division. Frequency divisions means that the uplink (mobile to tower) and downlink (tower to mobile) channels are mapped to different frequencies. In the case of GSM there is a 40 MHz difference between them, creating essentially pairs of frequencies from which one is used for uplink and the other is used for downlink. The pairs are identified via a unique number called ARFCN (Absolute Radio Frequency Channel Number).

The second technique, time division refers to the way of how these frequencies are utilized. It means that at a given point in time only a single device is allowed to transmit, so interference is avoided. Instead of points there are periods of time reserved for a device to transmit. These are called timeslots and on each frequency, there are 8 of them, each with the length of 577 µs. Usually two timeslots of a frequency are used for signalling traffic (broadcast and dedicated) so 6 is left for voice and data transmissions. The idea behind GPRS was to utilize these traffic slots for binary data transmissions. In the contrary with CSD (Circuit Switched Data) GPRS relies on a packet-based system to send and receive data, while CSD is closer to the classic modem world, although it doesn't use audio signal to transmit modulated data. GPRS was later extended to support higher data rated via multiplexing more traffic timeslots resulting in the EDGE (Enhanced Data rates for GSM Evolution) system.

2.2 UMTS

All the previously discussed technologies were created out of necessity to react to the trends the original creators of the standards didn't see, namely the need of (as fast as possible) Internet connectivity while on the move. As the amount of data being transferred by mobile devices kept growing higher and higher it was decided that a new, universal system shall be created that unifies the packet based mobile broadband world and the older circuit-switched network. This lead to the creation of UMTS, the Universal Mobile Telecommunication System standards.

Its main objective was simple: add fast data packet connections to already deployed GSM networks while trying to keep most of the components unchanged making the transition easier and possibly cheaper. It was evident that one of the bottlenecks in GSM was the way the air interface was designed to work. It was optimized for 1980s hardware and the transmission of voice data – not suitable for the requirements of the 1990s and the first years of the 21st century. Hence the air interface was designed new from the ground up with new multiplexing and new modulating utilized to fulfill the requirements. This also meant that new base stations (here called NodeBs) needed to be deployed making the switch from GSM to UMTS, also known from 2G to 3G more expensive and complicated. On the other hand, most of the core network infrastructure could be left unchanged since the whole circuit-switched architecture of GSM was kept, only the parts dealing with packet based data needed to be either changed or extended to suit the requirements of higher data transfer rates.

Most of the protocols used were also changed to create a future-proof architecture for both signaling and user data transmissions. Thanks to 3G for example RRC (Radio Resource Control) was created that is still used on the Uu interface of LTE (that is the radio link, or also known as the air interface) as well. One of the greatest invention however that was introduced in UMTS are the NAS (Non-Access Stratum) and AS (Access Stratum) layers. Their main role is to separate traffic intended to the elements of the radio network and other traffic that's intended to core network elements. The NAS traffic is piggybacking on top of the AS traffic, creating an OSI-like layered system of protocols. These layers were also adopted in later technologies, for example LTE.

2.3 LTE

The number of UMTS deployments were increasing every day, most GSM operators worked on fitting and integrating the new system with their current in-production components. However, the engineering body responsible for standards sensed that despite the effort to make UMTS the one and only system for mobile communication, it isn't optimal, it's the opposite: it is unnecessarily complex. Especially for operators in North America who didn't have previous GSM installations UMTS presented a solution that would require a great number of unknown and possibly unnecessary components. As a solution to this problem it was decided that a new standard needs to be created, one that would be easy to implement, flexible and provide mobile broadband service to the users. This is the first time that voice communication wasn't one of the main services provided by the system, but a minor feature of a much larger system component (the IMS – IP Multimedia Subsystem).

The new standard is called LTE or Long Term Evolution (4G), and it is currently the most supported mobile communication standard in the world. It unifies all network operators because all of them is either already offering services or is planning on introducing them to customers.

3. Architecture of LTE

As it was mentioned in the previous chapter LTE was created to address the mobile communication needs of the 21st century customers – fast and responsive connectivity to the Internet. To cope with this challenge a series of changes needed to be made to the previous generation of standards. The main ones are: usage of frequency, coding, modulating and multiplexing scheme of data and core network architecture.

Frequency auctions organized by state agencies changed a lot since the 1990s, making it possible for operators to acquire a larger chunk of radio spectrum. LTE was designed in a way that it makes use of the larger frequency bandwidth available. This also required the change of multiplexing used so that multiple users sharing a larger radio spectrum could be managed efficiently without interference.

Last but not least the core network architecture needed to completely changed to reflect the main design goal – efficiently forward packets containing both signaling and user data.

3.1 Design Concepts

First and foremost it was decided by the creator and maintainer of the standard (3GPP [2]) that LTE will be a completely packet-switched standard that carries out both signaling and transfer of user data in packets. Packets are chunk of data that have a source and a destination address. A packet switched network has a single purpose – deliver the packets on the shortest route possible to the destination address. The largest packet switched network is the Internet, so it was an obvious choice to use the protocols used to run the Internet in LTE as well: there is a great amount of experience gathered and it also makes the inter-working between the two networks less complex.

This lead to the design choice of having an IP based network forming the backbone of LTE's core network. All network elements are to communicate with each other using IP packets. This also eliminated the need for special networking equipment supporting proprietary protocols, meaning that in most cases using enterprise IP networking devices is sufficient to operate an LTE network. This wasn't true in the case of for example GSM where proprietary protocols required special equipment to run the network.

Another design decision was to introduce OFDMA (Orthogonal Frequency Division Multiple Access) to the air interface which is essentially an extension of OFDM to multiple users via assigning sets of subcarriers to individual users. OFDM is based on using a conventional modulation scheme to modulate closely spaced orthogonal subcarrier signals that carry data. In case of LTE three different modulations were chosen: QPSK, 16QAM and 64QAM giving it a data rate of 2, 4 and 6 bits per symbol respectively. The use of OFDM makes LTE capable of coping with severe channel conditions (multipath propagation fading for example).

LTE was created as a flexible network that can adapt the available frequency spectrum therefore several versions were defined that only differ in the occupied frequency bandwidth. The minimum bandwidth required is 1.4 MHz, after that there is 3, 5, 10, 15 and 20 MHz. Uplink and downlink could also be separated using two different schemes: time division (mostly used in Asia) and frequency division (used in Europe and USA).

Naturally the new standard needed to be backwards compatible towards previous networks (both 3G, GSM but also towards mostly US-based CDMA networks). This was accomplished using different gateways that are capable of translating between previous, possibly proprietary or even analog protocols or lines and LTE's packet switched, digital architecture.

LTE had to cope with contradicting requirements since it needed to bring previously unseen mobile data transfer speeds while trying to maintain low energy needs from the

clients so they can stay mobile for longer periods of time. In reality the emphasis was put on data transfer speeds, but low-energy modes were added while the user equipment is in idle mode creating a manageable compromise.
Observing the way LTE is built clearly shows that the creators tried to accommodate the greatest amount of use cases bringing a never-before-seen flexibility to the standard from the beginning of the design process with general success.

3.2 Components

LTE was created with simplicity in mind therefore there are not many network components defined at the first place. Most components are highly autonomous eliminating the need for extra controller elements (for example in a GSM network a handover needed to be orchestrated by a Base Station Controller, in LTE the base stations can communicate directly with each other and can manage a handover on their own).
There are two main groups of network elements: EUTRAN and EPC. EUTRAN stands for Extended Universal Terrestrial Radio Area Network and it contains base stations, which in LTE terminology are referred to as eNodeBs (evolved NodeBs from UMTS). EPC on the other hand means Evolved Packet Core and it is the core network part of the system containing elements needed to coordinate eNodeBs, store customer data and connect the system to other networks, mainly the Internet. Refer to Figure 3.1 for an overview of an LTE system.

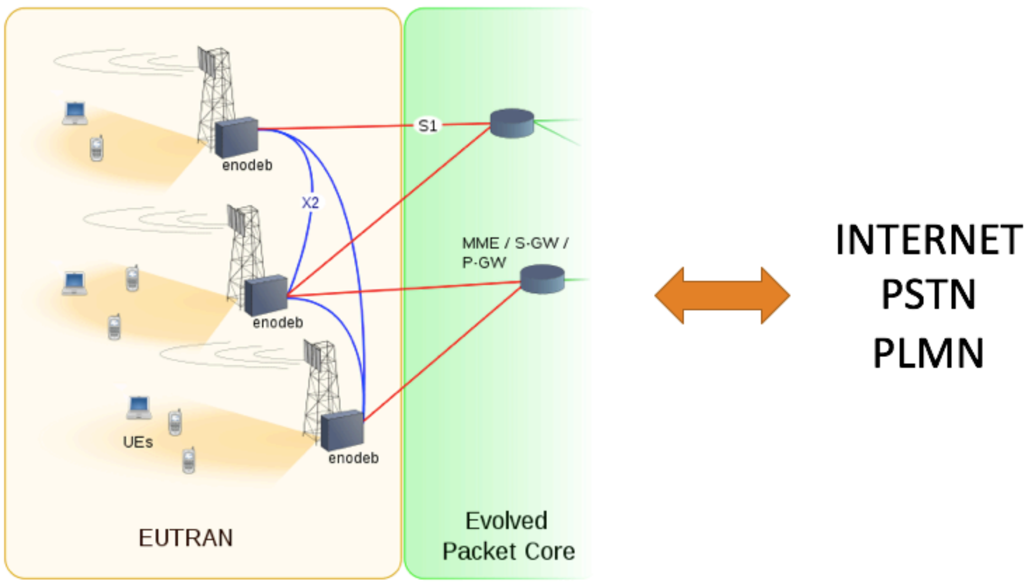


Figure 3.1 - LTE Architecture
From Crati, "EUTRAN architecture diagram". [Online]. Available: *https://commons.wikimedia.org/wiki/File:EUTRAN_arch.op.svg*

3.2.1 EUTRAN

The Evolved Universal Terrestrial Radio Area Network contains eNodeBs that are both connected to each other using the X2 interface defined in [3] and to the core network via the S1 interface [4]. The main task of eNodeBs is to provide radio access to the LTE network in a certain area. They handle all radio connectivity on their own, the core network is only concerned with higher level procedures creating a clean and well defined environment. Most eNodeBs are aware of their surroundings and capable of adjusting transmission power based on measurements of neighbor cells' signals making them a self-controlled, autonomous network.

3.2.2 EPC

The Evolved Packet Core is defined as a group of four different components: the MME, the HSS, the P-GW and the S-GW.
MME, or Mobile Management Entity is the main control element of the network. As its name suggests its task is to manage all UE (User Equipment) related transaction, for example authentication, paging, bearer activation and de-activation.

The second component is the main database that stores all customer data and keys called Home Subscriber Server or HSS. It generates authentication and session keys, and manages subscriptions related information.

The Serving Gateway (S-GW) is a network concentrator element that acts as an anchor gateway towards UEs. This means that the UE can change eNodeBs as it moves between cells but it will keep communicating with the same S-GW, making handovers continuous and seamless for the customers.

The Packet Gateway is concentrating all kinds of packet traffic from different sources: previous generation networks, maybe even non 3GPP networks (WiMAX for example) and Serving Gateways. Its task is to act as a single point of entry and exit for UE traffic.

Connectivity between these elements happens on well-defined interfaces [5]. As mentioned previously the MME connects to eNodeBs via the S1 interface. Besides S1 the MME also has an S6a and an S11 interface to connect to the HSS and the S-GW respectively. UEs are connected directly to the S-GW via the eNodeB using the S1-U interface. Last, but not least the S-GW is connected to the P-GW using the S5/S8 interface.

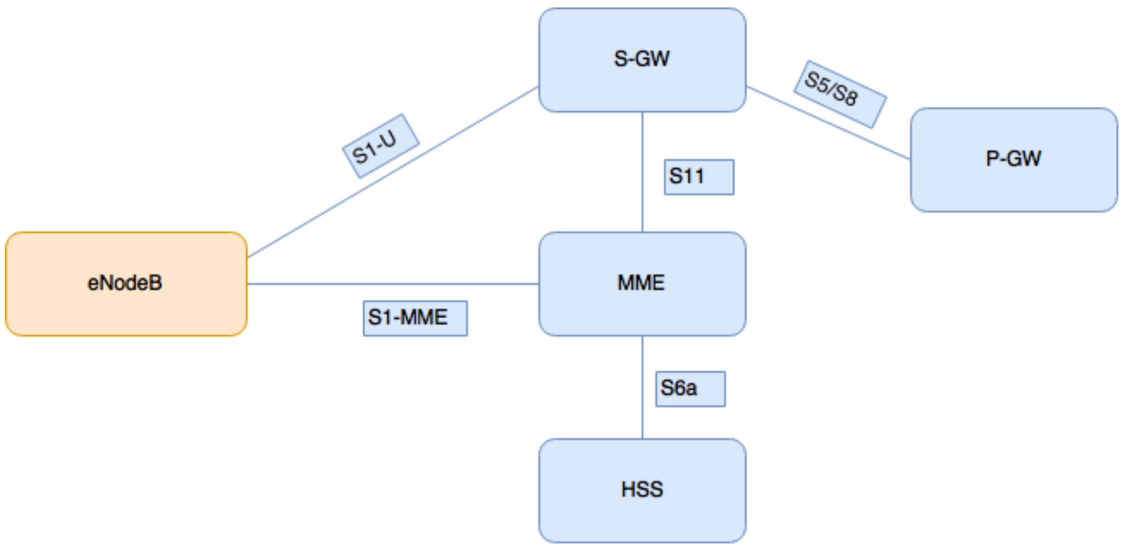


Figure 3.2 - EPC Interfaces

3.3 Protocols/Layers

LTE is based on the so-called SAE or System Architecture Evolution which essentially means that the whole network is based on IP connectivity between the elements. Besides adapting IP as its routing protocol LTE also uses another technique from computer networks: encapsulation. Encapsulation means that packets are built from a single payload that is defined based on one protocol, but it could be wrapped in a different type of packet creating a multi-layer packet. Each network element knows which layer it needs to parse in order to know what to do with the packet, creating an effective architecture.

The 3GPP decided not to adopt TCP because it isn't suitable for telecommunication purposes. In a telecommunication system availability and fault-tolerance are key factors and TCP with its rigid, point-to-point, transmission based protocol doesn't fulfill these requirements. Instead SCTP (Session Control Transmission Protocol) was chosen which is a message based, multi-path protocol that is more suitable for the purposes of LTE. It combines the advantages of UDP and TCP – it is message oriented meaning that instead of a continuous stream of bytes it knows exactly what a complete message is and can repeat it in case of packet loss, just like UDP, but also it ensures reliable and in-sequence transport just like TCP. Added features include the aforementioned redundant/multi path connections making it a robust protocol.

LTE also makes heavy use of UDP to create a fast and flexible (fault tolerant) way of forwarding user-data, achieving low latency and high availability. User-data on the S1-U interface is wrapped in UDP packets using GTP-U (GPRS Tunneling Protocol).

3.3.1 RRC

RRC, or Radio Resource Control is the protocol used on the Uu (air interface) of LTE. Its main purpose is to handle radio signaling e.g. connection establishment and release, broadcasting system information, bearer establishment, reconfiguration and release. It is built on the PDCP protocol and defined in [6] using ASN.1 definitions.

3.3.2 S1AP

S1 Application Protocol is the signaling protocol between the core network and the radio access network. It has many functions: mobility management, radio access bearer management, UE context management and location reporting. An S1AP connection begins with a simple message exchange between the eNodeB and the MME:

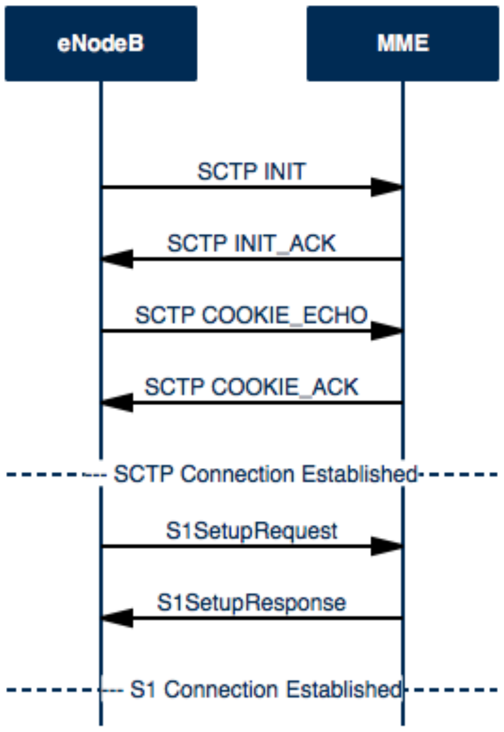
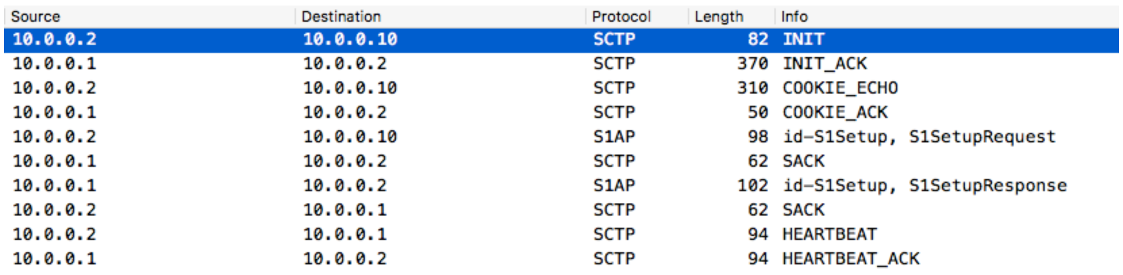


Figure 3.3 - S1 Connection Initialization Flow

The eNodeB first establishes the SCTP connection with the MME using the standard port 36412. After the connection is established the eNodeB initiates the S1 connection (using the SCTP connection) via sending the S1SetupRequest initiating packet to which the MME replies with a SetupResponse message. After that the connection is established. Figure 3.4 shows the packet flow in Wireshark, a widely used packet capture and analyzer tool that has support for parsing S1AP messages.



Source	Destination	Protocol	Length	Info
10.0.0.2	10.0.0.10	SCTP	82	INIT
10.0.0.1	10.0.0.2	SCTP	370	INIT_ACK
10.0.0.2	10.0.0.10	SCTP	310	COOKIE_ECHO
10.0.0.1	10.0.0.2	SCTP	50	COOKIE_ACK
10.0.0.2	10.0.0.10	S1AP	98	id-S1Setup, S1SetupRequest
10.0.0.1	10.0.0.2	SCTP	62	SACK
10.0.0.1	10.0.0.2	S1AP	102	id-S1Setup, S1SetupResponse
10.0.0.2	10.0.0.1	SCTP	62	SACK
10.0.0.2	10.0.0.1	SCTP	94	HEARTBEAT
10.0.0.1	10.0.0.2	SCTP	94	HEARTBEAT_ACK

Figure 3.4 - S1AP Connection Initialization (actual packets viewed in Wireshark)

After the connection has been established it is used both for UE transactions as well as eNodeB signaling related messages.

Every message needs to be one of the three types defined: initiatingMessage, succesfulOutcome and unsuccessfulOutcome. The message's type helps the receiving party determine the exact state of the transaction in the defined procedure.
The procedure code identifies the ongoing procedure based on the codes assigned in the standard. Every element has a so called 'criticality' field which defines the reaction of the receiving party in case of any error is detected when decoding the contents of the element. Criticality is defined as a three-value enum: reject, ignore and ignore with sender notification. The meaning of these is self-explanatory. There is also a so-called presence field which indicates whether the element contains information that's mandatory to be sent, or it's just optional.
Each S1AP message consists of one or multiple IEs or Information Elements that carry the actual data. All the possible IEs with all of their possible fields is defined in the standard using ASN.1 notation [4]

3.3.3 NAS

NAS, or Non-Access Stratum is the cornerstone of LTE's UE signaling. The name means that data exchanged via this layer isn't read, modified or interpreted by any of the forwarding entities – it is a direct connection between the MME and the UE itself. Naturally the encapsulation of a NAS message changes while being delivered (from an RRC message to a S1AP message and vice versa) but its contents remain the same (ensured by cryptographic functions as well). Over the NAS layer two protocols are

used: EMM (EPS Mobility Management) and ESM (EPS Session Management). EMM is used “to support the mobility of a user equipment, such as informing the network of its present location and providing user identity confidentiality.” (chapter 5.1.1 of [7]). ESM on the other hand has the task “to support the EPS bearer context handling in the UE and in the MME.” (chapter 6.1.1 of [7]).
So while both protocols are used for signaling one of them deals only with administrative tasks while the other does the actual heavy lifting of setting up and managing the data session of a user.

4. Security of LTE

During my research I first evaluated the overall security of LTE, then analyzed the targeted interface (Uu) deeper by going through the following steps: understanding the interface, inspect current, already published issues, find new, possible issues and test them using the environment I created.

4.1 LTE interfaces

In LTE there are two categories of interfaces that could be exploited by a malicious attacker: the public interface (the air interface) and the private interfaces (core network interfaces).

4.1.1 Private interfaces

From all core network interfaces the S1 interface stands out, because it connects the EPC with the EUTRAN meaning that it could be accessed by a skilled attacker by circumventing physical security and hijacking the traffic. There could be also a possibility that the S1 interface isn’t delivered via a private line, but on a shared (possibly fiber) connection. In this case there is a chance an attacker with certain access to networking equipment (switches, routers) can redirect S1 traffic creating a Thing-In-The-Middle situation. As a countermeasure against these attacks it is strongly recommended that operators turn on encryption (IPsec) on the S1 connection. IPsec is a strong, and totally transparent way (towards upper layers) of encrypting IP packets with very little overhead. This means that if the configuration is done properly then the attacker could only cause Denial-of-Service, or has to develop a 0-day attack which is considered to be a hard task according the risk model.

4.1.2 Public interface

LTE's air interface's protection however isn't this robust. In LTE the use of encryption is optional, it is fully up to the mobile network operator to turn it on or leave it off. This was done supposedly to speed up the deployment of LTE networks, however as most default options it is still active today in most of the deployed networks. Previously it was also allowed to disable integrity protection, but it's currently shouldn't be accepted by clients because of security concerns.

LTE uses a AKA (Authentication and Key Agreement) protocol that's based on 3G to authenticate a client and generate keys. During the execution of the protocol a master key (and from that multiple session keys) is generated using random values from the network and a pre-shared key. This pre-shared key is stored both in the operator's database and in the customer's SIM card module. It can't be read from the SIM card, the card itself handles the incoming challenge from the network, generates the master key and the signed response to be sent back to the network. Generating keys have been improved since 3G by adding more steps to the process which result in the multiple keys used to secure different parts of the transaction: there are keys to encrypt and integrity protect RRC messages (intended to the eNodeB), and there are keys to encrypt and integrity protect NAS messages (intended to the core-network). The complete key verification process, the used encryption and integrity protection algorithms were analyzed by researchers without finding any major weakness in them [8]. This means that if encryption is enabled on the network then unless an attacker gets hold of the master key or the session keys (via breaking into the core network) the communication via the LTE air interface can't be eavesdropped on.

Noe that I have established that LTE's Uu interface is not likely to be eavesdropped on, the next step was to research current known issues, understand their way of operation and search for other exploitable issues.

4.2 Current known issues

4.2.1 Unauthorized IMEI access (IMEI leak)

This vulnerability was disclosed in [9] by Benoit Michau and Christophe Devine. According to the LTE specifications a UE should never give out its IMEI (International Mobile Equipment Identity) when asked via an Identity Request except in two cases: the UE is trying to establish an Emergency Bearer (chapter 5.1.2, paragraph 2 of [7]), or security context has already been established meaning the Identity Request message is integrity protected (chapter 4.4.4.2 of [7]). This should prevent an attacker with a fake eNodeB from obtaining the IMEI of a target, because it cannot force the victim to start an emergency session, nor does he have access to valid keys to perform integrity protection. Figure 4.1 shows the expected message flow.

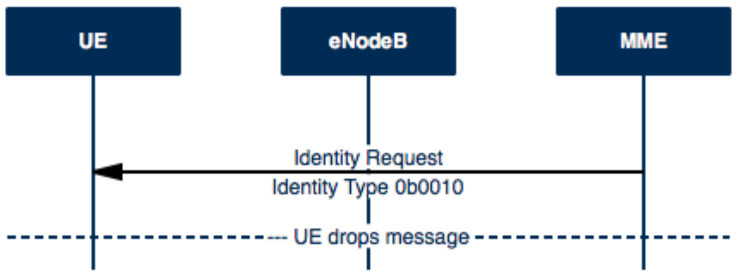


Figure 4.1 - Expected UE behavior upon receiving an unprotected Identity Request, Type 0b0010 (IMEI) or invalid type

According to the research [9] some baseband vendors don't comply with the specs. Their findings showed, that there is an error in the way the baseband software handles Identity Requests. This message type has a field called Mobile Identity information element that is coded as a 4-bit long property, but according to the standards only the last 3 bits are used to actually represent different identity types (see Table 10.5.4 of

[10]). However if one sets the fourth bit (the most significant bit) to 1 then vulnerable basebands will reply with their IMEI even if the request message isn't protected. The Identity Type to send is decimal 10 which is 0b**1**010 instead of the defined 0b**0**010 value.

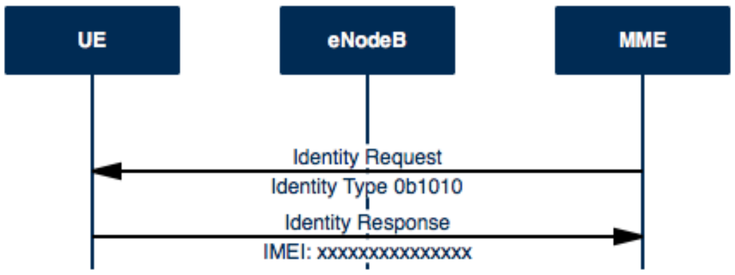


Figure 4.2 - Behavior of a vulnerable UE

4.2.2 Targeted, location dependent Denial of Service

This vulnerability is not an implementation issue, but a protocol bug meaning that all devices operating according to the specifications are affected by it. It was discovered by Ravishankar Borgaonkar and Altaf Shaik and made public in [11]

The problem arises from the fact that the network needs to have a way of discarding UEs that aren't relevant for some reason: they were reported stolen (so they are not allowed on the network), they don't have subscription for LTE services so they need to go back to 3G and so on. The way the network handles these is using a reject message (depending on what kind of transaction was initiated by the phone) with a specific cause. There are 110 different causes defined in the standard and they require different reactions from the client. Some simply instruct the UE to change back to 3G services and don't try to initiate contact with 4G, but others can cause the UE to change to "EMM-deregistered" state in which it is not connected to the network, so any incoming request intended to the target (call, SMS, data packet) would fail until the UE is rebooted or the SIM card is ejected (UE changed to EMM deregistered state) (see Table 9.9.3.9.1 - EMM causes in [7]).

Reject messages are unauthenticated, so they could be sent by an attacker.

After evaluation of the known issues it became obvious that layer 3 is the most interesting from a security point of view (all vulnerabilities listed were found in layer 3), it provides possible low hanging fruits for an attacker so it was chosen to be analyzed.

4.3 LTE Uu – Layer 3

In LTE layer 3 is the top signaling layer containing NAS, IP and RRC messages. All layers above it contain user data, that could be considered payload from our point of view.

Besides providing the most interesting, and possibly most vulnerable functions there are other advantages of testing layer 3: there are tools currently already available to carry out testing and it is a reliable layer to test, because lower layers that deal with radio connectivity could be taken for granted. This guarantees reliability in contrary to low, physical-radio level layer testing where it is not necessarily easy to differentiate transmission errors from actual bugs.

4.4 Protocol

Layer 3 is realized in LTE on the S1 interface. Messages are transmitted using the S1AP protocol. S1AP messages contain a dedicated Information Element containing NAS data, other IEs comprising the AS protocol.

The protocols this research decided to target are ESM and EMM, both are transferred via the NAS part of layer 3. These protocols are responsible for mobility management and session management in LTE. The first step of the analysis was to determine the attacker model. I decided to pick an attacker that has no access to cryptographic keys (either by cracking them, or stealing from the core network) but can use active devices (e.g. a fake eNodeB). I think this is a feasible, close-to-real-life scenario, because as my test environment demonstrates creating and running an eNodeB is easily accomplished via openly available free software and off the shelf hardware. As previously discussed cracking or stealing keys was ruled to be not feasible, since the former is currently not possible (in a feasible timeframe) according to the information available to the public, the latter would require serious effort of breaking into an operator's core network (which would also grant access to all user data rendering the attacking of the air interface mostly unnecessary).

Next I completed a risk analysis and selected the elements with the highest risk. According to my findings these are the messages that could be sent without encryption and integrity protection. They also suit the attacker model well.

The following messages were found (chapter 4.4.4.3 of [7]):

- Tracking Area Update Reject
- Identity Request (with certain conditions)
- Attach Reject
- Service Reject
- Detach Accept
- Authentication Request
- Authentication Reject

While browsing through the standards I also found out about a special, broadcast system that, as far as I know was never inspected. It's called CBS (Cell Broadcast Services) and it includes multiple emergency alerting systems (ETWS, CMAS, PWS) and other features (exchanging data with SIM).

These messages all contain elements that could be fuzzed to some extent (specific fuzzing). In the following I give some examples of fuzzing targets found during my analysis:

- All reject messages contain a reject cause code that is defined in the standard as a value between 2 and 113, giving the possibility to fuzz this field from 0 to a full byte (255).
- An identity request contains a 4-bit field defining the type of identity requested that could be fuzzed.
- An Attach Reject message is allowed to contain a complete ESM message container which has multiple fields available for fuzzing.

4.5 Attack Scenarios

Attack scenarios could be differentiated into two categories: attacking a UE and attacking a network. Attacking a UE could be beneficial to the attacker in different ways: he could be able to extract sensitive data, maybe even get remote code execution in the baseband processors context giving him total access of the target's network activities and data. In some cases when the baseband has access to the complete memory of the device (even the parts used by the device's operating system) this could lead to total control of the device. Sensitive data to be extracted includes identities (IMSI, IMEI) making later identification of the target possible. IMSI (International Mobile Subscriber Identity) is the ID of the SIM card, meaning that it identifies a customer. If an IMSI could be correlated with a certain area then the person owning the specific SIM card could be located and targeted. IMEI (International Mobile Equipment Identity) is the unique ID of the device connecting to the network. Since users doesn't often switch devices it could also be used for location tracking and targeting.

Attacking the network could yield even better results for an attacker, because disturbing the operation of the core network, or getting code execution on any of the network components attached to the core network could be catastrophic from the point of view of the operator. It could cause revenue loss, but worst of all: loss of customer trust. Attacking the network seems preferable for an attacker, but it requires a lot more effort and time in exchange.

Requirements for attacking a device requires vicinity to the target and the setup of a fake eNodeB so the attack payload can be transferred to the device. I will demonstrate later, that creating such a setup is not a complex task.

Attacking the network requires a rogue UE that can send arbitrary data into the network compromising possibly any element in it. There have been examples of both kind of attacks published [9] [11] [12] [13], but I decided for this research I will concentrate on the former, UE targeting attack.

5. Test Environment

When assembling the testbed used for this research I had three main focus points: I should use off the shelf hardware, I should use open-source software and I should test as many different baseband implementations as possible.
Basebands in contrary to devices (phones, LTE modems etc.) are not manufactured by a lot of market players, but only a handful companies. The dominant vendors are the following: Qualcomm, MediaTek, Samsung, Huawei and Intel. I was able to gather devices containing four different manufacturers from this list giving me a good coverage. All software I was able to find was created to be used with SDR (Software Defined Radio), so the choice was made to use such a device to handle all radio activities.

The final list of hardware included the following: 5 different phones, a National Instruments USRP B200 software defined radio board, 10 test SIM cards, and finally for verification of all the work a commercial LTE eNodeB (Nokia FlexiBTS MacroCell). The following phones were used:

- Huawei P2 (baseband manufacturer: Huawei)
- Allview P5 pro (baseband manufacturer: MediaTek)
- Samsung Galaxy Mega2 (baseband manufacturer: Samsung)
- Wiko Birdy (baseband manufacturer: MediaTek)
- Nexus 5 (baseband manufacturer: Qualcomm)

All phones were updated via their built-in update mechanisms to the latest updates available at 1st September, 2016.

5.1 SDR

I decided to use one of the most supported platforms available, currently manufactured by Ettus (part of National Instruments), the USRP (Universal Software Defined Radio Platform). This is one of the most supported SDR platforms available, it is compatible

with all the software I intended to work with. It also provides transmission capabilities and is affordable (so an attacker on lower budget can get it as well).
Software Defined Radio means that the hardware itself only acts as a generic radio hardware and all the processing is done entirely in software. It could be described in simple terms as an antenna and an ADC (Analog Digital Converter). The digitalized data flowing in from the device to the computer and it is used to reconstruct the complete waveform observed on the air. This enables creating complete radio solutions (from de-modulating to interpreting) entirely from software – a truly flexible solution. Since the hardware doesn't contain any part that's specific to a certain application it can be used for decoding all kinds of protocols and data that's in the frequency range of the SDR device.
Using SDR hardware to decode mobile communication signals off the air is a common way of understanding and researching a system. An example to this would be the gr-gsm project (previously known as airprobe) which is a complete decoder for GSM signals – from demodulation through applying error correction to interpretation of the bits.
There are multiple open-source projects that deal with LTE. In the following chapter I'm going to analyze them from the point of view of security researcher.

5.2 Open Source LTE software components

After the creation of the so-called Osmocom project [14] in 2008 multiple different pieces of open source software was created, aiming at implementing at least partially mobile communication system components. Most of Osmocom projects deal with GSM only, but the courage and the determination of the people behind it inspired the creation of other projects. Some of these try to tackle 3G, but most of them aims at 4G since that's considered to be the de facto technology for mobile broadband in the future. The following projects were taken into consideration to be used during this research:

- openLTE
- srsLTE
- LTE Base Station Software
- Open Air Interface

OpenLTE is a project created, and led by Ben Wojtowicz since 2011 [15]. It provides good quality code in C that is well commented via comments in the code. The definitions and header files from this project is used in multiple other projects as well. However it has a major bottleneck in terms of flexibility: it doesn't differentiate LTE components into separate software components, and therefore it doesn't implement the standard interfaces between those components either. This means that if one would like to change its behavior the source code needs to be modified, there are no interfaces through which one can inject a modified message. OpenLTE is similar to OpenBTS – it's a single project implementing multiple components in one, therefore it is less complex, but also less stable and not flexible.

SrsLTE [16] is a collection of code, a lightweight library for LTE related functions. It is written in C and it contains some pieces of openLTE's code. On its own it is just a library, so to use it one would need to create new software based on it. Software Radio Systems Ltd., the creator and maintainer of the project provides some examples, e.g. a UE implementation.
From the security tester's point of view srsLTE is great if there is enough knowledge and time to write new code, or there is a request to test an eNodeB's air interfaces from a rogue client's perspective (using the provided UE example). However during my tests the example UE application didn't connect to any LTE network because of a (currently) not known error. My verdict is that it needs more contributors and testing but it's already a great piece of software.

LTE Base Station Software [17] was created by Fabrice Bellard. Currently it is maintained by Amarisoft – a vendor specializing in manufacturing LTE test equipment. Since it is not freely available software, neither is it open-source anymore I did not do any tests with it. Evaluation of it needs to be done before using it in any project.

OpenAirInterface [18] is a project started at EURECOM in France. It is a joint graduate school and research center uniting multiple different universities. OpenAirInterface aims to implement all standard components and interfaces of LTE. This gives it a lot more flexibility and stability over the previously mentioned projects. From the viewpoint of security testing it is an ideal platform, because all individual components could be replaced, or modified to suit the needs of the researcher. It contains all LTE components: an SDR based eNodeB, MME, P and S gateways and HSS, as well as different emulators to use instead of real hardware.

5.3 Software Architecture

After careful consideration OpenAirInterface (OAI) was selected to be used during this research. The flexibility gained from having all components and all standard interfaces between them outweighed the cost of having to do occasionally longer drill-downs into the code while trying to solve an issue.

This thesis's focus is Layer 3 messages in LTE, so a way of generating and injecting such messages needed to reach the desired goal. First the idea was to modify the MME component of OAI to suit the needs of this research, but after analyzing the code I decided that instead of relying on unknown code I'd rather create my own, minimalistic implementation of an EPC. This way instead of doing a top-down approach trying to strip away unnecessary functions from the MME module I could apply the bottom-up strategy, adding only the bare minimum features needed to function. This helped keeping the code simple and clean.

The resulting piece of software is called doMME. Its architecture is described in detail in the next chapter.

The final architecture could be seen on Figure 5.1.

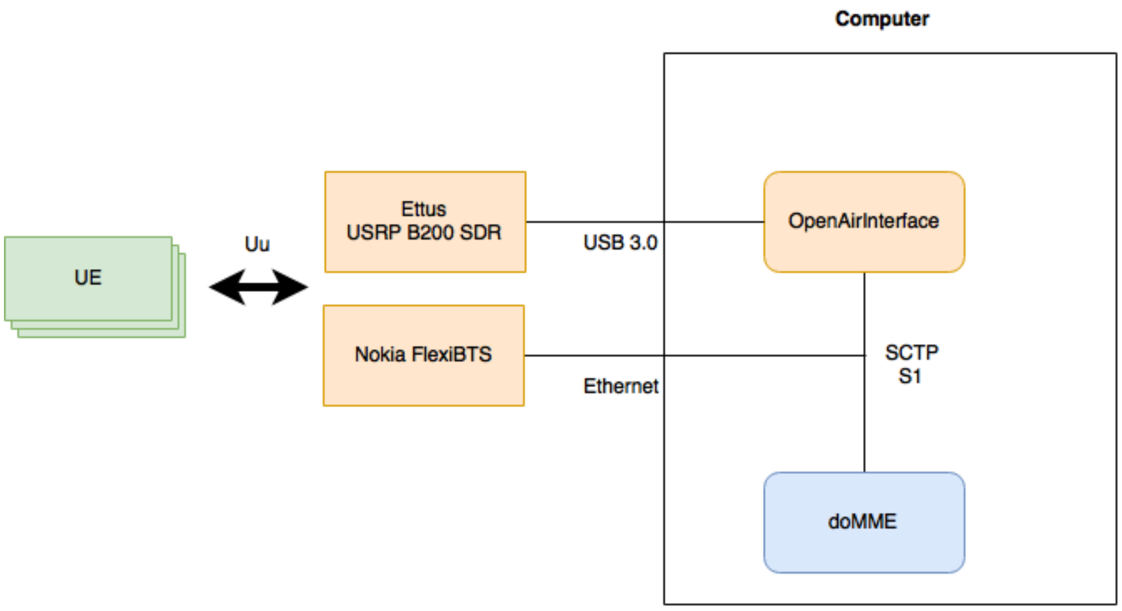


Figure 5.1 - Final architecture of the testbed (orange components: EUTRAN, blue: EPC)

OpenAirInterface acts as the eNodeB, using the USRP B200 as radio hardware. DoMME emulates a complete core network, and presents the S1 interface towards both OAI and the Nokia FlexiBTS (used to verify results).

The complete lab setup (except the Nokia FlexiBTS) could be seen on Figure 5.2.

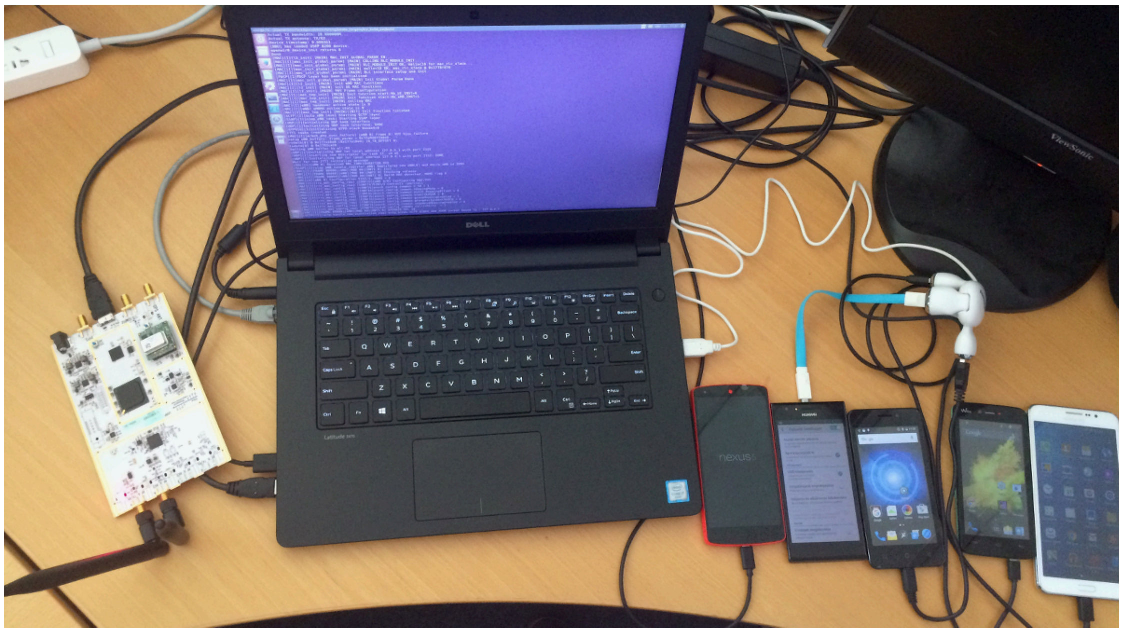


Figure 5.2 - Lab setup

6. doMME

DoMME was initially created to be a flexible platform capable of injection arbitrary LTE Layer 3 messages into an S1 connection for security testing, but in its current form it is capable of functioning as a minimalistic complete EPC emulator with experimental packet connectivity. It contains multiple modules that implement a function of a core network.

It is written entirely in Python and relies heavily on three different open source projects. First and foremost pysctp [19], which is a thin wrapper around the Linux kernel's SCTP support stack. It also uses libmich [20] and CryptoMobile [21], both of which is created and maintained by Benoit Michau. Libmich is a library implementing both S1AP, EMM and ESM messages in Python making message parsing a simple library call. CryptoMobile is a Python wrapper around a C implementation of all the cryptographic functions used in LTE and 3G.

6.1 Main Architecture

DoMME uses configuration files defining different message flows to determine what to answer to an incoming message. Flows can be seen as skeletons for a state machine defined per device. A device will always go through the states/messages defined in the flow. Flows can be chained together into 'scenarios', for example after an 'attachFlow' there is a possibility to call an 'smsFlow' that will deliver a text message to the device after attaching. Differentiating between devices is currently done on the order of connection initiation: ue0 will be the first device that tried to attach, ue1 the second and so on. There is a way to define a default action that will applied to all connecting devices.

Naturally there are exceptions when the flow can't be followed for different reasons, therefore doMME is capable of saving the current state of flow, reply with messages based on a different flow, then return to the original, saved flow. Although this makes the classic, state-machine based architecture more complicated it is sufficient for my, mainly proof of concept tasks.

6.2 Modules and Classes

In the following the different modules implementing a function of a core network are described.

Main class

The main class of doMME implements the SCTP listener and creates the base class called MME. It is also responsible for spawning, and maintainging a new eNB instance(s) as soon as a new eNodeB connects to doMME.

6.2.1 MME

MME is a class holding all, global configuration data of the running instance. It is being referenced by all running eNB threads in a read-only fashion to make it a thread-safe. The MME class provides access to the global logging class instance as well that allows all threads to print out notifications.

6.2.2 UE

The UE class holds all data related to a specific UE connecting to the MME. It has the current state of it (in which flow is it, at what state), cryptographic keys (if already generated), IMSI, IMEI, GUTI, IP address, mTMSI, GTP tunnel ID (if available)

6.2.3 eNBhandler

The eNB class acts as gateway between an eNodeB and any kind of client program. It does that by maintaining the SCTP connection with the eNodeB, and by providing a Unix Domain Socket. Through this socket any data received will be forwarded to the eNodeB via SCTP and any reply will be available as well. This architecture makes it possible for anyone to write software that just generates messages, other tasks such as maintaining connection, handling the initial S1 Setup messages are all taken care of. DoMME makes use of this architecture itself: its baseHandler class that deals with messages connects to this Unix Domain Socket as well.

6.2.4 baseHandler

This class does most of the heavy lifting providing the real functionality of doMME. First it connects to the aforementioned domain socket, and waits for messages. If there is an S1AP message it decodes the message's type, then calls the appropriate message parser. The message's type is used to determine which parser to call. Based on the return value of the parser function it determines what to send back. It keeps track of all UEs connected to the eNodeB it handles.

6.2.5 pagingHandler

The pagingHandler is a separate thread, because it needs to operate independently from the baseHandler – it needs to be capable of sending messages without waiting for a device to initiate contact. It handles paging, so when activated via its interface it starts sending out pagings every 2 seconds until it is signaled to stop, because the required UE initiated contact with the MME. It has an internal list of UEs to be paged.

6.2.6 Parser modules

There are two layers of parser modules: the first layer is responsible for parsing S1AP messages, the second layer decodes NAS messages (EMM and ESM). Based on the message they modify the UE instance if needed, and move its state forward (change to the next element in its flow).

6.2.7 Getter modules

These are the exact opposite of parser modules, they are assembling replies to messages. After a parser module returned the baseHandler calls the appropriate getter module based on the flow of the UE with the right parameters.

6.3 Interfaces

6.3.1 pagingHandler

The pagin handler exposes the following interfaces:

page(UE ue) – starts the paging of ue (which is a UE instance), returns void

stopPage(UE ue) – stops the paging of ue, returns void

wasPaged(UE ue) – return True if the specified ue is being paged/was paged already

6.3.2 S1AP parser modules

The S1AP parser modules expose the following interface:

parse(baseHandler base, dict pdu_dict) – parses the pdu_dict dictionary, and gets a pointer to the baseHandler so it can access global configuration options, logging etc. Returns None if the no answer is required, UE if the reply should be crafted based on a UE's flow

6.3.3 NAS parser modules

Depending on the module it provides either one of these interfaces. The S1 parser module is responsible for deciding whether the MME is contacted by a known UE, or an unknown UE (so is there context stored or not) and call the right function:

parse(baseHandler base, dict nas_dict) – parses the nas_dict dictionary if there is no context, returns a UE instance (either new, or already existing if found based on NAS message data)

parseWcontext(baseHandler base, UE ue, dict nas_dict) – parses the nas_dict dictionary with the known context of ue, returns UE instance

6.3.4 NAS getter modules

These modules return the appropriate NAS messages. All messages replying to an unknown UE (Tracking Area Update Reject, Attach Reject etc.) ignore the second parameter

get(baseHandler base, UE ue) – returns the NAS message based on the UE's properties as binary string, called by the S1 getter modules

6.3.5 S1AP getter modules

These modules return the appropriate S1AP messages (possibly including NAS messages from the previous group of modules)

get(baseHandler base, UE ue) – returns a complete S1AP message as byte string

6.4 Imported subsystems

The Authentication Center component (complete with customer database, challenge generation etc.), the CryptoMobile module (providing all encryption, decryption and intergrity protection), the GTPu Daemon (providing experimental IP connectivity to devices) were imported. They were made openly available by their author, Benoit Michau via GitHub [20]

6.5 Flows and Scenarios

Flows and scenarios are essential parts of doMME. They are built as complex types (list of tuples).

An example of a flow looks like this:

```
 flow = [(0,'page'), ('SERVICE_REQUEST','InitialUEMessage'),
                  (0, 'initialContextSetupRequest'),
                  (0, 'InitialContextSetupResponse'),
                  ('DOWNLINK_NAS_TRANSPORT', 'downlinkNASTransport')
           ]
```

This flow sends an SMS to the UE. It first pages the UE (the leading 0 means there is no NAS message to be included), then waits until the UE initiates a Service Request, sets up the Initial Context with the UE, then sends out the text as Downlink NAS transport.

An example of a scenario.conf looks like this:

```
[UEs]
ue0 = attach, sms, idle
```

This config file would result in the first UE being attached, then delivered a welcome SMS, then released into idle mode.

6.6 Operation

DoMME was created with two goals in mind: minimalistic functionality and extensible architecture. This lead to a modular software in which message parsing takes place on multiple levels (based on the packet's type, and the layers embedded in it). Each parser iterates through all the layers it can understand, and sets various properties of objects, or the transaction itself based on the interpreted data, then calls the parser from the inner layer. After the inner layer is done the parser will return with the appropriate response or object (which again is modified based on the data received).

Since most of the procedures described in the standard are built in a dialogue like way (question-answer) doMME uses a central state machine to go through a procedure. Procedures are called 'scenarios' and could be defined using a simple syntax by noting the message's type (which also determines which parser will work on it). After a message is received and parsed a reply is constructed. Replies to a message are constructed in a similar fashion to how parsers worked: the most inner layer gets created first, and then data gets encapsulated as required, then it leaves the network interface as an SCTP packet.

To know what to reply to a query the aforementioned 'scenarios' are used – they determine the response packets as well. This architecture isn't as simple as possible, but it has multiple advantages: an out of order, or unexpected packet could be detected immediately, since it doesn't match the predefined value in the scenario configuration file. It also eliminates the problem of cross linking files – meaning that there is no need for a parser to know what getter it needs to call, the scenario file will determine that, so if there is any change required then it needs to be done only at one place instead of multiple.

Flexibility also was a key point while creating doMME, therefore it is based on a modular system. Parsers and reply-constructors or getters could be easily added or modified. Current modules are files defining the required *parse* or *get* functions which usually provide a wrapper around the appropriate libmich call.

DoMME was tested with both Open Air Interface and commercial LTE eNodeBs and it proved to be fulfilling its original goals by providing the minimum set of messages and procedures needed to successfully run an LTE network.

7. Security Testing Concepts

The attitude towards security testing different systems changed radically during the last 5-10 years. Previously it was assumed that in closed systems (telecommunication is traditionally one of these) functionality and stability are the two most important properties to be tested since all participants of the system could be considered trusted. After multiple researchers demonstrated that security should indeed be taken seriously, security testing started to gain popularity and nowadays almost any kind of system that goes into production needs to be rigorously tested and evaluated. For this research two testing techniques were chosen to be applied: fuzzing and protocol analysis.

7.1 Fuzzing

Fuzzing is both a robustness and security testing technique that involves generating different invalid inputs for a system and then evaluating the reaction after feeding them to the system. There are many different ways to do fuzzing from completely random to very specific. In the case of random fuzzing the case generator (the fuzzer) has no, or very limited knowledge of the target protocol, so it basically just spits out random bits as different cases. It could be very effective to find parsing bugs, or just to test the robustness of the target. Specific fuzzing on the other hand is carried out using a lot of information about the target, because it targets only a specific part of it, for example a single field of a message (e.g. protocol code). This could be used to find boundary check errors, edge cases, or implementation errors in the main logic of the application.

7.2 Protocol Analysis

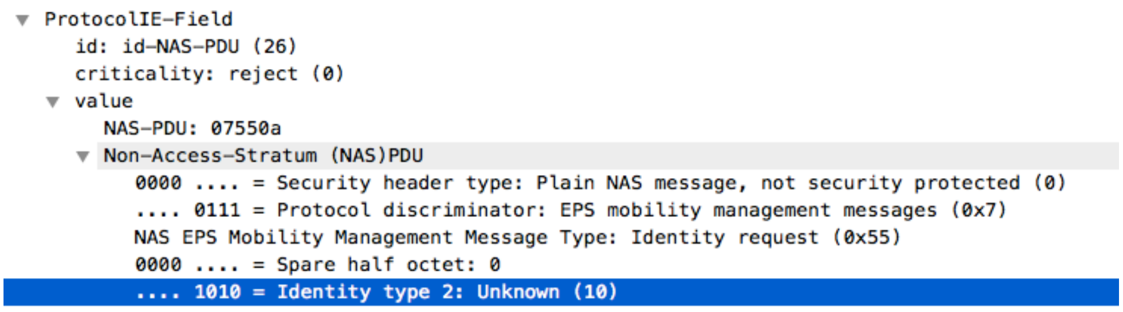
Protocol analysis is a technique that uses a static, human verification of the protocol specification to find logical errors or undefined edge cases by combining possibly multiple lower severity problems. The bugs resulting from this are usually high severity problems, because they are implementation independent so all devices following the specifications of the protocol are affected by them.

8. Results

I could reproduce the vulnerabilities researched in the analysis part, created a proof of concept fuzzing class and also was able to test out Cell Broadcast Services successfully. All of testing was done in a proper, isolated environment to avoid any kind of interference with commercial LTE networks.
The following vulnerabilities were responsibly disclosed by their respective authors to manufacturers.

8.1 Vulnerability #1 – Unauthorized IMEI disclosure

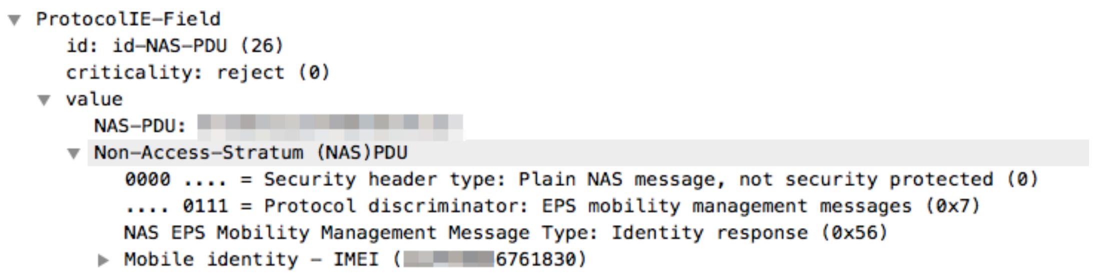
I could verify this bug easily: setting the Identity Type to an arbitrary value is easily achieved using doMME. Since an Identity Request could be sent anytime by the network I didn't need to design a specific flow of messages, instead I just set up my system to reply with an Identity Request to any kind of message arriving from the UE.



```
▼ ProtocolIE-Field
    id: id-NAS-PDU (26)
    criticality: reject (0)
  ▼ value
      NAS-PDU: 07550a
    ▼ Non-Access-Stratum (NAS)PDU
        0000 .... = Security header type: Plain NAS message, not security protected (0)
        .... 0111 = Protocol discriminator: EPS mobility management messages (0x7)
        NAS EPS Mobility Management Message Type: Identity request (0x55)
        0000 .... = Spare half octet: 0
        .... 1010 = Identity type 2: Unknown (10)
```

Figure 8.1 - Identity Request, Type 0b1010

After sending the message seen on Figure 8.1 I received a reply shown on Figure 8.2.



```
▼ ProtocolIE-Field
    id: id-NAS-PDU (26)
    criticality: reject (0)
  ▼ value
      NAS-PDU: [redacted]
    ▼ Non-Access-Stratum (NAS)PDU
        0000 .... = Security header type: Plain NAS message, not security protected (0)
        .... 0111 = Protocol discriminator: EPS mobility management messages (0x7)
        NAS EPS Mobility Management Message Type: Identity response (0x56)
      ▶ Mobile identity - IMEI ([redacted]6761830)
```

Figure 8.2 - Reply from a vulnerable UE containing the IMEI

As a testing measure I decided to try the Identity Type value 0b0010 or decimal 2 against all the devices. This is the defined value in the standard for IMEI, and should be discarded by all devices because of missing integrity protection (and non-emergency bearer). However, one of the devices ignored the problems and replied with its IMEI, which is clearly an implementation error of the state machine inside the baseband.

8.2 Vulnerability #2 – Targeted, location dependent Denial of Service

Exploiting this vulnerability could be done via any EMM message that has a cause Information Element in it. If the victim UE sends an Attach Request an Attach Reject could be used. The best way for an attacker to force a victim into sending a request is using Tracking Areas to his advantage.

Tracking Areas in LTE are groups of cells that are located in the same geographical area. The core network keeps track of all UEs and their current Tracking Areas so whenever there is a need to deliver service to any one of them the network is able to find it. To always keep an up to date Tracking Area Code assigned to a UE in the core network UEs need to send a so called Tracking Area Update Request every time they change to a different Tracking Area while in idle mode. If the UE is not idle there is no need to inform the network about changing tracking areas because in such a case a handover takes place, which is network coordinated (to ensure seamless transitions and continuous service). In idle mode however each UE is free to decide the cell it camps on based on the quality of the received signal and other parameters, therefore if it happens to change to a new Tracking Area it needs to inform the network.

This leads to the following attack scenario: an attacker chooses a random Tracking Area Code value for its fake eNodeB and also defines the Reject Cause to be used. Any UE that is in the vicinity of this fake cell could potentially decide to switch over to it. Once a UE decided to use the attacker controlled network instead of the real one it'll automatically try to send a Tracking Area Update Request message. The attacker needs to simply reject this message with the cause of his liking. Based on the cause value it could mean a simple downgrade attack (to 3G for example), a temporary loss in service or a permanent denial of service via putting the UE into EMM deregistered state. All of the reject messages are unprotected because if a network tries to reject an unknown UE then there is not necessarily a key available for it in the HSS.

This attack could also be extended to a location dependent denial of service. The attacker just needs to use a valid Tracking Area Code when setting up the fake cell, and set the reject cause to Tracking Area not allowed. In this case once the victim approached the fake cell it will store the Tracking Area Code in the SIM card as a not allowed area, meaning that if anytime later the UE gets into the Tracking Area the attacker forged it'll automatically lose connectivity to the network. It is like a bomb, planted in the SIM card that just waits for the right trigger to go off – possibly far away from the attacker, making it harder to detect an ongoing attack.

It is interesting to note, that the exact same kind of attack is possible on GSM networks as the author of this thesis pointed it out in [22]. It seems like this legacy of GSM still carries on in LTE.

This vulnerability may be thought as an inconvenience, because the victim can simply reboot his device or toggle airplane mode to get connected again, but one must not forget that not necessarily all devices connected to the network are user controlled and easily reachable. For example machine to machine communication that goes through LTE could be affected as well, and rebooting the mobile broadband component of those systems might present a rather great challenge, especially when the affected machine or sensor is located far away.

8.3 IMSI catcher

Building on the previous attack the authors (Ravishankar Borgaonkar and Altaf Shaik) went further and described the first IMSI catcher type of device that works on LTE. IMSI catching has been used by law enforcement agencies since the introduction of GSM. Their main goal is to determine whether a certain criminal is at a location or not. Usually IMSI catching is done during a major event that draws a lot of people to the same location. Law enforcement uses fake base-stations to collect the IMSIs (International Mobile Subscriber Identities) of all present devices and they try to find matches for criminals whose IMSI is known to them. If there is a match they can locate the suspect using the so called silent-call technique which simply forces the found device to transmit a high power beacon frame on a given frequency and timeslot. This used to be done using GSM, because in 3G and above the standards supposedly closed the possibility of creating a fake base-station by using mandatory integrity protection. However as it has been demonstrated chaining together the previous protocol vulnerability with another error one can create a fake LTE cell that is capable of collecting IMSIs.

Before describing the second vulnerability and combining it with the previous one an issue needs to be addressed, namely: how would UEs find the fake cell, and how can it be assured that they will select it? In GSM this was rather easy: one needed to operate on a frequency that the ME monitored (one of the broadcasted neighbor frequencies), and then simply manipulating the so called C1 and C2 values a fake station could present itself as the best option even though its signal power and quality wasn't necessarily better than the others'.

In LTE this system was changed, they introduced a system called 'absolute priority based cell reselection'. *"The principle of priority-based reselection is that UEs, in the IDLE state, should periodically monitor and try to connect to eNodeBs operated with high priority frequencies [24]. Hence even if the UE is close to a real eNodeB, operating the rogue eNodeB on a frequency that has the highest reselection priority would force UEs to attach to it. These priorities are defined in SIB Type number 4, 5, 6, and 7 messages broadcast by the real eNodeB [5]. Using passive attack setup, we sniff these priorities and configure our eNodeB accordingly."* (section IV/B of [11]).

Based on this information I modified srsUE, an example program that acts as a UE towards an eNodeB, to decode the required System Information Blocks for us. This was done to ensure the feasibility of the attack, because in the laboratory environment there was no need to tune to a different frequency – the isolation caused UEs to lose all connectivity, so as soon as I started the fake network they immediately tried to attach to it.

In the previous chapter it was proven sending a reject message with an arbitrary cause to the victim phone can be achieved without any problems. However, as it has been mentioned before, to collect an IMSI an Identity Request must be sent to the UE. The issue with this is the following: if a UE already has a security context then it should never accept unprotected NAS messages. Even though during the first attach procedure the UE must accept an unprotected Identity Request message (so the HSS knows which key should be fetched from the database) as soon as the security context has been established it will drop all unprotected messages, including an unprotected Identity Request. This makes the IMSI catcher attack far less effective, because the number of people turning on their devices for the first time to connect to the network while being in the vicinity of the IMSI catcher is low. On the other hand people with idle mode devices could walk into the range of the catcher easily. If there was a way to make a UE delete an already established security context and make it try to establish it again it would be possible to collect the IMSI of it easily. In [11] the authors describe such a method, which is surprisingly simple, but according to the specifications it should work – and it works in reality as well. The problem lies again with the EMM causes which could be used freely to change the state of the UE. It is described in [7] that if a UE receives a reject message with the cause #9 – "UE identity can't be derived by the network" it needs to immediately delete its stored security context. This cause was probably added to handle the case of an MME crash when all data is lost, and UEs need to create a new security context for themselves. In this case though it is exactly the type of message needed: it deletes the stored security context without it requiring any protection or encryption applied to this message. After this message is sent the UE will automatically initiate a new Attach procedure allowing us to reply with an unprotected Identification Request. After the UE replied with its IMSI it's time to create a fix to the other problem – namely how does the victim get pushed back to his original network so he doesn't realize his IMSI was captured. The easiest way in my opinion is to send an Attach Reject with the cause #12 - "Tracking Area not allowed". That would cause them to go back to the original network without any issues.
To see the complete message flow see Figure 8.3.

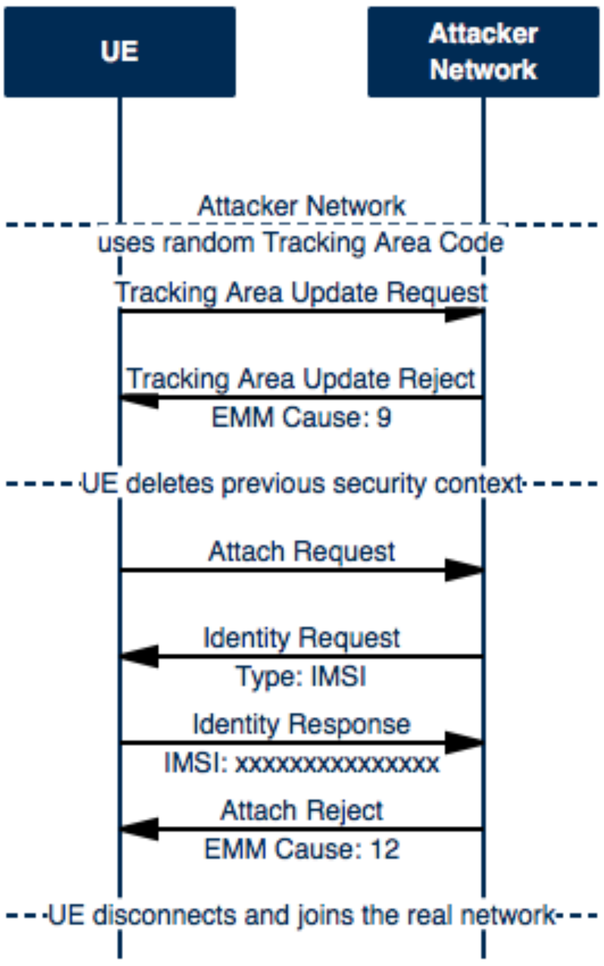
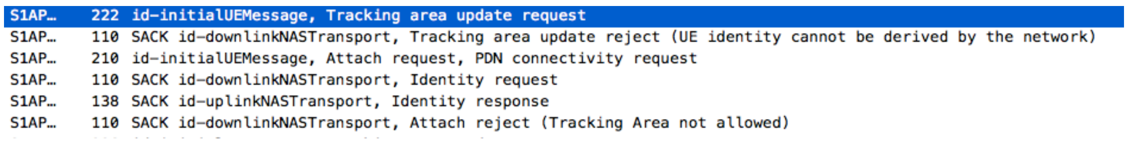

UE
Attacker Network
Attacker Network uses random Tracking Area Code
Tracking Area Update Request
Tracking Area Update Reject
EMM Cause: 9
UE deletes previous security context
Attach Request
Identity Request
Type: IMSI
Identity Response
IMSI: xxxxxxxxxxxxxxx
Attach Reject
EMM Cause: 12
UE disconnects and joins the real network

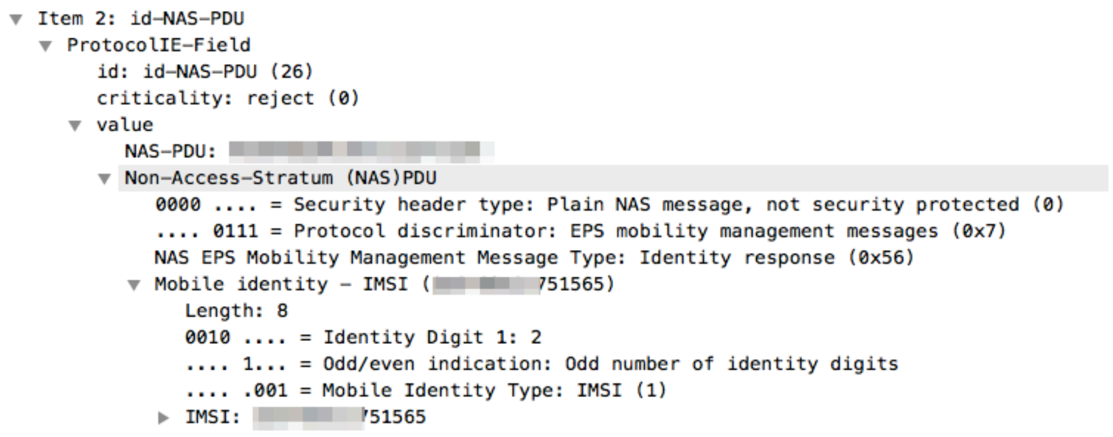
Figure 8.3 - IMSI catcher message flow

This IMSI catcher was tested as well, and it successfully caught multiple IMSIs. The packets could be seen in Wireshark on Figure 8.4 and 8.5.



```
S1AP…   222  id-initialUEMessage, Tracking area update request
S1AP…   110  SACK id-downlinkNASTransport, Tracking area update reject (UE identity cannot be derived by the network)
S1AP…   210  id-initialUEMessage, Attach request, PDN connectivity request
S1AP…   110  SACK id-downlinkNASTransport, Identity request
S1AP…   138  SACK id-uplinkNASTransport, Identity response
S1AP…   110  SACK id-downlinkNASTransport, Attach reject (Tracking Area not allowed)
```

Figure 8.4 - Packet capture of catching an IMSI



```
▼ Item 2: id-NAS-PDU
  ▼ ProtocolIE-Field
      id: id-NAS-PDU (26)
      criticality: reject (0)
    ▼ value
        NAS-PDU: [redacted]
      ▼ Non-Access-Stratum (NAS)PDU
          0000 .... = Security header type: Plain NAS message, not security protected (0)
          .... 0111 = Protocol discriminator: EPS mobility management messages (0x7)
          NAS EPS Mobility Management Message Type: Identity response (0x56)
        ▼ Mobile identity - IMSI ([redacted]751565)
            Length: 8
            0010 .... = Identity Digit 1: 2
            .... 1... = Odd/even indication: Odd number of identity digits
            .... .001 = Mobile Identity Type: IMSI (1)
          ▶ IMSI: [redacted]751565
```

Figure 8.5 - Identity Response containing the target's IMSI

8.4 Fuzzing results

Limitations of the available time frame caused us not to go into a lot of fuzzing, but I created a proof of concept code. DoMME was extended to be able to generate new messages instead of taking them from a pre-defined configuration file. Also a Python helper program was created that is capable of toggling the airplane mode on an Android phone connected via USB, so if one of the phones stops responding during testing it could be restarted without human interaction. The proof of concept code simply goes through the EMM cause field in a Tracking Area Update Reject from 0 to 255.

Test results didn't yield anything special – all of the tested UEs' securely implemented the handling of this particular field even when undefined values were used.

8.5 CBS

The last feature I inspected was the Cell Broadcast Service. This feature has been present both in GSM and in 3G. It is a simple way to deliver information to all devices located in a certain area via a broadcast (one-way) mechanism. LTE brings support to multiple different public broadcast systems present in different countries around the world: ETWS (Earthquake and Tsunami Warning System – Japan, and some parts of Asia), CMAS (Commercial Mobile Alert System – USA), KPAS (Korean Public Alert System) and EU-Alert (Europe). These are all used in emergency scenarios to warn the population of an area about an imminent threat.

The system architecture could be seen on Figure 8.6.

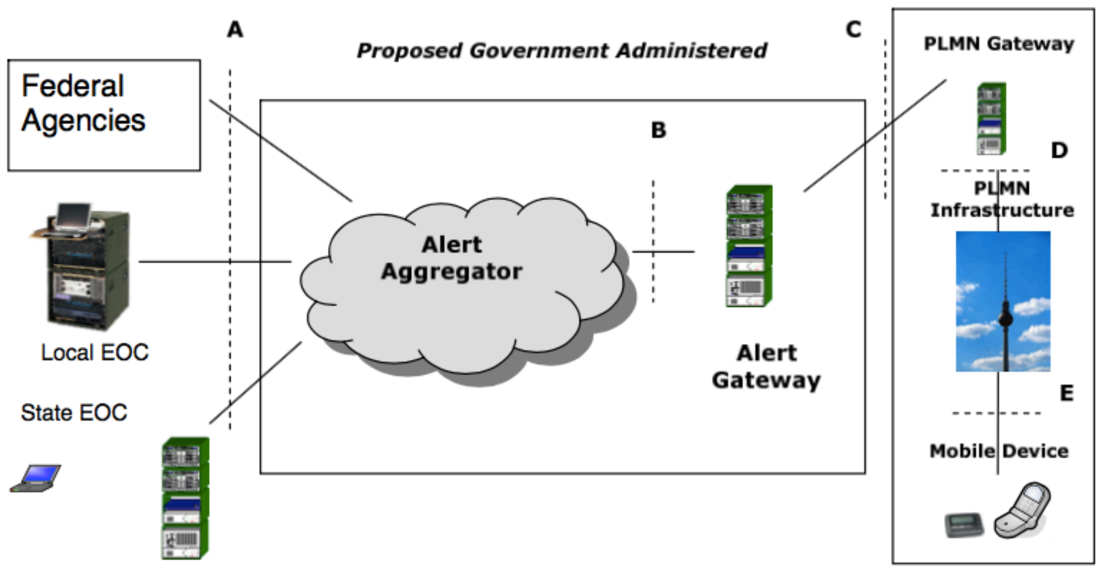


Figure 8.6 - PWS Architecture. From Chapter 6.1 of [23]

In LTE the main goal was to lower the latency of the system. The commission's target for latency was maximum 8 seconds between the alert being issued from one of the authorized agencies till it's displayed on the affected people's devices. They were able to meet this goal by defining System Information blocks (10 and 11) to carry the alert data. Since SIBs are always constantly broadcasted they are ideal for the alert systems. According to [23] (chapter 4.6.1) the following should be done when a UE receives emergency broadcast data:

- UE should use a dedicated alerting indication (audio and vibration) reserved only for Warning Notification purposes
- UE should display the message on the screen
- UE should stop the alert after user interaction is detected (buttons pushed, screen touched etc.)

Naturally I looked at this whole system from a security point of view, and it was discovered, that these messages are not authenticated or encrypted at all making them susceptible to forgery. Using the FlexiBTS (OAI currently has no support for this system) connected to doMME I was able to create an alert with just a few lines of code. The alert was displayed on the UE's screen and a loud, audible sound could be heard as well. It is important to note, that support for these systems is only added in Android 6.0 or above and in iOS 9 and above. Most alerts could be turned off in the device's settings, but according to US regulations the CMAS Presidential Alerts can't be disabled. However it must be noted that when the device is muted the alerting sound wasn't audible.

In summary: an attacker with a fake eNodeB and little bit of coding can send arbitrary text messages to all LTE enabled devices in its vicinity. The message(s) will definitely show up on the device, they can't be ignored and they can not be blocked. This opens up a whole new range of possibilities for spamming, phishing and social engineering. The mobile industry has been fighting against spam and phishing attacks originating from fake GSM base stations for the last couple of years. Before my analysis it was assumed that LTE would not be affected by such a problem, but this research proves that there are still ways to perform such fraudulent attacks.

CBS also features a possibility to initiate and carry out binary downloads to SIM cards in the area. This feature has already been demonstrated to be exploitable for malicious purposes if the SIM card isn't properly configured [24], although the use of binary SMS messages is a more feasible scenario then using a fake LTE cell with CBS to achieve the same result.

Warning systems in general are great, they truly could save lives in many cases by delivering information immediately to the masses. On the other hand misusing them could lead to serious problems, mass panic and possibly death of people as well. It has been demonstrated that only with a few lines of code it is possible to create a fake alert message. After that it could be distributed via fake LTE cells or even injected into a real, commercial network. In the least harmful way it is a free text message, that could be used for distributing spam (and it can't be blocked in any way) in the most harmful way it could mislead people and cause panic. For example imagine a scenario in which a presidential alert is sent out that there is going to be a nuclear attack in the middle of Manhattan.

An example of an empty presidential alert could be seen on Figure 8.7 (the device translated the alert's type to Hungarian, the caption means Presidential alert).

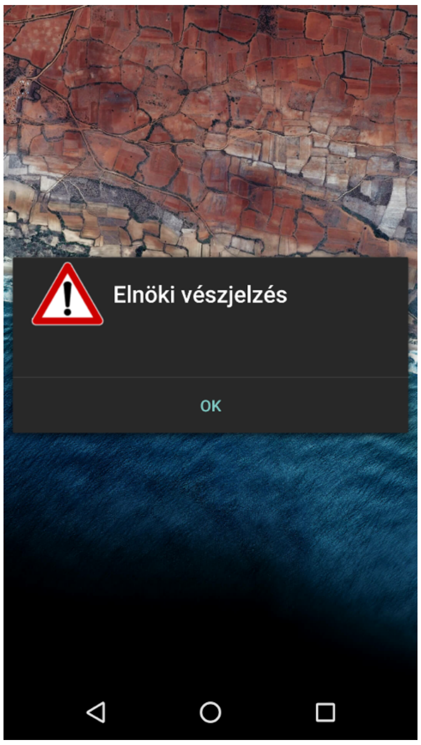


Figure 8.7 - Android 6.0 device displaying a CMAS - Presidential alert

9. Mitigations

I have shown multiple type of vulnerabilities in the previous chapter, now I would like to offer some ways to mitigate them. Implementation errors that caused the first vulnerability (IMEI leak) are fixed by vendors, so by updating to the latest baseband software version the vulnerability should be patched. Protocol errors on the other hand are not fixed easily, changing the way the standard is built would be very costly (all current equipment would need to be updated or replaced), however it is worth noting what should be changed:

To avoid Denial of Service all rejection messages should be treated with an exponential back off. This means once a device received a rejection it shouldn't stay in EMM deregistered state until it is rebooted, but instead it should keep trying after some time passed to connect again to the network. The waiting time should be increased

exponentially to relieve the network's load if the device is indeed should not be connected.
Avoiding IMSI catching is currently not possible, the way an established security context could be invalidated and then an Identity Request sent can't be altered because there are legitimate cases in which this procedure is needed. It needs to be noted though that the situation in which the UE is rejected because its identity can't be derived should be a rare occasion, so at least there should be a notification about it, so the user knows about it and can act upon it if necessary.
The issues surrounding CBS are serious and they require action. Adding a digital signature to the messages that is signed by a certificate which has been added to the baseband chips via software updates would significantly lower the threat of fake messages being delivered to the people.

10. Conclusions and future work

In the previous chapter I have shown multiple security weaknesses in the LTE-Uu interface. These weaknesses aren't compromising the security of the complete system since LTE's cryptographic foundations are currently considered to be solid. However multiple implementation and protocol errors showed that LTE isn't fault free, even though it is considerable better than its ancestors. I think that LTE needs to be constantly improved and carriers need to be forced to update their infrastructure and test their security regularly otherwise independent minor bugs could be chained together to possibly bring the whole system down.
In the future the fuzzing capabilities of doMME needs to be extended, it should be capable of testing different messages, generate and inject fully random binary blobs. Another good feature would be out of sequence message sending to see how a device reacts if it received an unexpected message in the middle of a procedure (e.g. SMS in the middle of an attach).

It needs to be noted that I only targeted one part of the interface, namely UEs via fake eNodeBs. In the future the other part of the interface needs to be evaluated, namely attacking network components via a rogue UE.

11. Acknowledgments

I'd like to thank all the people who helped and inspired me during this research: Nico Golde and Kevin Redon, Dr. Márton Csapodi from PPCU and my advisor László Szűcs who worked hard to keep me on track and in schedule, my parents and my family for their constant support and God for giving me the strength and patience needed to finish my thesis.

Bibliography

[1] GSMA. (2016, October) Global mobile trends. [Online]. Available: https://www.gsmaintelligence.com/research/2016/10/global-mobile-trends/580/

[2] 3GPP. 3GPP. [Online]. Available: http://www.3gpp.org

[3] 3GPP/ETSI. TS 36.423 - X2 Application Protocol (X2AP). [Online]. Available: http://www.etsi.org/deliver/etsi_ts/136400_136499/136423/13.05.00_60/ts_136423v130500p.pdf

[4] 3GPP/ETSI. TS 36.413 - S1 Application Protocol (S1AP). [Online]. Available: http://www.etsi.org/deliver/etsi_ts/136400_136499/136413/13.02.00_60/ts_136413v130200p.pdf

[5] 3GPP/ETSI. TS 23.002 - Network architecture. [Online]. Available: http://www.etsi.org/deliver/etsi_ts/123000_123099/123002/13.05.00_60/ts_123002v130500p.pdf

[6] 3GPP/ETSI. TS 36.331 - Protocol specification. [Online]. Available: http://www.etsi.org/deliver/etsi_ts/136300_136399/136331/13.01.00_60/ts_136331v130100p.pdf

[7] 3GPP/ETSI. TS 24.301 - Non-Access-Stratum (NAS) protocol for Evolved Packet System (EPS). [Online]. Available: http://www.etsi.org/deliver/etsi_ts/124300_124399/124301/13.05.00_60/ts_124301v130500p.pdf

[8] Pierre-Alain Fouque, Gilles Macario-rat, Cristina Onete and Benjamin Richard Stephanie Alt. A Cryptographic Analysis of UMTS/LTE AKA. [Online]. Available: https://eprint.iacr.org/2016/371.pdf

[9] Christophe Devine Benoit Michau. How to not break LTE crypto. [Online]. Available: https://www.sstic.org/media/SSTIC2016/SSTIC-actes/how_to_not_break_lte_crypto/SSTIC2016-Article-how_to_not_break_lte_crypto-michau_devine.pdf

[10] 3GPP/ETSI. TS 24.008 - Core network protocols. [Online]. Available: http://www.etsi.org/deliver/etsi_ts/124000_124099/124008/13.05.00_60/ts_124008v130500p.pdf

[11] Altaf Shaik, Ravishankar Borgaonkar, N. Asokan, Valtteri Niemi, Jean-Pierre Seifert. Practical Attacks Against Privacy and Availability in 4G/LTE Mobile Communication Systems. [Online]. Available: https://www.internetsociety.org/sites/default/files/blogs-media/practical-attacks-against-privacy-availability-4g-lte-mobile-communication-systems.pdf

[12] Philippe Langlois. Hacking HLR HSS and MME Core Network Elements. [Online]. Available: https://conference.hitb.org/hitbsecconf2013ams/materials/D1T2%20-%20Philippe%20Langlois%20-%20Hacking%20HLR%20HSS%20and%20MME%20Core%20Network%20Elements.pdf

[13] Chuck McAuley. DEFCON 24 - LTE is pretty fragile. [Online]. Available: https://cdn.shopify.com/s/files/1/0177/9886/files/phv2016-cmoorecmcauley.pdf

[14] Osmocom. Osmocom. [Online]. Available: http://osmocom.org/

[15] Ben Wojtowicz. OpenLTE. [Online]. Available: http://openlte.sourceforge.net

[16] SoftwareRadioSystems Ltd. srsLTE. [Online]. Available: https://github.com/srsLTE

[17] Fabrice Bellard. LTE Base Station Software. [Online]. Available: http://bellard.org/lte/

[18] EURECOM. Open Air Interface. [Online]. Available: http://www.openairinterface.org

[19] Philippe Langlois Elvis Pfützenreuter. pySCTP. [Online]. Available: https://github.com/philpraxis/pysctp

[20] Benoit Michau. libmich. [Online]. Available: https://github.com/mitshell/libmich

[21] Benoit Michau. CryptoMobile. [Online]. Available: https://github.com/mitshell/CryptoMobile

[22] Domonkos P. Tomcsányi. Hacktivity - Yet another way to cause DoS for GSM devices. [Online]. Available: https://hacktivity.com/en/downloads/archives/365/

[23] 3GPP/ETSI. TS 22.268 - Public Warning System (PWS) requirements. [Online]. Available: http://www.etsi.org/deliver/etsi_ts/122200_122299/122268/13.00.00_60/ts_122268v130000p.pdf

[24] Karsten Nohl. Rooting SIM cards. [Online]. Available: https://media.blackhat.com/us-13/us-13-Nohl-Rooting-SIM-cards-Slides.pdf

Appendix

Appendix A - List of acronyms

3GPP – 3rd Generation Partnership

ADC – Analog Digital Converter

ARFCN – Absolute Radio Frequency Channel Number

CBS – Cell Broadcast Services

CMAS – Commercial Mobile Alert System

GSM – Global System for Mobile Communication

EPC – Evolved Packet Core

ESM – EPS Session Management

EMM – EPS Mobility Management

ETWS – Earthquake and Tsunami Warning System

EUTRAN – Evolved UTRAN (see UTRAN)

FDD – Frequency Division Duplex

GTP – GPRS Tunneling Protocol

HSS – Home Subscriber Server

IMEI – International Mobile Equipment Identity

IMS – IP Multimedia Subsystem

IMSI – International Mobile Subscriber Identity

IP – Internet Protocol

LTE/4G – Long Term Evolution

MME – Mobility Management Entity

OAI –Open Air Interface

P-GW – Packet Gateway

SCTP – Session Control Transmission Protocol

S-GW – Serving Gateway

SIM – Subscriber Identity Module

TDD – Time Division Duplex

UE – User Equipment

UMTS/3G – Universal Mobile Telecommunications System

Appendix B – List of annexes

The attached optical disc contains the following:

- a copy of this document in PDF format
- complete source code of doMME (also available on github: https://github.com/domi007/doMME)
- figures
- a copy of all works cited