



**Università  
di Genova**

**MSc Computer Engineering**  
Software Platforms & Cybersecurity Curriculum

---

**Enhancing Security in 3GPP Compliant  
Applications: a Modular Testbed and Threat  
Scenarios for Mission Critical  
Communication Systems**

Flavio Bava

Academic Year 2023/2024

*Advisor*

**Prof. Enrico Russo**  
University of Genoa

*Co-advisor*

**Ing. Davide Desirello**  
Leonardo Cyber Security  
R&D Laboratory

# Abstract

Public Protection and Disaster Relief situations have always been a hard challenge to face in terms of communication. In such situations, real-time interactions among first responders and various agencies can make the difference between life and death. Unfortunately, standard communication networks don't cut it in crises because natural disasters, severe storms, power outages, and cyberattacks can disrupt them. So, what's the solution?

Robust, resilient networks that facilitate Mission Critical communications, which refers to the need to provide first responders with fast, reliable, secure communication at all times.

The proposed work focuses on security aspects of Mission Critical applications by identifying potential vulnerabilities that could represent a serious risk for the organization and its users. The objective is to provide a series of scenarios, fully related to components and procedures compliant with the Third Generation Partnership Project (3GPP) standards, that can assist other vendors in identifying potential vulnerabilities or misconfigurations in their products. The valuable aspect of this work lies in the fact that the Mission Critical domain, with its associated technologies, was not only learned from 3GPP Technical Specification (TS) documents but also with the help of a real-world application, kindly provided by Leonardo S.p.a.

# Table of Contents

<b>List of Figures</b>	<b>7</b>
<b>List of Tables</b>	<b>9</b>
<b>List of Abbreviations</b>	<b>10</b>
<b>1 Introduction</b>	<b>15</b>
<b>2 Background</b>	<b>20</b>
2.1 Mission Critical Services . . . . .	20
2.1.1 Mission Critical Push-To-Talk (MCPTT) . . . . .	20
2.1.2 Mission Critical Video (MCVideo) . . . . .	21
2.1.3 Mission Critical Data (MCData) . . . . .	22
2.2 System Description . . . . .	22
2.2.1 Application Plane . . . . .	23
2.2.1.1 IDMS & IDMC - RP CSC-1 . . . . .	25
2.2.1.2 GMS & GMC - RP CSC-2 . . . . .	25
2.2.1.3 CMS & CMC - RP CSC-4 . . . . .	25
2.2.1.4 KMS & KMC - RP CSC-8 . . . . .	26
2.2.1.5 MC service client/server . . . . .	26
2.2.2 Signaling Control Plane . . . . .	26
2.2.2.1 SIPCore . . . . .	28
2.2.2.2 Signaling User Agent & SIPCore - SIP-1 RP . . . . .	29
2.2.2.3 SIP core & the SIP AS - SIP-2 RP . . . . .	30

2.2.2.4	SIP database & the SIP core - AAA-1 RP . . . . .	30
2.2.2.5	HTTP Client & HTTP Proxy - HTTP-1 RP . . . . .	30
2.2.2.6	HTTP Proxy & HTTP Server - HTTP-2 RP . . . . .	31
2.3	Procedures & Flows . . . . .	31
2.3.1	Bootstrap Procedure . . . . .	32
2.3.2	Authentication & Authorization Procedure . . . . .	32
2.3.2.1	MCX User Authentication . . . . .	34
2.3.2.2	MCX User Service Authorization . . . . .	34
2.3.3	MC service operations . . . . .	35
2.4	OpenID Connect (OIDC) . . . . .	36
2.4.1	What is OAuth? . . . . .	36
2.4.2	OIDC vs OAuth . . . . .	36
2.4.3	How does OpenID Connect work? . . . . .	37
2.4.3.1	OpenID Connect Roles . . . . .	37
2.4.3.2	OpenID Connect claims and scopes . . . . .	38
2.4.4	OAuth Grant Types . . . . .	39
2.4.5	Authorization Code Grant Type . . . . .	39
2.4.6	Client Registration . . . . .	44
2.5	Session Initiation Protocol (SIP) . . . . .	45
2.5.1	SIP Methods . . . . .	45
2.6	What is a threat analysis? . . . . .	46
2.6.1	STRIDE . . . . .	47
<b>3</b>	<b>State of the Art</b>	<b>49</b>
3.1	3GPP TS & TR documents . . . . .	50
3.2	Involved Technologies . . . . .	50
<b>4</b>	<b>Testbed</b>	<b>52</b>
4.1	Overall Architecture . . . . .	52
4.2	Security Zones . . . . .	55

4.2.1	Public Network . . . . .	56
4.2.1.1	5G Radio Access Network . . . . .	56
4.2.1.2	5G Core Network . . . . .	56
4.2.2	Management Network . . . . .	57
4.2.3	Microservices Network . . . . .	57
4.3	Tools . . . . .	57
4.3.1	5G RAN . . . . .	58
4.3.2	5G Core Network . . . . .	59
4.3.3	Testing . . . . .	60
<b>5</b>	<b>Scenarios</b>	<b>62</b>
5.1	Choice of entities . . . . .	62
5.2	Assumptions . . . . .	65
5.3	Overview . . . . .	66
5.4	Description . . . . .	68
5.4.1	IDMS - Unprotected Dynamic Client Registration . . . . .	68
5.4.1.1	TS_01 Authorization_Code theft . . . . .	68
5.4.1.2	TS_02 Credentials theft through malicious redirect_uri . . . . .	70
5.4.1.3	TS_03 SSRF through logo_uri . . . . .	70
5.4.2	TS_04 SIPCore - SIP REGISTER Flooding . . . . .	71
5.4.3	TS_05 SIPCore - IP Spoofing through NAT re-assignment . . . . .	71
5.4.4	TS_06 SIPCore - P-CSCF Bypass . . . . .	71
5.4.5	TS_07 File Repository - Path Traversal . . . . .	72
5.4.6	TS_08 Information Disclosure via Bootstrap Service . . . . .	72
5.4.7	TS_09 - Tampering SRTP authentication tag . . . . .	73
5.5	Testing . . . . .	73
5.5.1	TS_01 . . . . .	73
5.5.2	TS_02 . . . . .	75
5.5.3	TS_03 . . . . .	76
5.5.4	TS_04 . . . . .	78

5.5.5	TS_07	79
5.5.6	TS_08	80
<b>6</b>	<b>Experimental evaluation</b>	<b>82</b>
6.1	Ethical Considerations	82
6.2	Reconnaissance	83
6.3	Attack Demonstration	86
<b>7</b>	<b>Discussion</b>	<b>94</b>
<b>8</b>	<b>Conclusion</b>	<b>96</b>
8.1	Future Works	97

# List of Figures

1.1	MCS Timeline . . . . .	16
2.1	Common functional model for Application Plane . . . . .	24
2.2	Functional model for signaling control plane . . . . .	27
2.3	UE general lifecycle using an MC service . . . . .	31
2.4	MCX authentication & authorization . . . . .	33
2.5	Login with OIDC options . . . . .	37
2.6	OIDC flow with Username & password authentication . . . . .	40
4.1	Overall network architecture . . . . .	53
4.2	Testbed's Security Zones . . . . .	55
5.1	Example of application that uses logo_uri . . . . .	77
6.1	SIPCore port discovery . . . . .	83
6.2	Sippts scan command . . . . .	84
6.3	Sippts REGISTER request & response . . . . .	85
6.4	Registration endpoint discovery . . . . .	86
6.5	Client Registration with malicious redirect_uri . . . . .	87
6.6	Attack Flow Diagram . . . . .	88
6.7	Tampered authentication request . . . . .	89
6.8	LocalStorage with code_verifier . . . . .	90
6.9	IDMS login form . . . . .	91
6.10	Fake page with Network Error . . . . .	92

6.11 Request with authorization_code sent to attacker's server . . . . .	92
6.12 Token request sent by the attacker & response from IDMS . . . . .	93



# List of Tables

2.1	Authentication Request required parameters . . . . .	41
2.2	Access Token request parameters . . . . .	43
2.3	Access Token Response parameters . . . . .	44
5.1	List of Scenarios . . . . .	67

# List of Listings

1	TS_01 - Unprotected Dynamic Client Registration Request . . . . .	74
2	TS_02 - Unprotected Dynamic Client Registration Request . . . . .	75
3	TS_03 - Unprotected Dynamic Client Registration Request . . . . .	76
4	Example response of UE initial configuration document . . . . .	81
5	Example response of configuration document with sensitive information . .	81

# List of Abbreviations

<b>3GPP</b>	.....	3rd Generation Partnership Project
<b>5G</b>	.....	Fifth Generation
<b>API</b>	.....	Application Program Interface
<b>CMC</b>	.....	Configuration Management Client
<b>CMS</b>	.....	Configuration Management Server
<b>CN</b>	.....	Core Network
<b>CSC</b>	.....	Common Services Core
<b>CSRF</b>	.....	Cross-Site Request Forgery
<b>D2D</b>	.....	Device-To-Device
<b>DFD</b>	.....	Data Flow Diagram
<b>DoS</b>	.....	Denial of Service
<b>GCS AS</b>	.....	Group Communication Service Application Server
<b>GMC</b>	.....	Group Management Client
<b>GMS</b>	.....	Group Management Server
<b>GPSDO</b>	.....	Global Positioning System Disciplined Oscillator
<b>gNb</b>	.....	gNodeb

**HTTP** ..... Hypertext Transfer Protocol

**I-CSCF** ..... Interrogating-Call Session Control Function

**IDMC** ..... Identity Management Client

**IDMS** ..... Identity Management Server

**IMPU** ..... IMS Public User Identity

**IMS** ..... IP Multimedia Subsystem

**JSON** ..... JavaScript Object Notation

**JWT** ..... JSON Web Token

**K8S** ..... Kubernetes

**KMC** ..... Key Management Client

**KMS** ..... Key Management Server

**LTE** ..... Long Term Evolution

**MCDData** ..... Mission Critical Data

**MCPTT** ..... Mission Critical Push-To-Talk

**MCS** ..... Mission Critical Services

**MCVideo** .... Mission Critical Video

**NAT** ..... Network Address Translation

**NF** ..... Network Function

**OIDC** ..... OpenID Connect

**OWASP** ..... Open Web Application Security Project

**P-CSCF** ..... Proxy-Call Session Control Function

**PKCE** ..... Proof Key for Code Exchange

**PLMN** ..... Public Land Mobile Network

**QoS** ..... Quality of Service

**R&D** ..... Research and Development

**RAN** ..... Radio Access Network

**RFC** ..... Request for Comments

**ROC** ..... Roll-Over Counter

**RP** ..... Reference Point

**RTP** ..... Real-Time Transport Protocol

**S-CSCF** ..... Serving-Call Session Control Function

**SDP** ..... Session Description Protocol

**SDR** ..... Software Defined Radios

**SIP** ..... Session Initiation Protocol

**SRTP** ..... Secure Real-time Transport Protocol

**SSRF** ..... Server-Side Request Forgery

**STRIDE** ..... Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privileges

**SUT** ..... System Under Test

**TCP** ..... Transmission Control Protocol

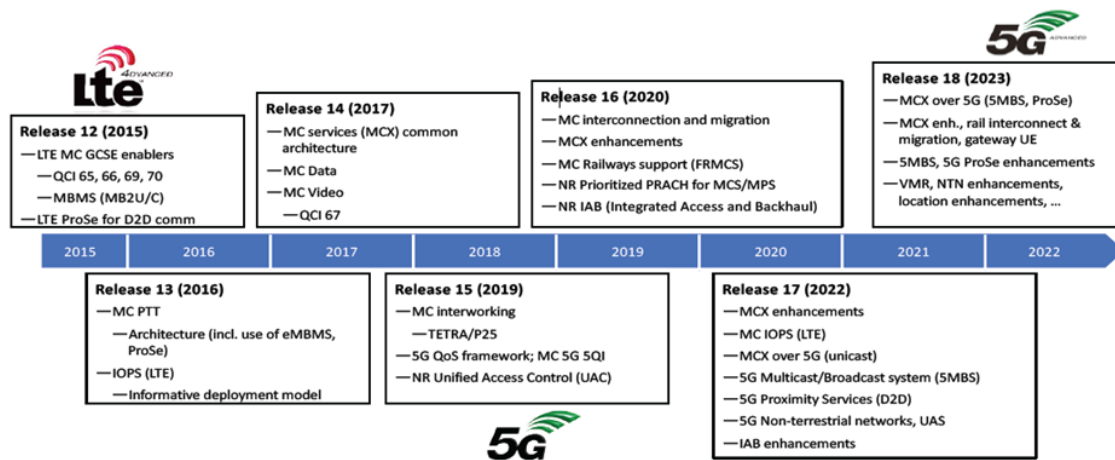
**TLS** ..... Transport Layer Security

**TR** ..... Technical Report  
**TS** ..... Technical Specification  
**UE** ..... User Equipment  
**UDP** ..... User Datagram Protocol  
**USIM** ..... Universal Subscriber Identity Module  
**VM** ..... Virtual Machine  
**XCAP** ..... XML Control Application Protocol  
**XML** ..... Extended Markup Language

# Chapter 1

## Introduction

In a time where Public Protection and Disaster Recovery (PPDR) situations are very frequent, public security forces need increasingly high-performance tools to intervene quickly and efficiently. Since the 1930s, Mission Critical communications have relied mainly on land mobile radio (LMR) systems, that consist of handheld radios (also known as “push-to-talk”), vehicle radios, base station radios, networks and repeaters. However, in recent years, several organizations in the telecommunication sector, have begun to invest in Mission Critical Services (MCS), in particular the 3rd Generation Partnership Project (3GPP), which started working on them in 2015, where Long Term Evolution (LTE) was the main technology. The advent of Fifth Generation (5G) networks in 2019, as shown in 1.1, with its improvements in various aspects, such as low-latency, high availability, and reliability, played a key role in the enhancement of MCS. MCx services can be seamlessly integrated into standard public mobile 5G networks, or alternatively, organizations have the option to establish privately owned infrastructures. The primary challenge faced by 3GPP during the standardization process revolved around determining the priority levels for each service. In earlier times, services like online gaming held a higher priority, but in current releases, MCx services have gained greater precedence. Consequently, public networks now ensure service availability, provided there is sufficient data throughput and coverage. The goal is to make users stand to gain by utilizing a single, secure device for a wide range of services, with virtually no geographical limitations, as long as network coverage exists.



**Figure 1.1:** MCS Timeline

The following description presents some use cases where MC applications could be implied.

**Public Safety** - A police department receives a call about an active shooter targeting a local university. As soon as this call arrives, they must organize a response team in minutes, dispatch the selected officers and medical personnel, and get to the scene as quickly as possible. Every minute that passes could mean another life is lost. In such a situation, Mission Critical communications must be guaranteed to coordinate the different teams and monitor the health conditions of each involved person. In particular, voice calls may be carried out between members of the same squad and between squad-leaders and the emergency coordination center, as they attempt to apprehend the active shooter. In addition, video calls may be carried out to show what first responders see during the operation, or by medical operators to show the condition of any injured individuals and, if necessary, prepare the hospital for possible operations on severely injured patients.



**State and Local Governments** - Suppose a hurricane hits a city. It might easily knock down cell towers, initiate a power outage, flood streets and cause immense physical destruction. Without Mission Critical communications, the following issues may arise:

- Agencies involved in rescue and recovery plans can't share information with each other.
- People injured during the hurricane call emergency forces but fail to get through.
- City managers struggle to connect with the appropriate transportation departments to clean up the roads.

Given a bit of context, one can imagine that a security hole in such systems could put at risk an entire operation, where human lives may be involved. Therefore it's essential to assess such applications.

## Motivation

As explained above Mission Critical Communications aims to provide first responders with fast, reliable, secure communication at all times. But what does mean secure? Many entities, not strictly related to audio/video communication, are involved in this type of application. For instance, if there's a vulnerability in Authentication procedures, that allows an attacker to impersonate one of the legit users in the system, bad consequences are very likely to happen. In addition, different kinds of technologies and protocols are present, spanning from telephony-related ones (eg. SIP), to TCP bi-directional communications (eg. Websockets), providing a good surface for an attacker. We focused only on entities that are described in the 3GPP Technical Specification documents (standard compliant), in order to make the work valuable for every Mission Critical application provider.

## Content of the Thesis

The thesis is composed of 7 more chapters, which are listed below with a brief description of their content:

**Chapter 2 - Background:** Explains which services are standardized and supported by MC applications and their main functionalities, the architecture of an MC application over a 5G system along with the division of its functional model between application and signaling plane and the main entities that compose these two planes. Next, it overviews the main procedures and flows that are carried out by a user who wants to make use of services provided by the MC organization. The final part of this chapter describes some technologies that are crucial for a Mission Critical application and highly implied in the proposed work.

**Chapter 3 - State of the Art:** Lists various resources that explore work that has already been done on security aspects of the MC domain and other works that are focused on specific technologies or protocols implicated in MC communication systems.

**Chapter 4 - Testbed:** Gives an overview of the testbed's network architecture, the identified subnets, and the characteristics of each of these subnets. Additionally, there is a description of both tools that were used to deploy the testbed infrastructure and the ones used to perform tests on it.

**Chapter 5 - Scenarios:** Starts with some considerations on the level of accessibility of the subnets mentioned in Chapter 4, what reasons brought us to focus the work on a single subnet, the Public Network, and a brief description of the entities, located within such a network, that were chosen to be analyzed. Next, there is a description of the identified scenarios, together with their mapping within the domain of the chosen threat modeling framework. The final part of the chapter presents some directives that can be used to test the identified scenarios or verify the preconditions for their reproducibility.

**Chapter 6 - Experimental Evaluation:** Showcases one of the proposed scenarios on a real, enterprise Mission Critical system, provided by the Leonardo company.

**Chapter 7 - Discussion:** Describes in a more conversational way the contributions brought by this work, already outlined in the **Contributions** section of this chapter. Additionally, there is a description of the proposed updates to the 3GPP standard together with the motivations that led us to this outcome.

**Chapter 8 - Conclusion:** Reports final considerations and potential future works that can be done on the Mission Critical domain.

## Contributions

The main contributions of this work are listed below.

1. The 3GPP standard for Mission Critical communication systems is extensive and highly detailed. A critical and in-depth study has highlighted the key security aspects and priorities to be addressed in the analysis of these systems.
2. A proposal of a fully functional testbed that can be used to assess standard MC applications, comprehensive of software/hardware technologies to deploy a Non-Public Stand-Alone 5G Network (i.e., RAN + Core), and useful tools to conduct security tests.
3. The development of threat scenarios based on both the studies of 3GPP technical documents and practical exploration of a real-world enterprise mission-critical (MC) system using the proposed testbed.
4. A major outcome of the above activity was the identification of potential weaknesses in the 3GPP standard specifications. As a result, we proposed updates to the 3GPP organization through their Coordinated Vulnerability Disclosure program to enhance Technical Specification Document 33.180 - Security of the Mission Critical service.

# Chapter 2

## Background

This chapter provides a brief description of **Mission-Critical services (MCS)**, the necessary background to understand the common functional architecture, procedures, and information flows, needed to support such services.

### 2.1 Mission Critical Services

As illustrated in 1.1, MCPTT was the first MC service to be standardized in 3GPP Release 13 (2016). Subsequent enhancements have been made over time, resulting in the current Mission Critical domain comprising three services: **MCPTT**, **MCVideo**, and **MCDData**. Below is presented an overview of these categories of Mission Critical services.

#### 2.1.1 Mission Critical Push-To-Talk (MCPTT)

MCPTT refers to a method by which first responders use a Smartphone as a walkie-talkie. Push-to-talk as the name states, allows two or more users to communicate with each other by pressing a button on their smartphones. Indeed, the MCPTT Service supports both private calls between pairs of users and group communication between several users in the same group. MCPTT allows users to request permission to talk and provides a mechanism to handle requests that are in contention (namely Floor Control). Additionally, it supports Alert or Emergency conditions that guarantee a higher priority to users in dangerous situations.

### 2.1.2 Mission Critical Video (MCVideo)

The **MCVideo service**, supports video media communication between several users (i.e. group call), where each user can gain access to the permission to stream video in an arbitrated manner, and private calls between two users. The MCVideo service includes the following features, described in 3rd Generation Partnership Project (3GPP) [7]:

- Video capture and encoding of the video information;
- Secure streaming and storing of the video information;
- Video decoding and rendering of the video information;
- Processing of the video information, including the ability to annotate video frames and recognize video features;
- Mission Critical and public safety level functionality (e.g. group sessions, affiliations, end-to-end confidentiality, emergency type communications) and performance (e.g. low latency);
- Transmission and control of the parameters relevant to those functions;
- Secure operation such that video information can be reasonably un-impeachable when used in evidentiary procedures;
- Definition and configuration of MCVideo groups and applications;
- Configuration of the MCVideo users' profiles and of the MCVideo UEs; and
- Interoperability with other services and systems.

### 2.1.3 Mission Critical Data (MCData)

MCData, as explained in 3rd Generation Partnership Project (3GPP) [8], defines a service for Mission Critical Data services. As well as voice services, current mission critical users have been increasing their use of data services, including low throughput services on legacy networks and data services on commercial networks. The MCData service needs to provide a means to manage all data connections of mission critical users in the field and provide relevant resources to the ones who need them. The MCData Service provides a set of communication services that will be directly used by the user or functions that will be called by external applications in control rooms. The MCData Service will reuse functions including end-to-end encryption, key management, authentication of the sender, etc, to provide group communications for data services. In addition, the MCData Service will provide a set of generic capabilities such as: **messaging, file distribution, data streaming, IP proxy**, etc.

## 2.2 System Description

The Mission Critical (MC) architecture's functional model is defined as a series of planes that operate independently, providing services to the connected planes and requesting services from other planes as required. Currently, there are two types of functional models:

**On-Network** - MC services are accessed by gaining connectivity from an access network (5G or LTE).

**Off-Network** - MC services are accessed via other methods such as D2D Proximity Based Services (ProSe).

The two models ensure that first responders can communicate even in extreme situations where connectivity from mobile networks is unavailable, such as in the event of a natural disaster or operations in rural environments. In these cases, solutions such as Integrated Access Backhaul nodes would be implied to improve network coverage extension. An example scenario with these conditions is illustrated in Plazzotta [4] - Figure 3.4.

In this work, we focused exclusively on the **On-Network** functional model, composed of two planes described below.

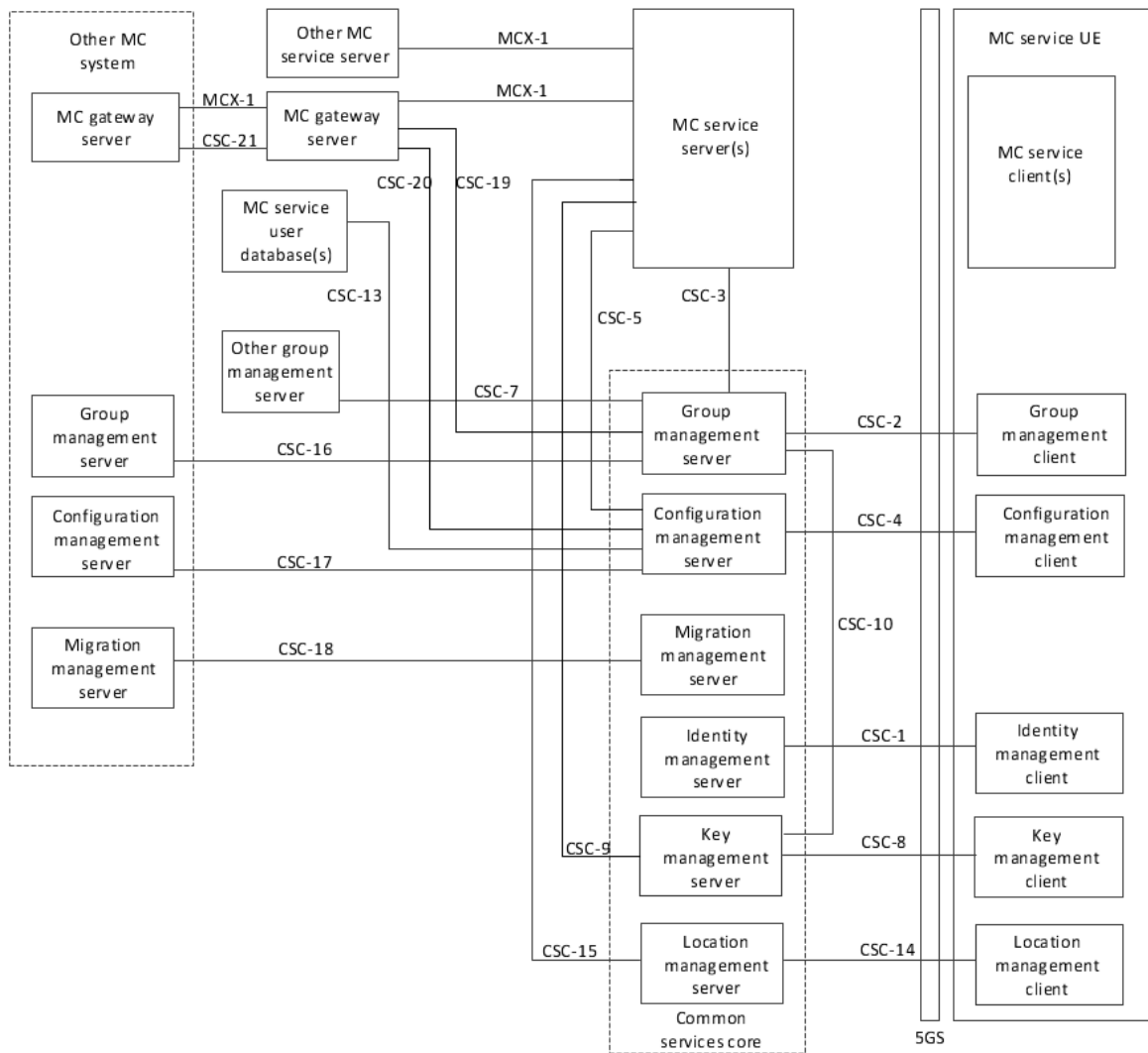
**Application plane** - The application plane provides all of the requisite services (e.g., call control, floor control, video control, data control, conferencing of media, provision of tones and announcements) required by the user, in addition to the necessary functions to support MC service. It uses the services of the signaling control plane to support those requirements.

**Signaling Control plane** - The signaling control plane provides the requisite signaling support for the establishment of user associations engaged in MC services, such as MCPTT calls or other forms of MC services. In addition, the signaling control plane provides access to and control of services across MC services. The signaling control plane utilizes the services of session connectivity.

### 2.2.1 Application Plane

Although the functional model across MC services may appear similar, each service has an individual application plane comprising its own set of functional entities, which are specific to that service. However, there's a set of functions and reference points that are common across all MC services and is referred to as the **Common Services Core (CSC)**.

The diagram in Figure 2.1, described in the On-network functional model section of ETSI [13], illustrates the main **CSC functions** and **Reference Points** of the Application plane.



**Figure 2.1:** Common functional model for Application Plane

As can be observed in 2.1, entities of the Common Services Core are associated with their corresponding client-side counterparts within the **MCX service UE**. The following subsections describe the **Application Plane**'s functional entities together with their reference point (RP).



### 2.2.1.1 IDMS & IDMC - RP CSC-1

The CSC-1 reference point, which exists between the **Identity Management Client (IDMC)**, located in the UE, and the **Identity Management Server (IDMS)** located in the CSC, provides the interface for user authentication and authorization within the Mission-Critical organization. One MC service user authenticates with the IDMS using its Mission Critical ID. The HTTP connection between the IDMC and the IDMS must be protected using HTTPS and must support OpenID Connect 1.0. A detailed explanation of the authentication procedure and the OpenID Connect technology is provided in [2.3.2.1](#).

### 2.2.1.2 GMS & GMC - RP CSC-2

The CSC-2 reference point links the **Group Management Client (GMC)**, located in the UE, with the **Group Management Server (GMS)**, located in the CSC. The GMC functional entity acts as the application user agent for the management of groups. The GMS facilitates the management of groups that are supported within the MC service provider, oversees the management of media policy information that is utilized by the UE for media processing, and oversees the management of group call policy information that is utilized by the UE for group call control.

### 2.2.1.3 CMS & CMC - RP CSC-4

The **Configuration Management Client (CMC)** acts as the application user agent for configuration-related transactions. It interacts with the **Configuration Management Server (CMS)** and provides and receives configuration data. They communicate via the CSC-4 RP that supports configuration of the UE by the MC service, and configuration of the MC service application with the MC service-related information that is not part of group management by the MC Service UE.

#### 2.2.1.4 KMS & KMC - RP CSC-8

Communication between the **Key Management Client (KMC)** and **Key Management Server (KMS)** occurs via the CSC-8 reference point. The KMS stores and provides security-related information to KMC and GMS over the CSC-10 RP. The primary purpose of these procedures is to ensure end-to-end secure communications for any endpoint, providing each of them with key material associated with its identity. Such keys are used to create a security context that protects media traffic in both private and group communications.

#### 2.2.1.5 MC service client/server

The MC service client and server are crucial for all MC service transactions. Each service, namely MCPTT, MCVideo, and MCDData has its own instantiation of this type of client and server, with specific functionalities. For instance, transactions between the MCPTT client and the MCPTT server occur via the MCPTT-1 Reference Point. For each MC service, the server must represent a specific instantiation of a Group Communication Service Application Server (GCS AS) to control multicast and unicast operations for group communications.

The remaining functional entities and reference points of this plane are specific to communication with a different MC system, in a different security domain, and thus are not described in this chapter.

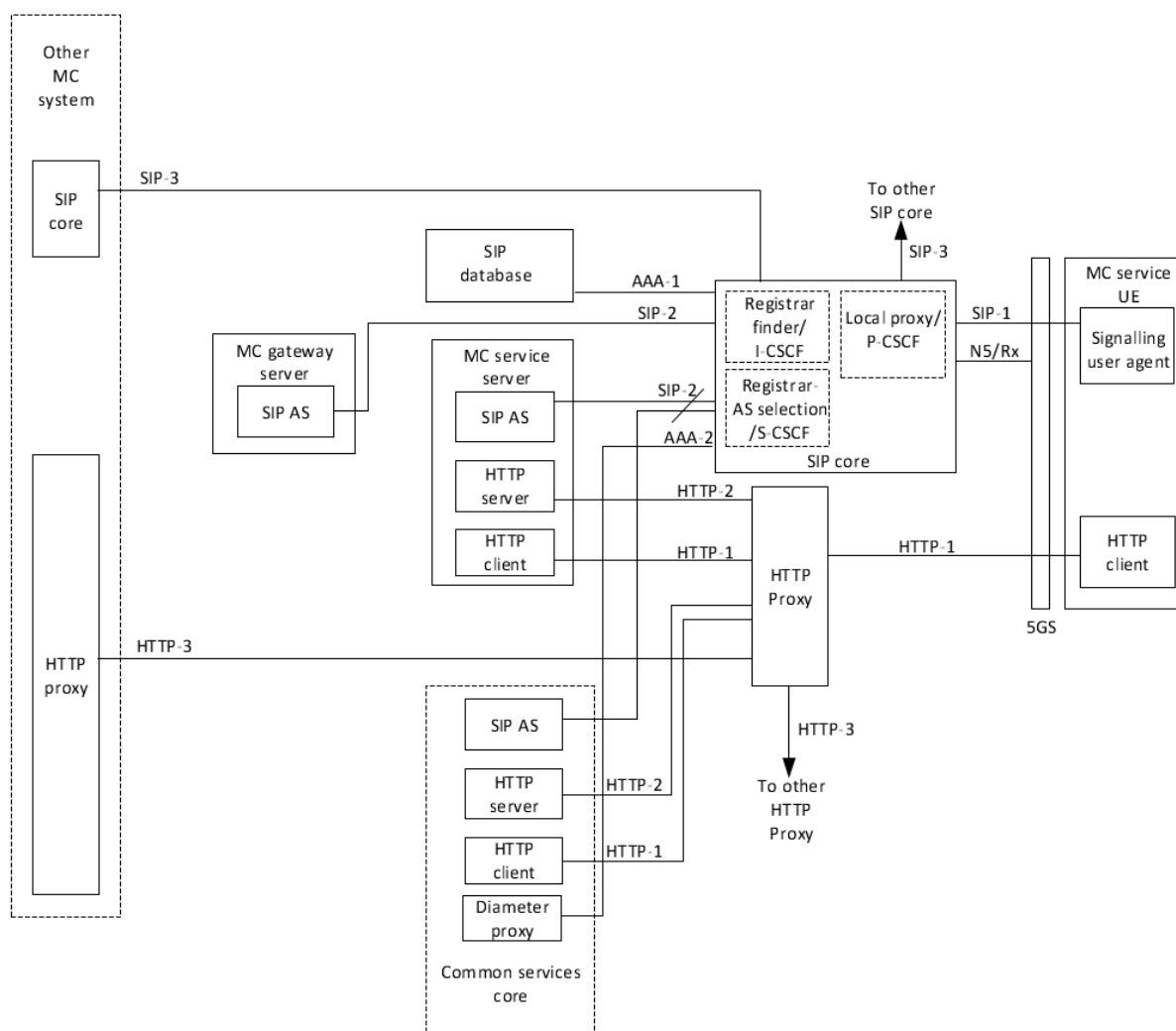
### 2.2.2 Signaling Control Plane

The **Signaling Control Plane**, as the Application Plane, is composed of entities connected via RP. Some of the RPs of the Signaling Plane are used by RPs belonging to the Application Plane for transport and routing of both subscription/notification and non-subscription/notification-related signaling. This plane doesn't include a CSC structure as the previous one, but involves two different types of entities:

**SIP entities** - Making use of SIP-x Reference Points. Involved in all transactions that use the **SIP** protocol.

**HTTP entities** - Making use of the HTTP-x Reference Points. Involved in all transactions that use the **HTTP** protocol.

The diagram in Figure 2.2, described in the On-network functional model section of ETSI [13], illustrates the main functional entities and Reference Points of the Signalling Control plane.



**Figure 2.2:** Functional model for signaling control plane

Before describing the functional entities and their reference points, is important to define the **SIPCore** which is the main component of the entire control plane.

### 2.2.2.1 SIPCore

The SIPCore is composed of different sub-entities that are responsible for registration, routing, and service selection.

It's relevant to outline that the **SIPcore** shall be either:

- compliant with 3GPP TS 23.228, i.e. the SIP core is a 3GPP IP multimedia core network subsystem; or
- a SIP core, which internally need not comply with the architecture of 3GPP TS 23.228, but with the reference points that are defined in subclause 7.5.3 (if exposed), compliant to the reference points defined in 3GPP TS 23.002.

In the provided description, we assume that the SIPCore is a 3GPP IP multimedia core network subsystem. As mentioned before, the SIPCore is composed of sub-entities, these are called **Session Control Functions (SCF)** and have different functionalities.

**P-CSCF** - Proxy-Call Session Control Function is the first point of contact for a **User Equipment (UE)** that needs to talk to the **SIPCore**. As its name implies, it behaves like a proxy, for instance, accepts requests and handles them internally or forwards them. These are some of the functionalities that the P-CSCF provides:

- Forwards SIP messages received from the UE to the SIP server (e.g. S-CSCF) whose name the P-CSCF has received as a result of the registration procedure.
- Ensures that the SIP messages received from the UE to the SIP server (e.g. S-CSCF) contain the correct or up-to-date information about the access network type currently used by the UE when the information is available from the access network.
- Forwards the SIP request or response to the UE

- Detects and handles an emergency session establishment request
- Maintains a Security Association between itself and each UE

Additionally, the P-CSCF can have the role of an **Application Function**, capable of interacting with the **Policy and Charging Architecture** located in **5G Core Network**, over the N5 interface, using the Npcf\_PolicyAuthorization service.

**I-CSCF** - Interrogating-CSCF is the contact point within an operator's network for all connections destined to a user of that network operator, or a roaming user currently located within that network operator's service area. Its main functions are to assign an S-CSCF to a user performing SIP registration and to route SIP requests received from another network towards the S-CSCF.

**S-CSCF** - Serving-Call Session Control Function performs the session control services for the UE. Subscribers are assigned an S-CSCF for the duration of their IMS registration to facilitate the routing of SIP messages as part of service establishment procedures. It maintains a session state as needed by the network operator for support of the services. Handles SIP transactions in different situations such as Registration procedures, session-related and session-unrelated flows, requests for an originating endpoint (UE or Application Server), and requests for a destination endpoint (terminating user/UE).

The following subsections describe the **Signaling Plane**'s functional entities and their reference points (RP).

#### **2.2.2.2 Signaling User Agent & SIPCore - SIP-1 RP**

The SIP-1 RP connects the signaling user agent in the User Equipment and the **SIPCore**. This RP is used for SIP Registration, Authentication and security to the service layer, Event subscription and notification, Overload control, Session management, and Media negotiation.

### **2.2.2.3 SIP core & the SIP AS - SIP-2 RP**

The SIP-2 RP exists between the SIPCore and the SIP Application Server (AS). The SIP AS supports influencing and impacting the SIP and supports event subscription and notification on behalf of the MC service server. This RP is used for the following functionalities:

- notification to the MC service server(s) of SIP registration by the MC service UE
- authentication and security to the service layer
- session management
- media negotiation

### **2.2.2.4 SIP database & the SIP core - AAA-1 RP**

The AAA-1 RP connects the SIPCore and the **SIP Database**. The SIP database contains information concerning the SIP subscriptions and corresponding identity and authentication information required by the SIP core. It's responsible for storing user-related information such as Numbering and addressing, SIP core access control information for authentication and authorization, MC service UE Location information, and signaling plane subscription profiles. Additionally, it provides support for control functions of the SIP core such as the I-CSCF and S-CSCF, needed to enable subscriber usage of the SIP core services. Depending on the deployment scenario, the SIP database can be provided by either the PLMN operator or the MC service provider.

### **2.2.2.5 HTTP Client & HTTP Proxy - HTTP-1 RP**

The HTTP-1 RP exists between the HTTP client in the UE and the HTTP Proxy at the edge of the Mission-Critical domain. The HTTP Proxy acts as a proxy for hypertext transactions between the HTTP client and one or more HTTP servers. The HTTP proxy must be situated within the same trust domain as the HTTP clients and HTTP servers that are located within an MC service provider's network. The HTTP-1 RP is employed by the UE to request the download of user and group configuration documents, provided by entities of the CSC (CMS, GMS, KMS, etc.).

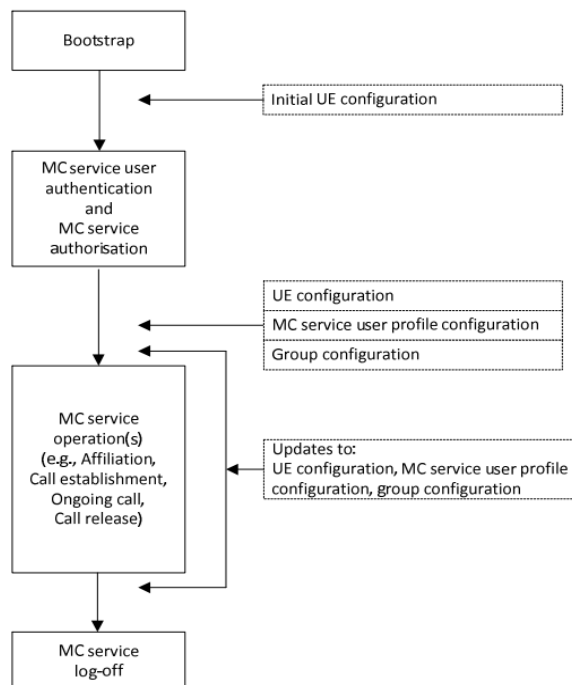
### 2.2.2.6 HTTP Proxy & HTTP Server - HTTP-2 RP

The HTTP-2 RP is utilized to establish a connection between the HTTP Proxy and the HTTP Server. It's used as a relay point by the HTTP proxy for requests initiated by the HTTP Client within the UE and directed towards the Common Services Core (CSC) and MC service servers.

As for the Application plane, other entities and RPs that are not useful for this work, are not described.

## 2.3 Procedures & Flows

This section provides descriptions and examples of procedures and flows useful to understand the role of some components described in 2.2 and scenarios described in 5. Figure 2.3 from ETSI [15], represents the lifecycle of an MC service UE that makes use of an MC service (MCPTT, MCVideo or MCDATA).



**Figure 2.3:** UE general lifecycle using an MC service

### 2.3.1 Bootstrap Procedure

Bootstrap procedures are used to provide the **MC service UE** with **initial UE configuration**. Specifically, it provides critical information, needed to connect with the MC System, to client components that reside in the UE, such as MC service client, Configuration management Client, Group Management Client, Identity Management Client, etc.. (described in 2.2.1). The data contained in the UE initial configuration can be stored either in the Mobile Equipment or in the USIM. If both locations are utilized for storage, the values stored in the USIM will have precedence. A few examples of information that can be found inside this configuration document are

**GMSURI** - group management service URI information which contains the public service identity for performing the subscription proxy function of the GMS.

**CMSXCAPRootURI** - configuration management server XCAP Root URI information.

**GMSXCAPRootURI** - group management server XCAP Root URI information.

**PLMN** - HPLMN code that identifies the Public Network.

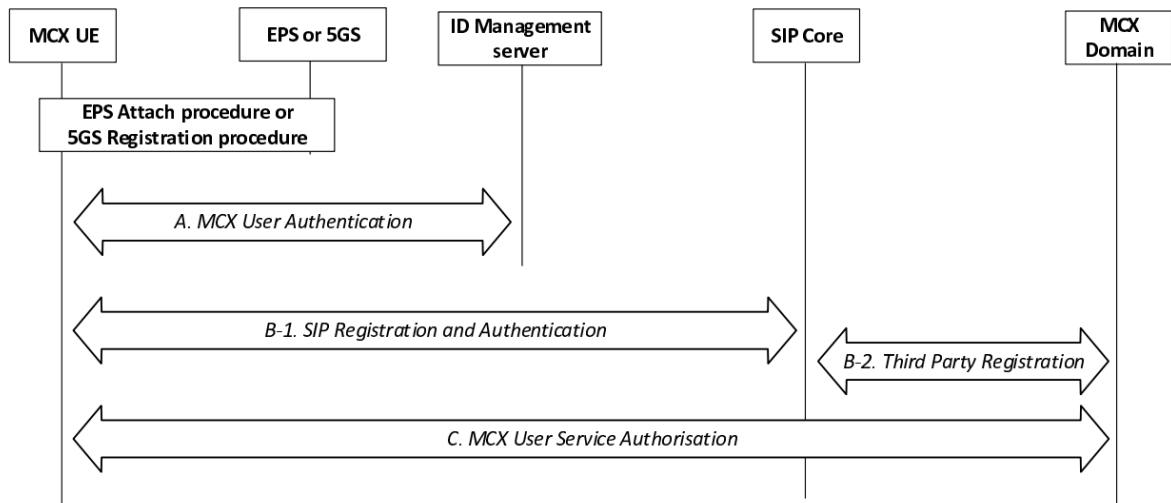
**IDMSAuthEndpoint** - identity management server authorization endpoint identity information.

A full list of the entries contained in the **UE initial configuration** document along with their description is defined in ETSI [14].

### 2.3.2 Authentication & Authorization Procedure

User authentication and authorization, as outlined in Section 5 of ETSI [9], are composed of the steps illustrated in 2.4





**Figure 2.4:** MCX authentication & authorization

Firstly, the **MCX User Equipment** must perform authentication with respect to the **5G system**. This step is mandatory for all UEs who wish to utilize services accessible through session connectivity. As it is not directly relevant to the Mission Critical domain, it is not addressed in this work. The MCX UE then performs the following steps to complete authentication of the user, authorization of the user, MCX service registration, and identity binding between signaling layer identities and the MC service ID(s).

**Step A** - MCX user authentication with Identity Management Server (IdMS).

**Step B-1** - SIP Registration and Authentication with SIP Core (SC).

**Step C** - MCX Service Authorization with MCX Domain.

If an MCX UE completes SIP registration in Step B before performing MCX user authentication in Step A and MCX user service authorization as part of Step C, the MCX UE shall be able to enter a 'limited service' state. In this limited state, where the MCX user is not yet authorized with the MCX service, the MCX UE shall be able to use limited MCX services (e.g. an anonymous MCX emergency communication). The MCX Server is informed of the registration of the MC UE with the SIP core through Step B-2.

### 2.3.2.1 MCX User Authentication

MCX User Authentication refers to Step A in Figure 2.4 and consists of the IDMS, located in the MCX Common Services Core (CSC), and the IDMC, located in the MCX UE. The IDMC, after having obtained the *IDMSAuthEndpoint* through the **Bootstrap Procedure**, can communicate with the IDMS via the **CSC-1 RP**, which is a direct HTTP interface that provides user authentication and shall support **OpenID Connect 1.0**. As explained in TS 33.180 ETSI [9], to support MCX user authentication, the IDMS shall be provisioned with the user's MC ID and MC service IDs (the MC service ID may be the same as the MC ID). A mapping between the MC ID and MC service ID(s) shall be created and maintained in the IDMS. When an MCX user wishes to authenticate with the MCX system, the MC ID and credentials are provided via the UE IDMC to the IDMS. The IDMS receives and verifies the MC ID and credentials, and if valid returns an **ID token**, **refresh token**, and **access token**, specific for that client. A detailed explanation of the process using OIDC is explained in 2.4.5. The HTTP connection between the Identity Management client and the Identity Management server must be protected using HTTPS, so eavesdropping traffic won't give any effective information about a user.

### 2.3.2.2 MCX User Service Authorization

MCX User Service Authorization refers to Step C in Figure 2.4 and is the function that validates whether or not an MCX user has the authority to access certain MCX services. In order to gain access to MCX services, the MCX client in the UE presents an access token, obtained during MCX User Authentication. The services that can be accessed with such access token are listed in 2.4.3.2. If the access token is valid, then the user is granted the use of that service. For MCPTT, MCVideo, and MCData user service authorization, the appropriate client component in the UE presents an access token to the appropriate server component, over **SIP** protocol. Each server must map and maintain the association between the **IMS Public User Identity (IMPU)** and the **MCX ID**. The SIP message used to convey the access token from the MCX client to the MCX server may be either a SIP REGISTER or a SIP PUBLISH message.

Referring to 2.3, the UE has completed the step in the second block "MC Service User Authentication and MC Service Authorization". The UE can now download UE configuration, MC service user profile configuration, and group configuration documents, necessary to perform the various MC service operations.

### 2.3.3 MC service operations

MC service operations include all functionalities that can be used by authenticated and authorized users, that have received UE configuration, MC service user profile, and group configuration parameters. Some of these functionalities are listed below

**MC service group (de)affiliation** - Group affiliation is the procedure by which a client is allowed to communicate within one or more specific groups that can support one or more MC services. For instance, there could be a group called "repo", that supports the MCDATA service, used to store files. Only users that are affiliated with such a group can upload or download files from the repository. The affiliation procedure can be explicit, where an MC service client indicates interest in one or many MC service groups to the MC service server, or implicit, where the MC service user's affiliations to MC service groups are determined through configurations and policies within the MC service and performed by the associated MC service server.

**MCPTT calls** - Calls making use of the MCPTT service. These can be either private or group calls, after successfully joining one or more groups supporting the MCPTT service.

**MCVideo calls** - The same as for MCPTT but with the addition of a video stream on top of the audio.

Some operations can be done only by users with a specific role. For instance, a Dispatcher User can manage group affiliations for all users within the MC organization.

## 2.4 OpenID Connect (OIDC)

OpenID Connect extends the OAuth protocol to provide a dedicated identity and authentication layer that sits on top of the basic OAuth implementation. It adds some simple functionality that enables better support for the authentication use case of OAuth.

### 2.4.1 What is OAuth?

The OAuth 2.0 authorization framework is a protocol that allows a user to grant a third-party website or application access to the user's protected resources, without necessarily revealing their long-term credentials or even their identity.

OAuth introduces an authorization layer and separates the role of the client from that of the resource owner. In OAuth, the client requests access to resources controlled by the resource owner and hosted by the resource server and is issued a different set of credentials than those of the resource owner.

Instead of using the resource owner's credentials to access protected resources, the client obtains an **Access Token**, a string denoting a specific *scope*, *lifetime*, and *other access attributes*. Access tokens are issued to third-party clients by an authorization server with the approval of the resource owner. Then the client uses the access token to access the protected resources hosted by the resource server.

### 2.4.2 OIDC vs OAuth


While OAuth 2.0 is about resource access and sharing, OIDC is about user authentication. Its purpose is to give you one login for multiple sites. Each time you need to log in to a website using OIDC, you are redirected to your OpenID site where you log in, and then taken back to the website. For example, if you chose to sign in to Auth0 using your Google account then you used OIDC. Once you successfully authenticate with Google and authorize Auth0 to access your information, Google sends information back to Auth0 about the user and the authentication performed. An example of login using OIDC is shown in [2.5](#).


Indirizzo e-mail\*


Continua

Non hai un account? [Registrati](#)

OPPURE

 Continua con Google

 Continua con l'account Microsoft

 Continua con Apple

**Figure 2.5:** Login with OIDC options

### 2.4.3 How does OpenID Connect work?

OpenID Connect slots neatly into the normal OAuth flows. From the client application's perspective, the key difference is that there is an additional, standardized set of scopes that are the same for all providers, and an extra response type: 'id.token'.

#### 2.4.3.1 OpenID Connect Roles

The roles for OpenID Connect are essentially the same as for standard OAuth. The main difference is that the specification uses slightly different terminology.

**Relying party** The application that is requesting authentication of a user. This is synonymous with the OAuth client application.

**End user** The user who is being authenticated. This is synonymous with the OAuth resource owner.

**OpenID provider** An OAuth service that is configured to support OpenID Connect.

### 2.4.3.2 OpenID Connect claims and scopes

The term "claims" refers to the key:value pairs that represent information about the user on the resource server. One example of a claim could be "user\_type":"mobile\_client".

Scopes, instead are an indication by the client that it wants to access some resources, the resource server may allow or reject the request. To make use of OIDC, the client application must specify the *openid* scope in the authorization request. MCX applications use a specific set of scopes, which are listed below.

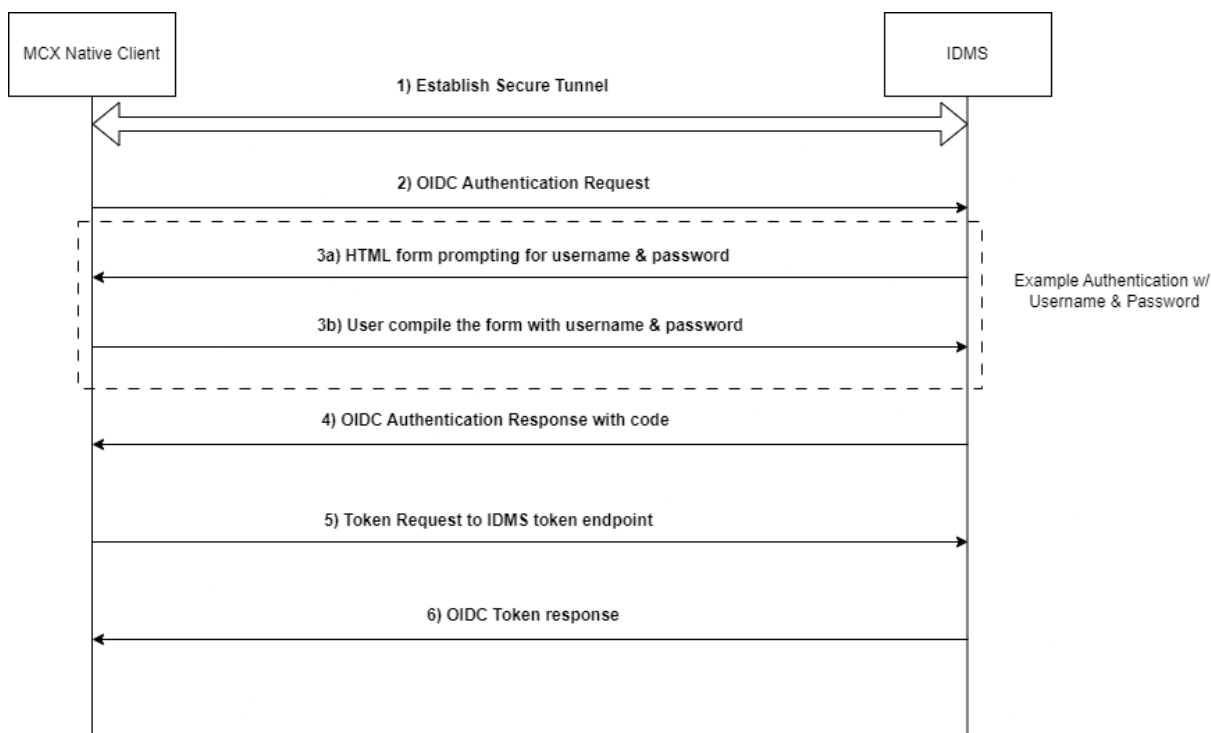
- 3gpp:mc:ptt\_service
- 3gpp:mc:video\_service
- 3gpp:mc:data\_service
- 3gpp:mc:ptt\_key\_management\_service
- 3gpp:mc:video\_key\_management\_service
- 3gpp:mc:data\_key\_management\_service
- 3gpp:mc:ptt\_config\_management\_service
- 3gpp:mc:video\_config\_management\_service
- 3gpp:mc:data\_config\_management\_service
- 3gpp:mc:ptt\_group\_management\_service
- 3gpp:mc:video\_group\_management\_service
- 3gpp:mc:data\_group\_management\_service
- 3gpp:mc:location\_management\_service

## 2.4.4 OAuth Grant Types

OAuth grant types, also called "flows", define steps and procedures involved in the OAuth process. Different grant types mean different ways in which the client application communicates with the OAuth service at each step. There are several different grant types but we focused on *authorization\_code* because as stated in the 3GPP Technical Specification Document TS 33.180 (Security of the Mission Critical (MC) service) - Annex B.4.2.1, "MCX clients fitting the Native application profile utilize the authorization code grant type with the IETF RFC 7636 [53] PKCE extension for enhanced security"

## 2.4.5 Authorization Code Grant Type

Before describing the **authorization\_code** flow, it's worth mentioning that all procedures are tied to a specific type of MCX client. Indeed as explained in Annex B.4.1 of ETSI [9], "MCX Connect will support a number of different MCX client types, including native, web-based, and browser-based. Only **native clients** are defined in this version of the MCX Connect profile". **Native** clients are devices such as mobile phones, that have secret parameters directly embedded in the application's source code, easily accessible through decompilers or other reverse engineering tools. The set of **native** clients in the OAuth2.0 specification, also includes single-page web applications, that in the Mission Critical domain could be either a Dispatcher or Discrete Listener. Figure 2.6 illustrates the **authorization\_code** flow with username and password authentication using an MCX Native client.



**Figure 2.6:** OIDC flow with Username & password authentication

The first step corresponds to **establishing a secure tunnel**, which means that the communication between the client and the IDMS must be protected using **HTTPS**, to avoid cleartext traffic. The next step consists of the **OIDC Authentication Request**. The parameters needed for such a step are described in Table 2.1



Parameter	Value	Required	Description
response_type	"code"	Y	Defines the flow type: authorization code flow
client_id	client_id *Must match the client_id assigned by the IDMS after the client registration phase. See 2.4.6.	Y	ID of the requesting client
scope	Values listed in 2.4.3.2	Y	list of space-delimited, case-sensitive strings that indicate which MCX resource servers the client is requesting access to (e.g. MCPPTT, MCVideo, MCDData, KMS, etc.)
redirect_uri	URL on which the client receives callback *Must match the redirect_uri registered with the IDMS during the client registration phase. See 2.4.6.	Y	The URI of the client to which the IDMS will redirect the authentication response in order to receive the authorization code.
state	random value	Y	An opaque value used by the client to maintain the state between the authentication request and authentication response. The IdM server includes this value in its authentication response.
code_challenge	random string	Y	base64url-encoded SHA-256 challenge derived from the code_verifier, verified in the final request to obtain the access_token.
code_challenge_method	S256	Y	hash method used to transform the code_verifier to produce the code challenge.

**Table 2.1:** Authentication Request required parameters

Subsequent steps **3a** and **3b** are specific for username and password authentication. The IDMS sends an HTML form prompting the user for their username and password. The user provides such data and in the case that these are correct, the flow proceeds to Step 4 - OIDC authentication response. The **authentication response** consists of the IDMS that issues an **authorization code**, which is delivered to the `redirect_uri`, specified in the authorization request. This code is specific for this grant type and is one of the parameters used to obtain `access_token`, `id_token`, `refresh_token`, etc. In addition to the authorization code, the `state` parameter should also be included in the authentication response. Its value should match the initial value specified in the authentication request otherwise the client is under a Cross-site Request Forgery (CSRF) attack and the final request to the `token` endpoint using that authorization code should be rejected.

After having obtained the `authorization code`, steps 5 and 6, respectively outlined as **OIDC Token Request** and **OIDC Token Response**, occur. The parameters contained in the OIDC Token Request are shown in Table [2.2](#)

Parameter	Value	Required	Description
grant_type	"code"	Y	Defines the flow type: authorization code flow
client_id *Shall match the value specified in the authorization request	client_id	Y	ID of the requesting client
code	authorization_code	Y	authorization code previously received from the IdM server as a result of the authentication request and subsequent successful authentication of the MCX user.
redirect_uri	URL on which the client receives callback *Must match the redirect URI specified in the authentication request.	Y	The URI of the client to which the IDMS will redirect the authentication response to receive the authorization code.
code_verifier	random value	Y	cryptographically random string that is used to correlate the authentication request to the token request.

**Table 2.2:** Access Token request parameters

In the case that all parameters provided in the Token request are valid, the Token response returns the values described in Table 2.3.

Parameter	Value	Required	Description
access_token	access_token	Y	Token inserted in all requests to the resource server via Authorization: Bearer header.
token_type	Bearer	Y	Type of token.
expires_in	value in seconds	Y	Indicates the lifetime of the access_token.
id_token	id_token	N	Token that contains information about a user.
refresh_token	refresh_token	N	Token used to obtain a new access_token when expired.

**Table 2.3:** Access Token Response parameters

The obtained `access_token` is then presented in every request that, once validated, grants access to services provided by CMS, GMS, KMS, etc.

## 2.4.6 Client Registration

In order to use the authentication and authorization features, at least one client application must be registered with the authorization server (IDMS). The MCX system administrator provides specific parameters such as application type (web, mobile, etc.), redirect uris, response type, grant type, etc. that will characterize the newly registered client application. The resulting registration responses return a client identifier (`client_id`) to be used at the authorization server and the client metadata values registered for the client. The client can then use this registration information to communicate with the authorization server using the OAuth 2.0 protocol. Clients can be pre-provisioned or dynamically registered by making a request to the **registration endpoint** as described in the Dynamic Client Registration document provided by the Open ID Foundation [19].

## 2.5 Session Initiation Protocol (SIP)

The **Session Initiation Protocol (SIP)**, was standardized by Internet Engineering Task Force (IETF) in 1999 (RFC2543). Later was accepted by 3GPP as a signaling protocol and became a crucial element for **IP Multimedia Subsystem (IMS)** architecture. An updated version was released in 2002 under RFC3261, which introduced new extensions, headers, and security solutions. As explained by IETF in Rosenberg [1], SIP is an application-layer control (signaling) protocol for creating, modifying, and terminating sessions with one or more participants. These sessions include Internet telephone calls, multimedia distribution, and multimedia conferences.

### 2.5.1 SIP Methods

The Session Initiation Protocol relies on methods that perform different tasks and allow the establishment of a session (call) between one or more users. The SIP protocol supports the following methods:

**REGISTER** - Used by a user agent to register itself with a SIP registrar server. The REGISTER method is used in the MCX SIP Registration step, illustrated in Figure 2.4, and helps in maintaining an up-to-date mapping between a user's SIP URI and their current IP address, enabling call routing and delivery.

**OPTIONS** - Used to query the capabilities of a SIP server or user agent. This method can be sent to request information about supported methods, media types, or other extensions without actually establishing a session.

**INVITE** - Used to initiate a new session (call) or modify an existing one. The INVITE method carries the session description (typically using SDP) to inform the recipient about the details of the proposed session, such as media types, codecs, and transport protocols. In the context of Mission Critical communications, also XML messages can be carried in INVITE requests, that contain information such as the session type, the group uri in the event of a group call, and the client id.

**CANCEL** - Sent to cancel a pending INVITE request before the session is established.

The CANCEL method allows the sender to abort an INVITE transaction if they change their mind or if there is no response from the recipient.

**ACK** - Sent to confirm the receipt of a final response to an INVITE request. The ACK method ensures the reliability of INVITE transactions by providing end-to-end acknowledgment.

**BYE** - Used to terminate an established session (call). The BYE method is sent by either party in the session to indicate that they wish to end the communication.

## 2.6 What is a threat analysis?

A threat analysis is a methodical process used to assess how a system, service, or process could be attacked, whether by malicious actors or unintentionally through misconfiguration. The outcome of a threat analysis is the identification of a system's vulnerabilities and the ways they might be exploited. In connected environments, it is crucial to adopt a "systems approach" for threat analysis. This approach considers the interactions and relationships between a system's components throughout its lifecycle. Here, 'system' includes not just the product (e.g., a physical component) but also the backend infrastructure, people, and processes it relies on and that rely on it. Connected environments often integrate various third-party services, making it essential to evaluate the trust and risk exposure these services bring. A threat analysis can uncover various types of threats within an organization, which can be categorized as follows:

**Accidental Threats** - These arise from misconfigurations or accidents that leave an organization exposed. Human error is one of the leading causes of cyberattacks today. By conducting a threat analysis, organizations can identify and address accidental errors before malicious actors exploit them.

**Intentional Threats** - The threat that every organization is worried about is the intentional threat. Intentional threats are those conducted by malicious entities to gain access to sensitive data within an organization and make a profit from it.

**Internal Threats** - These threats originate from within the organization. While organizations often focus on external threats and build robust security measures to prevent outside attacks, internal threats can be more dangerous. When an employee acts maliciously, they can have easier access to sensitive information, making internal threats particularly catastrophic.

### 2.6.1 STRIDE

The STRIDE model is a structured and iterative methodology used to evaluate a system and find potential vulnerabilities and weaknesses in its components. The six terms that compose the acronym are the following ones:

**S - Spoofing** - When a threat actor can impersonate a legitimate user in the system to gain an illegitimate advantage.

**T - Tampering** - When a threat actor is able to manipulate data exchanged between a legitimate user and the system. This usually happens when data integrity is not correctly validated or validated at all.

**R - Repudiation** - Occurs when we are not able to link an action or event to a unique individual. For instance, can someone do something and deny being the author of such an action?

**I - Information Disclosure** - When someone can obtain access to information they should not have access to. It can happen if data is communicated in an unencrypted form or weak cryptographic algorithms are used, but also if information is left inside publicly available resources, such as Javascript files.

**D - Denial of Service** - When a threat actor is able to affect a system in such a way as to make it unusable for others or disrupt it. This is usually performed either by overloading the system with a lot of traffic, that at some point cannot be processed anymore or by sending malformed input which breaks the parsing logic.

**E - Elevation of Privileges** - When an unprivileged user or process is able to gain access beyond their established permissions. It includes the scenario where a user with low privileges can obtain higher privileges through misconfigurations.

The STRIDE methodology aims to scope the work using **Data Flow Diagrams**. This process is usually called "Decompose the application", and these diagrams show the different paths through the system, highlighting the privilege or trust boundaries. However, the adopted approach can vary depending on the analyzed system. This step often calls out "assets", which can be any of the things that need to be protected, stepping stones, or things attackers want. Often times those assets are out of scope for a project and are a distraction. Other times, they're hard to identify in advance of an attacker drawing attention to them.



# Chapter 3

## State of the Art

To the best of our knowledge, all existing works that analyze security aspects of the Mission Critical domain, have been made only by the Third Generation Partnership Project (3GPP). They have a dedicated series of documents, the **33 series**, focused on security aspects of technologies described in their standards. The 33 series includes **Technical Specification (TS)** documents, which are useful for understanding and implementing security requirements, and **Technical Reports (TR)** that present studies or enhancements based on the procedures described in TS documents. However, these existing studies are based only on theoretical concepts, lacking some details that can be caught more easily through a practical assessment of a Mission Critical system. Additionally, Mission Critical applications involve technologies for which security studies have already been conducted. This chapter explores existing resources from the **3GPP 33 series**, specifically focused on the Mission Critical domain and its related procedures. These resources helped gather initial knowledge on how security is approached, which requirements are outlined, and the work that has already been done. Additionally, we list some resources that analyze technologies commonly used in everyday applications, which have been further contextualized within our domain of interest in this work. The chapter is organized into two sections that distinguish work done by 3GPP, strictly related to the Mission Critical domain, and work related to involved technologies.

## 3.1 3GPP TS & TR documents

This section aims to describe the existing work done by the 3GPP, with its document from the 33 series, specifically focused on the security aspects of the MC domain. The following resources represent the state of the art for our work:

### TS 33.180 - Security of the Mission Critical service

The Technical Specification Document 33.180 is the main reference describing security requirements that must be implemented when developing an MC application. Rather than providing possible attacks or vulnerabilities, it specifies the security architecture, procedures, and information flows needed to protect the MC system and its services. The provided guidelines are focused on the confidentiality and integrity protection mechanisms of Reference Points, particularly ones that interconnect client components with the MC domain, procedures for key derivation, key distribution, security measures for interworking, interconnection, and migration between more MC systems.

### TR 33.880 - Study on mission critical security enhancements

The Technical Report Document 33.880 contains a study of the security aspects of the Mission Critical service. It provides enhancements to security solutions defined in TS 33.179, entitled *Security of Mission Critical Push To Talk (MCPTT) over LTE*. The document is organized in three different parts that respectively outline potential **Key Issues** for all MCX services (MCPTT, MCVideo, and MCDData), potential **security solutions**, and a final **evaluation** of the proposed solutions.

## 3.2 Involved Technologies

This section describes works that have been done on technologies and protocols that are standardized in MC-related Technical Specification documents of the 3GPP and thus implemented in every Mission Critical application.

## **OWASP Web Security Testing Guide**

The guide is provided by the **Open Worldwide Application Security Project (OWASP)**, a nonprofit foundation, launched in December 2001, that works to improve software security. All of their projects, tools, documents, forums, and chapters are free and open to anyone interested in improving application security. One of their projects is indeed the **Web Security Testing Guide (WSTG)** [21], which has the goal to help people understand the what, why, when, where, and how of testing web applications

## **OAuth 2.0 Security Best Current Practice**

The OAuth 2.0 Security Best Current Practice document [6], published by the IETF's Web Authorization Protocol working group on June 2024, updates and extends the threat model and security advice given in RFC 6749, RFC 6750, and RFC 6819 to incorporate practical experiences gathered since OAuth 2.0 was published and covers new threats relevant due to the broader application of OAuth 2.0. The document provides a list of best practices, attacks, and mitigations on various grant types and features of the Auth 2.0 protocol.

## **SIPman: A penetration testing methodology for SIP and RTP**

SIPman [2] investigates if it is possible to create a penetration testing methodology for SIP and RTP, where the target group is penetration testers with no previous knowledge of these protocols. They performed exploratory testing on three different SIP server architectures, outlining previously discovered vulnerabilities and attacks, and mapping them into the STRIDE threat model. In the final part, they also provide an Appendix section with example payloads and tools used to carry out the testing process. Unfortunately, this work is based on the assumption that the traffic sent through the network is not encrypted due to the nature of the protocols, which would violate the strong security concept enforced in mission-critical communications, and thus the majority of listed vulnerabilities were not considered to produce our scenarios.

# Chapter 4

## Testbed

This chapter presents the description of our testbed, which comprises the overall network architecture, the identified **security zones**, their functionalities, and entities that reside within such zones. The final part of the chapter presents a list of hardware/software technologies and tools used for deployment and testing purposes.

### 4.1 Overall Architecture

The following components were used to understand and conduct tests on the Mission Critical system. Figure 4.1 illustrates the testbed's overall network architecture, including the attacker's workstation, that extended the already existing deployment provided by Leonardo.

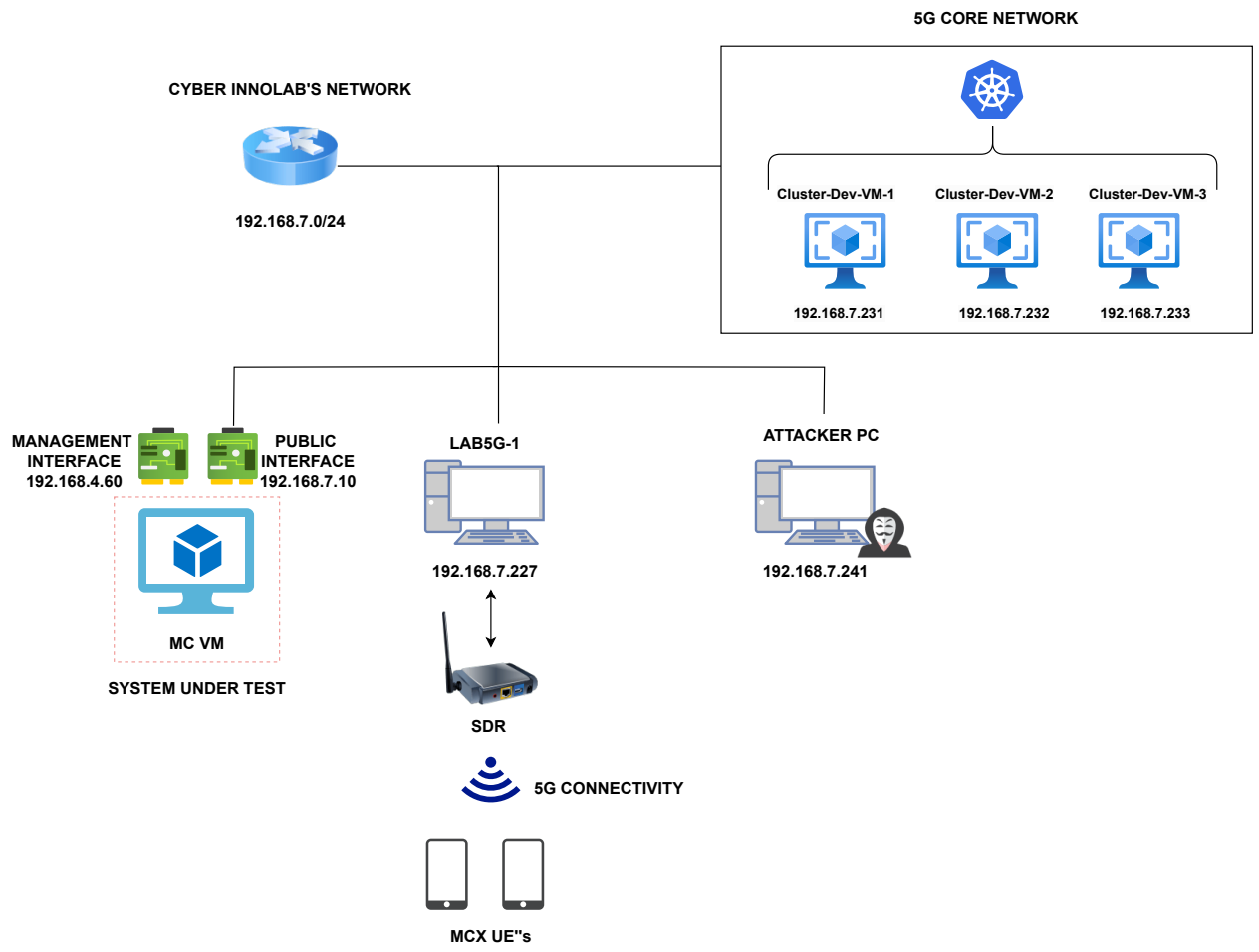
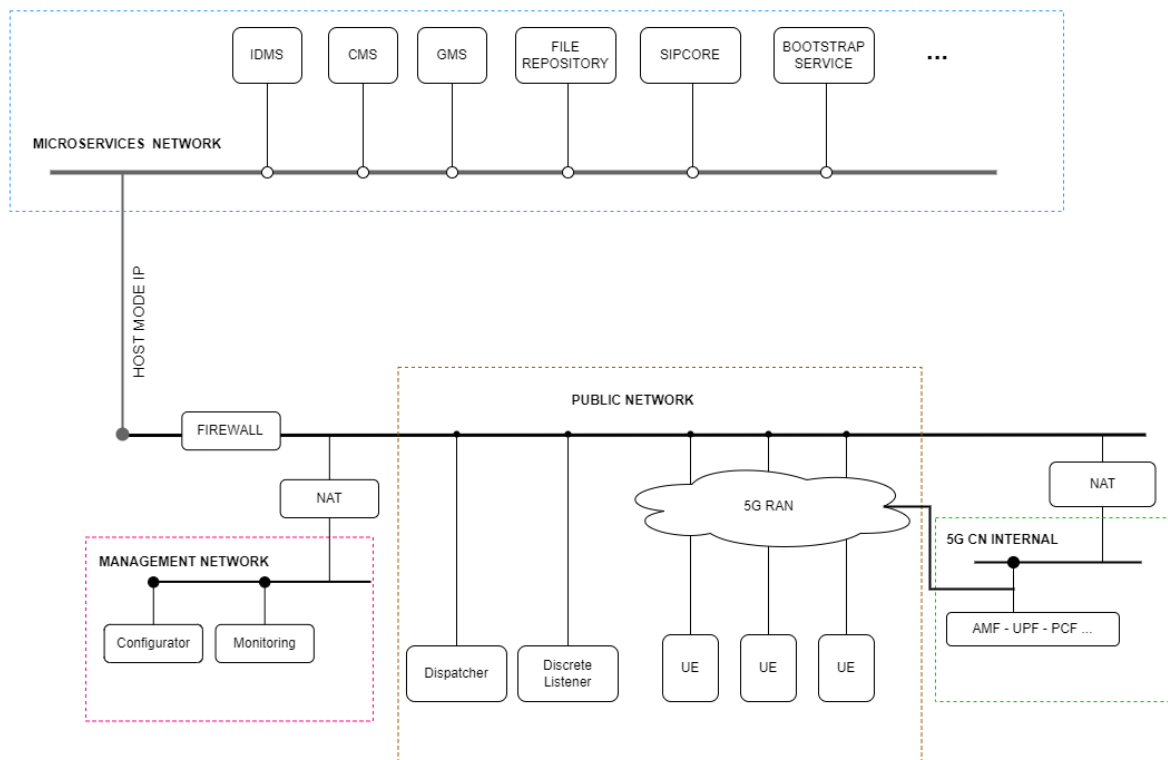


Figure 4.1: Overall network architecture

All hosts and Virtual Machines gain connectivity from Cyber Innolab's Network, which is the private network of the Leonardo Cyber Security R&D Laboratory. All machines have IP addresses in the range 192.168.7.0/24. The three VMs with addresses respectively 192.168.7.231, 192.168.7.232, and 192.168.7.233 were used to deploy the **5G Core Network**, with tools described in Section 4.3.2. The machine Lab5g-1 was used to deploy the **5G Radio Access Network (RAN)**, from which **MCX User Equipments (UE's)** gain connectivity. A description of the software and hardware tools implicated for the 5G RAN is presented in Section 4.3.1. The **attacker's PC**, with IP address 192.168.7.241, is the one used to run tests and has all the tools listed in Section 4.3.3 installed on it. The **MC VM** is the Virtual Machine that hosts **MCXPTT**, the system on which the practical assessment has been conducted. MCXPTT is a product developed by Leonardo, whose interests in emergency communication began in 2012, with the development of the XPTT system, an enhanced Push-to-Talk for smartphones with a web control platform for business users. Later, with the development of Mission Critical communications by the 3GPP organization, Alea S.r.l grew its interest in MC services and created MCXPTT. This application supports all MC services described in 2.1, compliant with the MC standard of 3GPP. MCXPTT provides an Android Client application that users install on their MCX User Equipments, a Dispatcher web application used for Control Room purposes, a Discrete Listener web application for Ambient Listening purposes, and an administrator web portal that allows you to configure each single parameter of the system in order to satisfy any operational requirement. The MC VM, which is identified as the **System Under Test (SUT)**, has two network interfaces, one called **Public** with IP address 192.168.7.10, and another one called **Management**, with IP address 192.168.4.60. In addition, the architecture shown in Figure 4.1 can be used as a starting point for other vendors that need to test their applications, as it is only required to replace the **System Under Test**, while the rest can be kept the same.

## 4.2 Security Zones

Section 2.6.1 outlines that STRIDE mainly decomposes the application using Data Flow Diagrams, which aim to identify the different paths through the system, highlighting the privilege or trust boundaries. Similarly to the STRIDE model that analyzes the interactions between components, in this work, we break down the system into **Security Zones**. A security zone is a subnet with similar security requirements and security levels. Figure 4.2 shows a diagram that illustrates the identified **Security Zones** within the overall architecture, described in the previous section, and the various entities that reside within such zones.



**Figure 4.2:** Testbed's Security Zones

### 4.2.1 Public Network

The **Public Network**, contained in the **brown rectangle**, refers to the subnet used by the clients, who need to reach and access entities exposed by the Mission Critical application. Clients with their User Equipment obtain connectivity via 5G Mobile Networks, while others like Dispatchers or Discrete Listeners obtain connectivity via Wi-Fi networks. Mobile Networks could be either operated by Mobile Network Operators (telecom operators), that offer wide area coverage with access available to any subscriber with a compatible device and a service plan from the network operator, or managed by a specific organization, enterprise, or vertical industry, designed to cover specific areas, with restricted access to authorized users or devices and dedicated resources. In our testbed, we used a 5G Stand-alone Network, which means that no elements from the 4G architecture are implicated. The 5G infrastructure is composed of a **Radio Access Network (RAN)** part and the so-called **Core Network (CN)**, which is a set of Network Functions (NF) that communicate with each other via Reference Points (RP) and handle different aspects of the system. Operators such as Police officers or Firefighters with the MCXPTT client application installed on their UE are connected to this network.

#### 4.2.1.1 5G Radio Access Network

A 5G **Radio Access Network (RAN)** relies on a fully coordinated, multi-layer network with low-band, mid-band, and high-band to provide wireless connectivity to devices and deliver the best network performance. It comprises antennas, radios, baseband (RAN Compute), and RAN software to enable incredible speeds and mobility. Telco Operators normally provide physical antennas, but a few open-source software solutions can be used in conjunction with specific hardware components, to replicate the same/similar behavior.

#### 4.2.1.2 5G Core Network

The **5G Core Network Internal**, highlighted with a **green rectangle**, refers to the subnet where all the containers/pods running the Network Functions (NF) are located. The 5G CN may be provided by telco operators or deployed on private servers.



## 4.2.2 Management Network

The **Management Network** subnet, contained in the **pink rectangle**, provides services for the **Public Network** and services that are not reachable from the outside. Concerning the **MC VM** in Figure 4.1, some services within this network can only be accessed through the **Management interface**, while others can be accessed via the **Public interface**. Internal services provided by the "Configurator" web portal, used by the MCX System Administrator, are available through this network. Some examples are managing existing organizations, customizing configuration parameters, creating new users, deleting or modifying the attributes of existing ones, and much more. In addition, this subnet could contain applications for monitoring purposes.

## 4.2.3 Microservices Network

The **Microservices Network**, highlighted with a **blue rectangle**, refers to the subnet where all containers run. These are executed on the **MC VM**, illustrated in Figure 4.1, configured in Docker Host-Mode<sup>1</sup>, and reachable on the **Public interface**. At this point, services that must be reached by clients on the **Public Network**, will have their port exposed through **Firewall** rules, while others can communicate only with internal services.

## 4.3 Tools

To build the testbed and run tests on it, several tools and technologies were adopted. The following subsections 4.3.1, 4.3.2 refer to software/hardware technologies used to build the infrastructure, which includes the 5G Radio Access Network (RAN), the 5G Core Network (CN) and the Mission Critical Software, while 4.3.3 describes tools used in my security analysis.

---

<sup>1</sup>Container's network stack is not isolated from the Docker host.

### 4.3.1 5G RAN

These tools were used to deploy a fully functional **gNodeB (gNb)**, which represents the base station for 5G systems. This component performs tasks like mobility management, Radio resource management, Radio signal processing, connection management, packet processing, and baseband processing. Additionally, provides Security, Quality of service (QoS), and charging.

**SDR** - The hardware part was based on **Software-Defined Radio (SDR)**, which is a radio communication system where components, that conventionally have been implemented in analog hardware (e.g. mixers, filters, amplifiers, modulators/demodulators, detectors, etc.), are instead implemented through software on a computer or embedded system. The SDR provides the radio front-end, making the communication on air very similar to a real 5G antenna provided by a Mobile Network operator. The used SDR was a USRP X310 from Ettus Research, which communicates with a PC through a high-speed link (10 GB), thanks to the USRP Hardware Driver (UHD).

**Clock Distribution Module** - Hardware module that provides high-accuracy time and frequency reference distribution. We used the OctoClock-G CDA-2990, which includes an internal **GPS Disciplined Oscillator (GPSDO)** that works by disciplining (or steering) the oscillator output to a GPS device or GNSS satellite signal via a tracking loop. This module is connected to the **SDR** and minimizes issues related to the phone not finding the cell and/or not being able to stay connected due to inaccurate clocks at the gNBs RF frontend.

**srsRAN** - Open-source software suite designed to implement 4G and 5G cellular network technologies. It provides a complete stack for both the Radio Access Network (RAN) and the Core Network. Deployed on a USRP SDR enables the creation of a fully functional mobile network, capable of supporting the latest wireless communication standards. We used **srsRAN Project** that is specifically tied to 5G SA networks.

### 4.3.2 5G Core Network

Enterprise and Open-Source solutions exist to deploy a **5G CN**. For our use-case, we choose **Open5GS**<sup>2</sup>, a C-language open-source implementation for 5G Core and Evolved Packet Core (EPC). The 5GC was initially an implementation of an Evolved Packet Core (EPC), later updated to support the 5G architecture. Open5GS contains a series of software components and Network Functions that implement the 4G/5G NSA and 5G SA core functions, but in our testbed, only 5G SA components were used. Several Network Functions are implemented but we will not explain them all, since this is not the scope of this thesis. It's relevant to know that

**User Plane Function (UPF)** - connected with the gNB over the N3 interface, carries the user plane traffic between the gNB and the Internet.

**Policy Control Function (PCF)** - connected over the N5 Interface with the Application Function (in our case **MCXPTT**), is responsible for managing policies that regulate various aspects of the network. These policies encompass a wide range of functions, including quality of service (QoS), network resource allocation, authentication, mobility, security, and more.

**Access and Mobility Management Function (AMF)** - Receives all connection and session-related information from the User Equipment (UE) over the N2 Interface. Still, it is responsible only for handling connection and mobility management tasks. All messages related to session management are forwarded over the N11 interface to the Session Management Function (SMF), which is not displayed in our diagram.

The entire 5G Core Network has been deployed on a **Kubernetes (K8S)** cluster with 3 Virtual Machines, respectively 1 Master node and 2 Worker nodes, using **Helm Charts**<sup>3</sup> generated by Gradiant, an organization that is actively researching and developing Cloud-Native Network Functions with special focus on 5G network and evolvable to 6G technologies.

---

<sup>2</sup><https://open5gs.org/>

<sup>3</sup><https://gradiant.github.io/5g-charts/>

### 4.3.3 Testing

The following tools were installed on the Attacker Machine illustrated in Figure 4.1, and used during our tests.

**Nmap** - alias for "Network Mapper", is a free and open-source solution mainly used to discover available hosts and open ports in the network. It supports TCP and UDP scans and provides pre-built scripts to fingerprint services and other information about the host.

**Wireshark** - The most known open-source solution for network traffic sniffing and protocol analysis. Supports packet dissection for a lot of protocols, including telephony-related ones, such as SIP and SRTP. Supports live packet capture or analysis of saved captures in specific file formats like pcap or pcapng. Runs seamlessly on most Unix systems and Windows ones.

**Burpsuite** - A very popular tool to conduct pentesting and vulnerability assessment of web applications. A community, professional, and enterprise version of this product is available, all tests made in our work can be performed with the community version, which doesn't require a license. It offers a large number of features, only ones relevant to this work are described below:

- **Proxy and Interceptor:** Intercept HTTP requests and allows users to edit body parameters, cookies, and headers on the fly.
- **Repeater:** Allow users to modify and repeat captured requests saved on the HTTP History.
- **Extensions:** Burpsuite supports extensions, that can be even developed by the community, mainly used to test specific technologies or vulnerabilities. For instance, **JWT Editor**, is an extension that decodes **JSON Web Tokens** and is capable of performing attacks on them.

**Sippts** - Offers a set of tools used to audit VoIP servers and devices using SIP protocol.

It supports several tools, some of which are listed below:

- **scan**: TCP and UDP scan of SIP servers to identify open ports and services.
- **invite**: checks for misconfiguration that can be abused to make calls without authentication.
- **send**: send a customized SIP message and analyze the response.
- **flood**: send unlimited SIP messages to the target.

**Scapy** - Powerful Python-based interactive packet manipulation program and library that can forge or decode packets of a wide number of protocols, send them on the wire, capture them, store or read them using pcap files, match requests and replies, and much more. Scapy is usable either as a shell or as a library, making it possible to develop your own tools.

**Ffuf** - Acronym of **F**uzz **F**aster **U** **F**ool, is a fast web fuzzer written in Go. Supports a huge amount of filters and match rules on status codes, content length, and number of words of HTTP responses, recursion to fuzz on already found subdirectories, and an interactive mode to add/remove rules on the fly without restarting the process from scratch. Like other web fuzzers, it makes use of **wordlists**, text files containing words, or specially crafted payloads that will be tried one by one in an automated way. Since there are a lot of variables that can be tested, such as request parameters, headers, cookies, and even HTTP verbs, one may be wondering how Ffuf knows where to perform tests. Ffuf has a special keyword **FUZZ**, all uppercase, that is used as a placeholder to identify where to substitute entries read from the wordlist. An example of such usage is shown in [5.5.5](#)

**Apache HTTP Server** - The Apache HTTP Server (“httpd”) is the most commonly used web server on Linux systems. It allows to host content on the local machine and receive HTTP requests with both GET and POST methods. Additionally, has a built-in functionality that logs all the requests to a specified file.

# Chapter 5

## Scenarios

This chapter presents various considerations that were made to choose the entities to analyze, the assumptions made to conduct the analysis, a brief overview of the identified **threat scenarios**, and their mapping according to the **STRIDE** model. Next, each scenario describes which procedures, explained in Chapter 2, are involved, and the identified misconfiguration(s). In the final part, we provide a possible way to test for some of the proposed scenarios using the tools and capabilities outlined in Chapter 4. The purpose of **threat scenarios** is to help other Mission Critical vendors identify potential misconfiguration or vulnerabilities in their software or provide some insights that can be used to develop similar tests on their specific solution. Additionally, we think that some scenarios could help developers understand peculiar aspects of some procedures that may be dangerous when adjusted from third-party implementations to satisfy mission-critical specifications, or directly not well explained in 3GPP standards.

### 5.1 Choice of entities

Several considerations were made in this process. Based on the **Security Zones** illustrated in Figure 4.2, the following explains why scenarios on different networks were excluded and the reasons that led us to focus the work on a single zone.

1. **5G Core Network** - Several works have been done on security aspects of the 5G core network, yet none of them applied to a mission-critical use case, but still we believe that the scenarios would be the same as those identified in a generic threat analysis. There is one exception, which is the interaction between the P-CSCF and the PCF over the **N5 interface**, mentioned in [2.2.2.1](#), which was intended to be analyzed, but due to some NAT issues in the testbed, it was not possible to do so. For these reasons, it was decided to exclude further work on this zone.
2. **Management Network** - This subnet represents a potentially advantageous entry point for an attacker, as it allows for interaction with non-exposed entities located within the **Microservices Network**. Furthermore, having access to the "Configurator" or "Monitoring" application could have significant implications for the overall security of a Mission-Critical organization. Even though the likelihood of a "configurator" web application being used to configure the entire organization is very high, the 3GPP doesn't mention the need or existence of such an application in any of its documents. Therefore, its presence and implementation can vary widely among mission-critical vendors. The same applies to the "monitoring" application, since there are many solutions on the market, the choice would be too variable and therefore not valuable for our work. For these reasons, it was decided to exclude potential scenarios that could arise from this security zone.
3. **Microservices Network** - This subnet, like the previous one, represents an optimal entry point for an attacker, since the network traffic between entities, that are not exposed to the clients, is typically not confidentiality or integrity-protected unless required by the standard. Nevertheless, gaining access to this security zone is challenging, as it requires either obtaining access to one of the containers or directly accessing the virtual machine (VM) where all the containers are running. Moreover, the configuration of these microservices may differ between mission-critical providers or even with different types of deployment of the same software (e.g. as explained in [4.2.3](#), we used the "All-in-One" solution, still one based on Clusters exists). As a result, potential scenarios in this security zone were not included in the analysis.

In light of the considerations made above, and in an effort to maintain a good trade-off between probability and standards compliance, it was decided to focus the work only on entities reachable within the **Public Network**. Although the *Firewall* is shown in the diagram in 4.2, it wasn't present in our testbed. However, in the analysis, we assumed its presence, considering only those entities that are required to be accessible by clients and, thus, with ports exposed even in the event of a Firewall.

The analyzed system components are listed below with an overall description of their functionalities. For a detailed explanation of the system architecture, refer to Chapter 2.

**SIPCore** - Based on RFC 3261, it's the main component of the entire signaling plane. Responsible for authenticating and authorizing clients within the SIP-based network, initiating, maintaining, and terminating SIP sessions. The SIPCore usually is exposed on **port 5060**, used to reach the **P-CSCF**, explained in 2.2.2.1.

**Identity Management Server (IDMS)** - Based on OpenID Connect 1.0, is responsible for the authentication of users within the Mission Critical organization. It issues an `access_token`, used to authorize requests for certain resources, through the "Authorization: Bearer" HTTP Header. These technologies are very prone to misconfiguration since security relies almost entirely on developers using the right combination of configuration options and implementing their own additional security measures on top. In short, there's plenty of opportunity for bad practice to creep in.

**Configuration Management Server (CMS)** - Implements an HTTP server used to distribute XML documents, containing configuration parameters of each user. In addition, it issues notifications to subscribed clients when a new version of the document is available, so everyone is provided with the latest updates on configurations.

**Group Management Server (GMS)** - Implements an HTTP server used to distribute XML documents containing configuration parameters of groups. In addition, it issues notifications to subscribed clients when a new version of the document is available, so everyone is provided with the latest updates on configurations.



**File Repository** - One of the main components of the MCDData service, it can be used to store files uploaded directly in the repository or attachments sent in private and/or group chat.

**Bootstrap Service** - It is crucial to retrieve information about the MCS UE initial configuration management object (MO) and default MCX user profile configuration management object (MO).

**User Equipment** - Mobile device used by first responders with the MCXPTT client application installed. It might be useful to know what information can be recovered or what are the potential threats that may arise, for example, if the UE can reach and send requests to 5G Network Functions, in case the device is stolen or if a malicious user accesses it, even for a short period. For this purpose, we assumed that the attacker could unlock the stolen phone but the user registered from it has been disabled and cannot access the client application or use its functionalities.

## 5.2 Assumptions

Before presenting the scenario list and its description, it is essential to clarify that the following assumptions have been made for this work.

- An attacker is within the **Public Network** and can access services with exposed ports, either using its own web browser or User Equipment.
- An attacker can eavesdrop on network traffic.
- Given the context in which **Mission-Critical** applications are utilized, an attacker is highly motivated and possesses a good understanding of the domain.

## 5.3 Overview

This section presents an overview of the identified scenarios, derived from an analysis of 3GPP Technical Specifications (TS), Technical Reports (TR) documents, and a practical assessment of the **MCXPTT** solution. The overview is presented in Table 5.1 that is composed of the following columns:

**ID** - An identifier of the threat scenario in the form TS\_number.

**Overview** - Brief description of the scenario.

**STRIDE** - Mapping of the scenario within the STRIDE domain.

**Trust Level** - Indicates what level of privileges an attacker is required to have. In the table, three different privileges are mentioned, and the description is provided below

- **Anonymous User:** The attacker is only required to be within the network, which is the base assumption of this work.
- **User with login credentials:** The attacker must have obtained in some way the credentials to login inside one of the web applications (Dispatcher or Discrete Listener).
- **User with valid access\_token:** The attacker must have valid access\_token (not expired) that has the authorization for the scopes outlined in 2.4.3.2.

**Showcased:** Indicates whether we provided a way to test for the scenario or not. Only scenarios that have the **checkmark**(✓) symbol in this column are listed in Section 5.5.

ID	Overview	STRIDE	Trust Level	Showcased
TS_01	Authorization Code theft due a malicious redirect_uri parameter registered exploiting unprotected dynamic client registration	STRIDE	Anonymous User	✓
TS_02	User login credentials theft due to a malicious redirect_uri parameter registered exploiting unprotected dynamic client registration	STIDE	Anonymous User	✓
TS_03	Sensible information disclosure through Server Side Request Forgery (SSRF) via logo_uri registered exploiting unprotected dynamic client registration	TI	User with login credentials	✓
TS_04	Flooding SIPCore with REGISTER messages in order to overload and slow down the entire signaling plane	D	Anonymous User	✓
TS_05	IP Spoofing through NAT re-assignment	SD	Anonymous User	✗
TS_06	SIPCore P-CSCF Bypass to circumvent authorization mechanisms	SD	Anonymous User	✗
TS_07	Sensitive information disclosure via Path Traversal on the File Repository component	I	User with a valid access_token	✓
TS_08	Sensitive Information Disclosure via Bootstrap Service	I	Anonymous User	✓
TS_09	Tampering SRTP authentication tag	TD	Anonymous User	✗

**Table 5.1:** List of Scenarios

## 5.4 Description

This section provided a more detailed description of scenarios listed in 5.1. It should be noted that TS\_01, TS\_02 and TS\_03 are presented under Section 5.4.1, which outlines the misconfiguration that opens the doors for such scenarios.

### 5.4.1 IDMS - Unprotected Dynamic Client Registration

As mentioned in 2.4.6, client registration is a fundamental step that must be taken in order to use the services offered by the IDMS. It's also stated that the registration can be done dynamically by contacting a so-called registration endpoint. The client registration request should contain the "Authorization" header to protect such a process from untrusted individuals. However, some providers will allow dynamic client registration without any authentication, which enables an attacker to register their own malicious client application. This can have various consequences depending on how the values of these attacker-controllable properties are used and a couple of them are shown below.

#### 5.4.1.1 TS\_01 Authorization\_Code theft

Before describing the scenario, it's relevant to point out that it follows what's written at the end of 2.4.5. More specifically, the scenario targets a single-page web application, with an attacker that operates from his own browser. As shown in Table 2.1, the `redirect_uri` parameter is used by the IDMS to know where to redirect the authentication response to receive the authorization code. Using the **Unprotected Dynamic Client Registration** misconfiguration, we can register our client application with a malicious `redirect_uri`, use it to hijack the authentication flow, steal the `authorization_code` and request an **access\_token** on behalf of the victim. The impact would be very high because this token grants access to many resources such as user configuration documents, group configuration documents, API used by Dispatchers or Discrete Listeners, and much more.

However, the following weaknesses make this attack vector harder:

1. Requires human interaction because the victim should open the hijacked link.
2. Stealing the `authorization_code` is not enough because **Proof Key for Code Exchange** (PKCE) protection is mandatorily required by the standard.

With regard to the first point, there is little that can be done. It is necessary to combine a certain degree of good fortune with social engineering in order to trick the victim into opening the link. Concerning the other one, some mechanisms can be leveraged to circumvent this protection, and a couple of them are described below.

### PKCE Downgrade Attack

PKCE Downgrade Attack is the first method that can be used to bypass such protection. Table 2.1 describes the `code_challenge` and `code_challenge_method` parameters, necessary for PKCE to apply. Next, Table 2.2 describes the `code_verifier`, generated at the beginning of the flow, stored somewhere, and then compared in the last request for the `access_token`. Following this preamble, the attack is composed of the steps below:

1. Attacker starts an authorization flow using its device, with a certain `code_challenge` in the authorization request.
2. Attacker intercepts the authorization request and removes the `code_challenge` parameter from it.
3. If the authorization server allows for flows without PKCE, it will create a code that is not bound to any `code_challenge`.
4. The attack now continues as explained in the first paragraph, so the attacker steals the `authorization_code`.
5. In the `access_token` request, the `code_verifier` parameter is omitted or sent as a placeholder with a random value, and since the **IDMS** sees that this code is not bound to any `code_challenge`, it will not check the presence or contents of the `code_verifier` parameter.

### **Code Verifier stored inside attacker-accessible location**

The second method that can be used to bypass **PKCE** protection is strictly related to where the `code_verifier` is stored when an authorization flow starts. If this variable is stored client side, in a location like the Browser's `localStorage`, an attacker can do the following steps

1. Start an authorization flow using its own device
2. `code_verifier` is stored in Browser's `localStorage`, then it can be easily accessed.
3. Steal the `authorization_code` via the malicious `redirect_uri` and request the `access_token` using the `code_verifier` saved at Step 2.

#### **5.4.1.2 TS\_02 Credentials theft through malicious `redirect_uri`**

This scenario is very similar to the previous one, instead of obtaining the `access_token` on behalf of the victim, the attacker obtains credentials used in the login process, as shown in Figure 2.6 - Step 3a. Once again this can be done via the `redirect_uri` to redirect the victim to a phishing page that resembles the original one but with the difference that credentials are sent to the attacker server instead of the legitimate one. In terms of stealthiness, this is less efficient compared to 5.4.1.1, because if the target application doesn't support the same user attached to multiple sessions, the moment the attacker logs in, the legitimate user would be logged out, noticing that something strange is happening.

#### **5.4.1.3 TS\_03 SSRF through `logo_uri`**

In the table 2.1, one can notice that a `logo_uri` parameter is present. This is one of the parameters that are defined as optional, indeed from TS 33180 Annex B.3 [9] "Other information about the client such as application name, website, description, logo image, legal terms to be consented to, may optionally be registered." The `logo_uri` is used by the **IDMS** to know where the logo image of the application is stored, so each time the application page is accessed, that URL is fetched and the logo is displayed.

Let us assume that the attacker knows the location of a page that contains sensitive information, such as <https://mcx.server/config/db-configuration>. However, the attacker is unable to access this page because it is only reachable from the internal network. If the URL of this page is set as the `logo_uri` during the client registration phase, it will be fetched each time the application is accessed with that specific `client_id`. This would result in the exposure of the content with the database configuration, which would normally be denied.

#### 5.4.2 TS\_04 SIPCore - SIP REGISTER Flooding

The SIPCore is designed in a way that accepts only REGISTER requests that include the Authorization header, then as an anonymous user, we can't directly affect other users. However, in a scenario where the MCX organization has a lot of users, an attacker can flood the P-CSCF with REGISTER requests causing a slowdown of the service or even its denial if the requests make the server crash.

#### 5.4.3 TS\_05 SIPCore - IP Spoofing through NAT re-assignment

If the "IP Address check" is used as an authentication mechanism for non-registration messages and the User Equipment (UE) loses connection without deregistering, it's very likely that the access network consequently re-assigns the IP address to another user, or a NAT re-assigns the port to another user. In such case, an attacker repeatedly attaches to the network hoping to be assigned the IP address or port of another user who dropped off without deregistering in IMS. If this indeed happens then any **non-registration** message sent by the attacker would be accepted by the IP address check mechanism in the P-CSCF as coming from the previous user.

#### 5.4.4 TS\_06 SIPCore - P-CSCF Bypass

As outlined in Section 2.2.2.1 and Section 2.2.2.2, the standard flow of the signaling plane communication normally happens between the Signaling User Agent located in the **MCX UE** and the **P-CSCF**, over the SIP-1 Reference Point.

However, in case the **S-CSCF** is exposed, a malicious user could try to bypass the P-CSCF and send SIP messages directly to the S-CSCF, which doesn't require authentication since messages usually come from the P-CSCF, considered a trusted entity. This is because some nodes in the IMS domain will trust SIP messages that contain one or more "asserted identity" headers. If a malicious user manages to bypass the P-CSCF, the following problems may arise:

- The P-CSCF would not be able to generate any charging information.
- The malicious user can spoof the identity of another legitimate user and potentially send non-registration messages, such as INVITE or BYE, on its behalf.

#### 5.4.5 TS\_07 File Repository - Path Traversal

Since one of the capability functions of the **MCDData** service is file distribution as explained in 2.1.3, for sure there will be a **File Repository** component with some functionalities to retrieve files posted directly to such repository or sent as attachments to group or private messages. An attacker who can steal the `access_token` as in TS\_01, would be authorized to access the **file download** functionality of the File Repository. If the download functionality uses request parameters like `file` or `file_id` that are not properly sanitized, an attacker could gain access to the content of sensitive files that are not supposed to be red.

#### 5.4.6 TS\_08 Information Disclosure via Bootstrap Service

As previously outlined in Chapter 2, the **Bootstrap** service is utilized by User Equipment to retrieve preliminary configuration documents. It should be noted that no authorization is necessary to access such information, as the URL for the **IDMS** is obtained during this preliminary phase. It may happen that to reduce the complexity of the system, the same **bootstrap** service is re-used to provide configuration data to other entities in the system, even non-standard compliant ones. This can cause potential disclosure of sensitive information, in case some resources don't require authorization or bad assumptions are made about the attacker's capabilities.



### 5.4.7 TS\_09 - Tampering SRTP authentication tag

As described in [9], the confidentiality data authentication of SRTP packets is applied using AEAD\_AES\_128\_GCM algorithm. It also states that: "The SRTP authentication tag may be appended to every 'r-th' packet as defined in IETF RFC 4771 [24] to provide the SRTP ROC counter to MC UEs performing a late-entry to the communication. A 'mode 3' integrity transform (RCCm3) shall be supported for transmitting the ROC within a 4-octet SRTP authentication tag." However, 'mode-3' represents the NULL-MAC (NULL-Message Authentication Code), which means that no integrity protection is applied to this r-th transmitted packet. For this reason, if an attacker modifies the ROC, the modification will go undetected by the receiver, resulting in the latter's loss of cryptographic synchronization until the next correct ROC is received. This implies that an attacker can perform a Denial of Service attack by only modifying every r-th packet.

## 5.5 Testing

The following section will provide some examples of how the scenarios can be tested or some of the described vulnerabilities can be identified. As already outlined in Section 5.3, only scenarios that have the **checkmark**(✓) symbol in the "Showcased" column in Table 5.1, are listed in this section.

### 5.5.1 TS\_01

To test for scenario TS\_01 we need to perform the following steps:

- Test for Unprotected Dynamic Client Registration and allowed redirect\_uris
- Test for PKCE Downgrade Attack

Firstly, we begin by testing if the OIDC Dynamic Client Registration is allowed without authorization and if the redirect\_uri parameter can be an arbitrary one or some kind of blacklist is applied. Send the request in Listing 1 to the **IDMS registration endpoint**.

```
POST /idms/register HTTP/1.1
Content-Type: application/json
Accept: application/json
Host: idms.example.com

{
  "application_type": "web",
  "redirect_uris":
    ["https://evil.com/callback"],
  "client_name": "TS-01",
}
```

**Listing 1:** TS.01 - Unprotected Dynamic Client Registration Request

In the event that the IDMS returns a response with **status code 201 - Created** and some fields like `client_id`, `client_secret`, etc. in its body, then it means that it's vulnerable and a new malicious client application has been registered. The misconfiguration lies in the fact that no **"Authorization"** Header has been provided in the request, allowing anyone to register their client.

Then we can proceed to test if the server is vulnerable to **PKCE Downgrade Attack**, using the **Proxy and Interceptor** functionality of the tool **Burpsuite**, described in [4.3.3](#). The following steps need to be performed in order to test for the vulnerability:

- Intercept the **authentication request** that contains the parameters described in [Table 2.1](#).
- Remove the `code_challenge` parameter and forward the request.
- Forward subsequent requests and stop on the last one to the **token endpoint**, which will have parameters listed in [Table 2.2](#).
- Remove the `code_verifier` parameter and observe the response.

If the response returns a **status code of 400** (indicating a "Bad Request") accompanied by an error description such as "Missing code verifier", it is plausible that the IDMS performs a verification of the parameter's presence within the request. In such an instance, it is recommended to attempt to provide the code verifier with a random value and observe the resulting response. In the event that the server again returns a response with a status code of 400 (Bad Request) and an error description similar to "PKCE verification failed", then this indicates that the IDMS is not vulnerable to the PKCE downgrade attack. Conversely, if the request to the **token endpoint** is made without supplying the code\_verifier or providing it with a random value and a **200 OK** response is returned with the access token, id token, and other parameters listed in 2.3, this indicates that the IDMS is vulnerable.

In case the IDMS is vulnerable to the two described misconfigurations, then scenario TS\_01 can be reproduced.

## 5.5.2 TS\_02

Scenario TS\_02 makes use of the **Unprotected Dynamic Client Registration**, already described in the previous scenario TS\_01 (5.5.1). To test its reproducibility send the following request to your **IDMS registration endpoint**

```
POST /idms/register HTTP/1.1
Content-Type: application/json
Accept: application/json
Host: idms.example.com

{
  "application_type": "web",
  "redirect_uris":
    ["https://evil.com/callback"],
  "client_name": "TS-02",
}
```

**Listing 2:** TS.02 - Unprotected Dynamic Client Registration Request

As in TS\_01, if the IDMS returns a response with **status code 201 - Created** and some fields like `client_id`, `client_secret`, etc. in its body, then it means that it's vulnerable and a new malicious client application has been registered. In this scenario, the malicious **redirect\_uri** is still employed but with a different objective. In TS\_01 the goal is to steal the **access\_token** and access resources that require authorization. Meanwhile, in TS\_02 the goal is to steal the victim's username and password and directly access the single-page web application interface.

### 5.5.3 TS\_03

Scenario TS\_03 as well as TS\_01 and TS\_02 depends on the **Unprotected Dynamic Client Registration** misconfiguration. However, this one exploits the `logo_uri` parameter instead of the previously used `redirect_uri`. As explained in 5.4.1.3, it is chosen a known location that contains configuration information of databases in production, like this one <https://mcxserver.example.org/config/db-configuration>. To test for this scenario, firstly a new client needs to be registered, using the request below.

```
POST /idms/register HTTP/1.1
Content-Type: application/json
Accept: application/json
Host: idms.example.com

{
  "application_type": "web",
  "redirect_uris":
    ["https://mcx.server/idms/oidc-callback"],
  "logo_uri": "https://mcx.server/config/db-configuration",
  "client_name": "TS-03",
}
```

**Listing 3:** TS.03 - Unprotected Dynamic Client Registration Request

It can be noted that this time the `redirect_uri` parameter is set to the legitimate one, while the `logo_uri` is set as the endpoint with the configurations of databases, normally accessible only from services inside the internal network.

Since the `logo_uri` is optional, now it's needed to test if the IDMS is using the logo functionality. This can be done with the following steps

- Go to a web application that uses OIDC, such as Dispatcher or Discrete Listener.
- Perform the login using username and password
- If after a successful login, one is prompted in front of a page that asks for authorization to a list of information, similar to the one in 5.1, this indicates that the application is using the `logo_uri` to retrieve the logo displayed under the application name, then scenario TS\_03 can be reproduced.



**Figure 5.1:** Example of application that uses `logo_uri`

## 5.5.4 TS\_04

Scenario TS\_04 consists of sending a high amount of SIP REGISTER requests for an extended period to overload and slow down the signaling plane. This can be achieved using the **Sippts** tool described in 4.3.3. More specifically the **flood** functionality of **sippts** can do this work with the following command

```
sippts flood -i <server-ip> -r 5060 -m REGISTER -p tcp
```

A brief description of the flags is present below:

- **-i:** specifies the IP address of the SIP server (SIPCore or IMS)
- **-r:** specifies the remote port (5060 is the default port for SIP servers)
- **-m:** specifies the SIP method used in the requests, in our case the **REGISTER** method was used, but **OPTIONS** and **INVITE** are supported.
- **-p:** specifies the protocol to use (TCP, UDP, and TLS are supported).

Additional flags can be provided to customize the attack, such as:

- **-n:** specifies the number of requests, if left blank as in our command, the default is non-stop.
- **-b:** specifies to send malformed headers that can be used to fuzz and identify bugs in the request parser.

To verify that the attack is working as expected, the usage of the CPU on the SIPCore/IMS container can be monitored using tools like **htop**. In the event that the test is conducted with a considerable number of users, it is recommended to observe the registration/de-registration or affiliation/de-affiliation processes to identify any potential slowdowns.

### 5.5.5 TS\_07

Before testing if the application is vulnerable to Scenario TS\_07, the following steps need to be performed:

- Login into the Dispatcher web application and grab the `access_token` which can be used to access the **File Repository** services.
- Download a file that can be either stored in the File Repository or sent as an attachment in a message and copy the URL of the request. For instance, we assume that the URL for such a request is the one below, where the **”file”** parameter expects the id of the requested file. [https://mcxserver.example.org/file-repository/download?file=<file\\_id>](https://mcxserver.example.org/file-repository/download?file=<file_id>)

Using the tool **Ffuf** described in 4.3.3, one can effectively test if the server is vulnerable to Path Traversal with the command below:

```
ffuf -u 'https://mcxserver.example.org/file-repository/download?file=FUZZ'  
-w wordlist.txt -H 'Authorization: Bearer <access_token>'
```

A brief description of the flags used in the command is provided:

- **-u**: specifies the full target URL, including parameters.
- **-w**: specifies the wordlist with the payloads that will be tested. To test for this scenario, the [LFI-Jhaddix<sup>1</sup>](#) wordlist can be used.
- **-H**: specifies Headers that will be included in the requests. For instance, we specified the **”Authorization”** header which is mandatory to access functionalities of the MCDATA service.

---

<sup>1</sup><https://github.com/danielmiessler/SecLists/blob/master/Fuzzing/LFI/LFI-Jhaddix.txt>

In order to test if one of the payloads worked, some filtering needs to be applied to show only **unique** responses. For instance, if all responses return a status code 400 - Bad Request, we want to isolate these outputs, as it is known that they correspond to a failed test. Responses can be filtered by providing additional flags to the previous command. An example of how all responses with status code 400 are excluded is shown below:

```
ffuf -u 'https://mcxserver.example.org/file-repository/download?file=FUZZ'  
-w wordlist.txt -H 'Authorization: Bearer <access_token>' -mc all -fc 400
```

The filter and matching process can vary considerably between different applications. Therefore, it is recommended to adapt the command above to suit the specific requirements of the intended use case. If some responses are shown after enabling filters, it's very likely that these are successful payloads, then need to be verified manually. For instance, if the output of Ffuf exhibits a request comprising the payload `../../../../../../../../etc/passwd` and the status code is 200, indicating a positive outcome, an attempt should be made to send such a request manually in order to observe the resulting response. If the response shows the content of the `passwd` file located in the `etc` directory, this indicates that the intended application is vulnerable to TS.07.

### 5.5.6 TS\_08

Scenario TS.08 does not have precise steps to follow in order to test it, since the names of paths and URLs can vary across different implementations. This section showcases a potential alternative in which this misconfiguration could arise. For instance, it is supposed that the User Equipment zend a request to this URL <https://mcxserver.example.org/bootstrap/init.xml?profile=ue> to retrieve the initial configuration document. In this example, we assume that the content illustrated in Listing 4 is returned in the response.



```
<ue-init-config>
  <idms-uri>https://mcx.example.com/idms</idms-uri>
  <cms-uri> ..
  <gms-uri> ..
</ue-init-config>
```

**Listing 4:** Example response of UE initial configuration document

If the same method to obtain preliminary data is used, for instance, by the **SIPCore**, it can happen that sensitive information is exposed because assumed that the request is solely initiated by the SIPCore itself and not by an adversary who changes the **profile** parameter. Consequently, a request to <https://mcxserver.example.org/bootstrap/init.xml?profile=sipcore> may return the following content with **credentials** of the SIP database described in 2.2.2.4.

```
<sipcore-config>
  <P-CSCF-uri>p-cscf@example.org</P-CSCF-uri>
  <db-credentials>
    <username>testuser</username>
    <password>testpassword</password>
  </db-credentials>
</sipcore-config>
```

**Listing 5:** Example response of configuration document with sensitive information

As said at the beginning of the section, the testing phase can vary across Mission-Critical applications. In the case that the **Bootstrap** service is implemented as a web server, the advice is to collect **paths** and **parameters** used for the bootstrap procedure in one or more wordlists and **fuzz** through them. To correctly perform the test is recommended to omit the "Authorization" header and observe responses with status codes different from 401, 403, and 404, respectively indicating Unauthorized, Forbidden, and Not Found. As in the previous scenario TS\_07 it's very likely that filtering and matching need to be applied to isolate only meaningful responses.

# Chapter 6

## Experimental evaluation

This chapter will demonstrate how an attacker located in the **Public Network** as outlined in Section 5.1 can reproduce the scenario TS\_01 described in 5.4.1.1. In the previous chapter, Section 5.5 demonstrated how to test whether such a scenario can be reproduced or not. In this section, we will instead present a detailed account of how to perform the full attack, still making use of the misconfigurations described in section 5.4. The reason behind the choice of demonstrating this attack is related to the fact that it is the most complex one as it requires human interaction, exploits more than one misconfiguration, and has a high impact in terms of resources that can be accessed. The chapter starts with an initial reconnaissance phase and then presents the attack demonstration.

### 6.1 Ethical Considerations

The exploration of the system and the practical attack presented in this chapter have been conducted on a development version of MCXPTT, used only in laboratory settings and for testing purposes. The work has been done under the supervision of the Cyber Security team of the Leonardo Cyber Security R&D Laboratory, which is an authorized and closed environment. The identified misconfigurations were duly reported, following company regulations, to the development team, with whom we communicated throughout the process. The issues were promptly addressed and resolved. Additionally, we have notified a possible update to the 3GPP, that can help to maintain completeness of technical specification documents and avoid misconfigurations raised due to standard's lack of detail.

## 6.2 Reconnaissance

The description of the scenario, presented in Section 5.4.1.1, mentions the usage of **Unprotected Dynamic Client Registration** misconfiguration, outlined in Section 5.4.1, that allows an attacker to register a malicious client using the **registration endpoint**. Apriori, the attacker doesn't know the precise location of such an endpoint, but this can be retrieved easily with the following steps:

1. Using the tool **Nmap** outlined in Section 4.3.3, discover the port where the SIP-Core/IMS is running with the command in Figure 6.1

```
( ⚡ ) nmap -p 5060 192.168.7.10
Starting Nmap 7.80 ( https://nmap.org ) at 2024-09-03 11:17 CEST
Nmap scan report for 192.168.7.10
Host is up (0.00062s latency).

PORT      STATE SERVICE
5060/tcp  open  sip

Nmap done: 1 IP address (1 host up) scanned in 0.09 seconds
( ⚡ ) █
```

**Figure 6.1:** SIPCore port discovery

In our scenario, the SIPCore is running on port 5060, which is shown to be opened and running the SIP service.

2. Using the tool **Sippts**, outlined in Section 4.3.3, launch a scan against the SIPCore using a REGISTER message. Figure 6.2 shows an example command to do so

```
( ↵ ) sippts scan -vv -i 192.168.7.10 -r 5060 -p all -m REGISTER
📞 SIPPTS BY PEPELUX
SIP SCAN
📄 https://github.com/Pepelux/sippts
🐦 https://twitter.com/pepeluxx
[✓] IP/Network: 192.168.7.10
[✓] Port range: 5060-5061
[✓] Protocols: UDP, TCP, TLS
[✓] Method to scan: REGISTER
[✓] Used threads: 6
```

Figure 6.2: Sippts scan command

Since the SIP protocol can flow on TCP, UDP, and TLS, the scan is run using all supported protocols in order to miss false negatives.

Sippts sent a REGISTER request and received a response from the SIPCore. The conversation is showed below in Figure 6.3

```

[+] Sending to 192.168.7.10:5060/TCP ...
REGISTER sip:192.168.7.10 SIP/2.0
Via: SIP/2.0/TCP 10.0.0.1:43667;branch=hn7o7jud4jdukbqlfzbz87qylibq0d9q7u66yycx30o4jmtkon30hcmmf
From: <sip:100@192.168.7.10>;tag=b1b7a7be
To: <sip:100@192.168.7.10>
Contact: <sip:100@10.0.0.1:43667;transport=TCP>;expires=120
Call-ID: e2c924628344029c612f06a5d2ce6030
CSeq: 1 REGISTER
Max-Forwards: 70
User-Agent: pplsip
Allow: INVITE, REGISTER, ACK, CANCEL, BYE, NOTIFY, REFER, OPTIONS, INFO, SUBSCRIBE, UPDATE, PRAC
Expires: 120
Content-Length: 0

[-] Receiving from 192.168.7.10:5060/TCP ...
SIP/2.0 401 UNAUTHORIZED
Via: SIP/2.0/TCP 10.0.0.1:43667;branch=hn7o7jud4jdukbqlfzbz87qylibq0d9q7u66yycx30o4jmtkon30hcmmf
From: <sip:100@192.168.7.10>;tag=b1b7a7be
To: <sip:100@192.168.7.10>;tag=TOTAGe2c924628344029c612f06a5d2ce6030
Contact: <sip:100@10.0.0.1:43667;transport=TCP>;expires=120;expires=120
Supported: replaces, 100rel, norefersub, timer
Call-ID: e2c924628344029c612f06a5d2ce6030
CSeq: 1 REGISTER
User-Agent: Mission Critical System 1112 2023
Allow: INVITE, ACK, CANCEL, MESSAGE, BYE, SUBSCRIBE, NOTIFY, INFO
WWW-Authenticate: Bearer authz="https://192.168.7.10/idms/.well-known/openid-configuration"
WWW-Authenticate: Digest realm="test.org", nonce="4B3900EE3A8DA5B62EFF2E2C", qop="auth", algorit
Content-Length: 0

-----
| IP address | Port | Proto | Response | User-Agent | Type |
-----
| 192.168.7.10 | 5060 | TCP | 401 UNAUTHORIZED | Mission Critical System 1112 2023 | Server |
-----

```

Figure 6.3: Sippts REGISTER request & response

As expected, the response returns a 401 Unauthorized, but there's an interesting Header **WWW-Authenticate** that occurs two times and in its first occurrence contains a paramter `authz` equal to this endpoint `https://<IP-Address>/idms/.well-known/openid-configuration`. This endpoint is known as the **OIDC discovery url** and its exposure is not a misconfiguration, indeed it is used by authentication libraries and relying parties to discover authentication URLs, public signing keys, and other service metadata. Visiting this endpoint will reveal the location of the **registration endpoint** as illustrated in Figure 6.4

```
"code_challenge_methods_supported": [
  "S256"
],
"end_session_endpoint": "https://192.168.7.10:443/idms/session/end",
"grant_types_supported": [
  "implicit",
  "authorization_code",
  "refresh_token",
  "urn:ietf:params:oauth:grant-type:device_code"
],
"id_token_signing_alg_values_supported": [
  "PS256",
  "RS256",
  "ES256"
],
"issuer": "http://localhost:3101",
"jwks_uri": "https://192.168.7.10:443/idms/jwks",
"registration_endpoint": "https://192.168.7.10:443/idms/reg",
```

Figure 6.4: Registration endpoint discovery

Once the registration endpoint has been identified, the reconnaissance phase is complete, and we can proceed to describe the attack. It should be noted that this is not the sole method for discovering the registration endpoint; however, we believe it is the most stealthy, as it involves only a few requests, rather than fuzzing, which generates a considerable amount of "noise."

### 6.3 Attack Demonstration

The first step of the attack is to register a new client with a malicious `redirect_uri` that corresponds to the IP address of the attacker. This is done by exploiting the **Unprotected Dynamic Client Registration** misconfiguration described in Section 5.4.1. The request in Figure 6.5 shows the registration of a client with a `redirect_uri` that points to the **attacker's machine**. It should be noted that the field of the IP address registered as `redirect_uri`, corresponds to the IP address of the attacker's machine illustrated in Figure 4.1.

```

Request
Pretty Raw Hex
1 POST /idms/reg HTTP/2
2 Host: 192.168.7.10:443
3 Cache-Control: max-age=0
4 Sec-CH-UA: "Not-A.Brand";v="99", "Chromium";v="124"
5 Sec-CH-UA-Mobile: ?0
6 Sec-CH-UA-Platform: "Linux"
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
  Gecko) Chrome/124.0.6367.60 Safari/537.36
9 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*
  ;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Sec-Fetch-Site: none
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Accept-Encoding: gzip, deflate, br
15 Accept-Language: en-US,en;q=0.9
16 Priority: u=0, i
17 Content-Type: application/json
18 Content-Length: 149
19
20 {
21   "redirect_uri": [
22     "http://192.168.7.241"
23   ],
24   "logo_uri":
25     "http://192.168.7.10:16077/internal/config/additions/serviceboot.xml?id=kms.test.org"
  }

Response
Pretty Raw Hex Render
11 X-Download-Options: noopen
12 X-Content-Type-Options: nosniff
13 X-Permitted-Cross-Domain-Policies: none
14 Referrer-Policy: no-referrer
15 X-XSS-Protection: 0
16 Pragma: no-cache
17 Cache-Control: no-cache, no-store
18 X-Frame-Options: DENY
19 Strict-Transport-Security: max-age=31536000; includeSubdomains; preload
20 X-Content-Type-Options: nosniff
21 Content-Security-Policy: default-src 'self' http: https: data: blob: 'unsafe-inline'
22
23 {
24   "application_type": "web",
25   "grant_types": [
26     "authorization_code"
27   ],
28   "id_token_signed_response_alg": "RS256",
29   "post_logout_redirect_uri": [
30   ],
31   "require_auth_time": false,
32   "response_types": [
33     "code"
34   ],
35   "subject_type": "public",
36   "token_endpoint_auth_method": "client_secret_basic",
37   "introspection_endpoint_auth_method": "client_secret_basic",
38   "revocation_endpoint_auth_method": "client_secret_basic",
39   "require_signed_request_object": false,
40   "request_uri": [
41   ],
42   "client_id_issued_at": 1716303145,
43   "client_id": "m1RtJwDrwmSq7uBc0N4_N",
44   "client_secret_expires_at": 0,
45   "client_secret":
46     "IGGO-jedC6e00L397EzuntgWopGp01TeX0QFMFgf1PyRSLIEVtIfj2M7bfoRctD7EepBaVVsJaxEw0TSL45A"
47   ,
48   "logo_uri":
49     "http://192.168.7.10:16077/internal/config/additions/serviceboot.xml?id=kms.test.org",
50   "redirect_uri": [
51     "http://192.168.7.241"
52   ],
  }

```

**Figure 6.5:** Client Registration with malicious redirect\_uri

After the registration, it is recommended to grab the values of client\_id, client\_secret, and redirect\_uri, which will be used in further steps.

The next phase is referred to as the effective attack and since it's composed of several steps, a **Flow Diagram** of such an attack is illustrated in Figure 6.6.

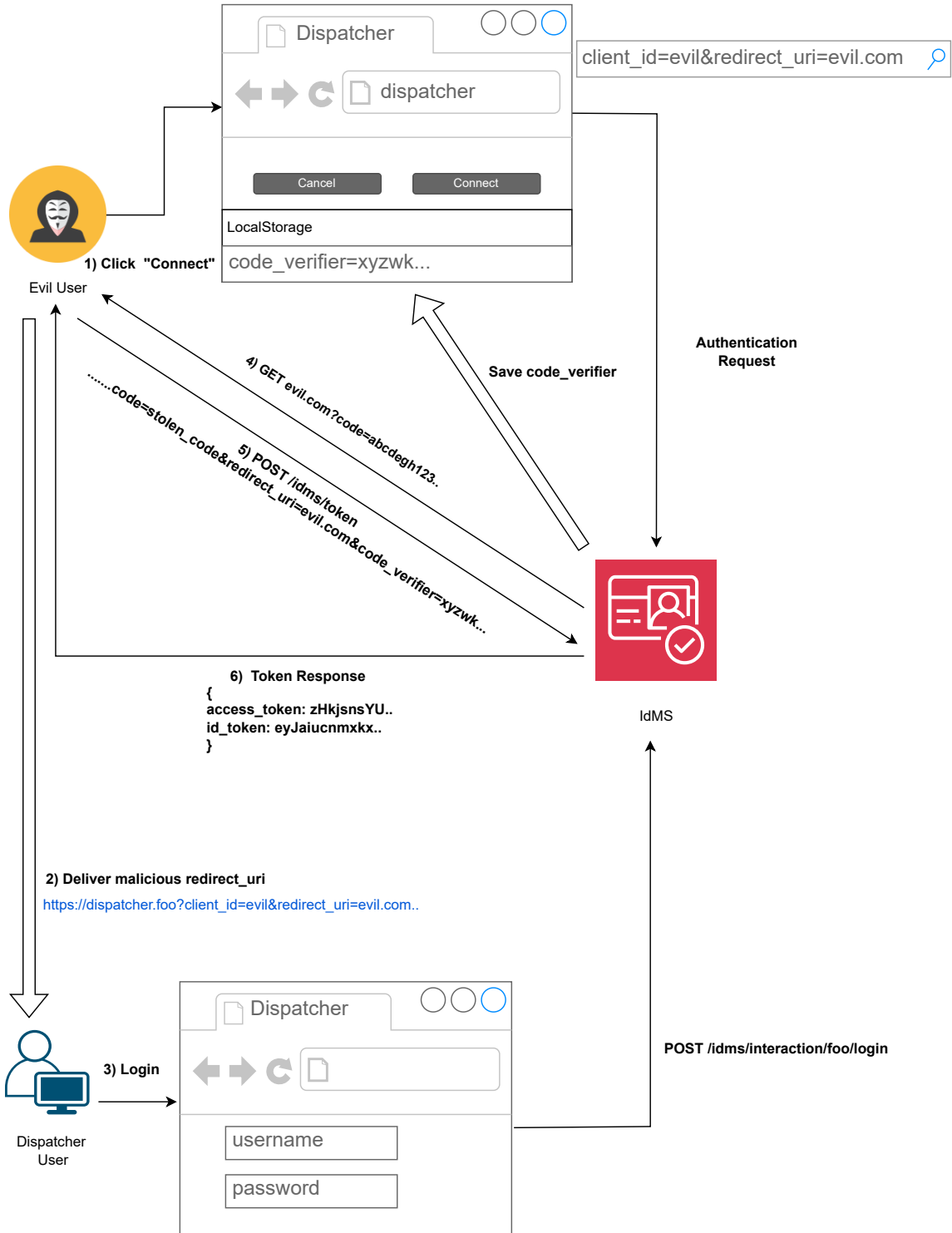


Figure 6.6: Attack Flow Diagram



The description of each step is provided below:

1. The attacker initiates the OAuth flow by clicking the "Connect" button, intercepts the Authentication request using the tool Burpsuite and substitutes the `client_id` and `redirect_uri` parameters with the ones belonging to the malicious client registered before. The `client_id` of the malicious client is `m1RtJWDrWm5q7uBcON4_N` and the `redirect_uri` is `http://192.168.7.241`, then the tampered request is the one in Figure 6.7.

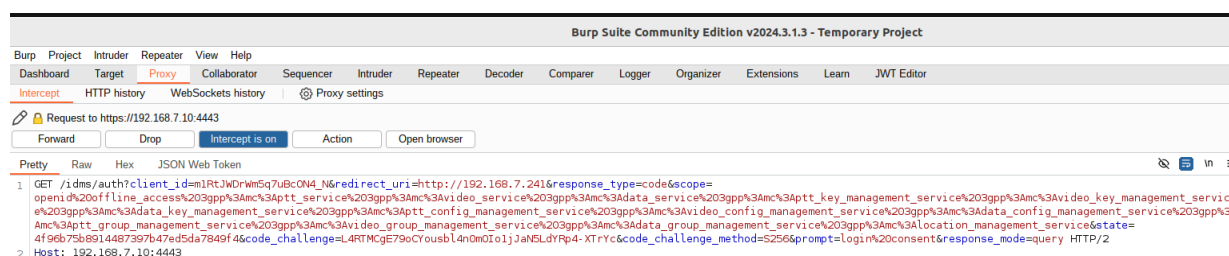
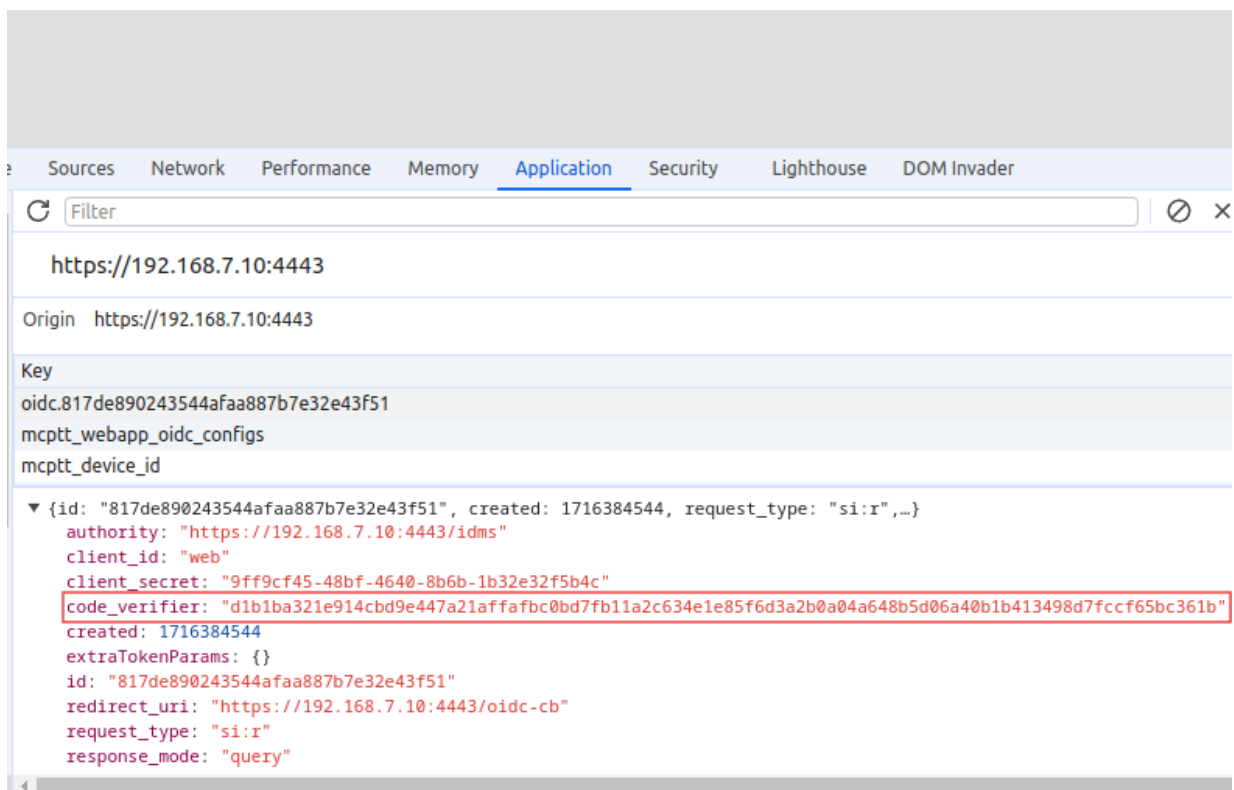


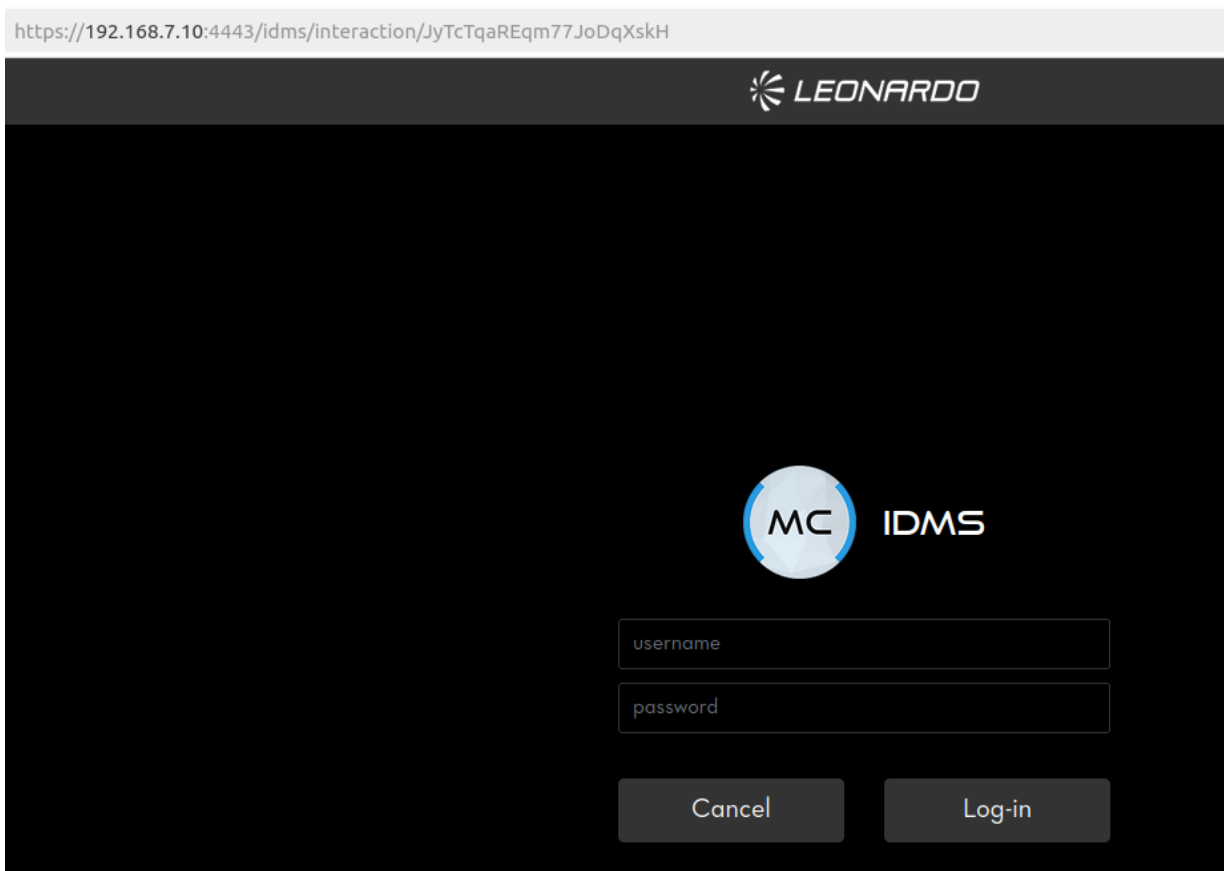
Figure 6.7: Tampered authentication request

Before proceeding with step 2, the attacker must collect the `code_verifier`. To do so exploits the fact that some information is stored in an accessible location, like the **Browser's LocalStorage**, as explained in Section 5.4.1.1. Therefore, the attacker grabs the value of the `code_verifier` and saves it for the last step. Figure 6.8 shows the LocalStorage with the generated `code_verifier`. The attacker then copies the URL of the tampered request and drops such request.



**Figure 6.8:** LocalStorage with code\_verifier

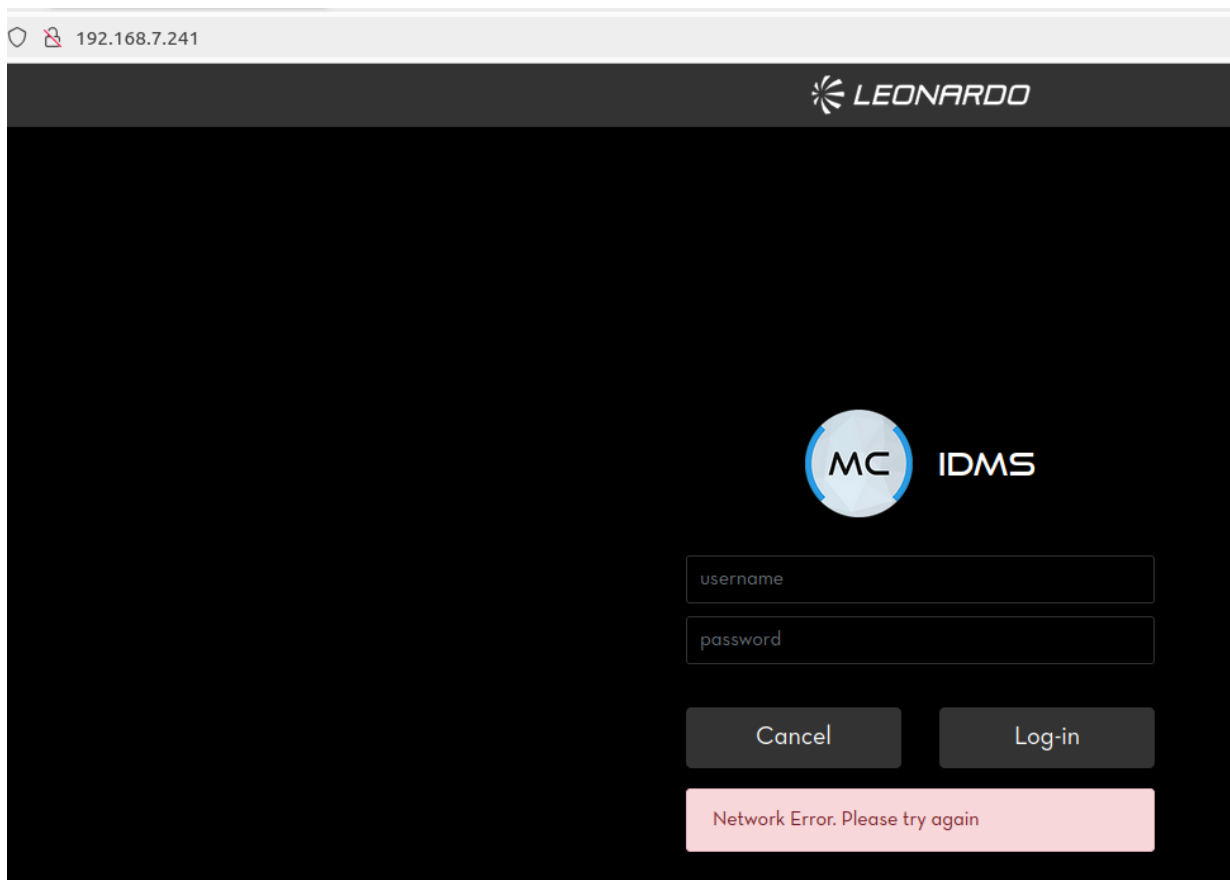
2. The request's URL with the poisoned parameters is delivered to the victim, for instance using some Social Engineering vector. Once, the victim opens the link, is prompted with the login page that asks for a username and password. The login form is showed in Figure 6.9



**Figure 6.9:** IDMS login form

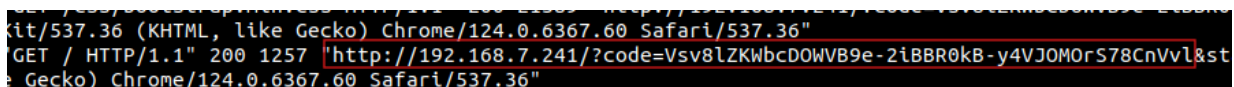
3. The victim provides a username and password and clicks the "Log-in" button to confirm.
4. The flow proceeds with the **authentication response**. During this step, the attacker is hosting a fake page resembling the original application, that displays a generic error message, saying that the login was not successful. The IDMS redirects the victim to the attacker's page due to the poisoned `redirect_uri`, to which the `authorization_code` has been sent. To host the fake page and collect the `authorization_code`, it was used the **Apache2 HTTP Server** tool described in Section 4.3.3. The attacker can easily recover the code by observing the request parameters from the Apache2 access-log file. Figure 6.10 shows the fake page with

a generic error *"Network Error. Please try again."*, that does not arouse particular suspicion and will force the victim to go back to the initial page. It should be noted that Figure 6.9 and Figure 6.10 seem equal but differ on the IP address, indeed the first one has IP **192.168.7.10**, which is the MC VM, while the latter has IP **192.168.7.241**, which corresponds to the attacker's machine.



**Figure 6.10:** Fake page with Network Error

Figure 6.11 shows the request with the `authorization_code` sent from the IDMS to the attacker's server and logged in the Apache2 access-log file.



**Figure 6.11:** Request with `authorization_code` sent to attacker's server



# Chapter 7

## Discussion

In this work, we delved into the 3GPP standards for Mission Critical (MC) communication systems, which are very extensive and detailed. The main focus was identifying the key security aspects, priorities, and technologies that need attention in these systems. The descriptions provided in Section 2.2 and Section 2.3 were not considered as a contribution but can still be useful as they consolidate information from various 3GPP documents into a single chapter. Since these documents are not easy to piece together without prior familiarity, this work can help accelerate the learning process for future students who wish to delve into the field of Mission Critical systems. Next, we provided the architecture of a fully functional testbed, set up to conduct the exploration of our MC system, which due to its modularity, can be used to perform tests on other MC applications by simply changing the **SUT** highlighted in Figure 4.1. One of the core contributions of this work is the development of **threat scenarios** based on insights from both 3GPP technical documents and practical experience gained while testing a real-world enterprise MC system. Additionally, we provided a **testing guide** that aims to give visual examples of how some of the proposed scenarios, previously described by words, can be evaluated using the tools listed in Section 4.3.3. The exploration process yielded two potential vulnerabilities in the *3GPP TS 33.180 - Security of the Mission Critical service* document, that may arise due to omissions on security aspects related to OpenID Connect, implemented in the IDMS, and involved during the MCX User Authentication procedure.

Particularly, we identified that updates on the **Client registration** process and the **PKCE extension**, mandatorily required for MCX clients fitting the Native application profile, would provide completeness to the TS document and would help developers avoid common misconfiguration that can harm users in the system. The motivations that led us to this outcome are described below.

**Client Registration** - The client registration described in TS 33.180 [9] - Annex B.3 quotes the following sentence: "Before a client can obtain ID tokens and access tokens (required to access MCX resource servers) it shall first be registered with the IdM server of the service provider as required by OpenID Connect 1.0. **The method by which this is done is not specified by this profile**". Since a new client can be registered via the client registration endpoint as explained in Section 2.4.6, there is the possibility that the IDMS is vulnerable to the **Unprotected Dynamic Client Registration**, explained in this work in Section 5.4.1, which could be abused to register new client applications with malicious purposes. For this reason, we think that it would be worth mentioning this type of vulnerability in the TS document and providing guidelines on either disabling the registration endpoint or enforcing the presence of an access token in the client registration request.

**PKCE extension** - In Section 2.4.4 it was outlined that MCX Native Clients must utilize the authorization code grant type with the PKCE extension for enhanced security. Therefore, as presented in Table 2.1, the `code_challenge` parameter is mandatorily required in the authentication request. We think that it would be worth mentioning the possibility of the **PKCE Downgrade Attack**, introduced in the 16th version of the *OAuth 2.0 Security Best Current Practice* and described in our work in Section 5.4.1.1, as it could help developers to ensure that if there was no `code_challenge` in the authorization request, the final request to the token endpoint must be rejected, even if it contains the `code_verifier`.

# Chapter 8

## Conclusion

This document provides an introduction regarding the evolution of Mission Critical communications and an overview of various use cases where they are implied. The introduction includes also the motivation and the contributions brought in this work. Next, we provided the necessary background to understand the main components and points of contact that make up the functional architecture of this type of system, as well as standard procedures and flows involved in the lifecycle of client/server interactions. The background also describes the principal key points of the STRIDE threat model, used to map threat scenarios identified in the analysis. The proposed work presented the testbed, used in our laboratory, which allowed us to study a real implementation of a Mission Critical system and conduct tests on it. We also highlighted the value of such a testbed, which can be used by other vendors who want to perform tests on their system and may need help with a possible starting architecture or necessary security tools. In the last two chapters, we presented the main contribution of this work, which gives an initial overview, a subsequent detailed description, and a testing guide of some threat scenarios based on a deep study of 3GPP Technical documents, combined with a practical exploration of an enterprise Mission Critical system. These scenarios target components and functionalities that will be implemented or very likely to be implemented in every application, therefore being useful for other vendors that want to test their systems. We also revealed that some aspects regarding the main document for the security of the Mission Critical service could be enhanced, avoiding potential misconfiguration introduced by the standard's lack of detail.



## 8.1 Future Works

Currently, the Mission Critical domain is still relatively unexplored, particularly in the security context, leaving a considerable scope for future research. Due to the complexity of this type of application, interactions between two Mission-Critical systems, which may occur within the same domain or in a remote domain, were not examined in this study. The present study did not examine interactions over the N5 interface, between the Mission-Critical application and the PCF Network Function situated within the 5G core network. It would be necessary to explore much more in-depth the potential threats introduced by a stolen device, which in this work has been approximately analyzed due to time constraints. Additionally, the 3GPP standard occasionally indicates that confidentiality protection is not required for specific traffic when the underlying access network provides it, therefore it would be worthwhile to investigate scenarios involving False Base Stations. An example of such a statement can be found in ETSI [10], Chapter 7.2, NOTE 5, which states, "The P-CSCF may be configured to never apply confidentiality, e.g., because it trusts the encryption provided by the underlying access network". Since it is widely understood that encryption requires computation, which can lead to slower processing speeds, it would be interesting to conduct research on a system with a high user volume to identify the types of protection that are likely to be disabled due to the overhead they introduce.

# Bibliography

- [1] et. al. Rosenberg. *SIP: Session Initiation Protocol*. Tech. rep. [Accessed: 27-05-2024]. IETF, 2002. URL: <https://datatracker.ietf.org/doc/html/rfc3261>.
- [2] Wallgren E. and Willander C. (2022). “SIPman: A penetration testing methodology for SIP and RTP”. [Accessed: 08-06-2024]. MA thesis. Blekinge Institute of Technology, 2022. URL: <https://bth.diva-portal.org/smash/get/diva2:1661144/FULLTEXT01.pdf>.
- [3] TCCA’s Critical Communications Broadband Group. *Mission Critical Broadband Applications*. <https://tcca.info/documents/April-2022-MC-Broadband-Applications.pdf> [Accessed: 31-05-2024]. 2022.
- [4] Carlo Maria Plazzotta. “Mission Critical Services: Evolution of QoS Management Over 5G Networks”. [Accessed: 10-03-2024]. MA thesis. University of Padova, 2022. URL: [https://thesis.unipd.it/bitstream/20.500.12608/36031/1/Plazzotta\\_Carlo\\_Maria.pdf](https://thesis.unipd.it/bitstream/20.500.12608/36031/1/Plazzotta_Carlo_Maria.pdf).
- [5] IP Access International. *Mission Critical Communications Explained — IP Access International*. <https://www.ipinternational.net/mission-critical-communications-explained/> [Accessed: 08-06-2024]. 2023.
- [6] OAuth Working Group. *OAuth 2.0 Security Best Current Practice*. Tech. rep. [Accessed: 10-06-2024]. IETF, 2024. URL: <https://datatracker.ietf.org/doc/html/draft-ietf-oauth-security-topics>.
- [7] 3rd Generation Partnership Project (3GPP). *3GPP TS 22.281 - Mission Critical Video Services*. [Accessed: 29-07-2024]. 3GPP. URL: [https://www.3gpp.org/ftp/Specs/archive/22\\_series/22.281/](https://www.3gpp.org/ftp/Specs/archive/22_series/22.281/).
- [8] 3rd Generation Partnership Project (3GPP). *3GPP TS 22.282 - Mission Critical Data Services*. [Accessed: 29-07-2024]. 3GPP. URL: [https://www.3gpp.org/ftp/Specs/archive/22\\_series/22.281/](https://www.3gpp.org/ftp/Specs/archive/22_series/22.281/).
- [9] ETSI. “*3GPP TS 33.180 - Security of the Mission Critical (MC) service v17.9.0*”. [Accessed: 17-06-2024]. URL: [https://www.etsi.org/deliver/etsi\\_TS/133100\\_133199/133180/17.09.00\\_60/ts\\_133180v170900p.pdf](https://www.etsi.org/deliver/etsi_TS/133100_133199/133180/17.09.00_60/ts_133180v170900p.pdf).

- [10] ETSI. *3GPP TS 33.203 - Access security for IP-based services*. [Accessed: 20-06-2024]. URL: [https://www.etsi.org/deliver/etsi\\_ts/133200\\_133299/133203/17.01.00\\_60/ts\\_133203v170100p.pdf](https://www.etsi.org/deliver/etsi_ts/133200_133299/133203/17.01.00_60/ts_133203v170100p.pdf).
- [11] ETSI. *3GPP TS 33.210 - Network Domain Security (NDS)*. [Accessed: 23-06-2024]. URL: [https://www.etsi.org/deliver/etsi\\_ts/133200\\_133299/133210/17.01.00\\_60/ts\\_133210v170100p.pdf](https://www.etsi.org/deliver/etsi_ts/133200_133299/133210/17.01.00_60/ts_133210v170100p.pdf).
- [12] ETSI. *ETSI TS 123 280; Common functional architecture to support mission critical services; Stage 2*. [Accessed: 22-07-2024]. ETSI. URL: [https://www.etsi.org/deliver/etsi\\_ts/123200\\_123299/123280/18.10.00\\_60/ts\\_123280v181000p.pdf](https://www.etsi.org/deliver/etsi_ts/123200_123299/123280/18.10.00_60/ts_123280v181000p.pdf).
- [13] ETSI. *ETSI TS 123 289 - Mission Critical services over 5G System; Stage 2*. [Accessed: 22-07-2024]. ETSI. URL: [https://www.etsi.org/deliver/etsi\\_ts/123200\\_123299/123289/18.09.00\\_60/ts\\_123289v180900p.pdf](https://www.etsi.org/deliver/etsi_ts/123200_123299/123289/18.09.00_60/ts_123289v180900p.pdf).
- [14] ETSI. *ETSI TS 124 483 - Mission Critical Services (MCS) Management Object (MO)*. [Accessed: 31-07-2024]. ETSI. URL: [https://www.etsi.org/deliver/etsi\\_ts/124400\\_124499/124483/18.03.00\\_60/ts\\_124483v180300p.pdf](https://www.etsi.org/deliver/etsi_ts/124400_124499/124483/18.03.00_60/ts_124483v180300p.pdf).
- [15] ETSI. *ETSI TS 124 484 - Mission Critical Services (MCS) configuration management; Protocol specification*. [Accessed: 29-07-2024]. ETSI. URL: [https://www.etsi.org/deliver/etsi\\_ts/124400\\_124499/124484/18.06.00\\_60/ts\\_124484v180600p.pdf](https://www.etsi.org/deliver/etsi_ts/124400_124499/124484/18.06.00_60/ts_124484v180600p.pdf).
- [16] Huawei. *Dividing a Network into Security Zones*. [Accessed: 22-05-2024]. URL: <https://support.huawei.com/enterprise/en/doc/EDOC1100172313/b0cfa445/dividing-a-network-into-security-zones>.
- [17] IETF. *OAuth 2.0*. [Accessed: 14-05-2024]. URL: <https://oauth.net/2/>.
- [18] Microsoft. *The STRIDE Threat Model*. [Accessed: 24-04-2024]. URL: [https://learn.microsoft.com/en-us/previous-versions/commerce-server/ee823878\(v=cs.20\)](https://learn.microsoft.com/en-us/previous-versions/commerce-server/ee823878(v=cs.20)).
- [19] Open ID Foundation. *OpenID Connect Dynamic Client Registration 1.0*. [Accessed: 14-05-2024]. URL: [https://openid.net/specs/openid-connect-registration-1\\_0.html](https://openid.net/specs/openid-connect-registration-1_0.html).
- [20] OWASP. *Threat Modeling Process*. [Accessed: 25-04-2024]. URL: [https://owasp.org/www-community/Threat\\_Modeling\\_Process](https://owasp.org/www-community/Threat_Modeling_Process).
- [21] OWASP. *Web Security Testing Guide*. [Accessed: 30-04-2024]. URL: <https://owasp.org/www-project-web-security-testing-guide/latest/>.
- [22] Software-defined Radio. *Software-defined Radio*. [Accessed: 08-07-2024]. URL: [https://en.wikipedia.org/wiki/Software-defined\\_radio](https://en.wikipedia.org/wiki/Software-defined_radio).
- [23] Techslang. *What is a Security Zone*. [Accessed: 22-05-2024]. URL: <https://www.techslang.com/definition/what-is-a-security-zone/>.

- [24] VMWare. *What is Threat Analysis*. [Accessed: 18-04-2024]. URL: <https://www.vmware.com/topics/glossary/content/threat-analysis.html>.