ISITEP
D5.3.2 – SECURITY MANAGER DESIGN DESCRIPTION

Document Manager:	Marco Carli	RM3	Editor

Programme:	Inter System Interoperability for Tetra-TetraPol Networks
Project Acronym:	ISITEP
Contract Number:	312484
Project Coordinator:	Selex ES
SP Leader:	RM3

Document ID N°:	ISITEP_D5.3.2_20151021_V1.0	**Version:**	V1.0
Deliverable:	D5.3.2	**Date:**	21/10/2015
		Status:	Approved

Document classification	**PUblic**

Approval Status	
Prepared by:	Marco Carli (RM3)
Approved by (WP Leader):	Alessandro Semproni (RM3)
Approved by (SP Leader):	Federica Battisti (RM3)
Approved by (Coordinator)	Paolo Di Michele (SES)
Security Approval (Advisory Board Coordinator)	Etienne Lezaack (BFP)

CONTRIBUTING PARTNERS

Name	Company / Organization	Role / Title
Marco Carli	RM3	Contributor
Federica Battisti	RM3	Contributor
Federico Frosali	SES	Revisor
Claudia Olivieri	SES	Revisor

DISTRIBUTION LIST

Name	Company / Organization	Role / Title
Federico.Frosali, Claudia.Olivieri, Andrea Campodonico	SES	WP2.2 participant
Etienne Lezaack, Simon Verdegem, Yves Cawet, Marc Vandenbroeck, Marie Carlsson	BFP	WP2.2 participant
Marianne Storrosten, Michel Duits	DNK	WP2.2 participant
Anita Galin, Anna Falkdrugge, Peter.Hedman, Robert Danelius, David Arnljots	MSB	WP2.2 participant
Ronald Van.Der Wal, Herman Van Sprakelaar, Hans Borgonjen	V & J	WP2.2 participant
Jaakko Saijonmaa, Risto Toikkanen	CAS FI	WP2.2 participant
Serge Delmas, Jean-Pierre Quemard, Herve Mokrani, Eric Lorfeuvre, Dominique Eustache	CAS FR	WP2.2 participant
Daniele Biondini, Luciana Favia, Ivano Luciani, Giuseppe Pierri, Franco Pangallo, Mario Manzi	ISCOM	WP2.2 participant

Theodore Tzamos, Michael Spyridakis, Haritou, Dimitris Androutsopoulos	NETTECHN	WP2.2 participant
Cor Verkoelen, Frank Fransen, Bram Verheesen, Marcel Vanderlee	TNO	WP2.2 participant
Ramon Ferrús, Oriol Sallent	UPC	WP2.2 participant/Leader
All Company Project Managers	All involved companies	Members of the Steering Committee
Elina MANOVA	EC DG REA	EC Programme Officer
General Public	NA	NA

REVISION TABLE

Version	Date	Modified Pages	Modified Sections	Comments
V0.1	13/07/2015	All	All	Initial version (RM3)
V0.2	07/10/2015	All	All	Revised version (RM3)
V0.3	08/10/2015		4.x	Security architecture, security functionalities (RM3)
V0.4	08/10/2015	All	All	Revision (SES, RM3)
V1.0	21/10/2015	All	All	Final release

Publishable extended abstract

This deliverable provides the definition and design of the Security Manager (SM) for the new enhanced terminals designed and implemented in the ISITEP project. The possibility to grant a secure interoperability among TETRA and TETRAPOL network is of crucial importance. In this framework, the overall security depends on several factors, such as the secure migration between TETRA and TETRAPOL networks and the use of a secured terminal. In particular, the use of an open source Operating System (i.e., Android) poses severe challenges concerning threats and vulnerabilities that are addressed in this document.

The security relies both on the authentication mechanisms implemented in TETRA networks and the available authentication functionalities implemented in TETRAPOL networks and on the application level security that will be granted by hardening of the Android platform and security protocols implemented through the SM.

This WP is devoted to define Security Software Manager on ISITEP terminals addressing the following issues:

- Securing the enhanced terminal by considering the physical deployments and the exposed interfaces.
- Securing the enhanced terminal at the application level by counteracting all possible threats caused by the use of the Android OS.
- Defining security protocol for exploiting the enhanced terminal capabilities, including communication and added-value application services.
- Exporting to the end-user the security level in use by the TETRA and by the TETRAPOL modems (e.g. security class).

CONTENTS

1 INTRODUCTION

The aim of WP5.3 is to define and implement a Security Software manager on the new enhanced terminal designed and implemented in the ISITEP project.

The enhanced terminal is a bi-technology terminal composed by a TETRA and by a TETRAPOL modem controlled by a single control unit deployed inside an Android device.

TETRA and TETRAPOL modems have embedded security functions that provide: authentication of the modem on the relevant Network Infrastructure and confidentiality of communications through encryption mechanisms. TETRA and TETRAPOL technologies provides also Enable / Disable capability that ensure against eavesdropping or poisoning information from a stolen or loss terminal.

The Android device is one of the innovations introduced by the ISITEP project; it provides an open and programmable platform where TETRA and TETRAPOL technologies are integrated and where interoperability enabling applications and PPDR added value applications are deployed.

Using an open source Operating System (i.e., Android) poses severe challenges concerning threats and vulnerabilities that will be addressed in this deliverable.

The outcomes of this deliverable will be used by User Interface and Business Logic Manager (WP 5.6). It will also be used for terminal integration and testing (WP 5.7) and for terminal and services validation and qualification (WP 5.8).

The document is organized as follows:

- Section 1: Introduction
- Section 2: Scope of the document
- Section 3: Definition and abbreviation
- Section 4: “Enhanced terminal security threat analysis”. This chapter analyses the security threats discussed in D2.2.2 for the PPDR terminals, by considering the implementation of the enhanced terminal solution.
- Section 5: “Android Security Framework”. This chapter analyses the security issues in the Android Operative System (O.S.) security threats of Android O.S., and the possible solutions to face the security gaps of the Android O.S.
- Section 6: “Requirements” contains the requirements applied to the SM software components.
- Section 7: “Design description” contains the design description of the SM, including a description of its functionalities and interfaces.
- Section 8: “Software design” covers the software design of the SM components, the state machine diagram and detailed sequence diagrams describing these functionalities.
- Section 9: “Requirement traceability” contains the security requirements traceability in the Security Manager Design Description.
- Section 10: Conclusions
- Section 11: References

2 DOCUMENT SCOPE

The aim of D5.3.2 is the definition of the design of the Security Manager. This tool aims at providing a secure framework for the enhanced terminal and ISITEP communications.

WP5.3 exploits the outcomes of WP2.2 and in particular, of D2.2.2 - Candidate Security Requirements in which the basics of security at network level, gateway level and at terminal level have been addressed. The network related security threats have been analyzed in D2.2.2, resulting in the definition of security requirements. This deliverable analyzes the security threats concerning the terminal and it addresses the overall solution of the enhanced terminal.

The security aspects addressed in this deliverable deviate from what was originally described in the Description of Work. This is due to the fact that, the security manager does not address or configure the security of TETRA and TETRAPOL, since TETRA and TETRAPOL modems have embedded security functions and the manipulation of sensible data, like authentication and encryption keys, is not exported by the modems for security reasons.

This deliverable addresses:

- the security of the overall solution of the enhanced terminal with respect to the Security Requirements expressed in D2.2.2;
- the security of the Android O.S..

.

3 DEFINITIONS AND ABBREVIATIONS

3.1 Definitions

This section is intended to capture the definitions of some key terms used in the document for the purpose of increased consistency. Most of the definitions are obtained from official 3GPP and ETSI documents:

Access control: the prevention of unauthorized use of resources, including the use of a resource in an unauthorized manner.

Authentication: the act of positively verifying that the identity of an entity (network, user) is the same as the claimed identity.

Confidentiality: the property that information may not be available or disclosed to unauthorized individuals, entities or processes.

Data integrity: the property that data has not been altered or destroyed in an unauthorized manner.

Encryption: the conversion of plaintext to cipher text.

Key: a sequence of symbols that controls the operations of ciphering and deciphering.

Key management: the generation, selection, storage, distribution, deletion, archiving and application of keys in accordance with a security policy.

Migration: act of changing to a location area in another network (either with different Mobile Network Code and/or Mobile Country Code) where the user does not have subscription (e.g. ITSI in TETRA) for that network. In this document, migration is used as a synonym of roaming.

Plaintext: information (including data) which is intelligible to all entities.

Roaming: utilization of a mobile terminal in a network other than the one where the mobile is subscribed but on which the mobile can still be located and operated by agreement between the respective network operators.

Security domain: a set of entities and parties that are subject to a single security policy and a single security administration. The network security design can consider different domains and sub-domains to surround and delimit the responsibilities in network management and security control.

Security service: a service provided by a layer of communicating open systems which ensures adequate security of the systems or of data transfers.

Security threat: a security threat is defined as a potential violation of security. Examples of security threats are loss or disclosure of information or modification/destruction of assets. A security threat can be intentional like a deliberate attack or unintentional due to an internal failure or malfunctions.

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

Acronym	Definition
AI	Air Interface
AIE	Air Interface Encryption
DAC	Discretionary Access control

IP	Internet Protocol
IPC	Inter-Process Communication
ISI	Inter System Interface
MAC	Mandatory Access Control
OS	Operating System
PC	Professional Computer
PPDR	Public Protection and Disaster Relief
PS	Public Safety
SDS	Short Data Service
SELinux	Security Enhanced Linux
SwMI	Switching and Management Infrastructure
VPN	Virtual Private Network

4 ENHANCED TERMINAL SECURITY THREAT ANALISYS

According to the discussion reported in the D2.2.2 [2], Security Requirements, the ISITEP Enhanced Terminal may be prone to the following security threats:

- threats from mismatching security requirements between roaming terminals and visited networks, this is common to all PPDR terminals;
- threats from the exposed interfaces;
- threats on the use of enable and disable functionalities.

In addition to threats provided in ISITEP D2.2.2, in this document will be discussed also threats introduced by the use of the Android device. Android O.S. has been designed with a strong focus on security; despite this, the Android platform is prone to different attacks. In the following, it will be described Android security framework, the main vulnerabilities of the system and possible solution to cover these security gaps.

In the following paragraphs will be discussed requirements applicable to the enhanced terminal to face the security threats discussed in ISITEP D2.2.2 [2].

4.1 Security threats from mismatching security requirements between roaming terminals and visited networks

Roaming PMR terminals are usually certified to the security requirements of the home country, while they may not be compliant with the requirements of the visited country. Security certification processes are established at national level and may be quite in some countries. It is therefore a difficult task the certification of terminals at European level.

In order to face this security gap the enhanced terminal should be able at least to export toward the end-user the security status of the TETRA and TETRAPOL modems: the end-user shall be aware if his communications are encrypted or not.

4.2 Security threats from exposed interfaces

The Android device used in the enhanced terminal is provided with the following exposed interfaces:

- USB
- WiFi
- BlueTooth

In *normal* condition, the USB interface is used to connect the control unit, deployed on the Android device, with the TETRAPOL modem. In *administration* condition when an administrator operator needs to upgrade installed SW or to modify the terminal configuration the USB interface is used for transfering data.

WiFi interface is used only in the vehicular deployment to connect the control unit deployed on the Android device with the TETRA modem. (In the Hand-Held solution the TETRA modem is embedded in the Android device).

Bluetooth interface is not used.

The security threats discussed in ISITEP D2.2.2 [2] are:

- *Unauthorized access to TETRA or TETRAPOL modem [**T8.1**]*

- *Eavesdropping of conversations related to the attacked MS [**T8.2**]*

These are two typical threats faced by PPDR telecommunication systems, both TETRA and TETRAPOL radios are projected in order to cover such security gaps, but the overall solution of the enhanced terminal is exposed to such threats.

Moreover it shall be considered also that physical interfaces represent an access to the android device; therefore a hacker or an intruder may access also to the data stored in the android device (i.e. saved messages, photo or videos), or it may gain total control of the terminal.

Therefore only controlled access shall be provided through these interfaces or they shall be inhibited.

USB on the enhanced terminal is an OTG port, it can be used both as "device" port and as “host” port. In normal condition when the TETRAPOL device is connected this port is used in “host” mode, while in administration condition this port is used in “device” mode. Device mode should be disabled when the enhanced terminal is used by the end-user.

WiFi interface will be enabled only as access point and WPA mechanism will be activated in order to encrypt WiFi link and allow only authorized accesses.

Bluetooth interface is not used in the ISITEP enhanced terminal therefore it will be inhibited at kernel level.

4.3 Threats on the use of enable and disable functionality

TETRA and TETRAPOL technologies provide enable/disable capability. A typical use of this feature could be to bar a lost or stolen MS from full access to the system, whilst still allowing registrations, which identify the cell location of the MS.

Current ISI implementation does not provide Enable / Disable across ISI link, Enable/Disable is foreseen for phase 4. If a MS migrated in a visited network is stolen it is not possible to disable it in the visited network. The occurrence is estimated as possible

The security threats discussed in ISITEP D2.2.2 [2] is:

- *Continued use of stolen or lost trusted terminal in visited network [**T3.1**]*

In order to face this security gap at the start-up the user should be required to make an input (e.g. a personal identification number (PIN)) before activate TETRA and TETRAPOL service.

5 ANDROID SECURITY FRAMEWORK

5.1 Android OS elements

The Android OS structure is shown in Figure 1.

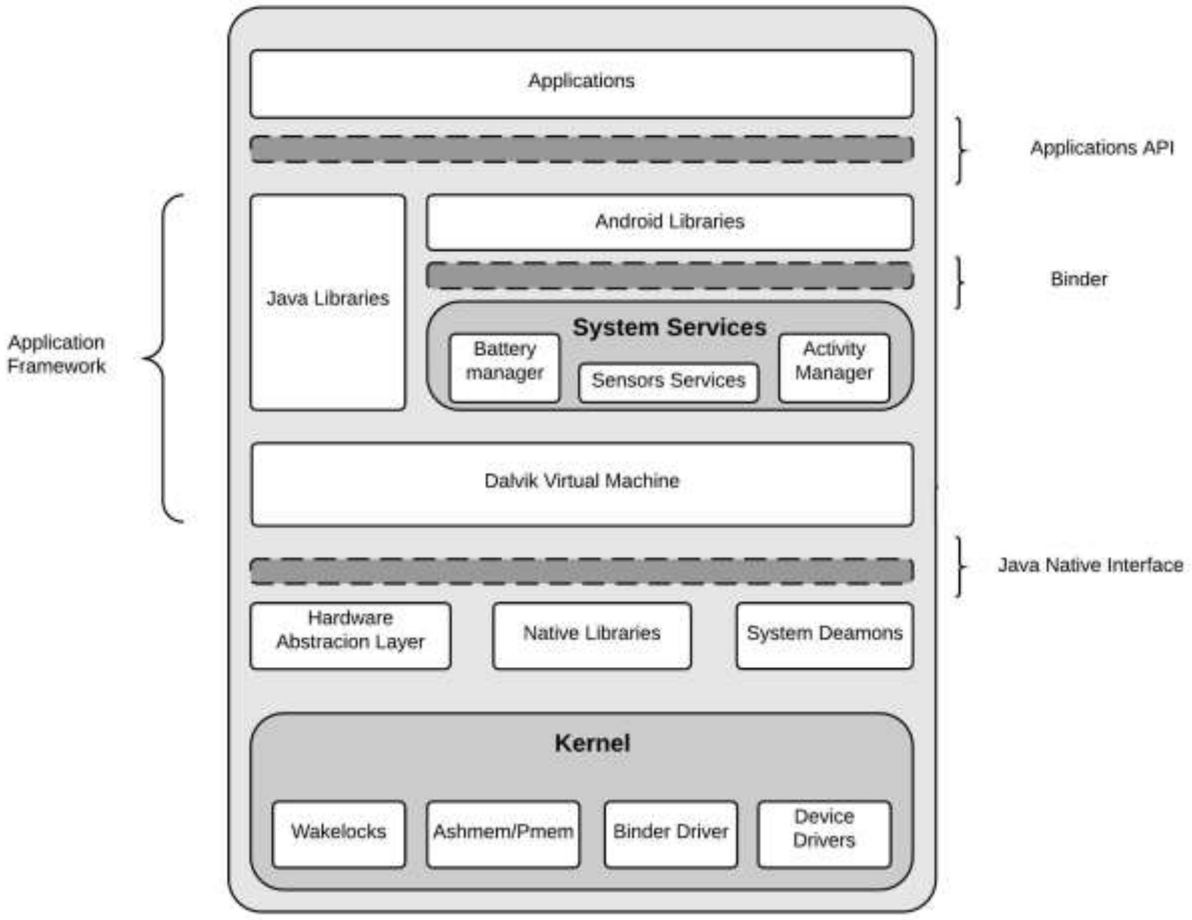

Figure 1 Block diagram of the Android Security Framework

The Android kernel is a modified version of the classic Linux kernel. In this module hardware drivers and new elements for enabling the terminal mobility are integrated. In particular, since the mobile environment is characterized by limited system memory and battery power, the kernel contains mechanisms for memory handling (i.e., for sharing memory between processes, or for killing processes as soon as available memory undergoes a threshold) and for limiting battery power consumption.

In more details, Android OS is designed to go into sleep mode as soon as it detects no user activity. Since some applications (e.g. GPS) require the device to stay on even if there is no user interaction for a long time, an ad hoc mechanism (i.e. *wakelock*) is provided for keeping the device active.

A key feature for security is IPC (Inter-Process Communication). It describes the modalities used for Android components interaction:

- *intents*: messages that each component can send and receive. It is used for sharing data between processes, for starting services or activities, for invoking broadcast receivers and so on;
- *bundles*: are abstract entities used to exchange data between applications;
- *binders*: they allow activities and services to obtain a reference to another service.

Most of IPCs in Android is handled by the Binder kernel module. Higher level IPC mechanisms, such as intents, also rely on the Binder. The Binder implements a proxy-like communication protocol, in which each subject requests the needed IPC operation and undergoes a security check according to the implemented permission system. The permission system will be discussed in Section 0.

An important difference between the lower and the upper layer is that the kernel is implemented in C/C++ while the components above the Dalvik virtual machine (see Figure 1) are implemented in Java.

The security of systems is strictly dependent on the installed applications; however, in mobile systems, the applications have unique structures that must be analyzed in order to understand how they affect the system security.

An Android application is a collection of abstract objects attaining different functionalities. The relevant components for this security analysis are:

- *activities*: they represent the main interaction facility. Usually, every screen that an application can display correspond to an activity;
- *intents*: they are the main IPC facility at application level. They declare the task that must be performed and can be unicast or broadcast, depending on the task. Intents are also used to start activities and perform intra-application tasks;
- *application manifest*: it is an XML file that contains a description of the application, including a declaration and description of every components that it uses:
- *services*: they are used to perform long running tasks or to grant a capability to other elements of the system.

5.2 Android security principles

The Android OS is based on the isolation security principle.

Smartphones and tablets contain information that must be protected, such as the user identity and billing information, together with a variety of entertaining applications, collected often from unverified sources. In such a context a strong isolation mechanism is needed.

Android is based on a Linux kernel and it supports modified versions of the classical Linux security mechanisms.

Traditionally, Linux makes use of a discretionary access control model, based on permissions. Each user, identified by a user ID, is the owner of his resources and can therefore grant other privileges on the aforementioned items. Furthermore, the user can be member of groups used for mapping organization layers in the system access control mechanism. These groups are identified by means of a group ID.

In Android the concept of user and groups is leveraged to enforce isolation. A user ID is assigned to each application in the system. In this way, it is possible to deal with applications as if they are users, denying access to files and resources that are not within its authorization level. Applications are further isolated at process and file system level. Interactions with external elements are handled through the permission system. Permissions in Android represent the capability to access high level functionalities of the device, such as Internet access or camera usage. The OS defines permissions for his own basic hardware and capabilities; however, this set of permission can be modified and/or extended by manufacturers and application developers.

Permissions are organized in four classes, according to the security impact of the capability requested. The four classes are:

- Normal permissions that are considered harmless for the system and are therefore granted without further controls.
- Dangerous permissions require user approval to be granted, since they can potentially harm the system.
- Signature permissions are granted only if the requesting application has the same signature (i.e. the code is signed with the same public key) as the code declaring the permissions itself.
- SignatureOrSystem permissions behave similarly to signature permissions: the system grants this permission only to applications that are in the Android system image or that are signed with the same certificate as the application that declared the permission. The "signatureOrSystem" permission is used for certain special situations where multiple vendors have applications built into a system image and need to share specific features explicitly because they are being built together.

Permissions must be accepted when the application is installed and, later, it is not possible to deny any of them without uninstalling the application. Moreover, all of the permission requested must be granted in order to succeed with the app installation.

To each permission it is associated a Linux group and a corresponding group ID. If a permission is successfully granted to an application, the associated user will become member of the group associated with that permission.

The permission system together with application isolation are the core of the Android *sandboxing*. Usually, the term sandboxing refers to the execution of potentially malicious software in a protected environment for preventing them to harm the underlying system. In Android, sandboxing refers to the set of measures granting process isolation.

The access control criteria defined through the permission system are enforced whenever normal inter-process communication take place. This procedure is handled by the Binder. The Binder handles inter-process communications through a proxy-like communication model: the binder verifies if the security criteria are satisfied for each transaction and delivers the operation request and the associated data.

Typically, in operating systems based on discretionary access control, there is a user that has the highest permission level thus being able to operate on every file that is the *root* user or *super-user* in Linux-based systems. The set of root-level permissions are not accessible to users in Android. This is due to the fact that acting as root may cause security flaws and allow unskilled users to perform harmful modifications to the system. The procedure that allows users to act as root is called *rooting* and it is generally discouraged by original equipment manufacturer.

The discretionary model for access control has proven to be solid in Linux distributions for many years. However, in high security contexts, such as the military one, DAC models do not provide the required level of security. A possible alternative access control model is MAC. In a MAC system, the security is defined through policies that users are unable to modify, even if those policies are applied to their own files. In a MAC system there is not necessarily a root user, while there might be several high privilege users. Android features MAC in the form of SELinux.

SELinux implements a role based access control by handling interactions between subject and objects through policies. Every entity in the system is labelled with a context that consists of username (not to be confused with the user ID), a role and a domain. Subjects (usually processes) can transition between domains and roles to gain different capabilities but only if this is allowed by the policy. This means that there is a one-to-many mapping from domains to roles and from roles to usernames. The absence of a super user in MAC systems enables a higher security level by means of a better role separation. In this way, it is possible to reinforce a traditional DAC system and to mitigate the impact of security vulnerabilities discovered in the system. SELinux can be in three different statuses:

- *Disabled*: SELinux is present but not enabled.
- *Permissive*: SELinux is not enforcing policies, but is logging anything that violates the policy specifications.
- *Enforcing*: SELinux enforces the policy and logs any attempted violation.n

In this way SELinux enables a precise auditing of the system, as it features advanced tools for log processing.

The Android implementation of SELinux is almost identical to the classical one. The main difference is that, in the versions preceedings the 5.0, SELinux only partially enforces security in the system. In particular:

- Android 4.2: SELinux is available in the kernel but disabled at compile time;
- Android 4.3: SELinux is available and compiled but set to permissive mode;
- Android 4.4: SELinux is set to enforcing mode, but it enforces policies only on four-core system domains;
- Android 5.0: SELinux is set to enforcing mode on all of the system domains.

The device administration framework can be useful to enforce security measures on the system, although it is not directly part of the system. The framework is composed by a set of API and programming facilities that allow the definition of system administrators, that are entities used to enforce high level policies to the user (in contrast to SELinux, that defines policies at system level).

A system administrator can be enabled and disabled by the user in the settings menu. While this may seem lackluster in terms of security, it is important to notice that the action of disabling an administrator triggers specific events such as the cancellation of sensitive data or even a factory reset (i.e. the wiping of the data partition). Besides enabling remote wipe of the device, a system administrator can also enforce the use of a password and security level of the aforementioned. Android 4.0 added the possibility to force camera disabling and it is arguable that subsequent updates will add additional policies. By not complying with the policies defined, the user incurs in restrictions on the use of the device decided by the programmer.

5.3 General Security Threats

Despite being designed with a strong focus on security, the Android platform is prone to different attacks. In the following sections, the main vulnerabilities of the system are discussed.

5.4 Threat modelling

Attack trees [14] can be used for modeling a threat on a system. Basically, a threat is represented as a high-level objective that the attacker wants to achieve, with the list of possible modalities. The attacks are then decomposed on a lower hierarchical level, estimating conditions and situations needed for the attack to be carried out. For each logical step, it is possible to insert a line between two blocks to indicate a countermeasure, implemented to hinder the attack.

One of the advantages of this threat modelling method is the possibility of making the security analysis clear and easy to understand, therefore reducing the chances of misunderstanding and the presence of “blind spots” in the system security configurations. Although more complex security models have been developed in literature for better representing the dynamics of complex attacks, the adopted model is more suitable to the high level description of the terminal security.

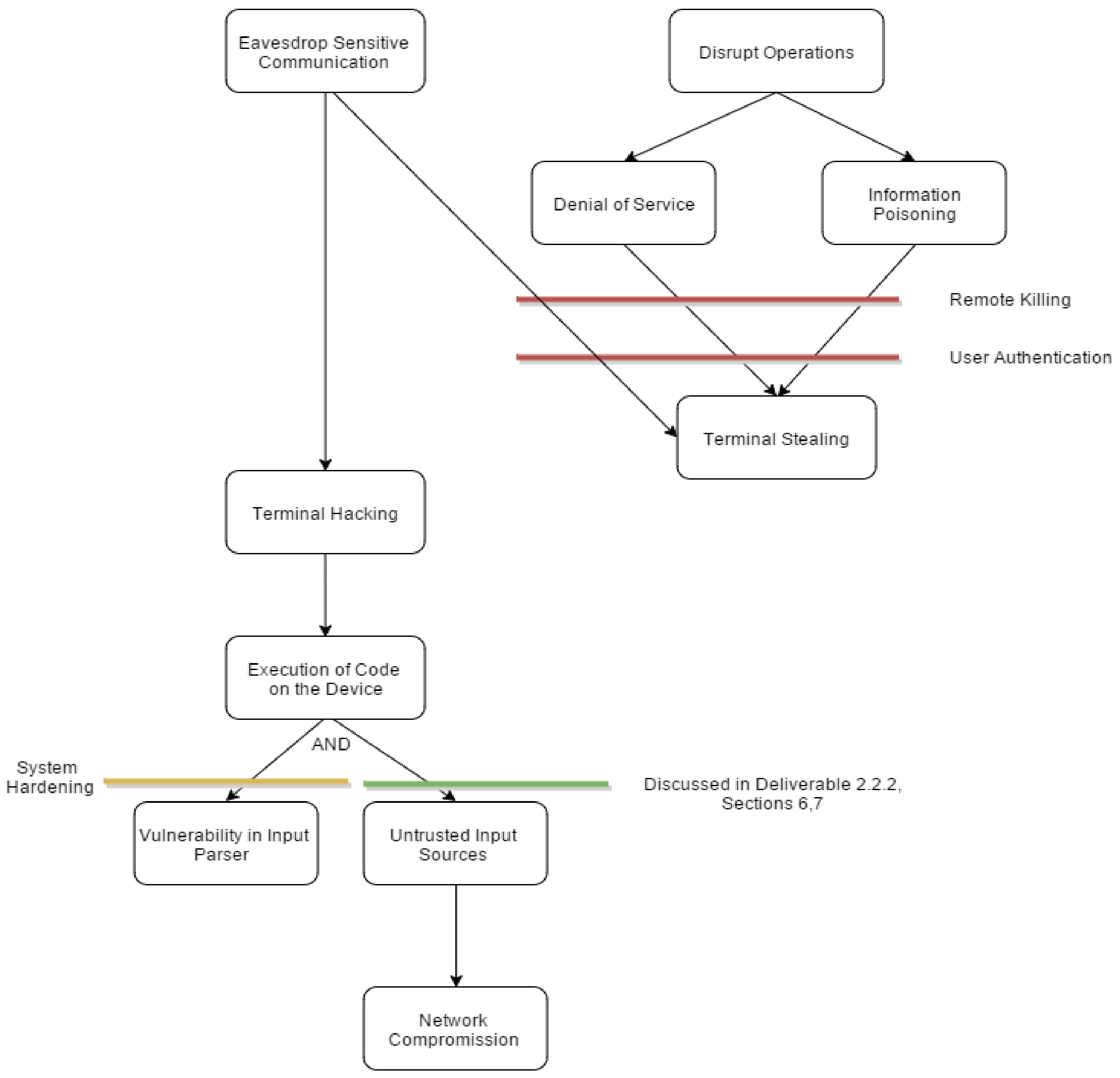

Figure 2 Attack trees threat modelling

In Figure 2, the attack tree model is shown. As already stated in this deliverable we extend the security analysis detailed in D2.2.2; in particular, the aspects related to the terminal security are analyzed.

In the ISITEP framework, an adversary will probably pursue attacks different than in traditional mobile systems. In fact, dealing with commercial devices is fruitful for an attacker since a set of attacks are available, resulting in high gain for the attacker thanks to the widespread of the technology and the low background in security of the typical end user. Example of such attacks are information stealing and subscription to costly service benefitting the attacker. The context of PPDR is quite different: devices are highly controlled and secured and they typically use only few, secure communications services, thus reducing potential point of contacts with an adversary.

Adversaries willing to attack such contexts are typically very motivated, given the value of the information and the existing security mechanisms.

In this context, an attacker is typically interested in hindering PPDR forces operations. Disruption of operations can be achieved by thwarting the communication, thus realizing a denial of service, by poisoning the information exchange with false information, or by eavesdropping data regarding operations and exploiting them for malicious purposes.

If the possibility of leveraging vulnerabilities of the network, such as session keys eavesdropping, is very low the attacker needs to compromise a device in order to achieve his objective. That is, if the terminal is successfully remotely hacked or if it is stolen. Nevertheless, in order to carry out a successful hacking, the attacker would need to establish at least a one-time contact with the terminal to deliver the malicious code and a continuous channel to gather the leaked data. Therefore, if the network is secure, the possibility of this type of attack is very low.

The other option relies on the terminal stealing. Terminal stealing or loss can have severe consequences as the attacker would have a way to access the private network. This eventuality is particularly dangerous in the case of an attacker eavesdropping the channel or poisoning the information. In case of denial of service, as a matter of fact, it is usually possible to trace the source of the disturbance back to the terminal and blacklisting it, restoring normal communications.

5.5 Hardening procedures

Hardening the Android platform is a process that shares similarities with traditional operating system hardening, but also exhibits unique properties due to the mobile nature of the system. At the time being, a limited number of resources have been published, regarding security best practice more than device hardening.

Given the absence of standardizations and official recommendations, the best procedure is to implement a hardening procedure following principles that have been proven successful in traditional operating systems security. The procedure discussed hereafter will be treated as if the context is known but the exact specifications and use cases are not specified. Further hardening of the system, carried on with respect to the use cases, will be treated in the following sections.

5.6 Platform Hardening

As stated before, Android is developed as a mobile OS focusing on enhancing user experience by exploiting data coming from multiple sources. In order to use Android as a secure OS for the enhanced terminal, it is necessary to modify the system. The set of modifications to be performed in order to enhance the system security is known as hardening.

Hardening has to be performed at both application and system level. The procedure is designed to enforce the isolation and minimization security principles. More generally, it aims at minimizing the system attack surface, in terms of sources of external input and internal interactions.

The following sections will list hardening procedures for all of the system layers involved.

5.7 Application level Hardening

The application level of the Android OS is composed of the native applications and of new application installed by the user.

Hardening of the application layer is performed in two steps:

- Selective elimination of applications that do not fit the use case
- Minimization of the permissions used by the remaining applications

The first step is required since the system contains many applications that be considered useless in the PPDR context (i.e., the e-mail client or the calculator).

The second step is performed by editing the remaining applications for removing useless or dangerous capabilities. The applications are then re-compiled together with the system. This step is performed since smartphone applications typically connect to several services for acquiring data used to enhance user experience. An example of this procedure is the contact application that connects to the Internet for acquiring data from social network. These procedures do not provide any useful functionalities and constitute a potential security issue.

5.8 Service level hardening

Android uses services to perform long-running tasks in background. Services perform system administration tasks in background. Applications also uses services to perform background task without blocking the user interface and to provide capabilities to other system components.

This hardening phase will focus on the system service. It is useful to note that the open source OS basic release does not include Google services or any OEM-specific component. The service minimization affects only services related to capabilities not used in the final version of the terminal (e.g. NFC related services) and services implementing superfluous capabilities (e.g., content synchronization services).

5.9 System level hardening

At system level, SELinux is used to reinforce isolation of system components. SELinux policies are defined to:

- force access control on the system procedures, as it run only if the first layer of checks is successful;
- enforce fine-grained isolation through the development of use case specific policies.

Specifically, the policies are developed by creating domains surrounding the added-value applications. SELinux-type attributes are then specified for the files used by the application, so that a given application is only able to read, write, or execute on an authorized list of files, thus furthermore restricting interactions with other system components.

6 REQUIREMENTS

6.1 Android Operative System

REQ#1. Bluetooth shall be disabled on the Android platform used by the enhanced terminal.

REQ#2. On the Android platform used by the enhanced terminal WiFi shall be enabled as access point with WPA enabled.

REQ#3. On the Android platform used by the enhanced terminal the USB shall be by default configured as host (OTG shall be disabled by default).

6.2 Security Manager

The Security Manager (SM) is one of the applications deployed inside Android Device of the enhanced terminal. All the applications designed for ISITEP enhanced terminal are characterized by a common framework:

- an application communicates with the others using the Android Broadcast Bus;
- an application exploits the interfaces defined in D5.2.2 (to verify the communication service) and D 5.6.2 (to exchange messages with control room applications);
- an application is composed by an always-running service and eventually by an HMI for user interaction;

The Security Manager is local to the enhanced terminal; therefore it shall not exchange data with control room applications and it does not use the message interface D 5.6.2.

In order to fit the Security Manager inside the ISITEP enhanced terminal the following requirements have been established:

REQ#1. The SM shall run as a service.

REQ#2. The SM shall be able to activate the CM App when the end-user has been authenticated on the device.

REQ#3. The SM, through the PlatformIF specified in D5.2.2, shall be able to receive notifications from the TETRA and TETRAPOL Modems returning information about the correct registration\authentication status on the TETRA / TETRAPOL networks.

In order to face security threats addressed in chapter 4 the following requirements have been established:

REQ#4 The SM shall be able to configure the USB port as device when an administrator operator has been authenticated to the SM. (Note: In this way the administrator operator will be able to perform software upgrade.)

REQ#5. The SM shall be able to encrypt\decrypt data.

REQ#6. The SM shall be able to handle authentication mechanisms when the user requires to access to stored data

7 DESIGN DESCRIPTION

7.1 Architecture graphical interface

The operator will visualize the traditional Android launcher interface, with a limited number of icons with respect to traditional Android systems.

The icons that are relevant for this deliverable are the ISITEP HMI and the SM HMI.

The ISITEP HMI, when opened, will request operator authentication. This functionality is described in details in Section 7.3.

If the authentication procedure is successful, the operator will visualize a HMI, similar to the traditional Android launcher, containing the added-value applications. This interface model is similar to the Android for Work interface 0.

The SM HMI will allow administrators to toggle security settings for performing device maintenance. This functionality is described in details in section 7.4.

7.2 Architecture functionalities

The security manager is in charge of the dynamic security functionalities of the system. In more details : the user authentication, providing remote administration capabilities based on the security status on the network, protecting data stored on the terminal, and granting administrator level access to authorized operators.

7.3 Operator Authentication

The security manager will perform the authentication of the operator using the terminal. The authentication information shall be requested when the ISITEP HMI is started.

The requested credentials will be univocally associated with the operator. The terminal behaviour will therefore be associated with the actions of a single operator.

Failing the authentication will prevent access to the ISITEP HMI.

The communications manager will not start if the user is not logged in the application.

After a maximum number of failed authentication attempts, set by policy, the terminal will wipe the device data.

7.4 Maintenance mode

The security manager will allow administrators to perform maintenance on the terminal.

The SM GUI will allow administrator authentication and it will enable the USB debugging modality.

USB will then be used by administrator to update and manage the system.

7.5 Remote killing

The security manager will support the remote wipe of the device data. This functionality will complement the remote killing feature already available in the TETRA and TETRAPOL modems. In this way stolen/lost terminals will not be able to connect to the network and will not leak sensitive information.

If the security manager receives a remote wipe request, it will first check the connection status using the connection manager application. If the connection is trusted, the security manager will proceed to wipe the device, performing a factory reset of the Android device.

7.6 Data confidentiality

The security manager will encrypt data received by the device as an additional protection in case the terminal is stolen/lost. The operator will be able to display the incoming message content without being required to insert his/her credentials.

After the first visualization, the data will be stored in encrypted form, with a key derived from the operator password. The algorithms used for key derivation and encryption on the device will be decided according to the security level required (i.e., AES 256 bit).

The operator will be able to select the data to be visualized from a list in the corresponding added-value applications.

The operator will choose from a list of encrypted data identified by the date and hour of arrival.

If the authentication fails more than a pre-set number of times, the security manager will wipe (i.e. factory reset) the device.

7.7 Architecture Interfaces

In Figure 3, the block diagram representing the interfaces of the security manager is shown.

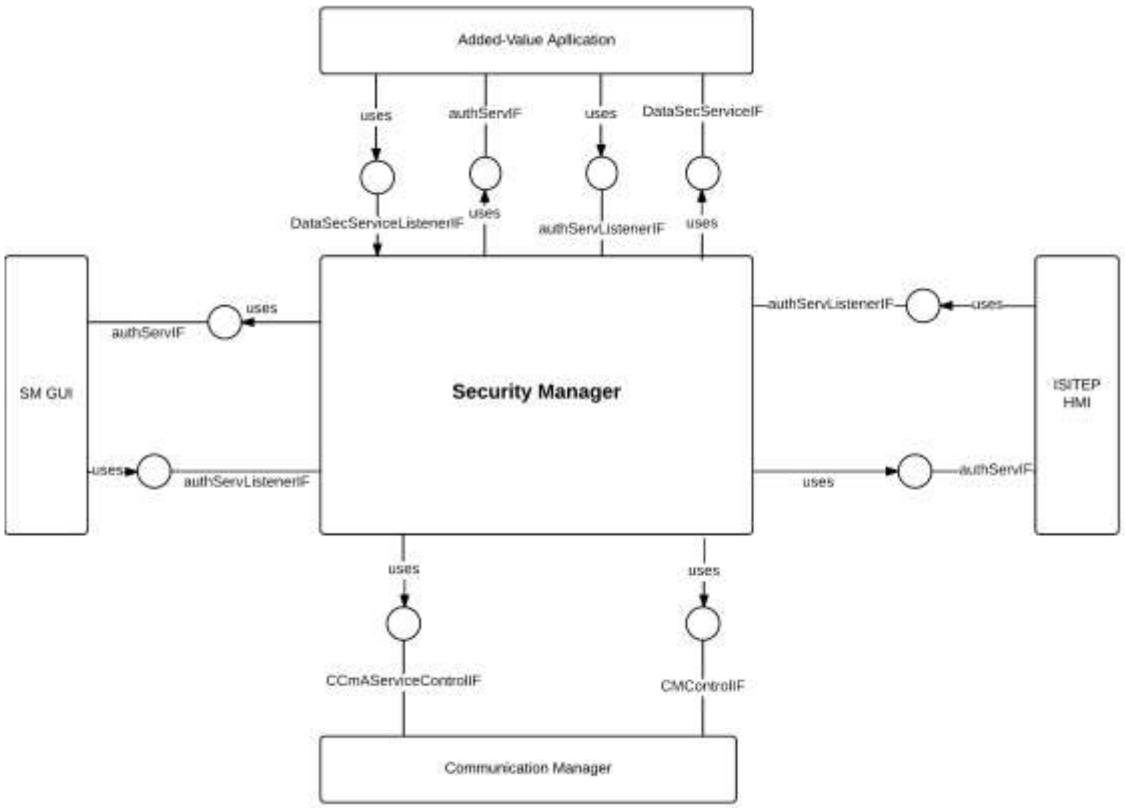

Figure 3 Security Manager Interfaces

7.8 Authentication service interface

The authentication service interface is used by the SM to perform operator authentication.

The interface is used to authenticate operators when they first start the ISITEP HMI, when they request to visualize encrypted stored data or when an administrator performs maintenance on the system.

The interface is implemented by mean of intents.

- authServListenerIF
  - authenticateOpReq(bundle opCredentials)

  This message is used to verify the validity of the operator credentials. If the sender does not get a reply after a pre-defined time, it will retry the operation.

  If the authentication fails, the message sender will prompt a message to the operator. The message will make the operator aware of the failed authentication, displaying the number of authentication attempt left.

  If the operator exceeds the max number of attempt, the security manager will wipe the device data.

  - logoutCurrentUser()

  This message is sent by the ISITEP HMI when the operator requests to logout from the ISITEP HMI itself.

If the ISITEP HMI does not get a reply after a pre-defined time, it will retry the operation.

When the logout procedure is complete (i.e. the communication manager is stopped), the SM will send a confirmation of the operation.

If the ISITEP HMI does not get a reply after a pre-defined time, it will retry the operation.

- logoutAdmin()

This message is sent by the SM HMI when the operator requests to logout from the SM HMI itself.

If the SM HMI does not get a reply after a pre-defined time, it will retry the operation.

When the logout procedure is complete (i.e. the SM HMI is stopped), the SM will send a confirmation of the operation.

- authServIF
  - authSuccesful()

  This message is sent by the SM to confirm successful user authentication.

  If the SM does not get a reply after a pre-defined time, it will retry the operation.

  If the authentication request is generated by the ISITEP HMI, the interface become available for the operator, giving access to the added-value applications. The SM will start the communication manager after confirming authentication.

  If the authentication request is generated by an added-value application, the SM will maintain a brief timer, during which the operator can access stored data without the need of authenticating again.

  If the authentication request is generated by the SM GUI, the administration GUI will be displayed.

  - authFailed(int attemptLeft)

  This message is sent by the SM if the credentials provided are not valid.

  If the SM does not get a reply after a pre-defined time, it will retry the operation.

  The SM manager will keep track of the number of authentication attempts made. If more than a pre-defined number of unsuccessful authentications attempts are made, the SM will wipe the device data. The authentication counter will be reset upon successful authentication.

  - logoutSuccesful()

  This message is sent by the SM after the user has requested to terminate the current session.

  It confirms that the operator has been successfully logged out of the device (e.g. the communication manager has been stopped or the SM HMI has been stopped).

7.8.1 Communication Manager Interface

This interface is used for communications between the SM and the communication manager.

It provides network security monitoring for the SM and it is used to start and stop the communication manager when the ISITEP HMI is started.

- **CMControlIF**
  - **activateCM()**

    This message is used by the SM to start the communication manager when the ISITEP HMI is started.

    If the SM does not get a reply after a pre-defined time, it will retry the operation.
  - **deactivateCM()**

    This message is used by the SM to stop the communication manager when the operator asks to be logged out of the system.

    If the SM does not get a reply after a pre-defined time, it will retry the operation.
  - **CCmAServiceControlIF**

    This interface and its messages are described in details in D 5.2.2. The communication manager sends asynchronous messages on this interface when network related events occur.

    The security manager uses the current security status of the network to verify if a remote input is to be trusted.

7.8.2 Data security Interface

This interface is used to protect device data in case the terminal is stolen/lost.

The interface is implemented by mean of explicit intents.

Added-value applications use this interface to request data encryption to the SM.

- **DataSecServiceListenerIF**
  - **archiveData(bundle data, int[ ] date, int[ ] hour)**

    This message is used by the added-value applications to request data encryption to the SM.

    If the added-value application does not get a reply after a pre-defined time, it will retry the operation.

    The request contains the data to be encrypted together with the details on the arrival date and hour that will be used to identify the data.

  - **retriveArchive(bundle encryptedData)**

    This message is sent by the added-value applications for requesting access to a particular encrypted element.

    The encrypted data is exchanged with the security manager as an argument.

    The security manager will send back the decrypted data.

- **DataSecServiceIF**
  - **decryptedData(bundle Data)**

    This message is used by the SM to return decrypted data to the added-value application.

  - **storeEncryptedData(bundle encryptedData)**

    This message is used by the SM to return encrypted data to the corresponding added-value application. The data are then stored by the added-value application.

8 SOFTWARE DESIGN

8.1 Application Architecture

The SM will be implemented as an Android application. The main application components are:

HMI: the graphical user interface that the administrator uses to perform maintenance task in the device.

SMService: an Android service that implements the long running functionalities of the SM.

8.2 State machine diagram

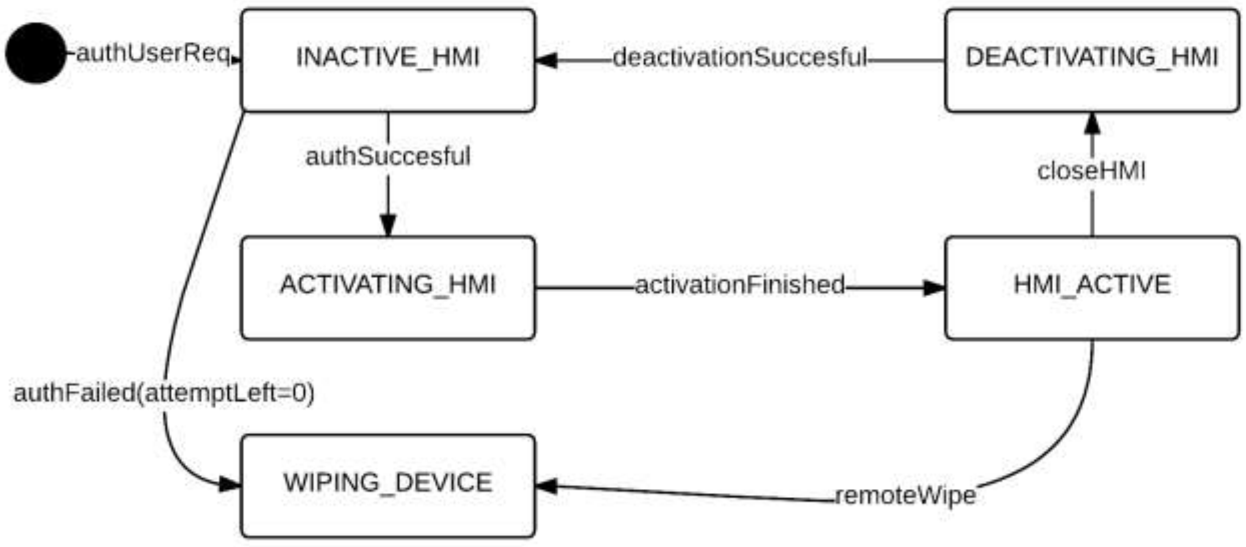

Figure 4 Security Manager State Machine Diagram

As shown in Figure 4, during its life cycle the CM can assume one of the following internal states:

- **INACTIVE_HMI**

  In this status, the ISITEP HMI is not active. The SM listens for authentication requests.

  This status is acquired when:

  - The device is switched on
  - The operator logs out of the device
  - The administrator logs out of the SM HMI

- **ACTIVATING_HMI**

  In this status the operator has been successfully authenticated and the SM is preparing the system.

  The SM proceeds to start the communication manager.

  The SM signals to the ISITEP HMI that the authentication has been successful.

  This status is acquired when:

  - The operator successfully authenticates on the system

- **HMI_ACTIVE**

  In this status the operator is logged in the system. The SM monitors the network security status through the communication manager.

  The SM offers data encryption and access to previously encrypted content for the added-value applications

  The SM listens to network commands, selectively executing them based on the network security status.

  This status is acquired when:

  - The operator successfully authenticates on the system and the SM ends the initial setup.

- **DEACTIVATING_HMI**

  In this status the operator has requested to log out of the system through the ISITEP HMI.

  The SM is stopping the communication manager and signals successful logout to the ISITEP HMI.

  This status is acquired when:

  - The operator requests to log out of the system.

- **WIPING_DEVICE**

  In this status the SM is wiping the device data (i.e. factory reset of the Android system).

  This status is acquired when:

  - The operator fails more than a pre-defined number of authentications attempts
  - The SM receive a remote wipe request and the network connection is trusted.

8.3 Sequence Diagrams

8.3.1 User authentication

In this phase the PPDR operator is starting the ISITEP HMI. The overall process is shown in Figure 5.

The ISITEP HMI requests the operator credentials and forward them to the SM for validation.

If the operator credentials are valid, the SM will respond with an *authSuccesful* message, starting the communication manager and authorizing the start of the ISITEP HMI.

Otherwise the SM will answer with an *authFailed* message, including the number of authorization attempts available before device wipe as an argument.

When the operator requests to close the ISITEP HMI to log out of the device, the SM will stop the communication manager and authorize the stopping of the ISITEP HMI, going back to the INACTIVE_HMI status.

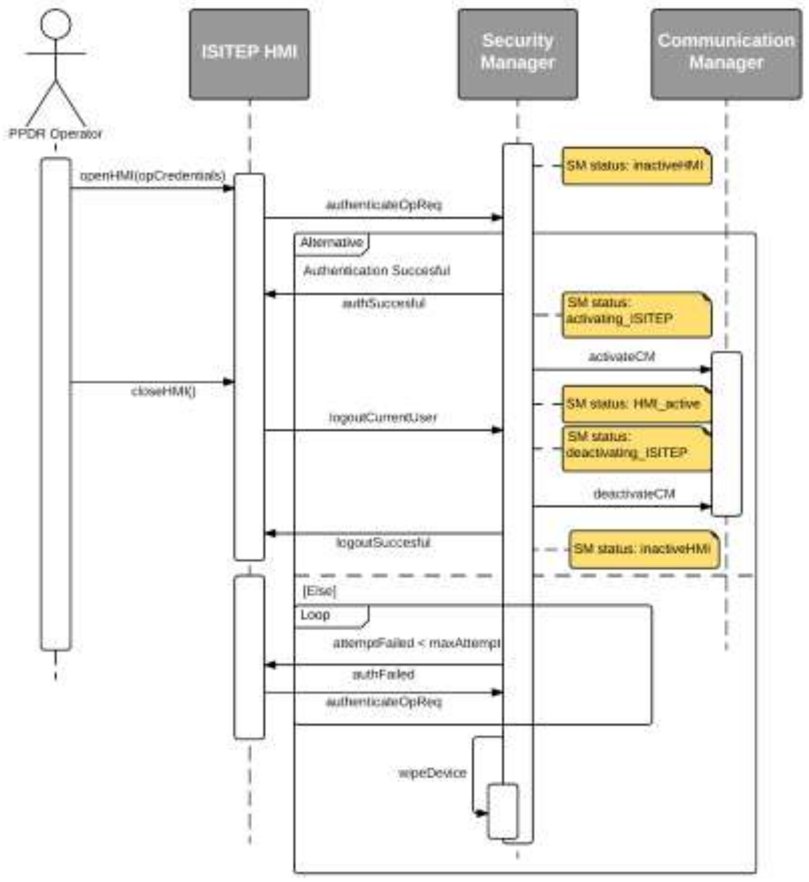

Figure 5 Operator authentication sequence diagram

8.3.2 Device wipe

In this scenario a network administrator wants to prevent information disclosure by deleting all information available on the device. The sequence diagram is shown in Figure 6.

The administrator sends a remote wipe signal to the device.

- Upon receiving the remote wipe request, the SM will verify the network state.
- If the network is trusted, the SM will proceed to wipe the device, otherwise the request is discarded.
- In both cases the SM will give feedback to the administrator about the action taken.

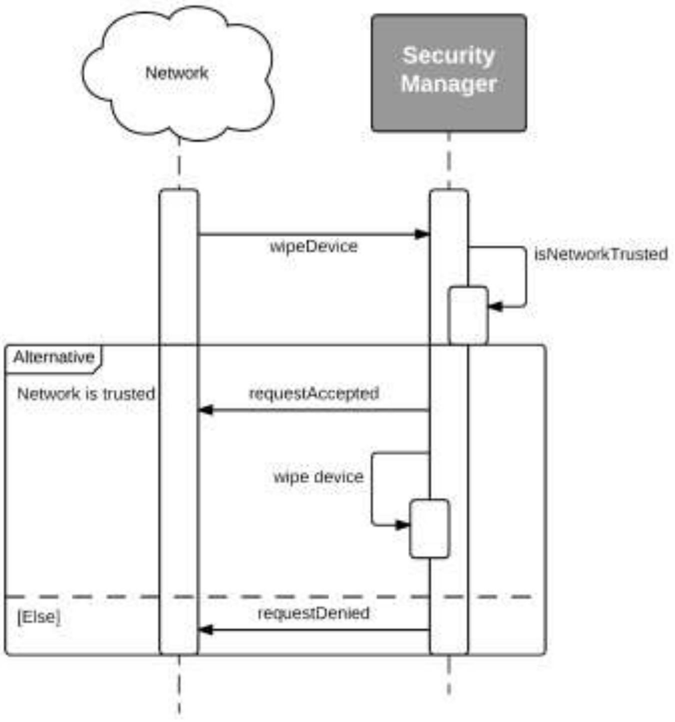

Figure 6 Remote wipe sequence diagram

8.3.3 Data protection

When an added-value application receives data from the network, the data are visualized on the device display. After the operator has visualized it and closed the HMI, the application will send an *enryptData* request to the SM, including as argument the data itself and metadata used to compute an identifier for the data (i.e. date and hour of arrival). The SM will encrypt the data received and save them with the identifier computed from the metadata. The SM will then send the encrypted file back to the added-value application. This process is summarized in Figure 7.

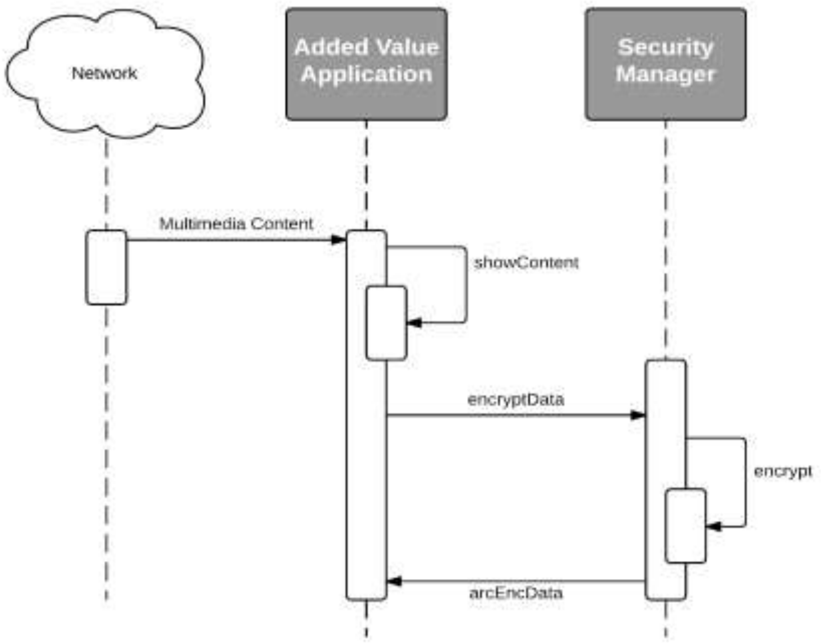

Figure 7 Data first encryption

As shown in Figure 8, if an operator (in the same ISITEP HMI session or in a successive one) wants to examine stored data, it will first have to choose the element from a list of identifiers. After choosing a particular element, the added-value app associated will send a *decryptData* request to the SM.

The SM will ask the operator to authenticate.

If the authentication is successful the data will be decrypted and sent back to the added-value application. Accessing to the data belonging to the active added-value application will not request authentication again for a pre-set period of time

In case the authentication fails, the operator will be able to try again a pre-defined number of times. If the authentication procedure fails more than *attemptLeft* times, the SM will wipe the device data.

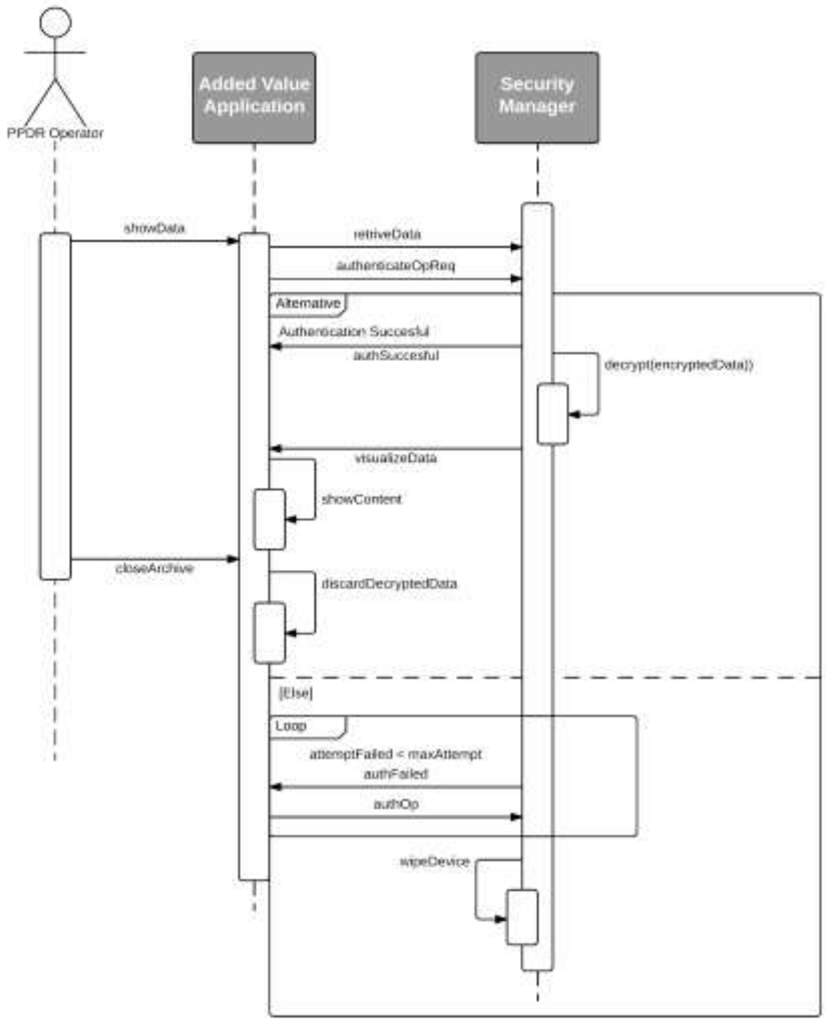

Figure 8 Encrypted data retrieval

8.3.4 Administrator Mode

If the operator starts the SM HMI, it will be asked to insert valid administrator credentials (see Figure 9).

If the authentication is successful the operator will get access to the administrator interface (i.e. the SM GUI) and will be able to enable USB debugging.

In case the authentication fails, the operator will be able to try again a pre-defined number of times. If the authentication procedure fails more than *attemptLeft* times, the SM will wipe the device data.

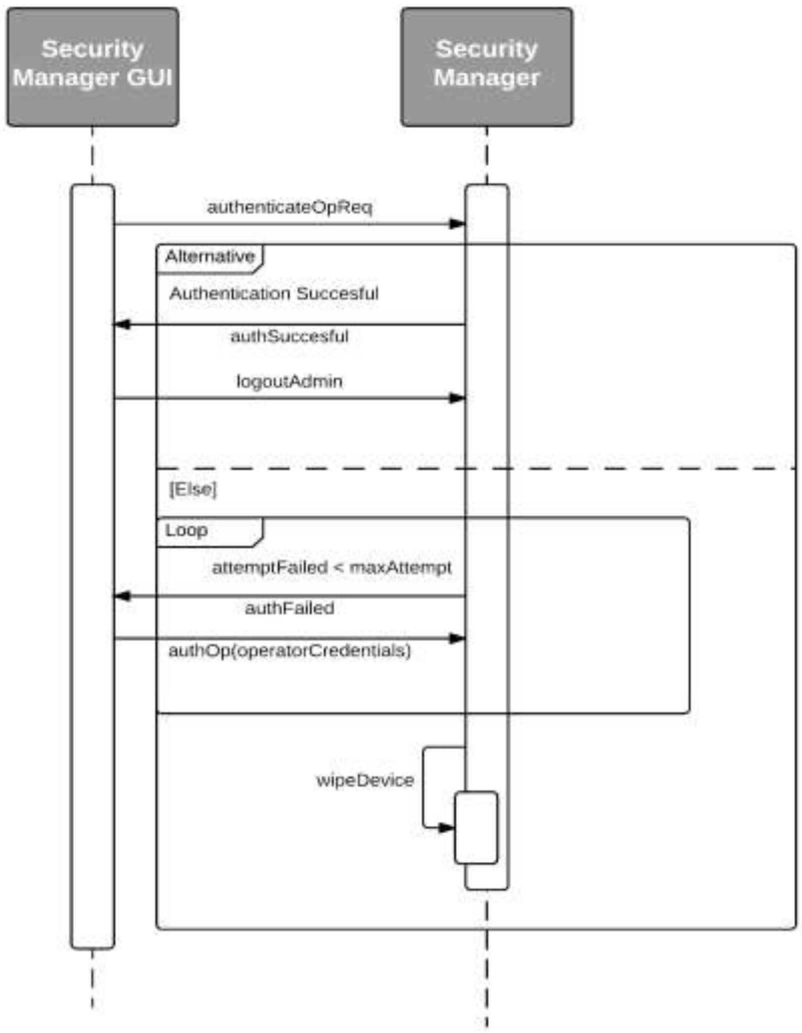

Figure 9 Administrator login sequence diagram

9 REQUIREMENT TRACEABILITY

Req. number	Ref.	Solution Addressed in the Design
I_SEC_TC03_02	D2.2.2 §7.2.3	The security manager shall accept commands to modify security configurations only if the network is currently authenticated. See 7.5.
I_SEC_TC07_12	D2.2.2 §7.2.7	The security manager shall be able to wipe the terminal data upon remote request if the network connection is trusted. See 7.5
I_SEC_TC08_01	D2.2.2 §7.2.8	In the vehicular configuration, a secure communication shall be established between the modem and the terminal.
I_SEC_TC08_02	D2.2.2 §7.2.8	In the vehicular configuration, a secure communication shall be established between the modem and the terminal.

10 CONCLUSIONS

In this deliverable the design of the Security manager for the ISITEP framework is detailed.

TETRA and TETRAPOL modems are equipped with security functions providing authentication of the modem on the Network Infrastructure and confidentiality of communications through encryption mechanisms. TETRA and TETRAPOL technologies provides also Enable / Disable capability that ensure against eavesdropping or poisoning information from a stolen or loss terminal.

In this framework, the use of an open source Operating System (i.e., Android) poses severe challenges concerning threats and vulnerabilities. In this deliverable theses vulnerabilities have been discussed and possible solutions for coping with such security faults have been designed.

The outcomes of this deliverable will be used by User Interface and Business Logic Manager (WP 5.6). It will also be used for terminal integration and testing (WP 5.7) and for terminal and services validation and qualification (WP 5.8).

11 REFERENCES

[1] ISITEP D 5.1.2 Enhanced Terminal Open Architecture
[2] ISITEP D 2.2.2 Security Requirements
[3] ETSI ETR 332: "Security Techniques Advisory Group (STAG); Security requirements capture", November 1996
[4] ETSI ETR 086-3: "Trans European Trunked Radio (TETRA) systems; Technical requirements specification; Part 3: Security aspects", January 1994
[5] ETSI TR 102 512 V1.1.1, “Security requirements analysis for modulation enhancements to TETRA”, August 2006
[6] ETSI EN 300 392-7 V3.3.1 (2012-07), “Terrestrial Trunked Radio (TETRA); Voice plus Data (V+D); Part 7: Security”
[7] 3GPP TS 21.133, “3G Security; Security Threats and Requirements”, December 2011
[8] 3GPP TR 22.893, “Study into identification of advanced requirements for IP interconnection of services; (Release 10)”
[9] ETSI EN 300 392-3-1 V1.3.1 (2010-08), “Terrestrial Trunked Radio (TETRA);Voice plus Data (V+D); Part 3: Interworking at the Inter-System Interface (ISI); Sub-part 1: General design”
[10] i3 Forum, “Security for IP Interconnections (Release 1.0)”, May 2011
[11] ETSI/TETRA(13)000023r1, TETRA#41, 11th - 13th March 2013, Title: ISI authentication, Source: Cassidian, BDBOS, MSB, ERVE, RIKS
[12] SFPG 12/26r2,”Security consideration in TETRA ISI authentication r2”, 10th September 2012
[13] European Parliament and European Council. Directive 1995/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data. 1995
[14] Thorsten Schreiber. Android Binder: Android Inter-process Communication. url: http://www.nds.rub.de/media/attachments/files/2012/03/binder.pdf.
[15] B. Schneier. “Attack Trees”. Dr. Dobb’s Journal, Vol.12. 1999.

[16] Google inc. “Android for Work” https://www.android.com/work/