[ST] VOS AutoSAR-OS Security Target v1.5

2023-06-30

VOS AutoSAR-OS V3.0.0 Security Target

华为技术有限公司

Huawei Technologies Co., Ltd.

Change History

Date	Version	Description
2023-01-19	1.0	Complete version
2023-03-04	1.1	Update Trust OSA’s Access Permission
2023-04-17	1.2	Update Table 3.2 TOE deliverable list and TOE type
2023-05-25	1.3	Update Table 1-1 document titles
2023-06-12	1.4	Update of A.PLATFORM and OE.PLATFORM to add physical attack protection
2023-06-30	1.5	Update of links in table 1 and editorial corrections on pages 1 and 26.

Table of Contents

1 ST introduction

1.1 ST reference and TOE reference

1.1.1 ST reference

ST title: VOS AutoSAR-OS V3.0.0 Security Target
ST version: 1.5
ST publication date: 2023-06-30
ST author: Huawei Technologies Co., Ltd.

1.1.2 TOE Reference

TOE name: VOS AutoSAR-OS
TOE version: V3.0.0

1.2 TOE Overview

1.2.1 TOE Type

The TOE is the kernel of the Huawei intelligent **vehicle control operating system** that is Compatible with the AutoSAR CP standard [AUTOSAR].

1.2.2 TOE usage and major security features

As the operating system kernel software for vehicle control, the TOE can be used in various Electronic Control Unit (ECU) products of a vehicle. The TOE provides a software platform for application developers to develop applications that control the ECU.

The major security features of the TOE include:

- SF.OSID: OS Object Identification
- SF.OSOM: OS Object Management
- SF.OSAM: OS Application Management
- SF.OSSP: OS Service Protection

- SF.MP: Memory Protection
- SF.SM: Stack Monitoring
- SF.TS: Task Scheduling
- SF.TP: Time Protection
- SF.EH: Error Handling and Exception Handling

1.2.3 Non-TOE hardware/software/firmware required by the TOE

The TOE runs on the hardware MCU of the ECU, and requires one of the following supported hardware architectures:

- Tricore：TC264,TC277,TC357,TC377,TC389,TC397；

"HAS Studio Configurator" is an ECU configuration tool for vehicle control software development. This configuration tool is required to configure the security attributes for each application. You can configure the TOE in preparation phase on the GUI and generate the dynamic code of the TOE. Dynamic code (as configuration parameters, including security configuration parameters) cannot be modified during running. It compiles with static code to generate a runnable TOE.)

The TOE does not include the applications running on the TOE, which need to be developed by the application developer. These applications are included into a final image using a configuration tool that fixes the security attributes for each application.

1.3 TOE Description

1.3.1 TOE logical scope

As the kernel of an operating system, the TOE provides functions

such as task management, task scheduling, resource (lock) management, event mechanism, interrupt management, clock, alarm and scheduling table, and exception handling for the normal operation of upper-layer applications. The hardware abstraction layer shields hardware differences upwards, provides standard AutoSAR OS APIs, and builds an AutoSAR OS-based software ecosystem. It is mainly used in the field of vehicle MCUs. Figure 1-1 shows the TOE system framework inside its operational environment.

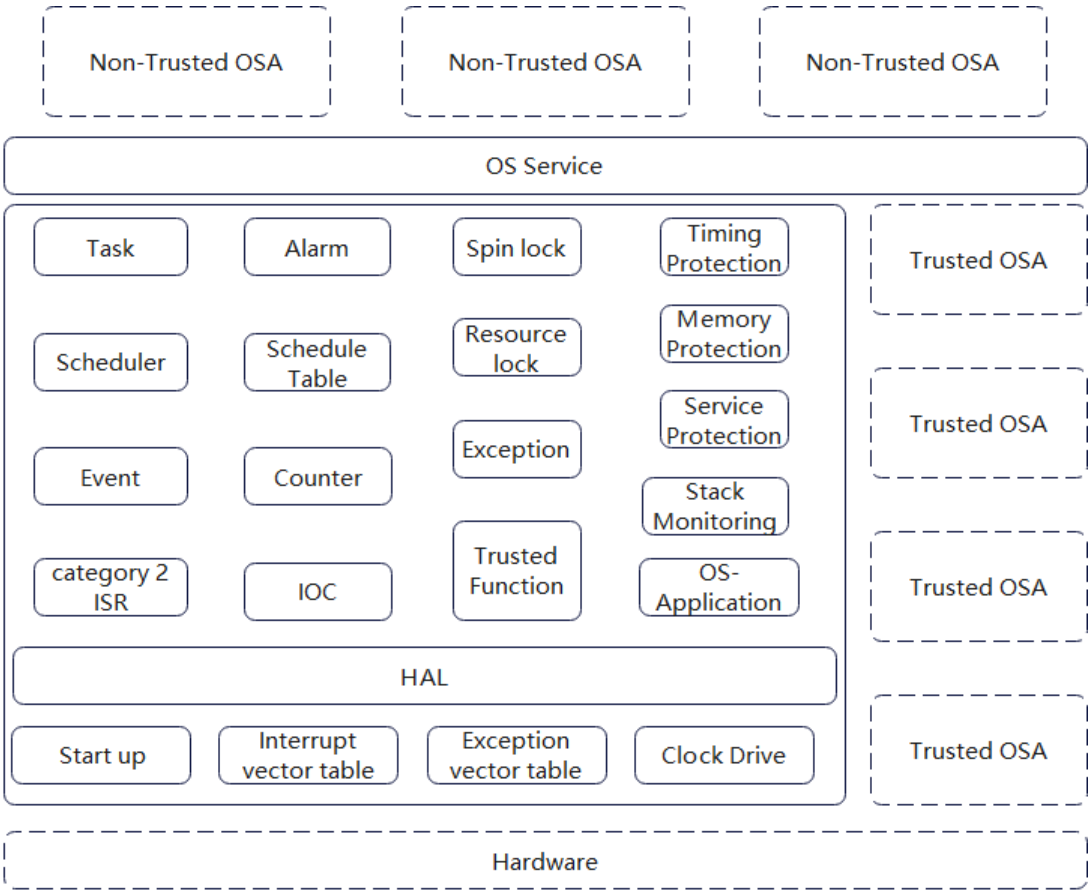

Figure 1-1 TOE System Framework

The TOE consists of the following security-related modules, as shown in Figure 1-2.

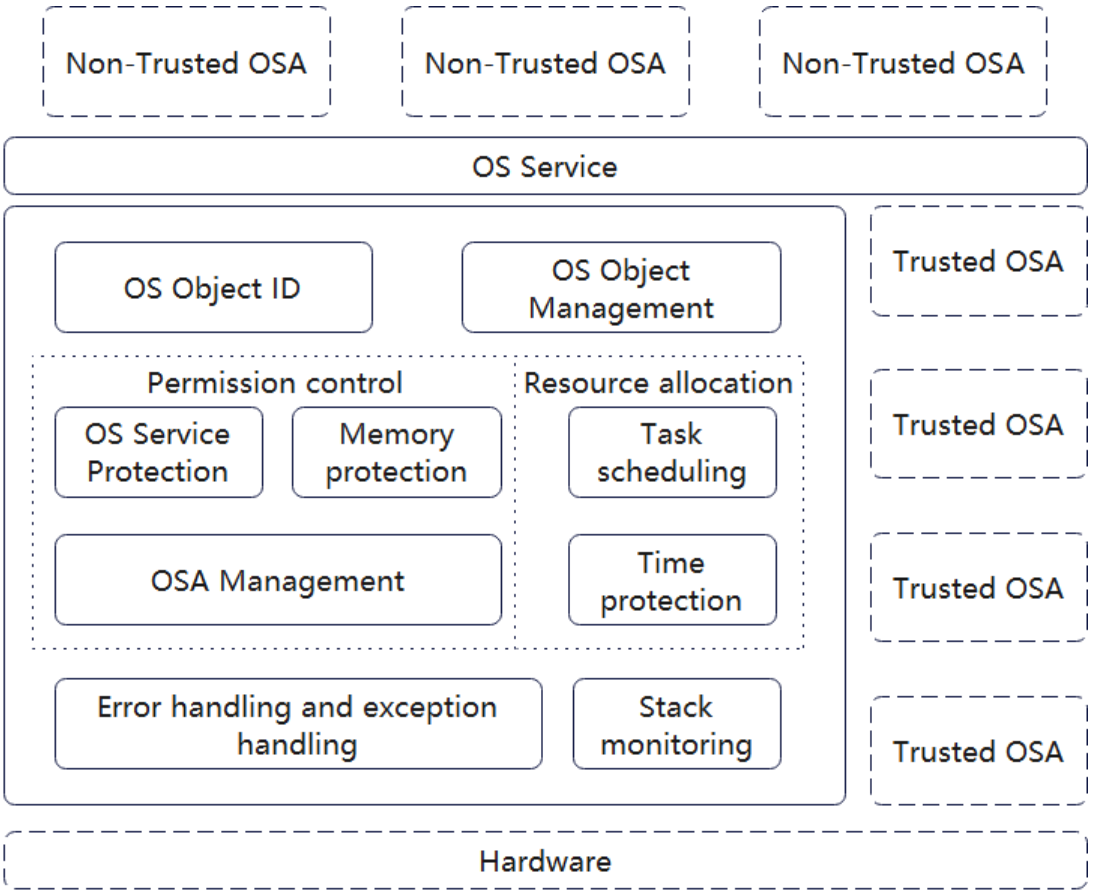

Figure 1-2 TOE Security Architecture

The following components are included, each providing its own Security Feature (SF):

- SF.OSID: OS Object ID
- SF.OSOM: OS Object Management

Permission control, which includes

- SF.OSAM: OSA Management
- SF.OSSP: OS Service Protection
- SF.MP: Memory Protection

Resource utilization, which includes

- SF.TS: Task Scheduling
- SF.TP: Time Protection
- SF.EH: Error Handling and Exception Handling
- SF.SM: Stack Monitoring

The following sections describes each of these features in more detail.

1.3.1.1 SF.OSID: OS Object Identification

The TOE allocates OS object IDs to various OS objects.

1.3.1.2 SF.OSOM: OS Object Management

The OS allows to manage OS objects by providing the corresponding structure array of the kernel OS object to initialize them. This can be supported by an OS configuration tool.

1.3.1.3 SF.OSAM: OS Application Management

An OS Application (OSA) is a collection of OS objects that manages the resources (data segments and code segments) shared by these objects. An OS object can only belong to one OSA. The OSA is also considered the basic unit for sharing resources and configuring permissions.

OSAs are classified into trusted OSAs, which run in the kernel privilege, and untrusted OSAs, which is a traditional user application running in the user privilege.

1.3.1.4 SF.OSSP: OS Service Protection

The TOE allows the user to configure permissions based on the OSA. Each OS object in the same OSA has the same permission. Based on the permissions, OSAs are allowed access to other OS objects or not.

1.3.1.5 SF.MP: Memory Protection

The TOE configures the non-TOE hardware MPU for each OSA data segment and code segment, and configures the read and write permissions on the MPU. When accesses are performed, the MPU will

accept or reject these accesses, and TOE operation will continue based on the MPU response.

1.3.1.6 SF.SM: Stack Monitoring

The TOE provides a stack monitoring function. The TOE allocates a task stack to each task to prevent stack operations from going out of bounds and provides a stack protection function. If the stack operation of a user task exceeds the defined threshold, exception handling will be triggered.

1.3.1.7 SF.TS: Task Scheduling

The priority-based FIFO scheduling mechanism of the TOE provides a reasonable CPU resource usage mode for tasks.

1.3.1.8 SF.TP: Time Protection

The user can configure the maximum running time of a single task. If the maximum running time exceeds the maximum running time (as reported by a non-TOE hardware timer, a time protection exception is triggered.

1.3.1.9 SF.EH: Error Handling and Exception Handling

- Error handling: When an error occurs, the TOE invokes ErrorHook to handle the error.
- Exception handling: When an exception occurs, the TOE saves the exception onsite data and triggers the ProtectionHook. Users can read the exception onsite data in the ProtectionHook for fault locating and classifying system or task exceptions in the ProtectionHook.

1.3.2 TOE Operational Environment

Hardware is the physical part of the TOE operational environment. The TOE needs to adapt to different hardware platforms to shield hardware

differences and provide unified basic services, including protection against hardware attacks. During the evaluation, the TOE runs on the Tricore chip. The hardware environment of the TOE includes CPUs, RAMs, ROMs (not mandatory), system clocks, interrupt controllers, and MPUs.

1.3.3 TOE physical scope

The physical scope of the TOE includes operating system software and the corresponding installation, configuration, and operation guidance, as shown in Table 1-1. The software part, as shown in the solid line in Figure 1-2, belongs to the TOE security architecture. Other contents are beyond the TOE physical boundaries. Hardware, firmware, trusted OSA, and untrusted OSA are not covered by TOE.

TOE upper boundary: The TOE provides OS system service interfaces for users and functions as the physical upper boundary of the TOE.

TOE lower boundary: The TOE runs on hardware and relies on the hardware to correctly implement functions such as the CPU, interrupt controller, clock, and MPU. The hardware interface is the lower physical boundary of the TOE, as it does not rely on any firmware.

Table 1-1 TOE deliverable list

Type	Delivery Item	Version	Method of delivery
Software	VOS AutoSAR-OS V3.0.0 iVOS 3.0.186.zip	.zip file	发布平台 https://support.huawei.com/enterprise/zh/software/index.html
Software signature file	VOS AutoSAR-OS V3.0.0 iVOS 3.0.186.zip.hwp7s	.zip.hwp7s	
Product guidance	VOS AutoSAR-OS V3.0.0 Product Usage Reference (Chinese version), HUAWEI VOS 3.0.0 Os 参考手册, Rev. 01, 2023-04-15	pdf	发布平台 https://support.huawei.com/enterprise/zh/software/index.html
	VOS AutoSAR-OS V3.0.0 Preparation Procedure, Rev.1.9, 2023-04-17	pdf	

Type	Delivery Item	Version	Method of delivery
	VOS AutoSAR-OS 3.0.0 Operation User Guide, Rev.1.4, 2023-03-04	pdf	

1.4 Terms and abbreviations

1.4.1 Terms

Assets: Entities that the TOE owner places value upon

TSF Data：Data for the operation of the TOE upon which the enforcement of the SFRs relies

User data: data from the user that does not affect TSF operation

Type 1 interrupt: interrupts that are not managed by the OS and do not use OS services

Type 2 interrupt: interrupts that are managed by the OS and can interact with the OS

Interrupt controller: A peripheral that receives interrupt inputs from other peripherals and sends interrupt signals to the CPU after arbitration

OS object: Logical component in the OS that includes tasks, clocks, alarms, scheduling tables, resource locks, and spin locks

Control block: describes all information about the required state of the OS object

Descriptor: A kernel data structure that describes OS objects

Task Control Block (TCB): Type of OS object that describes the required state information of the task

Task Descriptor (TDB): Type of OS object that describes the kernel data structure of the task configuration properties

OS Service: A service provided by the OS, accessed through a system call or API call

ACCESS_APPLICATION: A security attribute of an OS object that describes which OS Applications can access this object

1.4.2 Acronyms

Abbreviation	Description
EAL	Evaluation Assurance Level
OSP	Organisational Security Policy
PP	Protection Profile
RAM	Random Access Memory
ROM	Read Only Memory
SAR	Security Assurance Requirement
SFP	Security Functions Policy
ST	Security Target
TOE	Target of Evaluation
TSF	TOE Security Functions
TSFI	TSF Interface
OSA	OS Application
APP	Application(See OSA)
DB	Descriptor Block
CB	Control Block
TDB	Task Descriptor Block

TCB	Task Control Block
MPU	Memory Protection Unit
FIFO	First In First Out
ECU	Electronic Control Unit
MCU	Microcontroller Unit
AutoSAR	Automotive Open System Architecture
CP	Classic platform
HAL	Hardware Abstraction Layer
ISR	Interrupt Service Routine

2 Conformance claims

This ST claims conformance to Common Criteria version 3.1 revision 5, which comprises:

- [CC-1] Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and general model, Version 3.1, Revision 5, April 2017
- [CC-2] Common Criteria for Information Technology Security Evaluation, Part 2: Functional security components, Version 3.1, revision 5, April 2017
- [CC-3] Common Criteria for Information Technology Security Evaluation, Part 3: Assurance security components, Version 3.1, revision 5, April 2017
- [CEM] Common Methodology for Information Technology Security Evaluation, Evaluation methodology, April 2017, Version 3.1, Revision 5

The conformance claim for [CC-2] and [CC-3] is conformant, and for the assurance claims, conformance is claimed to the Assurance Level package of EAL4, augmented by AVA_VAN.5 and ALC_FLR.1.

3 Security Problem Definition

As an operating system, the TOE needs to ensure that any upper-layer user is under the kernel's control policy for the kernel's available to assets. TOE security issues defined in this chapter are mainly damage to kernel integrity, confidentiality, or availability caused by unauthorized access to kernel assets. Different TOE asset types face different security issues. This chapter provides a limited classification of kernel assets and then analyzes the threats that each asset may face from malicious and flawed users at the upper layer.

3.1 User

The user of TOE is the OSA task running on the kernel. Threat agents for all following threats are Untrusted OSA tasks.

3.2 Assets

The TOE aims to protect the following assets:

- OSA code and data
- OS services
- System resources

3.3 Threats

The following describes the threats to the assets that the TOE aims to prevent, in cooperation with the operational environment of the TOE.

3.3.1 Unauthorized access (T.UNAUTHORIZED_ACCESS)

A user task from an untrusted OSA accesses (e.g. through an OS service or through a kernel object) an OS object, belonging to another OSA (i.e., OSA data), which has not authorized the untrusted OSA to access its objects.

3.3.2 Invalid Memory Access (T.MEMORY_ACCESS)

A user task from an untrusted OSA tries to access the OSA data and code of another OSA by accessing its memory directly, by accessing kernel data, or by exceeding its stack space.

3.3.3 System resource monopoly (T.RESOURCE_MONOPLY)

A user task from an untrusted OSA claims all system resources (such as the CPU) for a long period of time, preventing other OSAs from

executing.

3.4 Assumptions

The following describes all assumptions on the operational environment of the TOE.

3.4.1 Hardware platform (A.PLATFORM)

It is assumed that the TOE is integrated into a hardware environment that provides (at a minimum) the following basic capabilities to support the secure running of the operating system:

- boot program verification and secure boot of the operating system,
- a memory protection unit (MPU),
- two independent clocks (system clock and time protection clock),
- protection against physical attacks, such as probing or physical manipulation,
- protection against malfunction attacks, where environmental stress is applied to cause a malfunction, and
- protection against side-channel attacks, where sensitive information leaks as a result of leakage.

3.4.2 Personnel (A.PERSONNEL)

It is assumed that the developers of AutoSAR CP (which is the end product that integrates the TOE) are trustworthy and well-trained personnel, who shall:

- develop Trusted OSAs and integrate the TOE according to the development guide and prevent malicious tampering and intentional damage to the TOE,
- configure all OSAs such that the security needs of all user data are respected (including access rights and task priorities), and
- ensure that Untrusted OSAs are correctly configured as untrusted during the integration using the configuration tool.

4 Security Objectives

4.1 Security objectives for the TOE

4.1.1 Permission-based Access Control (O.ACCESS_CONTROL)

The TOE shall provide permission-based access restrictions to OS objects and shall ensure that OSAs cannot access related OS objects without authorization.

4.1.2 Memory Access Control (O.MEMORY_CONTROL)

The TOE shall use the non-TOE hardware MPU to implement OSA isolation, in order to protect the storage of security critical code and ensure that resources such as RAM, ROM, and peripherals that exceed the OSA limit are not accessed illegally.

4.1.3 Priority (O.PRIORITY)

The TOE shall provide a CPU preemption policy based on task priorities to ensure that user tasks can use common CPU resources based on their priorities under the TOE management.

4.1.4 Time protection control (O.TIMING_CONTROL)

The TOE shall provide a time protection mechanism to ensure that the running time of each user task does not exceed a predetermined time window.

4.1.5 Stack Monitor (O.STACK_MONITOR)

The TOE shall provide a mechanism for monitoring stack operations. When a user task is running, the stack usage of the user task is monitored to prevent overwriting.

4.1.6 Secure state (O.SECURE_STATE)

The TOE shall provide error and exception handling mechanisms to ensure that the system is in a secure state when an error or exception occurs, preventing data damage or function abuse.

4.2 Security objectives for the Operational Environment

4.2.1 Hardware platform (OE.PLATFORM)

The hardware environment in which the TOE is integrated shall

provide the following required capabilities:

- boot program verification and secure boot of the operating system,
- a memory protection unit (MPU),
- two independent clocks (system clock and time protection clock),
- protection against physical attacks, such as probing or physical manipulation,
- protection against malfunction attacks, where environmental stress is applied to cause a malfunction, and
- protection against side-channel attacks, where sensitive information leaks as a result of leakage.

4.2.2 Personnel (OE.PERSONNEL)

The developers of AutoSAR (which is the end product that integrates the TOE) shall be trustworthy and well-trained in the use of the TOE. They shall:

- develop Trusted OSAs and integrate the TOE according to the development guide and prevent malicious tampering and intentional damage to the TOE,
- configure all OSAs such that the security needs of all user data are respected (including access rights and task priorities), and
- ensure that Untrusted OSAs are correctly configured as untrusted during the integration using the configuration tool.

5 Security Requirements

5.1 TOE Security Functional Requirements

Table 5-1 lists the security functional requirements involved in this TOE:

Table 5-1 Security Functional Requirements

SFR	Description

FDP_ACC.1/CAP	Subset access control (for OS objects)
FDP_ACC.1/MEM	Subset access control (for memory protection)
FDP_ACC.1/STACK	Subset access control (user task stack space access control)
FDP_ACF.1/CAP	Security attribute-based access control (for OS objects)
FDP_ACF.1/MEM	Security attribute-based access control (for memory protection)
FDP_ACF.1/STACK	Security attribute-based access control (User Task Stack Access Control)
FIA_ATD.1	User Attribute Definition
FIA_UID.2	User ID before any action
FMT_MSA.1	Security attribute management
FMT_MSA.3	Static attribute initialization
FMT_SMF.1	Management Function Specification
FMT_SMR.1	Security Role
FPT_FLS.1	Failure and Protection Safety Status
FRU_PRS.1	Limited Service Priority
FRU_RSA.1	resource allocation

The following notation is used for operations on the SFRs:

- Assignments and selections are indicated by bold text.
- Iterations are indicated by / followed by a label.
- Refinements are indicated by bold italicized text when text is added, and bold italicized strikethrough text when text is removed.

5.1.1 User Data Protection (FDP)

5.1.1.1 Subset Access Control (FDP_ACC.1/CAP)

FDP_ACC.1.1/CAP The TSF shall enforce the **Permission SFP** on

- **Subjects: User task**
- **Objects: OS object**
- **Operations: Invoke through OS service.**

5.1.1.2 Subset Access Control (FDP_ACC.1/MEM)

FDP_ACC.1.1/MEM The TSF shall enforce the **Memory SFP** on

- **Subjects: User task**
- **Objects: Logical memory**
- **Operation: read, write, execute.**

5.1.1.3 Subset Access Control (FDP_ACC.1/STACK)

FDP_ACC.1.1/STACK The TSF shall enforce the **Stack SFP** on

- **Subjects: User task**
- **Objects: Task stack**
- **Operations: Use stack (Read and write local variables, save stack pointers and return addresses, etc.).**

5.1.1.4 Security Attribute Based Access Control (FDP_ACF.1/CAP)

FDP_ACF.1.1/CAP The TSF shall enforce the **Permission SFP** to objects based on the following:

- **User Task: OSA_ID**
- **OS Object: OSA_ID, ACCESS_APPLICATION, PRIVILEGE**

FDP_ACF.1.2/CAP The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed:

- **If the User Task OSA_ID matches the OS Object OSA_ID access is allowed**
- **If the User Task OSA_ID is included in the ACCESS_APPLICATION of the OS Object, access is allowed**
- **In all other cases, access is denied**.

FDP_ACF.1.3/CAP The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: **none**

FDP_ACF.1.4/CAP The TSF shall explicitly deny access of subjects to objects based on the following additional rules: **none**.

5.1.1.5 Security Attribute Based Access Control (FDP_ACF.1/MEM)

FDP_ACF.1.1/MEM The TSF shall enforce the **Memory SFP** to objects based on the following:

- **User task: OSA_ID**
- **Logical memory: MPU response**

FDP_ACF.1.2/MEM The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: **access is allowed if and only if the MPU response indicates that the User Task OSA_ID is allowed to access the object**.

FDP_ACF.1.3/MEM The TSF shall explicitly authorise access of subjects

to objects based on the following additional rules: **none**.

FDP_ACF.1.4/MEM The TSF shall explicitly deny access of subjects to objects based on the following additional rules: **none**.

Application Note: The enforcement of this SFR does not rely on the correct functioning of the non-TOE MPU. The TOE is required to accept any response that the MPU gives, be it correct or incorrect.

5.1.1.6 Security Attribute Based Access Control (FDP_ACF.1/STACK)

FDP_ACF.1.1/STACK The TSF shall enforce the **Stack SFP** to objects based on the following:

- **User task: ID**
- **Task stack: Owner, used stack, stack address space**.

FDP_ACF.1.2/STACK The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: **access is allowed if and only if the User task is the owner of the Task stack and the used stack would remain within the stack address space after the operation**.

FDP_ACF.1.3/STACK The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: **none**.

FDP_ACF.1.4/STACK The TSF shall explicitly deny access of subjects to objects based on the following additional rules: **none**.

5.1.2 Identification and Identification (FIA)

1.1.2.1 User Attribute Definition (FIA_ATD.1)

FIA_ATD.1.1 The TSF shall maintain the following list of security attributes belonging to individual users: **see Table 5-2**.

Table 5-2 List of attributes related to security functions

No.	Attribute Name	Remarks
1	ID	Used to uniquely identify an OS object, maintained by the TOE.
2	OSA_ID	OS APPLICATION to which the OS object belongs.
3	OSA_TRUST	Whether the OS application is trusted or untrusted
4	ACCESS_APPLICATION	Attribute of an OS Object with a list of OSAs that can access it.
5	PRIVILEGE	The privilege of the object (kernel or user)

5.1.2.2 User Identification Before Any Action (FIA_UID.2)

FIA_UID.2.1 The TSF shall require each user to be successfully identified before allowing any other TSF-mediated actions on behalf of that user.

Application Note: The ID of an OS object is the unique identifier of the user identity. It is maintained by the TOE and cannot be modified by external systems. The TSF identifies the user by obtaining the ID of the current visitor and obtains related permission information to determine

whether to allow the current operation.

5.1.3 Security Management (FMT)

5.1.3.1 Management of security attributes (FMT_MSA.1)

FMT_MSA.1.1 The TSF shall enforce the **Permission SFP, Memory SFP, Stack SFP** to restrict the ability to **see Table 5-3** the security attributes **see Table 5-3** to **see Table 5-3**.

Table 5-3 List of attributes and their management

Operation	Security attributes	Authorised role
Modify	ID, OSA_ID, OSA_TRUST , ACCESS_APPLICATION, PRIVILEGE, User Task identity, Task Stack owner and stack address space	None
Modify	MPU response	MPU
Modify	Task Stack used stack	TSF

5.1.3.2 Static attribute initialisation (FMT_MSA.3)

FMT_MSA.3.1 The TSF shall enforce the **Permission SFP, Memory SFP, and Stack SFP** to provide **fixed** default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2 The TSF shall allow **no one** to specify alternative initial values to override the default values when an object or information is created.

Application Note: Before compilation, the security attributes of the OS kernel OS objects are configured and initialized in the OS configuration tool. The OS kernel cannot create objects during running. Trusted OSAs and untrusted OSAs cannot change the default values of the object

security attributes during running.

5.1.3.3 Specification of Management Functions (FMT_SMF.1)

FMT_SMF.1.1 The TSF shall be capable of performing the following management functions: **modify the MPU response**.

5.1.3.4 Security roles (FMT_SMR.1)

FMT_SMR.1.1 The TSF shall maintain the roles **MPU**.

FMT_SMR.1.2 The TSF shall be able to associate users with roles.

5.1.4 Protection of the TSF (FPT)

5.1.4.1 Fail secure (FPT_FLS.1)

FPT_FLS.1.1 The TSF shall preserve a secure state when the following types of failures occur: **parameter errors, insufficient permissions, CPU resource exhaustion, address errors, and hardware trigger-related errors (undefined instructions, interrupts, etc.) during system calls**.

5.1.5 Resource Utilisation (FRU)

5.1.5.1 Limited priority of service (FRU_PRS.1)

FRU_PRS.1.1 The TSF shall assign a priority to each subject in the TSF.

FRU_PRS.1.2 The TSF shall ensure that each access to **CPU resources** shall be mediated on the basis of the subjects assigned priority.

Application Note: The CPU resources are occupied by the CPU resources based on the priority-based FIFO policy. In normal cases, when user tasks need to be rescheduled, the TOE scheduling module selects the queue

head from the waiting queue with the highest priority. The previously executed task is suspended and placed at the tail of the corresponding priority queue for the next scheduling.

5.1.5.2 Maximum quotas (FRU_RSA.1)

FRU_RSA.1.1 The TSF shall enforce maximum quotas of the following resources: **CPU resources** that **subjects** can use **over a specified period of time**.

Application Note: The operating system kernel security function assigns a maximum executable time to each subject (user task) in the operating system kernel security function. If the task runs for longer than this period, the operating system triggers an exception. The TOE monitors whether the running time of a task exceeds the maximum execution time based on the non-TOE hardware clock module, setting a countdown of the hardware clock module when the task is started; if the countdown is finished, the time protection interrupt is triggered and the TOE is notified that the task timed out.

5.2 TOE Security Assurance Requirements

As stated in Section 2, this ST claims conformance to the package EAL4, augmented with AVA_VAN.5, and ALC_FLR.1, as defined in [CC-3]. They are included here by reference.

6 TOE Summary Specification

This section describes the security functions implemented by the TOE and how they ensure the related security requirements are met. As the operating system kernel, the TOE provides security functions as described in Section 3.1.

1. SF.OSID: OS Object Identification
2. SF.OSOM: OS Object Management
3. SF.OSAM: OS Application Management
4. SF.OSSP: OS Service Protection
5. SF.MP: Memory Protection
6. SF.SM: Stack Monitoring
7. SF.TS: Task Scheduling
8. SF.TP: Time Protection
9. SF.EH: Error Handling and Exception Handling

6.1 SF.OSID: OS Object Identification

6.1.1 Function summary

Each OS object has a unique OS object identifier (ID). The TOE maintains its OS objects throughout the life cycle of a task. The OS object ID of a task or other OS objects cannot be modified.

The TOE does not identify real physical users, but only the tasks that run on them. An OS object can be either the subject or the object of an operation.

When an OS object performs a controlled operation on a TOE asset, the TOE identifies the current user identity based on the OS object ID and determines whether to allow or reject the operation based on the permissions related to the OS object maintained by the TOE object.

6.1.2 Corresponding SFRs

This function ensures that the SFR FIA_UID.2 is met, as all subjects

are identified throughout their whole life cycle. Further, it ensures that the first item of FIA_ATD.1 is met, assigning IDs to all objects. Finally, these identifiers are subsequently used by the other security functions to meet all other SFRs claimed in this ST.

6.2 SF.OSOM: OS Object Management

6.2.1 Function summary

The OS manages the various objects that were defined by the user and grouped into OSAs. The user can use the OS configuration tool to generate the initialized global variable configuration array consisting of a control block <OS_Object>_CB and a data block <OS_Object>_DB. This array is generated based on the OS object and its attributes as defined by the user.

Note: an <OS_Object> can correspond to a Task, Counter, or Alarm. Task_CB and Task_DB are simplified into TCB and TDB in the implementation.

The OS object ID uniquely identifies the configuration array of each OS object due to SF.OSID. The OS object configuration array is read-only to the user. Each object descriptor block (DB) is a constant and cannot be modified. Each object control block (CB) has only the kernel control permission.

6.2.2 Corresponding SFRs

This function ensures that part of the fifth item of FIA_ATD.1 and part of FMT_MSA.1 is met by associating CBs with kernel privileges. By ensuring that the configuration array is read-only and that DBs cannot be modified, this function ensures that FMT_MSA.3 is met.

6.3 SF.OSAM: OS Application Management

6.3.1 Function summary

An OSA is a collection of OS objects as well as some resources (data segments and code segments), as shown in Figure 6-1. The user can configure OS objects for each OSA using the configuration tool. These OSAs use TOE functionality and are subject to the TOE security policies. The OS objects in the OSA share resources and permissions.

The configuration tool can further be used to mark OSAs as trusted or untrusted. Trusted OSAs run in the kernel privilege, whereas untrusted OSAs run in the user privilege. All attributes associated with this (OSA identity and trust level) cannot be modified once the TOE is operational.

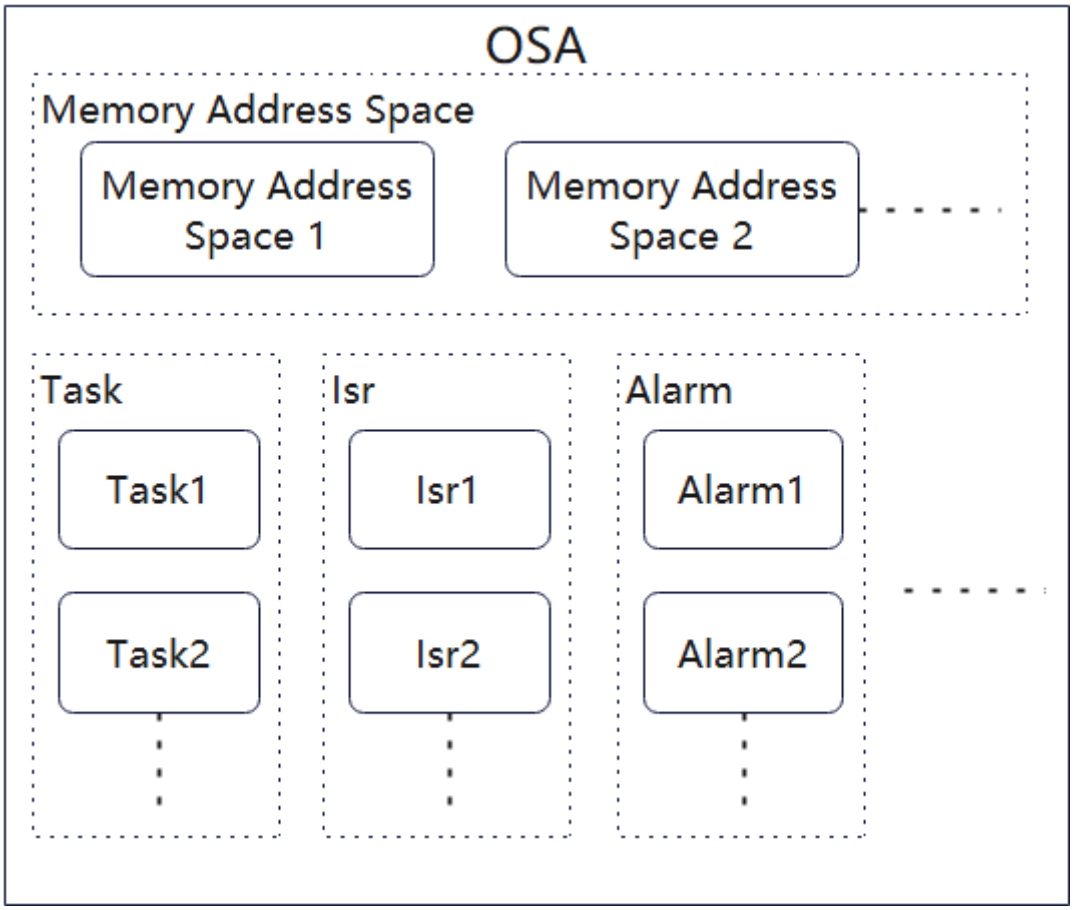

Figure 6-1 Logical block diagram of the OSA

6.3.2 Corresponding SFRs

This function ensures that the TOE meets the second and third items of FIA_ATD.1 by defining the OSA_ID and OSA_TRUST, as well as FMT_MSA.1 for those items, by preventing anyone from modifying them.

6.4 SF.OSSP: OS Service Protection

6.4.1 Function summary

Using the configuration tool, the user can configure the ACCESS_APPLICATION set of each OS object, which cannot be modified once the TOE is operational. The ACCESS_APPLICATION set is a list of all OSAs that are allowed to access this OS object. Therefore, any OS object of a corresponding OSA in this list is allowed to access the object as the subject. If the OSA to which the OS object belongs is not in the ACCESS_APPLICATION set of the object OS object, when the OSA invokes an OS service to perform an operation on the object, an error is returned and the operation is denied. As shown in Figure 6-2 below:

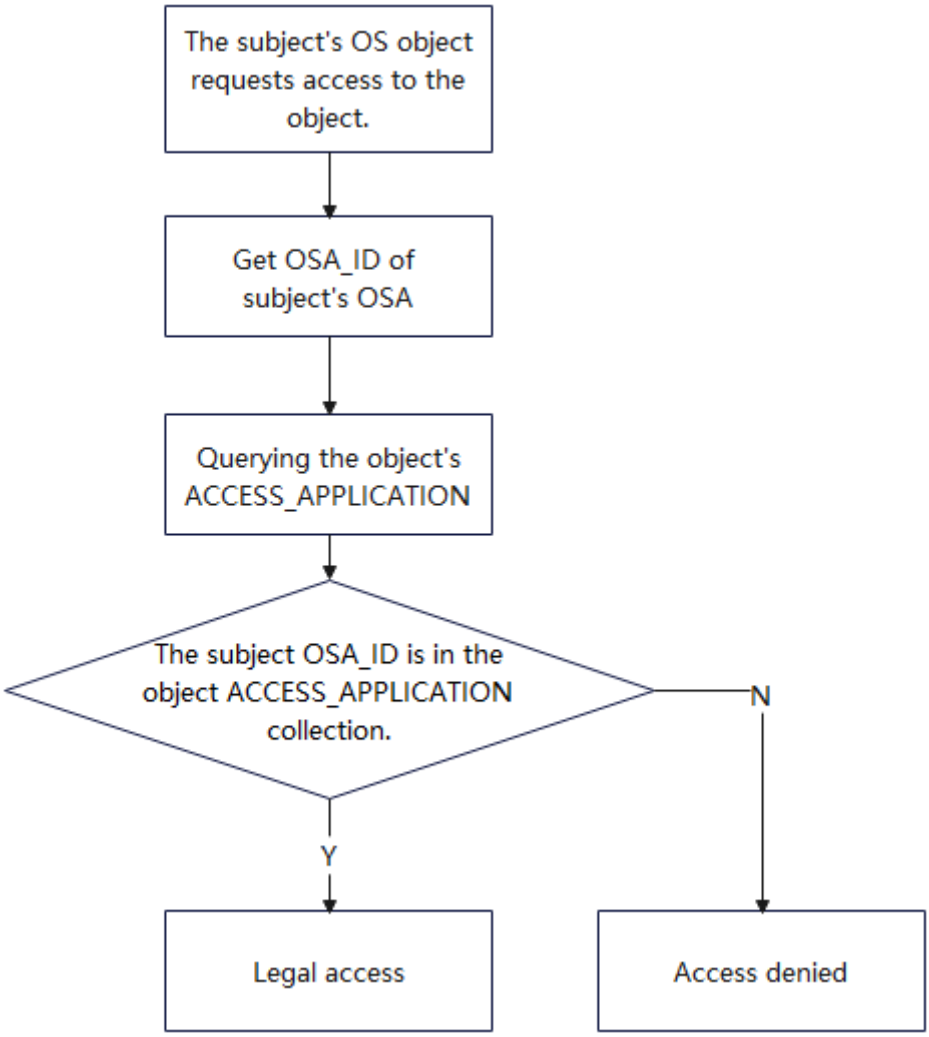

Figure 6-2 Process of accessing OS objects

Additionally, this function prevents untrusted OSAs and their objects running in the user space to access objects in the kernel space.

6.4.2 Corresponding SFRs

This function ensures that the TOE meets the fourth item of FIA_ATD.1 by defining the ACCESS_APPLICATION list, as well as the corresponding FMT_MSA.1, by preventing anyone from modifying it. It further ensures that the TOE meets FDP_ACC.1/CAP and FDP_ACF.1/CAP by enforcing the access restrictions based on the security attributes.

6.5 SF.MP: Memory Protection

6.5.1 Function summary

The TOE sets the upper and lower limits of the non-TOE HW MPU entity based on different OSA configurations (memory address space) that the user can configure using the OS configuration tool.

Based on this configuration, the non-TOE HW MPU will ensure that each OSA can access only the memory address space to which the OSA belongs, by providing responses whenever an access is requested. Based on these non-TOE HW MPU responses, the TOE grants or rejects access to memory areas for subjects.

During the running of the TOE, all data and code segments are allocated to different sections through compiler instructions. The compiler allocates each section to a specified location in the memory address space according to the linked file, as shown in Figure 6-3.

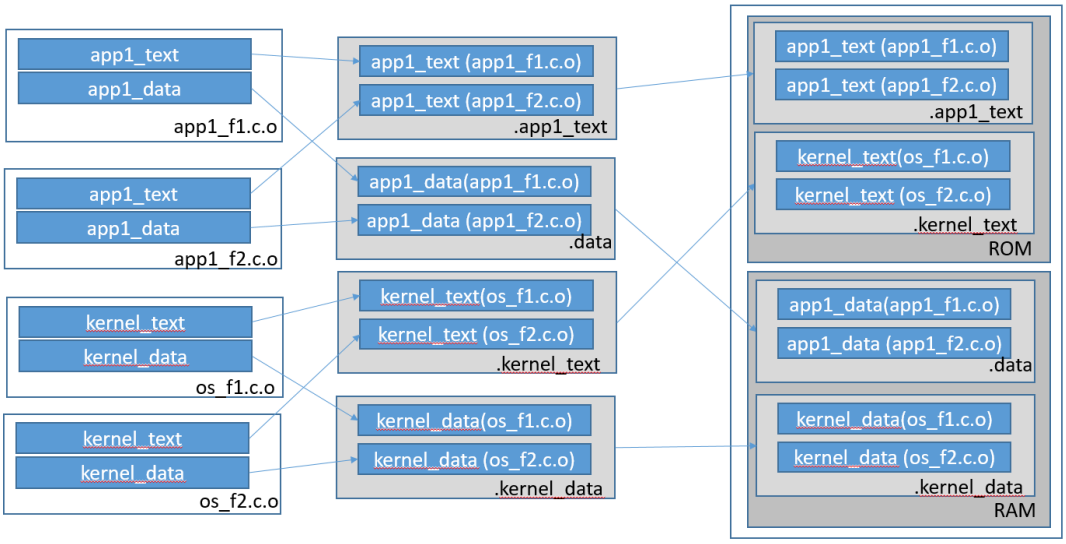

Figure 6-3 Memory allocation process

The untrusted OSA does not have read access to the RAM of the trusted OSA, and has read access to the kernel RAM（There is no user

data processed in the kernel RAM).

6.5.2 Corresponding SFRs

This function ensures that the TOE meets FDP_ACC.1/MEM, FDP_ACF.1/MEM, FMT_MSA.1 (for the MPU response), FMT_SMF.1, and FMT_SMR.1.

6.6 SF.SM: Stack Monitoring

6.6.1 Function summary

The TOE assigns a task stack to each user task. (private task stack or OSA shared task stack), where local variables required by user tasks are stored. If the task stack is too large, memory space may be wasted. If the task stack is too small, the stack operation may be overrun during task operation.

The TOE provides this security function of detecting stack overwriting, including hardware detection and software detection.

The hardware detection is based on SF.MP: When the TOE prepares to run a user task, the TOE assigns the non-TOE hardware MPU permission to the task stack. When running a task, the non-TOE hardware MPU ensures that the stack operation of the task does not cross the bounds.

The software detection: When a user task of the TOE uses the OS service, the TOE checks the stack pointer and stack top magic word. It checks whether the stack top pointer is out of bounds and whether the stack top magic word is changed. If this is the case, it will deny the access of the task and trigger exception handling.

6.6.2 Corresponding SFRs

This function ensures that the TOE meets FDP_ACC.1/STACK and FDP_ACF.1/STACK.

6.7 SF.TS: Task Scheduling

6.7.1 Function summary

The TOE implements a priority-based FIFO scheduling mechanism, where a larger value indicates a higher priority. Tasks with the same priority in the ready state are attached to the same kernel task ready-queue linked list, as shown in Figure 6-4. During scheduling, the scheduler selects the first node task in the ready list of kernel tasks, allows the task to occupy the CPU, and waits for the task to be preempted or actively releases the CPU. A user task can invoke the scheduler in the OS Service to temporarily abandon the CPU and enter the ready queue again. Then, the user task is scheduled to the end of the ready task with the same priority.

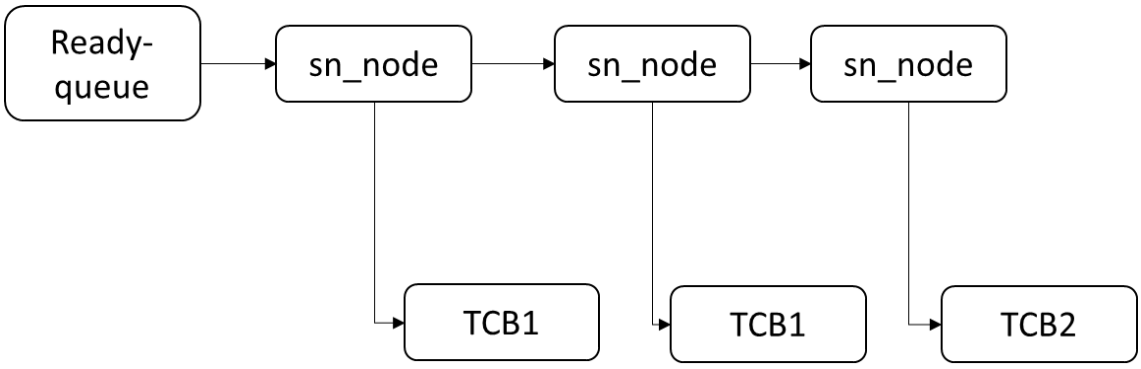

Figure 6-4 Kernel scheduling mechanism

The running priority of a task is different from the scheduling priority. If a task is configured as a non-preemptable task, the scheduling priority of the task is increased to the highest priority of all tasks after being scheduled. As a result, other tasks cannot preempt this task.

After a task obtains a resource lock, it automatically increases the priority of the resource lock to implement priority reversal, preventing other tasks sharing the resource lock from preempting the resource lock.

When a task occupies the CPU, it can actively abandon the CPU

usage by means of system calls or interrupts, and trigger re-scheduling. The presence of a higher-priority task may also interrupt a lower-priority task that is currently occupying the CPU and trigger re-scheduling.

Each time a task is activated, an sn_node is applied for and inserted into the ready-queue queue based on the priority. When the priority of the ready-queue header node task is higher than that of the current task, preemption occurs.

6.7.2 Corresponding SFRs

This security function ensures that the TOE meets FRU_PRS.1.

6.8 SF.TP: Time Protection

6.8.1 Function summary

The TOE implements a CPU resource allocation mechanism based on time. The duration of a single task is the duration of CPU usage when the user is in the running state in the suspend->ready->running->suspend or waiting->ready->running->waiting state. If the running-to-waiting or suspend state is set to 0, the single-time CPU running time is cleared. If the running state is preempted and enters the ready state, the single-time CPU running time is not cleared but is suspended. As shown in Figure 6-5 below:

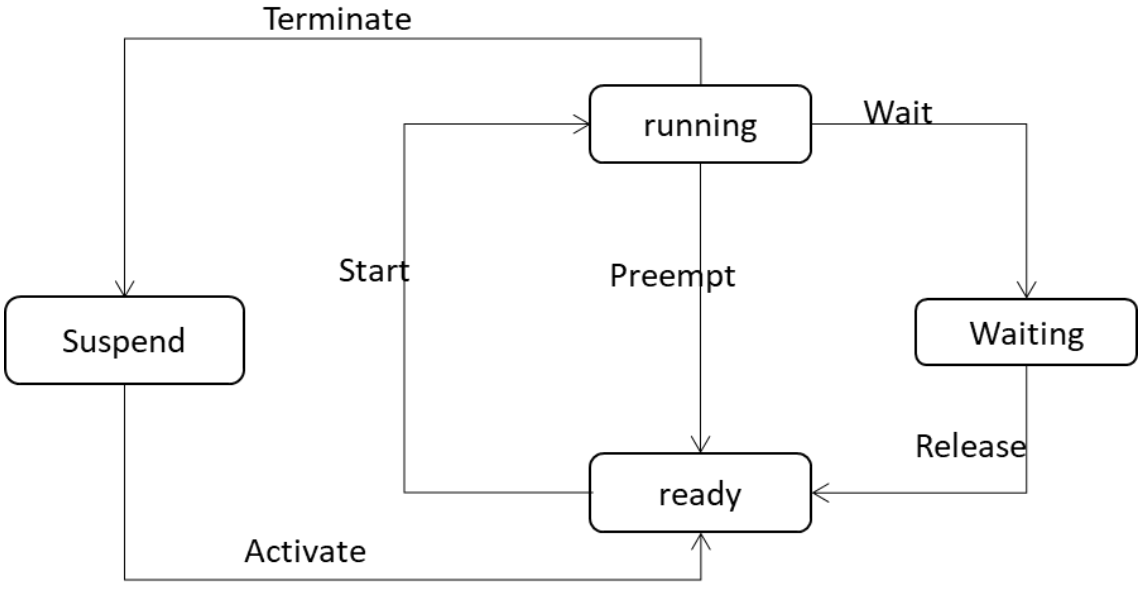

Figure 6-5 Task Transition

The user can define the maximum duration for a single task in the

OS configuration tool.

When a user task enters the running state, the TOE starts a dedicated hardware timer to count the time during which the user task can be executed. If the user task ends within the specified time, the task is cleared and the timer is stopped. Otherwise, the timer expires and an interrupt is triggered. In this case, the TOE enters the exception handling process.

6.8.2 Corresponding SFRs

This security function ensures that the TOE meets FRU_RSA.1.

6.9 SF.EH: Error Handling and Exception Handling

6.9.1 Function summary

The TOE implements error handling according to [OSEK/VDX] Section 11.2, which includes returning handling error codes, such as parameter transfer error and insufficient permissions. In addition, the TOE triggers ErrorHook, which is an optional user-defined function that can be used to handle the errors.

The TOE exception handling is implemented according to [AUTOSAR] Section 7.8. It includes the response to hardware bus exceptions, memory access exceptions, time protection exceptions, and stack overwriting exceptions. When an exception occurs, the system accesses the exception handling module and invokes the ProtectionHook function. The TOE receives the return value of the ProtectionHook function. The return value is the exception handling policy. The TOE handles exceptions based on different handling policies.

- Forcibly Ending a User Task
- Forcibly End User OSA
- Forcibly restart the operating system.

Users can implement the ProtectionHook function, add an exception handling process (including saving the site and printing logs) to the ProtectionHook function, and return an exception handling policy.

6.9.2 Corresponding SFRs

This security function ensures that the TOE meets FPT_FLS.1.

7 Rationales

7.1 Security Objective Rationale

Table 7-1 below illustrates that the Security Objectives completely address the Security Problem Definition. Furthermore, there is no security objective without a corresponding threat or assumption, which proves that every security objective is necessary.

Table 7-1 Mapping security objectives to threats and assumptions

No	Threats/Assumptions	Security Objective
1	Unauthorized access (T.UNAUTHORIZED_ACCESS)	Permission-based Access Control (O.ACCESS_CONTROL) Secure state (O.SECURE_STATE)
2	Invalid Memory Access (T.MEMORY_ACCESS)	Memory Access Control (O.MEMORY_CONTROL) Stack Monitor (O.STACK_MONITOR) Secure state (O.SECURE_STATE) Hardware platform (OE.PLATFORM)
3	System resource monopoly (T.RESOURCE_MONOPLY)	O.Priority (O.PRIORITY) Time protection control (O.TIMING_CONTROL) Secure state (O.SECURE_STATE) Hardware platform (OE.PLATFORM)
4	Hardware platform (A.PLATFORM)	Hardware platform (OE.PLATFORM)
5	Personnel (A.PERSONNEL)	Personnel (OE.PERSONNEL)

Unauthorized access (T.UNAUTHORIZED_ACCESS)

The permission-based access control (O.ACCESS_CONTROL) ensures that malicious applications or tasks cannot perform operations on OS objects through the OS Service without authorization. The TOE maintains a secure state (O.SECURE_STATE) upon access control violations,

which provides an exception handling mechanism to ensure that the system is in a secure state when an exception occurs and prevent attackers from damaging the access control mechanism.

Invalid Memory Access (T.MEMORY_ACCESS)

The memory access control (O.MEMORY_CONTROL) ensures that malicious applications or tasks cannot access TSF data and user data without authorization. Therefore, malicious applications or tasks cannot perform malicious operations on TSF data and user data, allowing them to damage data integrity or abuse the security functions of the TOE. The real-time stack monitoring (O.STACK_MONITOR) provides a stack protection mechanism for tasks to prevent memory data from being tampered with due to stack overwriting. The TOE maintains a secure state (O.SECURE_STATE) upon memory access or stack violations, which provides an exception handling mechanism to ensure that the system is in a secure state when an exception occurs and prevent attackers from damaging the access control mechanism. The hardware platform (OE.PLATFORM) helps mitigate this threat by providing the MPU.

System resource monopoly (T.RESOURCE_MONOPLY)

The priority (O.PRIORITY) policy ensures that malicious applications or tasks cannot occupy CPU resources when higher priority applications need to run, by preempting the lower priority tasks. Malicious applications and tasks cannot prevent tasks running at the same priority from using system resources, as they will be limited in their running time by the configurable time protection control (O.TIMING_CONTROL). The TOE maintains a secure state (O.SECURE_STATE) upon time limit violations, which provides an exception handling mechanism to ensure that the task time-out is properly handled. The hardware platform (OE.PLATFORM) helps mitigate this threat by providing the timer.

Hardware platform (A.PLATFORM)

This assumption for platform (security) services is directly met by the corresponding objective for the environment regarding the hardware platform (OE.PLATFORM).

Personnel (A.PERSONNEL)

This assumption is directly met by the corresponding objective for the environment regarding the personnel (OE.PERSONNEL).

7.2 Security Requirements Rationale

7.2.1 SFR tracing

Table 7-2 Security Functional Requirements

No.	SFR	Security Objective
1	Subset Access Control (FDP_ACC.1/CAP)	O.ACCESS_CONTROL
2	Subset Access Control (FDP_ACC.1/MEM)	O.MEMORY_CONTROL
3	Subset Access Control (FDP_ACC.1/STACK)	O.STACK_MONITOR
4	Security Attribute Based Access Control (FDP_ACF.1/CAP)	O.ACCESS_CONTROL
5	Security Attribute Based Access Control (FDP_ACF.1/MEM)	O.MEMORY_CONTROL
6	Security Attribute Based Access Control (FDP_ACF.1/STACK)	O.STACK_MONITOR
7	User Attribute Definition (FIA_ATD.1)	O.ACCESS_CONTROL O.MEMORY_CONTROL O.STACK_MONITOR
8	User Identification Before Any Action (FIA_UID.2)	O.ACCESS_CONTROL O.MEMORY_CONTROL O.STACK_MONITOR
9	Management of security attributes (FMT_MSA.1)	O.ACCESS_CONTROL O.MEMORY_CONTROL O.STACK_MONITOR
10	Static attribute initialisation (FMT_MSA.3)	O.ACCESS_CONTROL O.MEMORY_CONTROL O.STACK_MONITOR
11	Specification of Management Functions (FMT_SMF.1)	O.MEMORY_CONTROL
12	Security roles (FMT_SMR.1)	O.MEMORY_CONTROL
13	Fail secure (FPT_FLS.1)	O.SECURE_STATE
14	Limited priority of service (FRU_PRS.1)	O.PRIORITY

15	Maximum quotas (FRU_RSA.1)	O.TIMING_CONTROL

Permission-based Access Control (O.ACCESS_CONTROL)

This security objective is met by the SFRs FDP_ACC.1/CAP, FDP_ACF.1/CAP, FMT_MSA.3, FMT_MSA.1, FIA_ATD.1, and FIA_UID.2, that together define the permission-based access control policy and require that the security attributes related to this policy cannot be managed by subjects during runtime of the TOE.

Memory Access Control (O.MEMORY_CONTROL)

This security objective is met by the SFRs FDP_ACC.1/MEM, FDP_ACF.1/MEM, FMT_MSA.3, FMT_MSA.1, FMT_SMF.1, FMT_SMR.1, FIA_ATD.1, and FIA_UID.2, that together define the memory access control policy and require that the security attributes related to this policy cannot be managed by subjects during runtime of the TOE, with the exception of the MPU response that can be managed by the non-TOE hardware MPU subject.

Priority (O.PRIORITY)

This security objective is directly met by the SFR FRU_PRS.1, which requires the TOE to mediate access to the CPU resources based on priorities for each subject.

Time protection control (O.TIMING_CONTROL)

This security objective is directly met by the SFR FRU_RSA.1, which requires the TOE to limit access to the CPU resources based on the external non-TOE hardware timer for each subject.

Stack Monitor (O.STACK_MONITOR)

This security objective is met by the SFRs FDP_ACC.1/STACK, FDP_ACF.1/STACK, FMT_MSA.3, FMT_MSA.1, FIA_ATD.1, and FIA_UID.2, that together define the memory access control policy and require that the security attributes related to this policy cannot be managed by subjects during runtime of the TOE.

Security (O.SECURE_STATE)

This security objective is directly met by the SFR FPT_FLS.1, which requires the TOE to maintain a secure state when an error or exception occurs.

7.2.2 Dependency justification

All dependencies for security requirements are met, which are illustrated in Table 7-3. The set of SFRs including FMT_MSA.1, FMT_MSA.3, FMT_SMF.1, FMT_SMR.1, FIA_UID.2, and all iterations of FDP_ACC.1 and FDP_ACF.1 is a self-consistent set that satisfies all dependencies of its members. The SFRs FIA_ATD.1, FPT_FLS.1, FRU_PRS.1, and FRU_RSA.1 have no dependencies.

Table 7-3 Security Functional Requirements dependencies

No.	SFR	Dependencies	Met by
1	FDP_ACC.1/CAP	FDP_ACF.1	FDP_ACF.1/CAP
2	FDP_ACC.1/MEM	FDP_ACF.1	FDP_ACF.1/ MEM
3	FDP_ACC.1/STACK	FDP_ACF.1	FDP_ACF.1/ STACK
4	FDP_ACF.1/CAP	FDP_ACC.1 and FMT_MSA.3	FDP_ACC.1 and FMT_MSA.3
5	FDP_ACF.1/MEM	FDP_ACC.1 and FMT_MSA.3	FDP_ACC.1 and FMT_MSA.3
6	FDP_ACF.1/STACK	FDP_ACC.1 and FMT_MSA.3	FDP_ACC.1 and FMT_MSA.3
7	FIA_ATD.1	No dependencies	Not applicable
8	FIA_UID.2	No dependencies	Not applicable
9	FMT_MSA.1	FDP_ACC.1 or FDP_IFC.1, FMT_SMR.1 and FMT_SMF.1	FDP_ACC.1, FMT_SMF.1 and FMT_SMR.1
10	FMT_MSA.3	FMT_MSA.1 and FMT_SMR.1	FMT_MSA.1 and FMT_SMR.1
11	FMT_SMF.1	No dependencies	Not applicable
12	FMT_SMR.1	FIA_UID.1	FIA_UID.2
13	FPT_FLS.1	No dependencies	Not applicable
14	FRU_PRS.1	No dependencies	Not applicable
15	FRU_RSA.1	No dependencies	Not applicable

With regards to the assurance requirements, the package EAL4 meets all dependencies by design, the dependencies of the augmentation AVA_VAN.5 are all contained in EAL4, whereas the augmentation ALC_FLR.1 has no dependencies.

7.2.3 Chosen SARs

The package EAL4 is chosen to gain maximum assurance from positive security engineering based on good commercial development practices. The augmentation of AVA_VAN.5 is chosen to extend the required security level to the highest covered by the standard. The augmentation of ALC_FLR.1 is chosen to show that the developer implements security flaw remediation for the TOE and provides the associated information to TOE users.

8 References

[AUTOSAR]	Specification of Operating System, AUTOSAR CP, Release 4.4.0, 2018-10-31
[OSEK/VDX]	OSEK/VDX, Operating System, Version 2.2.3, 2005-02-17