Security Target

PikeOS Separation Kernel v4.2.3

Document ID	Revision	DOORS Baseline	Date	State
00101-8000-ST	20.7	N.A.	2019-01-21	App

Author: Dominic Eschweiler

SYSGO AG

Am Pfaffenstein 14, D-55270 Klein-Winternheim

This page intentionally left blank

This page intentionally left blank

SYSGO AG

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Purpose of this Document

This is the Security Target for the *PikeOS Separation Kernel*.

1.2 Document References

1.2.1 Applicable Documents

Ref.	Document ID - Document Title	Version
[Com12]	Common Criteria Sponsoring Organizations, Common Criteria for Information Technology Security Evaluation. Version 3.1, revision 4 (final), September 2012, http://www.commoncriteriaportal.org/thecc.html.	3.1, revision 4

Table 1: Applicable Documents

1.2.2 Referenced Documents

Ref.	Document ID - Document Title
[ANSSI15]	Agence nationale de la sécurité des systèmes d'information, Référentiel général de sécurité, Processus de qualification d'un produit de sécurité - niveau standard - version 1.2, 2015, https://www.ssi.gouv.fr/uploads/2015/07/RGS_qualif_standard_version_1-2.pdf.
[Bun08]	Bundesamt für Sicherheit in der Informationstechnik (BSI) and Sirrix AG security technologies, Protection Profile for High-Assurance Security Kernel: Version 1.14, June 2008, http://web.archive.org/web/*/http://www.sirrix.com/media/downloads/54500.pdf.
[Inf07]	Information Assurance Directorate, U.S. Government Protection Profile for Separation Kernels in Environments Requiring High Robustness. Version 1.03, June 2007, https://web.archive.org/web/20110108022547/http://www.niap-ccevs.org/cc-scheme/pp/pp_skpp_hr_v1.03/.
[LNIM10]	Timothy E. Levin, Thuy D. Nguyen, Cynthia E. Irvine, Michael McEvilley, Separation Kernel Protection Profile revisited: Choices and rationale, 4th Annual Layered Assurance Workshop (LAW), 2010, http://fm.csl.sri.com/LAW/2010/.
[OSPP]	Stephan Mueller, Gerald Krummeck, Helmut Kurth, Operating System Protection Profile, 2010, https://www.commoncriteriaportal.org/files/ppfiles/pp0067b_pdf.pdf.

Table 2: Referenced Documents

1.3 Abbreviations and Acronyms

Abbreviation / Acronym	Description
APEX	Application Executive
API	Application Programming Interface
ARINC	Aeronautical Radio, Inc.

ASP	Architecture Support Package
CC	Common Criteria for Information Technology Security Evaluation
CPU	Central Processing Unit
DMA	Direct Memory Access
EAL	Evaluation Assurance Level
ELF	Executable and Linkable Format
HASK	High-Assurance Security Kernel
I/O MMU	Input / Output Memory Management Unit
I/O	Input / Output
IPC	Interprocess Communication
IT	Information Technology
MILS	Multiple Independent Levels of Security
MMU	Memory Management Unit
OSEK	Offene Systeme und deren Schnittstellen für die Elektronik in Kraftfahrzeugen
OSP	Organizational Security Policy
OSPP	Operating Systems Protection Profile
POSIX	Portable Operating System Interface
PSP	Platform Support Package
RAM	Random Access Memory
RTEMS	Real-Time Executive for Multiprocessor Systems
SAR	Security Assurance Requirement
SFP	Security Function Policy
SFR	Security Functional Requirement
SKPP	Separation Kernel Protection Profile
SSP	System Security Policy
ST	Security Target
TOE	Target of Evaluation
TSF	Target of Evaluation Security Functionality
TSS_XXX	TOE Security Service XXX
USB	Universal Serial Bus
VMIT	Virtual Machine Initialization Table
XML	Extensible Markup Language

Table 3: Abbreviations and Acronyms

1.4 Terms and Definitions

Access Flag: An *access flag* is bit specifying whether a certain access operation is allowed.

Access Mode: An *access mode* is a set of access flags.

Application: An *application* is executable data. An application is a special case of "Executable".

Architecture Support Package: *An architecture support package (ASP) is part of the PikeOS Separation Kernel*. The ASP provides specific low-level functionality for each supported CPU architecture. Since the CPU instruction set is also CPU dependent, the generic components are CPU specific at the object code level.

The main responsibilities of an ASP are: (1) abstraction of data type representation, (2) processor exception handling, and (3) low level address space and memory management.

In operational use, the TSF always contains only one ASP.

Bootloader: See "Firmware".

Builtin File Provider: A *builtin file provider* is an executable provided by the TOE that implements file services.

Communication Object: Partitions can communicate with each other under the supervision of the *PikeOS Separation Kernel*. A *communication object* is an object exposed to one or multiple partitions with access rights as defined in the configuration data.

Configuration Data: The *configuration data* defines a set of rules on how the TOE behaves. The configuration data comprises a top-level configuration XML document called VMIT (Virtual Machine Initialization Table) and the property file system. Access to each property file system node is controlled by the file access control defined in the VMIT. The configuration data is defined during Step 3 of the system integration phase and used by the TOE to enforce the System Security Policy (SSP, Section 2.4.4.2). The default configuration is that there is no communication between any partitions. Any communication between partitions has to be explicitly allowed by the integrator in the configuration data.

CPU Architecture: The *CPU architecture* is characterised by design implementing the CPU instruction set. PikeOS has been evaluated for the CPU architectures x86 64-bit, ARMv7, or ARMv8.

Cyclic Periodicity: *Cyclic periodicity* is the repetition of a scheduling scheme: when the last time window of the scheduling scheme has finished, the scheduling scheme begins again with its first time window.

ELF File: An *ELF file* is a file in ELF format, used to load applications.

Executable: "Executable" is the term for an application within a partition or other executable code linked to PikeOS.

- An executable is *critical* if it can have access to critical hardware. Critical hardware can bypass PikeOS partitioning.

  Examples where a normal partition hosts a critical executable are:
  - An executable in a normal partition is critical if the integrator in the configuration assigns to it access to control a critical hardware which can do DMA, there is no I/O MMU in place, and the hardware can write to arbitrary system memory, thus tampering with PikeOS. Another example is an executable that has sufficient control over a hardware device allowing the executable to tamper with the device's BIOS or non-volatile memory. We refer the reader to the TOE User Manuals for more details.
- A *non-privileged executable* is an application in a normal partition that is not critical.

  Thus, a non-privileged executable can be an arbitrary executable, which can contain unintentional errors or be even malicious.

  A *privileged executable* is either: A critical application in a normal partition, or any application in a system partition, or the PSP, or a kernel device driver, or a system extension. That is, a

privileged executable has access to PikeOS API or to critical hardware that can bypass PikeOS partitioning. Therefore, a privileged executable must be non-evil, and the integrator must verify that it complies with the system design as well as that it will not interfere with PikeOS and the security policies defined in the VMIT.

External File Provider: An *external file provider* is an executable provided by third-party developers that implements file services and is confined by VMIT.

File Descriptor: A *file descriptor* is an identifier for a file.

Firmware: *Firmware* is hardware-specific software which comprises the following:

- Software and data stored in non-volatile memory of the hardware that initializes the hardware after the power on.
- Software that (fully or partially) loads the TOE into RAM memory and hands over the full control to the TOE. In particular, a TOE-external check of the TOE may be implemented in the bootloader (e.g. for “secure boot”).

Fusion Tool Chain: The PikeOS *fusion tool chain project* is a linker that links one or several executables and the *PikeOS Separation Kernel* for the x86 64-bit, ARMv7, or ARMv8 architectures. The tool-chain provides linker for the required CPU architectures, i.e. x86 64-bit, ARMv7, or ARMv8. The tool chain can be used on a development machine with Linux or Windows.

Hardware: *Hardware* is the physical part of the TOE operational environment on which the TOE is executed. Usually, hardware is a board with several components such as CPUs and I/O devices (e.g. serial interfaces, network adapters) etc. This ST considers firmware as part of the hardware.

Integration Tool Chain: The PikeOS *integration tool chain project* takes as input one or more applications, the VMIT, the property file system, and the results of the fusion tool chain and produces the product based on PikeOS, in the form of a ROM image. The tool-chain supports the required CPU architectures, i.e. x86 64-bit, ARMv7, or ARMv8. The tool chain can be used on a development machine with Linux or Windows.

Integrator: The *integrator* executes the system integration phase, which results in a product based on the TOE.

Interprocess Communication: *IPC* is a communication protocol in PikeOS to exchange messages within a partition or between partitions synchronously. The communication objects are IPC messages.

Isolation: *Isolation* of a partition is the absence of communication with other partitions, except partitions hosting the components implementing the system API, when no communication channels or shared resources between the partition and other partitions are configured. Isolation is a special case of separation.

Kernel Info Page: The *kernel info page* is a memory page that is mapped in read-only access mode to all partitions.

Kernel Device Driver: A *kernel device driver* is a software component supplied and approved by the integrator and linked with the *PikeOS Separation Kernel* via the kernel device driver API. A kernel device driver can provide specific functionality to applications within partitions and is protected from non-privileged executables by access control and resource management enforced by the *PikeOS Separation Kernel*.

Life Cycle: The *life cycle* phases for this TOE are development (source code development), manufacturing (compilation to binary) and delivery to the integrator. The *integrator* executes the system integration phase, which results in a product based on the TOE.

Memory Page: A *memory page* is an aligned and contiguous area of memory of a CPU architecture dependent size (e.g. 4096 bytes).

Normal Partition: A *normal partition* can use the API provided by PikeOS to partitions that only consists of non-privileged calls (“normal partition API”).

Non-Privileged Executable: See “Executable”.

Own (for a Task or a Thread): A partition *owns* a task if the task is assigned to it by the integrator in the VMIT. A partition owns a thread if the thread is created by one of its applications.

Partition: A *partition* is a logical unit maintained by the PikeOS Separation Kernel and configured by the SSP. A *partition* contains user data. For each partition, the PikeOS Separation Kernel provides resources. Resources of a partition comprise physical memory and allocated CPU time for each CPU.

Partition Switch: *Partition switches* are defined by SSP as part of the scheduling scheme and assign CPU(s) to another partition.

PikeOS Operating System: The *PikeOS Operating System* consists of the PikeOS Separation Kernel and additional system components such as the PSP, (kernel) device drivers, and system extensions. Such additional components are usually used for making the *PikeOS Operating System* fully compatible with its use-case and the used hardware.

PikeOS Separation Kernel: The *PikeOS Separation Kernel* provides the TSF and operates the product based on PikeOS. The TSF implements mechanisms to assign resources to partitions, providing the execution environments for applications, and implementing communication between partitions as defined by the configuration data. The PikeOS Separation Kernel is provided as binary for each CPU architecture, i.e. one instance for x86 64-bit, ARMv7, or ARMv8. The functionality of the PikeOS Separation Kernel is CPU architecture-independent thanks to the usage of the ASP that abstracts the underlying CPU architecture.

PikeOS: The term *PikeOS* is usually used in the documentation when the described system includes at least the PikeOS Separation Kernel. The terms *PikeOS* and PikeOS Separation Kernel are used synonymously in this ST.

Platform Support Package: A *platform support package* (PSP) contains a set of drivers for specific hardware components and is supplied and approved by the integrator. A PSP uses the TSF's PSP API. In operational use, the product based on PikeOS always contains exactly one PSP. A PSP is protected from non-privileged executables by access control and resource management enforced by the PikeOS Separation Kernel. The main tasks of a PSP are (1) platform initialization, (2) interrupt management, (3) hardware timer management.

Port: A *port* is a communication endpoint for message based communication. A port is either configured as source port or destination port. A sender writes a message to a source port and a receiver reads the message from the connected destination port. PikeOS supports two kinds of message transfer modes - queuing mode and sampling mode. The mode is defined by the port type. Only ports of the same type can be connected. Messages sent to a source queuing port are buffered in a queue until they are read from the connected destination port. The number of messages that can be buffered is a configurable property of a queuing port. A message sent to a source sampling port is only buffered until the next message is sent, the new message will replace the old one.

Privileged Executable: See "Executable".

Product Based on PikeOS: The *product based on PikeOS* is the output of the system integration phase (Section 2.4.4.2). The product based on PikeOS contains the configuration data in a representation readable by the PikeOS Separation Kernel, the PSP, system extensions, kernel device drivers, and partitions. PikeOS reads and set up the configuration, and confines non-privileged executables within their partitions according the SSP.

Property Node: A *property node* is a file in the property file system. The integrator can use the property file system to describe properties of hardware in the operational environment.

Resource: In this ST we consider *resources* of partitions and TSF data. The resources of a partition comprise physical memory and allocated CPU time for each CPU.

Separation: The TOE *separates* partitions by managing their accesses to and usage of resources, such as memory, devices, processors, and communication channels, as defined by the configuration.

Subject: In this ST the term *subject* is used for a thread in a partition, for an application, or for a partition as a whole depending on the context. In the running TOE the entity executing application code is a PikeOS thread. In a partition there can be multiple threads and each thread is uniquely assigned to its partition. Since this ST works on security policies at partition level, we abstract all

threads in one partition to the partition subject.

System Extension: A *system extension* is a software component supplied and approved by the integrator and linked with the PikeOS Separation Kernel via the system extension API. A system extension can provide specific functionality to applications within partitions and is protected from non-privileged executables by access control and resource management enforced by the PikeOS Separation Kernel.

System Partition: A *system partition* can use the whole API provided by PikeOS to partitions ("system partition API"), i.e. use both non-privileged (normal partition API) and additional privileged calls. The integrator can define a partition as a system partition in the PikeOS configuration.

System Security Policy (SSP): The configuration data uniquely defines the *System Security Policy* (SSP) consisting of configuration choices made by an integrator. The SSP defines partitions, setting their resources, defining communication objects, defining access to and parameters for PSP, system extensions, and kernel device drivers, setting their content and resources, setting *PikeOS Separation Kernel* attributes, comprising scheduling scheme, policy for memory cache handling on a partition switch to the extent supported by the operational environment's hardware, scheme for automatic handling of error conditions. The SSP is a subset of the VMIT.

Time Window: A *time window* is the basic scheduling entity of CPU time assigned to a partition that is specified by the offset from the start of a cyclical time period and the window duration.

TOE Security Service: A *TOE Security Service* is a logical part of the TOE that has to be relied upon for enforcing a related subset of the rules regulating how the SSP is maintained by the TOE.

TOE User Manuals: The *TOE User Manuals* are documentation provided with the TOE on how to use the TOE in general environments and in safety and security critical environments.

User: A "user" of the TOE is a partition.

Virtual Machine Initialization Table (VMIT): The *configuration data* defines a set of rules on how the TOE behaves. The configuration data comprises a top-level configuration XML document called VMIT (Virtual Machine Initialization Table) and the property file system. Access to each property file system node is controlled by the file access control defined in the VMIT. The SSP is a subset of the VMIT.

1.5 Imperative Terms

Use of the words **'shall', 'should', 'may'** and '**will**' within this document observe the following rules:

The word **'shall'** in a text expresses a mandatory definition.

The words **'should'** and '**may**' in a text express a non mandatory definition. 'should' is used, when a non mandatory provision is recommended, otherwise 'may' is used.

The word **'will'** in a text expresses a definition in cases a simple futurity is required. 'will' is also used to express a task (done by an individual or an organization), which is not controlled by this document.

2 ST Introduction

This is the Security Target for the *PikeOS Separation Kernel* also called PikeOS in this document..

2.1 ST Reference

- Title: Security Target PikeOS Separation Kernel v4.2.3
- Version: 20.7
- Issuer: SYSGO AG
- Keywords: Operating system, Separation kernel, MILS (Multiple Independent Levels of Security), Virtualization, Hypervisor
- Date: 21. January 2019
- TOE Version: 4.2.3

This document is the Security Target for the Common Criteria evaluation of the *PikeOS Separation Kernel*. It complies with the Common Criteria for Information Technology Security Evaluation Version 3.1 Revision 4 [Com12] (CC).

2.2 TOE Reference

The TOE is the *PikeOS Separation Kernel* version 4.2.3 running on the microprocessor family (x86 64-bit, ARMv7, or ARMv8) hosting different applications. The TOE is referenced as PikeOS 4.2.3 base product build S5577 for Linux and Windows development host with PikeOS 4.2.3 Certification Kit build S5577.

2.3 TOE Overview

2.3.1 Usage and Major Security Services of the TOE

The TOE is a special kind of microkernel, a separation kernel, which allows to effectively separate different applications running on the same platform from each other. Applications are hosted in partitions that can be separated from each other. Such non-privileged applications can also be operating systems. Non-privileged applications may be malicious, and even in that case the TOE ensures that the malicious applications are harming neither the TOE nor other applications in other partitions. The TOE will be installed and run on a hardware platform (e.g. embedded systems).

SYSGO defines *separation* as follows: The TOE *separates* partitions by managing their accesses to and usage of resources, such as memory, devices, processors, and communication channels, as defined by the configuration. *Isolation* of a partition is the absence of communication with other partitions, except partitions hosting the components implementing the system API, when no communication channels or shared resources between the partition and other partitions are configured. Isolation is a special case of separation.

The TOE is a real time separation kernel, thus, the partitioning is configured statically and the TOE does not include typical desktop operating system services (e.g. user login, printer drivers).

The major security services provided by the TOE are:

- Separation in space of applications hosted in different partitions from each other and from the *PikeOS Operating System* according to the configuration data,
- Separation in time of applications hosted in different partitions from each other and from the *PikeOS Operating System* according to the configuration data,
- Control of information flows between applications hosted in different partitions via assigning to the partitions communication objects and access rights to those,

- Management of the TOE (e.g. system partition API) and the TOE data (e.g. threads, tasks).

2.3.2 TOE Type

The TOE is a separation kernel with real-time support.

The typical *life cycle* phases for this TOE type are development (source code development), manufacturing (compilation to binary) and delivery to the integrator. The *integrator* executes the system integration phase, which results in a product based on the TOE.

The TOE may run on various hardware platforms. The hardware platform is not part of the TOE. The minimum requirements on the hardware platform comprise a CPU with a memory management unit (MMU) and support for different CPU privilege modes as well as having a suitable Platform Support Package (PSP). The supported CPU architectures are x86 64-bit, ARMv7, or ARMv8.

2.4 TOE Description

2.4.1 TOE Architecture

The TOE, including its architecture, and the TOE operational environment is depicted in Figure 1.

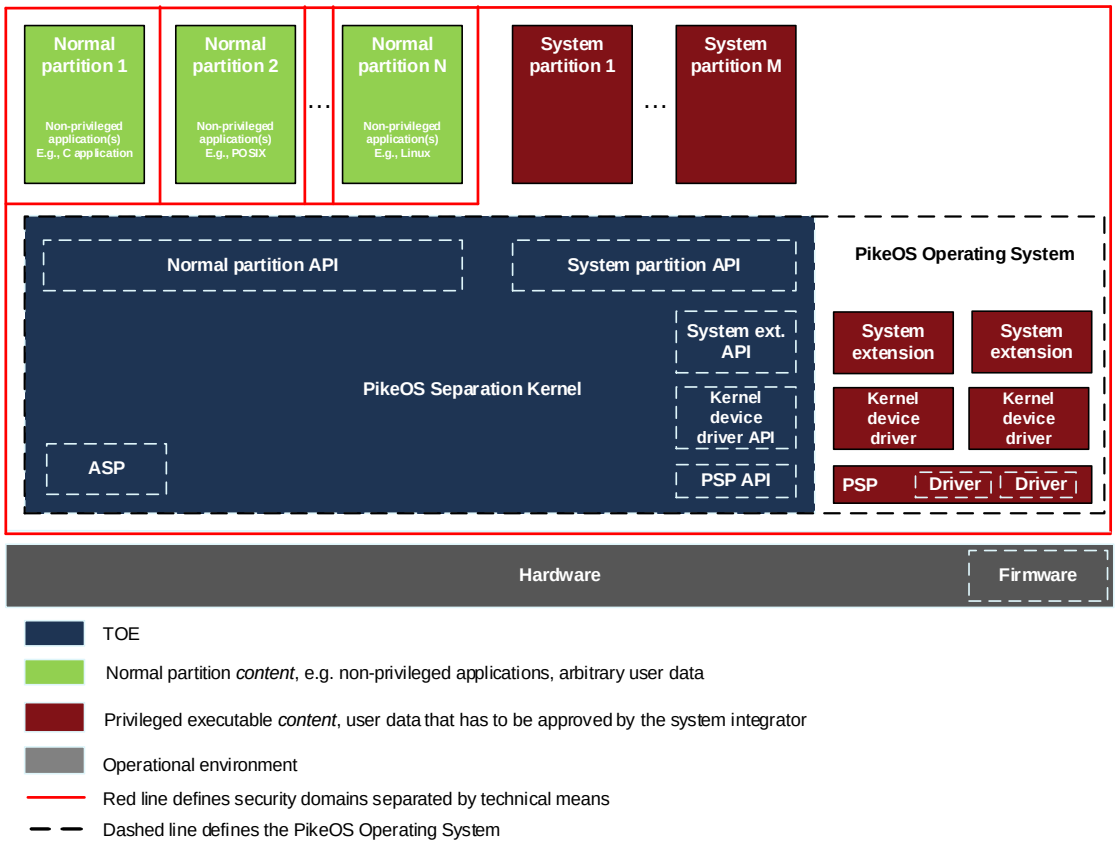

Figure 1: TOE and TOE Operational Environment during Operational Use

Figure 1 will be explained in detail in the next sections (Section 2.4.2 and 2.4.3).

2.4.2 TOE

The TOE is the *PikeOS Separation Kernel*, shown in Figure 1 by the blue box. It consists of the PikeOS Kernel and System Software, which contains all security functions (TSF) claimed in this ST, and the TSF data (e.g. configuration data and run-time data such as security attributes) necessary to configure and control them.

The *PikeOS Separation Kernel* provides the TSF and operates the product based on PikeOS. The TSF implements mechanisms to assign resources to partitions, providing the execution environments for applications, and implementing communication between partitions as defined by the configuration data. The separation kernel has the following interfaces (depicted as dashed boxes within the blue box in Figure 1):

- The *PikeOS Separation Kernel* provides APIs to normal partitions and system partitions as well as APIs to system extensions, kernel device drivers and the platform support package (PSP).
- An *architecture support package* (ASP) is part of the *PikeOS Separation Kernel*. The ASP provides specific low-level functionality for each supported CPU architecture. Supported CPU architectures are x86 64-bit, ARMv7, or ARMv8. In operational use, the TSF always contains only one ASP.

TSF data consists of:

- Configuration data: data used by the TOE to enforce the System Security Policy (SSP, Section 2.4.4.2).
- Run-time data such as security attributes.

2.4.2.1 PikeOS Operating System

The *PikeOS Operating System* consists of the *PikeOS Separation Kernel* and, depending on the configuration used, of additional software components as mentioned in Section 2.4.3. Such components are depicted in Figure 1 by the red boxes on the right side.

2.4.3 TOE Operational Environment

In a final product, e.g., an embedded system product based on the TOE, the TOE will use additional supporting software components to adapt the product to its operational environment, including the system's hardware. These additional components are provided by the integrator and are not covered by this evaluation and belong therefore to the TOE's operational environment. They interact with the PikeOS TOE through the TOE's well-defined interfaces and do not change the TOE binary. Some of the components execute privileged CPU instructions or use privileged PikeOS services and could therefore interfere with security policies enforced by the TSF at runtime. Those components are collectively referred to as *privileged executables* (see Section 2.4.3.1.3). It is outside the scope of this evaluation how the trust for the privileged executables will be established. The integrator of a product based on the TOE must establish trust for these privileged executables, e.g. the integrator may use a national scheme's accreditation for components and/or the product based on the TOE, a Common Criteria evaluation of the product based on the TOE, or the integrator may provide some other arguments to the user of the product based on the TOE.

The TOE's operational environment consists of everything outside the blue box in Figure 1. The various privileged (red) or non-privileged (green) components that may be added by the system or platform integrators are the following:

2.4.3.1 Partition

A *partition* is a logical unit maintained by the *PikeOS Separation Kernel* and configured by the configuration data. A partition contains user data. For each partition, the *PikeOS Separation Kernel* provides resources. *Resources* of a partition comprise physical memory and allocated CPU time for each CPU.

PikeOS supports two different kinds of partitions: normal and system partitions. Normal partitions, depicted as green boxes in Figure 1, are defined in Section 2.4.3.1.1. System partitions, depicted as red boxes in Figure 1, are defined in Section 2.4.3.1.2.

Partitions can communicate with each other under the supervision of the *PikeOS Separation Kernel*. This communication occurs via *communication objects.* A *communication object* is an object exposed to one or multiple partitions with access rights as defined in the configuration data.

2.4.3.1.1 Normal Partition

A *normal partition* can use the API provided by PikeOS to partitions that only consists of non-privileged calls (“normal partition API”). The integrator can define a partition as a normal partition in the PikeOS configuration.

2.4.3.1.2 System Partition

A *system partition* can use the whole API provided by PikeOS to partitions (“system partition API”), i.e. use both non-privileged (normal partition API) and additional privileged calls. The integrator can define a partition as a system partition in the PikeOS configuration.

2.4.3.1.3 Executable, Non-privileged Executable, Privileged Executable

“Executable” is the term for an application within a partition or other executable code linked to PikeOS.

- An executable is *critical* if it can have access to critical hardware. Critical hardware can bypass PikeOS partitioning.
  Examples where a normal partition hosts a critical executable are:
  - An executable in a normal partition is critical if the integrator in the configuration assigns to it access to control a critical hardware which can do DMA, there is no I/O MMU in place, and the hardware can write to arbitrary system memory, thus tampering with PikeOS. Another example is an executable that has sufficient control over a hardware device allowing the executable to tamper with the device’s BIOS or non-volatile memory. We refer the reader to the TOE User Manuals for more details.
- A *non-privileged executable* is an application in a normal partition that is not critical.
  Thus, a non-privileged executable can be an arbitrary executable, which can contain unintentional errors or be even malicious.
- A *privileged executable* is either: A critical application in a normal partition, or any application in a system partition, or the PSP, or a kernel device driver, or a system extension. That is, a privileged executable has access to PikeOS API or to critical hardware that can bypass PikeOS partitioning. Therefore, a privileged executable must be non-evil, and the integrator must verify that it complies with the system design as well as that it will not interfere with PikeOS and the security policies defined in the VMIT.

2.4.3.2 System Extension

A *system extension* is a software component supplied and approved by the integrator and linked with the *PikeOS Separation Kernel* via the system extension API. A system extension can provide specific functionality to applications within partitions and is protected from non-privileged executables by access control and resource management enforced by the *PikeOS Separation Kernel*.

2.4.3.3 Kernel Device Driver

A *kernel device driver* is a software component supplied and approved by the integrator and linked with the *PikeOS Separation Kernel* via the kernel device driver API. A kernel device driver can provide specific functionality to applications within partitions and is protected from non-privileged executables by access control and resource management enforced by the *PikeOS Separation Kernel*.

2.4.3.4 Platform Support Package (PSP)

A *platform support package* (PSP) contains a set of drivers for specific hardware components and is supplied and approved by the integrator. A PSP uses the TSF’s PSP API. In operational use, the product based on PikeOS always contains exactly one PSP. A PSP is protected from non-privileged

executables by access control and resource management enforced by the *PikeOS Separation Kernel*.

2.4.3.5 Communication Object

A *communication object* is a file, shared memory, a port, an IPC message, or an event.

2.4.3.6 Hardware

Hardware is the physical part of the TOE operational environment on which the TOE is executed. Usually, hardware is a board with several components such as CPUs and I/O devices (e.g. serial interfaces, network adapters) etc.

Hardware may also comprise firmware. *Firmware* is hardware-specific software which comprises the following:

- Software and data stored in non-volatile memory of the hardware that initializes the hardware after the power on.
- Software that (fully or partially) loads the TOE into RAM memory and hands over the full control to the TOE. In particular, a TOE-external check of the TOE may be implemented in the bootloader (e.g. for “secure boot”).

2.4.4 TOE Life Cycle

2.4.4.1 TOE Development, TOE Manufacturing

At the TOE manufacturer’s site (SYSGO), the TSF is developed (source code development), and manufactured (compiled to binary). The TOE manufacturer also produces the TOE User Manuals.

2.4.4.2 System Integration

At the integrator’s site, the TOE is integrated. Figure 2 presents the system integration phase of the TOE lifecycle. Components used to build the product based on the TOE are provided by different sources: application developers, integrators, and the TOE manufacturer (SYSGO). During system integration, the TOE, applications and other executables only have to be linked, i.e. their implementations do not need to be changed or recompiled.

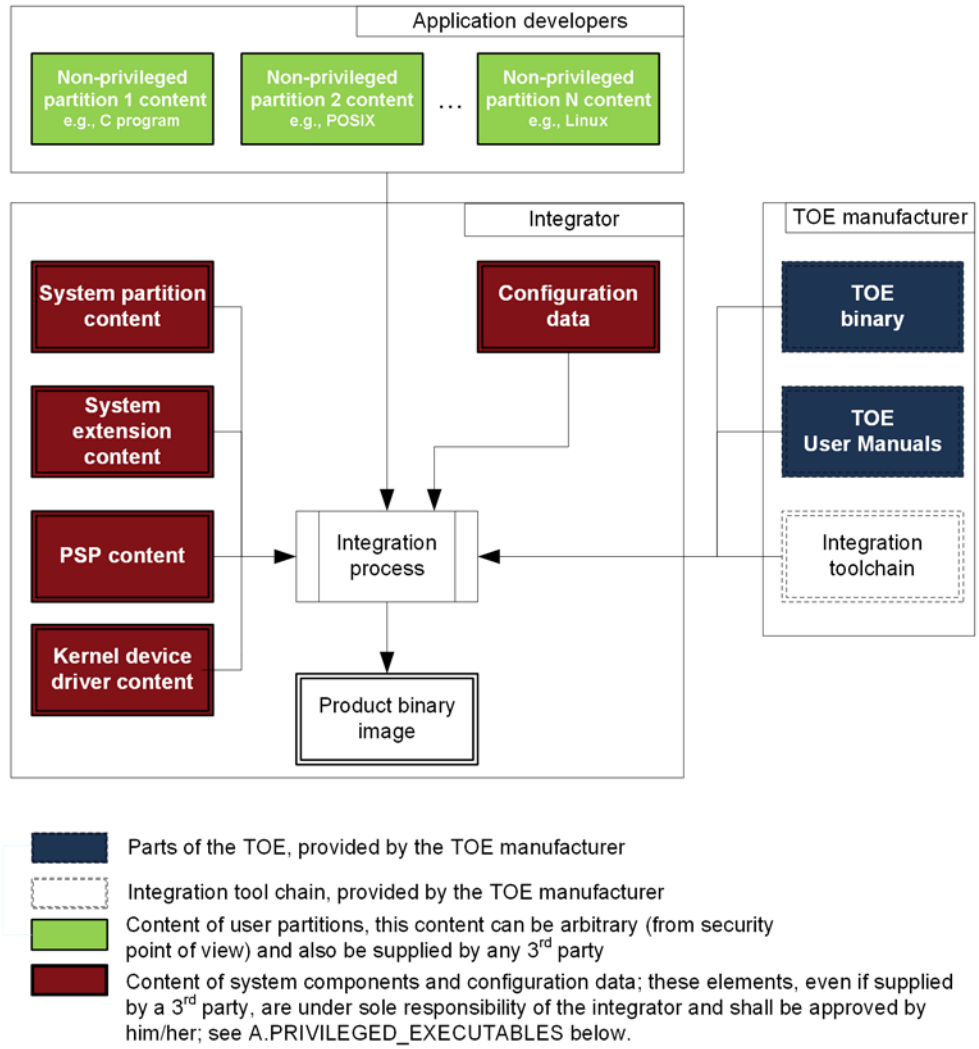

Figure 2: System Integration Phase of the TOE Lifecycle

The system integration phase can be split into the four steps: selection of the TOE operational environment, non-privileged and privileged executables, and deciding on system partitions and normal partitions (Step 1), fusion (Step 2), configuration of the TOE (Step 3), and integration (Step 4).

Step 1 Selection

The integrator selects hardware, and if applicable, firmware the TOE runs on.

The integrator installs the PikeOS 4.2.3 product and the PikeOS 4.2.3 Certification Kit on the development machine for the corresponding CPU architecture.

The integrator selects the content of components: PSP, optional system extension(s), optional kernel device driver(s), optional system partition(s), and normal partition(s) to be integrated in the TOE.

The content of any privileged executable shall be developed complying with the obligations given in Section 4.4 and be approved by the integrator.

Step 2 Fusion

In this step the integrator:

- provides configuration for the PSP, kernel device drivers, and system extensions, and

- uses the fusion tool chain to link the PSP, kernel device drivers, and system extensions with the *PikeOS Separation Kernel* binary image.

Step 3 Configuration

The integrator configures the product by:

- defining normal and system partitions, setting their content and resources,
- defining communication objects,
- defining access to and parameters for PSP, system extensions, and kernel device drivers, setting their content and resources,
- setting *PikeOS Separation Kernel* attributes, comprising:
  - scheduling scheme,
  - policy for memory cache handling on a partition switch to the extent supported by the operational environment's hardware,
  - scheme for automatic handling of error conditions, defining the meaning of the safe and secure state.

The result of this activity is the configuration data. The *configuration data* defines a set of rules on how the TOE behaves. The configuration data comprises a top-level configuration XML document called VMIT (Virtual Machine Initialization Table) and the property file system. Access to each property file system node is controlled by the file access control defined in the VMIT.

The configuration data uniquely defines the *System Security Policy* (SSP) consisting of configuration choices made by an integrator. The SSP defines partitions, setting their resources, defining communication objects, defining access to and parameters for PSP, system extensions, and kernel device drivers, setting their content and resources, setting *PikeOS Separation Kernel* attributes, comprising scheduling scheme, policy for memory cache handling on a partition switch to the extent supported by the operational environment's hardware, scheme for automatic handling of error conditions. The SSP is a subset of the VMIT.

The SSP is enforced by the TSF and it cannot be circumvented by non-privileged executables. The default configuration is that there is no communication between any partitions. Any communication between partitions has to be explicitly allowed by the integrator in the configuration data.

Step 4 Integration

The integrator uses the integration tool chain to create a product based on PikeOS (in the form of a binary image) from the selected components and the PikeOS configuration data, creating the *product based on PikeOS*, including configuration data in a representation readable by the *PikeOS Separation Kernel*.

2.4.4.3 Operational Use of the Product Based on the TOE

At power on the hardware is initialized and then the product based on PikeOS is loaded. Immediately after the product based on PikeOS has been loaded, the PSP gets executed. The PSP then starts the PikeOS separation kernel (TSF), the *PikeOS Separation Kernel* initializes itself and starts enforcing the SSP, i.e. PikeOS reads and set up the configuration, and confines non-privileged executables within their partitions according the SSP.

2.4.5 TOE Physical Boundary

The TOE is the *PikeOS Separation Kernel*. In Figure 1, each component within the blue box is within the TOE physical boundary. Each component outside of the blue box is outside of the TOE physical boundary. Thus, no hardware belongs to the TOE, and no software for kernel device drivers, PSP,

system extensions and no partitions belong to the TOE.

The TOE also includes the TOE User Manuals, which are used during the system integration phase. A list of TOE User Manuals is given by the following table:

Title	Architecture	Version
PikeOS Safety and Security Manual	Generic	20.21
PikeOS Safety and Security Manual (x86 AMD64 Supplement)	X86_64	20.16
PikeOS Safety and Security Manual (ARM7 Supplement)	ARM V7	20.17
PikeOS Safety and Security Manual (ARM8 Supplement)	ARM V8	20.18
PikeOS Platform Manual for x86-amd64 Boards	X86_64	4.2-245
PikeOS Platform Manual for ARM v7 Boards	ARM V7	4.2-195
PikeOS Platform Manual for ARM v8-A 64-bit Boards	ARM V8	4.2-195
User Manual PikeOS Certification Kit (X86_AMD64)	X86_64	20.9
User Manual PikeOS Certification Kit (ARM_V7HF)	ARM_V7	20.9
User Manual PikeOS Certification Kit (ARM_V8HF)	ARM_V8	20.9
PikeOS 4.2 Installation Guide	Generic	25.10.2018
PikeOS User Manual	Generic	4.2-342
PikeOS Kernel Reference Manual	Generic	4.2-111
PikeOS System Software Reference Manual	Generic	4.2-133
PikeOS PSP Developer's Guide	Generic	4.2-117
PikeOS Device Driver Programming Reference Manual	Generic	4.2-154
P4EXT PikeOS Native Personality Extensions	Generic	4.2-44
CENV C Language Programming Environment	Generic	4.2-22

The TOE on the corresponding architecture is provided through:

TOE on x86 64-bit

Short description	ISO name	Content
PikeOS 4.2.3 x86 64-bit	R4p2_PIKEOS_MT_CMB_X86_S5577.iso	PikeOS 4.2.3 build S5577 for x86 64-bit architecture
PikeOS 4.2.3 x86 64-bit Certification Kit	R4p2_PIKEOS_DK_CMB_X86_AMD64_-CERTKIT_S5577.iso	PikeOS 4.2.3 certification documentation for x86 64-bit

TOE on ARM v7

Short description	ISO name	Content
PikeOS 4.2.3 ARM v7	R4p2_PIKEOS_MT_CMB_ARM_V7HF_-S5577.iso	PikeOS 4.2.3 build S5577 for ARM v7 architecture
PikeOS 4.2.3 ARM v7 Certification Kit	R4p2_PIKEOS_DK_CMB_ARM_V7HF_-CERTKIT_S5577.iso	PikeOS 4.2.3 certification documentation for ARM v7

TOE on ARM v8

Short description	ISO name	Content
PikeOS 4.2.3 ARM v8	R4p2_PIKEOS_MT_CMB_ARM_V8HF_-S5577.iso	PikeOS 4.2.3 build S5577 for ARM v8 architecture
PikeOS 4.2.3 ARM v8 Certification Kit	R4p2_PIKEOS_DK_CMB_ARM_V8HF_-CERTKIT_S5577.iso	PikeOS 4.2.3 certification documentation for ARM v8

2.4.6 TOE Logical Boundary

The TOE provides the following TOE security services, abbreviated as TSS_XXX:

- TSS_SSA: Separation in space of applications hosted in different partitions from each other and from the *PikeOS Operating System* according to the SSP by using the underlying hardware, as shown by the red lines in Figure 1.

  Applications can be hosted in different partitions. Partitions get assigned resources (i.e. space) according to the SSP, which comprise memory ranges and a set of CPUs. The TSF enforces the corresponding part of the SSP by the enforcement of access control on partition content, per-partition provision of physical memory space and allocated CPU time for each CPU.

  By confining non-privileged executables into partitions, the TSF enforces that these applications can affect neither applications in other partitions nor the *PikeOS Operating System* itself.
- TSS_STA: Separation in time of applications hosted in different partitions from each other and from the *PikeOS Operating System* according to the SSP.

  Applications can be hosted in different partitions. Partitions get assigned CPU time (i.e. time windows) according to the SSP. The TSF enforces the corresponding part of the SSP by per-partition allocation of a predefined amount of CPU time for each CPU. On a partition switch CPUs will be reused.
- TSS_COM: Provision and management of communication objects.

  Applications hosted in different partitions can get assigned a set of communication objects. A *communication object* is an object exposed to one or multiple partitions with access rights as defined in the configuration data, thus allowing communication between partitions.
- TSS_MAN: Management of the TOE (e.g. system partition API) and the TOE data (e.g. threads, tasks)

  The *PikeOS Separation Kernel* restricts a non-privileged application to only manage tasks and threads within its partition. The *PikeOS Separation Kernel* provides an API to privileged applications to manage the TOE and the TOE data.

3 Conformance Claims

3.1 CC Conformance Claim

This Security Target (ST) conforms to the Common Criteria for Information Technology Security Evaluation Version 3.1 Revision 4 [Com12] (CC) as follows:

- Part 2 conformant
- Part 3 conformant

3.2 Protection Profile Claim

This ST does not claim conformance with any Protection Profile.

3.3 Package Claim

This ST claims conformance to the Evaluation Assurance Level 3 (EAL 3), augmented with ALC_FLR.3. Thus, this ST is EAL 3 augmented.

3.4 Conformance Rationale

Since this ST does not claim conformance to any protection profile, this section is not applicable.

4 Security Problem Definition

4.1 Assets

All assets within the same domain as the TSF (see Figure 1, red-line shape with TSF) are treated in this security target together with the asset TSF data (AS.TSF_DATA).

Asset Name	Description	Security Properties to be Preserved
Memory (AS.MEM)	RAM or ROM memory, memory mapped I/O, and port mapped I/O assigned to a partition by the integrator.	confidentiality, integrity, availability
Files (AS.FILE)	Access to files and access modes assigned to a partition by the integrator.	confidentiality, integrity
Ports (AS.PORT)	Each port is either a source port or a destination port. The integrator uses channels to specify from which source ports PikeOS transfers messages to which destination ports.	confidentiality, integrity
Interrupts (AS.INT)	Set of interrupts allocated to each partition, configured by the integrator.	confidentiality, integrity
Access to PSP-specific services (AS.PS)	For each access to PSP-specific services, the right to invoke that driver is configured by the integrator per partition.	confidentiality, integrity
CPU cores (AS.CORE)	Set of CPU cores allocated to each partition, configured by the integrator.	integrity
Memory reserved for exclusive access by the PikeOS hypervisor (AS.KMEM)	Memory reserved for exclusive access by the PikeOS hypervisor used on behalf of the resource partition, configured by the integrator. Application Note: This memory is within the same domain as the TSF.	availability
CPU processing time (AS.TIME)	The integrator assigns time windows to partitions. This defines the CPU processing time of each partition.	availability
Tasks (AS.TASK)	An application always has at least one task. Tasks are used to structure the assigned memory into address spaces. This asset consists of this structure. Application Note: The content of tasks is already covered by AS.MEM.	API protection
Threads (AS.THR)	An application always has at least one thread. The asset consists of all threads that can be created in a partition.	API protection
TSF data (AS.TSF_DATA)	TSF data consists of: Configuration data: Data used by the TSF to enforce the System Security Policy (SSP, Section 1.4.4.2).	confidentiality, integrity

Asset Name	Description	Security Properties to be Preserved
	Run-time data such as partition attributes, task and thread management data.	
System Partition API (AS.SYS_PART_API)	The *system partition API* is an interface to functions of the TSF available for system partitions.	API protection

Table 4: Assets

4.2 Threats

Assets are defined in Table 4 in Section 4.1. **An attacker is a non-privileged executable.**

T.DISCLOSURE

An attacker reads an asset of which the property confidentiality shall be maintained according to Table 4 and the SSP.

T.MODIFICATION

An attacker writes an asset of which the property integrity shall be maintained according to Table 4 and the SSP.

T.DEPLETION

By consuming resources of which the property availability shall be maintained according to Table 4 and the SSP an attacker makes these resources unavailable to the TOE itself and/or to non-privileged executables and/or to privileged executables.

T.EXECUTION

An attacker executes a management function of which the property API protection shall be maintained according to Table 4 and the SSP without being authorized to do so.

4.3 Organizational Security Policies

This ST defines no organizational security policies.

4.4 Assumptions

A.PRIVILEGED_EXECUTABLES

All privileged executables are approved by the integrator. The integrator thereby takes responsibility that the privileged executables have been developed according to the TOE User Manuals and do not violate the SSP. The integrator takes responsibility not to put privileged executables and non-privileged executables into the same partition.

Application Note: The TOE User Manuals provide detailed guidance on secure configuration and usage of the TOE.

A.HARDWARE

The underlying hardware, firmware and bootloader needed by PikeOS to guarantee secure operation provide the necessary properties, are working correctly and have no undocumented or unintended security critical side effect on the functions of the TOE.

The hardware must fulfill the following requirements, as explained in the TOE User Manuals:

1. Provide CPU(s) with at least two privilege modes (“user” and “supervisor” mode). Only the *PikeOS Separation Kernel* itself and privileged executables may run in the “supervisor” mode. Non-privileged executables always run in “user mode”. In “user mode”, only a limited set of instructions is available; in “supervisor mode”, all instructions are available.
2. The hardware shall have a MMU, which is capable of restricting accesses (e.g. destinations of load and store CPU instructions) of non-privileged executables to certain memory regions. The MMU shall only be configurable from a privileged CPU mode, thus, it can only be configurable through the TOE to configure the policies specifying these access restrictions. These policies are part of the SSP. During TOE run time, these policies are represented as page tables used by the MMU.
3. The hardware (CPU or CPUs) shall provide instructions to switch between privilege modes and to use the memory management to set up different segments of memory.
4. The hardware (CPU or CPUs) shall allow the TOE to reuse CPU(s) for different non-privileged executables, in a way that there is no residual information flow through CPU registers across a partition boundary.
5. The hardware shall provide default values for security-relevant settings at power-on (e.g. program counter, detailed instructions shall be included in the hardware reference manual). This supports the TOE reaching the initial safe and secure state.
6. If the hardware possesses any other active components beside CPUs or CPUs have operating mode(s) not under control of PikeOS, then the hardware shall provide support either to turn these components completely off or to control them as described in the TOE User Manuals. For example, if a device accessible by non-privileged executables can execute DMA, then all DMA shall be switched off or, in order to control DMA, the hardware shall provide an I/O MMU, with an I/O MMU driver protected by PikeOS.

The timer facilities provided by the hardware shall be sufficient for the timing requirements (e.g., timer resolution) of the product based on PikeOS. The CPU-specific requirements are met by all x86 64-bit, ARMv7, or ARMv8 CPUs specified in the TOE User Manuals for the selected CPU architecture.

A.EXCLUSIVE_RESOURCES

All resources required by PikeOS, its privileged executables, and its non-privileged executables are exclusively controlled by the TOE.

A.PHYSICAL

It is assumed that the IT environment provides the TOE with appropriate physical security, commensurate with the value of the IT assets protected by the TOE.

A.TRUSTWORTHY_PERSONNEL

The personnel configuring, integrating, and developing the product based on the TOE (integrator) are trustworthy, act according to the TOE User Manuals and are sufficiently qualified for this task.

Application Note: The system integration phase (Section 2.4.4.2) can be split into the four steps: selection of the TOE operational environment, non-privileged and privileged executables, and deciding on system partitions and normal partitions (Step 1), fusion (Step 2), configuration of the TOE (Step 3), and integration (Step 4). At each step of the system integration phase, the integrator shall follow the TOE User Manuals.

5 Security Objectives

5.1 Security Objectives for the TOE

OT.CONFIDENTIALITY

For each asset, the TOE shall preserve its confidentiality according to Table 4 and the SSP.

OT.INTEGRITY

For each asset, the TOE shall preserve its integrity according to Table 4 and the SSP.

OT.RESOURCE_AVAILABILITY

For resources assigned to partitions and to TSF data, the TOE shall preserve their availability according to Table 4 and the SSP.

Application Note: In this ST availability means that the TOE will provide a specified amount of a resource or ability to use some TOE services. This ST does not consider availability in the sense of physical availability such as tolerance to power failure.

OT.API_PROTECTION

The TSF shall prevent any execution of a management function reserved to privileged executables by non-privileged executables.

5.2 Security Objectives for the Operational Environment

OE.PRIVILEGED_EXECUTABLES

All privileged executables are approved by the integrator. The integrator thereby takes responsibility that the privileged executables have been developed according to the TOE User Manuals and do not violate the SSP.

OE.HARDWARE

The underlying hardware, firmware and bootloader needed by PikeOS to guarantee secure operation provide the necessary properties, are working correctly and have no undocumented security critical side effect on the functions of the TOE.

The hardware must fulfill the following requirements, as explained in the TOE User Manuals:

1. Provide CPU(s) with at least two privilege modes (“user” and “supervisor” mode). Only the PikeOS separation kernel itself and privileged executables may run in the “supervisor” mode. Non-privileged executables always run in “user mode”. In “user mode”, only a limited set of instructions is available; in “supervisor mode”, all instructions are available.
2. The hardware shall have a MMU, which is capable of restricting accesses (e.g. destinations of load and store CPU instructions) of non-privileged executables to certain memory regions. The MMU shall only be configurable from a privileged CPU mode, thus, it can only be configurable through the TOE to configure the policies specifying these access restrictions. These policies are part of the SSP. During TOE run time, these policies are represented as page tables used by the MMU.
3. The hardware (CPU or CPUs) shall provide instructions to switch between privilege modes and to use the memory management to set up different segments of memory.
4. The hardware (CPU or CPUs) shall allow the TOE to reuse CPU(s) for different non-privileged executables, in a way that there is no residual information flow through CPU registers across a partition boundary.
5. The hardware shall provide default values for security-relevant settings at power-on (e.g. program counter, detailed instructions shall be included in the hardware reference manual). This supports the TOE reaching the initial safe and secure state.
6. If the hardware possesses any other active components beside CPUs or CPUs have operating mode(s) not under control of PikeOS, then the hardware shall provide support either to turn these components completely off or to control them as described in the TOE User

Manuals. For example, if a device accessible by non-privileged executables can execute DMA, then all DMA shall be switched off or, in order to control DMA, the hardware shall provide an I/O MMU, with an I/O MMU driver protected by PikeOS.

Specific requirements to the x86 64-bit architecture are:

- The processors are operated in 64-bit mode.
- AMD64 instruction set architecture.
- Non-Execute bit (NX bit) support enabled in the BIOS.

Specific requirements to the ARMv7 architecture (Cortex-A7, Cortex-A9, Cortex-A15 …) are:

- The processors are operated in 32bit mode.
- Memory Management Unit (MMU) with Virtual Memory System Architecture.
- Vector Floating Point (VFP) / Advanced SIMD (Neon) extension.

Specific requirements to the ARMv8 architecture (Cortex-A35, Cortex-A53, Cortex-A57 and Cortex-A72) are:

- The processors are operated in 64-bit mode.
- Memory Management Unit (MMU) with Virtual Memory System Architecture.
- Vector Floating Point (VFP) / Advanced SIMD (Neon) extension

The timer facilities provided by the hardware shall be sufficient for the timing requirements (e.g., timer resolution) of the product based on PikeOS. The CPU-specific requirements are met by all x86 64-bit, ARMv7, or ARMv8 CPUs specified in the TOE User Manuals for the selected CPU architecture.

OE.EXCLUSIVE_RESOURCES

All resources required by PikeOS, its privileged executables, and its non-privileged executables are exclusively controlled by the TOE.

OE.PHYSICAL

The IT environment provides the TOE with appropriate physical security, commensurate with the value of the IT assets protected by the TOE.

OE.TRUSTWORTHY_PERSONNEL

The personnel configuring and integrating the TOE (integrator) and those installing and operating the TOE (system operator) are trustworthy, act according to the TOE User Manuals, and are sufficiently qualified for this task.

5.3 Security Objectives Rationale

The following table provides an overview for security objectives coverage (TOE and its environment) and also gives an evidence for *sufficiency* and *necessity* of the defined objectives. It shows that all threats and OSPs are addressed by the security objectives and it also shows that all assumptions are addressed by the security objectives for the TOE operational environment.

	OT.CONFIDENTIALITY	OT.INTEGRITY	OT.RESOURCE_AVAILABILITY	OT.API_PROTECTION	OE.PRIVILEGED_EXECUTABLES	OE.HARDWARE	OE.EXCLUSIVE_RESOURCES	OE.PHYSICAL	OE.TRUSTWORTHY_PERSONNEL
T.DISCLOSURE	X								
T.MODIFICATION		X							
T.DEPLETION			X						
T.EXECUTION				X					
A.PRIVILEGED_EXECUTABLES					X				
A.HARDWARE						X			
A.EXCLUSIVE_RESOURCES							X		
A.PHYSICAL								X	
A.TRUSTWORTHY_PERSONNEL									X

Table 5: Security Objectives Rationale

A justification required for *suitability* of the security objectives to cope with the security problem definition is given below:

5.3.1 Security Objectives Rationale: Threats

5.3.1.1 Threat: T.DISCLOSURE

If the security objective OT.CONFIDENTIALITY has been reached, the threat T.DISCLOSURE is completely eliminated.

5.3.1.2 Threat: T.MODIFICATION

If the security objective OT.INTEGRITY has been reached, the threat T.MODIFICATION is completely eliminated.

5.3.1.3 Threat: T.DEPLETION

If the security objective OT.RESOURCE_AVAILABILITY has been reached, the threat T.DEPLETION is completely eliminated.

5.3.1.4 Threat: T.EXECUTION

If the security objective OT.API_PROTECTION has been reached, the threat T.EXECUTION is

completely eliminated.

5.3.2 Security Objectives Rationale: Assumptions

Each security assumption in this Security Target is addressed by at least one security objective for the operational environment. This section maps assumptions to environmental security objectives and provides a rationale how the assumption is fulfilled.

5.3.2.1 Assumption: A.PRIVILEGED_EXECUTABLES

OE.PRIVILEGED_EXECUTABLES directly upholds A.PRIVILEGED_EXECUTABLES.

5.3.2.2 Assumption: A.HARDWARE

OE.HARDWARE directly upholds A. HARDWARE.

5.3.2.3 Assumption: A.EXCLUSIVE_RESOURCES

OE.EXCLUSIVE_RESOURCES directly upholds A.EXCLUSIVE_RESOURCES.

5.3.2.4 Assumption: A.PHYSICAL

OE.PHYSICAL directly upholds A.PHYSICAL.

5.3.2.5 Assumption: A.TRUSTWORTHY_PERSONNEL

OE.TRUSTWORTHY_PERSONNEL directly upholds A.TRUSTWORTHY_PERSONNEL.

6 Extended Components Definition

This ST does not include any extended components.

7 Security Requirements

This section defines security functional requirements (SFRs) and security assurance requirements (SARs), which apply for the TOE.

7.1 Security Functional Requirements

We perform assignment, selection, and refinement of the SFRs provided by the CC. The assignment operation is marked by square brackets "[]". The selection operation is marked in *italic*. In a refinement operation added text is underlined and removed text is ~~crossed out~~.

The iteration operation is used when a component is repeated on varying assets. Iteration is denoted by showing a slash ("/") and the iteration indicator after the component identifier. For example, FDP_ACF.1/CPA indicates an iteration of FDP_ACF.1 on 'communication port access'. Iterations applied to assets follow the order of Table 4 in Section 4.1 (assets). Where sentences are grouped by having the same indentation level, the terms "OR" and "AND" in uppercase are used to indicate their logical relation.

Configuration XML elements, attributes, and values are denoted in <angle brackets>. These configuration elements, attributes, and values have speaking names, and when the configuration maintains a list of <Foo> elements, then this list is referred to in the configuration as <FooTable>. For a detailed explanation on how to use the configuration to configure PikeOS see the TOE User Manuals. All symbols beginning with "p4_" or "vm_" are API symbols that PikeOS provides to applications. In this ST we list only those symbols that trigger a system call and are not convenience user space wrappers for other calls. See the TOE User Manuals for details.

Application Note: The SSP is a subset of the VMIT configured by the integrator.

The SFP is a set of rules that are parameterized by the SSP. These rules are fix-coded in the implementation of the TSF. Thus, the behavior of a product based on PikeOS depends on the SFP and SSP.

In the following the SFP is split up into sub-SFPs as follows:

- memory access control policy (MA)
- file access control policy (FA)
- communication port access control policy (CPA)
- interrupt access control policy (IA)
- PSP-specific services access control policy (PSA)
- CPU core access policy (CCA)
- IPC and event communication policy (IEC)

Application Note: In this ST the term *subject* is used for a thread in a partition, for an application, or for a partition as a whole depending on the context. In the running TOE the entity executing application code is a PikeOS thread. In a partition there can be multiple threads and each thread is uniquely assigned to its partition. Since this ST works on security policies at partition level, we abstract all threads in one partition to the partition subject.

7.1.1 User Data Protection (FDP)

7.1.1.1 Complete Access Control (FDP_ACC.2) and Security Attribute Based

Access Control (FDP_ACF.1)

7.1.1.1.1 FDP_ACC.2/MA Complete Access Control – Memory Access

FDP_ACC.2.1/MA: The TSF shall enforce the [memory access control policy] on [subjects: partitions, objects: memory] and all operations among subjects and objects covered by the SFP.

FDP_ACC.2.2/MA: The TSF shall ensure that all operations between any subject controlled by the TSF and any object controlled by the TSF are covered by an access control SFP.

Application Note: The TSF initializes the MMU of the CPU and sets up page tables in the memory allocated to the TSF to enforce this policy.

7.1.1.1.2 FDP_ACF.1/MA Security Attribute Based Access Control – Memory Access

FDP_ACF.1.1/MA: The TSF shall enforce the [memory access control policy] to objects based on the following [subjects: partitions, objects: memory areas, security attributes: partition ID, attributes defined for the memory area].

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

Access to physical memory M of type <VM_MEM_TYPE_ROM>, <VM_MEM_TYPE_RAM>, <VM_MEM_TYPE_IO_MEM>, or <VM_MEM_TYPE_IO_PORT> is allowed to a subject in partition PA if:

- M is specified in a <MemoryRequirement> MR in the <MemoryRequirementTable> contained within the <Partition> P and the attribute <IsPool> is set to <false>

AND

- the access operation is read or write or execute and the access mode <AccessMode> of MR matches <VM_MEM_ACCESS_RD> for read, <VM_MEM_ACCESS_WR> for write, and <VM_MEM_ACCESS_EXEC> for execute correspondingly

OR

- M is specified in a <MemoryRequirement> MR in the <MemoryRequirementTable> contained within the <Partition> P and the attribute <IsPool> is set to <true>

AND

- the access operation is read or write or execute and the <AccessMode> of MR matches <VM_MEM_ACCESS_RD> for read, <VM_MEM_ACCESS_WR> for write, and <VM_MEM_ACCESS_EXEC> for execute correspondingly

OR

- M is in the text segment of an ELF file referenced in a <File> element in the <FileTable> contained in a <Process> of the <ProcessTable> of the <Partition> P and the attribute <ExecInPlace> is set to <true>

AND

- the access operation is read or execute

OR

- M is specified in a property file system <prop_memmap> node PN

AND

- the <FileAccessTable> element of PA has an element of type <FileAccess> where the <AccessMode> attribute is AM and attribute <FileName> matches the PN

AND

- a subject in partition PA has successfully performed open operation (vm_open) on the property node name PN with access flags AF including <VM_O_MAP> and AF being subset of AM, resulting in a file descriptor FD

AND

- a subject in partition PA has successfully performed a property memory mapping (vm_prop_mem_map) operation with the file descriptor FD

AND

- the access operation is read or write or execute and compatible with AF

OR

- M is specified in a property file system <prop_portmap> node PN

AND

- the <FileAccessTable> element of PA has an element of type <FileAccess> where the <AccessMode> attribute is AM and the attribute <FileName> matches the PN

AND

- a subject in partition PA has successfully performed open operation (vm_open) on the property node name PN with access flags including <VM_O_MAP> and access flags being subset of AM, resulting in a file descriptor FD

AND

- A subject in partition PA has successfully performed a property I/O port mapping (vm_prop_ioport_map) operation with the file descriptor FD

AND

- the access operation is read or write

OR

- M has been received via mapping IPC (p4_ipc) with the requested access mode allowing to read or write M

AND

- the access operation is read or write respectively

OR

- M has been received via vm_map with access mode MAM

AND

- There is file F in entry <FileAccess> with <FileName> F in the <FileAccessTable> list contained within the <Partition> P

AND

- F is successfully opened (vm_open) according to FDP_ACF.1.2/FA with an access mode AM including <VM_O_MAP> and being a superset of MAM

AND

- the access operation is read or write or execute and the MAM matches <VM_MEM_ACCESS_RD> for read, <VM_MEM_ACCESS_WR> for write, and <VM_MEM_ACCESS_EXEC> for execute correspondingly

OR

- M is part of a memory page that the TSF has mapped in read-only access mode to all partitions (“kernel info page”)

AND

- the access operation is read

]

FDP_ACF.1.3/MA: The TSF shall explicitly authorize access of subjects to objects based on the following additional rules: [none].

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

Application Note: The operations on I/O ports are available only for the x86 platform. The TSF on x86 configures the CPU’s I/O port unit.

Application Note: The memory pool can be used by a partition in three ways: (1) an application in a partition uses memory allocation service calls (e.g. vm_malloc) to get memory from the pool; (2) the application ELF file is mapped to the memory allocated from the pool during partition initialization, if it is specified in the VMIT; (3) memory is mapped from the pool to partition during partition initialization, if it is specified in the VMIT.

Application Note: Shared memory in PikeOS is implemented as files provided by internal file provider shm. Shared memories are specified in the <SharedMemoryTable> and are processed in the same way as the disjunction for the case vm_map.

7.1.1.1.3 FDP_ACC.2/FA Complete Access Control – File Access

FDP_ACC.2.1/FA: The TSF shall enforce the [file access control policy] on [subjects: partitions, objects: files] and all operations among subjects and objects covered by the SFP.

FDP_ACC.2.2/FA: The TSF shall ensure that all operations between any subject controlled by the TSF and any object controlled by the TSF are covered by an access control SFP.

7.1.1.1.4 FDP_ACF.1/FA Security Attribute Based Access Control – File Access

FDP_ACF.1.1/FA: The TSF shall enforce the [file access control policy] to objects based on the following: [subjects: partitions, objects: files, security attributes: partition ID, attributes defined for the files in the <FileAccess> configuration element of the partition in the VMIT].

FDP_ACF.1.2/FA: The TSF shall enforce the following rules to determine if an operation among

controlled subjects and controlled objects is allowed: [

- an open operation (vm_open) on any file F provided by a builtin file provider, external file provider, system extension or kernel device driver to partition PA is allowed if:
    - The <FileAccessTable> element of PA has an element of type <FileAccess> where attribute <FileName> matches F

    AND

    - the access flags provided as an argument to the vm_open operation are subset of <AccessMode> element for file F
- a mount operation (vm_mount) on any volume V to partition PA is allowed if:
    - The <FileAccessTable> element of PA has an element of type <FileAccess> where attribute <FileName> matches V and <AccessMode> contains <VM_O_MOUNT> flag

    AND

    - the access flags provided as an argument to the vm_mount operation are subset of <AccessMode> element for the volume V
- a status request operation (vm_stat) on any file F by a builtin file provider, external file provider, system extension or kernel device driver to partition PA is allowed if:
    - The <FileAccessTable> element of PA has an element of type <FileAccess> where attribute <FileName> matches F
- an access operation (vm_close, vm_fstat, vm_fsync, vm_ioctl, vm_lseek, vm_map, vm_prop_read, vm_prop_write, vm_read, vm_read_at, vm_tdiscard_at, vm_test, vm_write, vm_write_at) on file F provided by a builtin file provider, system extension or kernel device driver to partition PA is allowed if:
    - the file has been successfully opened before with the access flags including <VM_O_RD> and the access operation is vm_read, vm_read_at, or vm_prop_read

    OR

    - the file has been successfully opened before with the access flags including <VM_O_WR> and the access operation is vm_write, vm_write_at, or vm_prop_write

    OR

    - the file has been successfully opened before with the access flags including <VM_O_MAP> and the access operation is vm_map

    OR

    - the file has been successfully opened before with the access flags including <VM_O_RD> or <VM_O_WR> and the access operation is vm_lseek

    OR

    - the file has been successfully opened before and the operation is vm_close, vm_fstat, vm_fsync, vm_ioctl, vm_test, vm_tdiscard_at

].

Application Note: Files can be provided by builtin file providers, system extensions, kernel device drivers, external file providers, or volume providers. An access operation itself on a file provided by a builtin file provider is implemented by the TSF. An access operation on a file provided by a system extension or kernel device driver is directly forwarded by the TSF, after validating the file descriptor, to the corresponding function of the system extension or kernel device driver after the check described FDP_ACF.1.2/FA. An access operation on a file provided by an external file provider is implemented as a direct IPC to the file provider. See FDP_IFC.2 and FDP_IFF.1. An open, status request or access operation on a file provided by a volume file provider is implemented as a direct IPC to the volume provider. See FDP_IFC.2 and FDP_IFF.1.

FDP_ACF.1.3/FA: The TSF shall explicitly authorize access of subjects to objects based on the following additional rules: [none].

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

7.1.1.1.5 FDP_ACC.2/CPA Complete Access Control – Communication Port Access

FDP_ACC.2.1/CPA: The TSF shall enforce the [communication port access control policy] on [subjects: partitions, objects: communication ports] and all operations among subjects and objects covered by the SFP.

FDP_ACC.2.2/CPA: The TSF shall ensure that all operations between any subject controlled by the TSF and any object controlled by the TSF are covered by an access control SFP.

7.1.1.1.6 FDP_ACF.1/CPA Security Attribute Based Access Control – Communication Port Access

FDP_ACF.1.1/CPA: The TSF shall enforce the [communication port access control policy] to objects based on the following: [subjects: partitions, objects: communication ports, security attributes: partition ID, port attribute <Direction> in the VMIT, <Channel> configuration elements in the VMIT].

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

- an open operation (vm_qport_open, vm_sport_open) on port PO is allowed to partition PA if:
    - the <QueuingPortTable> element of PA has an element of type <QueuingPort> where attribute <Name> equals PO or the <SamplingPortTable> of PA has an element of type <SamplingPort> where attribute <Name> equals PO
    - the argument direction to the open (vm_qport_open, vm_sport_open) operation is equal to the attribute <Direction> of the PO
- a status request operation, (vm_qport_iterate, vm_qport_stat, vm_sport_iterate, vm_sport_stat) on port PO is allowed to partition PA if:
    - the <QueuingPortTable> element of PA has an element of type <QueuingPort> where attribute Name equals PO or the <SamplingPortTable> of PA has an element of type <SamplingPort> where attribute <Name> equals PO
- an access operation on port PO successfully opened by PA is allowed if:
    - if the operation is read (vm_qport_read, vm_qport_read_routed, vm_sport_read), and the port PO was successfully opened with destination direction <VM_PORT_DESTINATION>.
    - if the operation is write (vm_qport_write, vm_qport_write_routed, vm_sport_write) and the port was successfully opened with destination direction <VM_PORT_SOURCE>.
    - all other access operations (vm_qport_clear, vm_qport_close, vm_qport_control, vm_qport_pstat, vm_qport_psync, vm_qport_test, vm_sport_clear, vm_sport_close, vm_sport_control, vm_sport_pstat, vm_sport_psync, vm_sport_test) on ports are allowed

].

FDP_ACF.1.3/CPA: The TSF shall explicitly authorize access of subjects to objects based on the following additional rules: [none].

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

7.1.1.1.7 FDP_ACC.2/IA Complete Access Control – Interrupt Access

FDP_ACC.2.1/IA: The TSF shall enforce the [interrupt access control policy] on [subjects: partitions, objects: interrupts] and all operations among subjects and objects covered by the SFP.

FDP_ACC.2.2/IA: The TSF shall ensure that all operations between any subject controlled by the TSF

and any object controlled by the TSF are covered by an access control SFP.

7.1.1.1.8 FDP_ACF.1/IA Security Attribute Based Access Control – Interrupt Access

FDP_ACF.1.1/IA: The TSF shall enforce the [interrupt access control policy] to objects based on the following: [subjects: partitions, objects: interrupts, security attributes: partition ID, attributes defined for files in the <FileAccess> configuration element of the partition in the VMIT].

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

The operation p4_int_attach of attaching to interrupt IN (i.e., to have an interrupt handler invoked when interrupt IN is triggered) is allowed to a subject in partition PA if:

- IN is specified in a property file system <prop_interrupt> node PN
- The <FileAccessTable> element of PA has an element of type <FileAccess> where the attribute <FileName> matches PN and <AccessMode> attribute AM
- A subject in partition PA has successfully performed an open operation (vm_open) on the property node PN with access flags including <VM_O_MAP> and being subset of AM, resulting in file descriptor FD
- A subject in partition PA has performed successfully vm_prop_int_grant with the file descriptor FD, giving a valid interrupt number IN

].

FDP_ACF.1.3/IA: The TSF shall explicitly authorize access of subjects to objects based on the following additional rules: [none].

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

7.1.1.1.9 FDP_ACC.2/PSA Complete Access Control – Access to PSP-Specific Services

FDP_ACC.2.1/PSA: The TSF shall enforce the [PSP-specific services access control policy] on [subjects: partitions, objects: PSP-specific services] and all operations among subjects and objects covered by the SFP.

FDP_ACC.2.2/PSA: The TSF shall ensure that all operations between any subject controlled by the TSF and any object controlled by the TSF are covered by an access control SFP.

7.1.1.1.10FDP_ACF.1/PSA Security Attribute Based Access Control – Access to PSP-Specific Services

FDP_ACF.1.1/PSA: The TSF shall enforce the [PSP-specific services access control policy] to objects based on the following: [subjects: partitions, objects: interrupts, security attributes: partition ID, attributes defined for files in the <FileAccess> configuration element of the partition in the VMIT].

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

The operation of invoking a PSP-specific service PS (p4_dev_call) is allowed to a subject in partition PA if:

- PS is specified in a property file system <prop_device> node PN
- The <FileAccessTable> element of PA has an element of type <FileAccess> where the attribute <FileName> matches to the PN and <AccessMode> attribute AM
- A subject in partition PA has successfully performed an open operation (vm_open) on the property node PN with access flags including <VM_O_MAP> and being subset of AM, resulting in file descriptor FD

- A subject in partition PA has successfully performed vm_prop_dev_grant with the file descriptor FD, giving access to the PSP-specific service PS

].

FDP_ACF.1.3/PSA: The TSF shall explicitly authorize access of subjects to objects based on the following additional rules: [none].

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

7.1.1.1.11FDP_ACC.2/CCA Complete Access Control – CPU Core Access Control

FDP_ACC.2.1/CCA: The TSF shall enforce the [CPU core access control policy] on [subjects: partitions, objects: CPU cores] and all operations among subjects and objects covered by the SFP.

FDP_ACC.2.2/CCA: The TSF shall ensure that all operations between any subject controlled by the TSF and any object controlled by the TSF are covered by an access control SFP.

7.1.1.1.12FDP_ACF.1/CCA Security Attribute Based Access Control – CPU Core Access Control

FDP_ACF.1.1/CCA: The TSF shall enforce the [CPU core access control policy] to objects based on the following: [subjects: partitions, objects: CPU cores, security attributes: partition ID, CPU cores in the <CpuMask> configuration element of the partition in the VMIT].

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

- The partition PA can run a thread on CPU core C if the <CpuMask> element of PA has set the C^{th} bit in the <CpuMask>.]

FDP_ACF.1.3/CCA: The TSF shall explicitly authorize access of subjects to objects based on the following additional rules: [none].

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

7.1.1.2 FDP_IFC.2 Complete Information Flow Control

FDP_IFC.2.1: The TSF shall enforce the [IPC and event communication policy] on

- [all subjects: partitions]

and all operations that cause that information to flow to and from subjects covered by the SFP.

FDP_IFC.2.2: The TSF shall ensure that all operations that cause any information in the TOE to flow to and from any subject in the TOE are covered by an information flow control SFP.

7.1.1.3 FDP_IFF.1 Simple Security Attributes

FDP_IFF.1.1: The TSF shall enforce the [IPC and event communication policy] based on the following types of subject and information security attributes: [

- subject identity: thread ID
- information security attributes: file access permissions to a file marked with <FileAccess> attribute <VM_O_FSPROV> or <VM_O_VOLPROV>]

FDP_IFF.1.2: The TSF shall permit an information flow between a controlled subject and controlled information via a controlled operation if the following rules hold: [

The operation signal event (p4_ev_signal) or send IPC (p4_ipc), from a thread in partition PA1 to a subject in partition PA2 or vice versa is allowed if

- the VMIT specifies that partition PA1 provides a file F with <AccessMode> attribute <VM_O_FSPROV>

AND

- the <FileAccessTable> of PA2 has an element of type <FileAccess> where attribute <AccessMode> is AM and attribute <FileName> matches F

AND

- PA1 has successfully registered the file provider (vm_fp_register) with the TOE

AND

- a subject in PA2 has successfully performed an open operation (vm_open) on file F with access flags being subset of AM

OR

- the VMIT specifies that partition PA1 provides a volume V with <AccessMode> attribute <VM_O_VOLPROV>

AND

- the <FileAccessTable> of PA2 has an element of type <FileAccess> where <AccessMode> is AM and contains <VM_O_MOUNT> and attribute <FileName> matches V

AND

- PA1 has successfully registered the volume provider (vm_vp_register) with the TOE

AND

- a subject in PA2 has successfully performed a mount operation (vm_mount) on V with access flags being subset of AM]

FDP_IFF.1.3: The TSF shall enforce the [additional information flow rules: none].

FDP_IFF.1.4: The TSF shall explicitly authorize an information flow based on the following rules: [none].

FDP_IFF.1.5: The TSF shall explicitly deny an information flow based on the following rules: [none].

7.1.2 Identification and Authentication (FIA)

7.1.2.1 FIA_UID.2 User Identification

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

Application Note: A “user” of the TOE is a partition.

7.1.3 Security Management (FMT)

7.1.3.1 FMT_MSA.1 Management of Security Attributes

FMT_MSA.1.1: The TSF shall enforce the [IPC and event communication policy, memory access control policy, file access control policy, communication port access control policy, PSP-specific services access control policy, interrupt access control policy, CPU core access control policy] to restrict the ability to [write] the security attributes [specified in the VMIT and property file system] to [system partitions].

7.1.3.2 FMT_MSA.3 Static Policy Attribute Initialization

FMT_MSA.3.1: The TSF shall enforce the [IPC and event communication policy, memory access control policy, file access control policy, communication port access control policy, PSP-specific

services access control policy, interrupt access control policy, CPU core access control policy] to provide [restrictive or integrator-defined] default values for security attributes that are used to enforce the SFP.

Application Note: The integrator defines the VMIT. The VMIT is used by the TSF as the source of initial SSP values. If an attribute is not defined by the integrator in the VMIT for a subject, then the default is to deny any operations involving this attribute, i.e. a white list security policy is implemented. This behavior is specified in FMT_MSA.3.1 as "restrictive default values". Attributes with integrator-defined values in the VMIT are also initial values for the TSF. This behavior is specified in FMT_MSA.3.1 as "integrator-defined default values".

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: The TSF does not have any functionality for specifying alternative initial values to override the default values.

7.1.3.3 FMT_MTD.1/SYS Management of TSF Data – System Partition API

FMT_MTD.1.1/SYS: The TSF shall restrict the ability to [invoke] the [System Partition API] to [system partitions].

Application Note: The complete definition of the System Partition API is given in the TOE User Manuals.

7.1.3.4 FMT_MTD.1/TASK Management of TSF Data – Tasks

FMT_MTD.1.1/TASK: The TSF shall restrict the ability to [

- activate (p4_task_activate)
- terminate (p4_task_terminate)
- modify (p4_comm_grant, p4_comm_link, p4_dev_grant, p4_dev_link, p4_int_grant, p4_int_link, p4_task_start, p4_task_donate, p4_task_hm_register, p4_task_hm_wait, p4_task_hm_wake)
- read (p4_task_get_attr)]

the [tasks] to [the owning partition or system partitions].

Application Note: Tasks are also TSF data. A partition *owns* a task if the task is assigned to it by the integrator in the VMIT.

7.1.3.5 FMT_MTD.1/THR Management of TSF Data – Threads

FMT_MTD.1.1/THR: The TSF shall restrict the ability to [

- create (p4_thread_create_syscall)
- delete (p4_thread_delete)
- modify (p4_fast_set_prio, p4_thread_alarm_syscall, p4_thread_ex_affinity, p4_thread_ex_exh, p4_thread_ex_priority, p4_thread_ex_regs, p4_thread_ex_sched_syscall, p4_thread_except, p4_thread_preempt, p4_thread_resume, p4_thread_set_regs, p4_thread_stop_syscall, p4_thread_yield)
- read (p4_fast_get_prio_syscall, p4_my_cpuid, p4_my_uid_syscall, p4_thread_get_attr, p4_thread_get_regs)]

the [threads] to [the owning partition or system partitions].

Application Note: Threads are also TSF data. A partition owns a thread if the thread is created by one of its applications.

7.1.3.6 FMT_SMF.1 Specification of Management Functions

FMT_SMF.1.1: The TSF shall be capable of performing the following management functions: [

- thread management
- task management
- TOE management (System Partition API)]

7.1.3.7 FMT_SMR.1 Security Roles

FMT_SMR.1.1: The TSF shall maintain the roles: [

- "system partition" and
- "normal partition".]

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

Application Note: The TSF supports roles on partition granularity.

7.1.4 Resource Utilization (FRU)

7.1.4.1 FRU_RSA.2/MEM Minimum and Maximum Quotas – Memory

FRU_RSA.2.1/MEM: The TSF shall enforce maximum quotas of the following resources: [

- System memory: the maximum amount of system memory is the sum of the sizes of all <MemoryRequirement> elements of type <VM_MEM_TYPE_RAM> with attribute <IsPool> set to <true> or of type <VM_MEM_TYPE_KMEM> assigned to that partition in the VMIT]

that *subjects*, which are the non-privileged executables in a normal partition can use *simultaneously*.

FRU_RSA.2.2/MEM: The TSF shall ensure the provision of minimum quantity of each: [

- System memory: the minimum amount of system memory is the sum of the sizes of all <MemoryRequirement> elements of type <VM_MEM_TYPE_RAM> with attribute <IsPool> set to <true> or of type <VM_MEM_TYPE_KMEM> assigned to that partition in the VMIT]

that is available for *subjects*, which are the non-privileged executables in a normal partition to use *simultaneously*.

7.1.4.2 FRU_RSA.2/TIME Minimum and Maximum Quotas – Processing Time

FRU_RSA.2.1/TIME: The TSF shall enforce maximum quotas of the following resources: [

- Processing time: the maximum amount of CPU processing time is the sum of the <Duration> attributes of assigned <Window> elements of its <ScheduleScheme> in the VMIT]

that *subjects*, which are the non-privileged executables in a normal partition can use *over a specified period of time*.

FRU_RSA.2.2/TIME: The TSF shall ensure the provision of minimum quantity of each: [

- Processing time: if time windows are assigned to a partition exclusively, the minimum amount of CPU processing time is the sum of the <Duration> attributes of assigned <Window> elements of its <ScheduleScheme> in the VMIT]

that is available for *subjects*, which are the non-privileged executables in a normal partition to use *over a specified period of time*.

Application Note: The "specified period of time" is the sum of the <Duration> attributes of all <Window> elements the <ScheduleScheme>. The schedule scheme is repeated with cyclic periodicity.

Application Note: If a window is assigned to more than one partition, i.e. the window is not assigned

exclusively, the integrator can use the <MaxPrio> attributes for partitions sharing the window to set up a sharing scheme for the CPU processing time within that window.

7.2 Security Assurance Requirements

This ST claims conformance to the assurance level EAL 3 augmented with ALC_FLR.3.

For the security assurance requirement AVA_VAN.2 of EAL 3, the following refinement is used in AVA_VAN.2.4E:

> "The evaluator shall conduct penetration testing, based on the identified potential vulnerabilities, to determine that the TOE is resistant to attacks performed by an attacker possessing Enhanced-Basic attack potential."

7.3 Security Requirements Rationale

The following table provides an overview for security functional requirements coverage also giving an evidence for sufficiency and necessity of the SFRs chosen.

	OT.CONFIDENTIALITY	OT.INTEGRITY	OT.RESOURCE_AVAILABILITY	OT.API_PROTECTION
FDP_ACC.2/MA	X	X		X
FDP_ACF.1/MA	X	X		X
FDP_ACC.2/FA	X	X		
FDP_ACF.1/FA	X	X		
FDP_ACC.2/CPA	X	X		
FDP_ACF.1/CPA	X	X		
FDP_ACC.2/IA	X	X		
FDP_ACF.1/IA	X	X		
FDP_ACC.2/PSA	X	X		
FDP_ACF.1/PSA	X	X		
FDP_ACC.2/CCA	X	X		
FDP_ACF.1/CCA	X	X		
FDP_IFC.2	X			
FDP_IFF.1	X			
FIA_UID.2	X	X		
FMT_MSA.1	X	X		
FMT_MSA.3	X	X		
FMT_MTD.1/SYS				X
FMT_MTD.1/TASK				X
FMT_MTD.1/THR				X
FMT_SMF.1				X
FMT_SMR.1	X	X		
FRU_RSA.2/MEM	X		X	
FRU_RSA.2/TIME	X		X	

Table 6: Coverage of Security Objectives for the TOE by SFR. “X” is for where a dependency to an objective exists.

7.3.1 Security Objective: OT.CONFIDENTIALITY

For all assets, the operations of non-privileged executables are controlled by the TSF:

- For the asset AS.MEM the SFRs FDP_ACC.2/MA and FDP_ACF.1/MA ensure that non-privileged executables can only access memory (AS.MEM) according to the SSP.
- For the asset AS.FILE the SFRs FDP_ACC.2/FA and FDP_ACF.1/FA ensure that non-privileged executables can only access files (AS.FILE) according to the SSP.
- For the asset AS.PORT the SFRs FDP_ACC.2/CPA and FDP_ACF.1/CPA ensure that non-privileged executables can only access communication ports (AS.PORT) according to the SSP.
- For the asset AS.INT the SFRs FDP_ACC.2/IA and FDP_ACF.1/IA ensure that non-privileged executables can only access interrupts (AS.INT) according to the SSP.
- For the asset AS.PS the SFRs FDP_ACC.2/PSA and FDP_ACF.1/PSA ensure that non-privileged executables can only access PSP-specific services (AS.PSA) according to the SSP.
- For the asset AS.CORE the SFRs FDP_ACC.2/CCA and FDP_ACF.1/CCA ensure that non-privileged executables can only access CPU cores (AS.CORE) according to the SSP.
- For the asset AS.TSF_DATA, the TSF configures the MMU to disallow non-privileged executables to access the memory of any of these other assets (i.e., the memory used for AS.TASK, AS.TSF_DATA and AS.THR). The TSF data also includes all security attributes that TSF uses to manage any asset (e.g. security attributes for file access rights or the schedule schemes used for assignment of CPU processing time).

FIA_UID.2 ensures that partitions are identified; FMT_SMR.1 provides security roles to partitions. FMT_MSA.1 restricts the ability to write the security attributes specified in the VMIT and the property file system to system partitions. FMT_MSA.3 provides restrictive or integrator-defined default values for security attributes.

FDP_IFF.1, FDP_IFC.2 ensure that IPC and event communication information flows originating from non-privileged executables are restricted to information flows allowed according to the SSP. FRU_RSA.2/MEM ensures that no information flow against the SSP can be initiated by memory depletion. FRU_RSA.2/TIME ensures that no information flow against the SSP can be initiated by CPU processing time depletion.

7.3.2 Security Objective: OT.INTEGRITY

For all assets, the operations of non-privileged executables are controlled by the TSF:

- For the asset AS.MEM the SFRs FDP_ACC.2/MA and FDP_ACF.1/MA ensure that non-privileged executables can only access memory (AS.MEM) according to the SSP.
- For the asset AS.FILE the SFRs FDP_ACC.2/FA and FDP_ACF.1/FA ensure that non-privileged executables can only access files (AS.FILE) according to the SSP.
- For the asset AS.PORT the SFRs FDP_ACC.2/CPA and FDP_ACF.1/CPA ensure that non-privileged executables can only access communication ports (AS.PORT) according to the SSP.
- For the asset AS.INT the SFRs FDP_ACC.2/IA and FDP_ACF.1/IA ensure that non-privileged executables can only access interrupts (AS.INT) according to the SSP.
- For the asset AS.PS the SFRs FDP_ACC.2/PSA and FDP_ACF.1/PSA ensure that non-privileged executables can only access PSP-specific services (AS.PSA) according to the SSP.
- For the asset AS.CORE the SFRs FDP_ACC.2/CCA and FDP_ACF.1/CCA ensure that non-privileged executables can only access CPU cores (AS.CORE) according to the SSP.
- For the asset AS.TSF_DATA, the TSF configures the MMU to disallow non-privileged executables to access the memory of any of these other assets (i.e., the memory used for AS.TASK, AS.TSF_DATA and AS.THR). The TSF data also includes all security attributes that TSF uses to

manage any asset (e.g. security attributes for file access rights or the schedule schemes used for assignment of CPU processing time).

FIA_UID.2 ensures that partitions are identified; FMT_SMR.1 provides security roles to partitions. FMT_MSA.1 restricts the ability to write the security attributes specified in the VMIT and the property file system to system partitions. FMT_MSA.3 provides restrictive or integrator-defined default values for security attributes.

7.3.3 Security Objective: OT.RESOURCE_AVAILABILITY

- For the assets AS.MEM and AS.KMEM, FRU_RSA.2/MEM ensures that limits are enforced according to the SSP on the minimum and maximum amount of memory (AS.MEM) and exclusive hypervisor memory (AS.KMEM) available to non-privileged applications in normal partitions.
- For the asset processing time (AS.TIME), FRU_RSA.2/TIME ensures that limits are enforced according to the SSP on the minimum and maximum processing time (AS.TIME) available to non-privileged applications in normal partitions.

These limits also ensure that resources used for AS.TASK, AS.TSF_DATA and AS.THR are not depleted through operations of non-privileged executables.

7.3.4 Security Objective: OT.API_PROTECTION

FMT_SMF.1 specifies that the management API can be used for management of threads, tasks and the TOE.

FMT_MTD.1/SYS ensures that the TOE prevents access from normal partitions to the system application API. FMT_MTD.1/TASK restricts the access that a normal partition has via the normal partition API to tasks that the normal partition owns. FMT_MTD.1/THR restricts the access that a normal partition has via the normal partition API to threads that the normal partition owns. All other APIs reserved for privileged executables (PSP, system extensions and kernel device drivers) only can be reached from executables linked to the TOE.

FDP_ACC.2/MA and FDP_ACF.1/MA ensure that the TOE prevents any execution of the APIs by non-privileged applications.

7.3.5 Security Assurance Requirements Rationale

EAL 3+ has been considered appropriate to ensure the robust and reliable separation of partitions.

ALC_FLR.3 has been included to ensure that integrators understand how to submit security flaw reports to SYSGO and how to register themselves with SYSGO so that they may receive these corrective fixes. AVA_VAN.2.4E has been refined to elevate the vulnerability analysis to one required for attack potential of “enhanced-basic” in order to support compatibility with [ANSSI15].

7.3.6 Security Assurance Requirements Dependency Analysis

In this section, we provide a dependency analysis for the security assurance requirements as defined by the CC. There are no unfulfilled dependencies.

This ST claims conformance to the standard EAL 3 package. For the EAL 3 standard package, all dependencies in CC v3.1 part 3 provided packages are fulfilled.

ALC_FLR.3 depends on: No dependencies.

8 TOE Summary Specification

This section describes how each TOE Security Service defined in Section 2.4.6 is implemented and covers its SFRs.

TSS_SSA: *Separation in space of applications hosted in different partitions from each other and from the PikeOS Operating System according to the SSP*:

Applications can be hosted in different partitions. Partitions get assigned resources (i.e. space) according to the SSP, which comprise memory ranges and a set of CPUs. The TSF enforces the corresponding part of the SSP by the enforcement of access control on partition content, per-partition provision of physical memory space and allocated CPU time for each CPU.

By confining non-privileged executables into partitions, the TSF enforces that these applications can affect neither applications in other partitions nor the *PikeOS Operating System* itself.

The TSF defines separated security domains via page tables. For memory (AS.MEM) the SFRs FDP_ACC.2/MA and FDP_ACF.1/MA are implemented because the TSF configures the MMU to use these page tables to confine single load/store operations within a predefined physical memory.

For interrupts (AS.INT) the SFRs FDP_ACC.2/IA and FDP_ACF.1/IA are implemented and ensure that non-privileged executables can only access interrupts according to the SSP.

For PSP-specific services (AS.PS) the SFRs FDP_ACC.2/PSA and FDP_ACF.1/PSA are implemented and ensure that non-privileged executables can only access PSP-specific services according to the SSP.

For CPU cores (AS.CORE) the SFRs FDP_ACC.2/CCA and FDP_ACF.1/CCA are implemented and ensure that non-privileged executables can only access CPU cores according to the SSP.

FRU_RSA.2/MEM (AS.MEM and AS.KMEM) is implemented because limits on the minimum and maximum amount of memory available to non-privileged executables in normal partitions are enforced according to the SSP.

Separation in space includes access control to devices via device drivers, e.g. Ethernet or USB drivers. The drivers can be configured by the SSP to be contained in a partition.

Separation in space also includes execution of industrial APIs/libraries. These APIs/libraries can be run as non-privileged executables, for example POSIX, ARINC 653 (APEX), Linux, RTEMS, OSEK, and thus cannot bypass the SSP.

TSS_STA: *Separation in time of applications hosted in different partitions from each other and from the PikeOS Operating System according to the SSP*:

Applications can be hosted in different partitions. Partitions get assigned CPU time (i.e. time windows) according to the SSP. The TSF enforces the corresponding part of the SSP by per-partition allocation of a predefined amount of CPU time for each CPU. On a partition switch CPUs will be reused.

FRU_RSA.2/TIME is implemented because limits on the minimum and maximum amount of processing time available to non-privileged executables in normal partitions are enforced according to the SSP.

TSS_COM: *Provision and management of communication objects:*

Applications hosted in different partitions can get assigned a set of communication objects. A *communication object* is an object exposed to one or multiple partitions with access rights as defined in the configuration data, thus allowing communication between partitions.

For communication ports (AS.PORT) the SFRs FDP_ACC.2/CPA and FDP_ACF.1/CPA are implemented and ensure that non-privileged executables can only access communication ports according to the SSP.

For files (AS.FILE) the SFRs FDP_ACC.2/FA and FDP_ACF.1/FA are implemented and ensure that non-privileged executables can only access files according to the SSP.

For memory that is used as shared memory (AS.MEM) FDP_ACC.2/MA and FDP_ACF.1/MA are

implemented because the TSF configures the MMU to use these page tables to confine single load/store operations within a predefined physical memory.

For IPC and event communication, FDP_IFC.2 and FDP_IFF.1 are implemented by providing communication objects and because information flows originating from non-privileged executables are restricted to information flows allowed by the SSP.

TSS_MAN: *Management of the TOE (e.g. system partition API) and the TOE data (e.g. threads, tasks):*

The TSF protects the confidentiality and integrity of TSF data and the availability of resources.

FIA_UID.2 is implemented by requiring each partition to be successfully identified before allowing any other TSF-mediated actions on behalf of that application. FMT_MSA.1 is implemented because the TSF restricts the ability to write the security attributes specified in the VMIT and the property file system to system partitions. FMT_MSA.3 is implemented because the TSF provides restrictive or integrator-defined default values for security attributes that are used to enforce the SFP. FMT_SMF.1 is implemented because the TSF provides functions for thread management, task management and TOE management (System Partition API). FMT_MTD.1/SYS is implemented because the TOE prevents any access of a non-privileged executable to the system partition API. FMT_MTD.1/TASK is implemented because the TOE restricts the access to tasks that a non-privileged executable has to tasks that its normal partition owns. FMT_MTD.1/THR is implemented because the TOE restricts any access of a non-privileged executable the access to threads that its partition to threads that its partition owns. FMT_SMR.1 is implemented by assigning the roles “system partition” and “normal partition” and associating each partition with a role.

9 Acknowledgment

Part of this ST is based on SKPP [Inf07, LNIM10], OSPP [OSPP], HASK-PP [Bun08]. This ST has benefited from the work in the TECOM (FP7 grant 216888), SeSaM (BMBF grants 01BY1120 to 01BY1123), PASS (BMWi grant 01 MD 16002D) and EURO-MILS (FP7 grant 318353) projects.