D2.4

Guidelines to use and apply PP for all involved stakeholders

Project number:	731456
Project acronym:	certMILS
Project title:	Compositional security certification for medium to high-assurance COTS-based systems in environments with emerging threats
Start date of the project:	1st January, 2017
Duration:	48 months
Programme:	H2020-DS-LEIT-2016

Deliverable type:	Report
Deliverable reference number:	DS-01-731456 / D2.4 / 1.0
Work package contributing to the deliverable:	WP 2
Due date:	April 2018 (M16)
Actual submission date:	4th May, 2018

Responsible organisation:	SRO
Editor:	Jan Rollo
Dissemination level:	PU
Revision:	1.0

Abstract:	We explain the PP for different groups of stake-holders: MILS system integrators, MILS system ST writers, MILS system IEC 62443 evaluators, separation kernel manufacturers, separation kernel security target writers, and CC evaluators. Section 1.3 has reading instructions, how to make best use of this document, depending on what your stakeholder role is.
Keywords:	MILS multiple independent levels of safety / security, PP protection profile, separation kernel, assurance, system view

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731456.

Editor

Jan Rollo, SRO

Contributors (ordered according to beneficiary numbers)

Helmut Kurth, Andreas Hohenegger, ATSEC

Amelia Álvarez de Sotomayor, Benito Caracuel, SCHN

Alvaro Ortega, E&E

Sergey Tverdyshev, Holger Blasum, SYSGO

Tomáš Kertis, UCO

Jan Rollo, SRO

Executive Summary

We explain how separation kernels are useful for building and certifying medium- and high-assurance systems. We explain how to read a generic document used in the certification of separation-kernel-based systems, the certMILS protection profile (PP), and how to write new PP-conformant security targets for specific separation kernels.

Contents

List of Figures

List of Tables

Chapter 1 Introduction

The certMILS project (http://www.certmils.eu/) aims at easing building and verification of complex critical systems by using a certain architecture for structuring systems into partitions, called MILS (Multiple Independent Levels of Security / Safety). The verification in certMILS anchors to existing standards for security verification wherever possible. The certMILS protection profile (PP) [1] for separation kernels is such an anchor. In this document, we describe how the PP is relevant and how it is to be read and used by different audiences.

1.1 Technical baseline to understand the document

1.1.1 What are the Common Criteria for Information Technology Security (CC)

The Common Criteria for Information Technology Security Evaluation ([2], abbreviated as “Common Criteria” or “CC”) is an international standard for security evaluations. The CC ([2] , Part 1, Section 6.2) describe the roles of:

- consumers, who can use the results of evaluations to decide whether a product fulfils their security needs,
- developers, who develop a product (manufacturer), and specify security requirements characterizing a product’s security functions,
- evaluators, who judge whether a given product conforms to its security requirements
- other roles, of which for this document, we only mention:
    - evaluation authorities (certification bodies), responsible for the management and oversight of IT security programs,
    - sponsors of an evaluation (in case the evaluation is not sponsored by the manufacturer).

In this document we also use the term “CC consultant”, as person who supports the work and/or communication of any of the above roles.

The CC is being used internationally since almost two decades, and public artefacts (PPs, STs, described in the ensuing Section 1.1.2) for more than 2000 evaluated products are available at [3].

1.1.2 What is a CC Security Target (ST) and a CC Protection Profile (PP)

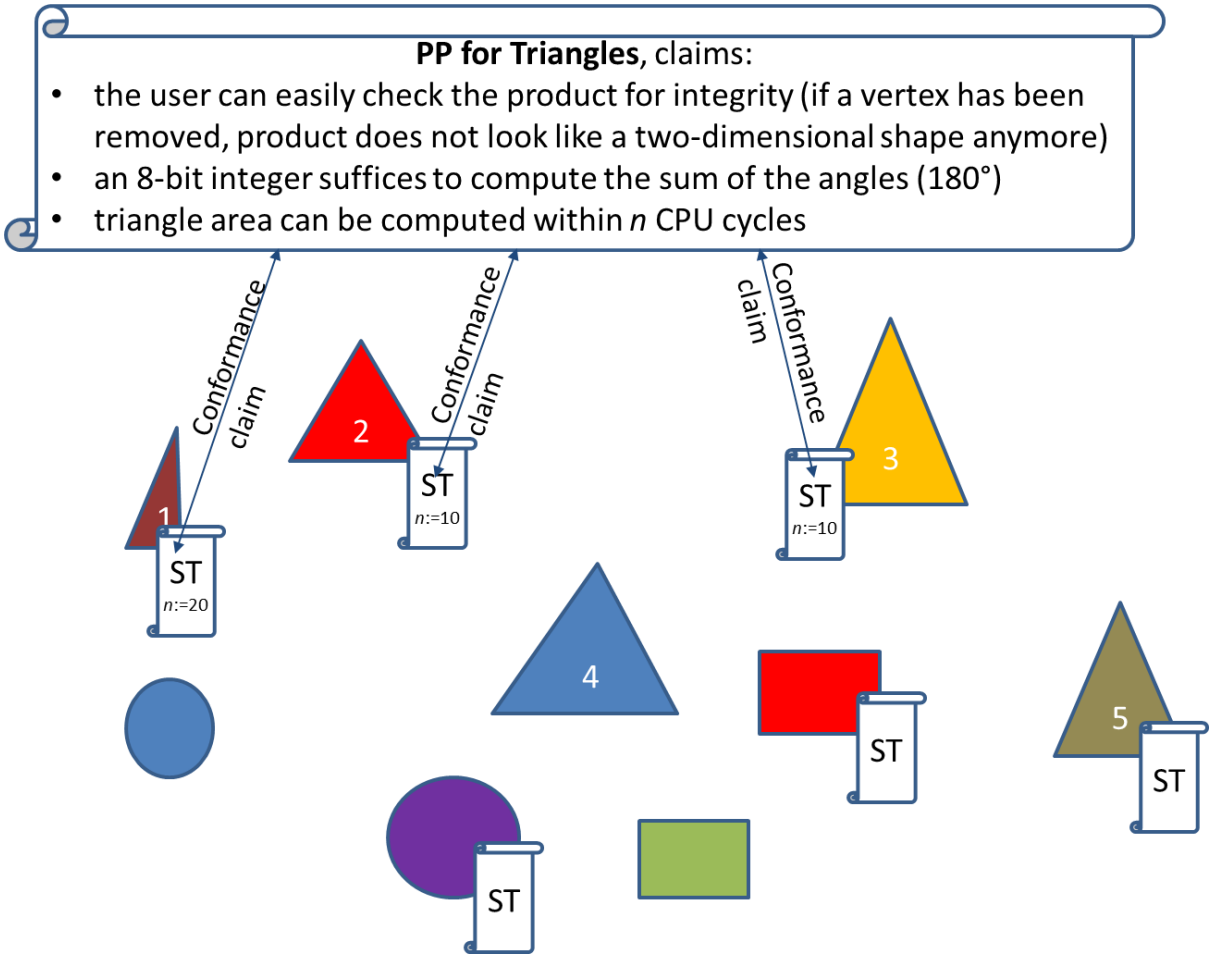

Figure 1: Products, some with STs, and a PP for triangle products to which some of the STs claim conformance

Figure 1 shows a number of shapes (triangles, rectangles and circles), that represent products on a hypothetical market. Some of the shapes in Figure 1 (not only triangles) have attached a paper, called security target (ST). The ST describes the security properties the product provides, and how that security properties are achieved at a high level. Writing an ST is the first step in a product's CC evaluation. ST writers can come from the manufacturer (at least on paper, the most typical case) and/or sponsors and/or CC consultants (an example for the latter is the ST for SuSE Enterprise Linux [4]) side. The ST is then used by the evaluator to (1) determine whether the security argument within in the ST is consistent, and (2) determine whether the product fulfils the claims made in the ST. As the decision is up to the manufacturers/sponsors, whether to perform a CC evaluation and whether to produce an ST or not, not every triangle in Figure 1 does have an ST attached to it: e.g. the manufacturer of triangle number 4 decided not create an ST.

While an ST describes an individual product, a PP is a CC document that describes a "type" (CC [2], Part 1, Section 9.3) of products. To illustrate the concept, the PP depicted in Figure 1 targets a hypothetical product class of triangles. The PP claims that for a triangle one can detect when an attacker has maliciously removed vertices (say, by visual inspection: if a vertex has been removed then the triangle is no longer a two-dimensional shape but a line), that the sum of angles (at least in a Euclidian space) is 180 degrees (which also has the beneficial side effect that each individual angle is less than 180 degrees and can be stored as 8-bit integer), and that the area can be computed in *n* CPU cycles, where the value of *n* is to be filled in by the manufacturer. A manufacturer/sponsor/evaluator who has written an ST for its products can choose to claim adherence to a PP or not. For example, manufacturer 5 has a triangle product and even an ST for

it, but has not opted to claim conformance to the triangle PP. On the other hand, not every product that has an ST can be conformant to the triangle PP: for instance the red rectangle in the lower-right middle part of Figure 1 would fail the requirement of the sum of angles (= 360 degrees) being representable by an 8-bit integer.

ST writers of product 1, 2, and 3 have opted to use the PP by means of a conformance claim. They also have instantiated the claim that the area can be computed in *n* CPU cycles with values they consider appropriate for *n*, e.g. 10 or 20 (recall *triangle area = ½ * height * width*, so this depends on how many CPU cycles a multiplication takes on a given hardware platform, whether the multiplication by ½ is implemented by the manufacturer as bit-shift operation etc.) In an ST, even when it claims conformance to a PP, it is possible to claim more than what is in the PP. For instance the manufacturer 1 might additionally claim that its triangle product has a right angle, and manufacturers 2 and 3 might additionally claim for their products that they have two triangle sides with the same length (isosceles).

Our preceding hypothetical example has shown how STs may claim to a PP or not. Switching back to the real world, on the world-wide market for products that have an ST, that ratio of products claiming conformance to a PP is about 50% [5], with individual PP adoption rates depending on communities (databases, smart cards, etc.). The certMILS PP offers the separation kernel community to standardize on a PP.

1.1.3 What is a separation kernel

A separation kernel (SK) is a special kind of operating system that allows to effectively confine applications into different containers. Each containers is called a “partition”, and partitions are denoted in green in Figure 2. The separation kernel (in blue in Figure 2) is installed and run on a hardware platform (depicted in grey in Figure 2) e.g. embedded systems or desktop class hardware). Figure 2 displays a separation kernel with three partitions on top.

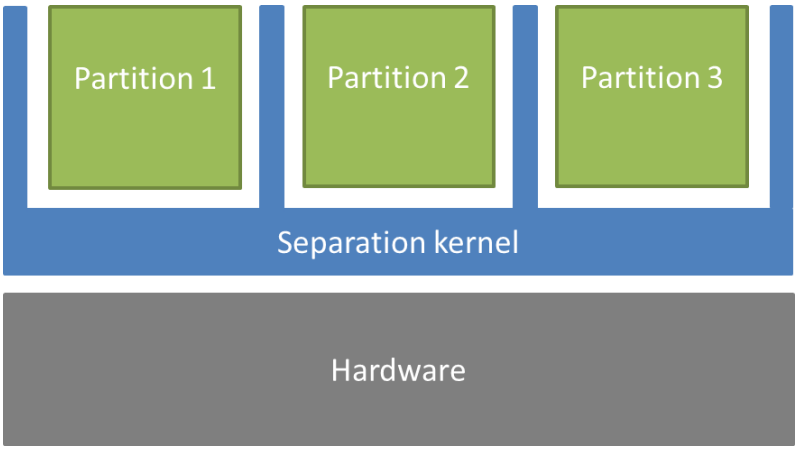

Figure 2: The separation kernel keeps partitions separate from each other.

At the next tier (a role often called “system integrator”, “integrator”, “MILS system developer”, “MILS system manufacturer” or “role performing the configuration of a system”), the separation kernel is used with a use-case specific application load. Applications for example be can be stand-alone applications, e.g. software for an actuator/a sensor, a file server, a routing system etc. But applications can also be entire “guest” operating systems. The case where applications are entire guest operating systems is called “virtualization”. Different virtualization concepts will be discussed in detail in Section 2.6.

1.1.4 Wrap-up of core terminology

Table 1 gives an overview of core terminology that is required for reading this document. Note that Table 1 only lists terms explained here in Section 1.1. For abbreviations used at other places in this document, they are usually explained where introduced, but for look-up also see the full table of abbreviations (Chapter 7).

Table 1: Core terminology used in this document

Term	**Abbreviation (if any)**	**Section the term was explained**
Common Criteria for Information Technology Security Evaluation	Common Criteria; CC	1.1.1
Integrator / system integrator	-	1.1.3
Partition	-	1.1.3
Protection Profile	PP	1.1.2
Security Target	ST	1.1.2
Separation Kernel	SK	1.1.3

1.2 Roles in MILS system development and evaluation

Figure 3 shows roles in MILS system development and evaluation.

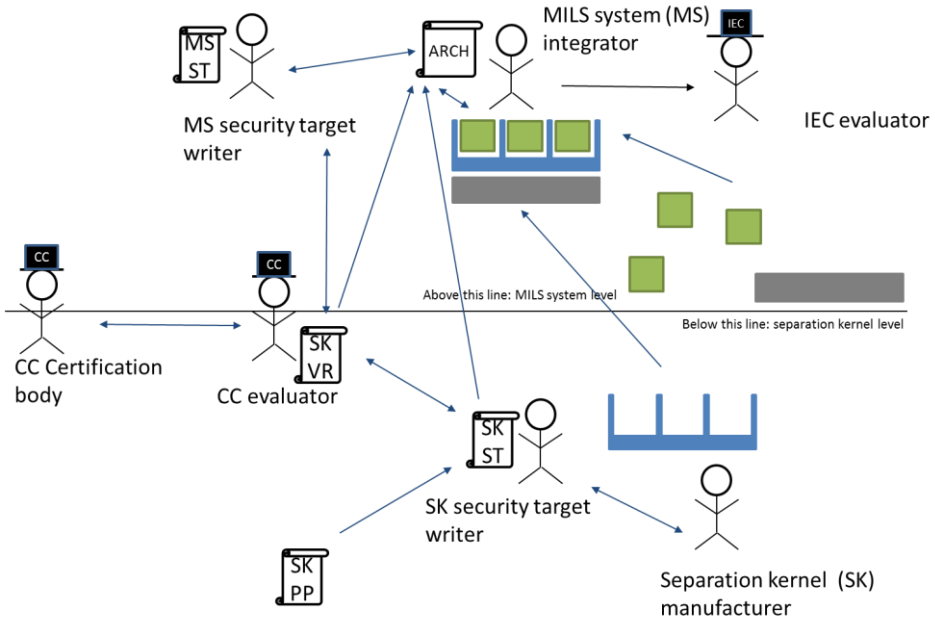

Figure 3: Roles in MILS system development and evaluation

MILS system development starts with a separation kernel that is developed (“separation kernel manufacturer” in bottom of Figure 3) and Common Criteria (CC) [2] evaluated (“separation kernel CC evaluator”). The CC evaluator, together with the separation kernel manufacturer, is in contact with the CC certification body. The figure also explicitly shows the separation kernel ST writer, who

uses the separation kernel PP (“SK PP” in the figure) and inputs from the separation kernel manufacturer and CC evaluator / CC certification body to write the ST. The role of the ST writer can be either at the separation kernel manufacturer or at the CC consultant. In the end of the evaluation, the CC evaluator produces a validation report for the separation kernel (“SK VR”), that is available to the MILS system integrator along with the ST itself.

In the next tier up (top of Figure 3), the separation kernel is used by a role often called “system integrator” to build his own system, with his own applications (green boxes) and hardware (grey box). The “system integrator” performs the configuration of a system and assigns the applications to the different partitions. By doing this, and by configuring resource access and communication channels (if any), this role defines a system security policy (SSP). Again at that level, for a CC evaluation, the MILS system integrator produces a security target of the MILS system (“MS ST” in the figure). Alternatively, we have also indicated an IEC 62443 evaluator. For an IEC 62443 evaluation it is not needed to provide a security target, however a security architecture (“ARCH”) needs to be provided. A security architecture is also needed for CC evaluation.

1.3 How to read this document, depending on who you are

Table 2 gives an overview of the sections of this document.

Table 2: Sections of this document

Number	Title	Content
Chapter 1	Introduction	This introduction, including technical baseline to understand the document.
Chapter 2	Use cases	Use cases, demonstrating what MILS systems can be used for.
Chapter 3	MILS system certification, using a PP-compliant separation kernel	How to use a PP-compliant SK in order to certify a MILS system that uses that SK.
Chapter 4	How to read the PP / PP-compliant ST(s)	How to read PPs, and STs for separation kernels. This section addresses the CC consumer of separation kernels.
Chapter 5	How to write an ST	How to write a PP-compliant ST for a separation kernel.
Chapter 6	Summary and Conclusion	Wrap-up.

How to make best use of this document depends on your role as is shown in

Table 3.

Table 3: Roles, systems, interests, and what to read first

Role	**System**	**Interests**	**What to read first**
MILS system integrator	MILS system	How to develop a certifiable (MILS) system	Start with Chapter 2. If, going “upwards”, you’d like to know how the use of a separation kernel eases certification of MILS system, then read on in Chapter 3. If, going “downwards”, you want to understand better the CC assurance of a PP-compliant separation kernel, you can continue with Chapter 4.
MILS system ST writer	MILS system	How to evaluate a MILS system for CC	See Chapter 3, in particular Section 3.1.1.
MILS system IEC 62443 evaluators:	MILS system	Howe to evaluate a MILS system for IEC 62443	Start with Chapter 3, in particular Section 3.1.2.
Separation kernel manufacturers	Separation kernel	Building a certifiable separation kernel	Start with Chapter 4. Skimming through Chapter 2 is still welcome.
Separation kernel security target writer	Separation kernel	Writing an ST, based on the PP.	Chapter 5 is specifically for you. Chapter 4 also may be useful.
CC evaluators, CC certification bodies	Separation kernel, MILS system	Getting insight on background of PP (due to familiarity with CC, probably not so much guidance is needed).	For separation kernels, in particular see Chapter 4 and Chapter 5. For MILS systems, see Chapter 3, in particular Section 3.1.1.

Chapter 2 Use cases

Use cases include, but are not limited to:

- control systems in planes, cars, trains, electrical networks, space or production facilities,
- environments that require virtualization with strong separation between virtual machines (VMs),
- embedded systems that can host multiple applications which need to be separated from each other,
- high-assurance information gateways/re-grader/firewalls/guard systems/routers,
- mobile devices with critical functions that need to be separated from the general applications (e. g. user authentication, payment, file encryption),
- systems with requirements on modular/composable assurance.

2.1 Use as small OS / security kernel

A separation kernel can be used simply as small-size operating system (OS). Its small size ensures that the entire OS code can be verified and validated rigorously. Such OS are also called “microkernel”, “microvisor”, or “security kernel”. In this scenario even a system with only one partition can be useful.

2.2 Static mixed-criticality systems

A typical use case is that of mixed-criticality systems. The configuration of such a system assigns applications of different criticality to different partitions. Mixed-criticality systems also can host applications of equal criticality level if the requirement is that these applications can interact only in a non-bypassable way. Of course, the separation kernel has to be certifiable according to the highest of the criticality levels of the applications in place.

2.3 Mixed-criticality systems with secure update

In these use cases, one or several applications, or the separation kernel itself (including its configuration), can be updated. In case of an update of the applications of a single partition, the content of a partition is updated without affecting other partitions/applications. In case of a separation kernel update, the partitioning mechanism enables and controls the update process, by e.g. implementing staged update / defence in depth. Appropriate authentication and authorization is either implemented in an application or in the separation kernel itself.

2.4 OS for dedicated security components

A separation kernel can act as OS for systems with dedicated security components (DSC), e.g. managing credentials, as described in the Common Criteria Development Board (CCDB) Working Group for DSCs essential security requirements [6].

2.5 Secure use of new functionality or legacy software

A separation kernel can be used to encapsulate into partitions a new functionality, which shall not interfere with an existing, potentially already safety or security-certified system. This additional functionality could, for instance, be a monitoring function, when an existing device is connected to the Internet.

In this case the separation kernel provides the basis for a virtualization system where an adapted general-purpose OS can execute within a partition and provide the same functions to its applications as the same OS executing directly on a hardware platform.

The other way round, it is possible to combine legacy software or interfaces with new higher-assurance components, thus restricting the need for certification to the higher-assurance components.

Mixed-critical systems can have different life-cycles:

- From-scratch-scenario: This scenario of introducing partitioning is applied when developing an application from scratch.
- Incremental scenario: This scenario is incremental: a new functionality (e.g. watchdog, customization interface) that is run on the same hardware is assigned a new partition once it is added to the system.

2.6 Types of applications

An application is associated to an executable binary. In a very general view, in a separation kernel, an application can be stand-alone applications, compiled from a system programming language [7] such as C, C++ or Ada (Figure 4, left-most partition, with two stand-alone applications) or OS also called “guest operating system”. OS can be virtualized or paravirtualized:

- A paravirtualized OS has adaptations, so that it e.g. can run in user-mode on the separation kernel.
- A fully virtualized OS is running in an (unchanged) form in which it also would run directly on some other hardware. For secure full virtualization with good performance, usually hardware support for an additional privilege level is needed that is reserved to hypervisors (e.g. Cortex A15 Soc provides hardware virtualization extensions [8]).

Figure 4 shows a fully virtualized OS (middle circle) and a para-virtualized OS (circle on the right)..Note that even where stand-alone applications are used, it might be enough to just have one stand-alone application in a partition. Conversely, a separation kernel could also allow a combination of stand-alone applications and (para-)virtualized OS within the same partition. For completeness and disambiguation, Figure 4 also shows the application within the (paravirtualized) operating system as circles with the legend “App”. However, the separation kernel does not need to be aware of these inner-OS applications and in general, when we speak of application in this document, we mean either “stand-alone applications” or entire guest operating systems, but not the applications maintained by the guest operating systems.

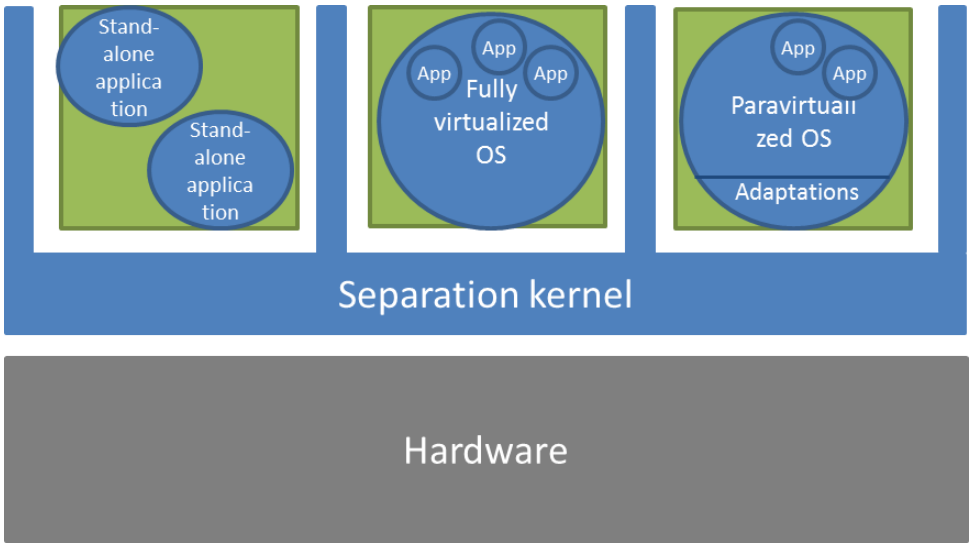

Figure 4: Different types of applications

A separation kernel product can be designed to be agnostic of the actual load, but it also may provide specialized support for certain application use cases (e.g. full virtualization might make want to use of platform-specific features).

In the following we will not always draw applications, in order not to clutter the figures. However, when something (a resource, an interface) is connected to a partition, this is with the understanding that the resource is always accessed from an application within that partition.

2.7 Partitioning examples

The MILS Protection Profile (PP) according to CC speaks about partitions [1]. One of the most important decisions to be made when using a MILS system, which is completely independent from choice of a separation kernel, is how to do partitioning. In the following we show published examples for partitioning layouts. This section shows some more sophisticated partitioning examples. It also can be skipped on a first reading.

2.7.1 Using partitions to represent zones

Partitioning can be used to represent different security zones (in the sense of the Aeronautical Radio Inc. (ARINC) avionics standard on security zones in aviation (ARINC 811 [9]) or IEC 62443-1-1 [10] , page 46-47).

2.7.1.1 EURO-MILS avionics demonstrator

Figure 5: EURO-MILS avionics gateway [11]

In Figure 5 the environment of the EURO-MILS avionics gateway [11] is depicted that connects different ARINC 811 domains [12], e.g. a high security level domain for the aircraft control domain and a low security domain for passenger-owner devices. Figure 6 gives an interpretation how these security domains of the gateway's environment are represented by different partitions. Each component connected with arrows, "Receiver Component", services of a network interface card "NIC" etc. is a partition. By the underlays in green and brown it is indicated that some partitions can

be seen as representing different domains (marked as domain A and domain B), that is the gateway's environment is represented by the gateway's partitioning.

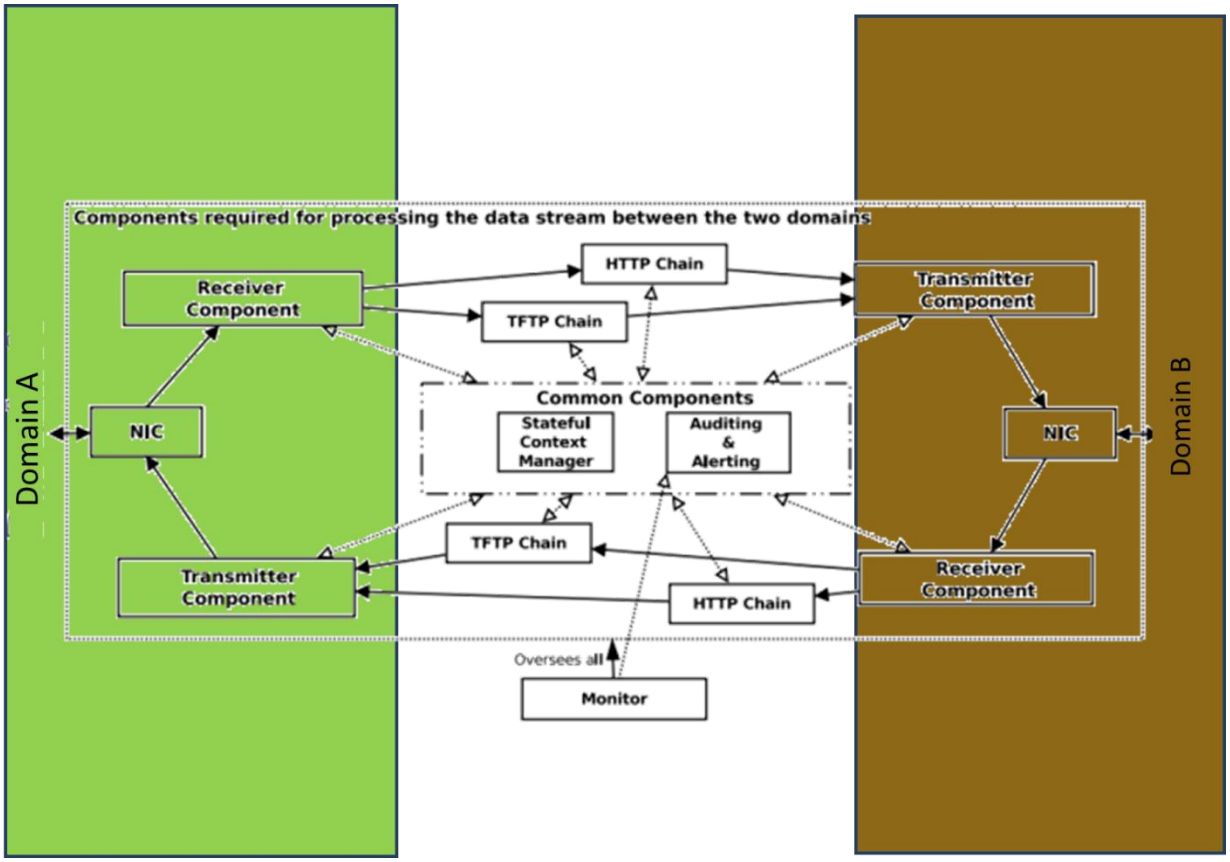

Figure 6: EURO-MILS avionics domains, based on [11], with additional color markup.

2.7.1.2 EURO-MILS automotive demonstrator

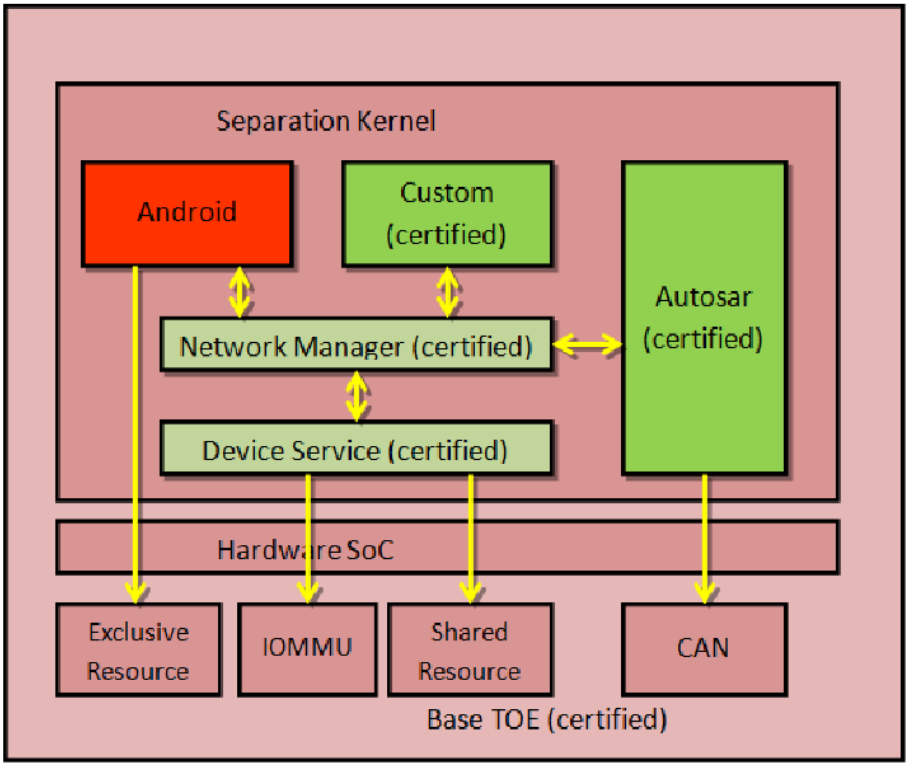

Figure 7: EURO-MILS automotive demonstrator, based on [13]

Figure 7 shows the EURO-MILS automotive demonstrator design [13]. The component is connected to the insecure Internet via Android and to the internal car control via the controller area network (CAN) bus. Again, the rectangles in the figure represent partitions ("Android", "Custom (certified)", "Autosar (certified)" etc.). The red partition is the internal representation of the insecure Internet. The green partitions are the internal representation of the car's control which needs to be protected.

2.7.1.3 certMILS brainstorming exercise

Table 4 summarizes a brainstorming exercise where similar zoning break-ups were proposed for possible component break-ups at the July 2017 certMILS technical meeting in Vienna.

Table 4: Brainstorming on zones done on whiteboard at certMILS Vienna meeting

Component	Zones (e.g. according to 62443 [10], or EN 50159 [14])	Separation kernel partitions
Component: Gateway connecting cat 3 network with cat 2 network	cat 2 network zone, cat 3 network zone	cat 3 partition, cat 2 partition
Component: Gateway connecting cat 2 network with cat 1 network	cat 1 network zone, cat 2 network zone	cat 1 partition, cat 2 partition
Component: railway gateway	safety zone, non-safety zone	security partition/safety partition
Component : Remote Terminal Unit for electrical substation	control; acquisition; communication	control partition; acquisition partition; communication partition

2.7.2 Representation of stakeholders

Another use of partitions, that was for discussed within certMILS, is to have different partitions for different roles. Figure 8 shows how different partitions could be assigned to different roles. E.g. a partition is assigned to a device manufacturer, another partition is assigned to the device's integrator (so that any possible integrator interference to the manufacturer partition could be controlled), and a third partition is assigned to an operator (so that any possible operator interference to the manufacturer and integrator partitions could be controlled). It has been discussed that also designs where there is "only" a split-up into manufacturer and integrator partition could be realized in a first step.

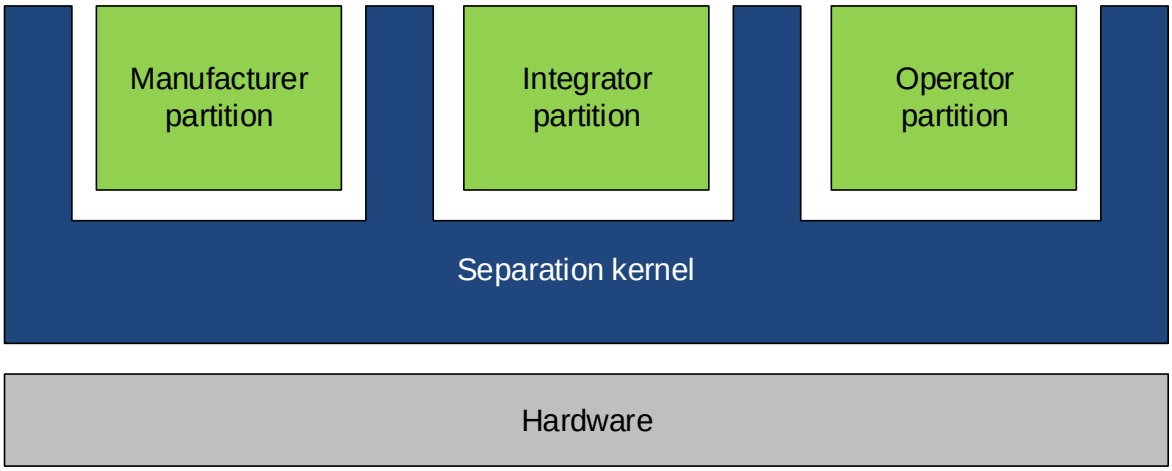

Figure 8: Partitioning based on source of content

Chapter 3 MILS system certification, using a PP-compliant separation kernel

SK PP compliance makes the certification your MILS system easier (Section 3.1) and we point to the certMILS security architecture templates in order to support your work of giving a security argument for composed system (Section 3.1.4).

3.1 What does SK PP compliance mean for CC, IEC 62443 and other certifications of composed products

There are different security certification standards such as CC (generic IT products), IEC 62443 (IoT/critical systems), and more very recently upcoming domain-specific standards (such as automotive J3061 [15]). The first two will be addressed in this section, as they are relatively advanced. Even if you want to security-certify to another standard, we expect that you will be able to interpolate from both approaches (for some hints on that, see Section 3.1.3).

The certMILS security architecture template (discussed in Section 3.1.4) can be applied to all system certifications that require a structured security architecture argument.

3.1.1 CC

If you do a CC certification of a composed system, then for the composed system you can use the separation kernel in two different approaches, formal or informal approach.

3.1.1.1 Formal composition approach

You adopt a formal approach based on the CC composite approach (in its CAP or smart card approach, as detailed in certMILS D1.1 [16]).

3.1.1.2 Informal composition approach

Here, you adopt an approach where the separation kernel is used more informally as a subsystem or module of the composed system, then you can use the security architecture templates (see Section 3.1.4). This approach is feasible if you want to avoid the overhead of a composed CC evaluation.

This approach also can be applied to use an SK certification with a given Evaluation Assurance Level (EAL) level for a MILS system evaluation with potentially higher EAL levels due to the following reasons:

- As the separation kernel only becomes a subsystem or module of the MILS system, the granularity of the separation kernel architecture automatically becomes upgraded. Example: if the separation kernel architectural description (ADV_ARC, ADV_TDS in CC terms) is at the level of subsystems (e.g. in EAL3), then in the composed system, when the separation kernel would be “downgraded” to a subsystem, the previous subsystems of the separation kernel become refined to modules. This means that the description and testing of the separation kernel will become automatically more fine-granular, giving you credits for some additional CC work units, where module-granularity (and not only subsystem-granularity) is required.
- The CC vulnerability analysis (CC AVA work unit) of the composed system still has to take into account the elevated vulnerability potential of the composed system anyway, but can

do this based on a particular configuration of the separation kernel, reducing the evaluation space.

3.1.2 IEC 62443

If you do an IEC 62443 certification of a composed system, then for the composed system you can use the separation kernel in two different approaches, formal or informal approach.

3.1.2.1 Formal composition approach via IsaSecure

The sufficiency of CC certification for operating systems is indicated in IsaSecure's guideline for the application of IEC 62443-4-1 [10], called SDLA-312 [17]. SDLA-312 [17] contains a table of requirements. In the table entry with requirement ID "SDLA-MIV-5", with title "Module Implementation & Verification", it is written that "*If the product includes a Commercial off the Shelf (COTS) operating system, then the operating system shall either meet the requirements of this development phase or be certified to Common Criteria EAL 3 or higher or be certified to a comparable security standard, or compensating controls must be included in the product to ensure that security vulnerabilities in the operating system do not result in vulnerabilities above a certain severity level in the product.*" As separation kernels are small operating systems, they can be subsumed under this guidance. Indeed, the small code size of separation kernels even is a strengthening factor here.

Note that at the moment, the IsaSecure approach is just one of several proposals on how to realize IEC 62443 [10] evaluations, so it is an open question, whether a simple reference to the above SDLA-312 [17] text would suffice also in the other approaches.

3.1.2.2 Informal composition approach

certMILS considers pursuing an approach where CC separation kernel assurance is used informally for 62443-based assurance, where the separation kernel supplier provides support to the MILS system integrator (Figure 9), who himself acts the supplier of a product for an ICS.

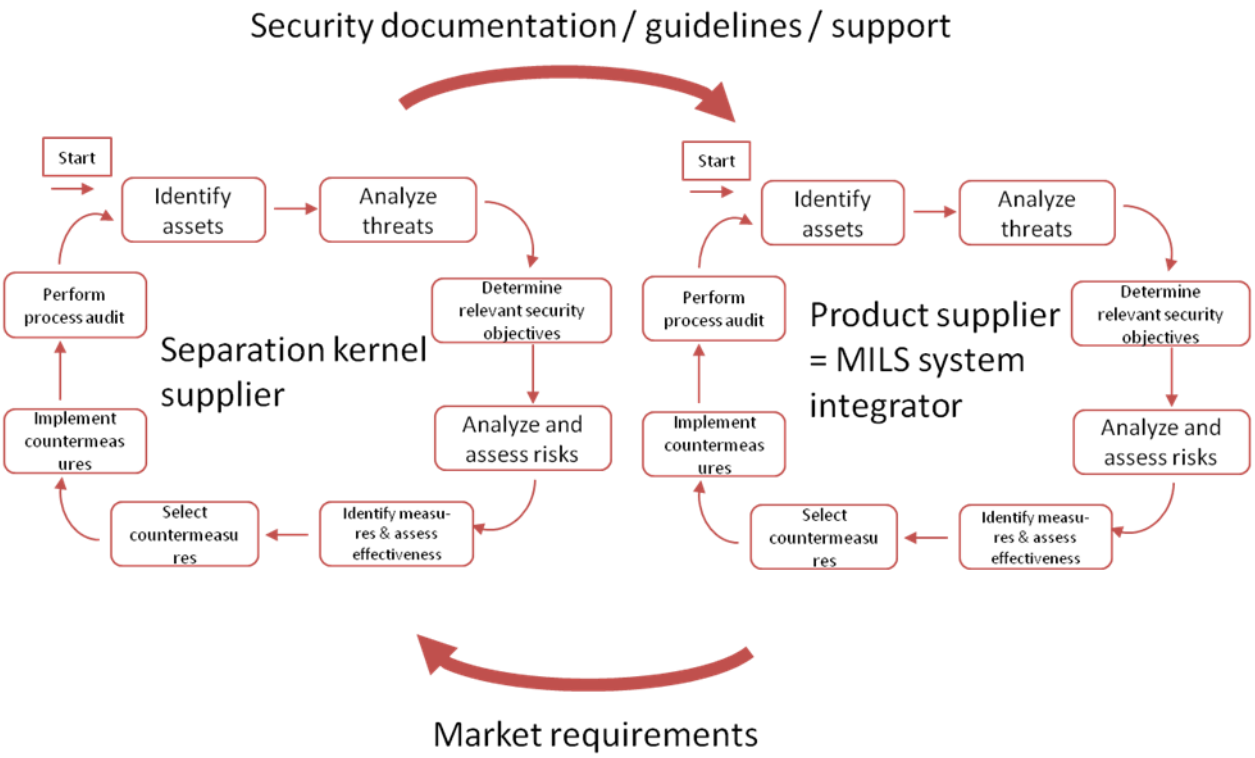

Figure 9: Chained security lifecycles in a MILS system, from D1.3 [18]

As explained in certMILS D1.3 [18], Section 4.3, different IEC 62443 [10] process and product requirements such as IEC 62443-4-1 process requirements as SR-2, SD-1, SD-2, SD-6 and IEC 62443-4-2 product requirements CR 5.1, CR 7.1 and CR 7.2 can be well mapped to the assurance obtained by a separation kernel.

Table 5: Mapping of CC activities to IEC 62443, from D1.3 [18]

CC activity	IEC 62443 counterpart
Development (ADV)	IEC 62443-4-1 Practice 3 Secure by Design; Practice 4 Secure Implementation
Testing (ATE)	IEC 62443-4-1 Practice 5 Testing
Vulnerability analysis (AVA)	IEC 62443-4-1 SV-4 Penetration testing
Configuration management capabilities (ALC_CMC)	IEC 62443-1-1 Section 8 Lifecycle
Configuration management scope (ALC_CMS)	IEC 62443-1-1 Section 8 Lifecycle
Delivery (ALC_DEL)	IEC 62443-1-1 Section 8 Lifecycle
Development security (ALC_DVS)	IEC 62443-4-1 SM-7 Development environment security
Life-cycle definition (ALC_LCD)	IEC 62443-1-1 Section 8 Lifecycle
Flaw remediation (ALC_FLR)	IEC 62443-4-1 Practice 6 Security defect management

If you plan to use a CC separation kernel in order to reason about the architecture of an IEC 62443 system, again, the security architecture templates from certMILS are available to support you (see Section 3.1.4).

3.1.3 Other standards

Other domain-specific standards that are based on IEC 62443 also suggest to the evaluator to take into account existing other certifications (i.e. these other certifications could be the SK's CC certification). For instance, for the railway sector, VDE 0831-104 [19], p. 19 states for compatible certifications that "*it is assumed that in future railway signalling systems components with IT security tasks are usually purchased on the marked and not developed in-house. It is assumed that such components already have a certification compatible with IEC 62443. If this does not apply, after checking the requirements one also could accept other certificates or compare with a reference system*". We understand this that CC evidence can be used in a VDE 0831-104 [19], railway certification context. For building a generic security argument, we refer you to the standard-agnostic security architecture template (see Section 3.1.4).

3.1.4 Use of security architecture template

certMILS provides support for your compositional SK-based certification not only conceptually, but also by a very practical artefact, that is a security architecture template. The security architecture template comes with an explanation how the security architecture can be used to certify the product described by the security architecture according to the standards CC and/or IEC 62443 [10].

For a security evaluation according to CC, IEC 62433, or another standard, you will have to demonstrate that the architecture of your product provides clear domain separation, self-protection and non-bypassability. If you use a separation kernel, then technically your arguments are very likely using the technical properties of the separation kernel, regardless what for the separation kernel is used.

Hence the certMILS project is providing a security architecture template that you can use as snippets / text blocks for your own system's security architecture, see certMILS deliverable D2.3 [20]. This security architecture template has been tested for writing the security architectures of the certMILS railway, subway, and smart grid prototypes.

Chapter 4 How to read the PP / PP-compliant ST(s)

The PP is intended to span a large range of separation kernels. This chapter discusses how to read the PP / PP-compliant ST(s) in CC documents. This section addresses the CC consumer. It is a guide how to understand texts that you will be likely to encounter while reading a PP-compliant ST and / or PP (both the base PP and the PP modules) and what they mean for comparing separation kernels.

4.1 PP versus ST: which one to read?

Formally, there is a situation where reading the certMILS PP [1] is completely irrelevant for a system integrator role defined in Section 1.2. This situation is characterized by the following:

(1) the system integrator has already decided which separation kernel he/she uses
(2) the separation kernel has its own CC-certified security target (ST) that is compliant to [1]

In this is situation, the ST already has taken care of the compliance to [1], and is probably more detailed than the necessary high-level PP.

However, other situations / concerns exist:

- the PP facilitates the comparability of different separation kernels evaluated according to it
- the system integrator wants to avoid vendor lock-in, and hence sees the PP-guaranteed properties as common baseline for his product
- the separation kernel to be used has not yet been certified according to the PP [1], but the vendor intends to do so
- the PP also can be helpful to have a different (more abstract) view on the product than the ST (see Section 1.1.2)

If one of above holds, then read on, as the rest of this section is written for this case.

4.2 Understanding MILS system diversity

4.2.1 Systems with trusted partitions

Some separation kernel implementations allow trusted partitions. These may allow their host applications to partially or fully circumvent the system security policy (SSP), e.g. to change scheduling or reassign memory. This kind of set-up can be useful to implement custom monitoring and control functionality. The separation kernel shall still control the untrusted partitions, and also control those parts of the SSP that the trusted partitions cannot bypass.

4.2.2 Common OS abstractions: Threads, tasks

Partitions are a concept unique to separation kernels. However, a separation kernel may optionally support additional interfaces mainly borrowed from general-purpose operating systems.

Thread: A thread is conventionally the smallest schedulable entity. That is, for a thread an OS maintains a CPU register context (i.e. the state of all CPU registers, often a one- or two-digit number), that can be stored away when the thread is scheduled away and be restored when the thread is scheduled in again. User programmable activities such as application code must be done

in a thread. Some separation kernel also might put exception handlers into threads. Also, for the virtualization of guest OS (virtual CPU), some separation kernels can opt to use one or many threads.

Address space/task: A virtual address space is an abstraction of physical memory, provided by a memory management unit (MMU). For separation kernels that are based on a memory protection unit (MPU), the concept of address spaces does not exist, unless similar concepts are provided by measures such as address rewriting at the compiler/linker level. In the following we describe the MMU case: The MMU manages chunks of physical memory, usually called pages. At the time of writing, on many systems, the size of a page is 4096 bytes. The MMU assigns on-the-fly pages to "virtual" addresses when access to any memory address is requested by application code. Address spaces make programming and deployment easier, as it is no longer needed to compile the applications into different memory regions. In the separation kernel and embedded systems domain, also often the synonym task is used.

Application and its relation to tasks/threads: An application (see Section 4.2), in a MMU-based system is typically associated to at least one task and (where the thread abstraction is used) at least one thread. The typical use-case is that when the application is running, the separation kernel uses the assigned quotas of threads and tasks to invoke a static configuration, that is when the application invokes a service call to start a new thread or new task it is checked that the assigned quotas are not exceeded. While in principle, other designs are technically possible, an often-encountered hierarchy is to:

- assign applications to a partition
- assign tasks to an application
- assign threads to a task

For example, in a separation kernel that does not support tasks, but that supports threads, applications and partitions, the separation kernel's design hierarchy would skip the task step, that is assign applications to a partition and assign threads to an application.

In a separation kernel that does neither support tasks nor threads (e.g. real-time operating system – RTOS), applications are simply assigned to a partition, and that's all.

Of course, even in systems built on separation kernels that have all these abstractions (i.e. partitions, applications, tasks, and threads), a system integrator may opt not to make real use of them for a given deployment, as illustrated by Figure 10.

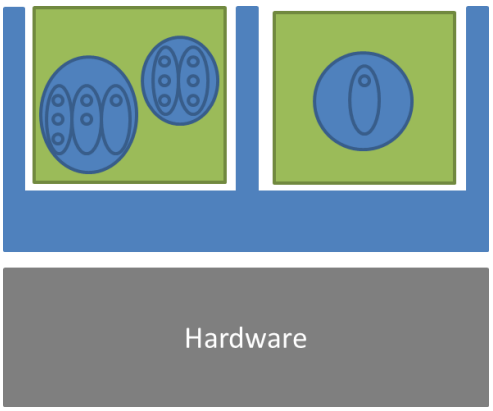

Figure 10: Threads, tasks, applications, partitions

Here the left side of Figure 10 shows a partition with two applications, containing 3 and 2 tasks. The individual tasks contain 3, 2, 1, 3, and 3 threads, respectively. On the right-hand side of Figure 10, there is a partition with just one application, which has just one thread that contains just one task.

4.3 SK properties evaluated in CC configuration

The minimum separation kernel (SK) properties that are evaluated, are separation in space and time. Separation means that resources are kept confined, unless their sharing is explicitly configured by the system integrator.

Separation kernels, as addressed by this document, can be used as a basis for systems that need to isolate different applications executing on the same platform from mutual influence. The separation kernel may allow / disallow the controlled flow of information between them.

A separation kernel at a minimum must be able to separate the partitions from each other, such that they may not share memory and also control all ways they might use to communicate. Further, it must ensure a minimum quota of each resource for a partition, that is, at least a minimum amount of main memory and a minimum quota of CPU time within a given time interval.

Generic security features implemented in a separation kernel are:

- separation in space: access control to memory assigned to partitions,
- control of communication between partitions,
- separation in time: CPU timeslices and/or CPU priorities and/or fixed allocation of partitions or applications to CPUs,
- non-bypassable, evaluable (minimal trusted computing base), always invoked, tamperproof.

The purpose of a separation kernel is to provide a basis for multiple partitions, each of them equipped with a set of resources. A separation kernel can leave the implementation of higher level functions, usually provided by common operating systems, like file systems, network protocols and application management, to the partitions. As such, a separation kernel can be seen as an operating system with minimized functionality.

However, there are cases where a separation kernel goes beyond the minimum functionality and that additional functionality also shall be CC-evaluated. For such use-cases, an ST can claim conformance to PP modules that extend the basic PP functionality, such as:

- Cryptographic services
- Secure boot
- Secure update
- Storage
- CPU time
- Network interface partitioning
- Information flow
- Management module
- Secure audit
- I/O MMU
- Hardware abstraction layer (HAL)

Having certain functionality evaluated at the separation kernel level, might make use of the separation kernel easier if the use case matches the evaluated functionality, because then that evaluated functionality needs no longer be trusted. On the other hand, the trade-off is that the more functionality there is to be CC-evaluated, the harder it is to get it evaluated to a high assurance level. Therefore, an SK vendor might opt not to certify certain functionality, even if it is available or could easily be implemented, which of course means that the non-certified functionality needs to be trusted without certification (the integrator may provide some other arguments to the user of the product based on the TOE).

Moreover, keep in mind that there might be the option to implement certain functionality in untrusted partitions rather than the SK itself. A use case is e.g. a cryptographic library that is used

locally from within only an untrusted partition, e.g. think of an untrusted partition that would connect to web a server via https.

4.4 SK scope evaluated in CC configuration

A separation kernel is a generic product, and it can be used in many kinds of MILS systems in many kinds of industry domains.

4.4.1 Security mechanism are enforced towards untrusted applications

In a CC evaluation, obviously the evaluator cannot try out all possible applications. However, what an evaluator can do is to check that security mechanisms are in place, and that untrusted applications cannot tamper with or bypass these security mechanisms

4.4.2 Trusted applications have to be evaluated by the system integrator

In other cases, a separation kernel may not be able to ensure that applications do not misuse privileges that allow bypassing the security policy of the separation kernel:

- Applications in the partition have been assigned access to separation kernel APIs that allow bypassing of partitioning (e.g. the ability to inspect the status of other partitions). Such privileged partitions still might be useful for implementing auditing or monitoring functionality, but of course they cannot be used to run arbitrary untrusted code.
- Applications in the partition have been assigned resources that could bypass the separation kernel. Such resources could be devices that have access to Direct Memory Access (DMA) and no I/O MMU is in place or devices that can overwrite volatile or non-volatile memory areas of the separation kernel.

Note that, while a separation kernel may not be always be able to control trusted applications, a separation kernel is still expected to provide technical means to protect trusted applications from untrusted applications.

4.4.3 (Optional) exemption of hardware abstraction layer

The base PP exempts a component called HAL (Hardware abstraction layer) from the separation kernel. HAL contains a set of drivers for specific hardware components and is supplied and approved by the integrator. The HAL is in the same security domain as the separation kernel. A HAL typically provides access to (or abstracts) hardware components such as e.g. timers or other types of interrupt handling. The HAL is protected from non-privileged applications by access control and resource management enforced by the separation kernel.

The HAL ensures that the separation kernel can execute with a given hardware. The HAL is invoked right after the bootloader, it sets up interrupt vectors and memory layout. At run-time it may provide HAL-specific services, e.g. to trigger the hardware to restart the MILS platform. At run-time the HAL also does initial interrupt handling (including timer interrupts). This initial interrupt handling then invokes other parts of the HAL for further processing. Such further processing can be, for timer interrupts, feeding the interrupt to a platform-independent scheduler implemented in the OS, or in other cases even a handler within a partition the interrupt is associated with.

The reason why the HAL can be exempt from the separation kernel certification is not so much technical, but rather due to product line maintenance: timers and interrupt handling details are not part of the (relatively stable) CPU architecture, but may differ with each board, that is the HAL allows some fine-tuning of the system. In desktop systems, the BIOS / UEFI plays a similar role.

Moreover, a HAL is mostly not defining its own security policies, but just implementing the security policies defined in generic SK at the lowest board-specific level (e.g. details of interrupt controller configuration). Hence, in the base PP, the HAL is intentionally omitted from the boundary of what is certified.

Given an SK and a HAL, there are the following strategies that can be followed to gain assurance:

1. You can carefully check the code of the HAL
2. Your SK vendor has carefully checked the code of the HAL
3. Your SK vendor has CC-certified the HAL standalone, and you do compositional CC certification
4. Your SK vendor has CC-certified the HAL as part of an ST, that used the PP HAL module

The first two items would not be considered as providing assurance in the CC sense. But, technically, all of the above options, might achieve the same degree of assurance.

Obviously, the SK depends on the HAL. However, the HAL also depends on the hardware itself. At the time of writing, COTS hardware is almost never CC-certified and needs to be trusted (one can discuss whether this should perhaps change in the future). Our position is that having, in form of a SK, some CC-certified base in a system is a good starting point for establishing its security. Having the HAL certified is nice to have, but always demanding it would be counter-productive to flexible market adoption of separation kernels, and thus not benefit the global cause of security at all. For instance, the recent spectre and meltdown findings stem from security in COTS hardware flaws indirectly affecting many CC-certified systems. Hardware findings will continue to be found, but this should not lead to that one gives up on the design of secure software systems.

4.5 Side channels, residual information flow

Information flow [21] can be defined by that a partition can directly or indirectly learn about the properties of other partitions. Some kinds of information flow can be quite innocent, e.g. a partition that is cyclically scheduled might detect this fact by simply asking for system time stamps.

In a scenario where information flow can be exploited, you often have two colluding partitions, i.e. partitions which cooperate to bypass the separation kernel's security policy. For instance, assume that partition 1 and partition 2 are supposed not to communicate with each other. If both partition 1 and partition 2 can obtain exclusive access to resource R, then during an agreed time interval, say 1 microsecond, partition 1 can choose to block R, to signal e.g. a "1" to partition 2 and partition 1 can choose not to block R, to signal e.g. a "0" to partition 2. The role of partition 2 is to "read out" the blocking status by trying to obtain R while partition 1 operating (if "1" is to be transmitted) or not operating on the resource (or if "0" is to be transmitted, not done).

Resource sharing between partitions also can come from the underlying hardware, and in some cases, for certain hardware and its uses, especially in the multicore domain, information flows via resource sharing conflicts might be unavoidable. For a measurement of such information flows on a separation kernel see Kuzhiyelil and Tverdyshev [22], for a general discussion in the MILS context we refer to Alves-Foss et al. [23] and Jean et al. [24]. Kemmerer [25] [26] gives a generic methodology to assess information flows.

However, in many safety standards, separation in space and time does not contain the aspect of collusion or even confidentiality, but rather focuses on that separation in space and time guarantees integrity, as can be seen in Table 6.

Table 6: Space and time partitioning in different safety standards, tabulated in [27]

Standard	**Space**	**Time**
IEC 61508 [28], Part 3, Annex F (F2, F4,	"Spatial: the data used by one element shall not be changed by another element. In particular, it shall not be changed by a non-safety related element.	"Temporal: one element shall not cause another element to function incorrectly by taking too high a share of the available processor execution time, or by blocking execution of the other element by locking a shared

Standard	Space	Time
F5)	a) Use of hardware memory protection between different elements, including elements of differing systematic capability. b) Use of an operating system which permits each element to execute in its own process with its own virtual memory space, supported by hardware memory protection. c) Use of rigorous design, source code and possibly object code analysis to demonstrate that no explicit or implicit memory references are made from between software elements which can result in data belonging to another element being overwritten (for the case where hardware memory protection is not available). d) Software protection of the data of a higher integrity element from illegal modification by a lower integrity element."	resource of some kind. a) Deterministic scheduling methods. For example, a cyclic scheduling algorithm which gives each element a defined time slice supported by worst case execution time analysis of each element to demonstrate statically that the timing requirements for each element are met; time triggered architectures. b) Strict priority based scheduling implemented by a real-time executive with a means of avoiding priority inversion. c) Time fences which will terminate the execution of an element if it over-runs its allotted execution time or deadline (in such a case, hazard analysis shall be undertaken to show that termination of an element will not result in a dangerous failure, so this technique may be best employed for a non-safety related element). d) An operating system which guarantees that no process can be starved of processor time, for example by means of time slicing. Such an approach may only be applicable where there are no hard real time requirements to be met by the safety related elements, and it is shown that the scheduling algorithm will not result in undue delays to any element."
ISO 26262 [29], Part 6, Annex D	"Memory: With respect to memory, the effects of faults such as those listed below can be considered for software elements executed in each software partition: • corruption of content; • read or write access to memory allocated to another software element. EXAMPLE Mechanisms such as memory protection, parity bits, error-correcting code (ECC), cyclic redundancy check (CRC), redundant storage, restricted access to memory, static analysis of memory accessing software and static allocation can be used." (Note: SIL D needs hardware support, see Part 6, 7.4.11.)	"Timing and execution: With respect to timing constraints, the effects of faults such as those listed below can be considered for the software elements executed in each software partition: • blocking of execution; • deadlocks; • livelocks; • incorrect allocation of execution time; • incorrect synchronization between software elements. EXAMPLE Mechanisms such as cyclic execution scheduling, fixed priority based scheduling, time triggered scheduling, monitoring of processor execution time, program sequence monitoring and arrival rate monitoring can be considered."
DO-178 [30], Section 2.4.1	"A partitioned software component should not be allowed to contaminate another partitioned software component's code, input/output (I/O), or data storage areas." (Note: A more explicit distinction between space and time can be found in ARINC-653 [12].)	"A partitioned software component should be allowed to consume shared processor resources only during its scheduled period of execution."

Another reason why limitation / elimination of information flow has not been included in the PP is that the PP shall not be restricted to high-level EAL levels. This is in line with the CC Recognition Arrangements (CCRA's) collaborative PP (cPP). The cPP guidance suggests to start at EAL2 for cPPs at least until "*the International Technical Community can demonstrate a rationale that activities up to and including EAL4 can be reproduced between schemes*". [31] However, source code only is delivered to the evaluator from medium EAL levels on (i.e. EAL4 and higher). It is simply not possible for an evaluator to evaluate information flow without source code and adequate resources for vulnerability testing. The same argument applies to residual information protection (FDP_RIP in CC terms). Note that however, even if it is not hardcoded in the PP, an SK vendor is always free to make additional statements on such properties in his SK's ST, if he wishes to strengthen the SFRs.

4.6 EAL level

The certMILS PP suggests to use EAL 4 augmented with ALC_FLR.2. Of course, an ST writer is generally free to go higher than the assurance levels or assurance components claimed in the PP. In particular, an ST that claims conformance to the certMILS PP can do this. The intention is here that we view the basic separation kernel functionality specified in the PP to be largely independent from assurance activities (EAL levels) and we do not want to hardcode a high EAL level before having received more community feedback. EAL 4 is motivated from a protection profile (OSPP [32]), that is used for desktop operating systems and used e.g. by Linux, Windows and MacOS. We think that separation kernels, with their smaller code base, can achieve at least as much evaluation assurance as their much bigger desktop counterparts. ALC_FLR.2 is motivated from its wide usage. However, as we are aware that EAL levels can spark discussion, at this point, we like to point out that, beyond the certMILS PP deliverable [1], you can discuss our current "Essential Security Requirements" (ESR) draft, including EAL level, which tries to make the certMILS insights on PP available for community review while giving it a less fixed format than the certMILS draft on the CC Users Forum (http://www.ccusersforum.org/).

4.7 Related systems, related work

We discuss related systems (Section 4.7.1) and related work on separation kernels (Section 4.7.2).

4.7.1 Type 2 hypervisors

Hypervisors have been classified as type 1 (bare hardware) and type 2 (application running on top of an OS) [33]. Typical MILS separation kernels are type 1 systems. Figure 11 shows another setup, where the virtualization is provided by a type 2 system, where a virtualization application runs on top of an operating system. Note that Figure 11 has an additional layer, if you compare it with Figure 2. An example of the type 2 approach is for example Oracle's VirtualBox software.

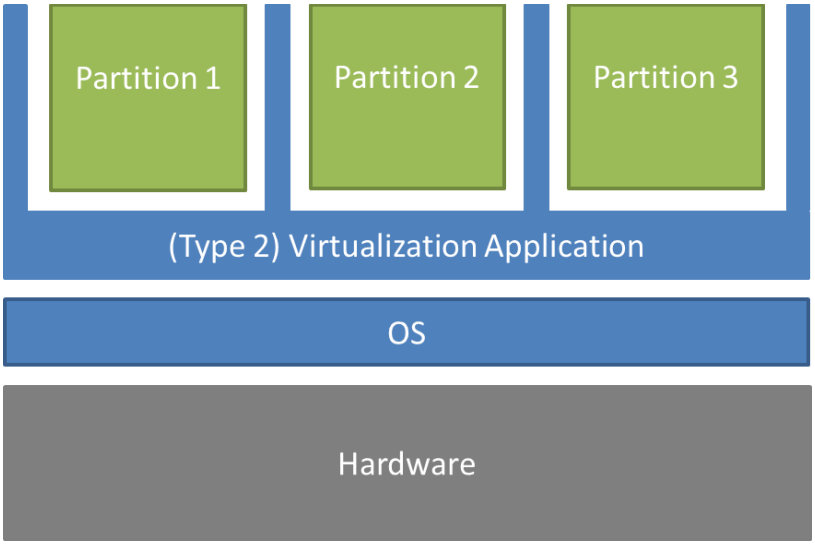

Figure 11: Type 2 hypervisor

In general, such type 2 hypervisors are unlikely to be separation kernels, as the design is much richer and more complex and harder to verify, especially when a general-purpose OS is being used. For low-assurance general-purpose operating systems or low-assurance virtualization solutions based on general-purpose operating systems there is already OSPP [32], the OSPP virtualization extensions [34] and GPOS PP [35].

4.7.2 SKPP and others

We are aware of related work like SKPP [36], the EURO-MILS PP [37], and High-assurance SK (HASK) PP [38]. However, to our knowledge none of these PPs have undergone CC User Forum (CCUF) community review. To our knowledge, while good reference material, the EURO-MILS and HASK PPs have never been applied. SKPP is probably the most known related work, as SKPP has indeed been applied to Green Hills Integrity twice in 2008 and 2010. However, SKPP was later retired for several reasons [39]:

- SKPP contained very many extended functional as well assurance requirements, and was hard to compare with more standard CC SFRs and Security Assurance Requirements (SARs). Our PP does not use extended components at all.
- SKPP, in particular, did not base its assurance on the common CC EAL scale, but rather invented its own definition of “high-robustness” which despite other disclaimers could be misunderstood as superset of EAL6 (e.g. [36], Section 6.2), leading to heated discussions about its EAL claims. At least until we have collected more stakeholder views’, our PP tries to keep away from demanding very high EALs.
- SKPP has been criticized for not adequately taking into account the difficulties of COTS hardware. Our PP recognizes that the assurance a CC evaluation can give is limited, and will usually not take all burden of proof away from the system integrator. Our approach rather encourages to make precise claims on the evaluation boundary (e.g. notion of privileged partitions, HAL, etc.).

Chapter 5 How to write an ST for a PP-compliant SK

This chapter explains how an ST writer for can write an ST for her PP-compliant separation kernel. To do so means to become more concrete on necessarily abstract concepts in the PP. This chapter supplements the instructions given in the PP itself.

5.1 Different abstraction levels for separation kernel assets

For writing an ST that complies to the PP, it helps to understand that there are different levels of asset abstraction:

- **Physical assets**, given by hardware, which exist on many platforms (physical memory, CPU time)
- The **application interface view**, which may comprise file descriptors, virtual memory addresses etc.
- The **system integrator view** for a particular separation kernel, which may be a mix of abstract resources (e.g. memory) and the application interface view resources (e.g. files)

Let us look at this in more detail: On all known architectures, separation kernels operate on certain “basic” physical assets such as memory and CPU time. The PP is quite conservative here, there are more physical assets, which certMILS had considered during drafting the PP, such as interrupts and devices (which also occur on many, if not the majority of systems). However, as there might be cases of separation kernels that do not directly interact with interrupts and devices, for instance think of an SK where all devices are memory-mapped, and for those systems, ST authors would have to argue for the absence of interrupts and devices. Having to argue why certain assets are absent from a given system is usually harder than to argue about addition of certain assets.

Secondly, similar to general-purpose operating systems, a separation kernel may provide logical abstractions for these basic resources. These abstractions are often called exported resources, and they can be anything like files, logical devices, avionics ARINC ports [12]. When a logical abstraction is described, in the SFRs the physical view probably will not occur. However the physical representation of the logical resource will have to be taken into account during security architecture analysis of the SK (e.g. if different logical or physical configuration options refer to the same physical abstraction, and allow to bypass each other, this has to be made clear in the system integrator documentations).

In its security functional requirements (SFR) section, a separation kernel ST may choose to argue on physical or logical resources. From our experience, policies in an ST will describe both the perspective of the system integrator as well as the perspective of run-time applications (which you also could call the application developer’s perspective). In the following we give some examples of how resources or exported resources would look like in an ST.

In the following, we will talk mostly about access control. Access control is expressed by requirements in the CC data protection family (FDP). Of these we are interested in complete access control (FDP_ACC) and in security attribute based access control (FDP_ACF). We are usually focussing on FDP_ACF.1.1, FDP_ACF.1.2 and FDP_ACC.2.1, because:

- FDP_ACC.2.2 does not contain any new text to be filled out by the ST author.
- FDP_ACF.1.3 and FDP_ACF.1.4 depend on the given ST, and it is difficult to give general guidance here, beyond that specifying “none” for additional rules could often suffice, when rules are already sufficiently established in FDP_ACF.1.2.

In the SFR texts, you will encounter the terms:

- Target of evaluation (TOE), which in our case is the separation kernel as software, possibly accompanied by guidance.
- TOE security functionality (TSF), which in is our case the functionality of the separation kernel that must be relied upon for the correct enforcement of the SFR.

5.2 Instantiation of SFRs for physical assets that are already explicitly mentioned in the PP

The PP already explicitly mentions the assets of memory and CPU time.

5.2.1 Memory

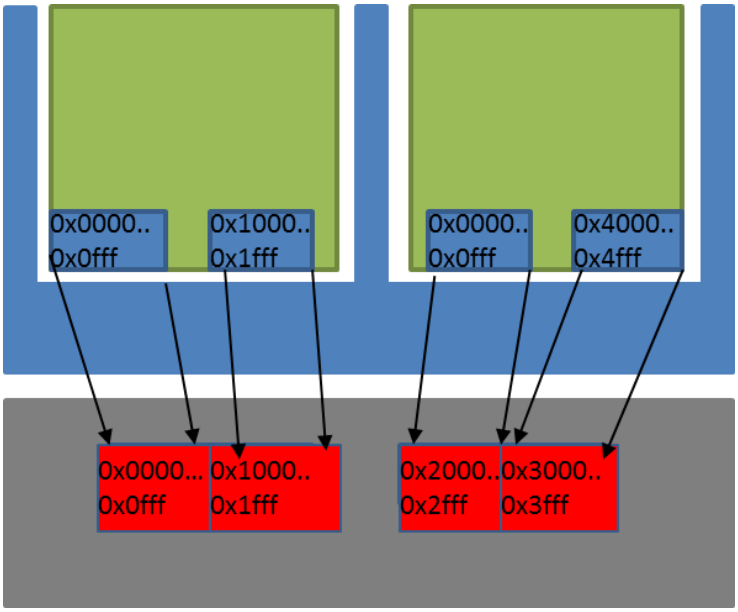

Figure 12: Virtual memory in a separation kernel

Figure 12 shows how virtual memory, attached to the partitions is mapped to physical memory (superimposed on the grey box). Observe that the page 0x0000-0x0fff occurs both in the left-hand side as well as in the right-hand partition but it is mapped to different physical addresses (0x0000-0x0fff and 0x2000-0x2fff, respectively), which is the whole point of address virtualization: different partitions can be assigned different address spaces.

- **Physical asset view**: the physical asset view only observes the partitioning of memory marked in red in the lower part of Figure 12. E.g. memory protection provided by MPU / MMU [40] ensures that applications do not read out other partitions' memory, by whichever means.
- **Application interface view**: The run-time application view will be to do "load" and "store" assembly commands to virtual addresses, marked in blue in the upper part of Figure 12.
- **System integrator view**: the system integrator will probably more think of the physical resources view (lower part of Figure 12), although in systems that support tasks (address spaces) she might assign specific address space identifiers to partitions (not shown).

In the PP, an access control policy is given from a system integrator view in FDP_ACC.2.1/MA, stating that the "The TSF shall enforce the [memory access control policy] on [subjects: partitions, objects: memory areas] and all operations among subjects and objects covered by the SFP". Hence the ST writer does not have to add anything new here.

The PP leaves it to the ST writer to specify in FDP_ACF.1.1/MA "*other security attributes*". If the ST writers opts to base the description of FDP_ACF.1.2/MA (see next paragraph) on task identifiers, then these task identifiers could be mentioned here, otherwise, "none" would be a possible instantiation.

The PP leaves it to the ST writer to specify in FPD_ACF.1.2/MA the "*set of rules that are used by the TSF to determine if access of a partition to a memory area is allowed*." In the ST, an ST writer would probably formulate FDP_ACF.1.2/MA either in terms of areas of physical memory, or in terms of different address spaces (system integrator view). Formulating an access-control based SFR *only* on virtual addresses (application view) would probably be meaningless, as the virtual addresses are not unique to partitions. A possible for formulation for the set of rules that are used by the TSF to determine if access of a partition is allowed could be: "access to memory in access mode XXX is allowed only if it is assigned to the partition by the configuration element YYY with access mode XXX." Access modes could be for example "read", "write", and/or "execute". Another possible formulation for the set of rules could be: "access to memory is allowed only if the memory is assigned to a task by configuration element XXX and the task is assigned to a partition by configuration element YYY." If there are possibilities to exchange memory at run-time (e.g. memory mapping IPC), this also should be mentioned in FDP_ACF.1.2./MA or (more generally) FDP_ACF.1/MA.

The PP contains an application note which also covers the application interface view, stating whenever there is an unmapped virtual address, new physical memory is taken from the physical memory assigned to a partition.

When the ST is used for evaluation, the evaluator should be aware, that the physical abstractions provided in this SFR have limitations: an ST might just speak of "memory regions" which in the implementation might consist of a whole memory hierarchy (e.g. non-volatile memory is used for swapping [41]). To give a consistent argument on the implementation side, at medium-to-high evaluation assurance levels (EALs) according to CC, is the task of the separation kernel's CC security architecture (in CC terminology, the ADV_ARC work units).

5.2.2 CPU time

This asset can be fulfilled by different SFRs. Hence it is realized by one or several PP modules. The guidance is given in the document on PP's CPU time modules.

5.3 Adding new SFRs for assets that are not explicitly mentioned in the PP

The PP explicitly describes the physical assets of memory and CP time, as all separation kernels have these common in common. The separation kernel you are writing the ST for probably has more physical assets, such as devices, interrupts, or logical assets, such as files, ports. For the reasons given in Section 5.1, we did not hardcode these into the base PP. But the base PP asks you as ST author how to specify these assets. In the following we give some examples how to add new assets to the PP.

Intentionally the definition of threats and security objectives (in CC terminology: security problem definition (SPD)) in the PP has been kept quite generic, so when you add assets, you probably can do so without changes to threats and security objectives. However, you will have to adapt yourself the list of assets and SFR instantiations. In the following we give examples how to do this. We start with physical assets such as interrupts and devices and proceed with logical assets such as files and ports.

Note that formulations used in this are just for illustration, and you might prefer other formulations. If you wish to use other formulations, you can do so without any justification, as the formulations in this document are just examples; they are *not* part of the PP.

5.3.1 Interrupts

In this section we show an example of how to add an asset for interrupts, identified by an interrupt ID. Of course, these SFR instantiations by necessarily are examples, e.g. you would insert meaningful values for “XXX”.

5.3.1.1 Asset table

In Section “Assets” of the ST add the following for asset “interrupts”:

Interrupts: The separation kernel has to ensure that interrupts that need to be handled by a partition are provided to the partition’s interrupt handling routine.

5.3.1.2 SFRs

In Section “Security Functional Requirements” of the ST add:

FDP_ACC.2/INT Complete Access Control – Interrupt Access

FDP_ACC.2.1/INT 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/INT 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.

FDP_ACF.1/INT Security Attribute Based Access Control – Interrupt Access

FDP_ACF.1.1/INT The TSF shall enforce the [interrupt access control policy] to objects based on the following: [subjects: partitions, objects: interrupts, security attributes used to enforce the policy: interrupt ID, partition ID].

FDP_ACF.1.2/INT The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: [a partition can register for, read, handle an interrupt if it is allowed by configuration element XXX]

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

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

5.3.2 Devices

In this section we show how to add an asset for hardware devices, identified by a device ID:

5.3.2.1 Asset table

In Section “Assets” of the ST add the following for asset “devices”:

Device: A device may be a real device or a virtualized device. A real or virtualized device may only be shared between partitions when this is deliberately configured. A device that is re-assigned during operation from one partition to another one must be cleared before it can be accessed by the new partition.

5.3.2.2 SFRs

In Section “Security Functional Requirements” of the ST add:

FDP_ACC.2/DEV Complete Access Control – Device Access

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

FDP_ACC.2.2/DEV 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.

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

FDP_ACF.1.1/DEV The TSF shall enforce the [device access control policy] to objects based on the following: [subjects: partitions, objects: devices, security attributes used to enforce the policy: device ID, partition ID].

FDP_ACF.1.2/DEV The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: [a partition can access a device if the device is configured to accessible by the partition by configuration element XXX, which contains device ID and partition ID].

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

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

5.3.3 Files

In this section we show an example of how to add an asset for files, identified by a file name. As this is the first logical asset, we this discuss, it might be helpful to review our different views (compare Section 5.1):

- **Physical asset view**: the physical asset would observes the partitioning of memory that is implied by the file abstraction, marked in the red in the lower part of Figure 13.
- **Application interface view**: The run-time application view will be that of files, that are opened by the applications. Once the files have been opened, the applications will even just see the files in the form of “file descriptors”, marked in blue in the upper part of Figure 13.
- **System integrator view**: the system integrator will probably more use the file view (upper part of Figure 13), where files are identified by names (“X” and “Y” in Figure 13).

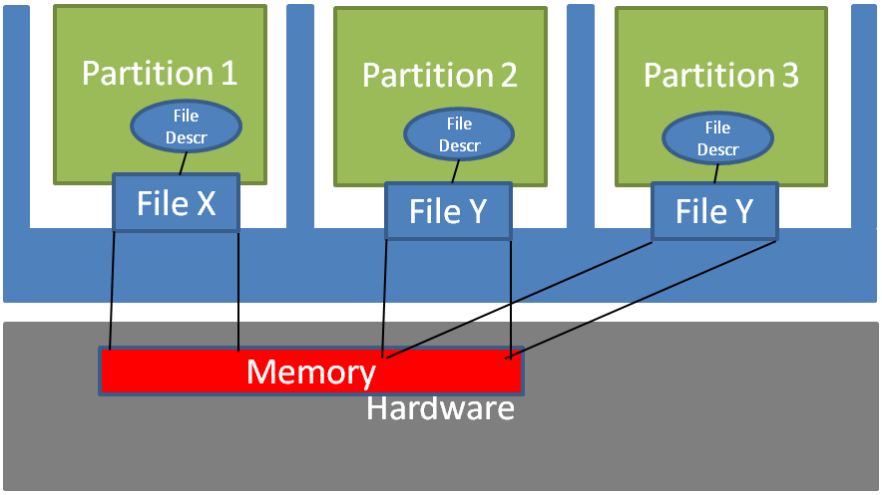

Figure 13: Abstraction levels for files

Files can be assigned exclusively (File X in Figure 13) or even be shared among partitions (File Y). In the SFR formulation example below, the physical asset view does not occur at all, but it has to be taken into account during discussion of security architecture. Moreover, if the configuration allows to have logical interfaces overriding physical asset assignments that also needs to be mentioned in the guidance to the system integrator.

The example formulation below combines the system integrator view (file names) and the application interface view (file descriptors).

5.3.3.1 Asset table

In Section “Assets” of the ST add the following for asset “file”:

Files: Access to files and access modes assigned to a partition by the integrator.

5.3.3.2 SFRs

In Section “Security Functional Requirements” of the ST add:

FDP_ACC.2/FILE Complete Access Control – File Access

FDP_ACC.2.1/FILE 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/FILE 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.

FDP_ACF.1/FILE Security Attribute Based Access Control – File Access

FDP_ACF.1.1/FILE The TSF shall enforce the [file access control policy] to objects based on the following: [subjects: partitions, objects: files, security attributes used to enforce the policies: file name, file handle, partition ID].

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

- An application in a partition only can open a file with name NNN in access mode XXX if it is assigned file NNN in configuration element CCC. The open operation creates a valid file descriptor.
- An application in a partition only can read/write/close a file if it has a valid file descriptor and the access operation conforms with the access mode XXX the application has been opened in.

].

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

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

A common run-time interaction for this would be that an application within a partition opens a file, obtains a file descriptor, and run certain commands such as reading, writing and finally closing. Obviously, the system has to deny access to files that are not configured to be available and also ensure that e.g. reading has a valid descriptor and does not pass beyond the length of the file.

The file abstraction provides an often more useful interface to the user than acting on raw memory. Hence, a separation kernel (SK) provider may choose to implement that interface to give the user a more meaningful access to the OS resources than interfacing raw memory.

5.3.4 Ports

In this section we show an example of how to add an asset for avionics avionics ARINC 653 [12] ports, identified by a port ID.

5.3.4.1 Asset table

In Section “Assets” of the ST add the following for asset “port”:

Port: Each port is either a source port or a destination port. The integrator uses channels to specify from which source ports the TSF transfers messages to which destination ports.

5.3.4.2 SFRs

In Section “Security Functional Requirements” of the ST add:

FDP_ACC.2/PORT Complete Access Control – Port Access

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

FDP_ACC.2.2/PORT 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.

FDP_ACF.1/PORT Security Attribute Based Access Control – Port Access

FDP_ACF.1.1/PORT The TSF shall enforce the [port access control policy] to objects based on the following: [subjects: partitions, objects: ports, security attributes used to enforce the policies: partition ID, port ID].

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

- a port can be opened in source/destination mode by a partition, if its port ID is assigned to the partition in configuration element XXX
- write access to a port is allowed if the port had been opened as a source port
- read access to a port is allowed if the port had been opened as a destination port

]

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

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

Chapter 6 Summary and Conclusion

There are many stake-holders in a PP already for a single-level certification. In certMILS we aim a two-level certification scenarios and in such a two-level certification scenario, that is the certification of a separation kernel and, ensuing, the certification of a MILS system, there are even more possible view-points. In this document we have documented what we anticipate to be practical additional information for some of these view-points.

By this text, we hope not only to have promoted the use of separation kernels and MILS systems, but also to have given an example of how to apply compositional certification by propagation of certified security properties.

Chapter 7 List of Abbreviations

Abbreviation	Translation
ADV	Development Assurance Evaluation Class in the CC
ALC	Life-cycle Support Assurance Evaluation Class in the CC
ARINC	Aeronautical Radio, Incorporated
ATE	Testing Assurance Evaluation Class in the CC
AVA	Vulnerability Assessment Evaluation Class in the CC
CAN	Controller Area Network
CAP	Composed Assurance Package in terms of CC
CC	Common Criteria for Information Technology Security
CCDB	Common Criteria Development Board
CCUF	Common Criteria User Forum
COTS	Common Off-the-Shelf
CPU	Central Processing Unit
DMA	Direct Memory Access
EAL	Evaluation Assurance Level in terms of CC
ESR	Essential Security Requirements in terms of CC
FDP	Data protection assurance family in terms of CC
GPOS	General Purpose Operating System
HAL	Hardware Abstraction Level
HASK	High-Assurance Separation Kernel
HTTP	Hypertext Transfer Protocol
ID	identifier
IT	Information Technology
MILS	Multiple Independent Levels of Security / Safety
MMU / MPU	Memory Management Unit / Memory Protection Unit

Abbreviation	Translation
MS	MILS System
NIC	Network Interface Card
OS	Operating System
OSPP	Operating System Protection Profile
PP / cPP	Protection Profile / collaborative PP in terms of CC
RTOS	Real-Time Operating System
SAR	Security Assurance Requirement in terms of CC
SFP	Security Function Policy
SFR	Security Functional Requirement in terms of CC
SK	Separation Kernel
SKPP	Separation Kernel Protection Profile
SPD	Security Problem Definition
ST	Security Target in terms of CC
TFTP	Trivial File Transfer Protocol
TOE	Target of Evaluation
TSF	TOE Security Functions

Chapter 8 References

[1] certMILS, “D2.1 Protection Profile,” 2018.

[2] Common Criteria Sponsoring Organizations, “Common Criteria for Information Technology Security Evaluation. Version 3.1, revision 5,” April 2017. [Online]. Available: http://www.commoncriteriaportal.org/cc/.

[3] Common Criteria Sponsoring Organizations, “Certified Products,” 2018. [Online]. Available: http://www.commoncriteriaportal.org/products/.

[4] IBM; atsec, “SuSE Linux Enterprise Server V 8 with Service Pack 3 Security Target for CAPP Compliance,” 2003. [Online]. Available: https://www.commoncriteriaportal.org/files/epfiles/0327b.pdf.

[5] F. Pattinson, “Deep Thought on PPs,” 2012. [Online]. Available: http://atsec-information-security.blogspot.com/2012/10/deep-thought-on-pps.html.

[6] CCDB DSC WG, “Dedicated Security Components (DSC) Essential Security Requirements Version 1.2,” May 2016. [Online]. Available: http://www.commoncriteriaportal.org/communities/CCDB_DSC_ESR_v1.2.pdf.

[7] Wikipedia, “System programming language,” 2018. [Online]. Available: https://en.wikipedia.org/wiki/System_programming_language.

[8] ARM Ltd, “Cortex-A15,” 2018. [Online]. Available: https://developer.arm.com/products/processors/cortex-a/cortex-a15.

[9] ARINC, Commercial Aircraft Information Security Concepts of Operation and Process Framework: ARINC standard 811, Aeronautical Radio, Inc., 2551 Riva Road, Annapolis, MD 21401, 2005.

[10] Isa, “ISA62443 Security for industrial automation and control systems Part 1-1: Terminology-- concepts and models,” 2016. [Online]. Available: http://isa99.isa.org/Public/Documents/ISA-62443-1-1-WD.pdf.

[11] K. Mueller, “Hardening High-Assurance Systems: MILS as Software Design for Avionics,” mar 2017. [Online]. Available: https://doi.org/10.5281/zenodo.571160.

[12] ARINC, Avionics application software standard interface: ARINC specification 653, Aeronautical Radio, Inc., 2551 Riva Road, Annapolis, MD 21401, 1997.

[13] I. Furgel, V. Saftig, T. Wagner, K. Müller, R. Schwarz and A. S.-F. Blomberg, “Non-Interfering Composed Evaluation,” 19 Jan 2016. [Online]. Available: http://dx.doi.org/10.5281/zenodo.47979.

[14] CENELEC, *EN 50159: Railway applications - communication, signalling and processing*

systems - safety-related communication in transmission systems, Brussels: CENELEC, 2010.

[15] SAE, J3061: Cybersecurity Guidebook for Cyber-Physical Vehicle Systems, 2107.

[16] certMILS, “D1.1 Regulative Baseline,” 2017.

[17] ISA Security Institure, “SDLA-312 Security Development Lifecycle Assessment Version 3.0,” 2014. [Online]. Available: http://www.isasecure.org/en-US/Certification/IEC-62443-SDLA-Certification.

[18] certMILS, “D1.3 Compositional Certification Methodology,” 2018.

[19] DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik in DIN und VDE, UK 351, Elektronische Bahn-Signalanlagen - Teil 104: Leitfaden für die IT-Sicherheit auf Grundlage IEC 62443, 2015.

[20] certMILS, “D2.3 Security Architecture Template,” 2018.

[21] B. W. Lampson, “A Note on the Confinement Problem,” *Comm. ACM,* vol. 16, no. 10, pp. 613-615, October 1973.

[22] D. Kuzhiyelil and S. Tverdyshev, “Timing Covert Channel Analysis on Partitioned Systems,” 2014. [Online]. Available: https://www.escar.info/images/Datastore/2015_escar_EU/16_Kuzhiyelil_escar_EU_2015.pdf (free registration required).

[23] J. Alves-Foss, P. Oman, R. Bradetich, X. He and J. Song, “Implications of Mult-Core Architectures on the Development of Multiple Independent Levels of Security (MILS) Compliant Systems,” 2012. [Online]. Available: http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA568860.

[24] X. Jean, D. Faura, M. Gatti, L. Pautet and T. Robert, “Ensuring Robust Partitioning in Multicore Platforms for IMA Systems,” in *Digital Avionics Systems Conference (DASC), 2012 IEEE/AIAA 31st*, 2012.

[25] R. A. Kemmerer, “Shared Resource Matrix Methodology: An Approach to Identifying Storage and Timing Channels,” *ACM Transactions on Computer Systems,* vol. 1, no. 3, pp. 256-277, 1983.

[26] R. A. Kemmerer, “A Practical Approach to Identifying Storage and Timing Channels: Twenty Years Later,” in *ACSAC*, 2002.

[27] S. Nordhoff and H. Blasum, “Ease Standard Compliance by Technical Means via MILS,” 2017. [Online]. Available: https://doi.org/10.5281/zenodo.571175.

[28] International Electrotechnical Commission, Technical Committee 65: Industrial-process measurement and control, Subcommittee 65A: System aspects, Functional safety of electrical/electronic/programmable electronic safety-related systems, 2.0 ed., 3, rue de Varembé, CH-1211 Geneva 20: IEC Central Office, 2010.

[29] ISO/TC 22, Road vehicles, Subcommittee SC 3, Electrical and electronic equipment, ISO 26262:2011 Road vehicles - Functional Safety, Case Postale 56, Geneva, Switzerland:

International Standards Organization, 2011.

[30] RTCA SC-205 / EUROCAE WG-71, DO-178C: Software Considerations in Airborne Systems and Equipment Certification, Radio Technical Commission for Aeronautics (RTCA), Inc., 1150 18th NW, Suite 910, Washington, D.C. 20036, 2011.

[31] CCRA, “Arrangement on the Recognition of Common Criteria Certificates in the field of Information Technology Security,” 2014. [Online]. Available: http://www.commoncriteriaportal.org/files/CCRA%20-%20July%202,%202014%20-%20Ratified%20September%208%202014.pdf.

[32] Bundesamt für Sicherheit in der Informationtechnik (BSI), “Operating System Protection Profile,” 2010. [Online]. Available: https://www.commoncriteriaportal.org/files/ppfiles/pp0067b_pdf.pdf.

[33] R. P. Goldberg, “Architectural Principles for Virtual Computer Systems,” 1973.

[34] Bundesamt für Sicherheit in der Informationstechnik (BSI), “OSPP Extended Package - Virtualization,” [Online].

[35] National Information Assurance Partnership (NIAP), “Protection Profile for General Purpose Operating Systems,” 2015. [Online]. Available: https://www.commoncriteriaportal.org/files/ppfiles/pp_os_v4.0.pdf.

[36] Information Assurance Directory, “U.S. Government Protection Profile for Separation Kernels in Environments Requiring High Robustness. Version 1.03,” June 2007. [Online]. Available: https://www.commoncriteriaportal.org/files/ppfiles/pp_skpp_hr_v1.03.pdf.

[37] I. Furgel and V. Saftig, “Common Criteria Protection Profile “Multiple Independent Levels of Security: Operating System” [V2.03],” 31 Mar 2016. [Online]. Available: https://zenodo.org/record/51582.

[38] Bundesamt für Sicherheit in der Informationstechnologie (BSI), Sirrix AG, “Protection Profile for High-Assurance Security Kernel: Version 1.14,” June 2008. [Online]. Available: http://web.archive.org/web/20110726034516/http://www.sirrix.com/media/downloads/54500.pdf.

[39] J. V. Wise, “SKPP Sunset Q & A,” 2011. [Online]. Available: https://web.archive.org/web/20130426160900/http://www.niap-ccevs.org/announcements/SKPP%20Sunset%20Q%26A.pdf.

[40] J. H. Saltzer and M. D. Schroeder, “The Protection of Information in Computer Systems,” 1975. [Online]. Available: http://web.mit.edu/Saltzer/www/publications/protection/, http://www.cs.virginia.edu/ evans/cs551/saltzer/.

[41] J. L. Hennessy and D. A. Patterson, Computer Architecture: A Quantitative Approach, 3rd ed., San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002.