

# ARC in Practice:

## Common Structured Approach for the Creating and Assessment of ADV\_ARC Aspect

Igor Furgel  
T-Systems

12<sup>th</sup> ICCC



12<sup>th</sup> ICCC  
27-29 September



International Common Criteria Conference

# What are we speaking about?

- Motivation and solution approach
- Security domain separation as the basis of consideration
- Structured approach:
  - Domain separation
  - Self-protection
  - Non-bybassability
  - Secure start-up and initialisation
- Advantages and conclusion



# Motivation

- The CC community is still actively discussing several practical aspects of the assurance family ADV\_ARC and, first of all, how it has to be handled by
  - developers,
  - evaluators and
  - certifiers.
- Again and again, developers wonder which evidences are concretely expected with regard to specific technologies like smart cards (ICs and complete products), secure microcontrollers, TPMs, crypto boxes, digital tachographs, etc.



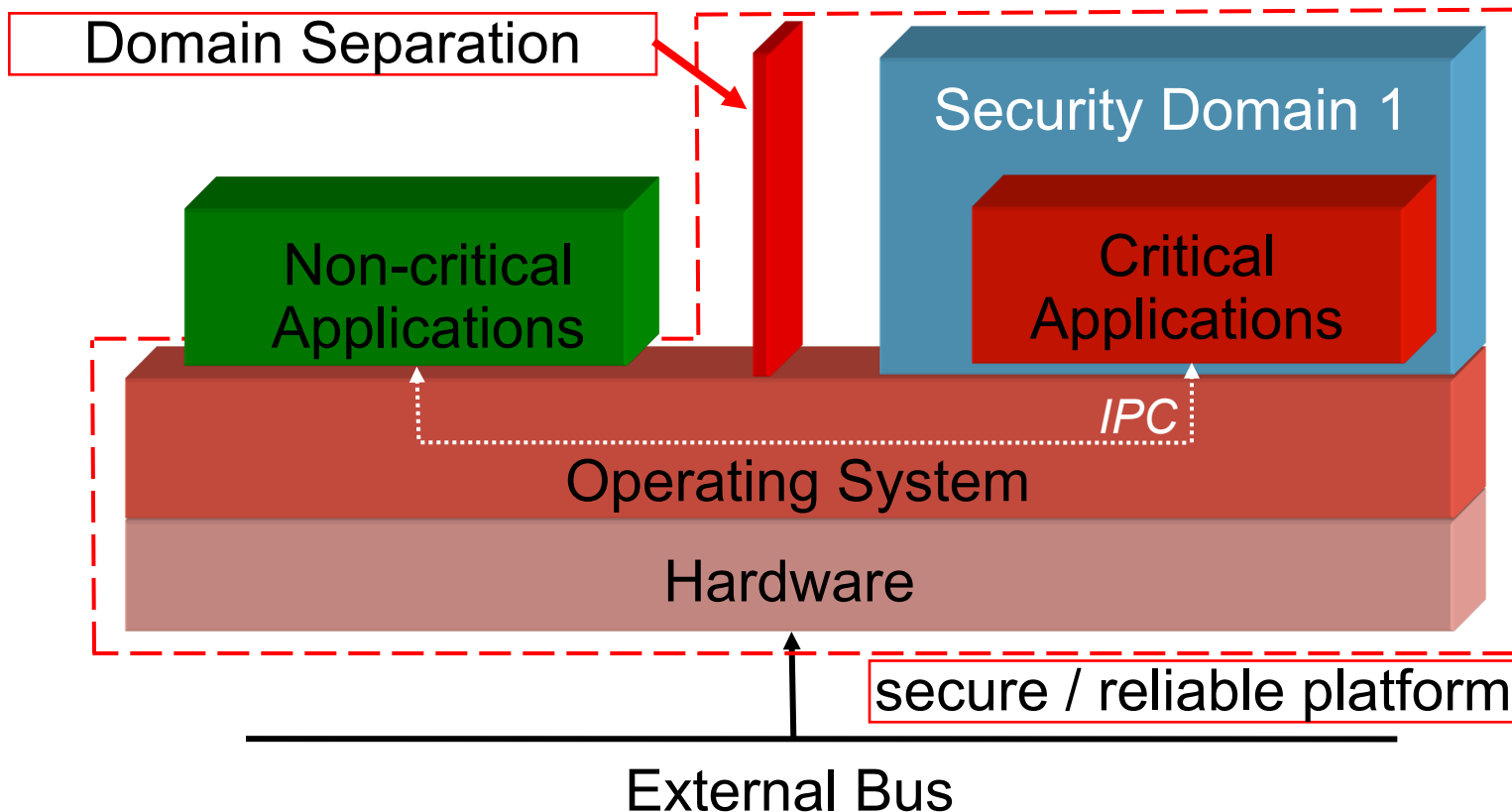
# Solution Approach

- In order significantly to reduce this uncertainty we developed a common structured approach for the creating and assessment of ADV\_ARC aspect.
- This structured approach was implemented in form of a list of general, **technology-independent** single measures (a kind of 'recipe') may be relevant for ADV\_ARC aspects.
- This approach may be used as a common support for **developers** and **evaluators** as well as for **certifiers** reminding them of single measures while writing, respectively verifying, ARC contributions.
- This common approach
  - joins major public experiences of several evaluation facilities
  - is compatible / covers major parts of the related trial JIL document
  - is part of **Guidelines for Evaluation Reports** issued by **BSI** (version 2.0 for CCv3.1, rev. 3; downloadable from the BSI homepage).



# Security Domain: Definition and Characteristics

- As a first step, the approach puts security domain separation in the centre of the consideration (see Appendix at the end of the presentation)



Good example for illustration: mass market products



# Structured Approach: Domain Separation

The 'recipe' assumes considering the following single measures supporting the chosen life cycle of the TOE (possible composite aspects are marked by COMP):

---

- Well-defined security domains / areas; a security domain can
  - span the entire TOE or
  - be represented by diverse operational modes enforcing different security policies (e.g. 'high' vs. 'low' security mode)
- Static vs. dynamic (run-time) separation of resources (storage media, CPU and other devices, if necessary).
  - Static separation of resources: dedicated data containers (e.g. files and directories; other devices, if necessary) and access control to them; this also contributes to the integrity of TSF-data (proper TSF-configuration).
  - (COMP) Dynamic separation of resources (e.g. RAM) in space (address management) and time (object reuse): a kind of information flow control.
- TSF-controlled Inter-Process-Communication (IPC) between the security domains (presumes  $\geq 2$  domains within the TSF): a kind of information flow control.



# Structured Approach: Self-Protection of TSF as a Means of Domain Separation

- (COMP) Against unintended information leakage: it covers e.g. information leakage via covert channels incl. countermeasures against side channel analysis.
- (COMP) Against physical manipulations by e.g. a periodic polling the states of the security relevant flags of hard- and software and an appropriate reaction to.
- If TSF cannot (sufficiently) secure itself, these aspects shall also be trapped by physical / organisational environment (assumptions in ST like a secure installation site and faithful administrator).



# Structured Approach: Non-bypassability of TSF: Technical Measures

is in large part a direct **result** of an effective **domain separation**.

In addition:

- The TSF-own periodic / before-usage **integrity check** of TSF-data and TSF-code (where applicable),
- (COMP) The TSF-own periodic / before-usage **verification** of security relevant **TSF-attributes** like e.g. HW / SW life-cycle flags or signatures over instructions / commands received,
- The TSF-own **verification** of the **quality metrics of TSF-data** like e.g. periodic / before-usage checks of
  - password properties (minimum length, character set, etc.),
  - (COMP) User-IDs (no administrator / supervisor),
  - (COMP) the properties of random numbers being used for TSF (key generation, initialisation of scrambling engine, other randomisations)



# Structured Approach: Non-bypassability of TSF: Procedural Measures

## In addition:

- Ensuring the **quality of source code** (programming techniques / developer's own guidelines for HW / SW development) like e.g.
  - definition of / adherence to a basic HW and SW architecture,
  - consistent definition / usage of program objects,
  - using automated code verifiers facilitating the avoidance of typical programming errors, which might result in malfunctions like erroneous parameter transfer, buffer overflows, etc.,
  - developer's internal product acceptance procedure.
  
- Such **procedural aspects** may preferably be addressed in the frame of the CC assurance **class ALC**. In such a case, the ADV\_ARC description / evaluation report should refer to ALC.



# Structured Approach: Secure Start-up / Initialisation

treats the following measures (separately for hard- and software):

- (COMP) A well-defined **sequence** for performing **self-tests** and the transfer of control to embedded software / operating system (HW and SW).
  - Thereby, possible **dependencies** between single HW and SW services should be regarded to.
  - Please note that not all services must unconditionally be activated at the very beginning, but perhaps merely **on demand**.
  
- **Power-on-reset hardware self-tests**
  - Checking **security relevant flags** like e.g. the states of sensors, current life-cycle flags of HW,
  - **Initialisation of single HW components**, e.g. RAM and other storage media, memory scrambling engine, memory management unit, etc.,
  - Integrity check of memory areas,
  - Integrity check of the TSF-data of HW (if applicable),
  - Testing single HW components, e.g. random number generator (such tests may also be triggered by embedded SW, see below)



# Structured Approach: Secure Start-up / Initialisation

## ➤ **Software self-tests** during initialisation

- (COMP) SW-sided initialisation of single HW components, e.g. RAM and other storage media, memory scrambling engine, memory management unit, etc.,
- (COMP) SW-sided tests of single HW components, if necessary; e.g. cryptographic co-processors like e.g. the tests of AES co-processor using test vectors, random number generator, etc.,
- Integrity check of the TSF-code and TSF-data of SW,
- Checking the life-cycle flags of SW (if present).
- (COMP) A well-defined sequence for the (physical and logical) activation of externally usable interfaces.
- (COMP) A well-defined sequence for starting different SW- (and, if necessary, HW-) services.



# Advantages of the Common Structured Approach and Conclusion

## ➤ Advantages

- Universal, technology-independent approach
  - Certifiers, evaluators and developers get a **common basis** refining the related requirements of CC; in such a way, ‘frictional losses’ between the ‘players’ can be reduced or even avoided increasing the efficiency of the entire process
  - Developers get a ‘scaffold’ guiding them through creating comprehensive ADV\_ARC contributions
  - Evaluators get a ‘leaflet’ reminding them of ADV\_ARC-relevant single measures, which may be implemented in a TOE
- A high **efficiency** of the approach has **successfully** been ‘tested’ for several smart card OS composite evaluations and digital tachographs:
- the developers knew, what they should describe,
  - the evaluators knew, what they should examine,
  - the certifiers knew, which evaluation methodology was applied.
- A related Change Proposal has already been submitted.

**Let us use the approach!**



**Thank you for your attention!**

**Dr. Igor Furgel**

**T-Systems International GmbH  
Security Analysis & Testing**

**Vorgebirgsstr. 49  
53119 Bonn**

**+49 228 98415120**

**igor.furgel@t-systems.com**



# Appendix I (cf. 9<sup>th</sup> ICCC)

## Security Domain: Definition and Characteristics

- Def.: *A security domain is a confined active physical and/or logical unit, where a single and homogeneous security policy is valid and applied. This security policy controls the behaviour of security services being provided in the context of this security domain.*
  
- The main generic characteristics of a security domain are the following:
  - *a security domain as a whole represents an encapsulated ‘unit’ (and can be considered as an object);*
  - *the externally visible and internal actions and reactions of this ‘unit’ represent its well-defined properties;*
  - *communication between such ‘units’ occurs by syntactically and semantically well-defined messages (and implements the relationships between the objects).*



# Appendix I (cf. 9<sup>th</sup> ICCC): Domain Separation, Self-Protection, Non-Bypassability and Initialisation

- This definition implies and upholds the following **simplification** for the known ARC aspects:
  - **Self-Protection** represents one of the *means* for Domain Separation, and
  - **Non-Bypassing** is one of the *effects* of Domain Separation.
- Even if there is no TOE-internal domain separation, there always is at least one security domain coinciding with the TOE itself and separating the TOE from external world.
  - The stripline in this case is the physical/logical scope of the TOE. The TSF shall maintain this 'shell'.
- Thus, there are two **central aspects** of ADV\_ARC:
  - Security domain separation and
  - Secure start-up / initialisation.

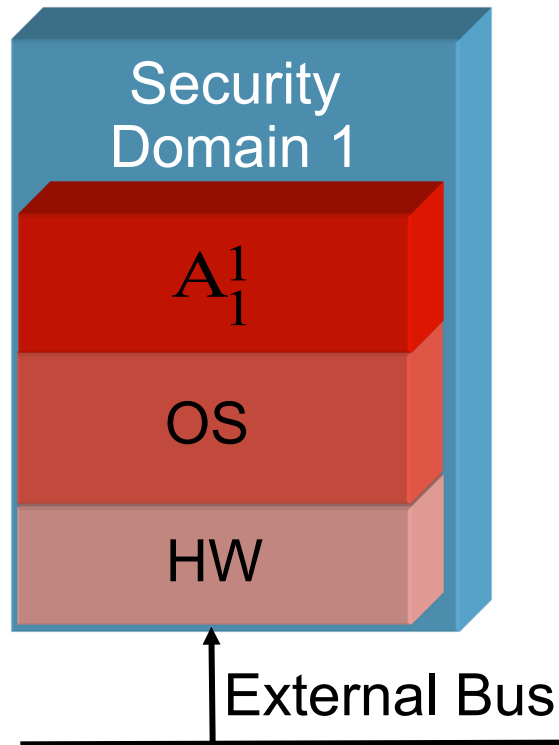


# Appendix I (cf. 9<sup>th</sup> ICCC): Means of Domain Separation

- Domain Separation can be achieved in physical and logical ways
  - In the case of *physical* separation, all communication passes through the external bus and the domain separation is trivially defined (but not trivially enforced).
  - In the case of *logical* separation, this requires the operating system to provide special services enforcing a domain separation.



# Appendix I (cf. 9<sup>th</sup> ICCC): Means of Domain Separation: Physical Separation



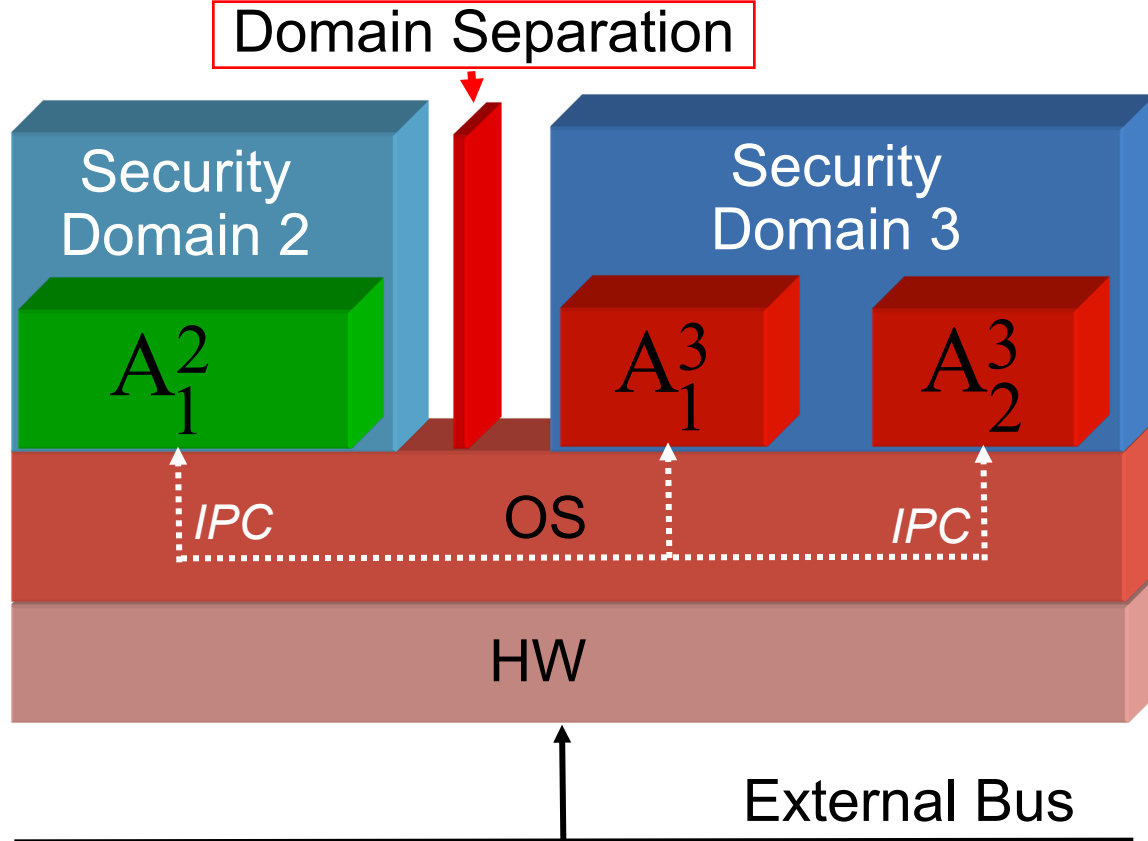
- Security Domain 1
- Application  $A_1^1$  (single module)
- Operating system (OS) maintains resources
- Hardware (HW) provides physical protection and may support OS (e.g. NMI)

## ➤ Segregation of modules:

- Keeping security through well-defined interfaces (*external* separation)



# Appendix I (cf. 9<sup>th</sup> ICCC): Means of Domain Separation: Logical Separation



- Security Domain 2
  - Application A<sub>1</sub><sup>2</sup> does not trust A<sub>1</sub><sup>3</sup> + A<sub>2</sub><sup>3</sup>
- Security Domain 3
  - Applications A<sub>1</sub><sup>3</sup> + A<sub>2</sub><sup>3</sup> do not trust A<sub>1</sub><sup>2</sup>
- Operating system manages all resources incl. IPC
- Hardware provides physical protection and must support OS (MMU)

## ➤ Segregation of modules:

- Maintaining security through well-defined interfaces (*external* separation)
- Divide-et-Impera state-space of complex applications (*internal* domain separation)

