Modeling Security Functional Requirements

Helmut Kurth

atsec information security

Outline

- The CC paradigm – an attempt of an explanation
- Putting it into context – a basis for modeling SFRs
- Mapping the model to part 2
- Deficiencies of CC part 2 components – and how to overcome them
- Suggestions for improvement

There is still a long way to perfection!

CC Paradigms

- User:
  - active entity outside of the TOE that requests services from the TOE (human user or external IT system).
- Subject:
  - Active entity within the TOE but outside of the TSF (such that services requested by a subject are mediated by the TSF.
- User-subject binding:
  - Service of the TSF that “binds” a user to a subject such that the subject may request services on behalf of the user.

Open issues:

- “Trusted subjects”; subjects not operating on behalf of a user.

CC Paradigms

- Object:
  - Passive entity controlled by the TSF.
- Resource:
  - Entity managed and controlled by the TSF.
- Information:
  - Anything a user may extract from the TOE by using services.

Open issues

  - Difference between “objects” and “resources”; entities that are sometimes passive (are operated upon), sometimes active; relation between information and the objects/resources they are stored in or processed by.

CC Paradigms

- Security attributes
  - May exists for users, subjects, information, objects, sessions, and resources.
  - Are managed by the TSF and used as part of the rules defining the security policy enforced by the TSF
- User data
  - Data stored in resources or objects controlled by the TSF but were the TSF do not interpret the data
- TSF data
  - Data stored in resources or objects controlled by the TSF which is used by the TSF as part of its operation
    - Example: security attributes, TSF internal state

Problems with the Paradigm

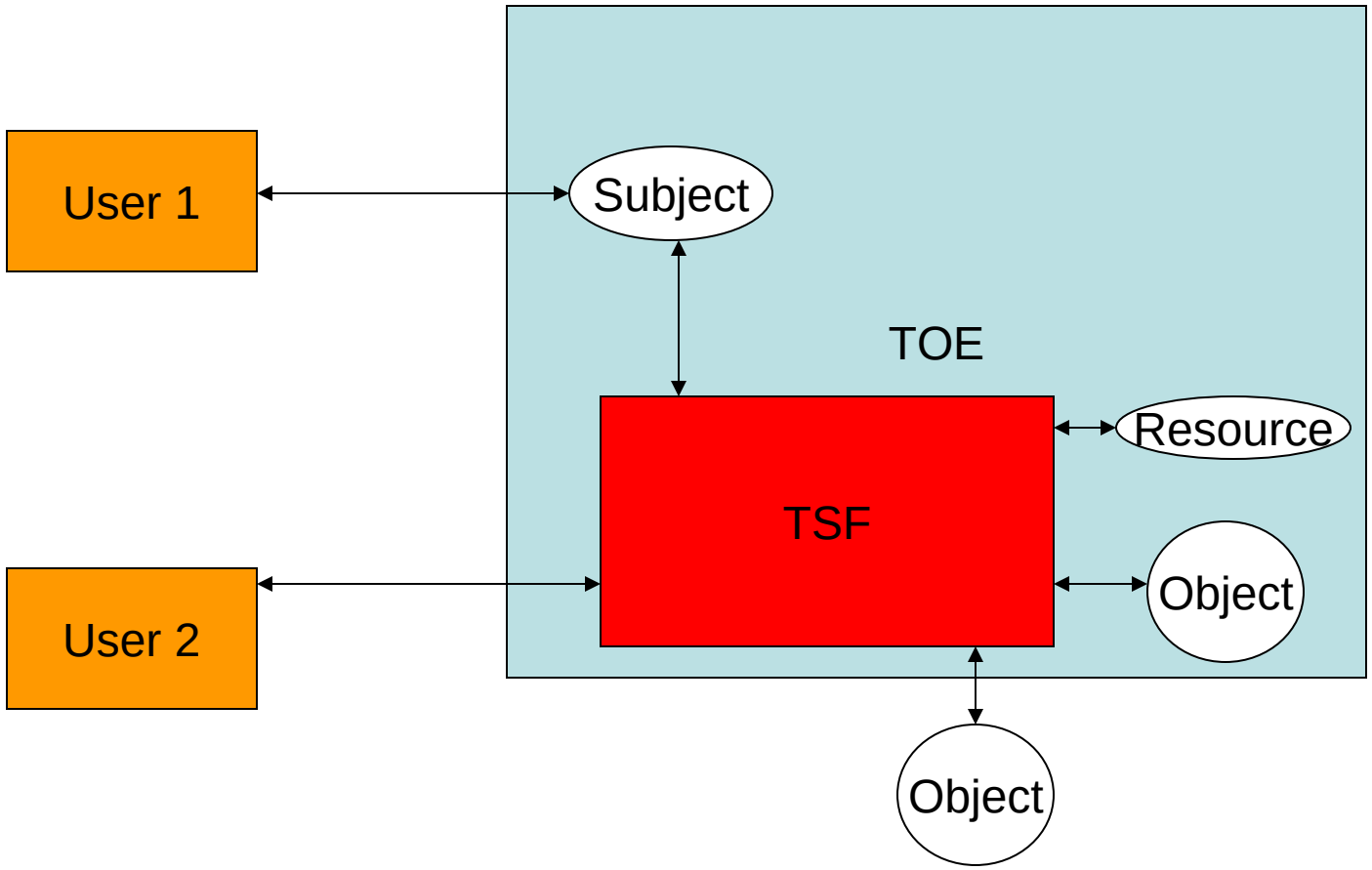
- No guidance is provided how to define/identify subjects, objects, resources, information, security attributes.
- No mapping from the paradigms to the security functional components is provided
- No consistency check between the paradigm and the functional components has been performed

This is one reason why part 2 of the CC is so hard to apply!

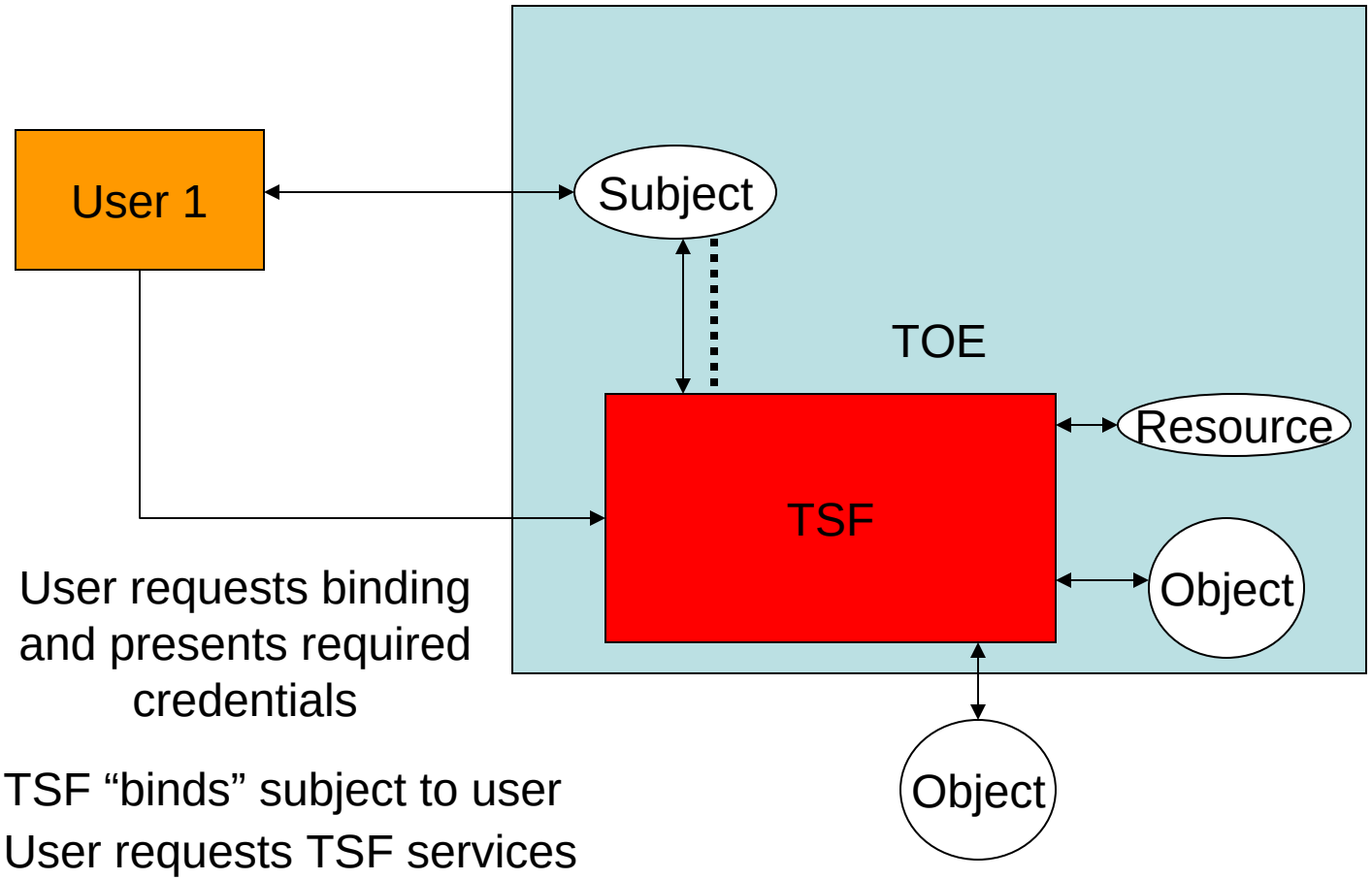
New Approach

- Let's stay with the terms used in the paradigm
- Let's develop a model of what we want to have as security functions
- Only when this is done attempt to map the model to part 2 of the CC!
  - Starting with part 2 of the CC when developing your model will bring you into trouble!
- What we suggest is a step-by-step approach to develop security functional requirements based on the CC paradigm

Policy elements in context



User-subject binding

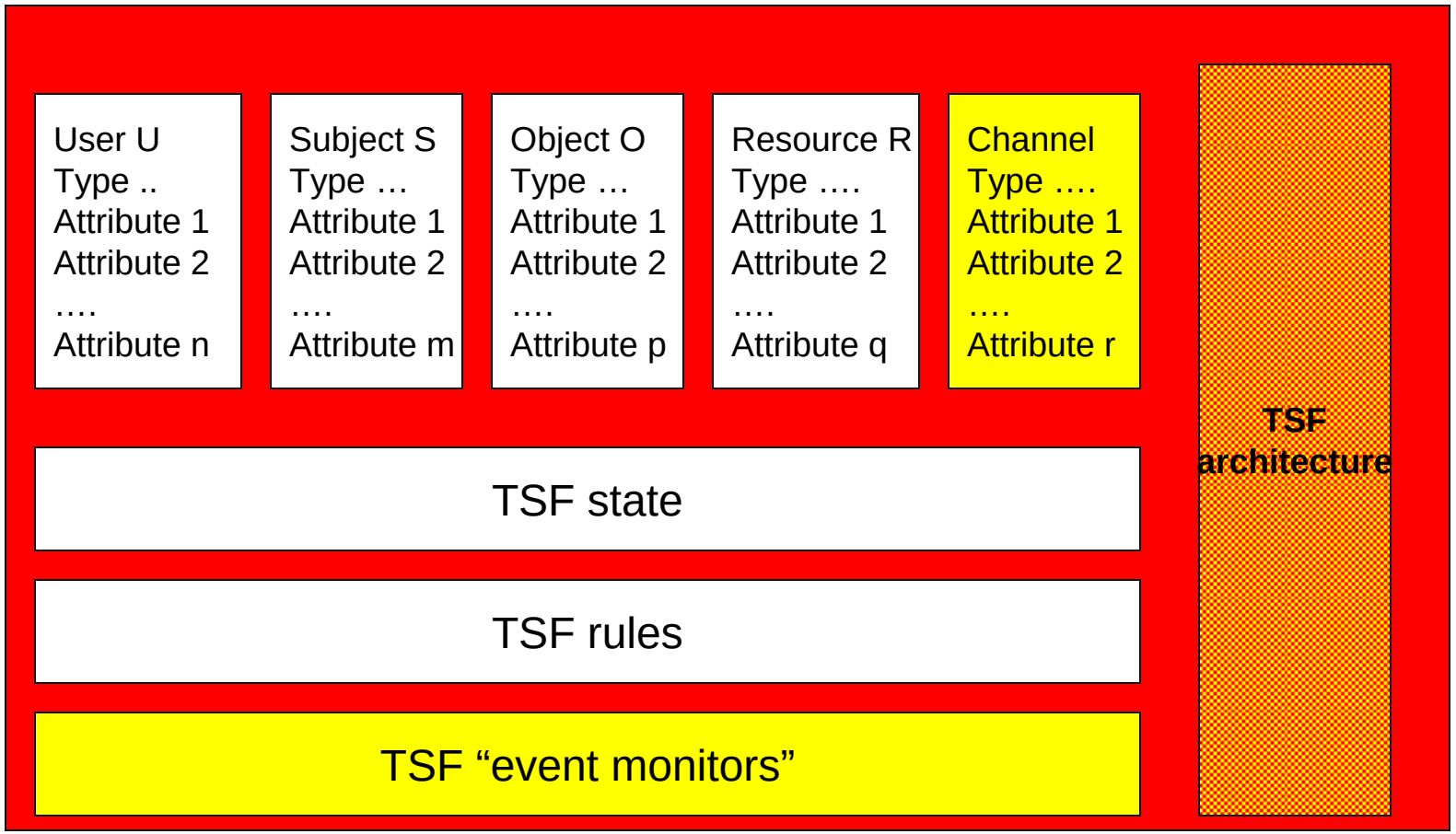


User requests binding and presents required credentials

TSF “binds” subject to user

User requests TSF services via subject

TSF data and policy elements



Additional policy elements - Communication Channels -

- Communication channels
  - Designates logical communication channels
  - May be characterized by security attributes like
    - Requires integrity protection
    - Requires confidentiality protection
    - Requires replay protection
    - Requires data authentication
    - ….
  - May have rules that determine initialization, management, use and termination of the channel

Additional policy elements - Event Monitoring -

- Defines events the TSF needs to react upon
  - Attempted violation of the policy
  - Detected failure of abstract machine or device
  - Reaching a specific state
  - ….
- Defines the rules how to react to events
  - Generate audit entry
  - Send message to external user
  - Modify state and/or security attributes of policy elements
  - Go to a specific state
  - ….

Additional policy elements
- TSF architecture -

- Additional rules to achieve security objectives
  - Separation between TSF and non-TSF portion of the TOE
  - Separation of different subject
  - Non bypassability
  - TSF internal information flow control
  - Consistency of TSF data
  - Availability of services

One may well argue that they are outside of the policy, but they are still required to satisfy valid security objectives

Removed policy element
- Session -

- Term used for traditional types of “terminal session”
- Can (with some interpretation) also be used for “session level protocols”
- Paradigm can be addressed by the new (broader) “channel” paradigm
  - Session establishment
  - Selection of session attributes
  - Limitation on concurrent sessions
  - Session locking
  - Session termination
  - “Access banner”, “Access history”

Removed policy element - Information -

- Does not really fit with the other elements
- Paradigm section states it is required for modeling information flow control
  - This is usually modeled via object and resource security attributes and access / use of the objects and resources with rules on the automatic initialization and management of those attributes
  - Requires support by architectural aspects
  - Is therefore removed as an element of the paradigms

Developing a security policy

- Step 1 – Element definition
  - Start with an initial set of users, subjects, objects, resources, channels
  - Start with an initial set of security attributes for each
    - Quite often one will identify that different “types” or “classes” of users, subjects, objects, resources and channels have different security attributes. Identify the types required
  - Define rules for creation, management and deletion of each element (if applicable)
  - Usually additional security attributes are identified by this process
  - Define rules for the initial values and the management of security attributes

Developing a security policy

- Step 2 – User interaction
  - Define the rules for users to interact with the TSF
    - Credentials to present
    - Rules when credentials are required
    - User-subject binding rules (if required)
    - Channels to be used
    - Rules for channel establishment
      - Setting the channel attributes
      - Other security relevant actions performed during channel establishment (like key establishment, access banner display)

Developing a security policy

- Step 3 – Object and resource usage rules
  - Define the rules for use of objects, resources by subjects and users
    - Usually different per object type and per resource type
    - Rules are usually based on security attributes of users, subjects, objects, resources, and channels used
    - Rules may also use TSF state information (like time, specific state like maintenance state, etc.)
      - Record the TSF state variables used
    - Definition of rules quite often identifies additional security attributes of the elements involved
      - Go back and define how those attributes are initialized and managed

Developing a security policy

- Step 4 – Import and export of objects
  - Export means: it is transferred out of the control of the TSF without sending it to a user connected to the TSF via a defined channel
  - Import means: it is accepted from some unknown source
  - Define requirements for import and export of objects
    - When import and export is allowed
    - What is required to be with the object when imported or exported
      - For example a defined set of security attributes
    - How the object is transformed and checked when imported
      - For example decrypted, integrity check, authenticity check, etc.
    - How the object is transformed when exported
      - For example encrypted, digitally signed, etc.

Developing a security policy

- Step 5 – Event definition, monitoring and management
  - Identify the events that need to be monitored
  - For each event, define the actions to be performed when an event happens, like
    - Write an audit record
    - Send a message to a user
    - Change the TSF state
    - Change security attributes of policy elements
    - ….

Developing a security policy

- Step 6 – TSF internals
  - Identify objectives that require to addressed by TSF internals (TSF architecture)
    - Separation
    - Reference mediation
    - Availability requirements
    - Information flow control requirements
    - Privacy
    - TSF internal integrity and consistency checks
    - Automated rules for modifying security attributes and TSF state variables
  - Some of those need to be supported by the rules defining the use of resources
    - For example information flow control requirements and privacy requirements may need support from rules defining usage of objects and resources

Mapping to CC part 2

- Basic question:
  - Once all this defined, can it mapped to the components of part 2?
  - Should be possible for the elements taken from the part 2 paradigms
- Answer:
  - It is only partly possible
    - Part 2 was not developed by putting the elements of the paradigm into context and consistently derive components from such a model

Attempted mapping

- Step 1 – Element definition
  - Element definition
    - Assumed to exist by part 2, no formal requirement to list element types and their security attributes
  - Element creation and initialization
    - Partly covered by FMT_MSA
  - Management of security attributes
    - Partly covered by several components in the FMT family
    - Not consistently addressed (too limited in the rules one can define)

Attempted mapping

- Step 2 – User interaction
  - Partly addressed by FIA
    - View of authentication is too narrow
  - Partly addressed by FTA
    - Too much related to the classical terminal session
  - Partly addressed by FTP
    - Not sufficient to model all security attributes of channels and their management

Attempted mapping

- Step 3 – Object and resource usage rules
  - Partly addressed by FDP and FRU
    - FRU also contains requirements on TSF internals
  - Management aspects partly covered by FMT
  - Many components are too restrictive to be applicable to many security policies
    - For example access control is restricted to access of subjects to objects, ignoring that there may be direct access of users
    - “usage” of resources is similar to “access” to object and requires similar flexibility in the definition of the rules

Attempted mapping

- Step 4 – Import and export of objects
  - Can be partly mapped to
    - FDP_ETC and FDP_ITC
    - FDP_UCT and FDP_UIT
    - FCO
  - Also here more flexibility in the definition of the rules is required

Attempted mapping

- Step 5 – Event definition, monitoring and management
  - Partly covered by FAU
    - Some requirements in FAU are related to the TSF internals
  - Parts of FDP_SDI
  - FDP_IFF.6
  - FIA_AFL
  - Several components in the FPT family
  - FRU_FLT
  - Also here flexibility is missing and the aspect is not addressed consistently

Attempted mapping

- Step 6 – TSF internals
  - Mainly addressed by FPT
  - Parts of FAU_STG
  - FDP_ITT
  - FDP_RIP
  - FDP_SDI
  - Parts of FPR
  - Parts of FRU
  - Many TSF internals need to be supported by usage rules and management functions!

Conclusion

- We have defined a framework for the definition of security functional requirements based on the paradigms defined in CC part 2
- We have identified that the structure of the components in part 2 do not follow a clear model
- We have identified that many components from part 2 do not present sufficient flexibility to model everything one can define with out framework
- Still most components of part 2 fit in our framework – some re-arrangement would enhance the understanding of part 2

Suggested future work

- Test the framework with different types of IT products and enhance it where necessary
- Arrange the components of part 2 around the framework
- Change components where more flexibility is required
- Remove redundant components
- Add missing components