An Attack Surface Driven Approach to Evaluation

Helmut Kurth
atsec information security corp.

Content

- What is the “attack surface”?
- Attack surface and TSFI – what is the difference
- Attack types examples
- Evaluation approaches focusing on the attack surface
- Suggestion for CC improvements

Attack Surface

- Any interface that a potential attacker has access to
  - Therefore the attack surface is a subset of the TSFI
- Different “types” of attacker may have access to different interfaces
  - In an operating system: system call interfaces, network interfaces, and GUI or command interfaces
- Some TSFI may not be part of the attack surface
  - Interfaces only accessible to “trusted users”
    - Trusted external IT systems
    - Trusted “administrators”
- Attack potential may be different for different interfaces
  - Depending on the “threat agents” that have access to the interface

Attack Surface – Definition

- Attack Surface (Wikipedia)
  - The attack surface of a software environment is scope of functionality that is available to any application user, particularly unauthenticated users.

- Definition misses the words “interfaces” and “untrusted”. A revised definition:
  - The attack surface of a software environment is the sum of interfaces that is available to any untrusted user, including unauthenticated users.
- Trusted users are assumed to not attack you
  - If they do, you are screwed up!

How an Attacker Works

- What are the interfaces I can access?
- How can I access them?
- What are those interfaces supposed to do?
- What type of flaws may exist?
- How can I find out if a flaw exists
  - Look for published flaws ("Google is your friend")
  - Develop a flaw hypothesis and validate by:
    - Simple testing
    - Reverse engineering / code analysis
    - Fuzz testing/stress testing

- **Most of an attacker's analysis is "close to the attack surface"**

Attack Surface in the Literature

- Term used often in the last 5 years
  - Although the method has been used for more than 30 years
- Everybody is talking about “attack surface reduction”
  - Almost no company is really reducing the size of the attack surface of existing products
- Determining the attack surface of a complex product may be hard
  - Almost no operating system vendor can provide a list of all the interfaces to the TSF of the OS!
- Possibilities to restrict access to interfaces are often not used
  - “Security by obscurity” still often used
    - “This interface can not be used, since it is not a documented interface”.

Consequences for Evaluations

- Evaluator should think like an attacker!
- He/she should do what an attacker does!
- Use the additional knowledge gained though the evaluation!
- Evaluation should focus on the attack surface (at least at lower assurance levels up to EAL4)!
  - Test “strange” parameter and parameter combinations
  - Look for “unusual” way of invoking interfaces
  - Do stress testing
  - Test for simple race conditions

Some Attack Surface related Attacks

- Buffer overflows
- Insufficient parameter validation
- Injection attacks
- Improper serialization
- Return values disclosing critical information

Many of those can be detected by an analysis "close to the attack surface"

CC Support of an Attack Surface Approach

CC has some attack surface related aspects:

- CC requires a ST to define the threats and "threat agents" (= potential attacker)
  - Different interfaces may have different "threat agents" with different attack potential!
- CC binds an "attack potential" to the level of AVA_VAN
  - Requires some kind of rating of the feasibility of an attack
- CC requires the "complete description of the TSFI" (EAL4 and above)
  - Helps to identify the complete attack surface as part of the TSFI

CC Problems

- All TSFI are treated equal
  - Regardless if it is part of the attack surface or not
  - Regardless of the “threat agent” that has access to the TSFI
- Focusing on “SFR-enforcing” functionality
  - As an attacker, I use any function that allows me to break the system!
  - Common attack surface related problems are not confined to security functionality
    - Buffer overflows
    - Race conditions
    - Insufficient parameter validation
    - Privilege escalation via call-back functions

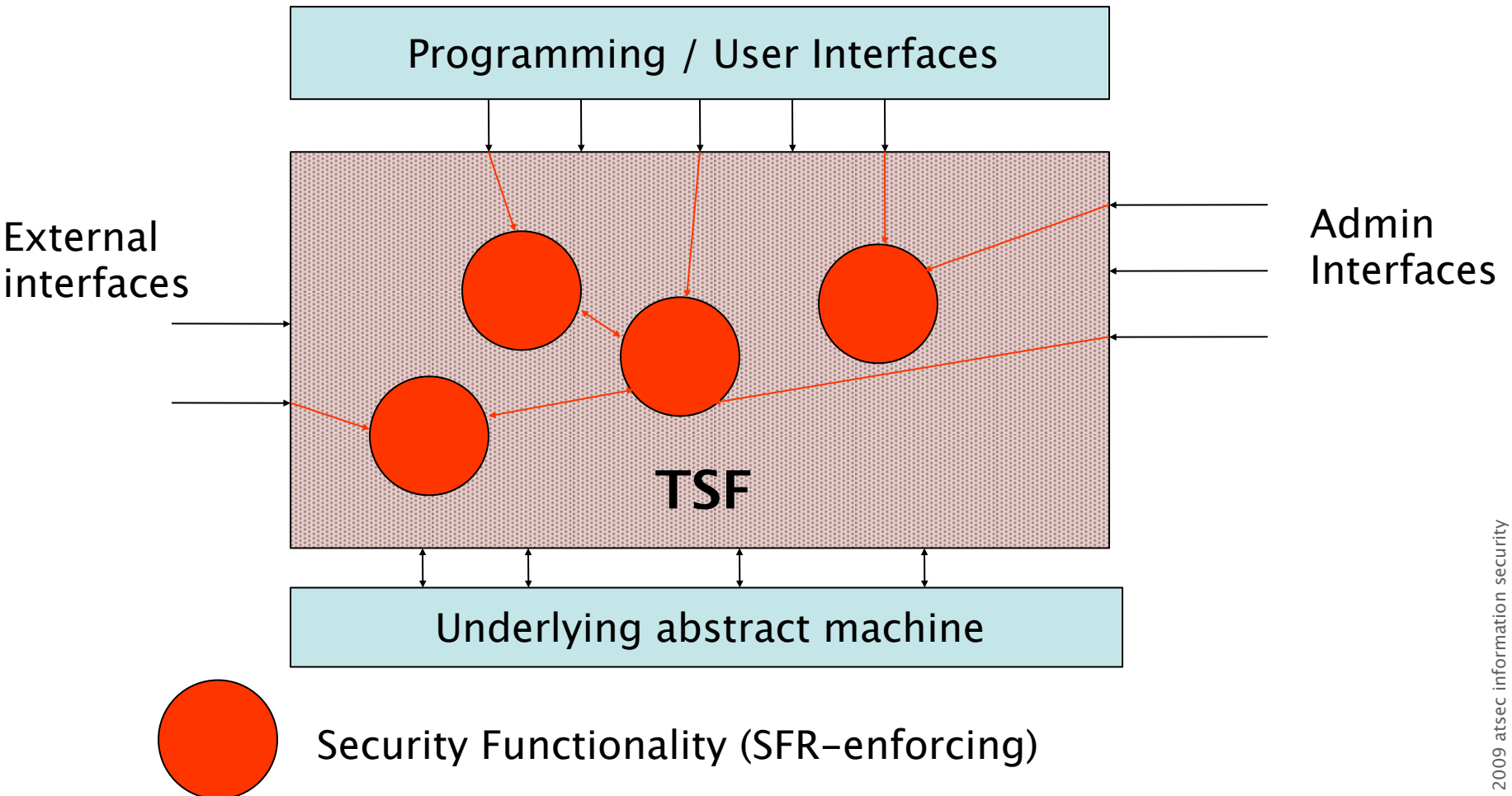
Problem: CC View of TSFI

- CC has the simple view of
  - External user or user program “invokes” a TSFI
  - TSF performs an action and returns to the user or user program
- There are other ways of interacting with the TSF!
  - TSF call-back functions
    - User registers “event” handler
    - TSF call the user registered event handler when the event occurs
  - TSF initiated interaction with a user
    - TSF initiated sending of network packages
    - Requesting a user action
  - “Output only” functionality (e. g. generating a storage dump)

Attack Surface Oriented Approach

- FSP requirements need to be modified to relate the TSFI to "thread agents"
  - More strict requirements when accessible to potential attackers
- TDS requirements need to be stricter for parts of the TSF that are "close to the attack surface"
  - Should be at the level of "SFR-enforcing"
  - Needs to allow analysis of
    - Potential buffer overflows
    - Insufficient parameter validation
    - Potential race conditions
    - Other direct attacks

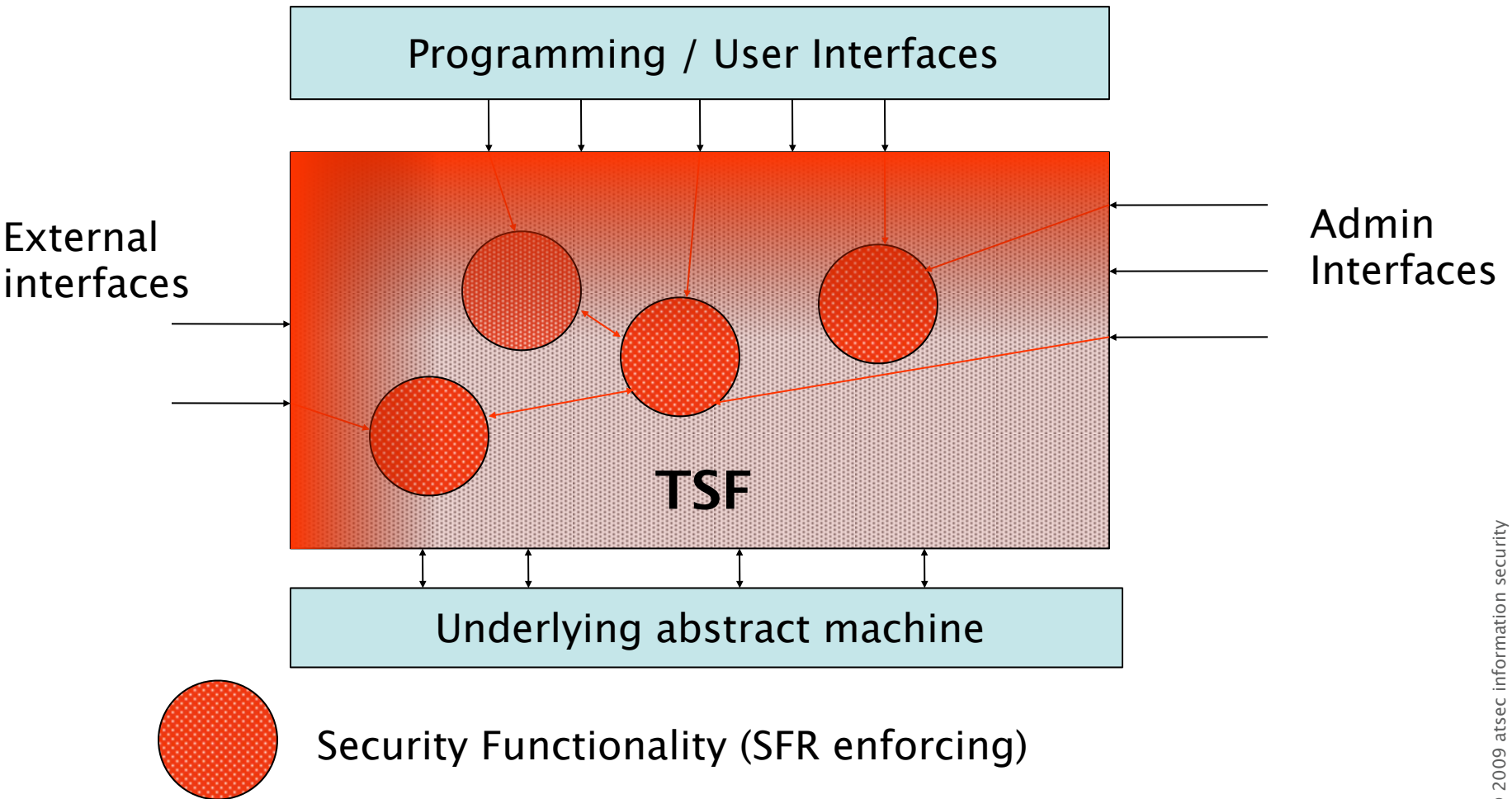
Current CC Approach



Current CC Approach

- Concentrates on functions implementing SFRs
- Does not require an extensive analysis of TSFI and design not related to security functionality
  - **ADV_FSP.4.2E** The evaluator shall determine that the functional specification is an accurate and complete instantiation of the SFRs.
  - **ADV_TDS.3.2E** The evaluator shall determine that the design is an accurate and complete instantiation of all security functional requirements.
- Does not distinguish between TSFI available to trusted entities and TSFI accessible by potential attackers
  - Admin interfaces require more analysis than non-security related interfaces accessible to a potential attacker (threat agent)
- Does not sufficiently take well-known attack methods into account!

Attack Surface based Approach



Attack Surface based Approach

- Concentrates analysis more on the attack surface
  - Like an attacker does
- Takes well-known attack methods into account
- Takes the type of users (=potential attacker) into account
- Reduces the evaluation effort for interfaces that are not part of the attack surface
  - All interfaces to trusted external entities
- Focuses on the areas where most vulnerabilities have been identified in the past
  - More likely to identify security problems
- Reduces the evaluation effort on the assessment of the correctness of the security functionality
  - Less design analysis and more testing for those

Suggestions for CC Improvements

- Request more detailed information on the implementation of interfaces that are part of the attack surface
  - Not just their parameter and effects
- Request more detailed information how common attack methods are addressed by the development
- Relax requirements on interfaces that are not part of the attack surface
- Relax design requirements for security functionality
  - Focus more on testing for correctness issues

Suggested CC Additions

- ADV_FSP.4
  - ADV_FSP.4.3E
    The evaluator **shall relate** the functional specification to the threat agents in order to determine the attack surface and the potential attacker.
  - ADV_FSP.4.4E
    The evaluator **shall determine** the type of attacks for each threat agent and as input to testing and the vulnerability analysis.

Suggested CC Additions

- ADV_IMP.1
  - ADV_IMP.1.4C
    The mapping between the TOE design description and the sample of the implementation representation **shall contain** a mapping of all TSFI that are part of the attack surface to their entry point within the TSF.
  - ADV_IMP1.2E
    The evaluator **shall confirm** that the mapping from the TOE design description to the implementation representation allows to identify the entry points of all TSFI that are part of the attack surface.
  - ADV_IMP.1.3E
    The evaluator **shall select** a sample of the TSFI that are part of the attack surface and analyze this sample for correct and complete parameter validation, exploitable buffer overflow, exploitable race conditions and potential privilege escalation.

Suggested CC Additions

- ATE_COV.2
  - ATE_COV.2.3C
    The analysis of test coverage **shall identify** the test cases where TSFI have been tested with invalid parameter.
  - ATE_COV2.2E
    The evaluator **shall confirm** that the test coverage includes test of all TSFI that are part of the attack surface with a sufficient number of test cases that test the TSFI with invalid parameter.
- ATE_IND.2
  - ATE_IND.2.4E
    The evaluator **shall test** a subset of the TSFI that are part of the attack surface using parameter values likely to be used by a threat agent attempting an attack.

Suggested CC Additions

- AVA_VAN.2
  - AVA_VAN.2.5E
    The evaluator **shall perform** an analysis of the attack surface and the threat agents to identify attack methods likely to be used by the threat agents.
  - AVA_VAN2.6E
    The evaluator **shall derive** penetration tests related to the attack surface and the identified attack methods, taking into account the developer testing and his analysis of the implementation representation.
  - AVA_VAN.2.7E
    The evaluator **shall conduct** the penetration tests related to the attack surface and the identified attack methods.

Summary

- CC now is too much focused on “correctness” of the security functionality
- Attacks usually use “side effects” of functions
  - Buffer overflow
  - Injection attacks
- Shifting focus on the attack surface increases the likelihood of identifying security problems
  - Increases the acceptance of evaluations

Questions?

Contact information:

Helmut Kurth helmut@atsec.com