Security-informed safety case approach to Analysing MILS Systems

Kateryna Netkachova, Kevin Müller, Michael Paulitsch, Robin Bloomfield

Collaboration: City University London, Adelard LLP, Airbus Group Innovations, Thales

International Workshop on MILS: Architecture and Assurance for Secure System
20 January 2015
Structure of the presentation

1. Introduction
   - Projects: SESAMO, EURO-MILS, CEDRICS
   - Integrated security and safety solution

2. Safety cases, security-informed safety cases

3. The layered assurance approach

4. Application to the MILS use case

5. Discussions and next steps
Safety Cases

**Safety case** – a documented body of evidence that provides a convincing and valid argument that a system is adequately safe for a given application in a given environment.

- Overall approach
- Claims, Argument, Evidence
- Top claim
- Split into sub-claims
- Structure of argumentation
Security-informed Safety Cases

**Justification of safety which specifically takes into account the impact of security.**

- Security consideration
- Impact on the Case Structure
- Some observations

Supply chain integrity.
Malicious events post deployment.
Design changes to address user interactions, training, configuration, vulnerabilities. Additional functional requirements that implement security controls.
Possible exploitation of the device/service to attack itself or others.
Levels of abstraction

- **L0 Policy and requirements** – the highest level of abstraction where the system represents its requirements, and defines safety and security policies and their interaction;

- **L1 Architectural layer** – the intermediate level where the abstract system components and architecture are analysed;

- **L2 Implementation layer** – the detailed level where the implementation of specific components and their integration within the specific system architecture are scrutinised.
Application to the MILS Systems

• Safety case perspective – not common using the CAE structure in avionics
• Case Study: MILS-based gateway controlling information flow between aircraft security domains
• Details of the approach
• Some observations from the Case Study
• Layered Assurance, Compositional Trustworthiness (LAW)
• Further directions and improvements
Use Case (ARINC 811)

How to securely connect multiple domains?

- **Maximum Security: Isolated Networks**
- **Lowest Costs: Shared Networks**

**Solution:** Gateway between domains
Logical scope

Domain A
High security level

Gateway

Domain B
Low security level

- Interfaces for communication outside the domain
- Bidirectional communication link

<table>
<thead>
<tr>
<th>domain A</th>
<th>User partitions</th>
<th>domain B</th>
</tr>
</thead>
<tbody>
<tr>
<td>App A1</td>
<td>Gateway</td>
<td>App B1</td>
</tr>
<tr>
<td>App A2</td>
<td></td>
<td>App B2</td>
</tr>
</tbody>
</table>

Separation kernel
High-level View of Gateway Components
L0 Policy and requirements

At L0 we consider both security and safety and we wish to claim that the policies are adequate. We address this by considering two main aspects:

- The definition of the individual policies
- The interaction of the policies
Defining integrated policy

Safety difficulty
- special operation modes
- severe operational challenge
- routine operational challenge
- minimum operational challenge

Security threat level
- Safety dominates
- Integrated policy and benefits
- Resolution of conflicts
- Security dominates
L0 – sketch of the general case

- Interaction between safety requirements and security policies is understood and trade-offs satisfactory
- Safety and safety policies are adequate
- Decomp
- Attributes of good policy
- Well defined properties
- Well defined environment
- Valid
- Other quality attributes
- Analysis of policy interactions and tradeoffs
- Constraints defined
- Addresses changes to threats (over time)
- Attacker capability analysis (intelligence)
- Security policy justification
- Safety requirements justification
- Safety requirements
- Security policy
- Safety analysis
- Security and safety interaction analysis

- Well defined - functional, non-functional properties
- Will map directly to the L1
- Will map to L1 and L2 when looking into personnel, incident reporting, component selection etc

- Security policy
- Valid
- Constraints defined
- Addresses changes to threats (over time)
- Attacker capability analysis (intelligence)
- Security policy justification
- Safety requirements justification
- Safety requirements
- Security policy
- Safety analysis
- Security analysis
L0 – sketch of the gateway case
Scope of the system

Context DFD representing the highest level view of the system (SDL Threat Modelling Tool).
L1 – architectural level

At this level various methods are used to analyse the system:

- A guideword based approach derived from the safety HAZOP analysis.
- An analysis of trust relationships
- Construction of attack scenarios and attack graphs
- STRIDE, the Microsoft threat modelling approach

In order to construct a case, at this level we need to take into account:

- The output from the L0 level of abstraction
- The identified and revised critical safety and security properties of the system
- Components that play essential roles in enforcing the critical properties
- A high-level architecture of the system representing components and their interaction
- Dynamic aspect to consider possible changes to the system in the future.
Level-n DFD model (SDL Threat Modelling Tool)
L1 general case

Critical properties of the system are enforced by the architecture and its components

- Critical properties are enforced initially
  - Consider critical properties
    - Critical property 1 is enforced
    - Critical property 2 is enforced
    - Critical property n is enforced
      - Component A
      - Component B
      - Component N

- Critical properties will continue to be enforced with any future changes
  - Phases of change accommodation lifecycle
    - Decision process OK
      - Updates developed correctly
      - Updates deployed correctly
      - Development process
    - Split by reasons
      - Will be expanded at L2 to analyse the implementation of the components
    - Decision for change as a response to failures or anomalies is made correctly
      - Decision for change due to the environment or objectives changes is made correctly
      - Monitoring, auditing components and based on the audits
## Hazop guidewords with respect to security

<table>
<thead>
<tr>
<th>Example guidewords</th>
<th>Impact on security attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Confidentiality</td>
</tr>
<tr>
<td>Late/too soon</td>
<td></td>
</tr>
<tr>
<td>As well as</td>
<td>Additional info (of same class, of different class)</td>
</tr>
<tr>
<td>Wrong</td>
<td>Classification (so inadvertent high to low)</td>
</tr>
<tr>
<td>More/less/intermittent</td>
<td></td>
</tr>
</tbody>
</table>
Identifications of hazards

Service interface hazards:

• Denial of service:
  – Are channels isolated from each other?
  – Are there any application limits on resource consumption?

• Service guarantees:
  – What service level guarantees does the gateway provide?
  – Are there any end-to-end checks at the application level?

• Man-in-middle attacks:
  – How does the application know that it’s talking to the gateway?
  – How does an application know that a message has come from a different security domain?

System operational hazards:

• The gateway is configurable, so there are hazards relating to incorrect maintenance or configuration of the system
• The audit log might contain sensitive information and therefore needs to be protected
• Physical access to the gateway during flight is considered impracticable, so the main threats come during maintenance when the plane is on the ground
### Example of Hazop applied to the case study

<table>
<thead>
<tr>
<th>No.</th>
<th>Element</th>
<th>Guide word</th>
<th>Deviation</th>
<th>Possible causes</th>
<th>Consequences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function considered:</td>
<td>Connecting to the gateway</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Connect to channels</td>
<td>OTHER THAN</td>
<td>Application connects to a channel other than gateways.</td>
<td>Another application from the same domain pretends to be and acts like a gateway.</td>
<td>Man-in-middle attacks.</td>
</tr>
<tr>
<td>2</td>
<td>Connect to channels</td>
<td>MORE</td>
<td>Too many messages are sent to gateway channels.</td>
<td>Broken or compromised application is sending too many requests.</td>
<td>Denial of service.</td>
</tr>
<tr>
<td>Function considered:</td>
<td>Gateway filtering</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Filter messages going through the gateway</td>
<td>AS WELL AS</td>
<td>Additional messages are allowed to pass through the gateway.</td>
<td>Error in filter specification or implementation.</td>
<td>Leakage of confidential data.</td>
</tr>
</tbody>
</table>
Analysis of trust relationships

• Identify trust relationships:
  – Gateway trusts Administrator
  – Auditing system trusts Gateway
  – Applications trust separation kernel
  – Applications in the same domain trust each other
• Identify consequences of breach of trust
• Assess risk
• Design mitigations as appropriate
## Example of analysis

<table>
<thead>
<tr>
<th>Breach of Trust</th>
<th>Consequences</th>
<th>Mitigation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gateway – Administrator</td>
<td>High Denial of service, loss of data integrity and confidentiality, man in the middle attack.</td>
<td>All security policies have to be operating and have to be identified by some authority. The gateway will only accept these security policies.</td>
</tr>
<tr>
<td>Gateway – Audit logs</td>
<td>Medium Loss of accountability and nonrepudiation, possible impact on confidentiality</td>
<td>Applications located in the domains can have their own logs documenting what they sent. No confidential data or data that can help facilitate an attack should be stored in logs.</td>
</tr>
</tbody>
</table>
STRIDE threat modelling (Microsoft)

STRIDE stands for:

- Spoofing (impersonating someone else)
- Tampering (modifying data)
- Repudiation (claiming not to have performed an action)
- Information disclosure (loss of confidentiality)
- Denial of service (deny or degrade service to valid users)
- Elevation of privilege (gain privileged access)
Attack scenarios – example 1

- Spoofing
Attack scenarios – example 2

- Spoofing, tampering
Attack scenarios – example 3

• Denial of service
Attack scenarios – example 4

Components required for processing the data stream between the two domains

- Receiver Component
- HTTP Chain
- Transmitter Component
- TFTP Chain
- Auditing
- NIC
- Transmitter Component
- TFTP Chain
- HTTP Chain
- Receiver Component
- NIC
- Domain A
- Domain B
## Example of STRIDE applied to the gateway

<table>
<thead>
<tr>
<th>Threat type</th>
<th>Security property</th>
<th>Brief explanation</th>
<th>Use case examples</th>
<th>Mitigation</th>
</tr>
</thead>
</table>
| Spoofing    | Authenticity      | Impersonating someone else. | 1) Application from domain B pretends to be a gateway or an application from domain A and sends something to domain B users.  
2) One application from domain B pretends to be another application from domain B and requests something from the gateway. | The inter-domain communication is controlled by a MILS separation kernel. The entire system is assembled and configured by a knowledgeable and highly trusted system integrator. |
| Tampering   | Integrity         | Modifying data    | One application from domain B intercepts and modifies the data sent to or from another application.  
[Man in the middle attack] | The inter-domain communication is controlled by a MILS separation kernel which prevents any interceptions and ensures the integrity of the messages. |
L1 gateway sketch

Critical properties of the system are enforced by the architecture and its components

Communication between domains is controlled properly

New binary file is deployed properly

Gateway application

Separation kernel

Critical information flow properties are enforced initially

Decisions for changes are made correctly

Decision making process

Split by reasons

Development process

Audit analysis

Decision for change due to the environment or objectives changes is made correctly

Monitoring and auditing components

Decision for change as a response to failures or anomalies is made correctly

Deployment process

Phases of change accommodation lifecycle

Any new binary file is correct

Should be expanded at L2 to analyse the implementation of the components

Critical information flow properties will continue to be enforced with any future changes

Aplications are separated and there is no unexpected communication bw them
**L2 – implementation level**

At this level we develop a detailed CAE structure to explain the behaviour of the specific components. This involves:

- Using the output from the L1 level of abstraction;
- Analysing the implementation details of every critical component;
- Creating an argument structure and elaborating the evidence to show that all the critical properties of the system are enforced;
- Documenting the results and providing traceability to the appropriate L0 and L1 security-informed safety case elements;

The case created at L2 level of abstraction is based on two types of technical information:

- General technical information produced and supplied with the components as part of the normal development process;
- Context-specific technical details derived from the analysis of the specific system implementation
PikeOS development & configuration *

* SYSGO AG, Using PikeOS, v3.4, 2014
L2 gateway fragment 1
L2 gateway fragment 2

PikeOS, configuration gateway app and the resulting ROM binaries are generated properly.

Building process:
- Binary specification file is correct
- Non-modified binary image is used
- Binary generator tool is OK

During/after:
- Binary not modified during the generation
- Binary not modified after it is generated

Protection/check:
- Check is performed before the binary code is run
- Signature and hash protection implemented
- Signed (against malicious)
- Binary hash is added and checked

No malicious code running during the binary generation
- Not corrupted
- OS is protected

Malicious/ non-malicious changes:
- Signed (against malicious)
- Binary hash is added and checked

Non-modified binary image is used
- RAM with autocorrection is used
L2 gateway sketch linked to Security Target
Discussions and next steps

General directions:
• Integrated security and safety process
• Impact of security on safety cases
• Safety case perspective in avionics

Some issues of Layered Assurance:
• Compositionality and composability
  – Topology, CAE structure
  – CAE Building Blocks
• Incremental certification and polymorphism
  – Impact analysis of changes on the assurance: revisiting aspects of CAE, change cases
• Abstraction layers
  – Three levels of abstraction, can be deployed recursively
  – Divide and conquer approach with different focus, lower risks

Additional research:
• Formalisation of reasoning within cases, linkage to formal models
• CAE building blocks tool support
• Further mapping to Common Criteria, other approaches

Anything else? Suggestions welcome!
Thank you for your attention! 😊