An Overview of Security requirements

Ludovic Apvrille
ludovic.apvrille@telecom-paris.fr

Journées GDR-GPL - groupe Ingénierie des Exigences
Plan

Introduction
LabSoC - Telecom Paris
Security

Security requirements

Security in SysML-Sec

Resources
Introduction

Security requirements

Security in SysML-Sec

Resources

Telecom Paris - LabSoC
Security in Dev. Cycles
Security in Dev. Cycles (Cont.)

- Security Verification
- System
- Security Countermeasures
- Security Requirements
- Attacks
- Vulnerabilities
- Security Risk
- Attacker Model
- evaluatedFrom
- takes as input
- satisfies
- target
- prevent
- respect
- rely on
- address

5/28 2021
Une école de l’IMT
Security requirements
Security Requirements

Definitions

- Defined in the scope of FP7 EVITA
- Automotive embedded architecture
  - Include the onboard networks

Hacker vs. attacker

- **Hacker**: smart use of objects / systems
- **Attacker**: criminal / terrorist
  - Financial gain
  - Harm / injury
Confidentiality

Confidentiality is satisfied when **authorized entities** are the only ones that can know a given **quantum of information**

**Example of requirements**

- The content of Messages sent from A to B shall be known only by A and B
- The state of a state machine shall be known only by its execution engine

**Typical countermeasures**

Message ciphering with secret symmetric keys
Privacy

*Privacy is guaranteed if the relation between the entity and the set of information is confidential.*

Typical sub-categories: anonymity, unlinkability.

**Example of requirements**

- In a social network, for non administrator users, the user of a message shall not be linkable to that message but two messages sent by the same user shall be linkable to each other.

**Typical countermeasures**

Data anonymisation, etc.
Integrity

*Integrity is satisfied when a *quantum of information* has not been modified between *two observations*.*

(Integrity is also called ”weak authenticity”)

**Example of requirements**

- The system shall ensure the integrity of messages sent from A to B
- The integrity of the instructions executed on the system processor shall be ensured

**Typical countermeasures**

Message Authentication Code with secret keys
Data origin authenticity

Data origin authenticity is satisfied when the data (quantum of information) truly originates from the author

(authenticity on origin is also called ”strong authenticity”)

Example of requirements

- All information received from sensors by the main controller shall be authentic in terms of origin.

Typical countermeasures

Asymmetric cryptography (public / private keys) with certificates provided by trusted authorities
Non-Repudiation

The non-repudiation of an action is guaranteed if it is impossible for the entity that performed the action to claim that it did not perform this action.

Example of requirements

- The payment system shall guarantee that neither the payer nor the billing system can deny a transaction once it has been performed.

Typical countermeasure

Digital signature (Identity, certificate, MAC, ...)

11/28 2021
Controlled Access (a.k.a. Authorization)

Controlled access is guaranteed if specified **entities** are the only entities that can perform the **actions** or access the **information**

Example of requirements

- Only explicitly authorized users shall be able to execute processes on the computer
- Controlled access to read data from a hard disk must be ensured

Typical countermeasures

User management (hashes passwords), firewalls.
Freshness

*Freshness is satisfied if a quantum of information received by an entity at the given time is not a copy of the same information received by the same or another entity in the past* (Usually related to replay attacks)

**Example of requirements**

- Freshness of all messages sent from A to B must be ensured.
- Execution of instruction in processor P must apply only to fresh instructions

**Typical countermeasure**

Timestamps / data id with integrity and authencity
Availability

*Availability is satisfied when a *service* or a *physical device* is operational*

(Usually related to Denial of Service Attacks - DoS)

Example of requirements

- The webserver must always respond in less than 1 second to requests
- The availability of the flight management system must be ensured

Typical countermeasure

Firewalls (traffic shaping, bans), redundancy, etc.
And many others . . .

. . . That are often domain-dependent.

E-voting system¹

- **Eligibility, Uniqueness**: Only eligible voters must be able to vote, and exactly once
- **Individual verifiability**: a voter can check that her/his own vote is correctly counted
- **Universal verifiability**: everyone can verify that all valid votes, and no others, were counted
- . . .

Method

1. Perform a SysML design (from functional requirements)
2. Define Security requirements
3. Relate the security requirements to SysML elements
4. Define an attacker model and attacks
5. Describe security requirements as security properties
6. Perform security verifications
7. Iterate . . .
Functional Model

Block System

- in chin(Message msg)
- out chout(Message msg)

Block Alice
- secretData : int;
- m : Message;

Block Bob
- m : Message;
- receivedData : int;

Message
- data : int;

Waiting for message
- chin(m)

Data stored
- messageReceived
- receivedData = m.data

Received data
- dataStored
- data : int;
Security Requirements in SysML

- **Integrity**: Integrity shall be satisfied for all messages exchanged between Alice and Bob.
  - ID=0
  - Text="Integrity shall be satisfied for all messages exchanged between Alice and Bob"
  - Kind="Non-functional"
  - Risk="Medium"
  - Reference elements="Alice ; Bob"
  - Targeted attacks=""
- **Authenticity**: All messages exchanged between Alice and Bob shall be authentic.
  - ID=0
  - Text="All messages exchanged between Alice and Bob shall be authentic"
  - Kind="Non-functional"
  - Risk="Medium"
  - Reference elements="Alice ; Bob"
  - Targeted attacks=""
- **Confidentiality**: The content of messages exchanged between Alice and Bob shall remain confidential.
  - ID=0
  - Text="The content of messages exchanged between Alice and Bob shall remain confidential"
  - Kind="Non-functional"
  - Risk="Medium"
  - Reference elements="Alice ; Bob"
  - Targeted attacks=""
- **Communication Between Alice And Bob**: The communication between Alice and Bob shall be secure.
  - ID=1
  - Text="The communication between Alice and Bob shall be secure"
  - Kind="Functional"
  - Risk="Low"
  - Reference elements=""
Security Requirements Related to Design

- **CommunicationBetweenAliceAndBob**
  - ID=1
  - Text="The communication between Alice and Bob shall be secure"
  - Kind="Functional"
  - Risk="Low"
  - Reference elements="Alice ; Bob"
  - Targeted attacks=""

- **Integrity**
  - ID=3
  - Text="Integrity shall be satisfied for all messages exchanged between Alice and Bob"
  - Kind="Non-functional"
  - Risk="Medium"
  - Reference elements="Alice ; Bob"
  - Targeted attacks=""

- **Confidentiality**
  - ID=2
  - Text="The content of messages exchanged between Alice and Bob shall remain confidential"
  - Kind="Non-functional"
  - Risk="Medium"
  - Reference elements="Alice ; Bob"
  - Targeted attacks=""

- **Authenticity**
  - ID=4
  - Text="All messages exchanged between Alice and Bob shall be authentic"
  - Kind="Non-functional"
  - Risk="Medium"
  - Reference elements="Alice ; Bob"
  - Targeted attacks=""

- **IntegrityOfMessages**
  - ID=6
  - Text="The integrity of message "m" sent from Alice to Bob shall be ensured"
  - Kind="Integrity"
  - Risk="Medium"
  - Reference elements=""

- **ConfidentialityOfAttributes**
  - ID=5
  - Text="The attribute "secretData" of Alice shall remain confidential"
  - Kind="Confidentiality"
  - Risk="Medium"
  - Reference elements=""

- **AuthenticityOfMessages**
  - ID=7
  - Text="The authenticity of message "m" sent from Alice to Bob shall be ensured"
  - Kind="Data origin authenticity"
  - Risk="Medium"
  - Reference elements=""
Attacker Model and Attacks

Attacker model: Dolev-Yao
The attacker can thus read / inject information from/to public buses
Adding Attacker Model and Security Properties

Security features

Security Property
- #Confidentiality Alice.secretData
- #Authenticity Alice.sendingMessage.m Bob.dataStored.m
Security Verification (1)

Security features

**Security Property**
- #Confidentiality Alice.secretData
- #Authenticity Alice.sendingMessage.m Bob.dataStored.m
Taking into account Confidentiality

Adding a pre-shared keys to Alice and Bob
Adding the necessary methods to Alice and Bob
Cyphering sent messages
Decyphering sent messages

Security features
#InitialSystemKnowledge Alice.sk Bob.sk

Security Property
#Confidentiality Alice.secretData
#Authenticity Alice.sendingMessage.m Bob.dataStored.m

<<datatype>>
Key
- data : int;
block
System
~ in chin(Message msg)
~ out chout(Message msg)
cryptoblock
Alice
- secretData : int;
- m : Message;

Bob
- m1 : Message;
- m : Message;

<<datatype>>
Message
- data : int;

waitingForMessage
chin(m1)
dataStored
messageReceived
m = sdecrypt(m1, sk)
receivedData = m.data

~ in chin(Message msg)
~ out chout(Message msg)
Security Verification (1)

- Adding a pre-shared keys to Alice and Bob
- Adding the necessary methods to Alice and Bob
- Cyphering sent messages
- Decyphering sent messages

Security features
#InitialSystemKnowledge Alice.sk Bob.sk

Security Property
#Confidentiality Alice.secretData
#Authenticity Alice.sendingMessage.m Bob.dataStored.m

Block diagram:
- System
  - in chin(Message msg)
  - out chout(Message msg)
- Alice
  - secretData : int;
  - m : Message;
- Bob
  - m1 : Message;
  - m : Message;
Handling Authenticity

**Security features**
- #InitialSessionKnowledge Alice.sk Bob.sk

**Security Property**
- #Confidentiality Alice.secretData
- #Authenticity Alice.sendingMessage.m Bob.dataStored.m

```
<<datatype>>
Key
- data : int;
```

```
<<datatype>>
Message
- data : int;
```

```
<<datatype>>
block
System
```

```
~ in chin(Message msg)
~ out chout(Message msg)
```

```
cryptoblock
Alice
- secretData : int;
- m : Message;
```

```
cryptoblock
Bob
- m1 : Message;
- m : Message;
```
Finally (1/2) . . .

Security features
#InitialSessionKnowledge Alice.sk Bob.sk

Security Property
#Confidentiality Alice.secretData
#Authenticity Alice.sendingMessage.m Bob.dataStored.m

![Diagram showing security features and property with blocks and messages]
Finally (2/2)

- **Security Requirement**: Integrity
  - ID=3
  - Text: Integrity shall be satisfied for all messages exchanged between Alice and Bob
  - Kind: Non-functional
  - Risk: Medium
  - Reference elements: Alice ; Bob
  - Targeted attacks: 

- **Security Requirement**: Confidentiality
  - ID=2
  - Text: The content of messages exchanged between Alice and Bob shall remain confidential
  - Kind: Non-functional
  - Risk: Medium
  - Reference elements: Alice ; Bob
  - Targeted attacks: 

- **Security Requirement**: Confidentiality of Attributes
  - ID=5
  - Text: The attribute "secretData" of Alice shall remain confidential
  - Kind: Confidentiality
  - Risk: Medium
  - Reference elements: 

- **Security Requirement**: Authenticity
  - ID=4
  - Text: All messages exchanged between Alice and Bob shall be authentic
  - Kind: Non-functional
  - Risk: Medium
  - Reference elements: Alice ; Bob
  - Targeted attacks: 

- **Requirement**: Integrity of Messages
  - ID=6
  - Text: The integrity of message "m" sent from Alice to Bob shall be ensured
  - Kind: Integrity
  - Risk: Medium
  - Reference elements: 

- **Requirement**: Confidentiality of Attributes
  - ID=5
  - Text: The attribute "secretData" of Alice shall remain confidential
  - Kind: Confidentiality
  - Risk: Medium
  - Reference elements: 

- **Element ref.**: DesignWithConfidentiality
  - ID=2

- **Element ref.**: DesignWithAuthenticity
  - ID=4
To Go Further . . .

- **TTool**: http://ttool.telecom-paris.fr
- **SysML-Sec**: http://sysml-sec.telecom-paris.fr