BWCPS-Generator: Security Aspects Integration

To understand the need for integrating security aspects to the BWCPS-Generator project, here are two use cases extracted from this document [1], along with the security concerns deducted from them and the implementation executed as security measures.

Use Case 1

Manufacturer uses more cost-effective machine than contractually agreed, or selects for him more favorable process parameters (costs, time) or uses cheaper raw materials.

Threats

- Replay with process data of another production run (different batch on the same machine)

- Process data from correct machine, but delivered batch from another machine

- Manipulation of the acquired process data

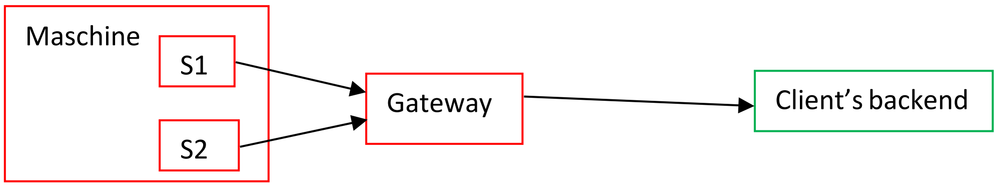
- Manipulation of the sensors, e.g. also replacement of a sensor (insensitive temperature sensor, provides supposedly stable process temperature)

Process-Data at Risk

Machine identification and machine parameters as well as measured values of the machine

Involved Systems

Sensors, Gateway, Backend of the client



Security Concerns and measures suggested

- Traceability: Assignment of the data to the machine (and produced batch)

    - Unique identification of edge devices through:

        - Asymmetric-cryptography (sensors sign data and time stamp with secret key)

        - Verification either in sink node or in gateway

- Data integrity: Prevention against manipulation of data

    - Signature of all received data (by the gateway)

    - Possible exchange of public keys

Use Case 2

The machine manufacturer accesses more process data collected by the machine than contractually agreed and thus obtains business secrets.

- Machine manufacturer monetizes aggregated data of its customers

- Machine manufacturer uses knowledge of processes of the customer (machine operator) to

  (a) advise the customer or

  (b) advise third parties

Process-Data at Risk

Machine parameters and measured values of the machine

Security Concerns and measures suggested

- Confidentiality: Protection of process data

Encryption of data by all parties with own key + decryption only possible when all keys are known (Key exchange)

Implementation

To integrate configurable security aspects to the project, a few adjustments had to be done to the metamodel. A SecurityMeasure-enum was introduced with three literals: ENCRYPT, AUTHENTICATE and ENC-THEN-AUTH, as well as an attribute in DataLink-entity of type SecurityMeasure.

The generator would use these model entries for generation. If any datalink needs security-related implementation, a security package in NodeConfiguration-project would be included, along with following additions

- The generated nodes that need security-related methods implement SecurableNode Interface.

- A SecurityManager instance manages all security related operations for every node using Java Cryptography Extension libraries. Another implementation of the SecurityManager (SecurityManagerCrypto) that uses the apache.commons.crypto library for encryption would be included but not yet used in the current implementation of the nodes.

- The asymmetric encryption algorithm for key exchange used is RSA.*

- The symmetric encryption algorithm for data exchange used is AES. *

- The symmetric algorithm used for signing data is HMAC.*

In the following table are the changes made to the Metamodel and the generated project according to the security concern these changes are meant to achieve.

Security Concern	Security Measures Chosen*	Technical application		Implemented
		BWCPS-Metamodel Changes	Generated implementation	
Data Integrity	-Exchange secret keys asymmetrically between nodes -Sign data symmetrically using common key	Enum-Literal: AUTHENTICATE	-Sign method calculates the mac signature and appends it to the data to send. -Verify method calculates the reference mac signature and compares it to the mac signature received with constant time comparing function.	✓
Traceability				
Confidentiality	-Exchange secret keys asymmetrically between nodes -Encrypt data symmetrically using common key	Enum-Literal: ENCRYPT	Encrypt method encrypts the message and appends it to the initialization vector (IV) used. Decrypt method reconstruct IV and decrypts message.	✓
CIA	-Exchange secret keys asymmetrically between nodes -Encrypt-Then-Authenticate policy using common key	Enum-Literal: ENC-THEN-AUTH	EncryptThenSign encrypts the message then signs it and appends the initialization vector (IV) used to the signature and the encrypted message. VerifyThenDecrypt verifies the signature like mentioned above and if it is valid, decrypts the message received.	✓

*It is suggested to use pure asymmetric cryptography in all security measures taken. As asymmetric cryptography has the disadvantage of being slow as it is explained in this paper [2], it was decided to replace this policy with a hybrid proposition, which was implemented according to this article [3].

**The time stamp for traceability was considered a part of the SensIDL dataset and is not going to be considered in the implementation.

[1] Sichere und vertrauensvolle Prozessdatenverarbeitung: Systematik der Risiken, Akteure, Schutzziele und Schutzmaßnahmen anhand von Anwendungsfällen in Produktionsanlagen

[2] Comparison of Symmetric and Asymmetric Cryptography with Existing Vulnerabilities and Countermeasures: Kumar, Y., Munjal, R., Sharma, H. (2011)

[3] Using AES with RSA for File Encryption and Decryption in Java: AuthorJay Sridhar (2017)