# Leveraging Determinism in Industrial Control Systems for Advanced Anomaly Detection and Reliable Security Configuration

Hadeli Hadeli, Ragnar Schierholz, Markus Braendle, Cristian Tuduce, Sebastian Obermeier
ABB Switzerland Ltd. Corporate Research - Industrial Software Systems
Baden-Daettwil, Switzerland
hadeli.hadeli@ch.abb.com

**Abstract:** Today, industrial automation and control systems (IACS) are more and more based on common IT technologies, leveraging open standards. Because of that IT security has become an important issue and a key challenge.

This paper presents an approach that leverages the unique characteristics of IACS, in particular the formal system description and the deterministic behavior. This allows reliably detecting anomalies and reproducibly generating configurations for security mechanisms such as firewalls. In particular, this paper elaborates how to extend common IDS technology to also detect an IACS specific anomaly, the missing of required traffic, for two exemplary applications: electrical substation automation and process automation.

**Keywords:** intrusion detection systems, anomaly detection, system description files, SCD, SCL, IEC 61850

## 1 Introduction

Cyber security for Industrial Automation and Control Systems (IACS) has become a much discussed topic in the recent past. Despite the lack of solid data on incidents and attacks available for research and still comparatively few publicly known vulnerabilities in IACS, there is common agreement among the experts that IACS need to better address cyber security [1][2][3][4]. Documents such as NERC CIP [5] reflect this need, and standards such as ISA 99/IEC 62443 [6] or IEC 62351 [7] are trying to help the industry achieve better security levels [8].

### 1.1 Problem Statement

Computers are used in almost every system nowadays. This includes the IACS. While in the past, IACS used their own proprietary software elements such as proprietary protocols or components, it is becoming more common that IACS use commercial-off-the-shelf (COTS) components as their foundation. This includes hardware (common servers, PC's and network infrastructure), operating systems (e.g. Microsoft Windows), databases (e.g. Microsoft SQL Server) and network technologies (e.g. Ethernet and

TCP/IP). Adopting these COTS components allows the industry to develop more efficiently and sell IACS at lower prices. Nonetheless, the introduction of COTS also introduces some risks associated with the components as the COTS. For instance, COTS components are designed for consumer or enterprise use, not for use in critical infrastructures such as IACS [1][3]. Thus, it may not prioritize availability of the entire systems, which is paramount to the IACS [2][4].

Traditional IACS software relied much on the special purpose components and isolation of systems to achieve security. Thus, traditional IACS technology itself has little to no protection against attacks. However, it is exposed to such attacks now by increasing interconnectivity between IACS and other systems such as enterprise software. Another example concerns on the security of the software itself. The use of COTS software in IACS does not mean that it is more secure than the use of COTS in its business environment counterparts. By using COTS in IACS, the entire control systems inherit all the security risks that are designated for particular COTS.

Adding to these risks is that IACS are increasingly connected to external systems such as enterprise systems or remote access solutions and that organizations that use IACS are primarily concerned about their physical process, e.g. a manufacturing plant or a power transmission and distribution system. As a consequence, there should be a better way to deploy the COTS in IACS environment to ease the reliable and safe IACS operation in a secure way. Examples of mechanisms to secure IACS involving COTS components. Furthermore, traditional IACS technology which has been designed for isolated system use and thus has little to no protection against attacks, is exposed to such attacks now by increasing interconnectivity between IACS and other systems such as enterprise software including automatic deployment of updates and patches or intrusion detection or prevention systems (IDS/IPS).

All in all, current deployment of IACS faces two critical concerns:

1. IACS have higher reliability and availability requirements than enterprise or even consumer systems and therefore, also the security mechanisms have to be more reliable as well [9]. For example current IDS/IPS technology shows false positive rates [10], which could lead to the blocking of critical messages in an IACS. Locking user accounts due to failed authentication attempts as is common in enterprise systems could lock out a legitimate user in an emergency situation.
2. The addition of security mechanisms increases the complexity of commissioning and operations. The mechanisms have to be configured properly with consideration of the specific IACS they are supposed to protect. The configuration has to be maintained in order to maintain the level of protection. For example virus signatures have to be updated frequently otherwise the anti-virus software will be useless. This is a costly activity and often prone to error, especially when the maintenance has to be done manually and there is little to no formal guidance on how the configuration should be, for instance Wool [11] shows that many corporate firewalls are configured improperly.

Commonly, the IACS is installed and engineered once and only modified if the process is modified or continuously managed [9]. This is mainly due to the strict validation requirements in critical infrastructures which make changes quite costly due to extensive testing that has to be done. Such modifications usually happen under strict change management; they are extensively tested and documented in detail.

### 1.2 Contribution and Paper Organization

This paper covers the abovementioned concerns, and a solution is proposed for each of them. In this paper, the unique characteristics of communications in an IACS are highlighted. Moreover, this paper shows that it is possible to leverage the pre-engineered and well-defined deterministic traffic pattern and characteristics of IACS in order to setup better security for IACS. A framework to automatically generate configuration data for common security controls is presented and a new control taking advantage of specific characteristics of IACS is presented. Finally, a prototypical implementation of this approach is shown.

The remaining sections of the paper are organized as follows. The next section discusses unique characteristics of IACS network traffic. Section 3 provides an in-depth discussion on system description files and how they can be leveraged. Section 4 addresses the framework to automatically generate configuration data for security measures. Section 5 provides the prototyping and the result, while Section 6 concludes the paper.

## 2 Unique Characteristics of IACS

IACS usually control critical processes and thus their failure can quickly lead to consequences as drastic as injury, loss of life or environmental pollution. IACS have further unique characteristics that differentiate them from enterprise or even consumer systems. While enterprise systems change fairly frequently and expose a behavior that is difficult to predict, IACS are only changed under strict change management and they are highly deterministic in their behavior. This is particularly true for the communication patterns among the different components of a distributed IACS. An example of this determinism can be found in time constraints: certain messages are expected to be sent by a sender, and received by a receiver, in a given period of time or at a definite point of time. The semantics of this type of traffic can be different from industry to industry. In the following, we provide two examples to show this.

In an IEC 61850 [12][13] based substation automation system, Intelligent Electronic Devices (IED's) are sending out Generic Object Oriented Substation Events (GOOSE) messages at a certain frequency, such as 200Hz, to other IED's. These GOOSE messages are meant to inform the receiving IED's that the sender is active and ready to take part in providing particular logical functionality, for example an interlocking function or a protection function. In this setup, the substation automation system should ensure that the GOOSE messages arrive at the correct recipient and within a given time boundary. Failure to deliver the GOOSE message to the correct recipient at a correct time may result in a failure of the particular functions.

A similar setup can also be found in process automation applications. For instance, from time to time, sensors in a process automation system provide sample values to the controller. This is to inform the controller on the current state of the relevant part of the process. The sampling is done in a certain frequency (e.g. the sampling interval could be 500 msec, i.e. the rate is 2 Hz). A very simple example of this is a temperature sensor and a steam boiler controller. In a larger scale, this can be sensors that monitor the cooling system in a nuclear reactor. Similar to the previous example in substation application, when this sampling function fails, the steam boiler can be overheated, and can cause system failure or even casualties.

In an IACS, this type of anomalies, missing or delayed sampling messages, will be passed on to the control center. However, the central control system lacks information on the primary cause of this anomaly. Thus, it is also interesting to analyze this anomaly from security perspective. From security perspective, there can be two possible causes: intentional causes and non-intentional causes. Possible non-intentional causes are malfunctioning systems, e.g. IED's (in power system automation) or sensors (in process automation system), a broken network link, or misconfigured IED's. Examples of most likely intentional causes are vengeance by a disgruntled employee, or cyber attacks. For example a man-in-the-middle attack may modify and thereby delay a message.

Another aspect in the deployment of security up to this present time is the fact that available security measures only raise an alert when unexpected/undesired traffic, such as traffic known as malicious, appears in the system. No alert is raised on the disappearance of the expected/required traffic. This research addresses this issue and shows the importance of detecting such missing expected traffic.

Unlike in the business or enterprise environment, all components in the IACS are well planned and defined. In addition, the configuration for those components is well-planned and documented in system description information typically created during the engineering of a specific IACS. In most cases today, the configuration of IACS is stored in a form of electronic files. Having the electronic files enables a better archiving of the configuration settings and makes it easier to access particular details of the configurations. This file is further referred to as system description file in this paper.

One important part of the system description file contains a full description of the overall communication pattern in the IACS. In the ideal case, the communication pattern covers all communication related aspects in the system. This includes communicating actors, mean of transport for communication, type of communication, protocols, contents, and even the communication constraints such as timing constraints.

Available information in the system description file gives benefit in analyzing and deriving the legitimacy of traffic observed inside the IACS. With the given information, it is possible to derive a comprehensive communication model. Any deviation from that model can be seen as an anomaly or a sign of an intrusion. For example, assume that a process automation with IACS contains 3 entities, namely a, b and c. The communication pattern for this particular IACS is defined in a system description file as follows:

- Only the following protocols are allowed:
    - MMS – Manufacturing Message Specification (ISO/IEC 9506-1 and ISO/IEC 9506-2 [14]) – TCP port 102
    - MODBUS/TCP – TCP port 502 [15]
    - RNRP - Redundant Network Routing Protocol – UDP port 2423
- MMS traffic is used to update available entities in the system with the most recent system information. Each update should be received by all entities at every 200ms (5 Hz).
- The MODBUS/TCP traffic is used for different purposes in the IACS
- RNRP traffic is used for redundancy purpose

From the above-mentioned traffic pattern, it is possible to derive and determine in advance the expected communication patterns in the system. The result of the pre-classification based on the given short description is represented in Table 1.

<table>
<tr><td colspan="4">Received traffic</td></tr>
<tr><td colspan="3">Specified</td><td rowspan="2">Unspecified</td></tr>
<tr><td colspan="2">Has time constrained information (entity a, b, and c receive update via MMS every 200ms)</td><td>Has no time constraint (only MODBUS/TCP and RNRP allowed)</td></tr>
<tr><td>Fulfill the time constraint</td><td>Fail to fulfill the time constraint</td><td rowspan="2">Expected traffic (communication between entities using MODBUS/TCP and traffic for redundancy purpose)</td><td rowspan="2">Unexpected traffic (appearance of traffic type other than MMS, MODBUS/TCP and RNRP (particularly to/from entity a, b, and c))</td></tr>
<tr><td>Expected traffic (entity a, b, and c receive update via MMS every 200ms)</td><td>Unexpected / anomaly traffic (any of entity a, b, and c either receive update via MMS in the interval larger than 200ms or not receive any update at all)</td></tr>
</table>

*Table 1 – Example of Extracted Traffic Patterns from a System Description File*

## 2.1 System Description File in the Area of Power Systems Application

In the area of power systems application, electrical substations are used to route electric power through an energy transmission and distribution system. The flow of power has to be constantly monitored by substation automation systems, including IED's, in order to trip safety functionality in case of short circuits or other malfunctions. Nowadays, substation automation systems are often based on the IEC 61850 standard, which

defines all involved devices in a substation, their functions and associated communication in a system description file [16]. The IEC 61850 standard enables interoperability between compliant devices from different vendors. It uses Ethernet-based protocols as a mean of communication. In this system, the configuration of all devices in the substation is stored in a so-called Substation Configuration Description (SCD) file. This format is defined in the IEC 61850 Substation Configuration Language (SCL), which is based on XML. The SCD files are used for configuration of electrical substation devices and interoperable exchange of engineering data between engineering tools.

The SCD file contains the following information about substation automation systems [17]:

- Header description
- Substation description
- Communication description
- IED description
- DataTypeTemplates

The most relevant part for this research is the communication description part. By extracting information from communication description, it is possible to derive the following explicit information:

- SubNetwork. A SubNetwork groups one or more ConnectedAP (Connected Access Point) that support a given (communication) protocol type and which can communicate with each other.
- Message, telegram. This part tells how data (signal values) is grouped into messages, and with which characteristics are they sent (especially timing). Communication behaviors can be extracted from this part and mapped to a physical network.
- Signal flow. It describes any kind of data objects or attributes/signals (mostly coming from IED's) that are required in order by one or more logical nodes to perform a function.

Figure 1 provides an example of a communication part of a simple IEC 61850 SCD file. By observing the communication part of this SCD file, the following explicit information can be extracted: SubNetwork information, type of SubNetwork, IP address of any IEC 61850 based devices, IP subnet mask of devices, IP gateway, GOOSE control block settings, such as: multicast MAC address, application ID, VLAN priority, $t_{min}$, $t_{max}$, etc.

Furthermore, there is some information which is important to the operational of the system, however it is not explicitly written in the system description file. For the rest of the paper, this information is called implicit information. In the case of SCD file, there is information that is not written in the SCD file, but the information is important for

the operational of the substation automation system. This information can be acquired from discussion with the IEC 61850 experts.

- MMS traffic is a valid traffic. It is used for vertical communication in the substation automation systems [18]. Any entity can receive a report or request via MMS.
- Only devices with an explicit declaration of GOOSE control block can send out GOOSE messages
- GOOSE messages are never broadcasted outside the sender's IP subnet
- If a time server is installed, there will be time synchronization traffic (SNTP or IEEE 1588). According to the standard, only 512 variants of multicast MAC addresses are allocated.

There is device specific traffic, such as particular protocol(s) used between the device and the configuration tool

```
<Communication>
<SubNetwork name="AA1WA1" desc="8-MMS" type="8-MMS">
<ConnectedAP iedName="AA1D1Q09A1" apName="S1">
        <Address>
          <P type="IP">192.168.8.1</P>
          <P type="IP-SUBNET">255.255.0.0</P>
          <P type="IP-GATEWAY">192.168.1.1</P>
          <P type="OSI-AP-Title">1,3,9999,23</P>
          <P type="OSI-AE-Qualifier">23</P>
          <P type="OSI-TSEL">0001</P>
          <P type="OSI-PSEL">00000001</P>
          <P type="OSI-SSEL">0001</P>
        </Address>
     </ConnectedAP>

   <ConnectedAP iedName="AA1D1Q10A1" apName="S1">
        <Address>
          <P type="IP">192.168.9.1</P>
          <P type="IP-SUBNET">255.255.0.0</P>
          <P type="IP-GATEWAY">192.168.1.1</P>
          <P type="OSI-AP-Title">1,3,9999,23</P>
          <P type="OSI-AE-Qualifier">23</P>
          <P type="OSI-TSEL">0001</P>
          <P type="OSI-PSEL">00000001</P>
          <P type="OSI-SSEL">0001</P>
        </Address>
        <GSE ldInst="LD0" cbName="gcb_A">
          <Address>
          <P type="MAC-Address">01-0C-CD-01-00-00</P>
            <P type="APPID">3002</P>
            <P type="VLAN-PRIORITY">4</P>
          </Address>
          <MinTime unit="ms">4</MinTime>
          <MaxTime unit="ms">1000</MaxTime>
        </GSE>
      </ConnectedAP>
  </SubNetwork>
</Communication>
```

*Figure 1 – An Example of a Communication Block of an IEC 61850 SCD File*

## 2.2 System Description File in the Area of Process Automation Application

In the area of process automation applications, an example of a system description file can be taken from a specific ABB product, namely System 800xA, which is a distributed control system used to supervise and control a manufacturing system, or other industrial processes.

The system description file for System 800xA is called Setup Package File [19]. One important part from the Setup Package File is the NodeInformationOptions part. An extracted NodeInformationOptions of a System 800xA Setup Package File can be seen in Figure 2.

```
<NodeInformationOptions
Type0="operatorClient" Name0="Client"
Adapter10="172.16.4.71"
Adapter20="172.17.4.71"
Adapter30="-"
Adapter40="-"

Type1="AspectServer" Name1="AS"
Adapter11="172.16.4.21"
Adapter21="172.17.4.21"
Adapter31="-"
Adapter41="-"

Type2="ConnectivityServer" Name2="CS"
Adapter12="172.16.4.22"
Adapter22="172.17.4.22"
Adapter32="172.16.80.22"
Adapter42="172.17.80.22"
/>
```

*Figure 2 – An Example of a Very Simple System 800xA Setup Package File*

In this simple example of System 800xA Planner File, the following explicit information can be derived:

- There are three nodes in the system, namely: Operator Client, Aspect Server and Connectivity Server
- Every node has at least two adapters. One node has four network interfaces. Each network interface has a different IP address.

In addition to the abovementioned explicit information, implicit information can be derived by understanding the roles and setup of every node. The following list shows a simplified example of the extracted implicit information.

- The Operator Client node can receive input only via CSLib proprietary protocol
- The Aspect Server can be accessed via TCP port 3338 (anet-b – OMF data b) and 3339 (anet-l – OMF data l)

- The Connectivity Server communicates via TCP port 102 (MMS)
- By default, RNRP protocol (TCP/UDP port 2423) is used for redundancy purpose.

From the abovementioned examples (section 2.1 and 2.2), we have seen that the combination between explicit and implicit information provides sufficient information that can be used to create a model of the network traffic that is expected/desired. The next section discusses the framework that takes the system description file as input and generates the configuration for different security measures.

# 3 Leveraging the Model of Expected Traffic for a Better Security Setup

The previous section has elaborated in detail the captured information in the system description file. From the available information, it is possible to derive the model of expected communication. Since a control system is highly deterministic, anything that is not pre-specified can be considered as an anomaly.

To be able to gain the benefit of the determinism of the information, it is necessary to develop a framework that takes the system description files as input and construct the model of expected communication in the system. Next, the model can be later used to generate configuration files for different security measures. Such framework is called translation framework for the rest of this paper.

The translation framework works by taking the system description files as input. For different type of system description files, a specific parser has to be developed. The parser processes the system description file. Next, the process information is passed to the next component of the framework. This particular component constructs a model of the communication pattern of the system. In order to build a complete comprehensive model, the framework needs additional input, such as the implicit information.

Implicit information, from experts' discussion and other means, is structured and captured in machine-readable, for example XML-based, documents as well. These need to be parsed by a special parser. Afterwards, the parsed information from both explicit and implicit information are combined and built into a comprehensive communication model.

After having a complete comprehensive communication model, the next step is to generate the configuration files for different security measures and purposes. This is done by the configuration file generator. The configuration file generator is the processor that links and matches the requirement for the particular security measure with available information in the comprehensive communication model. Examples of security measures are firewall, intrusion detection systems, VPN connections, IPSec stacks, etc. In order to support a particular security measure, the information that is needed in order to configure such security measure must be available in the system description file in some form. For instance, to setup a firewall, at least the following information is needed: allowed protocols, ports, and connection peers.

By taking the generated configuration files as input, the security measures know the behavior towards certain type of traffic. Examples of the behavior are: to block or to allow traffic, to alert on malicious traffic or to alert on the omission/delay of required traffic. Note that a different generator module is needed for different types of security measures.

Afterwards, the generated configuration files are submitted to the security measures for further setup, operation and maintenance. The translation framework is shown in Figure 3.

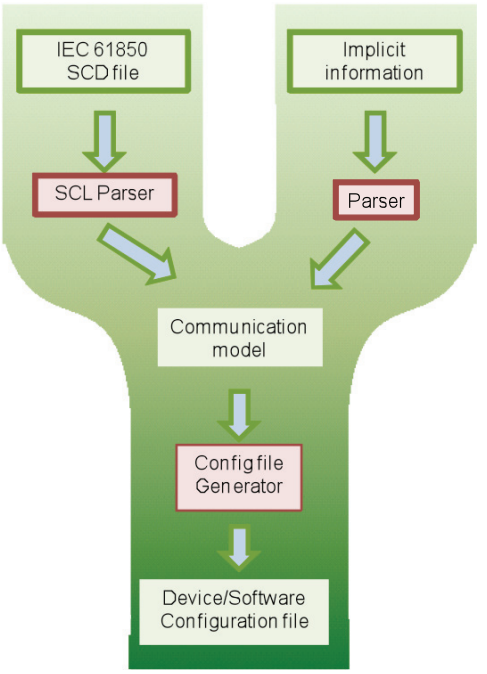


*Figure 3 – The Translation Framework*

To show how the overall system works, the next part presents the implementation of the framework and three examples of how the framework works.

# 4 Implementation and Prototyping

In order to prove the concept, a prototype is developed. This prototype generates configuration for different security measures. This paper presents how to use the input from system description files to generate configuration for missing or delayed traffic detector and firewall. The prototype consists of the following components, namely:

1. The parsers. The parsers of this prototype parse the IEC 61850 SCD file and ABB System 800xA Setup Package File as well as a custom file format capturing implicit information.
2. The communication model. The communication models is represented in an XML format
3. Security measures configurator. This processor is implemented in the XSLT language. It takes the communication model as input and then writes out suitable configuration for the particular security measure.

## 4.1 System Description File in the Area of Process Automation Application

Necessary inputs to build the communication models are explicit information and implicit information. In the developed prototype, the explicit information is stored in XML format and the implicit information is stored in an Excel sheet. In order to generate the communication model, the following constraints and requirements are taken into consideration:

1. The schema must represent the information about the topology of a network, and the constraints of the patterns of traffic allowed, forbidden and required between its nodes
2. The communication patterns described in the intermediate representation can be between IP addresses (IP to IP), Ethernet MAC addresses (MAC to MAC), but also mixed (IP to MAC or MAC to IP)
3. Every single node can have one or more IP or MAC addresses, and for each IP assigned to it there can be more one or more TCP/UDP ports it listens to, e.g. in the case of nodes which provide services to other client nodes
4. Every communication pattern, which is represented by an element named Behavior, can be described by 3 elements (see Figure 4):
    - an address of the source node
    - an address of the destination node
    - the policy of the pattern, which can be either of the followings:
        - Traffic that might be present (ALLOWED traffic)
        - Traffic that should not be present (FORBIDDEN traffic)
        - Traffic that should be present (REQUIRED traffic)

All behaviors are grouped in a sequence called Behaviors, which can contain one or more elements of type BehaviourType.

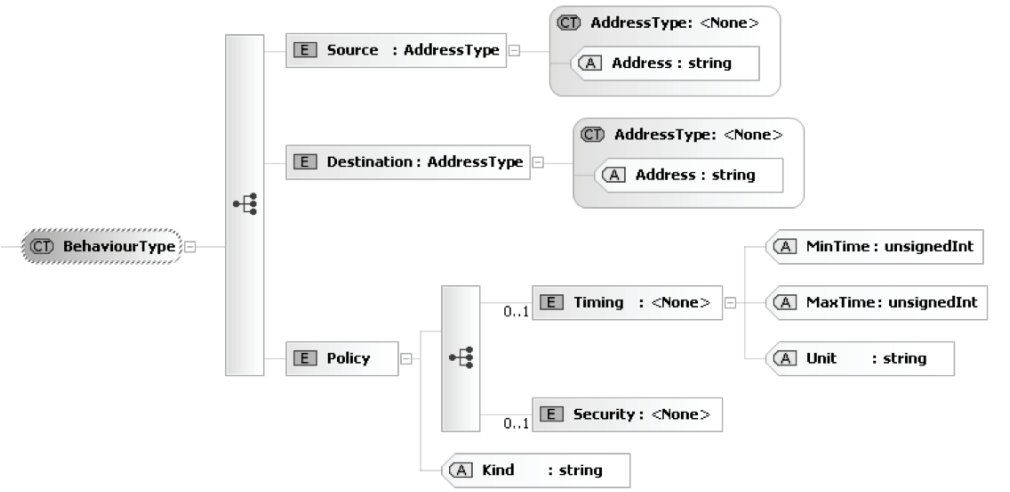


*Figure 4 – Behaviour Type*

## 4.2 Prototype to Detect Missing Expected Legitimate Traffic

From the developed prototype, it is possible to generate configuration for a missing/tardy traffic detector for both power systems application and process automation application. For this implementation, the open source Snort Intrusion Detection System [22] is chosen. However, Snort on its own cannot detect missing expected traffic. In order to enable Snort to detect missing expected traffic, a custom preprocessor is needed. Snort preprocessors are modules that inspect every packet that enters the network interface before the packet is passed to the detection engine. The preprocessor is used to detect any problem that cannot be detected with rules, i.e. attacks that cannot be detected by pattern matching. In this case, our preprocessor allows detection of Denial of Service attempts or delays by man-in-the-middle attacks, malfunctioning of network equipment inside substations, and in general of any situation that causes the absence or delay of required network traffic. The operational of the Snort IDS is illustrated in Figure 5.

In order for the preprocessor to work properly, the following inputs are needed:

1. Number of expected traffic types to be observed. Each traffic type is represented by a single timer.
2. Expected time interval of appearance for each type of traffic. This will determine the length of a timer.

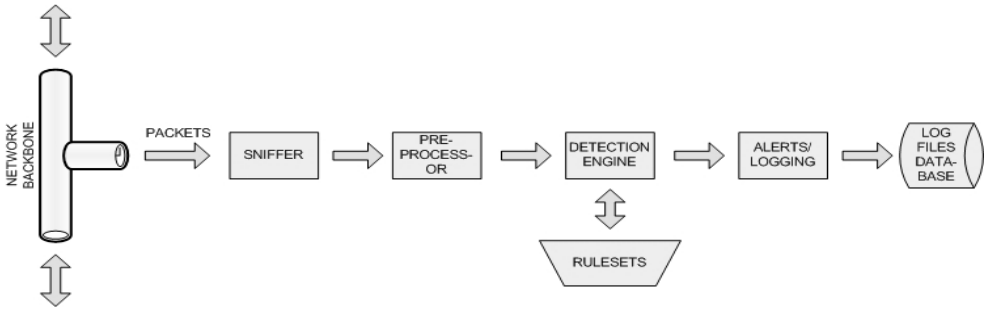


*Figure 5 – Architecture for Snort IDS*

Having taken the above-mentioned information, the preprocessor works as follows:

1. While initializing the preprocessor, a timer is created. The number of timers depends on the number of observed traffic types. For example, if there are three types of traffic to be observed, three timers are created.
2. When a timer is created, it is subsequently set to the time slightly bigger than the expected interval for each type of observed traffic.

   $t_{timer} = t_{wait} + t'$

   while: $t_{timer}$ = the set time at the timer; $t_{wait}$ = time on which the observed traffic appears; $t'$ = corrected time

   The reason of adding the "corrected time" is to provide a small interval for the preprocessor to check the packet that arrives right at the maximum waiting time. Note that the detector should not alert if the expected traffic is seen exactly in the specified time period. For instance, if a traffic type A is expected to arrive every 5 seconds, when the traffic arrives exactly at time 00:05, the detector should not alert that the traffic is delayed or missing. However, if it is not seen after the specific time, it should alert on the missing traffic. Another reason to add the "corrected time" is to provide a time for the preprocessor to reset its monitoring timer in time to avoid the generation of an alert, in this case a false positive.
3. When the expected traffic is observed, the timer is reset.
4. If the timer is expired and the expected traffic is not yet observed, an alert is raised.

### 4.2.1 Implementation for Power Systems Application

The prototype for substation automation systems is explained in more detail in [20]. This prototype takes the example as shown in Figure 1. From this example, the

configuration to detect missing GOOSE traffic is generated. The retrieved information from the system description file is as follows:

- Subnetwork name: AA1WA1
- Type of subnetwork: 8-MMS (IEC 61850)
- There are two entities in the system
- Default traffic in the system: MMS
- Only IED 2 can send out GOOSE control blocks
- GOOSE traffic from IED 2 is to be multicast between 4ms and 1000 ms

| IED 1: AA1D1Q09A1 | IED 2: AA1D1Q10A1 |
|---|---|
| • IP address: 192.168.8.1<br>• Subnet: 255.255.0.0<br>• IP gateway: 192.168.1.1 | • IP address: 192.168.9.1<br>• Subnet: 255.255.0.0<br>• IP gateway: 192.168.1.1<br>• GOOSE control block settings:<br>▪ Multicast MAD address: 01-0C-CD-01-00-00<br>▪ Application ID: 3002<br>▪ VLAN priority: 4<br>▪ $t_{min}$ = 4ms<br>▪ $t_{max}$= 1000 ms |

*Table 2 - Summary of the Retrieved Information per IED*

This information is further constructed into the comprehensive communication model. For this example, the focus is emphasized only on the information about GOOSE control blocks. From the retrieved information, it is clear that only IED AA1D1Q10A1 can send out GOOSE messages.

Furthermore, the GOOSE message has to be sent out every 4 to 1000 ms. The multicast address is 01-0C-CD-01-00-00. This information implies that an anomaly occurs if:

- No GOOSE control block is sent from IED 2 for more than 1000 ms
- A GOOSE control block is sent out from IED other than IED 2
- A GOOSE control block is sent to a multicast address other than 01-0C-CD-01-00-00

To test this, the following rule is generated and submitted to the Snort before it is activated:

```
Required 192.168.9.1 any → 01-0C-CD-01-00-00 any 1000 millisecond
```

This rule has Snort check that there is GOOSE traffic from a device with address of 192.168.9.1 to the MAC address 01-0C-CD-01-00-00. Note that the GOOSE message is a message that uses Layer 2 (Ethernet). Therefore, in operation, the MAC address of the sending device is needed. However, only the IP address of the entity is provided in the SCD file. The MAC address under the "IED 2: AA1D1Q10A1" is the multicast MAC address (the address to which a GOOSE message is sent). Thus, when the Snort IDS receives the rules, a command is sent using ARP to discover the MAC address of 192.168.9.1. Then, Snort extended with our preprocessor can detect any missing traffic according to the given rules. Using standard Snort features, a rule can be generated to alert when the entity with IP address 192.168.8.1 also sends out a GOOSE message. A test environment with IED's is setup in our lab, see Figure 6. Our test result shows that when the timing requirements of the GOOSE message are violated, Snort raises an alert.

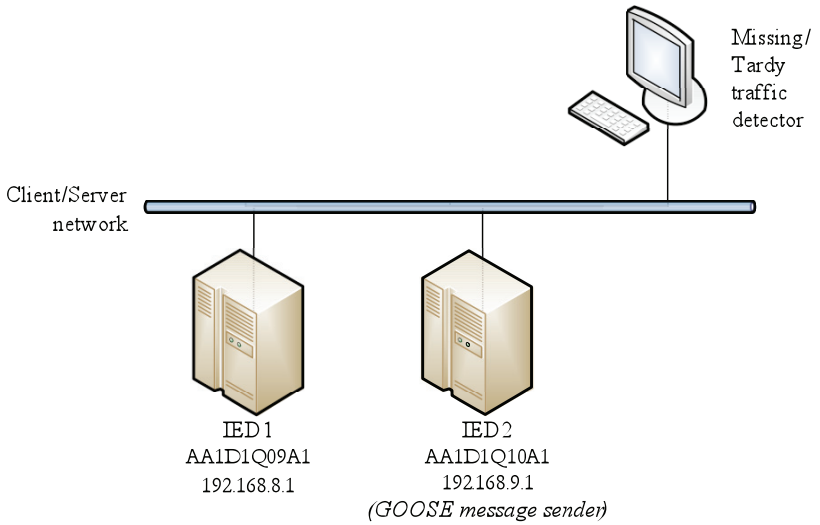


*Figure 6 – Laboratory Setup for Testing the Detection of Missing GOOSE Traffic*

### 4.2.2 Implementation for Process Automation Systems Application

An example of detecting missing traffic in the process automation application can be seen in the following simplified scenario. Assume that the system expert provides more implicit information regarding the installation of the process automation as depicted in Figure 2. According to the expert, the Connectivity Server is connected to a controller and regularly queries data from the controller every 20 sec. The data is sent to the controller using MMS protocol using TCP port 102. The controller has two network adapters with the following IP addresses: 172.16.80.101 and 172.17.80.101.

The setup of the system that can detect the missing required traffic is seen in Figure 7. In this example, each type of network (Client/Server network and Control network) has its own switch. The missing/delayed traffic detector is connected to the switch at the Control network.

The first step is to extract both the explicit and implicit information from the Planner file. The following information is extracted:

- There are three nodes in the system, namely a Operator Client, an Aspect Server and a Connectivity Server.
- The Operator Client connects to the network via 172.16.4.71 and 172.17.4.71.
- The Operator Client talks via the proprietary CSLib protocol.
- One Aspect Server. The server connects to the network via 172.16.4.21 and 172.17.4.21.
- The Aspect Server receives connection via TCP port 3338 and 3339.
- One Connectivity Server that has four network adapters. The IP addresses for the network adapters are 172.16.4.22, 172.17.4.22, 172.16.80.22, and 172.17.80.22 respectively.
- The Connectivity Server talks via TCP port 102 (MMS).
- The Connectivity Server is connected to the controller via 172.16.80.22 and 172.17.80.22. It talks with the Controller via MMS message (TCP port 102).
- The Controller connects to the Connectivity Server via 172.16.80.101 and 172.17.80.101
- The Connectivity server polls update from the Controller every 20 seconds.
- For redundancy purpose, RNRP traffic (TCP/UDP port 2423) is legitimate inside the system

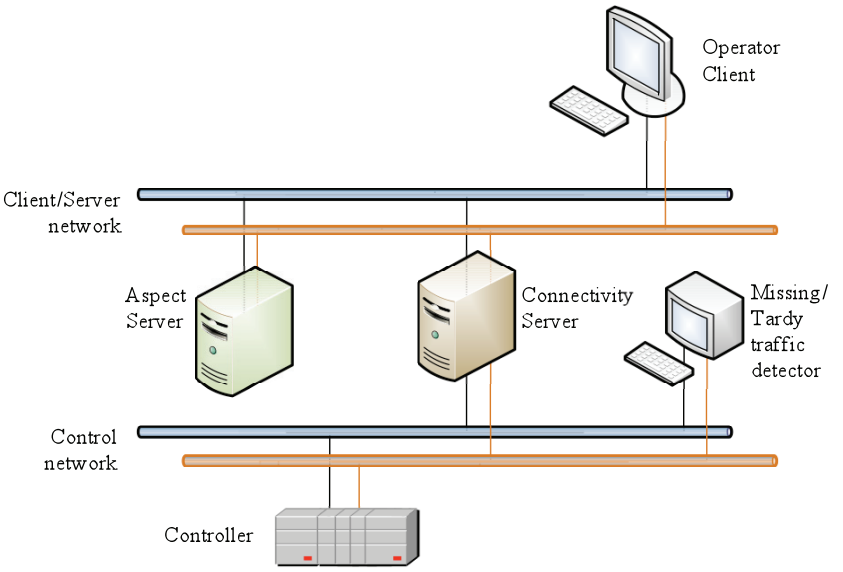


*Figure 7 – Illustration of the Simple Process Automation Example*

The abovementioned information is constructed into the comprehensive communication model. In this example, communication between the Connectivity Server and the Controller happens every 20 sec. When the communication is not observed every 20 sec, the detector raises the alert. For implementation, the same Snort implementation as described above is used. An example of the rule that is submitted to Snort is:

```
Required 172.16.80.101 tcp 102 → 172.16.80.22 any 20 second
```

From the control system perspective, any missing required messages will be noticed by the central control system. However the central control system may lack information on the cause of this anomaly especially when it is due to security reasons. If the proposed security systems are deployed at different spots in the system, it is possible to capture a bigger picture of the anomaly by correlating different captured security events. For instance, if a Connectivity Server receives input from the controller intermittently, this may be seen as a dropped packet from control system perspective. However, this could also be caused by an attacker disabling the controller and spoofing the messages. This activity will not be seen from the control room. However, by correlating events that are recorded at different points in the system, it is possible to detect that the controller is no longer sending required messages and that another node is spoofing these messages. This topic is an ongoing research.

### 4.2.3 Prototype to Generate Configuration File for Different Type of Security Measures

Another implementation of the translation framework is the ability to generate configuration file(s) for additional security measures. The next example shows how the framework generates rules for firewall devices to be used in the IACS. For this purpose, revisit the example on Figure 2.

To generate firewall rules, normally the following typical information is needed:

- Filtering rules - A tuple of origin IP address, origin port, destination port, destination IP address, allow or block action.
- Masquerading (for firewalls that perform network address translation) - A tuple of internal IP addresses, external IP addresses.
- Basic network information - DNS address, available IP addresses, VLAN, etc.

As elaborated in Section 3, the comprehensive communication model is described and stored as an eXtensible Markup Language (XML) file. In accordance to that design choice, the generation of configuration file for security measures is most reasonable to be done using The Extensible Stylesheet Language Family (XSL) Transformation – XSLT. To transform a certain XML file into another format using XSLT, it needs:

- one or more XML source documents
- one or more XSLT stylesheet modules

- the XSLT template processing engine (the processor), and
- one or more result documents.

The transformation happens when the XSLT processor takes two input documents, i.e. XML source document and the XSLT stylesheet and then produces an output which is a human-readable document.

In this prototype, for each type of security measure of which the configuration file is to be generated, a dedicated XSLT stylesheet needs to be developed. For instance, if a configuration files for a firewall is to be created, then a specific XSLT stylesheet for that particular firewall needs to be created. To test the prototype, the process automation example is chosen. From this example, a generic firewall rules are created. Necessary information is queried from both the explicit and implicit information from the Setup Package File, see Figure 2. The filtering rules can be derived from the implicit information of the Setup Package Files. For instance, the open ports for communication and type of traffics are source to generate the filtering rules. An example of the implicit information can be seen in Table 3. In our implementation, the implicit information is stored in a Microsoft Excel file.

| Legitimate IP Address | Legitimate TCP Port | Legitimate UDP Port |
|---|---|---|
| 172.16.4.21 | 3338, 3339, 2423 | 2423 |
| 172.17.4.21 | 3338, 3339, 2423 | 2423 |
| 172.16.4.22 | 102, 2423 | 2423 |
| 172.17.4.22 | 102, 2423 | 2423 |
| 172.16.80.22 | 102, 2423 | 2423 |
| 172.17.80.22 | 102, 2423 | 2423 |
| 172.16.80.101 | 102, 2423 | 2423 |
| 172.17.80.101 | 102, 2423 | 2423 |

*Table 3 – Examples of Legitimate IP Addresses and Ports for Communication*

As for basic network information, the Setup Package File provides sufficient information on available IP addresses. Figure 2 provides examples that summarize the relevant information. By taking the input from Figure 2 and available implicit information, the required firewall rules can be derived. The following shows examples of the rules.

```
ACCEPT any → 172.16.4.21 tcp 3338
ACCEPT any → 172.17.4.21 tcp 3338
ACCEPT any → 172.16.4.21 udp 2423
ACCEPT any → 172.17.4.21 udp 2423
ACCEPT any → 172.16.4.22 tcp 102
ACCEPT any → 172.17.4.22 tcp 102
ACCEPT any → 172.16.80.22 tcp 102
ACCEPT any → 172.17.80.22 tcp 102
…
DEFAULT DENY
```

Note that the default firewall rule must be denying all connections. The same logic in deriving firewall rules can also be used to generate configuration for other security measures, such as IPSec.

# 5 Conclusion

In this paper, the pre-engineered deterministic characteristics of an IACS are leveraged in order to derive a model of legitimate and illegitimate traffic in that IACS. The primary source of information for the communication pattern can be found in system description files that are used to configure IACS. The system description files contain information that can be useful in generating configuration files for security measures automatically. The necessary information for configuring the security measures is compared and the common information needed by the security measures is distilled.

A framework that extracts the information needed for the generation of a comprehensive communication model from a system description file is developed. Next, the communication model is used to generate different configuration files for security measures. In this paper, two implementations are shown. First, the focus is emphasized on a unique type of anomaly in an ICS which is when an expected traffic is missing or delayed. By using Snort and a custom preprocessor, the authors were able to detect the expected traffic that arrives at a wrong time or even detect missing traffic. Secondly, by taking the system description file as input the firewall rules are generated as an example for other types of common security measures which could be automatically configured this way.

---

**About the Authors** – Hadeli Hadeli works as a Scientist at ABB Corporate Research. He holds a PhD in Engineering and Master's degree in Mechanical Engineering from the Katholieke Universiteit Leuven – Catholic University of Leuven, Belgium and a Master's degree in Industrial Engineering & Engineering Management from Bandung Institute of Technology, Indonesia. His current research interest is in security for industrial control systems and multi-agent systems for coordination and control.

Ragnar Schierholz works as a Principal Scientist at ABB Corporate Research. He is the Cluster Lead for Security and Remote Operations in ABB's Industrial Software Systems research program. He is a member of various standardization bodies, such as the ISA 99 or the IEC TC 57 WG15 and a recognized member of the international industrial control system security community. He holds a PhD in Information Systems from the University of St.Gallen, a Master's degree in Computer Science from Western Michigan University and a Master's degree in Information Systems from the University of Paderborn. He is a member of the IEEE Computer Society and the ACM.

Markus Braendle is the Control System Security Manager for ABB's Power Systems division. He heads the Power Systems Security Council and is with his team responsible for the security of ABB products and systems within the Power Systems division. Markus is an active member of several security standardization efforts and working groups, e.g. within IEEE or Cigre, and a recognized member in the industrial control system security community. He holds a PhD and a Masters degree in Computer Science from the Swiss Federal Institute of Technology Zurich.

Cristian Tuduce is an independent consultant with special interest in the architectural aspects of network self-organization and security. His past research includes exploring the solution space of wireless ad hoc network monitoring and security for industrial control systems. He received a Ph.D. in computer science from the Swiss Federal Institute of Technology (ETH) in Zurich, Switzerland.

Sebastian Obermeier works as a Scientist at ABB Corporate Research. He holds a PhD and Diploma in Computer Science from the University of Paderborn, Germany. His past research areas include secure multiparty computation and the use of database technology within mobile ad-hoc networks.

## References

[1] D. Dzung, M. Naedele, T. von Hoff, and M. Crevatin, "Security for industrial communication systems," Proceedings of the IEEE, vol. 93, no. 6, pp. 1152–1177, June 2005.

[2] E. Byres and J. Lowe, "The myths and facts behind cyber security risks for industrial control systems," in Proc. of VDE Kongress, October 2004

[3] M. Naedele. "IT Security for Automation Systems", in: Industrial IT Handbook. CRC Press, 2005

[4] E. Byres, J. Carter, A. Elramly and D. Hoffman. Worlds in collision: Ethernet on the plant floor. In ISA Emerging Technologies Conference. Instrumentation Systems and Automation Society, Chicago, October 2002

[5] NERC. Reliability Standards series CIP 002-1 to 009-1 - Cyber Security, February 2006, available at http://www.nerc.com/page.php?cid=2|20

[6] ANSI/ISA 99.00.01 "Security for Industrial Automation and Control Systems Part 1: Terminology, Concepts, and Models" (and other parts of the same standard)

[7] IEC TS62351-1 "Power systems management and associated information exchange – Data and communications security Part 1: Communication network and system security – Introduction to security issues" (and other parts of the same standard)

[8] M. Naedele, "Standardizing Industrial IT Security A First Look at the IEC approach," in Proc. 10th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 05), September 2005.

[9] D. Holstein and J. Diaz, "Cyber security management for utility operations," in Proc. 39th Annual Hawaii Conference on System Science (HICSS-39), 2006.

[10] R. A. Kemmerer, G. Vigna, Intrusion Detection: A Brief History and Overview, Security and Privacy, No. 27, 2002

[11] Wool, Avishai, A Quantitative Study of Firewall Configuration Errors, IEEE Computer, 37(6):62-67, 2004

[12] IEC 61850 - Communication networks and systems in substations, International Electrotechnical Commission.

[13] IEC 61850-7-2 – Communication networks and systems in substations – Part 7-2: Basic communication structure for substation and feeder equipment – Abstract communication service interface (ACSI), International Electrotechnical Commission.

[14] ISO 9506 – Industrial automation systems - Manufacturing Message Specification (MMS), International Organization for Standardization. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=37079

[15] MODBUS Messaging on TCP/IP Implementation Guide, Modbus-IDA, 2006

[16] IEC 61850 – Communication networks and systems in substations, International Electrotechnical Commissions, http://www.iec.ch/

[17] IEC 61850-6 – Communication networks and systems in substations -- Configuration description language for communication in electrical substations related to IEDs, International Electrotechnical Commission, http://www.iec.ch/

[18] Wimmer, W. "IEC 61850 SCL - More than interoperable data exchange between engineering tools", PSCC 2005 (15th Power System Computation Conference), Liege/Belgium, 22.08.-26.08.2005

[19] IEC 61850-8-1 – Communication networks and systems in substations – Part 8-1: Specific Communication Service Mapping (SCSM) – Mappings to MMS (ISO 9506-1 and ISO 9506-2) and to ISO/IEC 8802-3, International Electrotechnical Commission.

[20] IndustrialIT 800xA – System System Version 5.0 SP2 – Automated Installation, ABB, 2008

[21] H. Hadeli, R. Schierholz, M. Braendle and C. Tuduce, Generating Configuration for Missing Traffic Detector and Security Measures in Industrial Control Systems Based on the System Description Files“, Proceedings of the IEEE Homeland Security Technologies 2009, Boston – USA

[22] Snort - the de facto standard for intrusion detection and prevention, http://www.snort.org, 2008