

Human-Assisted Intrusion Detection for Process Control Systems

Martin Naedele and Oliver Biderbost

ABB Corporate Research
CH-5405 Baden-Dättwil
Switzerland
martin.naedele@ch.abb.com

Abstract. With increased connectivity to public networks, information system security has started to become an issue also for automation and process control systems used to operate industrial manufacturing plants and critical infrastructure like power plants, or electricity/water/gas transmission grids. A solid security architecture for a networked system should contain not only defensive mechanisms like firewalls, but also detection and reaction capabilities. In today's automation system security systems the latter two tend to be neglected, as the existing mechanisms are not a good fit to the specific characteristics and requirements of such systems, and only few dedicated concepts and mechanisms exist. In this paper we present an approach to intrusion detection for process control systems that includes the human process operator, who is not an IT security expert, as a critical detection and decision element in the detection process, by utilizing his specific process control experience. The paper explains the specific security relevant characteristics of process control systems, shows how to translate security information to the process control user interface paradigm, and describes the architecture and functionality of our prototype implementation.

1 Introduction

1.1 Motivation

The influence of automation systems pervades many aspects of our everyday life. In the shape of factory and process control systems they enable high productivity in industrial production, and in the shape of electric power, gas, and water utility systems they provide the backbone of technical civilization.

Up to now, most of these systems were distributed, but isolated from each other and public networks like the Internet. During the last couple of years, however, due to market pressures and novel technology capabilities, it has become a trend to interconnect automation systems to achieve faster reaction times, to optimize decisions, and to collaborate between plants, enterprises and industry sectors. Initially, such interconnections were based on obscure, specialized, and proprietary communication means and protocols. Today, more and more open and standardized Internet technologies are used for that purpose. For these reasons, information system and network security is now also an issue for industrial automation systems.

1.2 Problem statement

The theory of time-based security [2] postulates that any information system security architecture must include both defensive elements (e.g. firewalls) as well as detective (e.g. intrusion detection systems, IDS) and reactive elements.

The deployment and use of IDSs, however, encounters a number of practical problems. The foremost problem (see e.g. the papers in [3]) is the high number of false positives (false alarms) created by today's rule-based as well as anomaly-based IDSs. These alarms have to be manually investigated by a human expert, and often these resources are not available to react in real-time on all IDS alerts. Also, IDS alert messages that require specialist knowledge to understand or to evaluate with respect to seriousness, prevent a first-level evaluation by a non-expert.

Process control systems have various characteristics pertinent to these issues that make them different from normal commercial/office IT systems:

The main concern and security objective is availability in order to retain (safe) control of the plant. Integrity violating attacks may also have very severe consequences, though they tend to be much more difficult to execute than availability attacks, as detailed knowledge of the specific plant, both with respect to the IT part and the physical process and equipment is required. As far as attacks on availability are concerned, both malicious and incidental (e.g. device failures) anomalies are relevant.

Connectivity of the process control system to outside networks including the company intranet is normally not mandatory for the automation system, and although extended periods of disconnection are inconvenient, they will not have catastrophic consequences - after all, many automation systems nowadays still run completely isolated.

Information transferred into the process control network can be restricted to certain types of content, e.g. only static web pages, but no scripts or other executable code.

The topology and configuration, both of HW and SW, of the automation system part, which contains the safety critical automation and control devices, is comparatively static. Therefore all involved devices and their normal, legitimate communication patterns (regarding communication partners, frequency, message size, message interaction patterns, etc) are known at configuration time, so that protection and detection mechanisms can be tailored to the system. Modifications of the communication system are rare enough to tolerate a certain additional engineering effort for reconfiguring the security settings. Static structure and behavioral patterns also make the process of anomaly discovery for intrusion detection easier and produce less false positives.

Process control systems are usually monitored continuously, around the clock, by dedicated staff. A defense architecture for process control systems should make use of this fact and incorporate process operators as valuable security elements, even though they do not have IT or even IT security expertise.

The involvement of a human in the anomaly detection process is powerful, because the operator is in his evaluation not, like IDS applications, restricted to single packets or a limited state history for his analysis, but can flexibly apply his experience of normal daily or weekly patterns and knowledge about specific unusual authorized activity in the plant, e.g. maintenance or re-engineering, without having to laboriously reconfigure and re-tune the rulebase of an automated IDS. The issue of a time-varying baseline and the consequences for the performance of automated anomaly-based intrusion detection surprisingly does not seem to get a lot of attention in current research, as seen from the papers in e.g. [3].

1.3 Contributions

This paper describes:

- The idea of reducing information system security status into a set of quantitative metrics that do not require knowledge of IS security terminology for understanding.
- A concept for visualizing these metrics and presenting them to an industrial process control operator within his normal work context in such a way as to enable him to detect unhealthy conditions on the network intuitively, using his experience with process monitoring and the human ability for visual pattern matching.
- The software architecture and operation of a prototypical implementation of a system integrating information system security information with an industrial process control system.

2 Previous Work

IS security for automation systems and even more so intrusion detection for automation systems is today still a field in which only little research is done. The only previous work on the combination of process control paradigms and IS security paradigms we are aware of is [4]. There the authors suggest to apply statistical process control (SPC) to a set of "indications and warnings" attributes they attach to each security relevant object in a computer system for anomaly detection. However, the application domain is not process control/automation systems, and no details about the implementation are given.

IT system management and IT security system management frameworks and applications, e.g. CA Unicenter or IBM Tivoli target an audience of IT and IT security specialists with qualitative output presenting relevant technical details. They do not abstract security information into real-time, quantitative scalar and trend information that would allow a non-expert to simply identify differences to a "normal" state of operation.

3 System functionality

3.1 Characteristics of the "attack space"

The restriction to only few, well-defined services and the predictability of usage and network traffic as stated in section 1.2 allows to considerably narrow down the types of attacks a process control system has to be defended against:

Most of the common attacks occurring on the Internet today performed by exploiting known or novel server vulnerabilities can in this case be neglected entirely, simply because the respective services that are attacked (e.g. SMTP, IRC, etc) do not run inside the process control network.

Attacks exploiting vulnerabilities of client applications inside the network accessing resources on the Internet are not an issue either, if the services accessed outside the network are known in advance and restricted by technical and/or administrative means to provide only static content.

Due to the clearly defined and small set of known and necessary traffic most attack attempts can be simply detected on the IP address/port level without having to go into content inspection.

In addition, statistical Intrusion Detection Systems can be used to scan for statistically unusual activity and thus potentially detect novel attacks, unknown to signature based systems. These systems also perform better, that is, with less false alarms, when deployed in environments with rather static activity patterns. Furthermore, the alarm firings may also be treated in a quantitative manner.

Due to the small circle of legitimate users and their deterministic login behavior, e.g. according to shift changes, unusual login events are with high probability security critical.

3.2 Relevant security status data sources

Quantitative data carrying meaningful information about the state of the IT part of the process control system and potential security events can be obtained from routers, switches, proxies, host and application logs and performance counters, as well as a statistical IDS. Also the output of security mechanisms that would normally produce detailed qualitative event information, such as rule-based IDSs or firewalls could be aggregated into quantitative, trendable data.

Relevant state variables include, but are not restricted to:

(Filtering) routers, firewalls, proxies, and reverse proxies: Incoming/outgoing traffic per unit time, passing and rejected, per protocol, per source address, per destination address, per rule; user login and (rule) management activities; authentication failures.

Network/switch: Total traffic, per protocol, per source address, per destination address, as percent of available bandwidth.

NIDS: Rule firings per unit time, per rule; dropped traffic per unit time; user login and (rule) management activities; authentication failures; time of most recent rule update.

Hosts: User login (number, account, privilege) and (rule) management activities; authentication, authorization failures; processor load; uptime; disk space; Security/audit events.

Applications (e.g. web servers): Authentication successes and failures; connection attempts per unit time; failed requests;

3.3 User interface

Since the human process operator has to be able to decide about the severity of the overall situation on the network without the knowledge of an IT security expert, data that is presented to him has to be restricted to simple, consistent principles. There is no use in presenting to him for example an alert of the IDS directly, with detailed information about the characteristics of the network traffic that caused the IDS to trigger the alarm. Everything has to be translated to indications and language he understands. The following principles on how to present event data to a process operator have been followed in the prototype implementation:

Explicit alarms and operator messages should be shown only for events with very low frequency. From his experience with the production process, an operator is used to the fact that an alarm message indicates that there is something seriously wrong with the process and

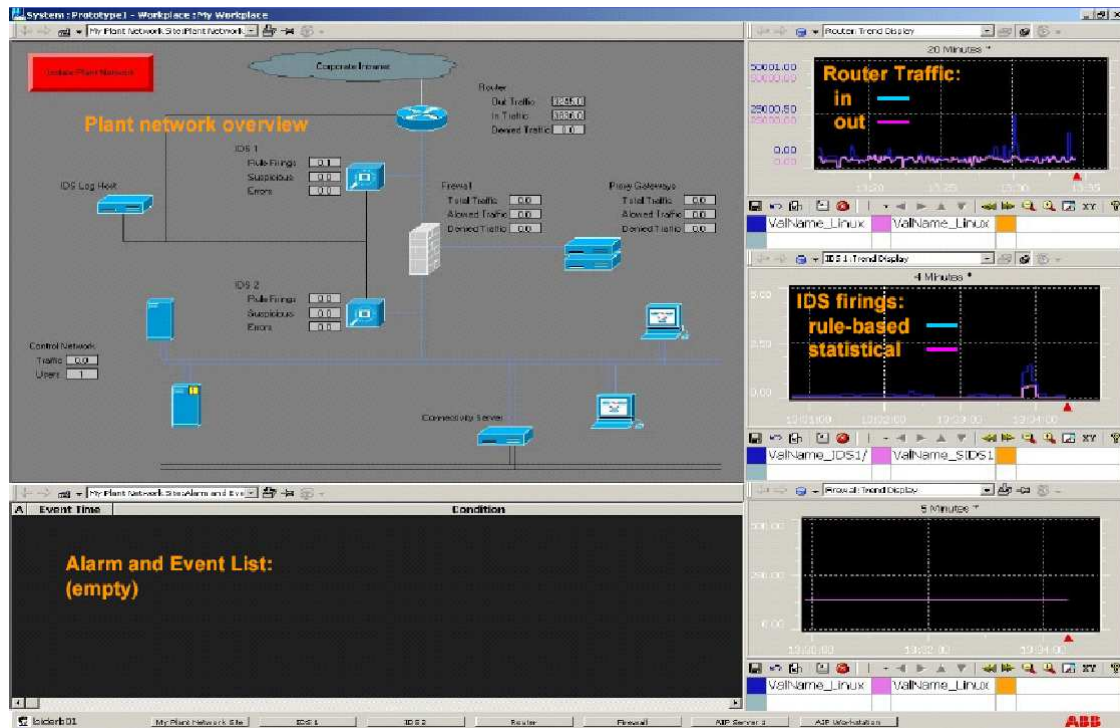


Fig. 1. Screen shot of the prototype process control system HMI for intrusion detection.

requires immediate action. If he now sees an alarm about events of uncertain relevance every other minute, he will simply start to ignore them.

Events with high frequency should not be shown to the operator as a textual message, neither should events, which have high false alarm rates. Such events should rather be treated quantitatively. They can be counted - perhaps weighted with their severity - and then shown to the operator as graphical trend charts that show the overall level of suspicious activity of a certain type.

Alarm messages shown to the operator should be very precise, not with respect to their cause in the form of a cryptic IDS alert message, but with respect to the consequence/danger for the system, e.g. "An unknown/unauthenticated user has successfully accessed a critical service on the local network".

The security relevant information is presented to the process operator in the form of a process picture, where in this case the "process" is the information system of the process control system (see Fig. 1). The process picture is annotated with numerical and color/icon-coded information about the current process/system state. Chart displays visualize historical trends in important parameter values over different time spans. The operator thus uses exactly the same SW environment and the same conceptual monitoring paradigms for the information system security process as for the production process.

Possible reactions of the operator to detected anomalies include: Isolation of network segments, alerting expert IT security staff, powering off of certain devices, switching to backup

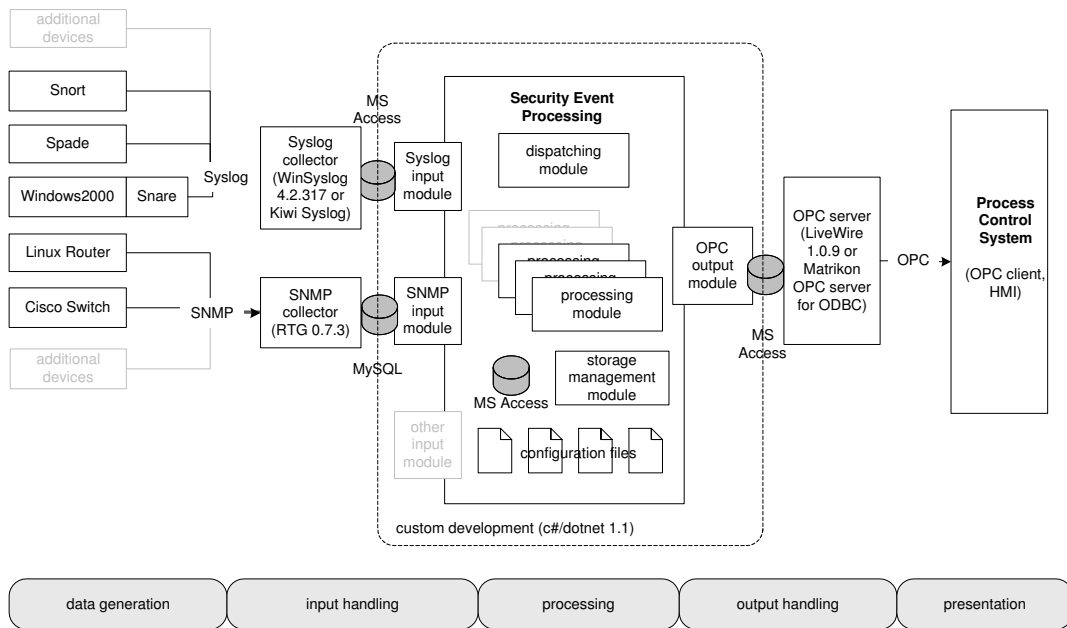


Fig. 2. Schematic system architecture overview.

systems, reinstallation of data and applications, start of an antivirus scanner - not running continuously in this environment due to concerns of non-deterministic performance decreases -, or even shutdown of the plant. These reactions should be predefined and preconfigured, and made available to the operator via buttons on the user interface.

4 System implementation

4.1 Architecture

The architecture of the prototype should allow a wide variety of input data sources, flexible handling of inputs and processing, and rely as much as possible on existing tools.

These requirements led to the five-tier architecture schematically presented in in Fig. 2:

Data generation Many different data sources are possible, as described in Section 3.2. In the prototype we used the traffic rates on a Linux host configured as router and a Cisco Catalyst 2926 switch, selected messages from the Windows 2000 event log converted to Syslog via Intersect Alliance SNARE, and the alert firing rate of a Snort IDS with the Spade statistical IDS.

Input handling Arbitrary protocols can be supported for pull and push data gathering, each via a combination of an off-the-shelf data gathering tool and a custom input module. The prototype supports Syslog and SNMP. The input modules normalize the incoming information for use in the processing modules.

Processing The processing modules (see Section 4.3) convert raw input data into (quantitative) information to be presented to the user.

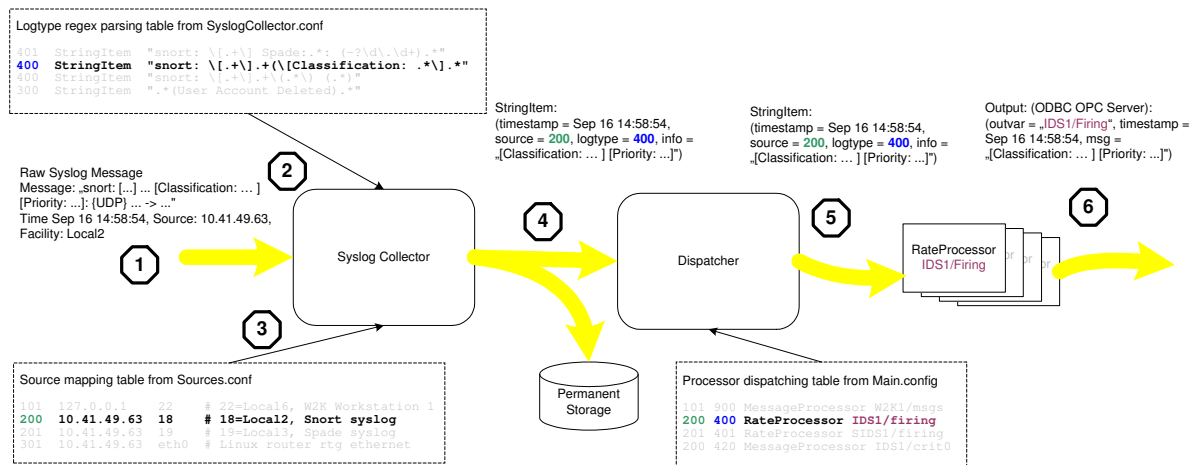


Fig. 3. Sample message processing cycle.

Output handling To bridge the protocol gap between IT systems and process control systems, the security status information is made available as OPC variables. OPC-DA [1], which is Microsoft (D)COM based, is the leading vendor neutral standard protocol for exchanging information within automation systems. The OPC output module could be replaced by a different output module e.g. for web based thin client presentation of the system state, or for using the new XML web service protocol OPC-XML-DA instead of OPC-DA via DCOM.

Presentation The actual and historical values of the security relevant OPC variables are presented on a process picture in the human machine interface of the process control system. The trend chart facility of the process control system allows to zoom in and out to visualize patterns on e.g. hourly, daily, weekly, or monthly basis. The presentation tier also includes input means to initiate countermeasures.

The input and processing modules are written in C# on Microsoft dotnet 1.1. The different tiers of the prototype are decoupled from each other via databases. While this may not be the most performant solution for a final product, it is a convenient and flexible mechanism for the prototype, in particular as the off-the-shelf data gathering tools like RTG for SNMP and WinSyslog for Syslog use databases by default.

4.2 Operation

Each processing cycle starts after a configurable waiting period. The processing cycle consists of polling all the inputs and processing the resulting messages. Fig. 3 shows in more detail how the custom part of the system processes incoming messages. It also demonstrates the extension points in the application where functionality can be added or changed simply by modifying the configuration files without having to recompile any part of the framework:

1. A data source, here for example the Snort NIDS, submits security state relevant information, e.g. about a rule firing, via the Syslog protocol to the Syslog server, which writes it

into an ODBC database. These two steps outside the custom processing system are not shown in the figure. The Syslog input module now polls the database and retrieves the entry.

2. In a first regular expression matching step the input module determines the logical type of the incoming message.
3. In a second matching step the Syslog facility info and the IP address of the source device are mapped to a logical source identifier.
4. The last processing step of the input module is to create a data carrying object, which it passes to two destinations: To the storage module, which maintains a revolving log of all received messages for in depth analysis in case of an incident handling procedure, and to the message processing dispatcher.
5. The dispatching module determines from the logical message type and source the processing module instance(s) the message should be sent to and which OPC variable should be modified. As the framework uses exclusively the abstract interfaces for access and reflection for object lifecycle management, new processing modules can be added to the system simply by making the dotnet class available and adding the necessary mapping lines in the `Main.Config` file. Each processing module is via special method calls informed about the start and end of each processing cycle. This allows the processing modules to not only process each single message that is passed in, but also to treat the messages of each processing cycle as a batch, e.g. to calculate rates (events per unit time) or to remove duplicates received within the processing cycle.
6. The processing module(s) instruct the OPC Server wrapper which OPC variable has to be set to which value as result of the processing.

4.3 Processing modules

The prototype system implements so far four types of processing modules:

Value The "value" module extracts a numerical value from the incoming message and sets the corresponding output (OPC) variable to this value. This processing module is used for data sources like processor load or host uptime.

Message The "message" module maps the incoming message to a text template and fills the variable parts in this template with information extracted from the incoming message. This processing module is used for input messages like system startup/shutdown (indicating the system concerned), account and policy management operations as well as login failures (indicating the account concerned). The resulting composed message string is passed to the output variable.

Increment/decrement The "increment/decrement" module maintains a numerical state based on state change messages. It reads, modifies, and writes the associated output variable. This processing module is used for input messages informing about user login/logout.

Rate The "rate" module calculates events per unit time from a batch of event notifications received in a configurable time period which may or may not be identical to the processing cycle time. This processing module is used in the prototype to calculate (S)IDS firing rates as well as router traffic rates from information about the accumulated amount of packets that have passed/been rejected by the router.

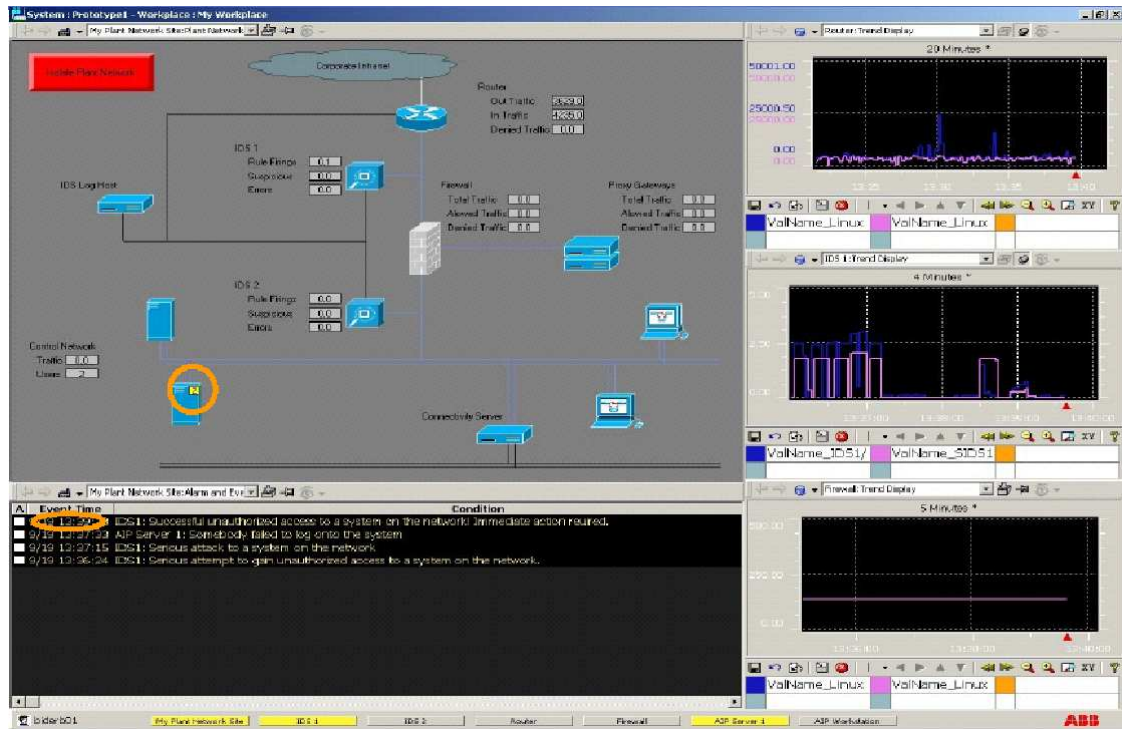


Fig. 4. Screen shot showing scanning activity and a change in the number of logged in users.

4.4 Limitations

The prototype implementation encountered a number of technical limitations, which could, however, easily be overcome in a product realization: The databases used, MS Access and MySQL, both suffered from performance and stability problems in certain situations. The processing modules could not directly create alarms in the alarm facility of the process control system, as the two OPC server demo versions from Matrikon and PlantLIVE did not support the OPC-AE (alarm & event) standard, so that a work-around using OPC-DA status flags polled by the HMI had to be used, which sometimes led to loss of alarms. Also, the regular expression parsing mechanism in the input modules may become a bottleneck once a larger number of different expressions have to be tested per message.

5 Results and further work

Our limited experiments with the approach and the prototype tool proposed and described in this paper so far have shown that various vulnerability scans that an attacker is likely to execute (e.g. using tools like nmap or nessus) produce distinctive anomaly patterns (see Fig. 4). Similarly clear patterns are expected for worms and other denial-of-service attacks. When our prototype was connected to the corporate intranet we could also observe traffic patterns changing over the day due to regular system management activities by the corporate IT group.

Next we plan to install and operate the system in a real factory setting to validate our hypothesis that quantitative presentation of the security status allows meaningful response by non-experts.

Based on the results of such a field trial a further step towards productization will be to select a subset of metrics which deliver the best information to noise ratio, and at the same time are applicable to a large number of process control system instances to reduce the customized engineering effort for each system installation.

Additional work needs to be done to create an ergonomic information security process picture.

6 Conclusions

A concept and prototypical implementation has been developed to collect and present information system security relevant parameters, events, and alarms, collected by a variety of devices in the automation network in the form of quantitative trends that a non-IT-expert process operator is used to from his main activity - monitoring the production process - and therefore feels comfortable interpreting. In addition, using the human mind as pattern matcher and relevance evaluator, overcomes some of the problems with false positives in conventional intrusion detection systems.

In our ongoing work we will conduct experiments with users to evaluate and improve usefulness and usability of our approach. Recent requests from customers that have shown that there is a strong need for an intrusion detection capability in process control systems that can not be satisfied by conventional IDSs, encourage us to pursue this approach further.

References

- [1] OPC Foundation. <http://www.opcfoundation.org/>.
- [2] Winn Schwartau. *Time based Security*. Interpact Press, 1999.
- [3] G. Vigna, E. Jonsson, and C. Kruegel, editors. *Recent Advances in Intrusion Detection (Proceedings 6th Int. Symposium RAID 2003)*. Number 2820 in LNCS. Springer, 2003.
- [4] N. Ye, J. Giordano, and J. Feldman. A process control approach to cyber attack detection. *Communications of the ACM*, 44(8):76–82, 2001.