DOI: http://dx.doi.org/10.14236/ewic/ics-csr2014.1

A SysML Extension for Security Analysis of Industrial Control Systems

Laurens Lemaire
KU Leuven
Department of Industrial Engineering
Gebroeders Desmetstraat 1, 9000 Ghent, Belgium
laurens.lemaire@cs.kuleuven.be

Jorn Lapon
KU Leuven
Department of Industrial Engineering
Gebroeders Desmetstraat 1, 9000 Ghent, Belgium
jorn.lapon@cs.kuleuven.be

Bart De Decker
KU Leuven
iMinds-DistriNet
Celestijnenlaan 200A, 3001 Heverlee, Belgium
bart.dedecker@cs.kuleuven.be

Vincent Naessens
KU Leuven
Department of Industrial Engineering
Gebroeders Desmetstraat 1, 9000 Ghent, Belgium
vincent.naessens@cs.kuleuven.be

The security of Industrial Control Systems (ICS) has become an important topic. Recent attacks have shown that inadequately protecting control systems could have disastrous consequences for society.

This paper presents an extension for the Systems Modeling Language (SysML), allowing for the extraction of vulnerabilities from an industrial control system model. After a control system is modeled in SysML, the model is converted into input for a formal reasoning tool. This tool contains a logic theory which is used for the vulnerability extraction. The rules in this logic theory are inferred from the ICS-CERT vulnerability database and ICS security standards. Once the vulnerabilities have been extracted, they are included in the SysML diagrams of the model.

The modeling approach allows the user to quickly see which changes to the system get rid of the reported vulnerabilities. It is also possible to mark certain components as compromised to see the consequences of attacks on these components for system security as a whole. The resulting analysis can be used to strengthen the security of the control system.

Keywords: Industrial control systems security, SysML

1. INTRODUCTION

Industrial Control Systems (ICS) are used for remotely controlling and monitoring critical infrastructures such as power stations, wastewater treatment facilities, or nuclear plants. These infrastructures are labelled critical because they provide society with essential resources like water and electricity. If these services stop functioning correctly, the consequences could be catastrophic: Large parts of cities could be left without electricity, raw sewage could spill into parks and rivers, or in the worst case nuclear meltdowns could occur with release of radioactive material as a result.

Security should be a major priority when designing and maintaining industrial control systems. Unfortunately this was not the case until the Stuxnet worm showed the powerful effects that attacks on these systems could have Matrosov et al. (2011); Falliere et al. (2011). By over-pressuring centrifuges and over-speeding their rotor blades, the Stuxnet worm was able to cause serious problems at the Natanz fuel enrichment plant in Iran Langner (2013). Other notable ICS incidents include the Maroochy Shire sewage spill in Australia Abrams and Weiss (2008), the Slammer worm disabling the David-Besse nuclear power plant in Ohio Poulsen (2003), and discovery of worms and Trojan backdoors like Duqu Chien et al. (2012) and Night Dragon Cyberattacks (2011) that gather information about control systems to make future attacks like Stuxnet possible.

Because of such incidents, research has been dedicated to the security of industrial control systems. Organisations such as NIST, ISA, and ISO are defining standards regarding security in these systems Stouffer et al. (2011); ANSI/ISA (2013); ISO/IEC (2008). Research groups have written numerous papers that review security issues in industrial control systems and list cyber attacks that they are vulnerable against Cheminod et al. (2013); Zhu et al. (2011); Cárdenas et al. (2008).

1.1. Related work

Several tools have been developed to assess security in control systems. Homeland Security has created CSET, the Cyber Security Evaluation Tool Homeland Security (2014). This tool checks compliance of a system with a chosen standard through a question and answer method. CSET does not provide an architectural analysis, and also does not allow the user to reason about compromised components and their effects on the system security, contrary to our method.

The KTH in Stockholm has developed CySeMoL, the Cyber Security Modeling Language Sommestad et al. (2013, 2010), a tool for estimating the probability that attacks succeed against an enterprise system. Similar to our method, CySeMoL allows users to change their system architecture and view the resulting changes on the attack probabilities. However, only the attacks that have been defined by the tool creators are considered. CySeMoL has to be updated when new attack methods are discovered. For their attack probabilities, CySeMol assumes that the attacker is a penetration tester who only has access to public tools and only tries to attack the system for one week. Previous ICS incidents like Stuxnet have shown that more powerful attackers must be considered.

ValueSec has redesigned Lancelot Pérez and Machnicki (2013), a tool previously used for evaluating the security of complex ICT systems in the financial sector. It is now a risk management platform that enables users to analyse security risks and their business implications for the smart grid and SCADA (Supervisory Control And Data Acquisition) environment. SCADA systems are a subset of ICS Stouffer et al. (2011). Lancelot allows a user to define the system's assets and to attribute risk profiles to them. It then performs security reviews to detect risks and compliance issues, and prepares mitigation plans to deal with the risks that are found. Risk in Lancelot is defined as "the potential damage that can be caused when something goes wrong with an asset or when someone/something takes advantage of an asset's vulnerabilities". It is assumed that the user of the program already has knowledge of the vulnerabilities in his system. Lancelot does not help with identifying vulnerabilities.

There are several other tools or methods to conduct risk assessment, but they also assume that the vulnerabilities are already known Francia III et al. (2012). Most risk assessment methods start with a meeting between the team performing the assessment and the system engineers den Braber et al. (2003). During this meeting they brainstorm about possible vulnerabilities that the system could have. Our tool automates this phase and the results could, hence, be used as input for risk assessment. When using our tool, a system engineer only has to enter the system architecture, assign the relevant security properties to components, and the vulnerabilities are extracted automatically.

1.2. Contribution

This work presents a formal methodology to detect vulnerabilities in industrial control systems. The control systems are modeled in the Systems Modeling Language (SysML), and then converted into input for the Inductive Definition Programming framework (IDP) Wittocx et al. (2008); Bogaerts et al. (2012). There, a logic-based theory using induction rules automatically infers vulnerabilities and corresponding attacks that could occur in the system. Both the modeling method and the IDP framework are explained in detail in further sections. Changing the security properties of some components allows users to reason about scenarios in which attackers have breached or compromised certain components. It is also possible to analyse the effects of system changes, newly discovered weaknesses, implemented countermeasures, etc.

This paper introduces the approach and describes the work that has been done so far. The modeling method is validated on a case study.

1.3. Outline

The structure of this paper is as follows. Section 2 contains some background on industrial control systems, explaining common architectures and detailing the differences between ICS and IT with respect to security needs. Section 3 introduces the Systems Modeling Language (SysML). In Section 4, the methodology of our approach is explained. Section 5 presents the IDP framework, which is used for the reasoning, and explains how control systems are modeled in this framework. In Section 6 we discuss the reasoning aspect of the IDP framework. Section 7 contains a validation on a case study, a brewery. Finally, Section 8 concludes the paper and contains future work.

2. OVERVIEW OF INDUSTRIAL CONTROL SYSTEMS

An industrial control system consists of a number of field devices that are being supervised from a centralised location Fovino et al. (2010); Galloway and Hancke (2013). The industrial network contains common network elements such as workstations, an e-mail server, databases, etc. In addition, it contains a Master Terminal Unit (MTU), and a Human Machine Interface (HMI) workstation. These can both be used to monitor and control the field devices. There is also a historian that is used to log time-based process data. The field devices contain several sensors and actuators that are being locally controlled by Programmable Logic Controllers (PLCs) or Intelligent Electronic Devices (IEDs).

In the past, these controllers and the communication protocols they use would be specifically tailored to the system. Control systems used to be isolated and proprietary, making it harder to mount attacks against them. Security was not a concern, other than preventing unauthorized access to the premises. Nowadays, the systems contain commercial, off-the-shelf components, and are often connected to the corporate network of the company that owns the system, as well as the internet ENISA (2011). These changes have made it easier to access and manage the system, but at the same time, they have also made it a lot easier for attackers to do harm.

Often a distinction is made between SCADA (Supervisory Control And Data Acquisition) systems and DCS (Distributed Control Systems) Stouffer et al. (2011). SCADA systems are geographically more spread out than DCS, and their field devices might be thousands of kilometres away from the MTU. They are mainly used in the energy, water, and transport sector. In DCS the network of controllers is less spread out. For example a production environment like a factory might use a DCS to control all operations. Our methodology can be used to model both types of ICS.

ICS differ from standard IT systems in architecture and security concerns. For IT systems the literature often speaks of the CIA triad - Confidentiality, Integrity, Availability, since those are the three most important security features Purdue (2004). Due to the critical nature of the supervised operations in ICS, there are other important dependability concerns such as safety and reliability. When something goes wrong in an ICS, there might be lives at stake. This is rarely the case in IT systems. It has lead to the use of the SRA (Safety, Reliability, Availability) triad in ICS Adesina (2012).

With different security requirements should come different security solutions. Unfortunately this has not always been the case. Quite often, attempts are made to fix problems in ICS with IT security countermeasures Larkin et al. (2012). Sometimes this works, but in other cases it has had the opposite effect. A technique often used in IT systems is automatic software updates, sometimes requiring a reboot. This has a negative effect on the network availability and reliability requirements of ICS systems and may therefore not be possible Tom et al. (2008).

It follows that new methods and tools for improving the security of industrial control systems must be considered.

3. THE SYSTEMS MODELING LANGUAGE (SYSML)

SysML is a modeling language, derived from UML, that is used in Model-Based Systems Engineering (MBSE). MBSE is defined as “the formalized application of modeling to support system requirements, design, analysis, verification, and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases.” Crisp (2007). The advantage of MBSE over traditional Document-Based Systems Engineering is that the former produces a system model contained in a model repository, while the latter produces a variety of documents, making it more difficult to perform traceability and assess change impacts.

UML is mostly used in software engineering. SysML reuses part of UML and adds diagrams for modeling systems and systems-of-systems. It is developed and maintained by the Object Management Group (OMG). SysML includes nine diagrams which are shown in Figure 1. Four diagrams concern the system behaviour, four concern the system structure, and the final one is a requirements diagram. SysML can not only be used to model software, but also hardware, information, processes, personnel, and facilities. A description of each diagram is out of the scope of this paper.

ICS operators are increasingly often using SysML to model control systems. For instance, General Electric Transportation Systems (GETS) uses it for production of their railway signaling applications Ferrari et al. (2011).

SysML is a language, various tools support it and allow a user to model with SysML. For this work, an open source SysML environment is required to create the extension that converts SysML models into suitable input for the reasoning tool. Papyrus has been chosen for this purpose. It is an Eclipse tool that provides complete support for SysML.

3.1. Security in SysML

Currently none of the diagrams in SysML provide a way to consider system security. A paper by Oates et al. (2013) talks about the lack of system security in Model-Based System Engineering, and in

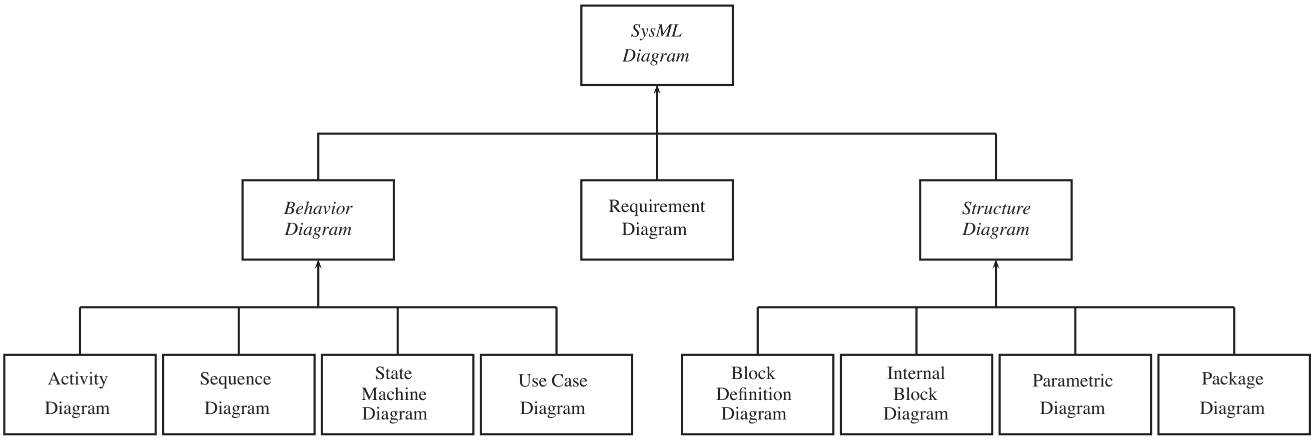

Figure 1: *The nine diagrams included in SysML.*

particular in SysML. In that paper they say that little work is done from the perspective of automatically pulling relevant information from an existing model to highlight security vulnerabilities. That is what this work aims to achieve. At the end of their paper they suggest a threat agent diagram, based on the threat-risk relationship from ISO/IEC (2008). When our method finds any vulnerabilities, they can be included in SysML in a similar diagram.

4. METHODOLOGY

The aim of this research is to create a SysML extension which allows a user to identify vulnerabilities in an industrial control system based on the system model. It is also possible for the user to reason about the effects of newly discovered weaknesses, successful attacks, and system changes on the system security. The formal reasoning part of the extension contains a collection of logic rules that are used in the vulnerability extraction. The user input is the system architecture. Figure 2 presents an overview of the methodology. The components of this figure will be explained in detail in further sections.

An industrial control system is modeled in the Systems Modeling Language (SysML). Then the model is converted into input for our formal reasoning framework, we use the Inductive Definition Programming framework (IDP) for this purpose. This framework contains a logic theory that infers vulnerabilities from the system architecture. Some of these rules are taken from the ICS-CERT vulnerability database, others from standards and guidelines such as the ones from ISA, DHS and NIST Stouffer et al. (2011); ANSI/ISA (2013); Security (2011). The result is a collection of system vulnerabilities, these will then be included in the SysML model.

This approach can be useful for users who are designing a new control system, as well as for operators who want to check the security of an existing one.

5. INDUCTIVE DEFINITION PROGRAMMING FRAMEWORK

The Inductive Definition Programming framework (IDP) is an extension of first order logic. It adds additional functionality such as aggregates, partial functions, inductive definitions, etc. The framework is used to solve search problems using model expansion. It takes as input a partial model and a list of constraints in the form of logic rules, and returns one or more complete models that satisfy these constraints Wittocx et al. (2008); Bogaerts et al. (2012).

The input for an IDP instance consists of a **vocabulary**, a **theory** and a **structure**. The theory contains all the logic rules and inductive definitions that will be used to draw conclusions. In the structure part, the system under consideration is modeled. In the vocabulary, the types, predicates, and functions that will be used in the theory and the structure are specified.

In this problem domain, namely ICS, the same vocabulary and theory can be reused to model all systems. The structure differs from system to system, as it contains the components of the specific ICS that is being modeled. Below, an overview is given of what an ICS model looks like in the IDP framework.

In the vocabulary, supported types include:

– type $SystemPart$

– type $Component$ isa $SystemPart$

– type $CommChannel$ isa $SystemPart$

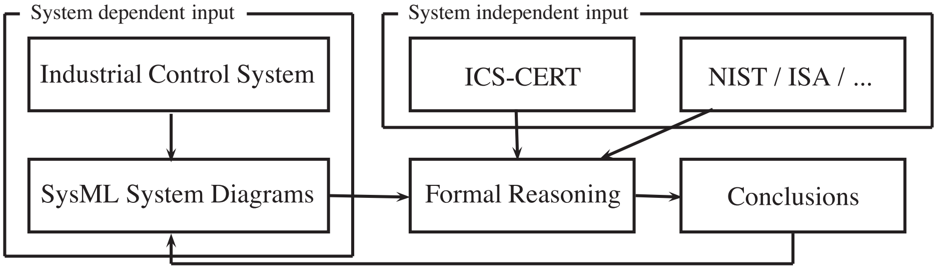

Figure 2: *An overview of the methodology.*

It is specified that $Component$ and $CommChannel$ are subtypes of $SystemPart$. These types are currently "empty" and will be initialised in the structure part with the components and communication channels that are present in the ICS under consideration.

Types $Property$ and $Vulnerability$ are constructed from a set of constants in the vocabulary using the $constructed\ from$ command. This allows a user to consider the same security properties and vulnerabilities in all ICS, without having to list them in the structure every time.

– type $Property$ constructed from $\{Integrity, Authentication, \ldots\}$

– type $Vulnerability$ constructed from $\{Spoof, UnsafeMediaAccess, \ldots\}$

Predicates are also defined in the vocabulary. Here are two examples:

– $HasProperty(SystemPart,\ Property)$

– $HasVulnerability(SystemPart,$ $Vulnerability)$

The first predicate will allow users to associate security properties with system parts. Pairs of properties and system parts will be enumerated by the user in the structure. The second predicate does not get initialised by the user. The tool will use the rules specified in its theory to deduce vulnerabilities in system parts, and these will be represented by this predicate. When the control system has been modeled and the vulnerability extraction is done, a list of pairs of the $HasVulnerability$ predicate will be returned, and can be consulted to find the vulnerabilities in the system. When a user wants to reason about the effects of compromised components, he can also define vulnerabilities in the structure using this predicate.

In the theory, logic rules are specified that allow the tool to find vulnerabilities based on security properties. Examples of rules are given below:

– $\forall x[CommChannel]$: $HasVulnerability(x, Spoof) \leftarrow$ $\neg HasProperty(x, Authentication)$

– $\forall x[SystemPart]$: $HasVulnerability(x, UnsafeMediaAccess) \leftarrow$ $HasProperty(x, NoMediaPolicy)$

The first rule states that all communication channels that do not have the authentication property are vulnerable to spoofing attacks. Without authentication, an attacker is able to send malicious commands to parts of the control system. This can have severe consequences, as he may be able to shut down the system.

The second rule concerns media access of components. If there are no policies regarding the usage of USB devices on a component, then there is a risk of malware being uploaded. This could be done on purpose by malicious individuals that have access to the component, or employees could accidentally bring in compromised media.

In the structure part, all the components and communication channels of the specific system being modeled are listed and are given the appropriate security properties by instantiating the $HasProperty$ predicate where applicable. Some examples of components and properties are given below.

– $Component = \{$HMI, MTU, Historian, $\ldots\}$

– $HasProperty = \{$(HMI, NoMediaPolicy), (MTU, Authentication), $\ldots\}$

The first line specifies the components in the system being modeled. The second line binds properties to these components. The first property states that there is no policy regarding media access to the HMI. The second property says that there is some form of authentication required to access the MTU. Additional predicates can specify the authentication factor and authentication methods (password, token, biometrics, ...) to allow for better rules regarding authentication vulnerabilities.

Once the system has been represented using the above types, predicates, and functions, a partial model is made and the logic rules act as constraints. Under these constraints, the full model is created by IDP's model expansion. When the model expansion is finished, we are interested in the $HasVulnerability$ predicate which will list the vulnerabilities for each component and communication channel. After a first run of the program, the security properties of components can be altered or vulnerabilities can be added to model the effects of successful attacks or changes to the system. The framework for our model is shown in Figure 3.

6. SYSTEM INDEPENDENT REASONING INPUT

The IDP logic theory which will draw the vulnerabilities from the model input is largely based on two system independent resources: The ICS-CERT vulnerability database and standards regarding ICS security such as the ones from NIST, ISA, ISO, and others. Rules from these resources will allow us to identify vulnerable components. Then, another set of rules assesses the impact of these vulnerable components on system security as a whole.

The ICS-CERT vulnerability database is managed by the Department of Homeland Security. It contains a collection of all the known vulnerabilities in hardware or software related to Industrial Control Systems, including PLCs, HMIs, historians, industrial operating systems, and so on. They have two collections on their site: ICS alerts and ICS advisories. Alerts inform the reader about newly discovered vulnerabilities, advisories contain fixes and mitigations for existing vulnerabilities. Both collections are fully included in our IDP logic theory.

As an example, consider $ICS - ALERT - 14 - 015 - 01$ which reports a buffer overflow in the Ecava IntegraXor HMI software ICS-CERT (2014). There are predicates in IDP that allow a modeler to specify the type and version of certain components. For instance the predicate:

$PLC(Component, Product, Version)$

For each PLC component in the system or system-of-systems, the user can specify the vendor and version information by using this predicate. For instance there could be a component "plc1" which is a Siemens Simatic S7-1200 PLC. Then the following tuple would be added to the list of PLC predicates in the structure:

$(plc1, SiemensSimaticS7, 1200)$

Similar predicates exist for industrial operating systems, HMIs, historians, third party software, etc. Hence, based on the above alert, the following rule is added to the logic theory:

$//$ ICS-ALERT-14-015-01
$\forall x[Component] : HasVulnerability(x, BufferOverflow) \leftarrow HMI(x, EcavaIntegraXor, 4.14380).$

It says that, for all components x, if x is an HMI of type Ecava IntegraXor and version 4.14380, component x contains a buffer overflow vulnerability.

Other rules come from the various standards and guidelines that have been consulted, including the NIST standard on Industrial Control System security Stouffer et al. (2011), the ISA/IEC-62443 standard ANSI/ISA (2013), an extensive document on ICS security from Homeland Security Security (2011), other standards and guidelines ENISA (2011), as well as various papers Cheminod et al. (2013); Zhu et al. (2011); Cárdenas et al. (2008). From these resources, rules can be inferred regarding:

- **Authentication.** There should be multi-factor authentication. Using tokens and biometrics is encouraged for critical components as an operator might forget a complicated password in case of emergency.
- **Passwords.** Where passwords are used they should follow certain guidelines regarding password length and types of characters employed to ensure strong passwords. Default passwords should be changed on all devices.
- **Media Access.** USB ports should be blocked wherever they aren't necessary. When USB access is required, USB devices should be controlled prior to usage. Otherwise, employees could accidentally upload viruses or malicious individuals could do this on purpose.
- **Physical Access.** Physical access to sensors and actuators should be prevented if this can negatively affect the process. If this is difficult to achieve, camera surveillance should be considered.
- **Hardening.** Hardening means reducing the vulnerability surface by removing unnecessary or unused services, software, and usernames.
- **Network Architecture.** It is good practice to put historians and databases in a DMZ. That way there is no direct connection required from the corporate network to the control network, they only have to be able to access the DMZ.

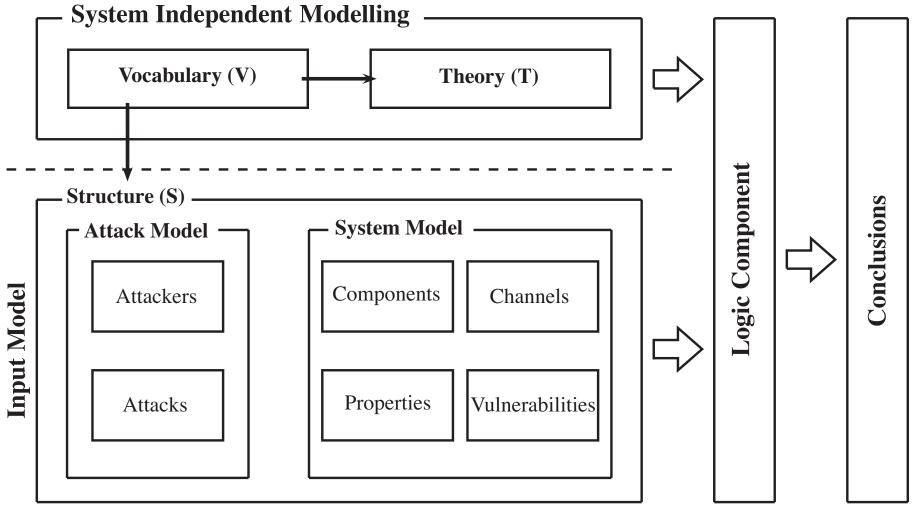

Figure 3: *Framework for ICS Modeling in IDP.*

The above rules allow the system to extract vulnerabilities on a component level. Additional rules are in place to deduce the impact of these component vulnerabilities on the system. For instance if a malicious user can exploit a buffer overflow to cause a denial of service of a component, the framework would report what other components will be affected by this. These rules have not been fully implemented yet.

7. VALIDATION

The parts of the methodology that are already implemented have been validated on a real case study. There are several reasons for this. In order to get familiar with modeling in SysML, it helps to model a small system. It is also an opportunity to test Papyrus and see if it is suitable for our purposes. The system is also modeled directly in IDP, this gives us an idea of what predicates will be necessary to model these systems and it will make the conversion from SysML to IDP easier. The reasoning can be tested on a component level, we can see if IDP behaves as expected when vulnerabilities are introduced or system parts are changed.

The chemical department at the campus of KAHO Sint-Lieven owns a brewery. This brewery is a small industrial control system. It contains four cauldrons where the brewing process takes place. A pump ensures that the brew flows from cauldron to cauldron. There is an electrical enclosure with a Programmable Logic Controller inside. A supervision computer is installed to monitor and control the process. Currently all supervision has to be done locally inside the brewery, but there are plans to make remote controlling and monitoring possible in the future.

To start, we modeled the full system in SysML using Papyrus. Showing the full result is not feasible as it consists of many diagrams. One of the Internal Block Diagrams is shown in Figure 4. Internal Block Diagrams are used to describe the internal structure of blocks. They show how parts and ports are connected. This particular IBD shows the internal working of the supervision component of the brewery. The brewery has a PLC in an electrical enclosure, theres an HMI with a touchscreen from where a user can change parameters or send commands to the process. This is done over a profibus link. A second IBD called “Process” is connected to the Process block in the Supervision IBD, and the details of the process containing the cauldrons and the pump go in there.

Modeling in SysML and Papyrus is intuitive and there is ample documentation available online to help the modeler out.

We then modeled the system directly in IDP. All components and communication channels were listed in the structure and the relevant properties were bound to them. Some additional predicates were defined to capture all the security-related properties. For instance, the supervision computer requires authentication in the form of a password. Authentication methods and the authentication factor can be tied to components that require authentication as follows:

- $AuthenticationMethod(Component, Method)$
- $AuthenticationFactor(Component, Factor)$

A list of authentication methods is provided in the Vocabulary, including $Password$, $Token$, $Biometric$, etc. Type $Factor$ contains integers.

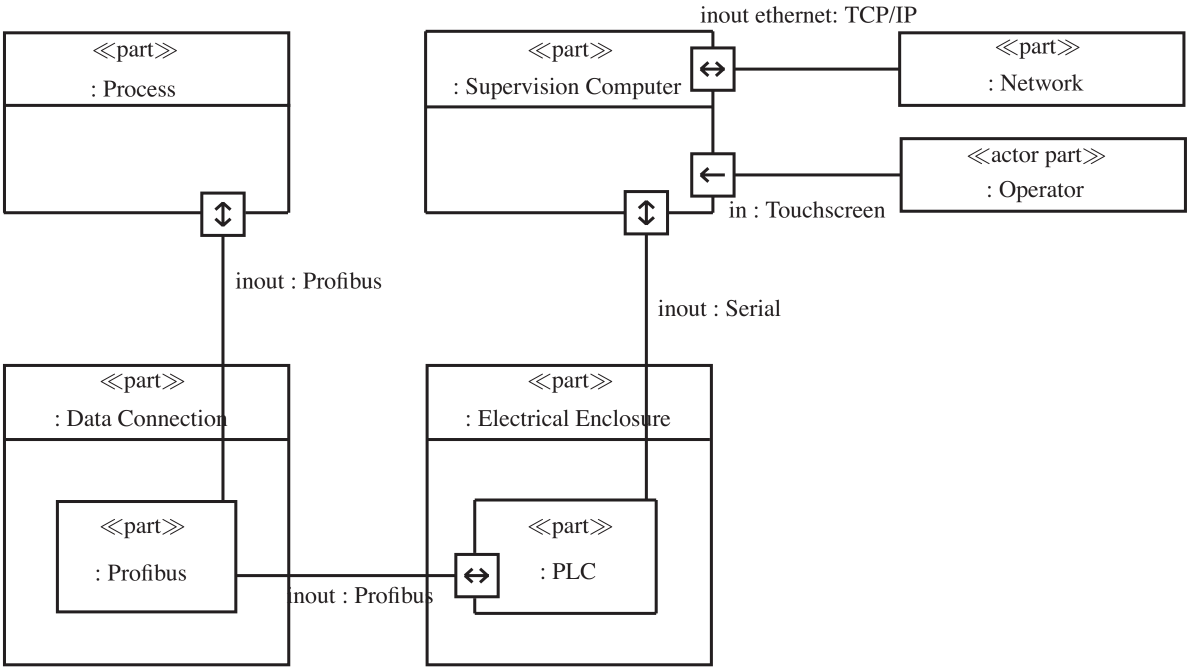

Figure 4: *A SysML Internal Block Diagram in Papyrus.*

After the system was modeled, the reasoning rules extracted the component vulnerabilities, some of which are listed below:

- $HasVulnerability(PLC, PhysicalAccess)$
- $HasVulnerability(HMI, LowAuthenticationFactor)$
- $HasVulnerability(HMI, UnsafeMediaAccess)$

The brewery can be accessed by anyone, so there are several physical access concerns. The PLC is inside an electrical enclosure but this enclosure isn't locked. The supervision PC requires only a password to gain access, this is not very secure. There is also a USB port on which any media device can be used. None of the components of the system are in the ICS-CERT vulnerability database.

Once the first set of vulnerabilities appeared, changes were made to the model in IDP to see which changes to the system would make the vulnerabilities disappear. This all works as intended, and suggestions have been made to the brewery operator to improve the security of his system.

8. CONCLUSION

A SysML extension for security analysis of Industrial Control Systems is being developed. The control systems are modeled in SysML, using Eclipse Papyrus, and then converted to IDP input, where the analysis is done. To do this analysis we have a logic theory containing rules derived from ICS-CERT and various ICS security standards. The modeling approach and the reasoning aspect have been validated on a brewery case study.

8.1. Future work

Modeling the system is implemented and tested both in SysML and IDP. The vulnerability extraction rules are also in place. What remains to do is to create the conversion from SysML to IDP input, as well as creating the diagrams that will show the vulnerabilities in the SysML model. The logic rules that infer the system vulnerabilities from the component ones are also not fully implemented yet.

When complete, the extension will be tested on a case study. Talks are ongoing with the Flemish Environment Agency to use one of their water installations and their supervisory centre as a full-scale case study.

REFERENCES

Abrams, M. and Weiss, J. (2008) Malicious control system cyber security attack case study– Maroochy Water Services, Australia.

Adesina, T. (2012) The state of industrial control systems security and national critical infrastructure protection: Emerging threats.

ANSI/ISA-62443-3-3 (99.03.03) (2013) *Security for industrial automation and control systems Part 3-3: System security requirements and security levels.*

Bogaerts, B. et al. (2012) The IDP framework reference manual.

Cárdenas, A. A., Amin, S., and Sastry, S. (2008) Research challenges for the security of control systems. In: *HotSec*.

Cheminod, M., Durante, L., and Valenzano, A. (2013) Review of security issues in industrial networks. *IEEE Trans. Ind. Informat.*, 9 (1). 277–293.

Chien, E., OMurchu, L., and Falliere, N. (2012) W32.Duqu: The precursor to the next stuxnet. In: *Proc. of the 5th USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*.

Crisp, H. (2007) Systems engineering vision 2020.

Cyberattacks, G. E. (2011) Night dragon.

den Braber, F. et al. (2003) The CORAS methodology: Model-based risk assessment using UML and UP.

ENISA (2011) Protecting industrial control systems: Recommendations for Europe and member states.

Falliere, N., Murchu, L., and Chien, E. (2011) W32.Stuxnet Dossier.

Ferrari, A. et al. (2011) Adoption of model-based testing and abstract interpretation by a railway signalling manufacturer.

Fovino, I., Coletta, A., and Masera , M. (2010) Taxonomy of security solutions for the SCADA sector, deliverable: D 2.2, version: 1.1.

Francia, III, G. A., Thornton, D., and Dawson, J. (2012) Security best practices and risk assessment of SCADA and industrial control systems.

Galloway, B. and Hancke, G. P., (2013) Introduction to industrial control networks. *IEEE Commun. Surveys Tuts.*, 15 (2), 860–880.

Homeland Security - HCC (2014) CSET: Cyber security evaluation tool.

ICS-CERT (2014) Ecava IntegraXor buffer overflow vulnerability.

ISO/IEC 21827 (2008) *Information technology – security techniques – systems security engineering – capability maturity model (SSE-CMM)*. Geneva, Switzerland.

Langner, R. (2013) To kill a centrifuge: A technical analysis of what Stuxnet's creators tried to achieve.

Larkin, R., Lopez, J., and Butts, J. (2012) Evaluation of traditional security solutions in the SCADA environment. In: *Proceedings of the 7th International Conference on Information Warfare and Security*. 399.

Matrosov, A. et al. (2011) Stuxnet under the microscope.

Oates, R., Thom, F., and Herries, G. (2013) Security-aware, model-based systems engineering with SysML. In: *Proceedings of the 1st International Symposium for ICS & SCADA Cyber Security Research*. 78.

Pérez, J. M. and Machnicki, D. (2013) VALUE-SEC D5.3–Description of developed tools and data.

Poulsen, K. (2003) Slammer worm crashed Ohio nuke plant network.

Purdue (2004) RASC: Confidentiality, integrity and availability (CIA).

Security, H. (2011) Common cyber security vulnerabilities in industrial control systems.

Sommestad, T., Ekstedt, M., and Holm, H. (2013) The cyber security modeling language: A tool for assessing the vulnerability of enterprise system architectures. *IEEE Syst. J.*, 7 (3), 363–373.

Sommestad, T., Ekstedt, M., and Nordström, L. (2010) A case study applying the cyber security modeling language.

Stouffer, K., Falco, J., and Scarfone, K. (2011) Guide to industrial control systems (ICS) security.

Tom, S., Christiansen, D., and Berrett, D. (2008) Recommended practice for patch management of control systems.

Wittocx, J., Mariën, M., and Denecker, M. (2008) The IDP system: A model expansion system for an extension of classical logic. In: *Proceedings of the 2nd Workshop on Logic and Search*. 153–165.

Zhu, B., Joseph, A., and Sastry, S. (2011) A taxonomy of cyber attacks on SCADA systems. In: *Internet of Things (iThings/CPSCom), 2011 International Conference on and 4th International Conference on Cyber, Physical and Social Computing*. 380–388.