

Security-Aware, Model-Based Systems Engineering with SysML

Robert Oates
Rolls-Royce plc.
Software Centre of Excellence
PO Box 31
Derby DE24 8BJ
robert.oates@rolls-royce.com

Fran Thom
Rolls-Royce plc.
Software Centre of Excellence
PO Box 31
Derby DE24 8BJ
francis.thom@rolls-royce.com

Graham Herries
Rolls-Royce plc.
Software Centre of Excellence
PO Box 31
Derby DE24 8BJ
graham.herries@rolls-royce.com

Addressing the cyber security demands of industrial control systems is a challenging task as the systems are typically built from equipment with limited computational power and integrated into highly complex, infrastructure-critical systems. To ensure the security of such systems, security considerations must form a ubiquitous part of the design and maintenance processes. This work provides an overview of the literature regarding the key issues faced by engineers attempting to secure industrial control systems, appraises work done to integrate cyber security into the systems engineering process and puts forward recommendations for the future of security-aware systems engineering through the extension of SysML to incorporate a Security viewpoint on the model. These recommendations include the presentation of a novel threat model profile that forms the basis of the SysML extensions.

Industrial control systems, SysML, Systems Engineering

1. INTRODUCTION

The vulnerability of industrial control systems (ICSs) to cyber-attacks is now a widely recognised problem which poses a significant threat to both economic and physical infrastructures. In recognition of this threat, dedicated task forces are being created by private organisations and governments alike (DHL 2011), (ENISA 2011) and (CPNI 2011). Cyber security for ICSs is a challenging problem as ICSs present a number of unique obstacles to the field of cyber security (DHL 2009). In addition, much cyber security expertise comes from the field of computer science, and many practitioners have a limited knowledge of the unique demands of ICSs.

This paper presents work performed on linking cyber security techniques to model-based systems engineering (MBSE) and SysML. MBSE is an approach which connects to every stage of the engineering lifecycle, created to address the need to handle the huge complexity faced by modern engineers and to formalise the design and test of electro-mechanical systems with computational power (INCOSE-TP-2004-004-02). SysML is a modelling language based on UML2.0 from the Object Management Group, designed to facilitate MBSE (OMG 2012).

In Section 2 work will be presented from the spheres of cyber security, model-based systems engineering and security-aware systems engineering. In Section 3 the potential advantages of using SysML to design secure ICSs are explored and a novel extension to SysML in the form of a security view is presented. Section 4 contains conclusions and speculates on the future directions of the field.

2. RELATED WORK

2.1. Cyber Security for Industrial Control Systems

Until recently, the fields of cyber security and industrial control systems have developed separately from one another. Cyber security experts tend to be from the field of computer science and the designers and implementers of ICSs tend to be control engineers. This lack of commonality between the two groups is identified as a key problem in securing ICSs (GAO-04-140T). However the fields have been brought together by the increased threat of cyber-attacks on ICSs. In (GAO-04-140T) five issues attributed with being the primary causes for increases in the threat of ICS attacks are presented: 1) The increased use of Commercial Off The Shelf (COTS) technology with

pre-existing vulnerabilities. 2) The increased connectivity of the devices used to form ICSs. 3) Incompatibilities between “traditional” cyber security techniques and the equipment used to construct ICSs. 4) Lack of security for the external network connections to ICSs. 5) Securing information about the structure of ICSs is often considered a low priority, making it easier to identify vulnerabilities within those systems. Despite the increased likelihood of attack, many organisations have no ICS security strategy and consider such systems as beyond the scope of their IT security policies, resulting in ICSs being connected to externalised networks with no auditing, no usage policy and no risk assessment (NAERC 2007). Where ICS-specific security policies do exist, the deployment of standard cyber security protection, beyond basic authentication and firewalling, is extremely challenging as most cryptographic techniques require a significant amount of computational power (DHS 2009). The Department of Homeland Security’s “Recommended Practice” for ICSs identifies a series of technical barriers to the adoption of standard cyber security techniques by the creators of ICSs (DHS 2009) but political barriers are also prevalent, generally stemming from a perceived lack of economic justification for modifying systems which are functioning adequately (GAO-04-140T 2003).

Deploying firewalls to the interfaces of ICSs is an effective way to protect against many threats (DHS 2009), but by protecting the borders of systems alone, there is an inherent brittleness which leaves systems vulnerable to insider or geographically proximal attacks. A more effective strategy is to complement the standard firewalling and authentication techniques with processes that continuously monitor the security landscape surrounding the systems under an organisation’s control and the adoption of ubiquitous security through design. The implementation of on-going processes to monitor threat landscapes is a relatively trivial task, made simpler by the existence of automated vulnerability search tools such as SHODAN (Matherly 2009) and online archives of controller vulnerabilities such as The Department of Homeland Security’s ICS-CERT database (ICS-CERT 2013). However, designing systems to be inherently secure, is not only technically difficult, in terms of the computational resources typically available on ICSs, but also requires a fundamental shift in the way the software for ICSs is designed.

2.2. Model-Based Systems Engineering and SysML

Model-Based Systems Engineering is defined by INCOSE as “the formalized application of modelling 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.” (INCOSE-TP-2004-004-02).

MBSE is a structured, system-theoretic and holistic methodology that utilises models to facilitate System Engineering. A systems-theoretic approach places emphasis on both the whole system and the complexity manifested in the interactions between its constituent parts. Although holistic, it is complementary to the (more traditional) reductionist approach; relying on extracting the necessary information about the parts to construct an abstract description of the whole system. MBSE encompasses (reductionist) hierarchies of information for example, describing the desired functional behaviour of a system through functional (hierarchical) decomposition or the definition of physical assemblies and sub-assemblies that also form natural hierarchies. MBSE augments this approach with the ability to interrelate the information about behavioural and physical parts that cut across these hierarchies. Interrelationships that span hierarchies are often referred to as ‘cross-cutting’ and are used both to build and maintain an understanding of the intricate network of interactions within complex systems.

A barrier to using models within the domain of systems engineering is that the notations used in Analytical Models are often specific to a specialist domain of engineering. Using a specific notation out of context for qualitative and descriptive models typically results in the notation being abused. The semantics of the notation are distorted to satisfy an alternate meaning which negates the purpose of a notation, to facilitate communication required to gain a shared understanding. The shared understanding is difficult to attain as everyone’s interpretation of the notation is based on different semantics. To alleviate the problem of all systems engineers having to learn every notation used in analytical models in order to integrate the overall system, the Systems Modelling Language (SysML) was created. Based on UML 2.0, SysML provides systems engineers with a coherent, standardised way of representing system components and their interactions in a way that is easily transformed, manipulated and maintained. A full description of SysML is beyond the scope of this paper. However the interested reader is directed towards the Open Modelling Group’s SysML website (Open Modelling Group 2012) which contains documentation and examples.

A key way that SysML supports MBSE is through the implementation of views. Systems Engineers are required to address multiple, interrelated concerns (e.g. non-functional quality attributes) about a system of interest at the same time. In practice it is difficult to analyse multiple interrelated issues simultaneously. A common strategy for

managing this task is a technique called “Separation of Concerns” (Dijkstra 1974). A concern (or related set of concerns) becomes the focus of the Systems Engineer’s attention and aspects of system of interest are defined based on satisfying the needs of the one concern. Other concerns then become the focus of attention and the definition of the system of interest is updated. In effect the Systems Engineer is “being one- and multiple-track minded simultaneously”. Concerns can be revisited as many times as is necessary. Using separation of concerns requires an iterative and incremental approach to systems engineering to evolving a definition of the system of interest. In SysML each “concern” is a specific “view” of the system model, designed to capture the necessary information to address it (BS ISO/IEC 42010 2011). Most of the information required to support engineering decision making can be described using SysML. The SysML can be extended (extending the semantics of existing model elements with view-specific semantics) meaning that in time all the necessary information to support a specific view will be integrated into a model.

In the standard views supported by SysML, security is not a consideration. This lack of consideration is possibly rooted in the fact that in highly generalised standards, such as IEC 15288 for Systems Engineering, neither security nor safety receives a great deal of attention (ISO/IEC 15288 2008). Whilst there are guidelines for their incorporation, they are considered optional features of system lifecycle.

2.3. Security-Aware Systems Engineering

As discussed, there is an inherent danger to relying on purely topological restrictions, such as firewalling and DMZs, when securing ICSs. As well as these measures, systems engineers and developers need to ensure that security is a ubiquitous property of the systems being produced.

Security-aware systems engineering carries with it a number of benefits. Most obviously, systems with additional security are less prone to downtime and disruption caused by malicious users. International standard IEC 21827 (Systems Security Engineering – Capability Maturity Model) outlines additional side benefits for engineering organisations pursuing a security-orientated approach as: “savings with less rework from repeatable, predictable processes and practices; credit for true capability to perform, particularly in source selections; focus on measured organizational competency (maturity) and improvements” (BS ISO/IEC 21827 2008). Given the benefits in terms of reliability and cost, a body of work exists in the area of security-aware systems engineering and software engineering.

There are several international standards which apply to the software/systems engineering of

secure systems. As discussed, IEC 21827 (Systems Security Engineering – Capability Maturity) has relevance to the field by introducing the motivation for requiring secure systems and by outlining a means of assessing an organisation’s procedures for the development of such systems (BS ISO/IEC 21827 2008). However its primary focus is assessing an organisation’s maturity in terms of having established cyber security processes. This standard complements IEC 13335 (Security Techniques – Management of Information and Communications Technology Security) which provides clear guidelines on the formation of cyber security policies as an on-going and coherent activity across an entire organisation (BS ISO/IEC 13335 2004). Of note is IEC 13335’s formal definitions of the core components of cyber security namely: confidentiality, integrity, availability, non-repudiation, accountability and authenticity. These definitions are discussed in more detail in Section 3.3. Both IEC 13335 and IEC 21827 are ICT network-orientated and lack any ICS specific information.

IEC 62443 (Security for industrial process measurement and control) is specifically targeted at ICSs. One way in which the standard offers support is through the presentation of guidelines for network topologies that protect the external interfaces of ICSs. The ICS-specific advice on combining electronic and physical barriers to protect ICS networks takes into account that not only is the field interested in designing secure systems from scratch, but also adapting older, insecure designs to enhance their utility in the future (BS IEC/PAS 62443 2008). A weakness of the standard is its scope is chiefly limited to the topological structure of the network, and lacks a deeper, holistic perspective.

The Carnegie Mellon’s Software Engineering Institute has published a number of “Secure Design Patterns” through its CERT program (CMU/SEI-2009-TR-010 2009). Designed to maximise potential reuse in a number of design scenarios, these high-level templates provide strategies for use within architecture, module design and implementation for security conscious systems. Each pattern is inspired by a common security vulnerability such as erroneous elevation of permissions (a common exploit used to execute malicious code or access resources that the user does not have the correct level of clearance to access) or incorrect input validation (a common exploit used to inject SQL commands via an interface to a database.) Each pattern is made up of a number of predefined elements that dictate the circumstances where the design pattern could be of use, the entities involved and sample code and uses. By analysing the information flow and use cases of systems, it is possible to identify functionality and interactions that fit with the pre-

existing design patterns, and create solutions that avoid common pitfalls. This has been suggested as an important role of UML in generic secure systems development (Jürjens 2004).

With many standards and guidelines targeting secure architectures and design patterns encouraging secure development, it is clear that security is a multi-scale, systemic issue. UMLsec is an extension of UML purely aimed at software development. It is an attempt to allow security-aware development to be incorporated into UML through the introduction of a number of new stereotypes that explicitly record security-specific concepts such as encrypted links (Jürjens 2002). In related work Jürjens provides an in-depth analysis of several standard UML diagrams that can be used to augment a security-focussed analysis of software systems (Jürjens 2004). For the physical elements of the system it relies on the deployment diagram to view physical connections between devices. There are significant limitations to applying this technique to ICSs as the complexity of the hardware aspects of the system are not captured by a standard deployment diagram and as such, valuable information about the security of ICSs is lost.

Whilst UMLsec is a useful step for ICSs, it is much more useful to consider the system-wide perspective supported by SysML. AVATAR is a SysML-based environment developed as part of the European project EVITA, a consortium of security, academic and automotive partners exploring security-aware on-board networks for the automotive industry (EVITA 2008). AVATAR extends SysML with a number of security-based stereotypes and functions, including the use of temporal functions within the standard state-machine diagram that allows delays between states and minimum processing times to be modelled (Pedroza 2011). Other extensions are designed to allow traditional cryptographic algorithms to be added as parts of the model. A formal proving language is then used to verify that the security requirements of the system are being met. An important step forward, the techniques suggested by the authors focus heavily on the cryptographic aspects of cyber security, with a less detailed exploration of the systemic aspects, such as identifying architectural vulnerabilities and access controls. This focus on encryption makes identifying vulnerabilities beyond confidentiality breaches unrealistic with AVATAR in its current form. In the future it may be possible to incorporate elements of AVATAR and UMLSec into the proposed SysML extension to take advantage of their cryptographic representations, but this would need to be done in response to an identified need to explicitly represent the cryptographic techniques used by the system of interest.

Additional attempts to explicitly model security concerns within SysML and UML exist within the military sector (Jamjoom 2012). The majority of these approaches make modifications to the existing diagrams to allow security requirements to be explicitly described. However, little work is done from the perspective of automatically pulling relevant information from the existing model to highlight security vulnerabilities.

An alternative to building on UML or SysML models is the utilisation of probabilistic relational models (PRMs). PRMs are similar to class diagrams in UML but utilise probabilistic attributes (both quantitatively and qualitatively) to represent probabilistic attributes of classes and probabilistic relationships between attributes. In (Somestad 2012) a PRM-based language for modelling enterprise level architectures is proposed (CySeMoL). A probabilistic inference engine generates attack probabilities (i.e. the probability that an attack will both be performed and succeed) based on data fed from previous attacks. The advantage of this technique is that system architects can reuse components and inherit previously stored data on attack vectors. In addition, all known attack types, including denial of service, are recorded, providing a complete view of the attack landscape. The technique is focussed at enterprise networks and whilst there are similarities, it is noted that this technique also suffers from being unable to represent physical access to system. A further disadvantage is that the reliance on accurate probabilistic estimates makes it difficult to apply to industrial control systems, an environment where the number of attacks is significantly smaller than enterprise networks, and, it could be argued, that the victims of such attacks are less likely to report the incident. From a practical perspective, an extension of PRMs will currently suffer from PRMs lack of uptake in industry. SysML is currently the de facto standard amongst systems engineers, and a highly extensible language. The incorporation of such probabilistic models is beyond the scope of this work, but a possible future direction.

3. SYSML FOR SECURITY-AWARE INDUSTRIAL CONTROL SYSTEMS

In order to secure ICSs, it is important to mitigate potential threats at the local, network and system level. SysML is already a powerful modelling language and could provide immediate benefits to the security-conscious developer. In Section 3.1 some of those benefits are presented and in Section 3.2 an extension to SysML is presented.

3.1. Existing SysML

There are obvious and immediate benefits to introducing SysML to the design of secure ICSs. Primarily, SysML provides a standard way for systems engineers and security specialists to view systems of interest, going some way to break down the barriers between the two disciplines.

Models are typically built either as part of the lifecycle of the system or, in the case of legacy systems, built up retrospectively for documentation purposes. As such, using the models as part of the security process of an ICS fits with the typical business usage of SysML with little additional modelling overhead.

As models are generally treated as “living documents” maintained to reflect design choices and revisions, they are a valuable source of information for the cyber security specialist, who requires an accurate, up-to-date model of a system in order to adequately secure it.

In control systems, security threats are not always immediately apparent at a local level. Seemingly innocent parameterisations of subsystems could yield catastrophic results when their collective effects are considered. Being able to pre-empt such interactions is an important step in securing the system from malicious use and only possible with a coherent, system-wide view of both the software and hardware components of the system.

Of particular use would be the physical models of the system (allowing easy determination of control points for critical systems); and the component model (allowing the system architecture to be compared to known secure design templates and enhanced accordingly.)

3.2. Extending SysML to be Security Aware

As discussed, SysML in its original form has a great deal to offer stakeholders interested in securing ICSs. However, the notion of adding new features to extendable modelling languages in order to better view security concerns has precedent and SysML supports this endeavour. As a crucial concern to the systems engineer, security is a prime target for the basis of a specific view into the model data. Through the separation of concerns paradigm, this allows the systems engineer to address security issues specifically as a core part of the system life cycle.

For the duration of this work all modelling will be performed using Atego’s “Artisan™” software.

The type of information that a security view should contain should be determined by a wider group, working towards a generic standard. Here a “first draft” of such a standard is presented, to provide a basis for discussion and development.

3.2.1. Developing a Threat Agent Profile

In order to identify what information is useful to a security view, it is important to represent concepts

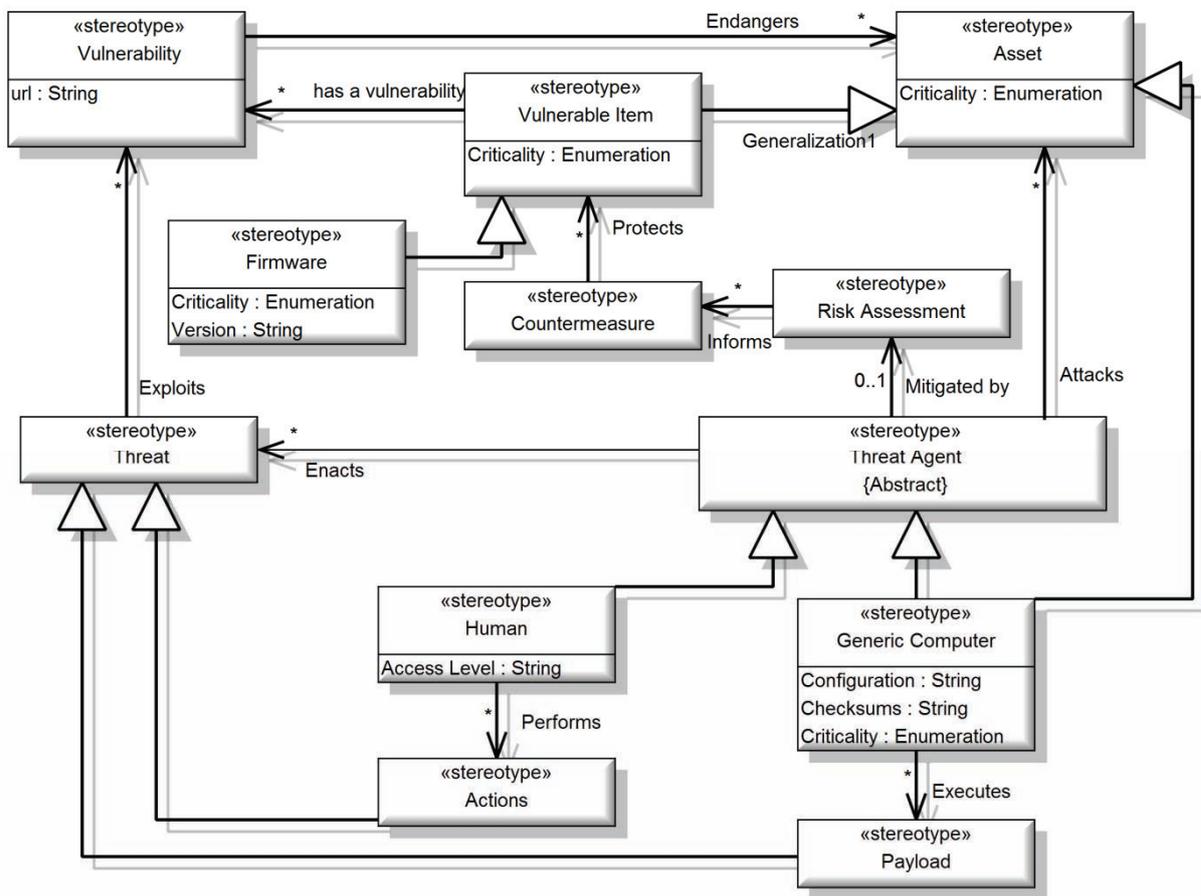


Figure 1 : A SysML Threat Agent Profile

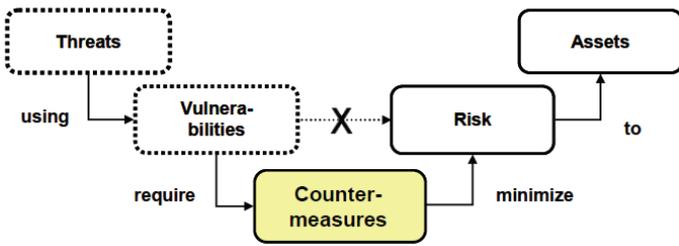


Figure 2 : Threat-risk Relationship (taken from BS ISO/IEC 21827, 2008)

that are pertinent to ICS security within the SysML framework. Figure 2 presents the threat-risk relationship from (BS ISO/IEC 21827 2008). This model is a high-level description of the dangers posed by cyber security to ICSs. It identifies that threats exploit vulnerabilities, which endanger assets. Vulnerability-specific countermeasures minimise the risk to assets. These relationships help to define the connections between threats, vulnerabilities, countermeasures, risk and assets, but SysML’s holistic view of a system allows us to expand those relationships to cover a wider range of concepts. Figure 1 presents a more comprehensive threat model designed to form the basis of a security view for SysML.

Figure 1 is a series of interconnected stereotypes that can be applied to blocks when modelling a system. In addition to those concepts presented in Figure 2, the SysML threat agent profile describes how threats are the product of threat agents and elucidates on assets that are vulnerable as a result of known vulnerabilities.

Threat agents can either be a human or a generic computing device (PLC, laptop or similar) that has been infected with malware and is now able to pass an additional malware payload on. Human threats are actions (such as inserting a USB stick into a computer or deliberately deactivating a safety-critical system) and computing threats are modelled as the delivery of malware payloads or through enacting a more generic threat (such as the sending of malicious commands). Computing devices are simultaneously threat sources and assets to be protected due to the cascading nature of multi-payload attacks. The model also captures that countermeasures are sourced from risk assessments of specific threat agents, closing the loop from threat agent to countermeasure.

3.2.2. Developing a Data Model Profile

Whilst the threat agent profile provides details about how threats interact with assets and vulnerable items it lacks a model of how assets and data interact. Obviously without incorporating the relationship to data, from a cyber security perspective, the model is not yet complete. Figure 3 is a graphical representation of the data model which captures the relationship between assets, data storage devices and communication protocols.

Currently the data model’s information about communication protocols is limited as it fails to capture the multi-layer nature of communications protocols. For this work, the “Communications Protocol” stereotype should be considered as a model of physical connections between devices or, in the case of wireless networks, an acknowledgement that data is capable of flowing between connected assets.

3.2.3 Extending the Modelling Tool

With the relevant profiles contained within the system, it is possible to apply these stereotypes to elements with existing models.

Asset: An asset defines any resource that could be the target of an attack. A representation of the “criticality” of the asset is stored to allow different assets to be prioritised in terms of their importance to the operation of the system.

Communication Path: The communication path stereotype can be applied to any relationship between model elements and any block which represents a communication protocol such as TCP/IP. It ensures that all communication paths are capable of representing if they are encrypted or not and an assessment of their criticality to the system.

Data Storage: Data storage devices are assets and data handlers, so can represent if the data stored is encrypted and how critical the storage device is.

Firmware: Firmware is a stereotype that is a specific type of vulnerable item. It has a version representation that allows a computer security auditor to search vulnerability databases for potential vulnerabilities.

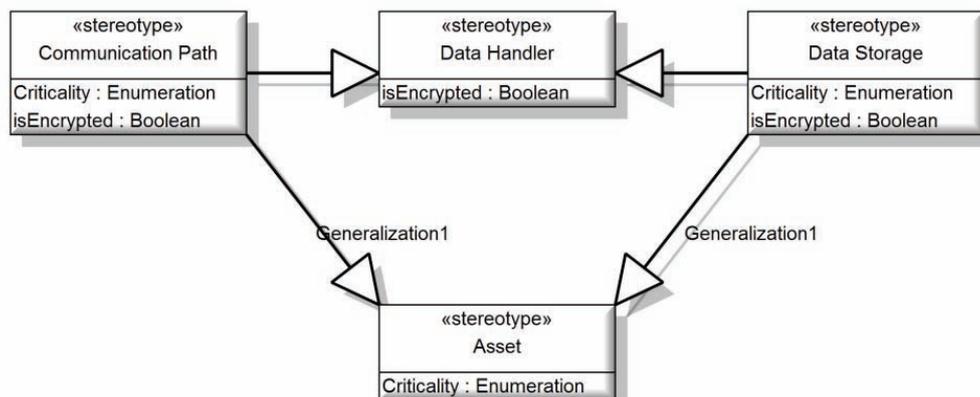


Figure 3: SysML Data Model Profile

Generic Computer: The generic computer stereotype represents any PC, PLC or other computational device. It ensures that all computational devices have a means to represent their configuration, checksums and criticality. The configuration is currently listed as a string, but in more advanced versions of the model should be a data type capable of accurately defining the deployed software pertinent to the use of the ICS. Likewise, the checksums representation should represent the message digests of the software and data held as part of the configuration, to enable checking for Trojan attacks.

Vulnerability: Vulnerability is a stereotype that is applied to classes. These classes capture the url of known vulnerability reports and allow them to be associated with specific vulnerable items.

Vulnerable Item: A vulnerable item is stored in the threat model as a type of asset. It should have a countermeasure that protects it and is associated with a vulnerability class.

3.3. A Cyber Security Perspective

The addition of threat and data models to SysML allows new information to be incorporated into the model and a security view (i.e. a collection of visual representations of the system to display the new information) to be created. The security view should address the primary security concerns as set out by BS ISO/IEC 13335 (2004).

3.3.1. Confidentiality

Confidentiality is defined as “the property that information is not made available or disclosed to unauthorized individuals, entities, or processes” (BS ISO/IEC 13335 2004).

In order to address confidentiality, it is important to address how information is communicated. The communication path stereotype enforces the need

for communications to be tagged as either encrypted or not. By ensuring that communications between devices are flagged in this way it is immediately apparent from diagrams showing the communications between devices, whether the data passing between them is secure. Note that by employing a simple Boolean flag the systems engineer is free to declare channels as encrypted and leave the details of the encryption technique for more detailed specification at a later date. At this time there is no formal representation of encrypted data. Identifying data as being just another asset would make it possible to place it within the threat model but it would not apply an “isEncrypted” flag, as that is not a concept which is applicable to all asset types.

3.3.2. Integrity

Integrity is defined as “the property of safeguarding the accuracy and completeness of assets” (BS ISO/IEC 13335 2004).

The issue of integrity for ICSs is important as attacks that compromise the user-defined application data running on PLCs and the interface libraries between PC base stations and PLCs have been recorded in the wild, such as the now infamous Stuxnet.

The use of SysML adds an additional meaning to “integrity” as the security view will be of no use for continuous security monitoring if the integrity of the model to the system is compromised.

By ensuring that all process-critical software and data has a recorded checksum, it is possible to run periodic, automated checks to ensure that the current configuration has maintained its integrity. Recording the checksums for all software and data in the tool chain is feasible for an ICS as updates are typically less frequent than those on standard IT systems and should always be done following a controlled deployment strategy. Such a strategy

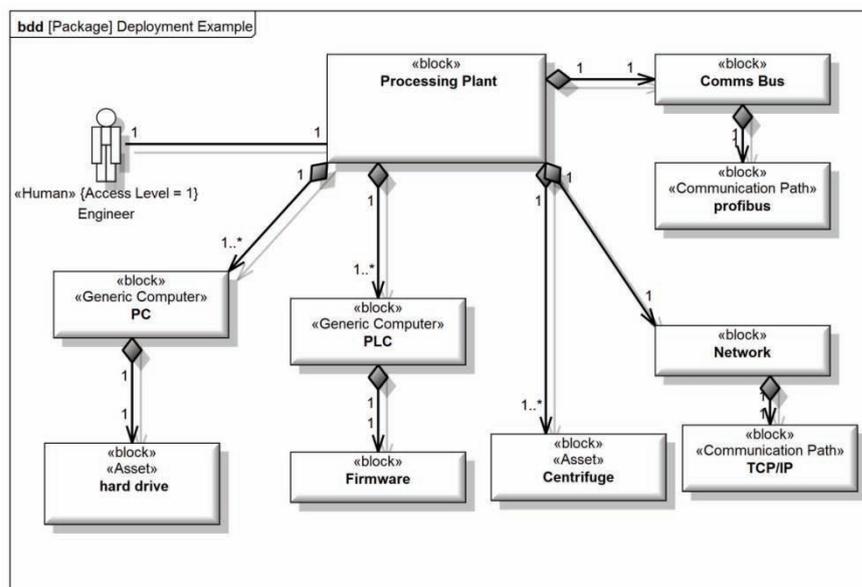


Figure 4: Block Deployment Diagram

should include the updating of system models to reflect the changes.

3.3.3 Availability

Availability is defined as “the property of being accessible and usable upon demand by an authorized entity” (BS ISO/IEC 13335 2004).

At their least harmful, attacks which compromise the availability of an ICS by denying users the ability to access remote services or start the process, can have a significant financial cost. At worst, the service being denied is some internal control mechanism, which, if denied enough time to complete its control computation within the specified time slice, could lose control of the electro-mechanical system of interest.

Therefore, it is important to not only examine interfaces to check for possible sources of denial of services to users, but also to work backwards from critical assets to look for links to threat agents that could bombard a controller with messages and ensure that sensible message handling procedures are being followed.

Much of the information required to perform these assessments is included in the original SysML diagram. However, the additional stereotyping of ports into vulnerable items makes it possible to query the tool to identify all ports that have not been associated with a mitigation action.

In (Jensen 2010) the availability of control systems used within the energy sector is investigated through interviews with a utility’s staff. The systems considered with the SCADA, outage management and distribution management components. The investigation yielded that many outages were rooted in the software applied to both the primary

and backup systems, meaning that hardware redundancy was undermined when software could not be relied upon. This has ramifications for security in that, unless degenerate redundancy techniques are employed (whereby the backup system has a completely separate design to the primary), vulnerabilities found in the software of one system will exist in the other.

3.3.4. Non-Repudiation

Non-repudiation is defined as “the ability to prove an action or event has taken place, so that this event or action cannot be repudiated later” (BS ISO/IEC 13335 2004).

The actors in a control system are able to trigger events through the explicit sending of commands or the reporting of variables that are associated with pre-defined triggers. In order to ensure the non-repudiation of an industrial control system, a major factor is the accurate recording of log data identifying who issued what command. This interacts with the importance of data integrity, as a malicious user may deliberately edit logs to compromise the non-repudiation of the system. In addition, the data recorded in the logs is of little use if the originator of the command has not been authenticated.

3.3.5 Authenticity

Authenticity is defined as “the property that ensures that the identity of a subject or resource is the one claimed. Authenticity applies to entities such as users, processes, systems and information” (BS ISO/IEC 13335 2004).

For ICSs authenticity has two interactions: a direct interaction (through the authentication of devices being the device that they claim to be) and an

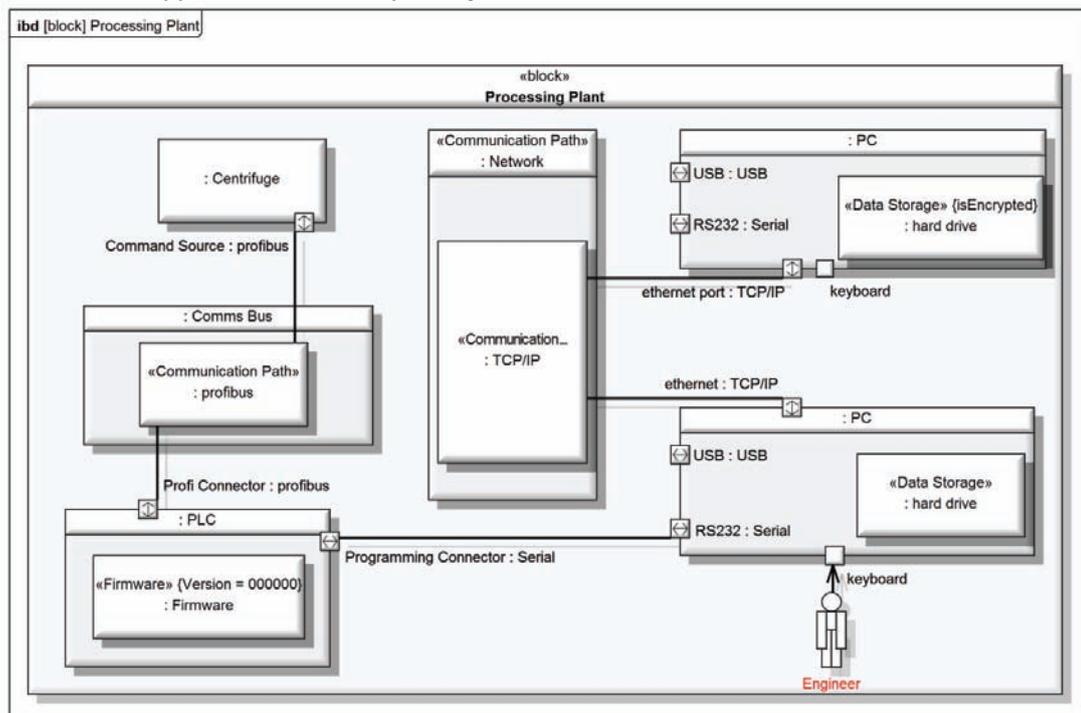


Figure 5: Internal Block Diagram

indirect interaction (through the role authentication plays in the process of authorization.) Authentication is an extremely complex issue in ICSs as for safety critical systems the need to authenticate could add an unacceptable amount of time between detecting and responding to an event. To add further complexity, the actions associated with handling an emergency situation (emergency shutdowns etc.) are often exactly the type of actions that a malicious user would benefit from accessing. Having policies where security is relaxed during emergencies is dangerous as it adds extra incentive to malicious users to cause a failure and may tempt actual users to perform potentially dangerous actions to circumvent safeguards. In addition encryption and authentication add bandwidth overhead to a communications stream, which is undesirable for real-time control.

Once a user is authenticated, it is possible to move to the separate process of authorization. Humans, as threat targets, should be associated with an access level which makes assessing the risk that they pose to the system more realistic. It is difficult to identify other items within the threat model that should be associated with a security level, as the specifics of implementing an authentication process are beyond the scope of a high level model. The asset stereotype has not been associated with a security level as a single device may have multiple modes of functionality that each has its own access level. For example, it is common to have parameterisation at a higher level of security than straight on/off commands. This is a potential area for further work.

3.4 Example System

To illustrate the benefits of security aware SysML, an example system is presented and analysed from a security perspective. The system of interest is a small processing plant comprising two PCs, a control PLC and a centrifuge. The block deployment diagram in Figure 4 illustrates the component relationships. Stereotypes are displayed as text within double-angled brackets (as is the norm for SysML).

More detail about the relationships between the blocks is given in the internal block diagram in Figure 5. Certain “tags” (elements from the stereotypes) are displayed in addition to the stereotypes. This clearly shows the firmware version and identifies that only one of the modelled PCs is encrypted.

The system can now be reviewed from a security perspective. The addition of the security-relevant data makes the model itself a useful resource. In addition, the model can be queried to collate lists of important features. In Figure 6 the output of the modelling tool for this model is given. It collates

together lists of all of the vulnerable items, allowing a security auditor to review and mitigate each element. Data storage devices are also collected together, and can be individually checked to call up properties such as their encryption status and criticality. Likewise, communication paths are brought together so exhaustive checks can be performed. For larger systems, checking each link is unlikely to be feasible, but as assets, communication links are prioritised at model building time to allow a security auditor to zone in on important connections. Even mis-categorised links should be easily determined by simply looking at the model.

Specific vulnerabilities, stored as classes in the system, can be associated with vulnerable items, allowing systems engineers to quickly find the information required to mitigate the problem. In Figure 6 a “PLC DLL” vulnerability is associated with an RS232 port that highlights the potential threat from the Trojan attack.

4. CONCLUSIONS

It has been demonstrated that SysML can be extended to be security-aware. The advantage of using SysML over other modelling languages for ICSs is that SysML lends itself far better to control systems modelling than software-orientated languages. The profiles added to SysML have been based on terminology from relevant computer

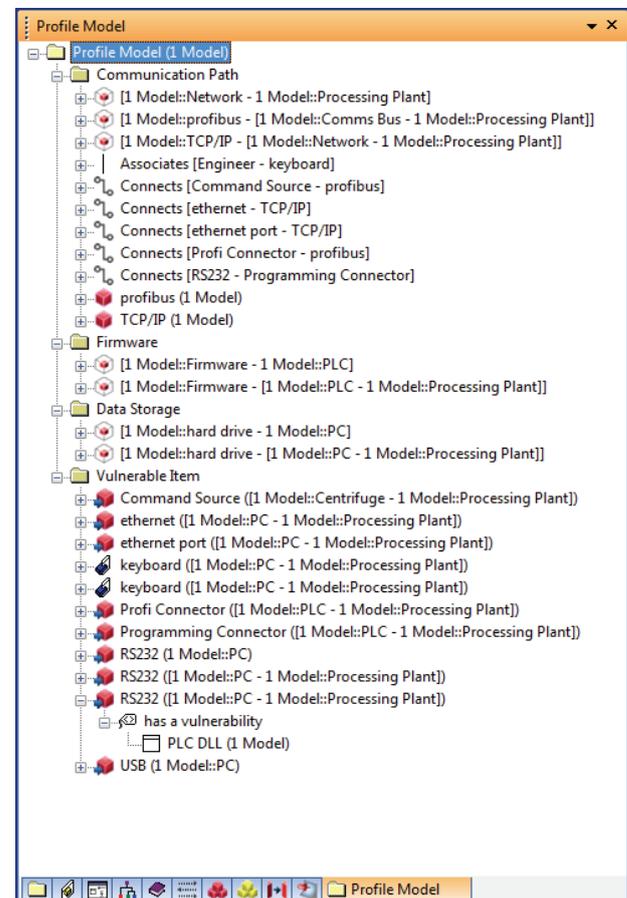


Figure 6: Output from the modelling tool

security standards and can be easily applied to new and existing models. It has been shown that information about the model can then be converted into reports, making the continuous process of securing systems against potential attacks easier. This work presents an initial view that should be explored with members of the Object Management Group, computer security experts and industrial control experts. Ultimately, these profiles can be refined and incorporated into SysML as an integral part of the modelling process, encouraging model-based systems engineers to consider security as one of the core concerns of system design.

Whilst this technique shows promise in terms of the traditional five security concepts, it is important to observe that attacks that rely on subverting the design of a control system (such as supply chain attacks) are difficult to pre-empt using system modelling.

In the future the security aware SysML can form the basis of a number of security improvements to the generation of ICSs. Secure design processes and standards that examine the role of each phase of the product life-cycle and its relationship with security should be created for the ICS environment. Such processes could be assisted by formal verification tools that would ensure systems are robust and fit for purpose in the modern world.

5. REFERENCES

- BS ISO/IEC 13335 (2004) Security Techniques – Management of Information and Communications Technology Security. London, BSI
- BS ISO/IEC 21827 (2008) Security Techniques – Systems Security Engineering – Capability Maturity (SSE-CMM). London, BSI
- BS ISO/IEC 42010 (2011) Systems and Software Engineering – Architecture Description. London, BSI
- BS IEC/PAS 62443 (2008) - Security for Industrial Process Measurement and Control. London, BSI
- Centre for the Protection of National Infrastructure (CPNI). (2011), Cyber Security of Industrial Control Systems, CPNI
- CMU/SEI-2009-TR-010 (2009), Secure Design Patterns. Carnegie Mellon, Software Engineering Institute.
- Department of Homeland Security (DHL). (2009), Recommended Practice: Improving Industrial Control Systems Cybersecurity with Defense-In-Depth Strategies, DHL
- Department of Homeland Security (DHL). (2011), ICS-CERT Incident Response Summary Report, DHL
- Dijkstra, E.W. (1974), On the Role of Scientific Thought. Selected Writings on Computing: A Personal Perspective, 60
- European Network and Information Security Agency (ENISA). (2011), Protecting Industrial Control Systems, ENISA
- EVITA (2008), EVITA: E-safety vehicle intrusion protected applications. www.evita-project.org (April 2013)
- GAO-04-140T (2003), Critical Infrastructure Protection: Challenges in Securing Control Systems, United States General Accounting Office
- INCOSE-TP-2004-004-02 (2007), Systems Engineering Vision 2020, International Council on System Engineering
- Industrial Control Systems Cyber Emergency Response Team (ICS-CERT) (2013), ICS-CERT Advisories and Reports Archive <http://ics-cert.us-cert.gov/ics-cert/archive.html> (April 2013)
- ISO/IEC 15288 (2008), Systems and Software Engineering – System Life Cycle Processes, Geneva, International Organisation for Standardisation
- Jamjoom, M.M., Alghamdi, A.S. and Ahmad, I. (2012), Service Orientated Architecture Support in Various Architecture Frameworks: A Brief Review. Proceedings of the World Congress on Engineering and Computer Science 2012, Volume II, San Francisco, USA
- Jensen M., Sel, C., Franke, U., Holm H. and Nordstrom, L. (2010), Availability of a SCADA/OMS/DMS System – A Case Study. Proceedings of ISGT Europe, 2010
- Jürjens, J. (2002), UMLsec: Extending UML for Secure Systems Development. 5th International Conference on the Unified Modeling Language (UML 2002), Volume 2460, Pages 412-425
- Jürjens, J. (2004), Secure Systems Development with UML. Berlin: Springer
- Matherly J. (2009), SHODAN Computer Search Engine. www.shodanhq.com (April 2013)
- North American Electric Reliability Council (2007), Top 10 Vulnerabilities of Control Systems and Their Associated Mitigations-2007, NAERC
- Open Modelling Group (OMG), (2012), OMG Systems Modeling Language: The Official SysML site, OMG <http://www.omgsysml.org/>
- Pedroza, G., Apvrille, L. and Knorreck, D., (2011), AVATAR: A SysML Environment for the Formal Verification of Safety and Security Properties. In New Technologies of Distributed Systems (NOTERE), Paris, France.