Context-Aware Security Solutions for Cyber Physical Systems

Kaiyu Wan[1] and Vangalur Alagar[2]

[1] Xi'an Jiaotong-Liverpool University, Suzhou, Jiangsu, China
kaiyu.wan@xjtlu.edu.cn
[2] Concordia University, Montreal, Canada
alagar@cs.concordia.ca

Abstract. The integration of physical systems and processes with networked computing has led to the emergence of a new generation of engineered systems, called Cyber-Physical Systems (CPS). These systems are large networked systems of systems, in which a component system may itself be a grid. In this paper we survey the current state of the art of CPS, identify the issues surrounding security control, and investigate the extent to which context information may be used to improve security and survivability of CPS.

Keywords: cyber-physical systems, secure control, context-awareness, security architecture.

1 Introduction

Cyber-Physical Systems (CPS) [3] integrate computing and communication in a super-large scale in order that its capabilities will include controlling and monitoring the physical world in which it is embedded. These systems are large networked systems of systems, in which a component system (also called site) may be a grid, or a real-time reactive system, or a system that provides ubiquitous computing environment. A direct link between two CPS components is itself a network, because each component might be a grid and several links exist between these two grids. The largeness, the heterogeneity in computing, communication platforms, and resources having domain specific properties make CPS hard to comprehend, design, and operate. The NSF program description [6] states the grand view that CPS initiative is "to transform our world with systems that respond more quickly, are more precise, work in dangerous and inaccessible environments, and provide large scale distributed services." The strategic application domains of CPS [6] include monitoring and managing large-scale physical infrastructures (demand-driven power distribution, environmental monitoring and protection, and water systems management), health care (perpetual life assistance for disabled, high confidence medical aid to remote areas), transportation (congestion control, safe evacuations), and defense systems (avionics). In all these applications there is a need to provide timely services at every *context* of service request. Since service requests may be generated by (non-technical)

P.C. Vinh et al. (Eds.): ICCASA 2012, LNICST 109, pp. 18–29, 2013.

humans and physical devices (that are no longer dumb) the system must be intelligent enough to either provide or refuse services at different contexts, while assuring *safety* and *security*. Breach of security or lack of security might force the system to violate safety properties. Many of the CPS applications are safety-critical. Failure to provide services at the right contexts or providing services to unauthorized entities in any context can result in irreparable harm to the physical world surrounding it. This is the motivation for us to identify security issues in CPS and investigate the role of context-awareness in improving the extent of CPS security. We have structured the paper as follows: In Section 2 we make precise the notion of critical assets and bring it out the relevance of context. In Section 3 we motivate the need for a context representation and situation evaluation. In Section 4 we formalize the notion of context-awareness, using the context representation. In Section 5 we describe a generic context-aware security architecture and explain how context-awareness can be effective for CPS security. In Section 6 we comment on certain aspects of CPS security where context-awareness may not be relevant, and conclude the paper with a brief summary of our ongoing work.

2 Critical Assets

We use the term *asset* in a generic sense to denote an entity that is relevant and essential to CPS operation. All entities required for perception of the environment, computation of CPS processes and control units, and communication between CPS sites are assets. An asset might be a resource, or a physical device, or a controller, or a switch, or a protocol. In order that CPS operations progress without failure and CPS survives external and internal attacks it is essential to *secure* the assets. We label an asset *critical* if its quality degradation adversely affects the physical world surrounding it, and also mitigates to other system assets disrupting the operational goals of the system. It is absolute that critical assets are protected during their life time.

The two basic questions are 'who labels a resource as critical?' and 'how critical assets are to be classified'. The criticality level can be assessed only by an expert in the resource domain. A risk model is necessary to assess the critical level of an asset, and measure the damage caused by its unreliable performance or untimely availability. So, the answer to the first question is 'resource experts label critical resources'. As an example, an expert in the water sector domain might conclude that an interference with the operation of water treatment equipment is highly risky because it will cause chemical imbalance in drinking water, which in turn is catastrophic to the clients of that water company. So, the domain expert will label the water treatment equipment as a critical asset. In answering the second question we contend that a resource expert might classify a resource as *highly critical* in one context and *not critical* in some other context. As an example, a control engineer might conclude that if the control system software in a power plant is tampered by unauthorized personnel, it is likely to alter drastically the behavior of power distribution, causing blackouts. So, 'control

system software' should be labeled as a critical asset. However, the control engineer with data from the marketing team might conclude that in some regions blackout might be tolerable, whereas in many other regions it will not be acceptable. So, the control system software may be labeled *highly critical* in the regions where blackouts are unacceptable, and labeled *not critical* in the regions where blackouts are tolerable. In general, the resource domain experts should classify assets and determine their criticality levels in different contexts. Based upon this classification, security policies may be enforced in a context-aware security architecture to secure the critical assets in CPS. For the sake of definiteness let us assume $\{CL =$*most critical (mc)*, *critical* (cc), *average critical* (ac), and *not critical* (nc)$\}$ is the set of labels used by domain experts to label resources.

3 Context-Dependence

The life-cycle of an asset has several states. An asset might be *discovered*, or *produced*, or *procured*, or *requested*, or *idle*, or *allocated*, or *delivered*. In each state several contexts might exist and the asset needs to be secured in these contexts. It is impossible to identify all possible contexts in each state. However, the application domain expert should be able to identify the *situations* in each state where the asset should be tightly controlled. Situations may be formulated as *rules*. Business rules and legal rules governing asset states are known to the application domain expert, they can be put together as a framework to constrain the situations. Any context that validates a situation then becomes a context of interest. Thus, searching for contexts of interest and formulating situation constraints are related to one another.

Context information is *heterogeneous* and *multi-dimensional*. As an example, water is an ubiquitous asset. A business rule of a water sector company might be "*water may not be sold to a client who is located in a zone Z either because the company is not permitted to supply water in zone Z or the water quality does not meet the standards of zone Z*". As another example, copper mined in a location is a local asset. The legal rule "*copper may not be made available in cities within 100 km from the mine*" constrains the asset allocation. As a final example, consider nuclear power asset. In many countries strict laws might forbid or restrict the sale and use of this asset. These examples suggest that (1) locality (region), type of asset, and quality information are the heterogeneous and multi-dimensional context information, and (2) actual contexts for allocating the assets should satisfy the business and legal rules (situations). Therefore, context information must be structured in such a way that situations can be evaluated at the contexts.

Context, as a first class entity, has been studied in many disciplines, including linguistics, philosophy, and AI. However its importance in ubiquitous computing is unmatched by other disciplines. In computing, context was made popular by the seminal works of Dey [4] and Winograd [9]. A formal context representation and context calculus was developed by Wan [7]. Wan [8] gives a survey of contexts, logic of contexts, and context calculus. It is this notation and ideas that we

briefly sketch below and use it to formalize context-awareness. A *context space* is defined for an application in a domain and contexts are constructed within that space. A context space includes a finite set of *dimensions* and a *type* associated with each dimension. The typed values are called *tags* along each dimension. The asset manager chooses the context space by defining the dimensions and tags that are necessary to safeguard the asset access, and regulate asset distribution. Such dimensions are usually hidden in the rules governing asset states. A set of generic dimensions and the tag types, suggested in Wan [8] are WHO, $WHAT$, $WHERE$, $WHEN$, and WHY. Their meanings and the tag types associated with them are given below.

- WHO: This dimension is used to specify the owner of the asset, or the role of the client requesting the asset. Its type is either *String* or *enumeratedset*.
- $WHAT$: This dimension is used to specify the type of asset available or requested. Its type is *String*.
- $WHERE$: The target location where the asset is to be delivered. Its type is either *String* or *enumeratedset*.
- $WHEN$: The time when the asset is to be delivered. Its type is either *String* or an *Abstractdatatype*.
- WHY: The purpose for requesting the asset. The type is either *String* or *enumeratedset*.

Depending on the asset domain it is possible that additional dimensions may be included in context information. Domain-specific ontologies are to be made available for fine grained specifications and use of contexts. As an example, with the help of ontology dimension names such as 'LOCATION' and 'WHERE' might be considered as synonyms, and will share the same tag type. The toolkit developed by Wan [7] includes a context representation, relational semantics for contexts, and a context calculus. This toolkit expects a context sensing unit to transform the raw context information into the context notation $[D_1 : v_1, D_2 : v_2, \ldots, d_n : v_n]$, where v_i is the value sensed in dimension D_i. An example of a context in this representation is $c = [WHERE : mysite, WHEN : 11/20/2012, WHO : Alice, WHAT : Proal]$, the setting in which either *Alice* has the asset *Proal* (as asset owner) or *Alice* requests the asset *Proal* (as asset requester). The context calculus allows contexts to be composed, compared, and decomposed using relational operators. It allows a formal evaluation of situations at a context. We had suggested that the domain experts, with the help of business executives, first determine the situations for asset states, and next discover the contexts hidden in the rules. Therefore, it is justified to assume that a situation is encoded as a logical formula p on the dimension names and other variables. A given situation may be valid in many different contexts. Given a context c, in order to check that a situation p is true in context c, the dimension names in p are bound to the tag values in the definition of context c and p is evaluated. An example situation is the predicate $can\_deliver == (\mid x - WHERE \mid < 100) \wedge (d_2 < 3 + WHEN)$, where $\mid \ldots \mid$ denotes the distance expression and $(3 + WHEN)$ means within 3 days of specified time. When evaluated at c, we will get the

expression ($\mid x - mysite \mid < 100$) $\wedge$ ($d_2 < 11/23/2012$). Given values for the location variable x and date variable d_2 this expression will evaluate to either true or false.

3.1 Context-Dependent Labeling of Assets

In CPS it is hard to foresee all possible instances (contexts) when an asset will be accessed or requested. However, given the attributes of the asset the resource domain expert should have a knowledge about the situations governing its use. Each situation is encoded as a predicate[1]. We assign the same criticality label to an asset at all contexts that satisfy the situation predicate. To formalize this notation we define $\mathcal{OB}$, the set of objects in the system, $\mathcal{SU}$, the set of subjects in the system, $\mathcal{RO}$, the set of roles that subjects are allowed to play, $\mathcal{AS} \subset \mathcal{OB}$, the set of assets, $\mathcal{SI}$, the set of situations, and $\mathcal{CO}$, the set of contexts. The function

$$L : \mathcal{AS} \times \mathcal{SI} \rightarrow CL$$

assigns for $o \in \mathcal{AS}$ and $p \in \mathcal{SI}$, the label $l \in CL$. That is, $L(o, p) = l$. For $p \neq p'$, $L(o, p) \neq L(o, p')$. For $c \in CO$ and $p \in SI$ we write $\mathbf{vc}(c, p)$ to denote the validity condition that p is true in context c. So, the asset $o \in AS$ has the label $L(o, p) \in CL$ in context c if $\mathbf{vc}(c, p)$ holds. As the system dynamics changes, contexts might change which in turn might dynamically relabel an asset. To deal with dynamic contexts, the toolkit [7] is integrated into the security architecture shown in Figure 1. Thus, the architecture is *context-aware*.

3.2 Representation of Context-Dependent Access Control Policies

In [1] we have discussed a context-dependent grant-access policy with regard to managing identity of subjects in transaction-based systems. We adapt this approach for the purpose of CPS. Let $\mathcal{AC}$ denote a finite set of actions. We define access policies by functions. The function AS assigns to an individual $s \in \mathcal{SU}$ a set of signed actions, called *access rights*, on an object $o \in OB$. We write $+a \in AS(s, o, c)$, to affirm that the subject s is allowed to perform action $a \in \mathcal{AC}$ on object o in context c, and write $-a \in AS(s, o, c)$ to affirm that the subject s is not permitted to perform action a on the object o. Policies may exist for providing access rights to groups of objects and subjects. The function SG gives for a subject s the groups $SG(s)$ to which the subject s belongs. The function AG assigns to a group $g \in \mathcal{GR}$ a set of rights on an object $o \in OB$. If $+a \in AG(g, o, c)$ then every entity in g is allowed to perform action a on object o in context c. If $-a \in AG(g, o, c)$ then no entity in g is allowed to perform action a on object o in context c. Roles might be defined in the system such that an entity may assume a role in a certain context. Function SR gives for each individual subject $s \in S$, the set $SR(s, c)$ of roles assumed by s in context c.

[1] In general a temporal logic may be used. For simplicity we restrict to predicate logic framework.

The function AR defines for each role $r \in R$ in context c, the set $AR(r, o, c)$ of rights that r has on the object o in that context. We define the grant policy as a function SP, which for a subject s in context c grants or denies access to object o. We use the notation $\mathbf{vc}(c, p)$ to denote that the policy p (written as situation predicate) is valid in context c.

1. [P1:] *s is an individual subject* The subject s is granted to perform the actions explicitly allowed for it on the object o in context c if there exists no policy in context c that overrules that privilege .

$$SP(s, o, c) \ = \ if \ \mathbf{vc}(c, p) \ then \ AS(s, o, c) \ else \ \emptyset$$

2. [P2:] *s has a set of roles but is not a member of a group* The subject s is granted the right to perform an action a on an object o in context c if at least one of the roles in $SR(s) \neq \emptyset$ is authorized to access o and none of them is denied to access o in context c.

$$SP(s, o, c) \ = \ \{+a \mid p_r(a, s, o) \wedge a \in \mathcal{AC} \wedge r \in SR(s)\},$$

where
$p_r(a, s, o) \equiv \mathbf{vc}(c, p) \ \wedge \ + a \ \in \ AR(r, o, c) \ \wedge \ \sim \exists r' \in SR(s) \bullet (-a \in AR(r', o, c))$.

3. [P3:] *s has no roles and belongs to one or more groups* In context c the subject s belonging to the groups in $SG(s)$ is granted to perform an action a on an object o, if at least one of the groups in $SG(s)$ is authorized to access o in context c and none of the groups in $SG(s)$ is denied to access it in context c.

$$SP(s, o, c) \ = \ \{+a \mid p_g(a, s, o) \wedge a \in \mathcal{AC} \wedge g \in SG(s)\},$$

where
$p_g(a, s, o) \equiv \mathbf{vc}(c, p) \ \wedge \ + a \ \in \ AG(g, o, c) \ \wedge \ \sim \exists g' \in SG(s) \bullet (-a \in AG(g', o, c))$.

4. [P4:] *s has a set of roles and belongs to one or more groups* Using the predicates defined in the previous two steps we define

$$SP(s, o, c) \ = \ \{+a \mid +a \in (p_r(a, s, o) \cap p_g(a, s, o)) \wedge r \in SR(s) \wedge g \in SG(s)\}$$

If A and B are arbitrary sets of subjects then the subjects in the set $A \cup B$ has permissions $(\bigcup_{s \in A} SP(s, o, c)) \bigcap (\bigcup_{s \in B} SP(s, o, c))$, and the subjects in the set $A \cap B$ has permissions $(\bigcup_{s \in A} SP(s, o, c)) \bigcup (\bigcup_{s \in B} SP(s, o, c))$.

4 Modeling Context-Awareness

Awareness induces the system elements to become proactive. We say the system is *self-aware* if it knows its full list of active users, assets used, policies in force, and process states. Thus, self-awareness may be characterized by (1) a set of *Users* who are active subjects in the system, (2) a set of active *Assets*, (3) a set of *Permissions*, namely access policy controlling access to assets,

and (4) a statement of *Purpose*, an application-specific key-word defining the set of application specific tasks. Self-awareness is also called internal awareness. The internal awareness is thus a set of contexts built upon the four dimensions $User$, $Asset$, $Permission$, $Purpose$. Let $UC = \{UC_1, \ldots, UC_m\}$ be the set of user categories, and $DC = \{DC_1, \ldots, DC_k\}$ be the set of asset categories which are to be protected. We regard UC_i's and DC_js as dimensions. Let PC denote the purpose dimension and PM denote the $Permissions$ dimension. Assume that the tag set along each UC_i is the set of user names, the tag set along each DC_i can be the set of identifiers to assets, the tag set for PC is $\{Legal, Administrative, Marketing\}$, and references to the grant policies are tags for dimension PM. Contexts with these dimension/tag sets are *Internal Security Contexts* (ISC). Based on the initial state of the system a set of ISC contexts can be constructed. As system evolves, ISC contexts will undergo modifications. A context of type ISC provides awareness information to protect the internal states of the system. An example of ISC context is $[UC_1 : u, DC_2 : j, PC : Communication]$, meaning that client u with role UC_1 is allowed access to the asset referenced by j in category DC_2 for communication purposes.

We say the system is *context-aware* if it knows the potential set of clients seeking to access it, the physical devices with which it has to interact in its environment and their statuses, and information fed to it through a sensor network connected to it. Context-awareness is also called external awareness. From the policies governing the system interaction with its environment external-awareness contexts can be constructed. Such policies includes privacy rules, obligation rules, and other exceptions. We call this context set *External Security Contexts* (ESC). An example of ESC context is $[WHO : Alice, WHAT : file, RLOC : Shanghai, WHY : Medical]$. It is the setting where *Alice* is accessing the resource $file$ from the remote location *Shanghai* for $Medical$ purposes. CPS nodes use context-awareness are sharpening *perception* and *adaptation*. Based upon the recent context-aware information received the system reasons about the appropriate action to be taken in a timely manner.

5 Context-Aware Security Features for CPS

In this section we address *trustworthiness*, *security architecture*, and *secure asset flow* as the three issues for which context-awareness can provide safe solutions. These three are among the many challenges and claims raised in [2].

5.1 Trustworthiness

Cardénas et. al [2] emphasize the importance of trustworthiness notion and trust management schemes for CPS components. They state '*if trustworthiness metric of a component deviates significantly from the trust that is associated with the component, then the component may be regarded as insecure and its contribution toward the operation of CPS may be restricted or discarded.* Trustworthiness

is a system property [5] and needs to be verified by independent trusted authorities. The trustworthiness claims include one or more of the set { *safety features*, *security features*, *reliability features*, *guaranteed availability features*, and *accountability features*} . Accountability feature might reveal the independent authorities (contact numbers/emails) who can be contacted for verifying the trustworthy claims. A trusted authority (TA) has the knowledge and skills to verify these claims and provide a *ranking* (security level acceptance) for the system. So, we assume that a CPS has one or more TAs who will evaluate the system at each CPS site and award a ranking for it. The evaluation procedure is as outlined in the Orange book for *Trusted Computing Systems* except that it is context-dependent. A site may see the ranking of other CPS sites by browsing the directories kept by the TAs. By comparing its ranking with the ranking of other sites, a site might decide to get products and services only with those sites whose security rankings are either equal to or above its ranking. In particular, if site i requests an asset o from a site j in context c then the following must be true:

- $ranking(j, c) \geq ranking(i, c)$
- the asset o is released to site i together with (1) the context-dependent criticality level $L(o, p)$ (determined at site j), (2) the situation predicate p, and (2) the context-dependent grant function $SP(s, o, c)$, $s \in SR$.

The certificate from TA will make explicit the context in which the site was evaluated and the context of validity of the certificate itself. Hence site j must have its local policy adapted in order to use the asset received from site i; Otherwise, security may not be guaranteed for site i.

5.2 Security Architecture

In this section we emphasize the role of context-awareness for security, not specific algorithms for enforcing security. A generic context-aware architecture is shown in Figure 1. The architecture can be easily specialized to build more specific applications. Usually a ring of defenses exist to protect SCADA networking systems. A strong fire-wall protection is built to safeguard the SCADA network from both the internal corporate network and the Internet. Keeping this in mind we have introduced a fire-wall for the security architecture.

The toolkit in Figure 1 has the context space database containing the dimensions and their tag sets. It constructs contexts according to the syntax explained in Section 3, implements context operators [7], and evaluates situations in different contexts. As an example, in the expression $[TIME : d_1, WHO : Alice, WHAT : filetransfer, WHERE : Shanghai, WHEN : d_2, WHY : Auditing] \downarrow \{WHO, WHAT, WHY\}$ the operator $\downarrow$ is selection operator, its left operand is a context and its right operand is a set of contexts. The semantics of selection is 'extract from the context (left operand) the subcontext whose dimensions match those in the set (right operand)'. Thus, the above expression evaluates to $[WHO : Alice, WHAT : filetransfer, WHY : Auditing]$. In general, formal expressions for evaluating situations in context expressions can be

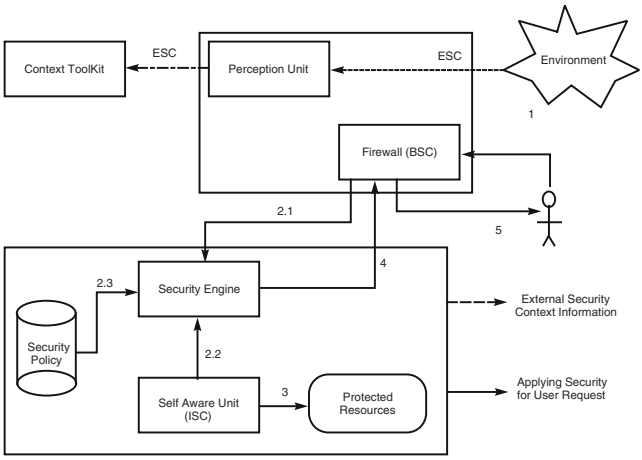

Fig. 1. Generic Architecture for Context-aware Security

written down. As an example, if c_1, c_2 and c_3 are three contexts, X is a set of contexts, and p is a situation predicate, the expression $\mathbf{vc}((c_1 \ominus c_2) \sqcap c_3) \downarrow X, p)$ requires the evaluation of the context from the expression $(c_1 \ominus c_2) \sqcap c_3) \downarrow X$ and then evaluating p in the resulting context. Such operations are necessary for managing mobile contexts and reasoning contextual behavior. The toolkit supports other units, such as 'Security Policy' unit, in the architecture wherever context information is required. In addition, context toolkit and protected assets are loosely coupled so that adding or removing of new assets and modifying their respective access policies can be achieved in a transparent manner. The security policy base is usually pre-defined by system administrators and may change periodically. The perception unit can be either a sensor network that monitor the environment or simply a user interface. The self-aware unit monitors the internal states in the system and maintains a history of state changes. The security engine applies security policies internally to data and operations, and at the fire-wall level for authentication of user requests. The labels shown in the figure roughly indicate the sequence of interactions: [1.] Stimulus from the environment processed by the 'Perception Unit' to construct external contexts. [2.1.] Fire-wall forwards stimulus after applying boundary security context. [2.2.] Security engine collects internal security context information from internal system states. [2.3.] Security engine refers to security policy base. [3.] Security engine applies security policy relevant to security contexts to protected assets. [4.] Security engine sends the result to the environment through fire-wall. [5.] Fire-wall filters the response using boundary security context and forwards the result to the environment.

This architecture can be used to deter and prevent the compromise of assets local to a CPS site, such as data resources, actuators or other physical devices. A collection of mobile sensors can be deployed to detect intruders and take

preventive action. The prudent defensive action is the gradual shutdown of the actuator, when a threat is detected. By preventing external attack on physical devices we are improving *availability* and eliminating *denial of service*. In order to prevent *deception attacks* (prevent false information from being sent from sensors and controllers, which happen when sensors or controllers are themselves compromised) we need to strengthen ESC unit. This is a challenging problem.

5.3 Secure Asset Flow

The security of the asset transferred from one site to another CPS site cannot be guaranteed to be secure unless the network through which it is sent is made secure. Unfortunately, the traditional methods used for network security is not sufficient to assure the networked CPS sites. It is necessary to isolate CPS network from corporate IT networks and the Internet, surround CPS with several defense rings, and further inject it with strict security measures. Under this assumption, we discuss the following flow policy for CPS assets.

When a site j receives a request for asset 0 from a site i, it will validate the certificate of site i and verify the eligibility condition $ranking(j) \geq ranking(i)$. Once these are verified, the asset together with its context-dependent criticality level, and the context-dependent grant function for o both as determined at site j will be sent to site i. The asset may be transferred through a set of CPS sites. That is, $j = j_0, j_1, \ldots, j_k = i$ may be the CPS network path through which the asset is transferred. We call this an asset flow. Assets that flow are the cyber assets, such as data, control information, corporate policies, encryption keys, and protocols. The primary sources of threats to a flow are organizational threats from employees of the organization who manage the different sites on the flow path, control and other assets that enable the flow, and outside attackers. Even the best flow security can be invalidated by a poor people system. The human assets who manage a CPS network will include IT professionals, control engineers, security experts, and many non-experts. They are usually dispersed over different application domain or departments within an organization. People might be careless at best or untrustworthy at best in order that a security breach might arise. Context-awareness might help to prevent some of these security lapses. Some employees may use illegitimate means and violate local policies to access the information which is not legitimately required in their job related tasks. Context-dependent policy enforcements and authorizations will prevent such incidents. Some of the security lapses might be "unintentional" or "accidental". As an example, the information left on the screen of a computer can be seen by another employee who is not authorized to know it. Context-aware motion detection systems can catch such incidents and force remedial actions. Some actions include automatic log-outs whenever the system is left idle, and "warning" employees about their behavioral code. Another kind of threat is that employees who have authorized access to an asset violate the trust instituted in them. This problem is serious in a networked system such as CPS which operates in a decentralized fashion. To minimize this kind of abuse, we had suggested that the asset together with its context-dependent criticality level, and the context-dependent grant function for o as determined at the host site be

transmitted. The security at any site through which the asset passes should enforce its local policies that do not violate the intended access rights imposed by the host site. Another kind of threat is the attack sponsored by collaboration among a group of employees. Towards defending this kind of attack we had given an expression for calculating access rights for arbitrary sets of subjects in Section 3.2. Rotating employees to work in different departments under different contexts, and audit trails may also deter this kind of threat.

We contend that ESC context can be used effectively at every site in the flow path. The node j_r that receives from its predecessor node j_{r-1} not only the asset but also the context information which is the *union* of all context information from site j_0 to site j_{r-1}. That context is the ESC for safe keeping and safely forwarding the asset to the next site in the flow path. Since "purpose" is a ESC context is domain information, the security level clearance of the asset at a site is also domain-dependent. Consequently, the transmission channel (flow path) through which the request is sent must have a security clearance higher than or equal to that assigned for the asset. Moreover, site j_r sending the asset to site j_{r+1} should have the security clearance for sending it, and site j_{r+1} must have the security clearance to receive the asset. Assume that security levels for subjects are modeled by function S we impose three constraints for a secure flow from subject s_1 in site j_r to subject s_2 at site j_{r+1} while sending asset o along a channel σ. It is essential that the situation p for executing each action below should satisfy the context c of the action. That is, $\mathbf{vc}(c, p)$ must be true.

- [secure channel for asset o]: If $L(o, p) \leq L(\sigma, p)$, then in context c the channel σ is secure for asset o in context c.
- [s_1 can write on σ]: If $write+ \in SP(s_1, o, c)$ and $S(s_1, p) \leq L(\sigma, p)$, then the subject s_1 can write o on channel σ.
- [s_2 can read on σ]: If $read+ \in SP(s_2, o, c)$ and $S(s_2, p) \geq L(\sigma, p)$, then the subject s_2 can read o from channel σ.

6 Conclusion

Cardénas et. al [2] have put forth a list of open problems for CPS security, and Weiss [10] eloquently has brought out the security loopholes in existing electric power grids. The two conclusions that we derive from these studies are (1) there exists no study yet in weaving tightly context-awareness and security solutions for studying CPS security, and (2) technological solutions for CPS security, even if they are found, may not be realizable soon because of the non-coordination of efforts between control engineers, IT sector, government organization, and industry partners. Our research effort is mainly directed at technically feasible context-aware solutions for CPS. The security architecture proposed in this paper is for every CPS node. Context-based labeling of critical assets, context-dependent trustworthiness, and context-aware asset flow are ingrained at different levels of this architecture. Thus, CPS network will have a network of context-aware security architectures. Time is a specific instance of context. As

an example the context $[TIME : 10 : 30]$ is the clock time 10 : 30. The predicate $10 < TIME < 11$ evaluates to true in this context. Relative times can be modeled by a context with two dimensions, and time duration can be modeled using the 'directed range operator ⇀' [8].

Acknowledgments. This research is supported by Research Grants from National Natural Science Foundation of China (Project Number 61103029), Natural Science Foundation of Jiangsu Province, China, and Natural Sciences and Engineering Research Council, Canada.

References

1. Alagar, V., Wan, K.: Context Based Enforcement of Authorization for Privacy and Security in Identity Management. In: de Leeuw, E., Fischer-Hübner, S., Tseng, J., Borking, J. (eds.) Policies and Research in Identity Management. IFIP, vol. 261, pp. 25–37. Springer, Boston (2008)
2. Cardénas, A.A., Amin, S., Sastry, S.: Secure Control: Towards Survivable Cyber-Physical Systems. In: 28th International Conference on Distributed Computing Systems Workshops, ICDCS 2008, pp. 495–500 (June 2008)
3. Cyber-Physical Systems: Executive Summary, CPS Steering Group (2008), `http://varma.ece.cmu.edu/summit/CPS-Executive-Summary.pdf`
4. Dey, A.K., Abowd, G.D.: A Conceptual Framework and a Toolkit for Supporting Rapid Prototyping of Context-aware Applications. Human-Computer Interaction 16(2-4), 7–166 (2001)
5. Mohammad, M., Alagar, V.: A Formal Approach for the Specification and Verification of Trustworthy Component-Based Systems. Journal of Systems and Software 84, 77–104 (2011)
6. USA NSF Program Solicitation NSF-08-611 (2008)
7. Wan, K.: Lucx: Lucid Enriched with Context. Ph.d thesis, Concordia University, Montreal, Canada (January 2006)
8. Wan, K.: A Brief History of Context. International Journal of Computer Science Issues 6(2) (November 2009)
9. Winograd, T.: Architectures for Context. Human-Computer Interaction (HCI) 16(2), 401–419 (2001)
10. Weiss, J.M.: Control Systems Cyber Security - The Need for Appropriate Regulations to Assure the Cyber Security of Electric Grid. Testimony before The Committee on Homeland Security (October 2007), `http://chsdemocrats.house.gov/SiteDocuments/20071017164638-60716.pdf`