EM Enforcing Information Flow Properties using Compensating Events

Thoshitha T. Gamage * Bruce M. McMillin*
{ttgdp5@mst.edu, ff@mst.edu}
Department of Computer Science
Intelligent Systems Center & Energy Research and Development Center
Missouri University of Science and Technology, Rolla, MO 65409-0350

Abstract

Deeply embedded infrastructures are pervasive systems that have significant cyber and physical components, interacting with each other in complex ways. These interactions can violate a system's security policy leading to unintended information flow. Execution Monitor (EM) enforceability is the concept of monitoring a system during runtime for any security policy violations and terminating the execution if such violations occur. EM enforceable mechanisms require that the properties being enforced be restricted to safety properties. Information flow properties are considered non-EM enforceable because they can not be defined using safety properties. To bridge this gap, prior work has presented a monitor that predicts future possible events, then evaluates these as safety properties. Unfortunately, in a pervasive system, evaluating future possible events results in a physical, observable, change to the system. What is needed is a physical "undo" operation in which a physical setting can be explored, then undone in a way that no unintended information flow results. This paper presents the concepts of compensating events and a compensating couple which can be used to EM enforce information flow properties in pervasive systems.

1 Introduction

With the advances in information systems and communication networks, it has become very evident that in near future, everything including humans, will be *"plugged-in"* or *"connected"* either directly or indirectly. Although in many aspects this is quite beneficial in our day to day life, the heavy integration of physical systems with cyber systems introduce many security concerns as well. For example, consider a pervasive gas pipeline that transports a commodity (gas). In a management system, a particular distributor is in control of a certain section of the pipeline and is expected to have access to the gas flow information, but only of that specific section. Unfortunately, due to the interconnected nature of pipelines, any cyber interactions within the pipeline result in observable changes in the pipeline flow, leaking information about the cyber interactions.

In another example, consider a smart house - a house controlled and monitored by an automated system for the purpose of user convenience, security and energy efficiency. Here, various aspects of the house such as power, security, and temperature could be controlled either via an in-house computer system or with the use of remote means. Once a smart house has been configured, the occupants of the house will be given certain usage and administrative rights to the system. In doing so, the system is supposed to prevent non-occupants of the house from acquiring various status information (such as power usage level, temperature level, number of occupants present, number of open/closed doors or windows, etc.) of the house. Nevertheless, a person standing outside the house is still capable of deriving certain information about the house just by merely observing it. Here again it can be seen that observable physical changes in a system compromising information security.

Even though a system can be designed to be secure, run time issues in the system can invalidate development time assumptions. As more and more physical systems are integrated with cyber systems, these risks grows exponentially; in the cyber world, systems interact and communicate with other systems, allowing even more indirect security attacks. This is not because of carelessness of design or pure implementation flaws, but simply because such situations were not considered possibilities during the development phase of the system. The primary concept behind *Execution Monitoring (EM)* and EM enforceability is to evaluate security policy violations when a system is *" live, up-and-running "*

*This work was supported in part by NSF MRI award CNS-0420869, CSR award CCF-0614633, ERC award EEC-08212121 and in part by the Missouri S&T Intelligent Systems and Energy Research and Development Centers.

and take appropriate actions.

In this paper, we propose a new approach to EM enforce information flow properties in pervasive systems. In section 2, we present some of the background information and related work in the area of EM enforceability. This is followed up by a discussion on enforcing information flow properties in pervasive systems in section 3. We provide two possible ways of enforcing information flow properties in pervasive systems. In section 4, we introduce our sample pervasive system and the mathematical groundwork required for our proof including a case study of an actual information flow violation. In section 5, we formally define the concept of compensating events and compensating couple. Section 6 presents the compensating automata for pervasive systems followed by a summary and a discussion in section 7.

2 Run Time Evaluation and Enforcement of Security Policies

Assumptions made at development time might lead to security being compromised at runtime. Added to this are the physical characteristics of systems. Some of the reasons for such compromise are [1] hardware failures, specific design considerations and requirements and unforeseen changes in the supporting systems due to interaction with other systems. Execution Monitoring **(EM)** [2] is the concept of monitoring the execution[*] steps of a particular concurrent system at runtime, detection of enforced security policy violations and terminating the execution upon a violation. Security policies are defined for properties. By definition, a **property** [3] is a set of executions. A security policy is yet another way of defining properties; A security policy is a set of executions where each execution in the set satisfies a certain predicate where membership is determined on individual execution basis.

According to Schneider [2], for a certain security policy to be enforced, it needs to be a **safety property**. Safety properties are based on three requirements. In essence, these requirements are **(1)** that each execution - which is in the set defined for the particular property - individually being safe, **(2)** that executions being prefix closed i.e. the execution needs to maintain safety throughout each step and lastly, **(3)** that any occurrence of violation has a precisely identifiable point.

There are security properties in existence which can not be defined using the definition of a safety property. Since security policies are only defined for safety properties, a certain range of interesting security properties are limited from being enforced. As pointed out by [4] and [5], information flow properties can not be expressed using the Alpern and Schneider framework [3]; these properties can not be enforced with the security automata described by Schneider. Information flow properties[†] includes *noninterference* [6], *noninference* [7], *nondeducibility* [8], etc. Also, the prefix closed feature is a strong requirement which does not allow making predictions, or in other words, looking into the future to see how the execution will continue to behave. This is not a simple limitation; an execution which may seem to violate the policy at present, might actually end up being *safe* and complete its task if allowed to continue. Specially, when it comes to information flow properties, imposing such a hard limit on executions may end up not achieving the desired final state in any of the executions.

3 Enforcing Information Flow Properties

Information flow properties are not defined over a set of executions [4] rather over sets of execution sets. This leads to a conclusion that the decision to terminate an execution can not be purely based on a detected violation of a single execution. In order to EM enforce information flow properties, the existing EM enforceable mechanisms needs to be extended. Two possibilities are,

1. Providing a prediction mechanism for the future steps
2. Allow the execution to continue and taking counter measures if violations occur

3.1 Predicting the future

The first approach is to extend the existing security automata with a prediction mechanism. The purpose of the prediction mechanism is to enumerate through all possible future executions paths and determine the viable(safe) next state(s). If no safe state exist or the execution is not going to continue on to one of the safe states, the policy is considered violated and the the execution is terminated. Using this concept, Nagatou and Watanabe [9] have presented a runtime covert channel detection mechanism. In doing so, they extended the basic security automata for EM enforceable mechanisms by predicting the future. The underlying concept of their extended security automata is the **unwinding theorem**. The unwinding theorem allows appending an individual safe state to an sequence of states which has already been determined as secure. An execution monitor explores possible next states of the security automata. However, for the physical portion of the system, exploring the next state can lead to a side effect of observable actions that cannot be undone if this next state is found to be unsafe.

[*]An execution is termed as an infinite sequence of states.[3, 2]

[†]These properties are also termed as “possibilistic security properties [4]

3.2 Counter Measures

As pointed out earlier, if information flow properties are to be EM enforced, it is not reasonable to terminate the execution whenever a violation is detected; the information in question has already flowed. If a prediction mechanism is not present, the other alternative is to allow execution to continue and take appropriate counter actions. These counter actions could be either to rollback to a previous safe state and resume the execution from that point onwards or replace the rest of the execution with a pre-determined safe sequence [‡] or introduce counter events (named compensating events here afterwards). The difficulty of rolling back or expressing counter events in the physical part of the system is addressed in the next section.

4 Compensating Events in Pervasive Systems

In this paper, our focus is showing that the concept of compensating events can be used in enforcing information flow properties. In particular, we are interested in enforcing these in pervasive systems - systems with both cyber components and physical components. We consider the gas distribution network as our example pervasive system. Figure 1 shows a particular subsection of a gas distribution network. Here, three different distributors A, B and C control three sections of the pipeline. The pipeline represents the physical layer of the system. The Remote Terminal Unit (RTI) acts as the interface between the physical layer and the cyber layer. The RTI is used to acquire the state of gas (*read operation*) as well as to affect changes to the flow (*write operation*). A human operator working in front of a Human Machine Interface (HMI) can control a particular set of RTIs connected to that HMI. The HMI acts as the cyber component of the system while RTI stands the middle ground. Similar applications can be found in water and power distribution networks [10] as well.

In the pipeline system, as noted in the introduction, changes in one part of the system are reflected in other parts of the system. From basic laws of fluid flow it can be seen that, at steady state,

$$f_A = f_B + f_C$$

where f_s is the amount of gas flow in pipe section s. Suppose now B lowers the flow in section B to $f_B - f'$ by changing valve positions at RTU_B. In order to maintain the overall gas flow stability of the pipeline network, either f_A and/or f_C will have to be re-adjusted to new stable values.

[‡]This actually seems like a basic violation of the definition of a safety property. But, information flow properties can not directly be defined using safety/liveness [3] thus, a single trace level approximation defined for safety does not apply here.

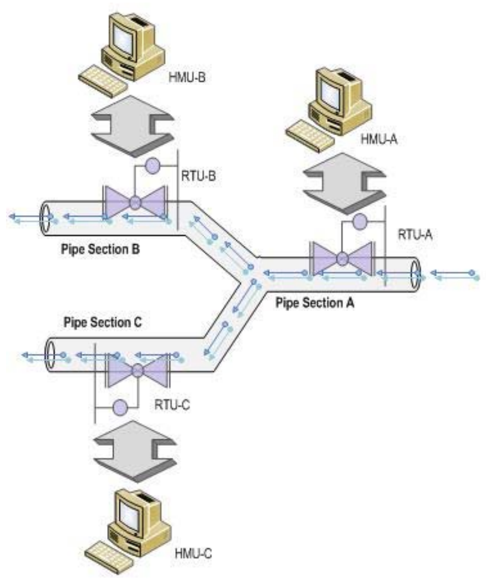

Figure 1. A Subsection of the Gas Distributing Network with different distributors controlling different sections of the pipeline

The aggregated change adopted by the rest of the network Δf could be any of the followings.

$$\begin{aligned} \Delta f &= \{f'^{\downarrow} f_A\} \; or \\ &= \{f'^{\uparrow} f_C\} \; or \\ &= \{\alpha^{\downarrow} f_A \; and \; \beta^{\uparrow} f_C : \mid \alpha - \beta \mid = f'\} \end{aligned}$$

where $\uparrow$ and $\downarrow$ indicate increase or decrease in flow, respectively. Note that there are other similar possible combinations for the last case of Δf. In order to formally define compensation, we introduce the following definitions based on definitions of a *state machine*[11] and *state space*[4] found in literature.

Definition (State Machine). *A state machine consists of $S = \{s_1, s_2, \ldots\}$ set of subjects, $Q = \{q_1, q_2, \ldots\}$ set of system states, $I = \{i_1, i_2, \ldots\}$ set of inputs and $O = \{o_1, o_2, \ldots\}$ set of system outputs. Associated with each system state q_i is a set of state variables. An element of output $o_i \in O$ represents a snapshot of the all state variables at a particular instance. Changes to the state variables occur due to a subset of inputs $C \subseteq I$ called state transition commands. Thus, a state transition command will transition the state machine to a new state with a new(or modified) set of state variables. In addition, there are input commands which do not affect state changes, denoted by the set $Z \subseteq I$. A command is always triggered by a subject who is in a certain protection domain(defined below).*

Corresponding to the pipeline model presented above, a command z represents a *read operation* while a command c represents a *write operation*. The state transition function $T : C \times Q \to Q$ defines executing command $c_i \in C$ over the state q_j which transits the system to a new state q_k. A sequence of state transition commands C^* would produce a sequence of state transitions $T^* : C^* \times Q \to Q$. With each state transition, a new set of state variables is added to O which is denoted by the output function $P : C \times Q \to O$. Correspondingly, C^* would produce a sequence of outputs $P^* : C^* \times Q \to O$.

Definition (Trace). *A trace σ is a sequence of inputs. This general definition helps defining a history of events for a state machine over a certain range. The range here could be a certain time period or system steps. Since $C, Z \in I$, an element $i \in \sigma$ could cause either a state transition or merely an output of state variables. Thus, an element of σ provides an abstract view of commands and outputs of the state machine at a particular instance. The trace space set Σ is the set of all possible traces for the state machine.*

We introduce $D = \{d_1, d_2, \ldots\}$ which is a set of protection domains. Associated with each protection domain is a set of subjects called a group. Without loss of generality, the partial ordering $d_{G'} < d_G$ denotes that information flow is prohibited from group $G \subseteq S$ to another distinct group $G' \subseteq S$. The domain in which a particular input is executed is represented as $dom(i)$.

Definition (Projection). *A projection on a trace denotes the sequence of output a group G is authorized to observe. This is formally represented as $proj(G, \sigma, q_i)$ where q_i is the originating state for the trace. By definition a projection is a refined sequence of output denoted $proj(G, \sigma, q_i) \in P^*$*

Definition (Purge). *A purge on a trace will remove all elements in the trace that corresponds to the purging condition. For example, $\pi_G(\sigma)$ will remove elements in σ corresponding to the group G. The purging could also be done on certain set of inputs. For example, $\pi_{\bar{C}}(\sigma)$ will remove all inputs which are not state transition commands. What this will leave a sequence of state transition commands so $\pi_{\bar{C}}(\sigma) \in T^*$.*

4.1 Case Study: Violations of Information Flow Properties

To facilitate developing a basis for *compensation*, we present the following case study. We use the same example presented in Figure 1. Here, the subjects $S = \{A, B, C\}$ are the distributors and the input $I = \{read_s, write_s^{\uparrow}, write_s^{\downarrow}\}$ are the possible commands on RTI_s where $s \in S$. Suppose the system is in a stable state q_k when input $write_B^{\downarrow}(f')$ occurs. As seen from Δf, there are multiple ways of achieving next stable state but, for the simplicity of explanation, let's assume the following two traces.

$$\sigma_i = \{read_C(), write_B^{\downarrow}(f'), read_C(), write_A^{\downarrow}(f'), read_C()\}$$
$$\sigma_j = \{read_C(), write_B^{\downarrow}(f'), write_A^{\downarrow}(f'), read_C(), read_C()\}$$

Define $G' = \{C\}$ and $G = \{A, B\}$ and $d_{G'} < d_G$, meaning information flow is prohibited from G to G'. For noninterference to hold, the outputs a certain group of subjects can see should corresponds to inputs allowed to see [11]. Mathematically, this is stated as

$$proj(G', \sigma_i, q_k) = proj(G', \pi_G(\sigma_i), q_k)$$

Let's take two projections for G'.

$$proj(G', \pi_G(\sigma_i), q_k) = f_C, (1 + f')f_C, f_C \qquad (1)$$
$$proj(G', \pi_G(\sigma_j), q_k) = f_C, f_C, f_C \qquad (2)$$

Although with accordance to the definition of noninterference, subjects in G' are not suppose to know about the actions of G, the projection on σ_i provides enough information to derive that some change has occurred in the system. Also, actions of G has affected the output for G' leading to interference. If there weren't any interference, (1) should equal f_C, f_C, f_C. But the purged projection of σ_i does not agree the equation. This contradicts with the information flow restriction between the two groups. At the same time, looking at (2), the group G' is not able to derive any information. Here, although there were changes in the system, it is not visible, thus, it seems that there weren't any change. What this means is, by ordering the events in a certain way, it is still possible to keep a secret from disseminating outside a protection domain.

5 Event Compensation

Our argument is, in order to enforce information flow properties in pervasive systems, the two aspects (the cyber aspect and the physical aspect) need to be treated together and just considering one aspect alone is neither sufficient nor reasonable. By looking at (1) and (2) it can be seen that at the end of both traces, the value of f_C has returned to the original value as if nothing has happened. This was possible because the effect of input $write_B^{\downarrow}(f')$ on the pipeline has been compensated by the effect of $write_A^{\downarrow}(f')$. Furthermore, both these commands were issued by members of the same group. Lets assume that at state q_k, the system is *information flow safe* and there exists a trace σ_k which represents a history of *information flow safe* sequence of state transition commands on the system up until the stable state. This can be formally defined as,

$$c_k = \pi_{\bar{C}}(\sigma_k) : c_k \in T^*$$
$$proj(G', c_k, q_0) = proj(G', \pi_G(c_k), q_0)$$

where q_0 is the start state. We only consider state transition commands here since non-state transition commands such as read operations does not change the state variables. We define *compensating couple* as two consecutive compensating state transition commands both issued by subjects in the same protection domain. Let's consider two state transition commands c_{s_1}, c'_{s_2} by subjects s_1, s_2. If c_k was extended to $c_{k+1} = c_k + c_{s_1} + c'_{s_2}$ by appending a compensating couple c_{s_1}, c'_{s_2}, it can be seen that for both $s_1, s_2 \in G$ and $s_1, s_2 \in G'$

$$proj(G', c_{k+1}, q_0) = proj(G', \pi_G(c_{k+1}), q_0)$$

which preserves the definition of noninterference. At the same time, subjects in G' are unaware of any event occurrences in G. This in turn is a preservation of *noninference* security. In a noninference secure system, low level subjects are left in doubt as to whether or not any high level events have occurred. With corresponds to our model system, the objective would be to hide the collaborate actions of A and B from C. Thus, by taking $c_{s_1} \equiv write^{\downarrow}_{B}(f')$ and $c_{s_2} \equiv write^{\downarrow}_{A}(f')$, the already *information flow safe* state transition command sequence c_k is extended to c_{k+1} without violating *noninterference*. Event compensation disallows certain traces from continuing thus, all the allowed traces preserve the above noninference property. Further, a certain compensated trace after purging high level events is still a valid(legal) trace for the system. Formally this can be stated as,

$$\forall \sigma \in \Sigma : \pi_G(\sigma) \in \Sigma$$

From the above analysis, it can be stated that *noninterference* as well as *noninference* can be preserved with *event compensation*. As a result, certain weaker information flow properties (compared to above) such as *nondeducibility* is also preserved by the same concept.

In a purely cyber system, a compensating couple would represent ***event undo*** and in a pure physical world, this represents ***event complement***. In a pervasive system two subjects would first establish a secure connection and exchange information regarding a event undo. Once an agreement is reached, both parties would ensure that a compensating couple is executed. This will ensure other subjects outside the protection domain are kept unaware of the changes. At most, a third party may be able to observe a sudden pulse as shown in Figure 2, but this kind of a discrete pulse could arise due to other natural causes as well[§]. The objective of the two communicating parties would be to make the period of vulnerability as minimum as possible by making a pre-compensation agreement and a commitment for action.

[§]an air bubble or a clog in the pipeline, lighting striking a power transmission line etc.

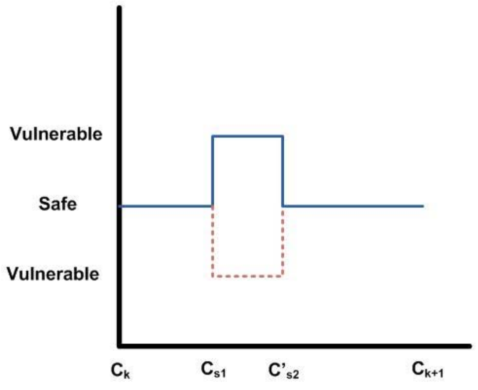

Figure 2. A Short Discrete Vulnerable Period in a Information Flow Secure Environment

In active pervasive systems, event compensation does make sense because, commands that affect the equilibrium of the system (write operations) need to be follow up with other events to maintain stability. That is, an isolated state transition command executed on the system is sure to break the equilibrium of the system and potentially leak information on a state change. For example, if distributor B is required to lower its gas flow due to some reason, this has to be first conveyed to the main distributor A, if B wants to keep the secret from leaking to other competitive distributors like C. Compensating couples help maintain stability while making sure unauthorized information dissemination is prevented outside the protection domain of execution. In between the execution of the compensating couple, the system reaches a non-stable(vulnerable) state (see Figure 2) and once this occurs, the security automata iterates through all possible next state transition commands (similar to what is shown in Δf) in search of a matching compensating event (the event which would makeup a compensating couple). On top of this matching command search is the actual cyber communication and the agreement for compensation occurring between two parties.

The compensating couple can also be viewed as a ***request-reply*** message passing between two subjects. Using the communication network, s_i *requests* s_j that he intends to execute a state transition command. s_j could decide either to acknowledge(reply) or ignore the request. Once an agreement has been made, s_i could issue the first command of the compensating couple while this will be followed up by the matching command by s_j. If ignored, the originating party could wait for a certain pre-defined response time and make a re-attempt.

6 Monitoring for Compensation

The security automata presented in [2] and the extended security automata in [9] are both based on executions being *prefix closed*. *Prefix closed*-ness does not directly apply for *compensation* as the first command of a compensating couple leads the state machine into a vulnerable state. Nevertheless, the objective is, once the state machine reaches a non-safe state, to make sure that the next command preserves compensation and leads the state machine back to a stable(safe) state. What compensation provides is an extended predicate for selecting the next state.

For the same state machine defined previously, we introduce the *compensation automata* as follows. The state space Q is divided into two sets. $W \subseteq Q$ is the set of stable(safe) states and $V \subseteq Q$ is the set of vulnerable(unsafe) states.

Definition (Compensation Automata). *The compensation automata consists of 5-tuples* (Q, Q_0, I', δ, W) *where,*

- *Q is a set automaton states*
- *Q_0 is a set of initial states for the automaton $Q_0 \subseteq Q$*
- *I' is a set of input symbols of the form (c_{i-1}, c_i) : $(c_{i-1}, c_i \in C)$*
- *δ is the a state transition function $\delta : Q \times I' \rightarrow 2^Q$ specified with predicates*
- *W is a set of final states $W \subseteq Q$*

The input symbol (c_{i-1}, c_i) is the last state transition command and the next state transition command under the read head. Initially, $c_{i-1} = \lambda$ meaning, at the very start, there is no input command. It is assumed that due to c_{i-1}, the state machine has already transit to $q_i \in Q$. if $q_i \in W$, c_i is the ***reply*** part of an earlier compensating couple. On the other hand, if $q_i \in V$, c_i denotes ***request*** part of a new compensating couple.

The automaton starts in $q_0 \in Q_0$ and then changes to the next state $q_j \in Q'$ based on the input symbol. The set of all possible next states are given as,

$$\bigcup_{q \in Q'} \delta(q, c_i)$$

δ is defined with a set of predicates. If $q_0 \in W$, the predicate is comparatively simple. The latest executed state transition command part c_{i-1} of the input symbol is ignored and the next state q_j is accepted with the condition that two subjects have agreed to execute a compensating couple. Formally, this can be represented as,

$$\{q_j \mid q_i \in Q' \wedge (\textbf{agreed} := true)\}$$

here **agreed** is pre-compensation agreement between two subjects to honor a compensating couple.

If $q_0 \in V$, the next possible set of states are further refined. Here, one half of the compensating couple has already been issued and the system has reached a non-safe state. The predicate must ensure to maintain the compensation. This is formally represented as,

$$\{q_j \mid q_i \in Q' \wedge q_i \in W \wedge (dom(c_{i-1}) \equiv dom(c_i)) \\ \wedge (c_i - c_{i-1} = \langle\rangle)\}$$

$c_i - c_{i-1} = \langle\rangle$ is the effect of compensation. Here, the next command c_i needs to lead the state machine to a stable state as well as compensate for the previous state transition command. In addition, the domain in which the next command is going to be executed needs to be equivalent to the domain of the previous command as well. The domain equivalency does not necessarily has to be a requirement of two commands issued in the same domain but, at least they should be compatible with each other. Notice here, we are not considering the pre-compensation agreement. Chances are, one party in the original agreement might have decided to pull-out of the agreement; In such a situation, the originating party needs to present an anti-transition command which is equivalent to the matching compensation command promised by the second party.

If Q' is not empty, the command sequence can be accepted and a state transition is possible. The validated transition command c_i can be proposed[¶] to be executed in the actual physical system. If c_i is leaving a non-safe state, the state machine ensures it reaches a safe state in the next step and information flow between undesired protection domains is prevented.

7 Summary and Discussion

In this paper, we have shown that the concept of *compensating events* and *compensating couple* can be used to make information flow properties being enforced in pervasive systems. The concept of compensation relies on secure communication between intended parties in pre-compensation period as well as the total commitment of both parties. Although some information can be leaked to unauthorized users, we argue that the period of vulnerability can be minimized. Added to the fact is that, in actual physical systems, such brief changes could also arise due to many other natural and physical causes as well so, as information flow is concerned, it is still possible to maintain information flow security properties.

This concept would appear to be applicable to other transportation networks such as electric power or water flow, but not applicable to ground or air vehicle movements.

[¶]similar proposals are made in [2] and in [9]

The reason is that the latter two infrastructures contain a sense of direction and it is physically infeasible to ”back up” a vehicle to undo an action.

References

[1] C. Serban, “Run time security evaluation for distributed applications,” Ph.D. dissertation, University of Missouri-Rolla, 1996.

[2] F. B. Schneider, “Enforceable security policies,” *ACM Transactions on Information and System Security*, vol. 3, no. 1, pp. 30–50, 2000.

[3] B. Alpern and F. B. Schneider, “Defining liveness,” Ithaca, NY, USA, Tech. Rep., 1984.

[4] J. McLean, “A general theory of composition for trace sets closed under selective interleaving functions,” in *Proc. IEEE Symposium on Research in Security and Privacy*, 1994, pp. 79–93.

[5] A. Zakinthinos and E. S. Lee, “A general theory of security properties,” in *Proceedings of the 18th IEEE Computer Society Symposium on Research in Security and Privacy*, 1997.

[6] J. Goguen and J.Meseguer, “Security policies and security models,” in *Proceedings of the 1982 IEEE Symposium on Security and Privacy*, 1982.

[7] C. O’Halloran, “A calculus of information flow,” in *Proceedings of the European Symposium on Research in Computer Security*, Toulouse, France, 1990.

[8] D. Sutherland, “A model of information,” in *Proceedings of the 9th National Computer Security Conference*, 1986.

[9] N. Nagatou and T. Watanabe, “Run-time detection of covert channels,” in *ARES ’06: Proceedings of the First International Conference on Availability, Reliability and Security*, 2006, pp. 577–584.

[10] H. Tang and B. McMillin, “Analysis of the security of information flow in the advanced electric power grid using flexible alternating current transmission system (facts),” in *Critical Infrastructure Protection*. Springer, 2008, pp. 43–56.

[11] M. Bishop, *Computer Security: Art and Science*. Addison Wesley, 2003, ch. 8, pp. 191–197.