A Security Policy Framework for eEnabled Fleets and Airports

Mirko Montanari, Roy H. Campbell
University of Illinois at Urbana-Champaign
{mmontan2, rhc}@illinois.edu

Krishna Sampigethaya, Mingyan Li
Boeing Research & Development
{radhakrishna.g.sampigethaya,mingyan.li}@boeing.com

Abstract—The future airport is predicted to be a highly net-centric system-of-systems with advanced networking and wireless technology to accommodate the "eEnabled aircraft," enhanced surface area operations, as well as growing business and societal demands. In this paper, we present a classification of security policies that need to be enforced in such modern airport systems. We propose a distributed architecture for policy-compliance monitoring that enables runtime verification of compliance in the multi-organization environments typical of large-scale infrastructure systems. Compared to current solutions, our monitoring architecture allows each organization to acquire independently information about the state of the infrastructure while respecting integrity, confidentiality, and separation-of-duty constraints that arise because of the interaction between parts of the infrastructure managed by different organizations[1][2].

TABLE OF CONTENTS

1. INTRODUCTION

A recent vision in commercial aviation is the "e-Enabled aircraft" that operates as an intelligent, mobile node in the Internet [1]. Safety, security, environmental, efficiency and economic benefits are provided to airlines, crew, ground personnel, passengers and business stakeholders. The performance of future eEnabled fleets, however, depends on the correct operation of network services provided by airport infrastructure [2]. As the complexity of airport infrastructure increases and changes become more frequent, manual control and operation become ineffective. Unintended or malicious changes in the configuration of a single system could introduce errors in the overall infrastructure configuration that might remain invisible and degrade or disrupt operations (e.g., [3]). Hence, for preserving performance gains of e-Enabling, such errors must be timely detected to prevent systems from operating in insecure or inefficient states.

In this paper, we focus on the problem of monitoring airport infrastructure operations to detect policy violations. Policies provide an efficient way to manage the operation of the airport infrastructure: each policy describes a portion of the correct operation state. When the system is operating outside the states specified by the policy, the system is potentially exposed to security problems and inefficiencies that should be corrected quickly. For example, a policy may mandate that aircraft must land (i.e., weight-on-wheels) and then access a remote airline system on the Internet for software update. If the aircraft accesses airline system before landing, then unwarranted aircraft safety concerns emerge about potentially malicious software updates or potential exposure of flight-critical systems to unauthorized external access. On the other hand, if the aircraft does not access the airline system even after landing, the error condition may create an undesirable delay in gate turn-around time for software refresh. Another policy may specify that maintenance devices must be disconnected from Internet when accessing airplane systems to avoid acting as unintentional stepping-stones for threats. While a dedicated monitoring infrastructure could be developed for each policy, such an approach may prove expensive and not scalable given size and frequency of policy modifications.

The proposed online policy monitoring system for airport infrastructure addresses the scalability problem by providing a general framework for acquiring information about infrastructure state at runtime and checking if the current situation is in violation of policies that govern infrastructure. However, since systems in airport infrastructure are typically owned and controlled by multiple organizations, such a monitoring system is faced with several challenges in the design of the architecture and in the selection of the devices to use for acquiring information. First, each organization must enforce policies independently. Second, the validation of a policy might require acquiring information about parts of the infrastructure managed by a different organization. This creates integrity and confidentiality problems, as business conflicts between competing organizations might create

[1] 978-1-4244-7351-9/11/$26.00 ©2011 IEEE
[2] IEEEAC paper#1377, Version 5, Updated 2011:01:11

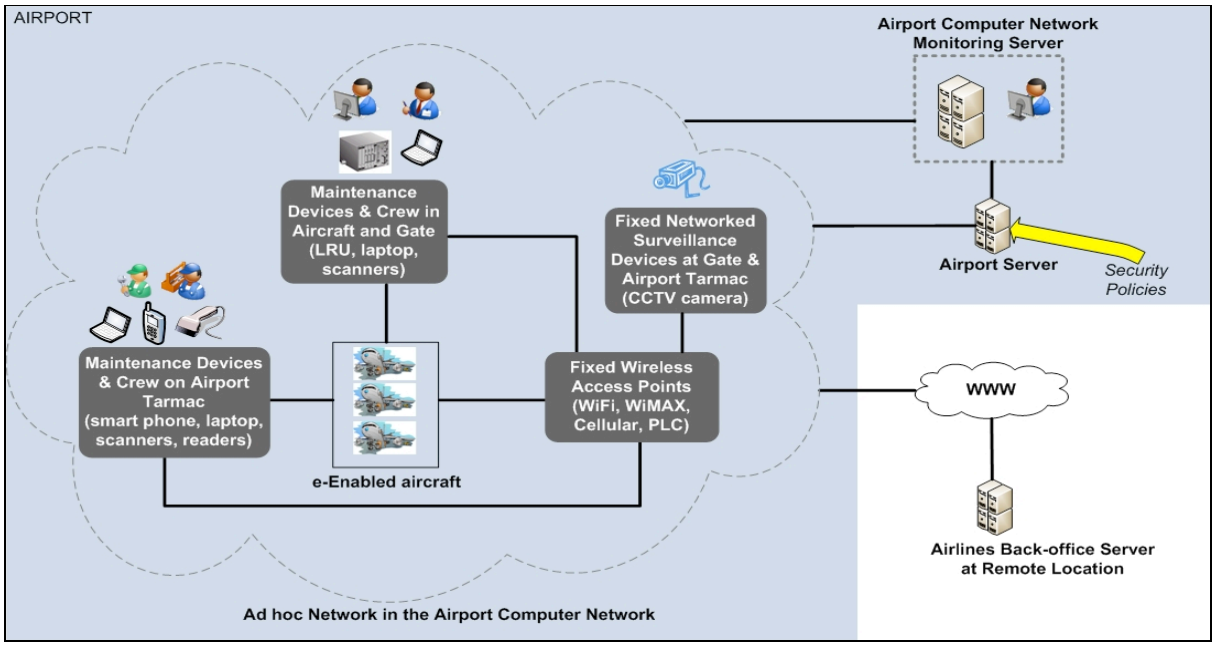

Figure 1: Architecture of the infrastructure of a future airport system.

reticence in sharing information or in relying on external information for analysis. Third, to protect against sabotage, malicious employees, and device errors, critical information for the analysis should be acquired redundantly and in ways that do not allow a single entity to corrupt all information sources (i.e., separation-of-duty constraints).

The contribution of this paper is two-fold. First, we present a classification of security policies for monitoring interactions between e-Enabled fleets, maintenance personnel and airport infrastructure. To our best knowledge, it is the first paper to establish the security policy framework for future eEnabled fleets and airports. Second, we present a system architecture that allows each organization to monitor the infrastructure independently. Our architecture is based on an algorithm that enables each organization to collect efficiently state information while respecting integrity, confidentiality, and separation-of-duty constraints that need to be enforced for the deployment of such monitoring in a multi-organization scenario.

The rest of the paper is organized as follows. Section 2 provides an overview of related work in the area of policy verification. Section 3 describes the infrastructure of modern airports considered and the type of policies that need to be defined in such environments. Section 4 provides a framework for formalizing such policies. Section 5 addresses the problem of multi-organization validation of policies. Section 6 describes our experimental results, and Section 7 concludes our work.

2. Related Work

Airport environment policy definitions have encompassed rules that regulate the use of runways by different types of aircraft [4]. In this work we provide a more general framework for the definition of policies that integrate the network infrastructure of the airport.

The problem of acquiring information about network infrastructure has been previously addressed by systems such as NetQuery [5] and standards such as WBEM [6]. However, none of the previous approaches is able to automatically deal with complex multi-organization constraints such as replication and separation of duty.

Other work addressed the problem of detecting potential security problems present in a system configuration. Nessus [7], TVA [8], and Mulval [9] are systems proposed for detecting the security consequences of erroneous configuration of infrastructure. The focus of this paper is on how the information about the state of the infrastructure can be acquired so that it can be analyzed with the use of such techniques.

3. e-Enabled Fleets and Airports

The network infrastructure of airports supports several applications, from interconnecting devices at check-in desks and gates to supporting the communication between aircraft and airline systems. In this work, we focus on the interactions between e-Enabled fleets, maintenance personnel, and airport infrastructure services. Figure 1 provides an abstract view of the system architecture that we consider representative of the e-Enabled airline system

architecture at airports [10]. To support the maintenance of the e-Enabled fleet, four classes of devices interact in the airport infrastructure:

(1) *Aircraft Hardware Maintenance:* Portable devices located on the tarmac that access aircraft systems to perform net-enabled operations (e.g., hardware inspection/repair, sensor readings). These typically belong to organizations that are leased or owned by airlines.

(2) *Connectivity:* Fixed wireless/wired network access points (e.g., WiFi, WiMAX, cellular, and power line) that interconnect e-Enabled aircraft with off-board systems on the Internet. These typically are owned by third party service providers.

(3) *Airline Maintenance Applications:* Portable devices that are used in the cabin and gate to communicate directly with aircraft systems for airline maintenance applications (e.g., software update, cabin access). These are typically owned/leased by the airlines.

(4) *Tarmac Security Monitoring:* Physical security services provided by CCTV camera, automatic door locks, and emergency devices. These are typically owned by the airport authority.

For supporting maintenance, we consider crew devices that (i) maintain the aircraft RFID-tagged hardware; (ii) manage software, and multimedia; and (iii) manage gates and control access to the aircraft cabins. Additionally, our system model considers other IT infrastructure that interacts with these devices. We include the interfaces with airline back-office servers that reside remotely and provide information, software and multimedia for the airline fleet. Security monitoring includes all physical security devices such as cameras or devices that enforce crew access control in restricted areas of / around the aircraft. Such devices communicate using the network infrastructure provided by the airport.

The online validation of compliance requires a device to share information about its state with the rest of the infrastructure. We assume that each device allows access to a portion of the system's state that it can "observe", i.e., that it can acquire because of its function. In this paper we focus on which information should be shared and not on how the information is shared by each device, as there are existing technologies through which devices can share information about the state. Network management systems, such as WBEM [6], allow information about the current state of the device to be queried from the network. Alternatively, secure introspection techniques [11, 12] can be used to monitor the state of a device in a way that is resilient to compromises. Moreover, it is possible for developers to integrate these capabilities in their software and allow authorized third party to obtain information about the system's state through application-dependent protocols.

We consider an adversary whose objective is lowering the performance gains of the e-Enabled aircrafts and airport systems by exploiting vulnerabilities in the policy monitoring system. The adversary attempts to provide false state information to the policy monitoring system so that policy violations are not detected (i.e., false negative), or false policy violation notifications are generated (i.e., false positive). The adversary can compromise a limited number of devices in the infrastructure, i.e., using a set of valid credentials (e.g., malicious insider), or by exploiting vulnerabilities in the devices. We assume that the adversary is not able to compromise a majority of devices that are part of the same redundant set used for policy validation (i.e., the same vulnerability is not present on the majority of devices that provide a critical piece of information about the state). For non-critical policies, however, we assume devices report their state information correctly, thereby considering threats only from unintentional misconfigurations and failures.

3.1. SECURITY POLICIES FOR E-ENABLING

The interactions between airport infrastructure, maintenance devices, and e-Enabled aircraft fleets create a complex system composed of hundreds of elements. This infrastructure needs to operate correctly to provide the services that maintain the airport and keep the aircraft operational. To enable a synergetic interaction between the different parts and to establish security requirements, organizations use security policies that define the correct configuration of each component and define the actions that are permitted by each user or device. While compliance to policies cannot guarantee perfect security, security policies provide a basic level of assurance against attacks that would be avoidable had proper security measures been taken.

For example, a security policy for e-Enabled aircraft might mandate that they must communicate with the ground systems to check for new updates, but only upon landing. Not performing this check before landing could be potentially a sign of other malfunctioning. It could potentially introduce malicious updates or delay the application of important updates. Similarly, a security policy might specify that maintenance devices be authorized to access the aircraft systems only if physically located on the runway near the plane.

In the definition of security policies for airports, we can classify policies in several classes based on the objective as follows:

(1) *Safety Related:* These are used to ensure safe and regulated operation of aircraft.

(2) *Access Control Related:* These are used to prevent unauthorized crew and device access to aircraft systems.

(3) *Business Related:* These are used to securely minimize operational costs of aircraft and ensure quality of network connections.

(4) *Airport Operation Related:* These are used to securely enable efficient allocation and ensure 24/7 availability of resources at the airport.

Policies can be defined on each type of service provided by aircraft or ground systems.

The definition of proper policies for the operation of the infrastructure is critically important. Even if policies are currently defined by several organizations for different parts of the airport infrastructure, policies are yet to be fully defined for regulating secure interactions between airport ground systems and e-Enabled aircraft.

Appendix A provides several examples the type of policies that need to be defined in each policy class. Note that several of these requirements are theoretical and formulated for use only in our research. For the rest of the paper we focus on three examples as representative policies for the different policy classes:

(2) Aircraft accessing a "ground-only" airline application must be in weight-on-wheels condition.

(5) Maintenance device must be disconnected from the Internet when accessing aircraft systems.

(11) Aircraft must automatically choose the most cost-effective wireless data link available that satisfies the minimum bandwidth requirement.

Infrastructure systems have mechanisms for enforcing the conditions specified in the policies. However, for multiple reasons the enforcement of policies might fail: software errors, hardware failures, or incorrect configurations might allow the system to reach a state that violates the policy. In this case, it is important to have a system that is able to detect such violations and notify the situation for rectification. Additionally, reliable logging of violations is essential to provide evidence of compliance to policies. To address these issues, we develop automatic methods for identifying when the infrastructure operates outside the conditions specified by the policy, while the process of acquiring information about the infrastructure state respects the confidentiality and integrity requirements posed by each organization.

4. FORMALIZATION OF POLICIES

Automatic detection of policy violations is impossible without knowing exactly which situations are considered policy compliant and which are not. To specify policies without ambiguities, it is necessary to express them in a formal language. Given its flexibility and expressiveness, we use the Datalog language [13]. This language has been used in previous work for the specification of access control policies [14] and security of network infrastructure [9]. In particular, we represent the state of the system using the RDF language [15] and we represent policies using inference rules [16].

Using RDF, each entity in the system is identified by a unique string called URI. For example, we identify the first runway of the SEATAC airport using *http://www.portseattle.org/seatac/run1*. Note that the URI is interpreted as a string and it is not required to identify an actual webpage. For simplifying notation in the rest of the paper we represent resources with strings starting with a lower case letter. The runway of the example is identified with seatac_run$_1$. Resources have a type that is organized in a class hierarchy. The state of the system is represented as a set of statements. Each statement represents a "fact" that it is true in the state of the system and it is expressed as a 3-tuple. For example, we can represent the fact that an aircraft landed on runway 1 using the notation *(aircraft$_1$, landed, seatac_run$_1$)*. In this statement, the first element (*aircraft$_1$*) is a resource and represents the *subject* of the statement (in this case the tail number that identifies an aircraft); the last element (*seatac_run$_1$*) is a resource and it is called *object* of the statement. The second element (*landed*) is called *predicate* and represents the type of relation between the subject and the object (in this case that the aircraft *aircraft$_1$* touched ground on runway *seatac_run$_1$*). Unique URI strings also identify predicates. A timestamp is associated to each statement to identify the time at which the statement is generated.

Policies are expressed as rules. A policy can be interpreted as *IF <condition> THEN <consequence>*. The condition is called *rule body*, and the consequence is called *rule head* and it is written as <body> → <head>. The body and the head of the rule are composed of *statement patterns*. A statement pattern is a 3-tuple similar to a statement, but it can have variables in the subject and object position. Variables are identified with an uppercase letter. For example, *(A, landed, seatac_run$_1$)* is a statement pattern. A statement pattern can *match* a statement if there is a substitution of variables that make the two statement equals. In our example, the substitution *A/aircraft$_1$* makes the statement pattern equals to the statement *(aircraft$_1$, landed, seatac_run$_1$)*.

The body is composed of a conjunction of statements, while the head is a single statement. For example, policy (17), example 2 can be encoded as *(A, landed, seatac_run$_2$), (A, useNetwork, WiMAX) → (A, violates, policy$_{17}$)*.

4.1. EXAMPLES OF FORMAL POLICIES

In order to be able to formalize policies we need to define a vocabulary of resource classes and predicates for expressing the state of the system. This vocabulary is called an *ontology*. We define the resource types *airport*, *runway, application, network, and maintenance_device.* We also define the special resource *internet* to represent the public Internet network, and the resource *ground-only* to indicate that application should be accessed only when in weight-on-wheel status.

Policy	RDF Rule	Example of Sources
(2) Aircraft accessing a "ground-only" airline application must be in weight-on-wheels condition.	*(A, type, aircraft), (P, type, application), (AL, type, airline), (R, type, runway),* *(A, logged_on, P), (P, part_of AL), (P, apptype, ground-only), ¬(A, landed, R) → (A, violation, policy$_2$)*	aircraft *aircraft$_1$*: (*aircraft$_1$*, landed, *) airline airline$_D$: (*, partof, airline$_D$), (*, logged_on, P), (P, type
(5) Maintenance device must be disconnected from Internet when accessing airplane system	*(D, type, maintainance_device), (A, type, aircraft), (N, type, network), (D$_1$, type, device), (D$_2$, type, device)* *(D, connected_to, internet), (D, connected_to, N), (N, partof, A) → (D, violation, policy$_5$)* *(D$_1$, rec_from, D$_2$) → (D$_1$, connected_to, D$_2$)* *(D$_1$, connected_to, D$_2$), (D$_2$, connected_to, internet) → (D$_1$, connected_to, internet)*	maintainance_device dev1: (dev1, connected_to, *) aircraft *aircraft$_1$*: (*, partof, *aircraft$_1$*)
(11) Airplane must automatically choose the cheapest available wireless data link that satisfies the minimum bandwidth requirement	*(A, type, aircraft), (N, type, network), (M, type, network), (AIR, type, airport), (SW, type, application),* *(A, connected_to, N), (N, partof, AIR),* *(A, airplane_min_bw, MINBW), (N, cost, NC), (A, inrange, M), (M, partof, AIR), (M, net_available_bw, AVMBW), (M, cost, MC), (M != N), (MC < NC) → (A, violation, policy$_{11}$)*	aircraft *aircraft$_1$*: (*aircraft$_1$*, connected_to, *), (*aircraft$_1$*, airplane_min_bw, *), (*aircraft$_1$*, inrange, *) airport seatac: (N, partof, seatac), (N, cost, *), (N, net_available_vw, *)

Table 1: Formal representation of three airport policies. The column *sources* contains statement patterns that summarize the set of statements generated by each device.

We define a set of predicates that describe the relation between resources. The predicate *type* (defined by the RDF standard) specifies that a particular resource belongs to a class. The predicate *violation* specifies that a resource is violating a policy. The predicate *part_of* describes that a resource is part of a larger resource. For example, we identify that an application p_1 is managed by the airline al_1 using the statement *(p_1, part_of, al_1)*.

We define predicates that are specific to the case of airports. In policy (2), the predicate *landed* specifies that the aircraft have weight-on-wheels on a runway. In policy (11), we define a set of predicates that provide information about the state of the network interconnections. We define the predicate *airplane_min_bw*, which indicates the minimum bandwidth requirement of the plane (collected from information about the current applications running on the aircraft). Similarly, we define the predicate *net_available_bw* that define the bandwidth available in a wireless network. The predicate cost identifies the cost of using a particular wireless network. The encoding of the rules in Datalog is shown in the first two columns of Table 1.

4.2. RUNTIME VALIDATION OF POLICIES

The verification of policies requires aggregating information about the state of the system. Once information is collected, we check if the overall state triggers policy violations.

In our architecture, the monitoring software running on the device can be configured to send messages that contain updates about the device state every time that there is a state change. For example, when the aircraft *aircraft$_1$* lands, a device *s* installed on the aircraft can send a message to notify that the state of the airplane changed. This message contains the statement (*aircraft$_1$, landed, seatac_run$_1$)*. The last column of Table 1 shows the devices providing the different parts of the information about the policy. As multiple organizations can verify independently the policies, each device might need to send its state update information to different entities. Section 5 describes an algorithm on device selection by each organization, in order to determine the state of the infrastructure relevant to policies.

Policies can also be used to define complex concepts from information about the system. For example, we use two rules in policy (5) to define the concept of "connected to Internet". We say that if a device D_1 that can receive messages from another device D_2, then D_1 is "connected" to

D_2. Also, we define that the connectivity is transitive by saying that if a device D_1 is connected to a device D_2 that is connected to the Internet, then device D_1 is considered connected to the Internet as well. Using this rule, we can detect complex indirect interconnections to the public Internet network.

Often, multiple devices generate the same piece of information. In policy (2), the information about the weight-on-wheel condition of the aircraft may be provided both by aircraft systems and by the ground radar of the airports. Our mapping algorithm takes advantage of these redundancies to enable the application of separation-of-duty constraints and redundant validation in the monitoring process.

5. MULTI-ORGANIZATION POLICY COMPLIANCE

The detection of policy violations in airport systems is complicated by the need of preserving the integrity and confidentiality of information about the system's state across the different organizations that manage the airport infrastructure.

Confidentiality problems arise because of the nature of the collaboration between organizations in large infrastructures. Often, competing organizations are required to interact to guarantee compliance to government regulations. However, sharing information about the state of the system might reveal to the competitor information about the organization's structure. For this reason, an organization might maintain confidential part of the state of the system to other organizations.

Integrity problems arise in the same context when there is limited trust between organizations. Information used for the auditing of policies that are considered reliable for an organization might not be considered trustworthy for another. An organization might want to protect itself against misuse by third party organizations, and hence auditing should not be based on potentially tainted information coming from external organizations.

Additionally, redundancy and separation-of-duty can be imposed as additional integrity constraints to protect against compromises of sensors by an attacker or by a malicious insider. When these constraints are in place, information for the validation of policies is acquired from multiple independent sensors. All these sensors provide information about the same portion of the state of the system. Using such redundant information, the correct state can be reconstructed even if a small number (i.e., the minority) of sensors are compromised.

It is important to assure that the sensors from which information is acquired are "independent". Malicious insiders can use valid credential to take control of sensors. If a set of valid credential is compromised, all sensors controlled using such authorization can be compromised. Separation-of-duty constraints are a special type of redundancy constraints that makes this type of attack harder: they specify that the set of sensors providing the redundant information for the validation of the policy should not be controlled using the same credential.

Type	Meta-policy	Condition
C	($aircraft_1$, $airline_D$)	Information about the state of the aircraft $aircraft_1$ cannot be forward to a competing airlines
I	(5, connected_to, $contractor_A$)	$Contrator_A$ is not trusted to provide the information about which networks are used by its devices
S	(1, * landed *, 3, true)	Statements (*, landed, *) of policy 1 need to be acquired form three independent sensors

Table 2: Examples of confidentiality (C), integrity (I), and separation-of-duty (S) policies.

Our system provides automated methods for mapping the policies audited by each organization to the sensors used for acquiring the information needed in the evaluation of the policy.

The architecture of the system is shown in Figure 2. Policies are verified at runtime by machines that integrate the information about the state. These machines are called *verifiers*. All devices that generate information that it is used for the evaluation of a policy are called *sensors* (i.e., they "sense" a portion of the system's state). Each organization independently manages one or more verifiers and it can select a set of policies to be validated on each of them. Organizations can place verifiers in multiple areas of the network to isolate them from direct attacks and provide redundancy to the policy validation process. Given this policy assignment, our system maps sensors to verifiers to assure that each verifier receives information to validate policies in a way that satisfies the organization's requirements about confidentiality, integrity, and separation-of-duty.

Confidentiality, integrity, and separation-of-duty requirements are formally specified as *meta-policies*. Meta-policies are constraints specified on how the information for validating policies is acquired from the system. Confidentiality meta-policies are specified by organizations and specify which information generated by sensors should not be shared with other organizations. Integrity meta-policies are specified for each organization and for each statement pattern in a policy. Using an integrity meta-policy, an organization specifies the set of organizations and sensors that are trusted for the policy verification.

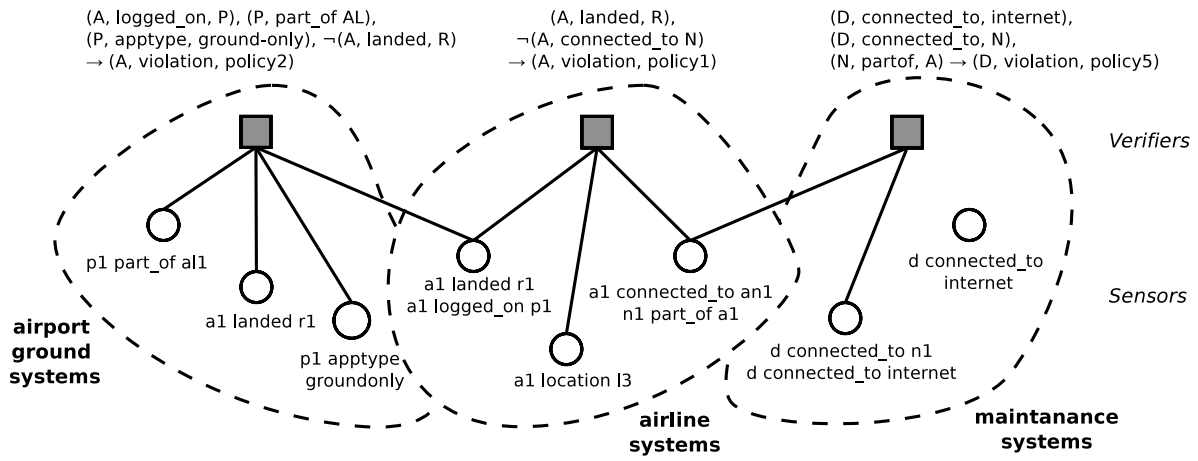

Figure 2: Example of the association between sensors and verifiers in a multi-organization online policy monitoring system. Different organizations manage independent *verifiers*, which acquire information from devices (*sensors*) under the administrative control of other organizations

Separation-of-duty policies specify that certain statements need to be confirmed using redundant and independent sources. Table 2 shows an example of these types of policies.

5.1. MAPPING MODEL

We represent a source device as a tuple $s_j = (O_i, U_i, S_i, C_i)$ where O_i is the organization that manages the sensors, U_i the set of users authorized to manage the sensor, S_i is the set of statements about the state that can be potentially generated by the device, and C_i is a function $C_i : E \rightarrow N$ that represents a cost of sending information to an organization E.

The union of all the data coming from the sensors is all the information required for performing the policy validation. At every instant in time t, only a subset $S_i'(t) \subset S_i$ of statement is true for a sensor i. For a system composed of n sensors, the union of all statements generated at time t by the sensors represents the complete state $S(t)$ of the infrastructure at time t (i.e., $S(t) = \cup S_i'(t)$ with $i=1,\ldots, n$). To validate policies, we create a knowledge base $KB(t)$ that integrates the state $S(t)$ and the policies. We use inference to check if it is possible to use the rules to infer statements that indicate policy violations from the state $S(t)$.

When rules are distributed across verifiers, it is not necessary to recreate the entire state at each of them. Each verifier needs to acquire the portion of the state that it is used by the policies assigned to it. For the purpose of assignment, we represent a policy p_i as the set of all potential statements S_p that can match any of the statement patterns of p_i. For example, in a system with two devices d_1 and d_2 that can have an attribute *state* with values on and off, the policy $(D_A$ *state S,* D_B *state S,* D_A *!=* D_B, $\rightarrow$ *fail)* has S_p = *{d1 state on, d1 state off, d2 state on, d2 state off}*.

A verifier is represented as a couple $v_i = (O, P)$ where O is the organization managing the verifier, and P is a set of policies assigned to the verifier.

Confidentiality meta-policies are defined as access control tuples (S, O), where S is a sensor and O an organization. If a tuple (s_i, o_j) is defined in the system, then the data generated by s_i can be sent to the organization o_j.

Integrity policies are defined as access control tuples (P, S, O). If a tuple (p_i, s_i, o_j) is present in the system then the policy p_i can use data generated by sensors managed by o_j as information for the statement s_i.

Redundancy and separation of duty policies are defined as tuples (P, S, R, D) where P is a policy, S a set of statements, R the redundancy with which these statements need to be validated, and D is true if separation of duty is required, false otherwise.

5.2. SENSOR-VERIFIER MAPPING

The sensor-verifier mapping associates each sensor with one or more verifiers that require its state information for the validation of policies. As multiple devices can provide the same information about the state, and policy validation needs to satisfy complex requirements, obtaining the least-cost mapping is not trivial.

Without considering meta-policies, a verifier needs to communicate with the least-cost subset of the available sensors that provide all statements required by the policies managed by it. Formally, if P_i is the set of statements required by policy p_i, and if a verifier is managing the policies $p_1,\ldots,p_n$, then the set of statements required by the verifier is $S_v = \cup_{i=1..n} Pi$. Each sensor provides a subset of statements S_j and has a cost of communicating the verifier

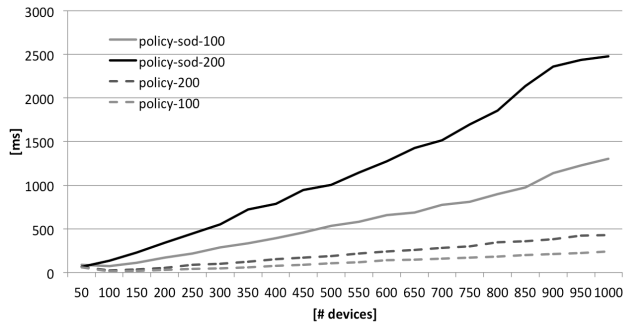

Figure 4: Computation time for sensor mapping as a function of number of devices in the system.

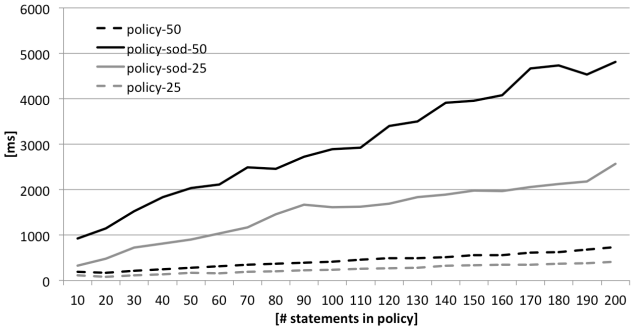

Figure 5: Computation time for sensor mapping as a function of policy size

c_j. The goal of the sensor-verifier mapping is to find the minimum cost subset of sensors that generate all statements S_v required by the verifier. In this form, the problem is NP-complete and the proof is based on a simple mapping of the minimum cover set problem [17] to instances of our problem without meta-policies.

We use the Chvatal [17] heuristic for the minimum cover set problem as a base for our mapping algorithm. This heuristic selects a set of sensors that minimizes the cost of acquiring all information required by the policies. However, our mapping requires taking into account meta-policies as additional constraints. Confidentiality and integrity constraints require certain sensors not to be used. Separation-of-duty constraints specify that, for a specific subset of the statements, the mapping needs to include multiple sensors that generate such statements.

To increase the chance of finding a feasible solution, our algorithm assigns sensors so that the most complex constraints are satisfied first. We prepare the list of sensors available to the verifier so that confidentiality and integrity constraints are always satisfied. To do so, we remove from the list of available sensors all the sensors that cannot send information to the verifier under consideration because of confidentiality constraints, and we remove from the list of statements provided by each sensor the statements that the verifier does not consider as trustworthy because of integrity constraints. Then, we assign sensors to satisfy every separation-of-duty and redundancy requirements. Last, we assign assure that all statements required by the verifier are mapped to at least one sensor.

Separation-of-duty constraints are satisfied by performing a heuristic search in the space of sensor assignments. For each constraint, we first select the sensor that maximizes ratio between the number of generated statements that are useful towards the satisfaction of the constraint, and the cost of the sensor. Once we have added such a sensor, for strict separation-of-duty constraints, we remove from the list of available sensors all other devices that share with it one or more users. We continue until we find a valid assignment. If we reach a point at which no sensors are available for satisfying the constraint, we backtrack and we select the next-best sensor that maximizes the ratio. The pseudo-code for this part of the algorithm is shown in Figure 3.

After this phase, we consider the remaining set of statements to cover and we use the Chvatal heuristic to minimize the cost of the selected sensors. The heuristic maps sensors so that every statement required by the verifier is provided by at least one sensor. At each step we add to the list of sensors the one that maximizes the ratio between the number of statements it provides over the cost of using such a sensor.

6. EVALUATION

In this section, we evaluate the execution time of the proposed heuristic algorithm to validate its applicability in the online monitoring architecture. As the goal of the evaluation is to measure the efficiency of the algorithm in a wide range of situations, we perform our experiments on simulated scenarios. We represent the domain of the infrastructure state using a set of randomly generated statements (N_{DOMAIN}). Information about such statements is provided (redundantly) by a specified number of sensors. For the application of separation-of-duty policies, we assign a variable number of users (U_{MAX}) to manage each sensor. All simulations are performed on a 2GHz Core 2 Duo system with 2 GB of RAM. All data shown in the graphs is the average of 10 executions.

Our first experiment shows the need of a heuristic algorithm for solving the sensor-mapping problem. We analyze the execution time for computing an optimal mapping using a standard branch-and-bound search algorithm. For a small infrastructure system with 40 devices, the time for computing the mapping is 565.3 seconds. For larger infrastructure systems, the time grows exponentially. As airport infrastructure systems might be composed of thousands of devices, such execution times are not acceptable. The sensor-mapping algorithm is run several times during the lifetime of the infrastructure, as devices are connected and disconnected from the system frequently and every time a new sensor-matching needs to be computed. Under the same conditions, our heuristic computes an approximated mapping in 0.073 seconds.

In the second set of experiments, we use our heuristic and we examine how the execution time grows with the size of the infrastructure system. We evaluate the time for computing the sensor mapping of policies without separation-of-duty constraint (*policy-n*) and of policies only with separation-of-duty constraints (*policy-sod-n*). We consider a policy composed of 25 statements, and the redundancy requirement in the separation-of-duty constraints is set to 2. Additionally, we vary the number of statements provided by each sensor. We consider two cases: 100 statements assigned to each sensor (policy-100, policy-sod-100), and 200 statements per sensor (policy-200, policy-sod-200). Statements are chosen randomly from an overall domain whose size increases linearly with the number of devices. We find that in all cases the execution time grows linearly with the number of sensors and remains under acceptable time for online mapping. These results are shown in Figure 4.

The third set of experiments examines how the number of statements in the policy affects the execution time. As with the previous set of experiments, we consider policies with and without separation-of-duty constraints, and we considered different numbers of statements on each sensor. For these experiments we fix the number of devices to 500. In all cases, the execution time grows linearly with the number of statements in the policy. These results are shown in Figure 5.

In summary, the heuristic algorithm provides an efficient way to perform online sensor mapping in our architecture, and the linear growth of the execution time allows our system to scale to large infrastructure systems.

7. CONCLUSIONS

The use of policy-based monitoring in the management of the interaction between eEnabled aircraft fleets and airport infrastructure provides a way to formalize the states of the infrastructure that each organization considers secure, safe, and efficient.

In this paper we presented a framework for the specification of infrastructure security policies in the eEnabled fleets and airport environment and we described an architecture that allows the online monitoring of such policies in a multi-organization scenario. Meta-policies are used to specify constraints in the interaction between sensors and verification server managed by different organizations, and they are also used to enforce constraints that use redundancy to reduce the possible consequences in the verification process of sabotages to sensors. The heuristic algorithm that we propose for the mapping of information sources and verification servers allows computing efficiently a solution that respects all confidentiality, integrity, and separation-of-duty meta-policies without increasing excessively the overall cost of the solution when compared with the optimal mapping.

To the best of our knowledge, this is the first paper to address the problem of establishing a security policy framework for future eEnabled fleets and airports. Future work will explore in detail major challenges presented in this paper. We are planning to implement the architecture presented in this paper on top of standard network management tools. Additionally, while this paper provides a framework for expressing and validating policies, it does not analyze the complex problem of defining and optimizing a complete set of security policies that it is suited for the airport environment. Future work should analyze such a problem in more detail.

REFERENCES

[1] "E-Enabling," Boeing Frontiers, 02:04, August 2003. http://www.boeing.com/news/frontiers/archive/2003/august/i_ca1.html

[2] R. D. Apaza, "Wireless communications for airport surface: an evaluation of requirements," *IEEE Aerospace Conference*, pp.1779-1788, March 2005

[3] David Oppenheimer, Archana Ganapathi, David A. Patterson, "Why do internet services fail, and what can be done about it?", *USITS'03, USENIX Symposium on Internet Technologies and Systems,* USENIX, 2003.

[4] Stephen D. Brady, Richard J. Hillestad, "Modeling the External Risk of Airports for Policy Analysis", *RAND European-American Center for Policy Analysis*, 1995

[5] Alan Shieh, Oliver Kennedy, Emin Gun Sirer, and Fred B. Schneider, "NetQuery: A General-Purpose Channel for Reasoning about Network Properties." in *USENIX Symposium on Operating Systems Design and Implementation*, 2008

[6] Distributed Management Task Force (DMTF), "Web-Based Enterprise Management (WBEM)." http://www.dmtf.org/standards/wbem, 2009.

[7] Tenable Network Security. "Nessus: the Network Vulnerability Scanner." http://nessus.org/nessus/, 2009

[8] Sushil Jajodia, Steven Noel, and Brian O. Berry. "Topological analysis of network attack vulnerability." *Managing Cyber Threats: Issues, Approaches and Challenges*, 2005

[9] Ou, X., W.F. Boyer, and M.A. McQueen. 2006. "A scalable approach to attack graph generation." in *ACM Conference on Computer and Communications Security*, ACM, 2006

[10] David Allen, "Electronic Flight Bag: Real-Time Information Across An Airline's Enterprise", *Boeing Aero Magazine,* Qtr 2.08, 2008.

[11] Monirul I. Sharif, Wenke Lee, Weidong Cui, Andrea Lanzi, "Secure in-vm monitoring using hardware virtualization" in *ACM Conference on Computer and Communications Security*, ACM, 2009

[12] Bryan D. Payne, Martim D. P. de A. Carbone, Wenke Lee, "Secure and flexible monitoring of virtual machines", in *Annual Computer Security Applications Conference*, IEEE Computer Society, 2007

[13] Stefano Ceri, Georg Gottlob, and L Tanca. "What you always wanted to know about Datalog (and never dared to ask)." *IEEE Transactions on Knowledge and Data Engineering,* 1989

[14] Avik Chaudhuri, Prasad Naldurg, G. Ramalingam, Sriram Rajamani, and L. Velaga. "EON: Modeling and analyzing dynamic access control systems with logic programs." in *ACM Conference on Computer and Communications Security*. 2008.

[15] W3C. "Resource Description Framework (RDF)." http://www.w3.org/RDF/, 2004

[16] Campbell, Roy H., and Mirko Montanari. "Multi-Aspect Security Configuration Assessment." in *ACM Workshop on Assurable & Usable Security Configuration (SafeConfig),* 2009

[17] Václav Chvatal. "A Greedy Heuristic for the Set-Covering Problem". *Mathematics of Operations Research*, 1979

APPENDIX A: EXAMPLES OF SECURITY POLICIES

A.1. SAFETY RELATED POLICIES

These policies are specified to ensure safety of aircraft.

(1) Approaching aircraft must establish a broadband wireless link when it enters weight-on-wheels condition.

(2) Airplane accessing a "ground-only" airline application must be in weight-on-wheels condition.

(3) Airplane must be parked at gate when accessing airline application Y.

(4) RFID tags in cabin must not be read when airplane is not parked at gate.

A.2. ACCESS CONTROL POLICIES

These policies are specified to prevent unauthorized access to aircraft.

(5) Maintenance device must be disconnected from Internet when accessing aircraft systems.

(6) Maintenance crew must login with proper credentials (such as passwords) in order to be able to use maintenance laptop to perform assignments.

(7) Users logged in maintenance laptop as maintenance crew cannot access files or perform security actions privileged to administrator only.

(8) Maintenance crew and device credentials must be authorized and authenticated before interacting with airplane systems.

(9) Maintenance crew and device must be located on airport tarmac when accessing external access point of aircraft.

(10) Maintenance crew and device must be located inside cabin when accessing internal access point of aircraft.

A.3. BUSINESS RELATED

These policies are specified to minimize operational costs of aircraft while maintaining quality of network connection.

(11) Aircraft must automatically choose the most cost-effective wireless data link available that satisfies the minimum bandwidth requirement.

(12) When current bandwidth drops below the minimum bandwidth threshold, aircraft must automatically search for the next cheapest available wireless data link.

(13) Maintenance devices should use the cheapest available wireless network access point to connect to Internet.

(14) RFID tag must not respond to a query issued less than a threshold distant to protect airlines proprietary data.

(15) While taxing, for a period of time, an airplane is allowed to be associated with two access points for smooth handover.

A.4. AIRPORT OPERATION RELATED

These policies are specified to enable efficient resource allocation at the airport.

(16) Technology based requirement. Examples:

- Airplane using WiFi technology must not use Terminal T1.
- Airplane using Cellular technology must use Gates G1-G5 at Terminal T2.
    - Airplane using WiMAX must use runway R1 for takeoff.

(17) Airport layout based requirement. Examples:

- Airplane using Gates G10 to G20 must use WiMAX technology.
- Airplane using runway R2 must not use WiMAX technology.
- Crew at tarmac below Gates 30-35 must not use Cellular.

(18) Time-based (network/airport demand based) requirement. Examples:

- Airplane using the airport computer network at terminal T3 between time t1 to time t2 must use WiMAX.
- Airplane using the airport computer network at Gates G40-G45 must not use WiFi.

BIOGRAPHY

Mirko Montanari is a PhD student in Computer Science at the University of Illinois at Urbana-Champaign. His research interests are in the areas of security and distributed systems. He received his Laurea Specialistica degree (MS degree) in Computer Engineering from the University of Bologna, Italy.

Roy Campbell is the Sohaib and Sara Abbasi Professor in the Department of Computer Science at the University of Illinois at Urbana-Champaign. Professor Campbell's research interests are the problems, engineering and construction techniques of complex system software. Security, continuous media, and real-time control and pose a challenge, especially to operating system designers. Ubiquitous, distributed and parallel systems require complex resource management and efficient implementations. Object-oriented design aids organizing software, supports customization and offer new approaches to building dynamic distributed systems and middleware. Over time, research in system software has become increasingly important and the construction of complex system software a focus for advanced software engineering techniques. His current research projects include security assessment of SCADA networks, operating system dependability and security, active spaces for ubiquitous computing, and the design of peer-to-peer distributed operating systems.

Krishna Sampigethaya is an advanced technologist at the Boeing Research & Technology, Bellevue, WA, working on performance assurance of cyber-physical systems, NextGen, vehicular networks, and the smart grid. He received MS and PhD in electrical engineering from the University of Washington. He was a program committee member for the 2008 NITRD workshop on Transportation CPS, co-chair of the 2009 Army Research Office CPS security workshop and co-organizer of SAE 2010 Future ATM Technology Symposium. Dr. Sampigethaya is technical area chair for aviation cyber security at the 2009 and 2011 SAE AeroTech, trustworthy aviation information systems area at the 2010-2011 AIAA Infotech@Aerospace. He is a IEEE and AIAA member. He is the founding chair for the SAE aviation cyber security technical committee and co-editor for the Proceedings of the IEEE special issue on cyber-physical systems.

Mingyan Li is an advanced computing researcher at Boeing Research and Technology (BR&T) and an affiliated assistant professor in the department of Electrical Engineering (EE) at University of Washington (UW). She received her Doctor of Philosophy degree from Network Security Laboratory in EE department at UW in 2006. Her research interests are in the area of network security and user privacy, with applications to next-gen airport wireless systems, sensor networks, RFID applications, software distribution systems, medical security systems, vehicular ad hoc networks (VANET), distributed storage, and secure multicast. She was leading Boeing-Siemens collaborative projects on wireless and RFID security. She is a recipient of BR&T silver teamwork award 2008, an IEEE PIMRC best student paper award 2007, the UW EE departmental Chair's Award 2006, and the outstanding Society of Women Engineer (SWE) Graduate award 2003

ACKNOWLEDGEMENTS

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors, and should not be interpreted as the views of The Boeing Company.