Chapter 11

CYBER-PHYSICAL SECURITY OF AIR TRAFFIC SURVEILLANCE SYSTEMS

Anusha Thudimilla and Bruce McMillin

Abstract Cyber-physical system security is a significant concern in the critical infrastructure. Strong interdependencies between cyber and physical components render cyber-physical systems highly susceptible to integrity attacks such as injecting malicious data and projecting fake sensor measurements. Traditional security models partition cyber-physical systems into just two domains – high and low. This absolute partitioning is not well suited to cyber-physical systems because they comprise multiple overlapping partitions. Information flow properties, which model how inputs to a system affect its outputs across security partitions, are important considerations in cyber-physical systems. Information flows support traceability analysis that helps detect vulnerabilities and anomalous sources, contributing to the implementation of mitigation measures.

This chapter describes an automated model with graph-based information flow traversal for identifying information flow paths in the Automatic Dependent Surveillance-Broadcast (ADS-B) system used in civilian aviation, and subsequently partitioning the flows into security domains. The results help identify ADS-B system vulnerabilities to failures and attacks, and determine potential mitigation measures.

Keywords: Cyber-physical systems, ADS-B system, integrity, privacy

1. Introduction

Recent years have seen significant increases in the development and deployment of cyber-physical systems – smart, mission-critical computing systems that are characterized by tightly-coupled embedded devices in physical environments [7]. Since cyber-physical systems comprise physical components, computational resources and communications in-

frastructures [14], it is equally important to ensure cyber security and physical security.

Cyber-physical systems are exposed to new forms of risk due to the tight couplings between their cyber and physical components. These risks have not been considered adequately in cyber-physical systems due to the lack of tools for identifying vulnerabilities that arise from the complex interactions between their cyber and physical components. These risks can be classified as: (i) cyber elements that impact the physical environment; and (ii) physical elements that impact the cyber components. This chapter addresses these risks by identifying failures and attacks using information flow analysis, an important cyber-physical security paradigm that determines if inputs to a process or system can change its outputs.

Air traffic surveillance systems have complex cyber-physical interactions. Significant increases in civilian airline traffic in recent years have led the Federal Aviation Administration (FAA) to introduce NextGen technologies to ensure flight predictability, efficiency and safety. Automatic Dependent Surveillance-Broadcast (ADS-B), a key NextGen component, is a powerful cyber-physical system that integrates computational intelligence with physical components to provide reliable and efficient communications between air traffic control (ATC) and aircraft. ADS-B uses the Global Positioning System (GPS) and onboard sensors to determine aircraft identity, position, altitude and velocity. Because ADS-B broadcasts all information over unauthenticated and unencrypted wireless channels, it is imperative to protect against false data injection, spoofing, flooding, jamming, message modification and eavesdropping attacks [10]. Unfortunately, ADS-B does not employ adequate security mechanisms [10, 12, 18].

This chapter employs information flow analysis to identify faulty components in the ADS-B system and detect attacks. Two scenarios are discussed, one involving aircraft altimeter failure and the other involving GPS satellite failure.

2. ADS-B System

ADS-B is an airborne surveillance system designed to enable seamless surveillance, and collision detection and avoidance, and to provide situational awareness in the air and on the ground. It replaces radar-based surveillance by having every aircraft broadcast its identity, position, altitude, velocity and other data over unencrypted data links once per second. An aircraft equipped with ADS-B collects this data from sources such as GPS, barometric altimeter and other nearby aircraft,

and processes the data to determine its accuracy and integrity. The processed information is then encoded and broadcast as an ADS-B message to nearby aircraft and ground stations. ADS-B also enables pilots to receive other information such as flight restrictions and weather data in real time.

An aircraft equipped with ADS-B broadcasts information in an omnidirectional manner so that it can be received by ground stations and other aircraft with compatible receiving devices. These broadcasts differ from other transponder interrogations such as those performed by the Traffic Collision and Avoidance System (TCAS).

ADS-B has two functional operations: (i) ADS-B OUT; and (ii) ADS-B IN. ADS-B OUT is a surveillance technology responsible for generating ADS-B broadcasts that transmit aircraft identity, position, altitude and velocity data in real time to air traffic control and nearby aircraft. ADSB-IN is used to receive transmissions from nearby aircraft and ground stations (i.e., ADS-B OUT information). The information includes weather updates, conflict detection and de-confliction guidance, graphical displays of the position of aircraft and along-track guidance [11].

3. Related Work

Cyber-physical systems are widely deployed to monitor and control operations in critical infrastructure assets. Due to their importance, it is vital to protect them from failures and attacks. Several methods have been proposed to address security issues posed by denial-of-service attacks [2], false data injection attacks [9], stealthy deception attacks [19] and replay attacks [13].

Information flow security is a powerful approach for preventing sensitive data from leaking to malicious entities. Its primary variants are static and dynamic approaches. A static approach merely executes information security policies whereas a dynamic approach uses labels to describe security levels and propagates the labels to ensure data integrity with respect to invariants or predefined policies. Much work in the area has focused on proving the non-interference property that describes information flow restrictions and using a combination of language features and system models to implement information flow security [16].

Considerable research has focused on using information flow analysis in aviation security. The real challenge in the case of information flow security is to apply the vast theory and language-based designs such as Jif [3] and Flow Caml [15, 17] to real-world problems [23].

In the context of ADS-B security, Kim et al. [8] have devised a timestamp method based on signal propagation time to identify and reject spoofed ADS-B messages between senders and receivers. Yang et al. [21] have proposed a lightweight security solution that integrates cryptoprimitives such as FFX and TESLA to ensure ADS-B message integrity and privacy [21]. Other researchers [22] have conducted similar work with a focus on congested data links and resource-constrained avionics. Thudimilla and McMillin [20] have employed ProVerif to identify attacks on ADS-B and TCAS, but their analysis is limited to proving observational equivalence (anonymity property) via process composition. Several researchers have investigated security vulnerabilities, failures and attacks in ADS-B systems [10, 12, 18]. A survey of the literature reveals that research in the area of ADS-B security either focuses on cyber attacks or physical component failures, but not both. This work stands out in that it considers cyber- and physically-enabled attacks and failures using automated graph-based information flow analysis to identify security risks and develop mitigation measures.

4. Threat Model

An adequate threat model is crucial to assessing security vulnerabilities in a system and determining mitigation measures. The following threat model is at the foundation of this work:

- *Source:* The threat source is an entity that initiates threats, which include system failures, adversarial attacks and environmental factors.

- *Goal:* The goal of the threat model is to capture system features that may lead to system failure and identify the features that are modified as a result of an attack.

- *Consequences:* The consequences include compromises to system integrity, safety or availability.

The threat model assumes that failures and attacks exhibit similar behavior, and they are arbitrary and unbounded. The adversary is assumed to have full control of the target – specifically, the adversary can eavesdrop, intercept and modify messages sent and received by a component.

The threat model also assumes that the adversary cannot exploit certain aspects of the system. Specifically:

- The adversary cannot corrupt or modify a proposed model.

Table 1. Modal logic symbols and descriptions.

Symbol	Description
W	Set of worlds.
φ	Boolean statement that is true or false in world $w \in W$.
$w \vdash \varphi$	Statement φ is valid in world w.
$w \models \varphi$	Values from world w cause φ to evaluate to true.
$\Box\varphi$	Statement φ always evaluates to true or false.
$V^i_{S_x}$	Valuation function of entity i with respect to state variable S_x.

- The adversary cannot modify more than half of the participating entities while traversing the graph with respect to a feature in the proposed model.

5. MSDND

Multiple security domain nondeducibility (MSDND), introduced by Howser and McMillin [5], engages modal logic [4] to address the shortcomings of traditional security models that partition a security space into two domains, high and low. Traditional models work well only when the boundaries of the security domains are defined precisely. The MSDND model checks for nondeducibility – the inability to deduce the values of two states in a system at any point in time. Computing the security domains for complex cyber-physical systems is difficult. However, this work addresses the issue by automating the MSDND process to partition security domains based on information flow traversal.

Table 1 lists the modal logic symbols used in this work along with their descriptions.

The state of a system is represented by a set of variables that help determine the future system states and outputs. A state variable φ represents the state of a dynamic system, which is evaluated to either true or false. For example, a mechanical system may contain state variables such as position and velocity that describe its components. A combination of state variables S_x is represented as:

$$S_x = \varphi_0 \wedge \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \ldots \tag{1}$$

The set of worlds W contains all possible combinations of m state variables $S_0, S_1, \ldots S_m$. If a state variable S_x has no valuation function, then MSDND fails to determine the value associated with the state variable or the value of any logical expression associated with the state variable [5].

A valuation function $V^i_{S_x}$ indicates the value of a state variable S_x as observed by an entity i in world w.

Each state variable is associated with a component identifier. Figure 1 shows the ADS-B system architecture with components represented as nodes and connections between components represented as edges. In the figure, Satellite 1 has the component identifier 0, Barometric Altimeter has the component identifier 6 and Control Panel has the component identifier 9. Every edge is labeled with attributes that denote the data that flows from one component to another [1]. The state variables associated with Satellite 1, Barometric Altimeter and Control Panel are denoted by φ_0, φ_6 and φ_9, respectively. These state variables are used later in the chapter to perform MSDND analysis.

Definition 1 (Multiple Security Domain Exclusivity). There exists some world with multiple states $S_a, S_b, S_c, \ldots$ in which, at any instant, the system is in one true state and all the others are false:

$$f(S_a, S_b, S_c, \ldots) = \begin{cases} \text{True} & \text{when one of } S_a, S_b, S_c, \ldots \text{ is True} \\ \text{False} & \text{otherwise} \end{cases} \tag{2}$$

In the MSDND model, an entity i is any part of the system that is capable of independent observation or action. The event system ES comprises multiple security domains SD^i as viewed by each entity i in the model. These domains may or may not overlap depending on the complexity of the event system [5, 6].

Definition 2 (Multiple Security Domain Nondeducibility). A system is MSDND if:

$$MSDND(ES) = \exists w \in W : [w \vdash \Box f(S_a, S_b, S_c, \ldots)] \wedge [w \models (\nexists V^i_{S_a} \wedge \nexists V^i_{S_b} \wedge \nexists V^i_{S_c} \ldots] \tag{3}$$

$$\bigcup_{i=1}^{n} SD^i = ES \tag{4}$$

An MSDND proof creates a logical argument based on conditions on the observable state of the system under consideration. These conditions are assessed for their valuation in a particular security domain. If no valuation function can be found, then the system is MSDND secure. This is a bad thing because it means attacker actions can be hidden in the security domain. Preventing MSDND is a good thing because it means the system can detect the attack.

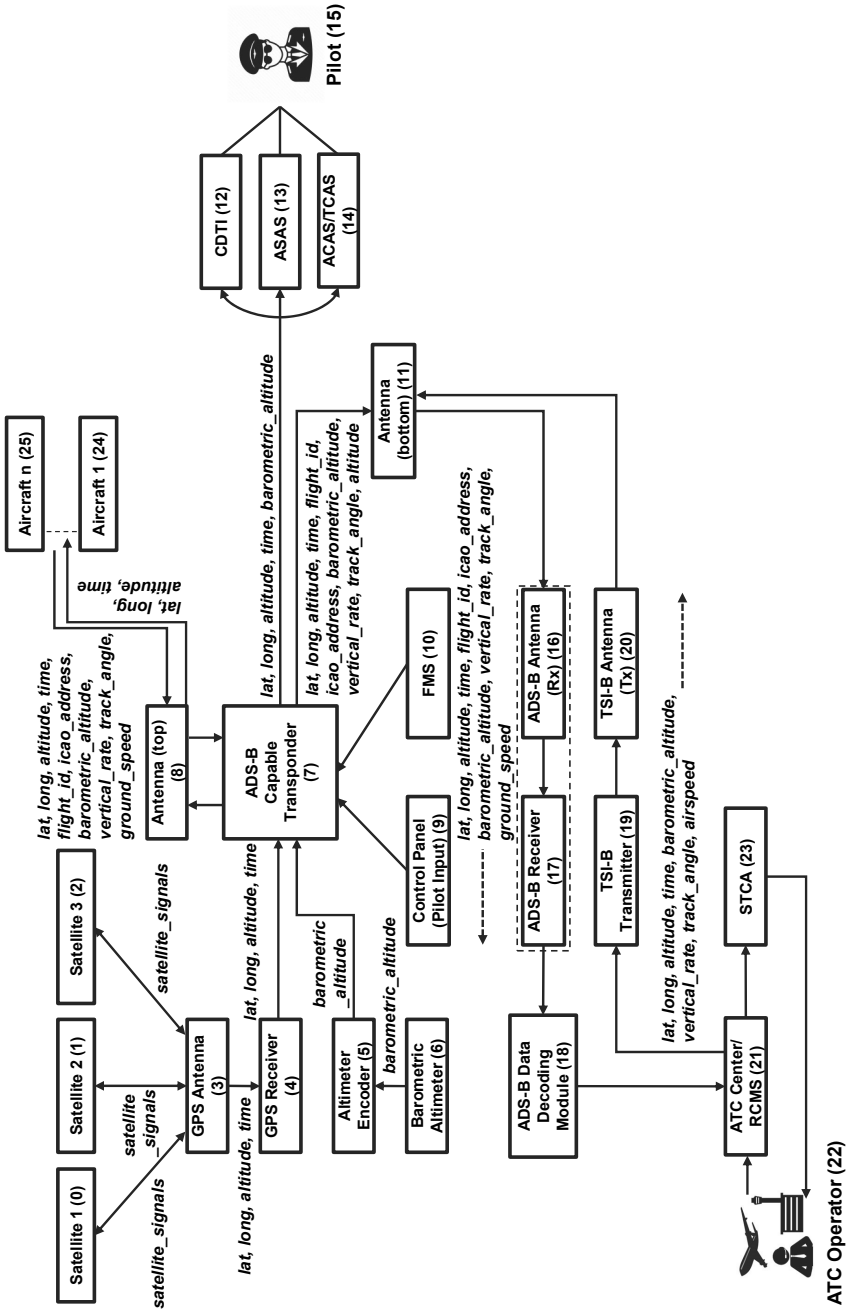

Figure 1. ADS-B system architecture with components (nodes) and connections (edges).

6. Graph-Based Detection System

The graph-based detection system presented in this chapter considers the entire aircraft system as a directed acyclic graph, where a node represents a system component and an edge represents the information flow between two nodes. Each edge has a set of labels, where each label has a value associated with it.

The graph-based detection algorithm has the following three steps:

1. Identify all the paths in the network using depth-first search with respect to a label and sort the result set in descending order of subgraph size.

2. Identify the subgraphs and eliminate them to obtain a reduced unique subgraph set.

3. In the reduced set, traverse each edge to check for discrepancies such as inconsistent values associated with a particular label. If there is a discrepancy, find the in-degree of each node.

4. A node with in-degree no less than three will break the MSDND property because it contains more than two information flow paths that help identify the faulty component or the component under attack that is sending incorrect data during design time or runtime.

Definition 3 (Directed Acyclic Graph/Subgraph). A directed acyclic graph $G = (V, E, L)$ comprises a set of nodes V, set of edges E and set of labels L associated with each edge.
A directed acyclic graph $S = (V_s, E_s, L_s)$ is a subgraph of directed acyclic graph $G = (V, E, L)$ iff $V_s \subseteq$ V, $E_s \subseteq$ E and L_s contains the label l under consideration.

Figure 1 shows the ADS-B system of an aircraft, which has 26 components (nodes). Each edge has a set of labels and values associated with the labels that represent information flows.

Figure 2 shows the ADS-B system in Figure 1 represented as a graph without edge labels. Each component is identified by its node identifier and the directional edges represent information flows.

Algorithms 1 and 2 are used to find all the graphs associated with each label and eliminate the subpaths to obtain unique graphs corresponding to each label. Source code for the algorithms is available at `github.com/anushaat/MSDND`.

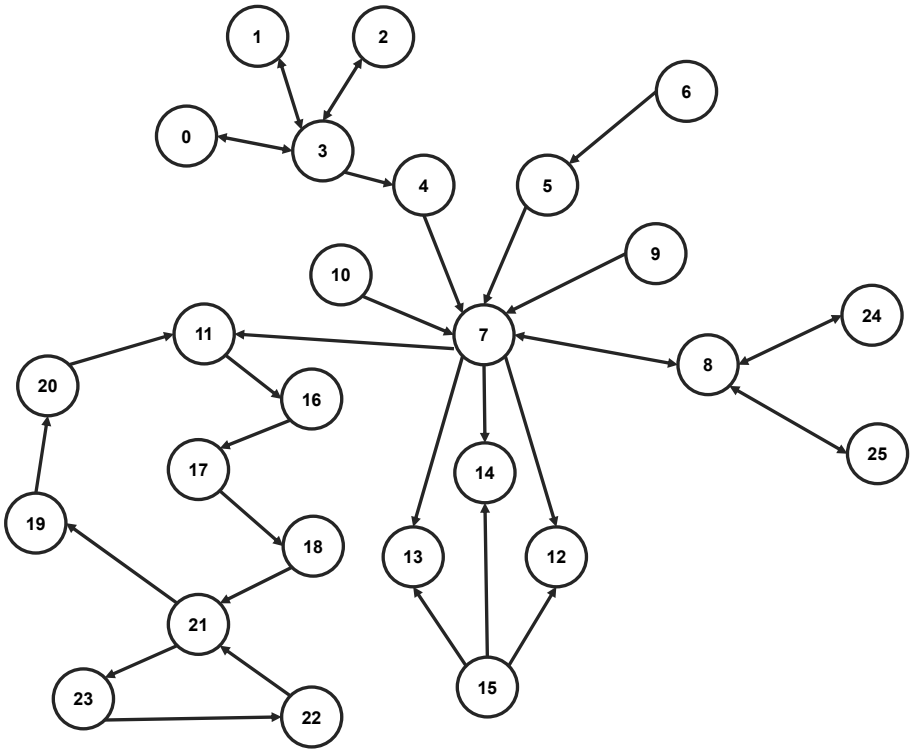

Figure 2. ADS-B system graph.

6.1 Finding Independent Paths

Algorithm 1 uses depth-first search to find all the paths with edges labeled l. The algorithm has a runtime complexity of $\mathcal{O}(\mid V \mid + \mid E \mid)$, where $\mid V \mid$ is the number of nodes and $\mid E \mid$ is the number of edges.

The following theorem proves the correctness of Algorithm 1:

Theorem 1. During a depth-first search of a directed acyclic graph $G = (V, E, L)$, vertex s is a descendent of vertex d iff the search discovers a path from s to d comprising entirely of edges labeled l.

Proof: Assume that a depth-first search is performed on the directed acyclic graph $G = (V, E, L)$ to determine the independent paths for each vertex $v_i \in$ V. It suffices to show that for any pair of distinct vertices $s, d \in V$, the graph G contains an edge from s to d. If $s = d$, then the path from s to d contains only s, which indicates it is source or initial node. If d is an immediate descendent of u, then the path from s

Algorithm 1 : Finding independent paths.

```
Inputs: graph G = (V, E, L); node u
Output: alg1-res (set of graphs associated with a label l ∈ L)

alg1-res ← null
on_path ← null
path ← null
visited ← null
label ← l
function DFS(G, u)
    visited.add(u)
    if on_path.contains(u) then
        alg1-res.add(path)
    else
        on_path.add(v);
        path.push(v);
        for all v ∈ adj[u] do
            if v.labels.contains(label) then
                DFS(G, u)
            end if
        end for
        path.pop()
    end if
    on_path.remove(u)
    return alg1-res
end function
```

to d contains label l. If d is a descendent of s, then all the edges on the simple path from s to d contain label l. □

6.2 Eliminating Subpaths

Algorithm 2 eliminates subpaths that occur more than once. This achieves non-redundancy, which helps identify the faulty component or the component under attack.

Algorithm 2 operates on the sorted result set (alg1-res) provided by Algorithm 1. The if-statement in the outer for loop obtains the longest path of length k and saves the paths with length k in the final result without any processing. The first inner for-loop uses a variant of the sliding window technique to obtain the non-redundant paths and save them in the result. The return statement returns the non-redundant paths in the list of paths provided by Algorithm 1. The runtime complexity of the algorithm is $\mathcal{O}(nmk)$ where n is the size of the result set, m is the subpath length and k is the sliding window size.

Algorithm 2 : Eliminating subpaths.

Input: alg1-res (set of graphs associated with a label $l \in L$ (Algorithm 1 result))
Output: alg2-res (set of independent paths associated with a label $l \in L$)

```
function EliminateSubPaths(alg1-res)
    alg2-res ← null
    maxSize ← alg1-res.get(0).size()
    for path ← 0 to alg1-res.size() do
        S ← alg1-res.get(path)
        k ← alg1-res.get(path).size()
        if k==maxSize then
            alg2-res.add(S)
        end if
        for all s in alg2-res do
            count ← 0
            N ← s.size()
            for i ← 0 to N-k+1 do
                if s(i)==alg1-res.get(i) then
                    for j ← 0 to k do
                        if s(i+j)==alg1-res.get(j) then
                            count++
                        else
                            break
                        end if
                    end for
                end if
            end for
            if count!=k then
                alg2-res.add(alg1-res.get(i))
            end if
        end for
    end for
    return alg2-res
end function
```

The following theorem proves the correctness of Algorithm 2:

Theorem 2. Let $G = (V, E, L)$ be a directed acyclic graph. Let l_m be a label in L based on which the graph traversal is done. The algorithm produces nonempty subpaths S_p with edges containing values associated with l_m in the result of Algorithm 1 (alg1-res).

Proof: Let S_p be the maximum-size subset of paths associated with label l_m in the result of Algorithm 1 (alg1-res). Let $(n_i, ..., n_j, ..., n_k)$ where $i < j < k$ be the set of nodes connected by edges E_p associated with label l_m in S_p. If the edges connecting $n_i \rightarrow n_j$ and $n_j \rightarrow n_k$ are equal

(i.e., $e_{ij} = e_{jk}$), then the result contains paths with label l_m because e_{jk} is an element of edge set E_p associated with label l_m. If $e_{ij} \neq e_{jk}$, let the edge set $E_p^{'} = E_p - \{e_{jk}\} \cup \{e_{ij}\}$. Upon substituting e_{ij} for e_{jk}, $E_p^{'} = E_p$ is obtained, which shows that e_{ij} and e_{jk} belong to the same set (i.e., have the same label). This is true because n_i is a child of n_j and n_k is a child of n_j. Since $E_p^{'} = E_p$, it can be concluded that $E_p^{'}$ contains edges with label l_m, and it includes e_{jk}. □

7. MSDND Analysis

This section presents an analysis of the vulnerabilities associated with the ADS-B air traffic surveillance system. This is accomplished in two steps. The first step is to identify the compromised component. The second step is to employ the graph-based model to identify information flow paths and use the MSDND model to identifying the faulty paths associated with the ADS-B system.

Two scenarios are presented in this section, altimeter failure and satellite failure. They demonstrate that MSDND is an effective tool for identifying system vulnerabilities by analyzing information flow paths. The analyses could be performed at design time to enable security mechanisms to be proactively implemented to handle component failures or attacks.

7.1 Altimeter Failure

This section analyzes an altimeter failure scenario. The result is expressed by the following theorem:

Theorem 3. In the event of an altimeter compromise, automated graph-based analysis can show that the MSDND model yields deducibility. This means that, despite the altimeter failure, critical information can flow to the pilot and air traffic control.

Proof: Consider a scenario in which the barometric altimeter is faulty and sends incorrect altitude data to the pilot. Specifically, the altimeter displays incorrect altitude values, which makes them nondeducible to the pilot.

Between ten to 60 graphs are associated with each label. The graphs are generated based on three labels: (i) *barometric_altitude*; (ii) *altitude*; and (iii) *airspeed*.

Figures 3 through 5 show three graphs generated with respect to the label *altitude*. For example, Figure 3 presents the information flow with

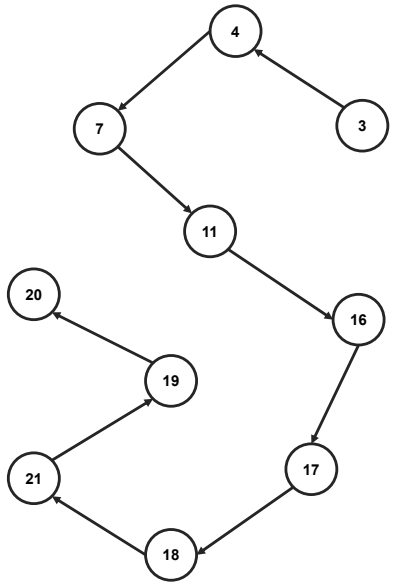

Figure 3. Graph 1.

respect to *altitude* from node 3 (GPS Antenna) to node 20 (TSI-B Antenna). After the graphs are generated, a variant of the sliding window technique is used to eliminate subpaths from the set of graphs.

Graph 3 in Figure 5 is a subpath of Graph 2 in Figure 4. Algorithm 2 is used to eliminate occurrences such as Graph 3 from the result set. After eliminating the subpaths, the value associated with each label is evaluated to check for consistency. If an inconsistency exists, then the in-degree of each node in the inconsistent set is computed. If a node has an in-degree of two or more (indicating that more than one information flow path carries similar information), then it is considered to have a valuation function that eventually breaks the nondeducibility property. If a valuation function exists for a node, then the incoming edges are evaluated to identify the faulty source.

Applying the MSDND model yields two security domains (sources of data) in this scenario: (i) SD^{BA} (Barometric Altimeter domain) and (ii) SD^{GPS} (GPS Satellite domain). Combining the valuation functions in SD^{BA} with respect to the *altitude* value from the Pilot domain yields:

$$S_{ba} = \neg\varphi_6 \wedge \neg\varphi_5 \wedge \neg\varphi_7 \wedge \neg\varphi_{15} \Rightarrow \nexists V^P_{\sim a} \tag{5}$$

Since the information from the Barometric Altimeter domain is faulty, the Pilot domain cannot evaluate the correctness of the *altitude* data.

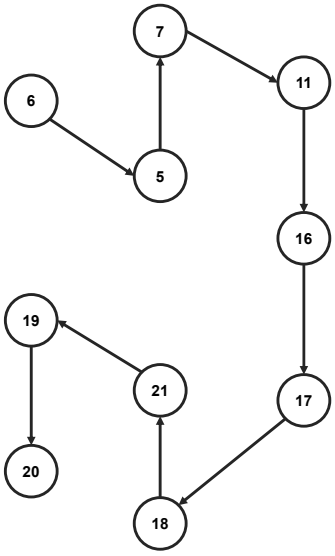

Figure 4. Graph 2.

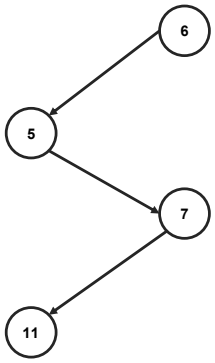

Figure 5. Graph 3.

Combining the valuation functions in SD^{GPS} with respect to the *altitude* value from the Pilot domain yields:

$$S_{gps} = \varphi_3 \wedge \varphi_4 \wedge \varphi_7 \wedge \varphi_{15} \Rightarrow \nexists V_a^P \tag{6}$$

Although the information received from the GPS Satellite domain is not faulty, the Pilot domain cannot evaluate the correctness of the *altitude* data because there are just two information flow paths.

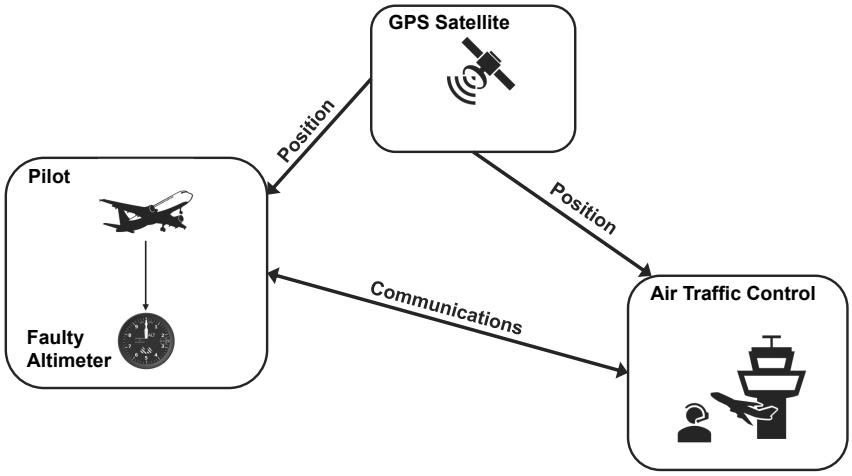

Figure 6. Pilot can deduce the correct altitude value.

According to Equations (5) and (6), the Pilot domain sees two different information flow paths that result in different *altitude* values. Combining the two equations yields:

$$MSDND(ES) = \exists w \in W : [w \vdash \Box f(S_{ba}, S_{gps})] \\ \wedge [w \models (\nexists V^P_{\sim a} \wedge \nexists V^P_a] \tag{7}$$

Therefore, the Pilot domain cannot deduce that the Barometric Altimeter domain is faulty and is sending incorrect *altitude* data. This situation can be resolved by having an additional information flow path that enables the Pilot domain to resolve the conflict.

The additional information flow path comes from the Air Traffic Control domain, which is responsible for sending *altitude* data:

$$S_{atc} = \varphi_{21} \wedge \varphi_{19} \wedge \varphi_{20} \wedge \varphi_{11} \wedge \varphi_7 \wedge \varphi 15 \Rightarrow \exists V^P_a \tag{8}$$

Combining Equations (5), (6) and (8) yields:

$$MSDND(ES) = \exists w \in W : [w \vdash \Box f(S_{ba}, S_{gps}, S_{atc})] \\ \wedge [w \models (\nexists V^P_{\sim a} \wedge \exists V^P_a \wedge \exists V^P_a] \tag{9}$$

Hence, the ADS-B system is not MSDND secure and the Pilot domain can deduce the correct *altitude* value and resolve the conflict by relying

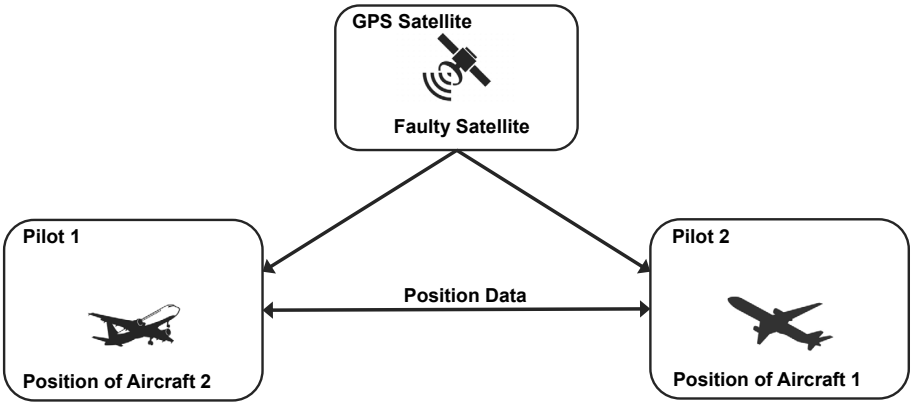

Figure 7. Satellite failure.

on the alternate information flow path from the Air Traffic Control domain (Figure 6). □

7.2 GPS Satellite Failure

This section analyzes a GPS satellite failure scenario. The result is expressed by the following theorem:

Theorem 4. In the event of a GPS satellite failure, automated graph-based analysis can show that the MSDND model yields nondeducibility. This means that critical information cannot flow to the pilots.

Proof: Consider a scenario where a GPS satellite responsible for sending position data is faulty and sends incorrect position data to two aircraft. Assume that the pilots of the two aircraft can communicate with each other. The potential for incorrect decisions by the pilots and a mid-air collision exist if the two pilots cannot identify the source of the incorrect data (Figure 7).

Pilots trust the position data sent by a GPS satellite. When two aircraft are near each other, both the pilots communicate based on the data received from the GPS satellite. This scenario has the Pilot 1 domain and the Pilot 2 domain. If pilots in the two domains cannot identify the problem soon enough, there could be a breakdown in safe aircraft separation. Automated MSDND analysis can enable the pilots to check the consistency of the information flow paths and identify the source of the faulty data.

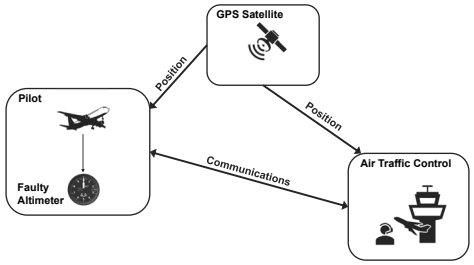

Figure 8. Graph 4.

Figure 8 shows the subgraph generated with respect to the labels *satellite_signals*, *lat* and *long* corresponding to the position data. After the graph is generated, a variant of the sliding window technique is used to eliminate the subgraphs from the set of graphs. Algorithm 2 is then applied to eliminate the subgraphs to avoid redundancy. In this case, no subgraphs exist and Algorithm 2 does not reduce the set of paths.

After eliminating the subgraphs, the value associated with each label is evaluated to check for consistency. If an inconsistency exists, then the in-degree of each node in the inconsistent set is computed. If a node has an in-degree of two or more (indicating that more than one information flow path carries similar information), then it is considered to have a valuation function and eventually breaks the nondeducibility property. If a valuation function exists for a node, then the incoming edges are evaluated to identify the faulty source.

In this case, none of the nodes have an in-degree greater than two. Therefore, the pilots of the two aircraft cannot deduce that the GPS satellite failure is causing the transmission of incorrect information.

The two security domains in this scenario are SD^{P1} (Pilot 1 domain) and SD^{P2} (Pilot 2 domain). After the flight position is retrieved, the Pilot 1 domain trusts the information sent by the Pilot 2 domain and vice-versa. However, the Pilot 1 domain and Pilot 2 domain cannot identify the problem until they are too close, leading to a breakdown in aircraft separation.

Combining the states in SD^{P1} with respect to the Pilot 1 domain yields:

$$S_{p1} = \neg\varphi_{a1} \wedge \neg\varphi_{a2} \wedge \neg\varphi_{a3} \wedge \neg\varphi_{a4} \wedge \neg\varphi_{a7} \wedge \neg\varphi_{a15} \Rightarrow \nexists V^{P1}_{\sim pos} \qquad (10)$$

Since the information received from the Pilot 2 domain is faulty, the Pilot 1 domain cannot evaluate the correctness of the *position* data.

Combining the states in SD^{P2} with respect to the Pilot 2 domain yields:

$$S_{p2} = \neg\varphi_{b1} \wedge \neg\varphi_{b2} \wedge \neg\varphi_{b3} \wedge \neg\varphi_{b4} \wedge \neg\varphi_{b7} \wedge \neg\varphi_{b15} \Rightarrow \nexists V^{P2}_{\sim pos} \quad (11)$$

Since the information received from the Pilot 1 domain is faulty, the Pilot-2 domain cannot evaluate the correctness of the *position* data.

Combining Equations (10) and (11) yields:

$$\begin{aligned} MSDND(ES) = \exists w \in W : [w \vdash \Box f(S_{p1}, S_{p2})] \\ \wedge [w \models (\nexists V^{P1}_{\sim pos} \wedge \nexists V^{P2}_{\sim pos}] \end{aligned} \quad (12)$$

Therefore, the pilots of the two aircraft cannot deduce that the GPS satellite failure is causing the transmission of incorrect information. Hence, the system is MSDND secure to the pilots and they cannot deduce the true positions of their aircraft. □

8. Conclusions

MSDND works by identifying independent information flow paths and partitioning them into security domains based on the consistency of information flows. The model effectively detects failures and attacks in cyber-physical systems with complex state transitions.

Two scenarios using MSDND in the ADS-B system were discussed, one involving an aircraft altimeter failure and the other involving a GPS satellite failure. The analyses were conducted using various interacting components in an aircraft ADS-B system. Such analyses, when applied to the entire system, enable manufacturers and pilots to identify vulnerabilities at design-time and in real time, respectively.

Future research will focus on automatically defining the optimal number of security domains for attack scenarios. This will be done by clustering system components based on validated information flow paths, where information flow paths qualified as secure would be in the secure security domain and the other information flow paths would be in the non-secure security domain. Each of these security domains would have additional security partitions based on the labels associated with the information flow paths. Research will also attempt to use MSDND to model confidentiality, integrity and availability vulnerabilities. Finally, future research will extend the MSDND model to evaluate cyber-physical risks in other critical infrastructures.

Acknowledgement

This research was supported by the National Science Foundation under Grant No. CNS-1837472 and by the Missouri S&T Intelligent Systems Center.

References

[1] B. Ali, W. Ochieng, A. Majumdar, W. Schuster and T. Chiew, ADS-B system failure modes and models, *The Journal of Navigation*, vol. 67(6), pp. 995–1017, 2014.

[2] S. Amin, A. Cardenas and S. Sastry, Safe and secure networked control systems under denial-of-service attacks, in *Hybrid Systems: Computation and Control*, R. Majumdar and P. Tabuada (Eds.), Springer, Berlin Heidelberg, Germany, pp. 31–45, 2009.

[3] S. Chong, A. Myers, K. Vikram and L. Zheng, Jif: Reference Manual, Department of Computer Science, Cornell University, Ithaca, New York (`www.cs.cornell.edu/jif/doc/jif-3.3.0/manual.html`), 2009.

[4] J. Garson, Modal logic, in *The Stanford Encyclopedia of Philosophy Archive*, E. Zalta (Ed.), Metaphysics Research Laboratory, Center for the Study of Language and Information, Stanford University, Stanford, California (`plato.stanford.edu/archives/fall2018/entries/logic-modal`), 2018.

[5] G. Howser and B. McMillin, A multiple security domain model of a drive-by-wire system, *Proceedings of the Thirty-Seventh Annual IEEE Computer Software and Applications Conference*, pp. 369–374, 2013.

[6] G. Howser and B. McMillin, Using information flow methods to analyze the security of cyber-physical systems, *IEEE Computer*, vol. 50(4), pp. 17–26, 2017.

[7] A. Humayed, J. Lin, F. Li and B. Luo, Cyber-physical systems security – A survey, *IEEE Internet of Things Journal*, vol. 4(6), pp. 1802–1831, 2017.

[8] Y. Kim, J. Jo and S. Lee, ADS-B vulnerabilities and a security solution with a timestamp, *IEEE Aerospace and Electronic Systems*, vol. 32(11), pp. 52–61, 2017.

[9] Y. Liu, P. Ning and M. Reiter, False data injection attacks against state estimation in electric power grids, *ACM Transactions on Information and System Security*, vol. 14(1), article no. 13, 2011.

[10] M. Manesh and N. Kaabouch, Analysis of vulnerabilities, attacks, countermeasures and overall risk of the Automatic Dependent Surveillance-Broadcast (ADS-B) system, *International Journal of Critical Infrastructure Protection*, vol. 19, pp. 16–31, 2017.

[11] D. McCallie, Exploring Potential ADS-B Vulnerabilities in the FAA's NextGen Air Transportation System, Graduate Research Project, Department of Electrical and Computer Engineering, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio, 2011.

[12] D. McCallie, J. Butts and R. Mills, Security analysis of the ADS-B implementation in the next generation air transportation system, *International Journal of Critical Infrastructure Protection*, vol. 4(2), pp. 78–87, 2011.

[13] Y. Mo and B. Sinopoli, Secure control against replay attacks, *Proceedings of the Forty-Seventh Annual Allerton Conference on Communications, Control and Computing*, pp. 911–918, 2009.

[14] F. Pasqualetti, F. Dorfler and F. Bullo, Cyber-physical security via geometric control: Distributed monitoring and malicious attacks, *Proceedings of the Fifty-First IEEE Conference on Decision and Control*, pp. 3418–3425, 2012.

[15] F. Pottier and V. Simonet, Information flow inference for ML, *Proceedings of the Twenty-Ninth ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pp. 319–330, 2002.

[16] A. Sabelfeld and A. Myers, Language-based information flow security, *IEEE Journal on Selected Areas in Communications*, vol. 21(1), pp. 5–19, 2003.

[17] V. Simonet, Flow Caml in a nutshell, *Proceedings of the First Applied Semantics II Workshop*, pp. 152–165, 2003.

[18] M. Strohmeier, M. Schafer, V. Lenders and I. Martinovic, Realities and challenges of NextGen air traffic management: The case of ADS-B, *IEEE Communications*, vol. 52(5), pp. 111–118, 2014.

[19] A. Teixeira, S. Amin, H. Sandberg, K. Johansson and S. Sastry, Cyber security analysis of state estimators in electric power systems, *Proceedings of the Forty-Ninth IEEE Conference on Decision and Control*, pp. 5991–5998, 2010.

[20] A. Thudimilla and B. McMillin, Multiple security domain nondeducibility air traffic surveillance systems, *Proceedings of the Eighteenth IEEE International Symposium on High Assurance Systems Engineering*, pp. 136–139, 2017.

[21] H. Yang, M. Yao, Z. Xu and B. Liu, LHCSAS: A lightweight and highly-compatible solution for ADS-B security, *Proceedings of the IEEE Global Communications Conference*, 2017.

[22] H. Yang, Q. Zhou, M. Yao, R. Lu, H. Li and X. Zhang, A practical and compatible cryptographic solution to ADS-B security, *IEEE Internet of Things Journal*, vol. 6(2), pp. 3322–3334, 2018.

[23] S. Zdancewic, Challenges for information flow security, *Proceedings of the First International Workshop on Programming Language Interference and Dependence*, 2004.