Security Requirements Analysis of ADS-B Networks

Thabet Kacem, Duminda Wijesekera, Paulo Costa
Center of Excellence in C4I
George Mason University
Fairfax, Virginia
[tkacem, dwijesek, pcosta]@gmu.edu

Alexandre Barreto
Instituto de Controle do Espaço Aéreo
Centro Tecnológico da Aeronáutica
São José dos Campos, SP - Brazil
barretoabb@icea.gov.br

***Abstract*— Due to their many advantages over their hardware-based counterparts, Software Defined Radios are becoming the new paradigm for radio and radar applications. In particular, Automatic Dependent Surveillance-Broadcast (ADS-B) is an emerging software defined radar technology, which has been already deployed in Europe and Australia. Deployment in the US is underway as part of the Next Generation Transportation Systems (NextGen). In spite of its several benefits, this technology has been widely criticized for being designed without security in mind, making it vulnerable to numerous attacks. Most approaches addressing this issue fail to adopt a holistic viewpoint, focusing only on part of the problem. In this paper, we propose a methodology that uses semantic technologies to address the security requirements definition from a systemic perspective. More specifically, knowledge engineering focused on misuse scenarios is applied for building customized resilient software defined radar applications, as well as classifying cyber attack severity according to measurable security metrics. We showcase our ideas using an ADS-B-related scenario developed to evaluate our research.**

Keywords— Ontologies, Misuse case, Cybersecurity, ADS-B

I. INTRODUCTION

Since first proposed in 1991 by Joe Mitola [1], the concept of Software Defined Radio (SDR) has received considerable research interest. The idea of migrating hardware-based radio functionality to software and, among other benefits, dynamically optimize the spectrum use is compelling. Not surprisingly, the concept is now applied to applications whose focus spans from cognitive radios to radar applications.

In particular, *Air Traffic Control* (ATC) systems research became a natural area for SDRs, due to a pressing need for modernizing its current standards, most stemming from the 1970s. In this context, *Automatic Dependent Surveillance-Broadcast* (ADS-B) has emerged as the leading technology for radar surveillance, and has been already deployed in Europe, Canada and Australia. The U. S. Federal Aviation Administration plans to have it deployed by 2020 as part of NextGen [2]. Most aircraft manufacturers are already equipping their newest models with ADS-B, which is present in aircraft such as the Boeing 777 and the Airbus A380.

In spite of its success, ADS-B has several critics. Strohmeier et al. [3] point out the huge lack of security caused by the fact that ADS-B sends its packets in clear text, making it vulnerable to attacks that target the confidentiality, integrity, availability and non-repudiation properties of the data. This concern is consistent with research done by Costin and Francillon [4] and Schäfer et al. [5], who described the possibility of eavesdropping, man-in-the-middle and denial of service attacks in simulated environments.

Unfortunately, most research efforts aimed at detecting and mitigating ADS-B vulnerabilities lack a systems engineering methodology, therefore failing to address the problem from a holistic perspective. For instance, many lack a comprehensive approach to perform attack analysis and mitigation, and assess their impact on applications of ADS-B technology, which we propose in this paper. Conversely, system engineering techniques such as *use cases* and *interaction diagrams* have been widely used in other domains to model the system's behavior and its interaction with users, which is done from the early designing steps in the system lifecycle.

Misuse cases [6] extend the concept of use case development to model potential undesirable behaviors. The technique has been gaining popularity in recent years as a means to enhance system security, by modeling undesirable behaviors, ensuring these are addressed during systems design. Misuse cases represent the threats to a system at a high level perspective, while the step-by-step details are represented using *mal-activity diagrams*. The latter is key for devising ways of thwarting attacks, but both are essential for designing resilient systems.

Another important technology for designing systems is *Ontology engineering*. Ontologies represent knowledge within a specific domain by formally describing its key concepts and the relationships among them. They allow for automated knowledge management and discovery via logical inferences and have been applied to a variety of applications, such as health care and artificial intelligence. Yet, there has been a surprising lack of research in the ontology community for designing secure SDR applications, and only a few have been proposing to leverage ontologies in this area (e.g. [7]).

Our work bridges this gap by proposing a new methodology for building resilient SDR applications that relies on ontologies. We leverage their reasoning capabilities to automate the modeling of use cases, misuse cases, mal-activity diagrams, mitigation case diagrams and mitigation activity diagrams, all within the design phase of the radar application in question. We present the approach in this paper, and contextualize our ideas using an ADS-B ATC scenario.

Our methodology brings three main contributions to the design of secure ADS-B systems. First, we applied semantic technologies in support to security and requirements modeling,

formalizing knowledge relevant to SDR systems for building resilient radar networks. To the best of our knowledge, this is the first approach to do so. Other research efforts that leveraged security ontologies either focused on security in general, such as [8], or on a specific domain, such as [9]. The work in this paper formalizes the knowledge of secure ADS-B systems in a way that can be extended to other SDR applications.

The second major contribution of this paper is the application of inferential reasoning to enhance security-related design activities. Examples include automated verification of whether the mal-activity and mitigation activity diagrams are consistent with misuse case and mitigation case diagrams respectively, and checking whether the mitigation techniques can effectively thwart the potential attacks. In this initial work, we used Protégé [10] to develop our ontology and the Pellet reasoner [11] to support the automated verification.

The third major contribution of our paper is the development of measurable security metrics to classify the detected attacks according to a taxonomy that we also define in this paper. We use the work in [12] as a reference when developing the metrics we defined for ADS-B applications.

The rest of the paper is organized as follows. Section II presents background information about ADS-B and enumerates some of the security issues discussed in the literature. Section III describes our methodology. Section IV illustrates the use of our methodology by presenting an application scenario. In Section V, we show how to classify the detected attacks using ontological rules and security metrics. Section VI describes related work in securing ADS-B applications, misuse cases, and mal-activities in security-related operations. Section VII has our conclusions.

II. BACKGROUND

One of the main contributions of ADS-B to ATC is its ability to provide better coverage, flexibility, cost-effectiveness, and simplicity than traditional radar. Further, ADS-B can either extend radar coverage or provide a similar service in locations without radar coverage - such as oceanic routes. It also reduces cockpit activity, since pilots would not need to provide constant updates. The costs involved in deploying and operating an ADS-B station are much lower than those observed in traditional radar stations [13].

The ADS-B protocol has two modes of operation: ***ADS-B Out*** and ***ADS-B In***. The first broadcasts aircraft position along with other data over the 1090MHz frequency band for commercial flights and 948MHz band for general aviation. ADS-B packets are encapsulated in *Mode S Extended Squitter* frames consisting of an 8 bits preamble used for synchronization and a 56 or 112 bits data block containing the ADS-B data. It is modulated via *Pulse Position Modulation* (PPM) at 1 Mbit per second rate. *ADS-B In* receives broadcasts from nearby aircraft. This feature is mostly used by ATC services so its deployment is not mandatory to aircraft.

ADS-B presents considerable advances when compared to *Primary Surveillance Radar* (PSR), which determines the approximate aircraft position by measuring the time a reflected pulse takes to reach back to its originating radar antenna. Because the emitted pulse is many orders of magnitude greater than the incoming reflected pulses, radar circuitry is extremely complex. ADS-B also has an advantage over *Secondary Surveillance Radars* (SSR), which relies on aircraft-borne transponders to transmit their positions. Unlike ADS-B, SSR must rely on cooperation by pilots and – mostly for that reason - its operation tends to be error-prone.

In spite of these advantages, ADS-B has its own share of limitations due to its vulnerability to cyber attacks. Several publications on ADS-B security (e.g. [4], [5] and [14]) used a simulated environment to demonstrate various types of attacks targeting this technology, mostly using low cost equipment. The primary source of vulnerability is that data is sent in clear text, without authentication or encryption. Some of the ADS-B attacks demonstrated in simulated environments are:

- **Eavesdropping:** performed with low cost radio devices operating at 1090 MHz combined with an open source implementation of ADS-B receiver. Basically, one can eavesdrop on all air traffic within the range of the radio device. Although eavesdropping is technically not an attack by itself, it is a prerequisite step for many others.
- **Injection Attacks:** performed by an attacker that emits ADS-B messages referencing a fake aircraft (i.e. "injects" a fake aircraft) that interacts with the trajectory of a real aircraft, forcing its pilot and the ATC services to adopt unintended actions to avoid collisions. These attacks usually rely on a preceding eavesdropping phase for capturing the required parameters.
- **Denial of Service:** these are basically a "brute force" version of injection attacks, if less elaborate. The idea is to insert a large number of fake aircrafts to the ATC's screen, causing a denial of service. Air traffic controllers will not be able to distinguish fake aircraft from real ones, or to prevent system crashes due to the heavy load.
- **Man in the middle:** these can be variations of the above, but with a person in the control of the attack. It is possible for an attacker to intercept live traffic, store ADS-B packets, modify them and retransmit the tampered ones back to create confusion in air traffic control operations.

III. METHODOLOGY

The main goal of our work is to help the software architect in designing the core system components with security as a first class citizen, instead of an afterthought. A key concept is our reliance on ontologies to provide the ADS-B system designer with an automated way of testing the security features in a cohesive fashion. We adopted Protégé [10] in this research due to its popularity and built-in reasoners, such as Pellet [11] - which we use to verify the correctness of the attack mitigation techniques. Figure 1 shows a high-level view of our methodology, and highlights the input it requires from the systems engineer. More specifically :

- Use case diagrams: system functionalities.
- Misuse case diagrams: undesired functionalities.

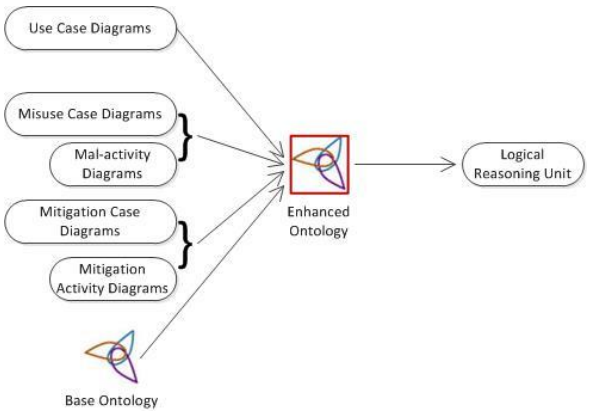

Figure 1: High level view of the methodology

- Mal-activity diagrams: sequence of actions refining a Misuse case.
- Mitigation case diagrams: counteractive functionalities that detect or mitigate undesired functionalities.
- Mitigation activity diagrams: sequence of actions that refine a Mitigating use case.
- Base ontology: Models classes, their relationships, and properties of the SDR domain.

Figure 2 shows the main concepts we have included in the base ontology. Our methodology precisely defines the meaning of "security" by specifying security in terms of desired and undesired system behavior. The proposed design process involves supporting the system designer to model the required and undesirable system functionalities using the classes, inter-class and intra-class relationships, and properties of the base ontology to produce the input listed in Figure 1. Logical reasoning is used in the process to ensure that the design entails the desired and undesired system properties, empowering the design team with an automated verification of the fact that their design is compliant with the design's security objectives (i.e. design by contract). The process outcomes can also be used as formal, accountable artifacts that can be independently verified.

IV. EVALUATION

To evaluate if our ideas would result in a more secure ADS-B network, we have designed an ATC scenario and developed an ADS-B application for this scenario using the proposed methodology. Our scenario includes a network topology consisting of the following components:

- Helicopters: the scenario focus on a helicopter operation that is supported by an ADS-B network.
- ATC Center: one main ATC Terminal receives aircraft track information from a radar and an ADS-B server to provide navigation instructions to the helicopters.
- ATC server: receives location updates from the remotely connected telecommunication sites and ADS-B stations. It stores the updates in a database and evaluates pre-defined security constraints, such as separation between helicopters.
- Telecommunication Sites: the scenario includes three (named T1, T2 and T3), which convey location updates to the ATC server and broadcast navigation instructions to helicopters using ADS-B stations.
- ADS-B Stations: each of the five stations (s1 to s5) receives ADS-B packets from helicopters, broadcasts these to the telecommunication site it is connected to, and forwards navigation instructions to the pilots.
- Communication Links: transmit data between the ATC server, ADS- B stations and telecommunication sites.

Our scenario leverages work such as Magazu [14] shows that attacking ADS-B networks can be relatively simple and inexpensive by purchasing a basic radio device (even a computer dongle) and using open source software such as GNU Radio [15] and Gr-Air-Modes [16] as an ADS-B receiver. In the scenario, the attacker can:

- **Tamper position:** The attacker receives location updates from a legitimate helicopter using an ADS-B receiver placed within the transmission range. Then, modifies ADS-B packets by either altering the hexadecimal content or by inserting GPS coordinates into the packet that may be inconsistent with the logical flight path.
- **Create a ghost helicopter:** The attacker introduces a new helicopter in the range of an ADS-B station so that it gets propagated to the ATC center, and consequently alters flight paths of legitimate helicopters. That is, if the fake trajectory interferes with the real aircraft, this will force active interference by the flight controllers.
- **Flood an ADS-B station:** The attacker overwhelms an ADS-B station with fake packets, affecting the control of helicopters within range of that station. That is, a

Figure 2: Base Ontology

Denial-of-Service attack.

- **Flood ATC/RADAR:** similar to the previous attack but this is done at a larger scale to overwhelm many or all ADS-B stations. If successful, this would adversely affect regional air traffic.

The following mitigations are viable against these attacks:

- **Check Hash:** Embed a hash of the ADS-B payload in the packet to preserve data integrity using pre-shared hashing metadata.
- **Rate Limiting:** Rate control the packets received from communication links of the ADS-B stations.

The core functionalities of the application are represented with use cases. The attacks to be prevented during the design phase are represented using misuse cases, and counteraction techniques are represented using mitigation cases. Taken together, these represent the high-level security objectives of the system.

To achieve security design objectives, our methodology requires more detail from the system architect, who has to define diagrams conveying the activities, mal-activities, and mitigation activities – all consistent with contemporary design activities for large-scale resilient systems. Figure 3 illustrates the combined view of these diagrams. In the figure, every lane is annotated with a name of an actor and the actions. Black ovals indicate mal-activities while white ovals indicate normal or mitigation activities. To facilitate understanding for the methodology, we now provide an overview of each lane:

- Helicopter lane: the black-filled circle designate the start of the "Broadcast location" usage scenario. Ovals "Get self location" and "Broadcast location" designate the two activities that are responsible, respectively, for getting the location of the helicopter and sending it via ADS-B Out. The black rectangle indicates a fork node. It models how location data is broadcasted to all nearby helicopters.
- Attacker lane: the three back ovals show how the misuse case "Tamper position" works. Oval "Receive location" indicates that the attacker received the location update from the legitimate helicopter. Oval "Tamper location" describes how the attacker crafts fake location inside the ADS-B packet while oval "Send fake location" broadcasts the altered packet back to the nearby ADS-B stations.
- ADS-B Station 2 lane: the black rectangle indicates a join node showing how the ADS-B station receives location updates from the helicopter and the attacker. The two ovals "Receive location" and "Send location" in this lane are two activities as part of "Replay Data" use case.
- Telecom Site 1 lane: the two ovals "Receive location" and "Send location" are also part of "Replay Data" use case and show how the ADS-B packets are replayed through the telecom sites.
- Comm Link 1 lane: oval "Transmit" indicates how the data inside the packet is physically transmitted. This activity is part of "Transmit data" use case.
- ATC Server: oval "Receive" designates that the ADS-B packets are received. However, oval "Check Hash" represents a mitigation activity as part of "Check Hash" mitigation case. It indicates that the ATC server checks the received hash against the hash it computes based on the payload of the received packet. The diamond indicates a decision node. Based on the outcome of the computation of the above described condition, the ATC server directs the flow of the whole scenario accordingly. If the result is a mismatch, then it connects to the oval "Discard" which is a normal activity indicating that the ATC server would just ignore the packet before ending the scenario by connecting to the double-edged black circle.
- ATC Center lane: if the result of the previous decision is a match, the oval "Display air traffic" will be connected. This oval is part of "Display air traffic" use case". Similarly, the scenario would end at this point by connecting to the double-edged black circle.

All the elements of Figure 3 can be mapped to the base ontology classes where, each lane is an individual of the Swimlane class and every label has the actor's name. However, this mapping depends on the characteristics of each sub-class of Actor. More specifically:

- Helicopter and Attacker: mapped to the Helicopter class.
- ADS-B station 2: mapped to the ADS-B_Station class.
- The black-filled circle: mapped to an individual of the Initial_Node class
- Double-edged black circle: mapped to an individual of the Final_Node class.
- Black rectangles: can be mapped to either the Join_Node class or the Fork_Node class, depending on the incoming and outgoing arrows. This is modelled by ontological restrictions linking each member of this class to the number of instance of the Node class connected to it.
- White ovals: indicate a normal activity and are

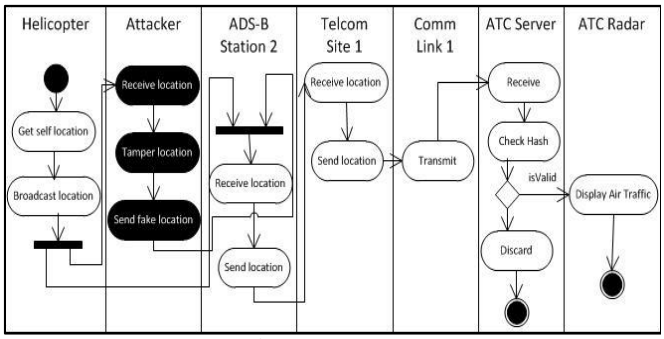

Figure 3: Combined view of activity, mal-activity and mitigation activity diagrams

considered individuals of the Normal_Activity_Node, while those indicating a mitigation activity are considered individuals of the Mitigation_Activity_Node.

- Black ovals: are individuals of the Mal-Activity_Node.

Arrows connecting the elements described above are mapped to object properties that relate two instances of two different classes. In our work, this is done using (Protégé) ontology rules, previously known as Semantic Web Rule Language (SWRL) rules [17]. Each rule implies the *consequent* (right hand side, a.k.a. head) from the antecedent (left hand side, a.k.a. body).

Let π be the statement of Theorem 1, described in Listing 1. It shows a rule that models the fact that "tamper location" misuse case "threatens" the "monitor air traffic" use case. The rule is part of the "threatens" use case / misuse case relationship in the scenario where every helicopter sends an ADS-B packet containing the required information.

Each packet has a location defined as {latitude, longitude, altitude}. When two packets sent from two different helicopters reach the ATC Server, the server compares their timestamps and their locations. If the timestamps are the same and the differences in the received longitudes, latitudes, and altitudes are greater than the predefined ε, then the reasoner will infer the "threatens" object property. The Pellet reasoner then gathers the data and object properties of the individuals concerned by the defined rule, and tries to infer the head - in this case the "threatens" object property. If it succeeds in doing "threatens" will appear as highlighted and we can get the corresponding explanation.

Let ψ be the statement of Theorem 2 provided in Listing 2. It shows the rule proving that the mitigation case succeeds in thwarting the previously detected misuse case that threatens a given use case of the system. It also tags the actor in question as malicious, and associates the attack with its swimlane (cf. Figure 3). Therefore tagging the associated object properties as

```
1   ADSBPacket(?p1), ADSBPacket(?p2),
2    ATC_Server(?a), Helicopter(?h1), Helicopter(?h2),
3   Misuse_Case(Misuse_Case_1),
4   TimedRelation(?tr1), TimedRelation(?tr2),
5   Use_Case(Use_Case_3),
6   broadcastADSBPacket(?h1, ?p1),
7   broadcastADSBPacket(?h2, ?p2),
8   genADSBPacket(?h1, ?p1), genADSBPacket(?h2,
9   ?p2), receives_updates(?h2, ?p1),
10  receives_updates1(?a, ?tr1), receives_updates1(?a,
11  ?tr2), receives_updates2(?tr1, ?p1),
12  receives_updates2(?tr2, ?p2), altitude(?p1, ?alt1),
13  altitude(?p2, ?alt2), icao(?p1, ?id1), icao(?p2, ?id2),
14  latitude(?p1, ?lat1), latitude(?p2, ?lat2),
15  longitude(?p1, ?long1), longitude(?p2, ?long2),
16  time_value(?tr1, ?t1), time_value(?tr2, ?t2),
17  subtract(?dalt, ?alt1, ?alt2), subtract(?dlat, ?lat1,
18  ?lat2), subtract(?dlong, ?long1, ?long2), abs(?adalt,
19  ?dalt), abs(?adlat, ?dlat), abs(?adlong, ?dlong),
20  equal(?t1, ?t2), greaterThanOrEqual(?adalt, 50),
21  greaterThanOrEqual(?adlat, 0.05),
22  greaterThanOrEqual(?adlong, 0.02), →
23  threatens(Misuse_Case_1, Use_Case_3)
```

Listing 1

```
1   ADSBPacket(?p1), ATC_Server(?a),
2   HashRelation(?pi1), Helicopter(?h1), Mal-
3   Activity_Node(Mal-Activity_1), Mal-
4   Activity_Node(Mal-Activity_2), Mal-
5   Activity_Node(Mal-Activity_3),
6   Misuse_Case(Misuse_Case_1),
7   Mitigation_Case(Mitigation_Case_1),
8   Swimlane(?s1), TimedRelation(?tr1),
9   comprise(Misuse_Case_1, Mal-Activity_1),
10  comprise(Misuse_Case_1, Mal-Activity_2),
11  comprise(Misuse_Case_1, Mal-Activity_3),
12  computes_hash1(?a, ?pi1), computes_hash2(?pi1,
13   ?p1), genADSBPacket(?h1, ?p1), occupies(?h1,
14  ?s1), receives_updates1(?a, ?tr1),
15  receives_updates2(?tr1, ?p1),
16  computed_hash(?pi1, ?cHash1), hash(?p1,
17  ?hash1), notEqual(?hash1, ?cHash1) →
18  Attack(TamperPosition),
19  isAssociatedWith(TamperPosition, ?s1),
20  mitigates(Mitigation_Case_1, Misuse_Case_1),
21   participates_in(?h1, Misuse_Case_1),
22  associated_obj_prop(Mal-Activity_1,
23  "receives_updates"^^string),
24  associated_obj_prop(Mal-Activity_2,
25  "genADSBPacket"^^string),
26  associated_obj_prop(Mal-Activity_3,
27  "broadcastADSBPacket"^^string),
28  isActorMalicious(?h1, true)
```

Listing 2

the names of the mal-activities associated with the attack.

The main idea here is that if the ATC Server receives a packet from a helicopter, then it computes its corresponding hash based on the packet's payload and compares it to the hash received in the packet. We assume that the ADS-B packet contains a hash in its payload. If these values match, then the ATC Server proceeds with broadcasting the packet. Otherwise, it marks the helicopter that sent the forged packet as malicious and the "mitigates" object property is proven to be valid. In order to prove this theorem, the Pellet reasoner checks the data and object properties of the individuals concerned by this rule and tries to infer the head of the rule. In this case, if it succeeds in doing so, the object relations in the head appear as highlighted and we can get the explanation of the proof.

V. CLASSIFICATION OF ATTACKS USING METRICS

In this section, we describe the taxonomy we have developed for the message injection ADS-B attacks. It is composed by three classes of attacks, classified based on the difficulty of implementation and the location of the radio device that was used by the attacker. These classes are as follows:

- **Medium-level attacks:** in this type of attacks, the attacker generates the malicious ADS-B messages to be injected in a typically random way and he does not move the equipment used to launch the attack. For instance, the attacker can send a massive amount of ADS-B messages whose locations are within the reach of the ATC Sever with fake aircraft identifier in order to obstruct the view of the radar display and thus prevent the air traffic controller from performing his duties.

- **Advanced-level attacks:** where an attacker uses sophisticated flight simulator programs along with the radio device in order to send a more realistic flight path that cannot be detected as fake easily. For example, one popular program that can be used to achieve this is FlightGear [18]. In this case, the location of the equipment used to perform the attack is fixed.

- **Expert-level attacks:** similar to the advanced-level attacks, except for the fact that the equipment used to launch the attack is located in an aircraft. This kind of attack is harder to test, since it requires sophisticated equipment and procedures.

Classifying attacks detected using the techniques described in the previous section requires collecting parameters, needed for deciding if an attack belongs to a particular attack class modelled in an ontology rule. We leveraged the work in [12], a well-known standard that provided us with a reference for checking under which category our metrics fell into. We defined three security metrics, which are described as follows:

- **Sender Location Difference**: the absolute value of the difference between the triangulated sender's location at two consecutive times t_i and t_j. We assumed having appropriated means of triangulation, which is needed for determining the location of the sender based on the received ADS-B packet. This metric is broken down into three sub-metrics which correspond to the differences between longitudes, latitudes, and altitudes.

- **Velocity**: which is the speed of the aircraft at a time *t*.

- **Estimated-to-Real Difference**: which is the absolute value of the difference between the estimated location of an aircraft and the location retrieved from the ADS-B packet at time *t*. We assumed the capability of estimating aircraft locations at all times. This metric is also composed by three sub-metrics, corresponding to difference of longitudes, latitudes and altitudes.

After careful consideration, we came to a conclusion that these three defined metrics fell under the Cyber Intelligence Threat Analysis category. After all, these metrics collect practical data about the attacks, and allow the security analyst to classify cyber attacks based on patterns of wrong behavior. In our methodology, these metrics are used by the Pellet reasoner to automatically classify the type of attack. The relationships between the described classes of attacks and the security metrics are described as follows:

- **Medium-level attacks**: an attack belongs to this class if the sender location difference and the velocity are equal to zero. An attacker, whose physical location does not change, is of course very likely to have such characteristics. Further, the estimated-to-real difference has to be greater than a predefined threshold for the longitude, latitude and altitude. Consequently, if the location retrieved from the ADS-B packet is not within the aircraft envelope, then such packet most likely comes from an attacker.
- **Advanced-level attacks**: An attack belongs to this class if the sender location difference and the velocity are equal to zero. The estimated-to-real difference would be within the predefined thresholds.
- **Expert-level attacks**: An attack belongs to this class if the velocity is comparable to the one of a real aircraft. Besides, the sender location difference cannot be equal to zero, and the estimated-to-real difference has to be within the predefined thresholds.

We now focus on how the proofs of the theorems are generated using ontological rules, similarly to the previous section of this paper. Due to space limitations, we restrict the explanation to the ontological rules used for computing the estimated-to-real difference metric, and for classifying an attack as belonging to medium-level attacks class respectively in Listings 3 and 4.

In Listing 3, we collect GPS properties of the malicious actor, after verifying that the packet he sent had reached the ATC Server. Then, we compute the properties of the estimated-to-real metric relatively to longitude, latitude and altitude.

In Listing 4, we collect the data provided by all the metrics and define the conditions for deciding whether an observed attack belongs to the medium-level attacks class.

We had to make several assumptions regarding the metrics. Firstly, we assumed that there is a mechanism to triangulate the true location of the sender of the packet, which would facilitate computing its location difference metric. Secondly, we assumed that it is possible to estimate the location of an aircraft at all times, which is required for computing the estimated-to-real difference metric.

For each metric used in this evaluation we have written a corresponding rule that the Pellet reasoner can use against the defined individuals to determine its value. The first rule is related to the sender location difference metric. Basically, it gets the triangulated sender locations at two consecutive time steps and calculates the absolute value of the difference of altitude, latitude and longitude. The second rule addresses the velocity metric, and extracts its value at a certain time by using the *ADSBPacket* and *TimedRelation* entities. The third rule,

```
1  ADSBPacket(?p1), ATC_Server(?a),
2  EstimatedRealDiff(?ed),
3  EstimationRelation(?er1), Helicopter(?h1),
4  TimedRelation(?tr1), estimates_position1(?a,
5  ?er1), estimates_position2(?er1, ?p1),
6  genADSBPacket(?h1, ?p1),
7  receives_updates1(?a, ?tr1),
8  receives_updates2(?tr1, ?p1),
9  estimated_alt(?er1, ?e1), estimated_lat(?er1,
10 ?e2), estimated_lon(?er1, ?e3),
11 altitude(?p1, ?alt1), isActorMalicious(?h1, true),
12 latitude(?p1, ?lat1), longitude(?p1, ?lon1),
13 abs(?adalt, ?dalt), abs(?adlat, ?dlat), abs(?adlon,
14 ?dlon), subtract(?dalt, ?alt1, ?e1), subtract(?dlat,
15 ?lat1,?e2), subtract(?dlon, ?lon1, ?e3) →
16 er_diff_alt(?ed, ?adalt), er_diff_lat(?ed, ?adlat),
17 er_diff_lon(?ed, ?adlon)
```

Listing 3

```
1   Attack(?at), EstimatedRealDiff(?er),
2   SenderLocationDiff(?sl), Velocity(?v),
3   hasMetric(?at, ?er), hasMetric(?at, ?sl),
4   hasMetric(?at, ?v), er_diff_alt(?er, ?er_alt),
5   er_diff_lat(?er, ?er_lat), er_diff_lon(?er, ?er_lon),
6   sender_diff_alt(?sl, ?dl3), sender_diff_lat(?sl,
7   ?dl1), sender_diff_lon(?sl, ?dl2), v_speed(?v, ?vs),
8   equal(?dl1, 0), equal(?dl2, 0), equal(?dl3, 0),
9   equal(?vs, 0), greaterThan(?er_alt, 40),
10  greaterThan(?er_lat, 0.5), greaterThan(?er_lon, 0.5)
11  → attack_category(?at, "Medium"^^string)
```

Listing 4

which deals with the estimated-to-real difference, gets the coordinates of both the estimated position and the real position before calculating the absolute value of the difference in terms of latitude, longitude and altitude.

The knowledge derived from these rules can provide reasonable insights into attack classification. We developed different rules to classify an attack according to its category in the taxonomy. For example, an attack that belongs to the medium-level category would have a null velocity, a null sender location difference, and its estimated-to-real metric would exceed the defined threshold. Conversely, the advanced-level category would have its attacks with a null velocity and a null sender location difference, but its estimated-to real metric would not exceed the defined threshold. This is expected, given the use of flight simulator versus generating random values in the medium-level category. Finally, an attack would be in the expert-level category if the velocity is comparable to a real aircraft, while its sender location difference would be greater than zero and its estimated-to-real metric would not exceed the pre-defined threshold.

VI. Related Work

In [7], Massacci et al. proposed an ontology for security requirements by extending existing ontologies with situational and context awareness. The authors contextualize their ideas by an ADS-B case study. This work is similar to ours but the main difference is that they focused on GPS spoofing attacks, while we address message-injection attacks that are more difficult to realize, as stated by [3].

Oltramari et al. [19] described an approach to support cyber operations by enhancing the situational awareness via a combination of cognitive modelling and ontology engineering. They plan to evaluate their approach by applying it to design a cyber defense application. However, their work is not specific to SDR applications, but to cyber operations in general.

In [8], Obrst et al. presented a methodology for building cyber security ontologies based on a malware ontology. This methodology outlines the steps that are required for building a cyber security ontology, and provide general guidelines for enhancing the cyber security domain with semantic models. The main difference between this work and ours is their focus on security from a general standpoint, starting from a wide characterization of malware. In our paper, we tackle the problem of security within the SDR domain by leveraging knowledge from semantic models and ontologies.

In [9], Ekelhart et al. introduced a framework for building security ontologies that assists in providing risk analysis. The authors used an incremental approach where they start with a generic security taxonomy formalized in an ontology and they enhance it by integrating risk factors, constraints, threats and countermeasures. This work concentrates on risk management involving IT-security tasks in a company, while our goal is to create a methodology to secure ADS-B networks.

In [20], Magklaras and Furnell proposed an approach to address internal IT misuse via a classification of misusers and their motives, as well as the implications of the misuse on the system. In our paper, we adopted a more flexible representation of misuses, which relies on misuse case and mal-activity diagrams. Moreover, their work describes security in general while ours focuses on security in ADS-B networks. The authors provided a mechanism of determining the threat level that is similar to our work, in which we classify the attacks according to the taxonomy. The main difference is that we employ theorem proving with a semantic-web inspired rule system, while their work is based on an analyzer module built as part of their proposed framework.

In [21], McCallie et al. assessed ADS-B security by detecting and classifying attacks that may target ADS-B applications. They provide some general recommendations on how to thwart these attacks. In contrast, we provide a methodology to be applied when analyzing the security of SDR applications.

Similarly, Costin and Francillon [4] demonstrated the lack of security of ADS-B by implementing attacks in a low-cost simulated environment. They did not focus on attack mitigation. In contrast, our methodology assists the systems engineer in formulating security requirements by precisely defining and verifying these for SDR applications, while using automated design verification for attacks and their mitigations.

In [22], Whittle et al. proposed a technique for modeling possible attacks and mitigating them. They employ misuse cases to model undesirable system behavior. The approach models misuse cases as aspects, inserts these in the core system features before integrating mitigation techniques. Then, they use the attacks as test cases to evaluate the design robustness. Although our objective is similar to theirs, but we base our methodology on ontologies to support the system design from the ground up with security as an integral design aspect. In contrast, they use prior work on state machines.

In [23], Sindre introduced the concept of mal-activity diagrams as an enhanced form of activity diagrams where each actor of the system, normal or malicious, occupies a swimlane and starts normal or malicious activity nodes. Our approach uses the concept of a mal-activity diagram and integrates it in the design process with the support of ontologies.

In [24], El-Attar presented a tool to convert a textual description of the system to a model taking into consideration the security aspects in term of misuse case and mal-activity diagram. This is achieved with support from two tools. One transforms the textual description to a context-free grammar,

which is used to build the first meta-model. The other creates the meta-model that captures the mal-activity diagrams. This work appears similar to ours, but El-Attar's main goal is to create meta-models from textual description. In contrast, we formally capture the diagrams using ontological rules and verify that the stated relationships between them exist using a theorem prover.

VII. Conclusion

ADS-B has emerged as a promising technology for optimizing the use of the air space while lowering costs and increasing the security of air traffic operations. Hindering this progress, many security vulnerabilities in the protocol have been discovered, generating a pressing need for a holistic, systems-oriented approach to properly address the problem. Within this context, in this paper we present a methodology that relies on time-tested, traditional requirements engineering while leveraging advanced semantic technology concepts to automate the process of requirement verification. We have tested the methodology in an ADS-B scenario, and were able to support the system design by translating security requirements into formally verifiable claims. Finally, we used logical reasoning to ascertain the validity of the mitigating solutions and classify the attacks using security metrics.

We plan to further evaluate the methodology in complex simulation environments that will provide a better understanding of its broader impact in designing resilient SDR applications. Future work on the methodology also involves standardizing its procedures, so they would be applicable to the field of SDR applications in a consistent fashion. In this paper we have focused on the initial phases of the system engineering life-cycle, but the methodology can be easily expanded to formalize and automate other parts of the systems engineering life cycle. Examples of the latter include supporting trade-off analysis of adding security features against their associated cost, validation and verification of the actual system based on stakeholder requirements (e.g. FAA specs for different types of systems), and others that would benefit from the formalization of the design process with a focus on its security requirements.

References

[1] J. Mitola, "The software radio architecture," *IEEE Communications Magazine*, vol. 33, no. 5, pp. 26–38, May 1995.

[2] "NASA - NextGen," 24-Oct-2014. Available at: http://www.hq.nasa.gov/office/aero/asp/airspace/index.htm.

[3] M. Strohmeier, V. Lenders, and I. Martinovic, "On the Security of the Automatic Dependent Surveillance-Broadcast Protocol," *arXiv:1307.3664 [cs]*, Jul. 2013.

[4] A. Costin and A. Francillon, "Ghost in the Air(Traffic): On insecurity of ADS-B protocol and practical attacks on ADS-B devices," 2012.

[5] M. Schäfer, V. Lenders, and I. Martinovic, "Experimental Analysis of Attacks on Next Generation Air Traffic Communication," in *Applied Cryptography and Network Security*, Springer Berlin Heidelberg, 2013, pp. 253–271.

[6] G. Sindre and A. L. Opdahl, "Eliciting security requirements with misuse cases," *Requirements Eng*, vol. 10, no. 1, pp. 34–44, Jan. 2005.

[7] F. Massacci, J. Mylopoulos, F. Paci, T. T. Tun, and Y. Yu, "An Extended Ontology for Security Requirements," in *Advanced Information Systems Engineering Workshops*, C. Salinesi and O. Pastor, Eds. Springer Berlin Heidelberg, 2011, pp. 622–636.

[8] L. Obrst, P. Chase, and R. Markeloff, *Developing an Ontology of the Cyber Security Domain.* .

[9] A. Ekelhart, S. Fenz, M. Klemen, and E. Weippl, "Security Ontologies: Improving Quantitative Risk Analysis," presented at the 40th Annual Hawaii International Conference on System Sciences, 2007. HICSS 2007, 2007, p. 156a–156a.

[10] H. Knublauch, R. W. Fergerson, N. F. Noy, and M. A. Musen, "The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications," in *The Semantic Web – ISWC 2004*, S. A. McIlraith, D. Plexousakis, and F. van Harmelen, Eds. Springer Berlin Heidelberg, 2004, pp. 229–243.

[11] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, "Pellet: A practical OWL-DL reasoner," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 5, no. 2, pp. 51–53, Jun. 2007.

[12] "MITRE's Making Security Measurable," *MITRE's Making Security Measurable*. Available: http://makingsecuritymeasurable.mitre.org/.

[13] "ADS-B Frequently Asked Questions (FAQs)," 07-Apr-2014. Available at: http://www.faa.gov/nextgen/implementation/programs/adsb/faq/#3.

[14] D. Magazu III, "Exploiting the Automatic Dependent Surveillance-Broadcast System via False Target Injection," 2012.

[15] *GNU Radio*. Available at www.gnuradio.org.

[16] N. Foster, "Gr-air-modes," Available: https://github.com/bistromath/gr-air-modes.

[17] I. Horrocks, P. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean, "SWRL: A Semantic Web Rule Language Combining OWL and RuleML," May 2004.

[18] "FlightGear." Available: www.flightGear.com.

[19] A. Oltramari, C. Lebiere, W. Zhu, L. Vizenor, and R. Dipert, "owards a Cognitive System for Decision Support in Cyber Operations," presented at the International Conference on Semantic Technologies for Intelligence, Defense, and Security (STIDS), 2013.

[20] G. B. Magklaras and S. M. Furnell, "Insider Threat Prediction Tool: Evaluating the probability of IT misuse," *Computers & Security*, vol. 21, no. 1, pp. 62–73, Jan. 2001.

[21] 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, no. 2, pp. 78–87, Aug. 2011.

[22] J. Whittle, D. Wijesekera, and M. Hartong, "Executable misuse cases for modeling security concerns," presented at the ACM/IEEE 30th International Conference on Software Engineering, 2008. ICSE '08, 2008, pp. 121–130.

[23] G. Sindre, "Mal-Activity Diagrams for Capturing Attacks on Business Processes," in *Requirements Engineering: Foundation for Software Quality*, P. Sawyer, B. Paech, and P. Heymans, Eds. Springer Berlin Heidelberg, 2007, pp. 355–366.

[24] M. El-Attar, "From misuse cases to mal-activity diagrams: bridging the gap between functional security analysis and design," *Software Systems Modelling*, vol. 13, no. 1, pp. 173–190, Feb. 2014.