An Approach to Represent and Transform Application-Specific Constraints for an Intrusion Detection System

Ayesha Babar, Fahim Imam, Thomas R. Dean
Queen's University
Kingston, Ontario, Canada
{ayesha.babar,fahim.imam,tom.dean}@queensu.ca

Jose Fernandez
Ecole Polytechnique
Montreal, Quebec, Canada
jose.fernandez@polymtl.ca

ABSTRACT

While the need for newer and more efficient network security techniques is increasing, refining the existing and proven techniques can also have potential benefits. One of the aspects of such improvements in the existing systems is making them flexible to modify. Currently, we have an intrusion detection system (IDS) that defines the normal patterns of a network behaviour using constraints. The IDS dissects the network packets into network information to evaluate the constraints. In this research, we extend the existing IDS to validate constraints defined on application data. We extend the IDS to further dissect the data within the incoming network packets. We define the data constraints to identify possible malicious inconsistencies in the application data of a closed network such as the Air Traffic Control (ATC) as an example. We use an ATC ontology for the ATC domain data representation and threat evaluation. We modify an existing ATC simulator and use it to generate both clean and malicious data. Rules and queries are then developed for these data using the ontology to represent detectable threats. The queries are then transformed into application data constraints readable by the IDS. While the transformation is defined as a manual process, the IDS will be updated with automated transformation in the future. The data constraints are written in the same domain-specific language (DSL) already used for the IDS that ensures real-time performance. In this paper, we present our approach to represent and transform application-specific constraints for our IDS along with examples.

CCS CONCEPTS

• **Security and privacy** → **Network security**; • **General and reference** → **General conference proceedings**; • **Networks** → Network reliability.

KEYWORDS

Intrusion Detection, Data Constraints, Program Transformation

ACM Reference Format:
Ayesha Babar, Fahim Imam, Thomas R. Dean and Jose Fernandez. 2020. An Approach to Represent and Transform Application-Specific Constraints for an Intrusion Detection System. In *CASCON '20: 30th Annual International Conference on Computer Science and Software Engineering, Nov 10–13, 2020, Toronto, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/1122445.1122456

CASCON '20, Nov 10–13, 2020, Toronto, Canada

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Cybersecurity is a vital concern as networks surround all aspects of our lives. While the internet may put our information and identity at risk, closed safety-critical network systems such as air traffic control systems (ATC) or nuclear plants may put lives at risk.

Intrusion detection systems (IDS) monitor networks for malicious behaviour. IDSs emphasize the detection of malicious data at the network level. However, malicious data can also occur at the application layer. For example, the ATC systems have come to rely more on Automatic Dependent Surveillance–Broadcast (ADS–B) to extend radar coverage. However, ADS-B has no authentication, and anyone with a software defined radio can transmit false ADS-B data.

We have previously described an IDS designed for network integrity of closed networks such as ATC [31]. This IDS detects abnormal behaviour at the network level using a constraint engine. In this research, we leverage the IDS to detect the presence of attacks at the application layer. It may be true that the application logic is equipped to deal with possible corruption of data. However, application logic is complex due to the fact that it must both validate and operate on the data. Malicious external data is not always obvious. Adding an additional check on the application level data provides in-depth defense to the system.

We use a simulated ATC system to produce simulated air traffic data. This data is parsed and translated into resource description framework (RDF) [14] graph database, using an ATC ontology. We use SPARQL [15] to develop queries that represent the integrity of the information. We then manually translate the domain level threats to the low-level constraints used by our IDS. This allows us to prototype the transformation, and identify changes needed in the implementation of constraint engine to support application level constraints.

The main contribution of our work are:

- Extension of existing constraint based IDS to identify data integrity.
- A specification of transformation of a threat from natural language to a domain specific language, used to generate a custom IDS.
- Testing and evaluation of data constraints with the existing IDS framework.
- Proposing required extensions in the existing framework for new proposed data constraints.

The structure of the rest of the paper is as follows. Section 2 provides a description of the existing framework, ATC simulation and ATC ontology. Section 3 discusses selected threat scenarios of the research. Section 4 describes the transformation process, followed by Section 5 to illustrate the transformation process. The evaluation of IDS and results are presented in Section 6 followed by the related work in Section 7. We conclude the paper and discuss the future work in Section 8.

2 BACKGROUND

To model the cyber threats we use an ATC simulator developed by Morel [26] to generate the ATC data for our IDS. Originally developed by Hasan et al. [17], the IDS detects intrusions based on anomalous network behaviour. The IDS is based on constraints capable of detecting anomalies in a limited access, closed networks such as ATC. Such networks are characterized by a limited number of protocols which makes it possible to define the normal network behaviour as constraints. The current version of the IDS detects intrusions based on protocol-specific constraints. One of the goals of this research is to extend the IDS to specify application-specific data constraints. The IDS can then detect anomalies in the data carried in the network packets. Figure 1 shows the IDS architecture along with the application level extensions. The modified IDS now implements two kinds of constraints: a) protocol-specific network packet constraints, b) application-specific data constraints. We refer to the former as the network constraints and the later as the data constraints.

Application Data. The application layer supports application and end-user processes. It provides application services for file transfers, e-mail, and other network software services [1]. In our research, ATC application data is generated by the air traffic control simulation. This data is embedded in the captured network packets and is parsed by an application data parser. Examples of application data are the speed of an aircraft or the position of an aircraft detected by radar. An example of a constraint on the data is that the speed of an aircraft is within a given range. The constraints that ensure integrity of the application domain data are referred as application data constraints or data constraints.

Network Data. The data captured by the IDS framework that deals with the network layer is referred to as 'network data' for the purpose of our research. This data is parsed by a network parser, and the constraints that check the integrity in this data are referred to as network constraints. Network constraints are already implemented and evaluated by the IDS. An example of this constraint is that the a publisher in the Real-Time Publish-Subscribe protocol (RTPS) [13] has previously declared that it is a participant.

2.1 The Intrusion Detection System

The input to the IDS framework is a network protocol specification written in the Structure and Context-Sensitive language (SCL) [23]. SCL describes the syntax and the semantic constraints of a given protocol. Since SCL supports both context dependant parsing and specifying general constraints, it is used to generate the two main components of the IDS: the *parser* and the *constraint engine*. The generated custom parser reads the network packets and converts them in a format readable by the constraint engine. The constraint

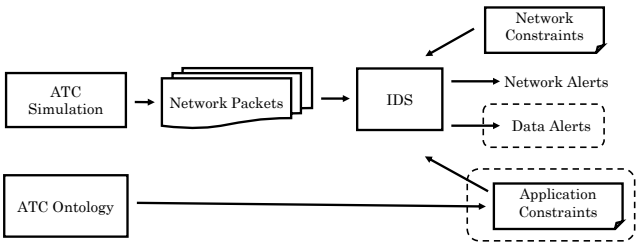

Figure 1: The IDS architecture with extensions.

engine validates these packets against the defined network constraints and generates alerts.

The constraints in SCL are first transformed into an intermediate DSL which describes the constraint tree life-cycle along with memory management. The constraints in the intermediate DSL are then used to automatically generate the constraint engine in C. The DSL describes the constraint tree life-cycle defined by Hasan et al. [16] and has the following four phases: Instantiate (I), Bind (B), Evaluate (E), and Destroy (D). We refer to the DSL as the IBED DSL based on the life-cycle phases. The first, instantiate, occurs when an initial packet of a constraint is encountered. This causes an instance of the internal data structure to be allocated for a constraint tree. The bind phase is used when additional packets are encountered that add information to a constraint tree. The evaluate phase adds the final data to the constraint and evaluates it. Since a constraint may be evaluated multiple times, the destroy phase is used when a packet is encountered that indicates that particular instance of the constraint is redundant. Details about the IBED DSL can be found in Rakha et al. [31].

In this approach, the constraints are intended to validate the last packet in the constraint. The previous packets in the constraint are used to provide needed information to validate the evaluation packet.

2.2 The ATC Simulation

The ATC Simulation is designed and developed by Morel [26] and generates the data used in our research. While the simulation is not a complete representation of an ATC system it provides the necessary components for our research [48]. The ATC is simulated over a closed Data Distributed Service (DDS) network using the RTPS protocol. The main components of the ATC simulation are shown in Figure 2. An existing ATC simulator, Euroscope [8], is used to generate and visualize ATC data using the FSD protocol (FSD and Euroscope in the figure). We use a multiplexer to split the data and transform it to DDS representations of Primary Surveillance Radar (PSR), Secondary Surveillance Radar (SSR), Automatic Dependant Surveillance-Broadcast (ADS-B) data.

2.3 The ATC Ontology

The ATC ontology defines the domain with the help of a controlled and precise vocabulary. When describing the ATC domain, we define the concepts that are present in the domain. For example, *speed* is a concept and it has a meaning and context in our domain. Some concepts can be explained using relations between other

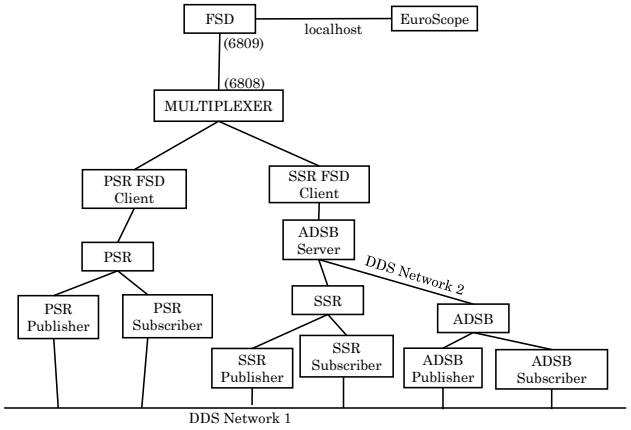

Figure 2: ATC Simulator Architecture Model, adapted from Morel [26]

Figure 3: ATOM detection process, adapted from Corriveau [21].

concepts and/or objects. Some concepts can be described as data structures. The vocabulary used for the ATC domain is concise, and includes classes, sub-classes and relations between them.

The ATC ontology research by Morel [26] is a practical application of the ATOM [21] process. Abstractions-Translation-Ontology-Method (ATOM) is a step-wise method to develop an ontology for a domain specific system.

The ATOM process produces three artifacts: the final ontology, the translation diagram, and the specification document. The final ontology is in the form of a Resource Description Framework (RDF) graph [30]. RDF is the most common way of representing ontologies. In RDF, an ontology is represented as a set of (Subject, Predicate, Object) triplets. The subjects and objects are the nodes of the graph, and the predicate is the property or relation between them. For example, the instance of a concept an aircraft that has a specific speed can be expressed in (PlaneA, hasSpeed, 370).

The ATOM process shown in Figure 3 is used to develop an approach to anomaly detection in the ATC domain. After examining the application data in the network packets, the concepts are identified, and the RDF model is created. The RDF model is then used to initiate the ontology. The nodes are the entities (e.g. airplane, radar, speed) and the edges are the relations between the entities. Further reasoning and flexibility can be added to the ontology by applying rules. The final stage is a querying mechanism, which is used to retrieve and update the information in the ontology. The query language we use is SPARQL (SPARQL Protocol and RDF Query Language) [27]. We have extended Morel's initial ATC ontology[1] for our research.

[1] Available at http://pyxis.ece.queensu.ca/graph/atc/ontologies/atc.owl

New modules have been added to the original ontology to support the representation of Flight Plan data, PSR report, and SSR report. The core ontology is modified to provide better organization to navigate its classes and properties. The current ontology provides the logical framework to consistently describe, query, and reason about different ATC attacks, including the types of attacks described in this paper. The ontology currently includes 72 classes, 39 object properties, 40 data properties, and 250 logical axioms.

2.4 IDS and the ATC Simulator Extensions

To support the evaluation of application-specific data constraints we have extended the existing IDS architecture as shown in Figure 4. An application protocol specification is used to generate a parser for the application specific data encoded in the network data. The ontology from the ATOM Process is shown in the upper right. It is used to initialize the graph database and also to derive a mapping specification that identifies the relationship between the low level data in the packets and the primitive entities and relations present in the data. The RDF mapping is used to automatically generate an RDF translator that populates the graph database with primitive entities and relations. This database can be enhanced with rules and a set of queries are identified that should be continuously evaluated by the constraint engine. These extensions in the upper box have been completed previously.

This paper describes the extensions in the lower box. We transform the queries to a set of application level constraints which is used to generate the application level constraint engine. This is currently a manual transformation and we are working to automate this transformation in the future.

The ATC Simulation was updated to take flight plan information from EuroScope and model as flight strips in the simulated ATC network. It was also updated to allow scripts that inject fake ADS-B data into the simulation.

3 THREAT SCENARIOS

The nature and requirements of command and control systems such as ATC differ from traditional IT systems. Cerchio et al. [9] identify the primary requirements of ATC systems as Integrity and Availability. Cerchio et al. also claim that airborne and seaborne environments are not often considered in security research. While the ground part of an ATC system is a closed network, it still receives outside information without verification. Threats against open communication networks are related "mainly to message insertion (confidentiality), modification (integrity) or suppression (availability)" [34]. Thus, ATC systems are vulnerable to potential attacks some of which are targeted directly at message integrity.

Automatic Dependent Surveillance-Broadcast (ADS-B) has become a key component of ATC systems. The U.S. Federal Aviation Administration (FAA) has required certain aircraft to have installed ADS-B by January 2020 [29]. The threats we consider in this paper are based on this mandate and are information attacks on ADS-B. Balduzzi et al. [2] identify several threats against Automated Identification System (AIS) a system similar to ADS-B for ships. AIS and ADS-B are examples of security critical networks. Both transmit information periodically and are enhance the situational awareness of entities in the system. ADS-B and AIS are subject to attacks

of the same nature, which are to intercept, modify, or delete the messages [22]. We use the same categorization as Balduzzi et al. [2] due to similarities between AIS and ADS-B. Their categorization shows that the attacks can be done at two levels: software and radio frequency (hardware). Among the software identified threats, spoofing and hijacking are two major categories, and both can be modelled under the data related attacks.

We implemented three threat models: Ghost Plane, Physical Law Violation and Spoofed Location. They represent breach to the integrity, confidentiality and authentication of the system. The ghost plane threat scenario simulates malicious ADS-B data of an aircraft that doesn't exit. The threat can be detected if it is within the range of primary radar, is at an altitude that is not in the radar shadow, but is not detected by the radar. This plane can have SSR or ADS-B updates. If outside the range of radar, a ghost plane can be detected if it violates the law of physics. That is, if it descends or ascends faster than the aircraft category, or turns too fast or too slow, or has a speed outside the range of the aircraft type.

Another attack is to monitor existing ADS-B broadcasts for an aircraft and immediately broadcast a new position that overrides the real position. This attack can be detected based on the time intervals of the ADS-B messages.

4 TRANSFORMATION PROCESS

Figure 5 shows the artifacts involved in both the IDS framework and the ATOM process. For both, the network packets generated by the ATC simulator are first parsed into useful data structures. The IDS uses the constraints defined in IBED DSL and protocol specification in SCL, and auto-generates the *C* code for the constraint engine. The constraint engine uses the generated *C* code to evaluate the constraints and ensures network integrity in real-time. The ATOM process translates the parsed network packets into RDF triples using a RDF translator. The resulting RDF is be stored in a graph database. SPARQL [15] queries are used to analyze and understand different aspects of data. The main purpose for these queries is to diagnose and investigate the packet data for constraints that can be used to assess the health of the data. We first transform the SPARQL queries from the ATOM process into SCL constraints and

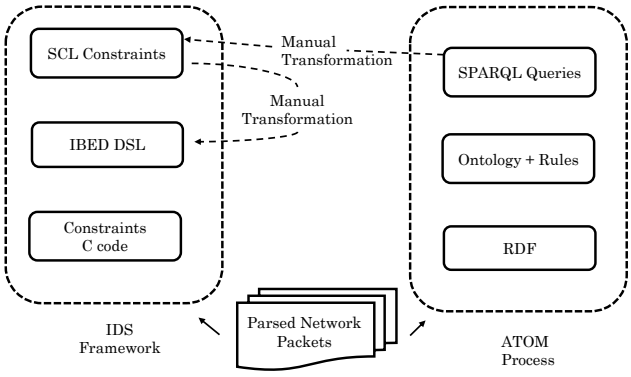

Figure 5: Transformation of SPARQL queries to IBED DSL constraints.

then transform the SCL constraints into the IBED DSL for our IDS framework.

SPARQL queries are used for exploratory purposes by domain experts to formulate the constraints at a high level. The IBED DSL constraints are used to detect intrusions at run time by the constraint engine. Both SPARQL and IBED DSL queries represent the threat in the application data domain which can be expressed in First Order Logic (FOL). The complete transformation process consists of six artifacts as shown in Figure 6.

The first artifact is a query specification in Natural Language. Each of the steps between the artifacts up until artifact 5 (IBED DSL of constraints) are currently manual in nature. The final step, used to generate the *C* code is partially automated. We now describe each step below and explain the involved representations.

Step 1: Query Specification in Natural Language.

We start by naming the queries that represent the respective threat. We define them as a concise statement in natural language. We try to remove as much syntactic or lexical ambiguities as possible. This definition helps in the true representation of the query.

This definition provides a basis for the FOL representation of the queries in the next step.

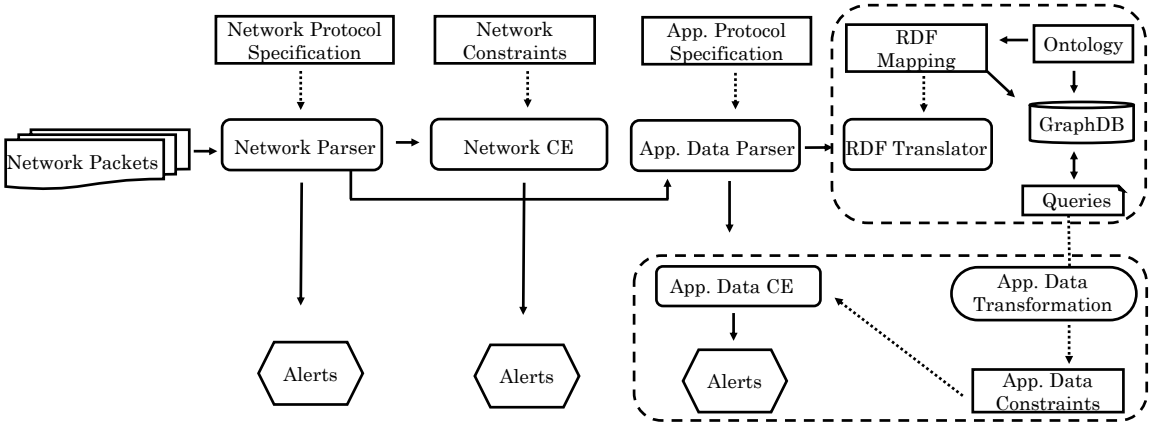

Figure 4: IDS runtime framework architecture.

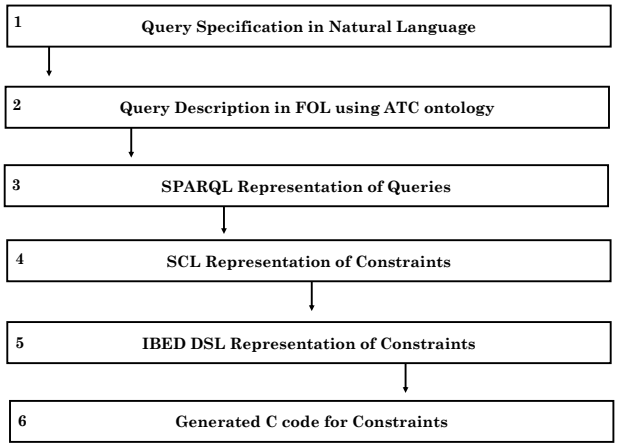

Figure 6: Transformation process from Natural Language to low-level constraint engine code.

Step 2: Query Description in FOL using ATC Ontology. In this step we decompose the natural language description of the queries into concepts and relations using the ATC ontology. Once a query is broken down into basic concepts and relations, we can represent it in FOL. For example, the FOL representation of the statement "if an aircraft has an SSR report and that SSR report has some reported speed s then s is the speed of that aircraft" is:

$$\forall x : Aircarft\ \exists r : SSRReport\ \exists s : ReportedSpeed$$
$$hasSSRReport(x, r) \wedge hasReportedSpeed(r, s) \rightarrow hasSpeed(x, s)$$

The FOL statements are used to construct the SPARQL queries as part of the next step.

Step 3: SPARQL Representation of Queries.

The FOL description use the ATC ontology vocabulary which gives context to the concepts and relations of the queries. The context forms the basis for the SPARQL query representation. We use RDF to represent and store application data. We translate the raw data of the packets to RDF and store in a graph database. After translating the FOL queries to SPARQL, the graph database provides an executable environment to refine the queries and test them against the simulated data.

Step 4: SCL Representation of Constraints.

This step maps the queries from the concepts in the ontological space as expressed by RDF and SPARQL to the network protocol space as expressed by SCL. This moves representation of the queries closer to the network level. For example, the concept `speed` is mapped to the protocol data `SSRModeSType.airspeed`.

The Structure and Context-Sensitive Language (SCL) is an extension of ASN.1 (a network specification language widely used by network engineers). SCL provides a higher abstraction compared to the IBED DSL, but is still attached to the representation and organization of the data given by the protocol specification.

SCL specifies the behavior of the IDS for the incoming packets. Each packet must have one or more constraints that specify the validity of the packet [18]. The SCL constraints specify the how the information in the specified packet depends on information in previous packets. For example, the maximum reasonable speed of an aircraft in an ADS-B packet depends on the type of the aircraft which was a field in an earlier flight strip packet.

```
1 <constraints>
2  <constraint>
3   TYPE: SINGLE-PACKET-ENV
4   VALID-ENV: @RTPS.DATA_P (SrcIP, DstIP, DstPort)
5 </constraint>
6
7  <constraint>
8   TYPE: MULTI-PACKET
9   VALID-SEQ: (1)RTPS.DATA_W, @RTPS.GAP
10  {
11    @RTPS.GAP.SrcIP == (1)RTPS.DATA_W.SrcIP
12    @RTPS.GAP.writerEntityID ==
13        (1)RTPS.DATA_W.writerEntityID
14  }
15  </constraint>
16 </constraints>
```

Listing 1: Syntax convention of SCL SINGLE and MULTI packet constraints [18].

Listing 1 shows an example of two constraints in SCL. The keyword `TYPE` indicates if the constraint is on a single packet (the value `SINGLE-PACKET-ENV`), or if it involves multiple packets (the value `MULTI-PACKET`). The `TYPE` is followed by the sequence of the packets required for the and the type of the target packet. The target packet is prefixed with the symbol '`@`'. For single packet constraints, there is only one packet involved, the target packet.

Listing 1 has a single-packet environment constraint (lines 2-4). Environment constraints refer to entities in the particular environment. Our constraint engine has two modes. When first run on a new system, environmental constraints record the information in the constraints, such as the IP addresses of RTPS participants (DATA_P), or the publishers of particular data topic. As such, environmental constraints list the fields to be memorized as part of the constraint.

For a multi-packet constraint, the target packet is always the last or second last packet in the sequence, as the constraint is written from the point of view of the last packet that triggers the constraint. It may by optionally followed by the packet type that indicates that the instance of constraint is no longer needed (prefixed with the symbol '`~`'. In Listing 1, a GAP submessage in the RTPS protocol must be proceeded by a publisher packet (DATA_W) that introduces the entity id in the gap packet.

Step 5: IBED DSL Representation of the Constraints. Transformation specification of constraint from SCL to an IBED DSL representation is the final step our transformation process. IBED DSL code maps the packets to the constraint tree life-cycle: instantiate, bind, evaluate and destroy. The IBED DSL constraint trees can perform real-time evaluation and provide efficient memory management for the constraint engine [31]. As part of this research the IBED DSL was extended to handle concepts that had not previously been used for constraints at the network infrastructure level. The details of the extension are described in section 6.2

Step 6: Generated C code for Constraints. The IBED DSL is the final step of the manual transformation. The IDS framework

uses the IBED DSL to generate low-level code for constraint engine evaluation. The generated code for the data constraints validate the application data integrity for incoming packets and raises application alerts. The IDS framework uses TXL (a language designed for source code transformation [6]) for auto transformation of DSL code to the C code. As part of this research, the translator was extended for the new concepts identified in step 5. Other than the extensions in the auto-generated code, IBED DSL representation also requires some extensions for correct transformation to c code. For our application domain constraints we manually fixed the generated code for testing.

5 TRANSFORMATION EXAMPLE

In this section, we illustrate the process with one of the threats we identified in section 3

5.1 Violation of Physical Law

This example represents one of the queries that might reveal malicious ADS-B data for an aircraft that doesn't exit. If the reported position of an aircraft is outside the range of a radar antenna, then there is no independent confirmation of the data. A malicious actor might not fully check the data for consistency before broadcasting it, particularly if they are modifying an existing attack. They may get the speed or other characteristics of an aircraft type wrong. This query checks that the speed of an aircraft is consistent with the category of the aircraft.

5.1.1 Step 1 - Query Specification in Natural Language. .

Query Title: The Speed violation of an aircraft at cruising altitude.

Query Definition: The speed of an aircraft is too slow or fast while flying at the cruising altitude, based on the speed range of the aircraft category given by the SSR reports.

Query Description: We identify the ontological concepts such as `Aircraft`, `SSRReport`, `Speed` and `AircraftCategory` along with their relations. Every aircraft has an aircraft category. In the simulation, ADS-B reports are an instance of an SSR data message. They are distinguished from SSR radar reports by the equipment field in the packet. The ADS-B packets for some aircraft may be received (relation `hasSSRReport`). Some SSRReports (there are several type) contain the speed of the aircraft (relation `hasReportedSpeed`).

5.1.2 Step 2 - Description of the Query in FOL using ATC ontology. An aircraft has SSR report and the aircraft is identified with a unique number called `target ID` in these reports. The SSR report has information about the aircraft. Information such as the speed of an aircraft in these reports can be expressed as:

$$\forall x : Aircarft \ \exists r : SSRReport \ \exists s : \ ReportedSpeed$$
$$hasSSRReport(x,\ r)\ \wedge hasReportedSpeed(r,\ s) \rightarrow hasSpeed(x,\ s)$$

The SSR reports for aircraft have other information about the aircraft such as the category of the plane. For example, a Boeing A380 belongs to the aircraft category `C` [19]. Each of these categories has a known minimum and maximum speed.

The ADS-B reports contain the speed of the aircraft, as well as the category of the aircraft. The following query that identifies aircraft whose speed is outside of the range of the category:

$$\forall x : Aircarft \ \exists r : SSRReport \ \exists c : AircraftCategory$$
$$\exists s : ReportedSpeed \ \exists m : MaxSpeed \ \exists l : MinSpeed$$
$$hasSSRReport(x,\ r) \wedge hasReportedSpeed(r,\ s) \wedge$$
$$hasReportedAircraftCategory(r,\ c) \wedge hasMaxSpeed(c, m)$$
$$\wedge hasMinSpeed(c, l) \wedge ((s\ >\ m)\ \vee (s\ <\ l))$$
$$\rightarrow hasViolatingSpeed(x,\ s)$$

Table 1 lists the ATC ontology relations used for the query. The relations `hasMaxCSpeed` and `hasMinCSpeed` refer to the maximum and minimum speed and are not part of the ontology vocabulary. For these queries added as extra relations in the graph database. The Table 1 shows the types of the domain and range for each the properties.

Domain	Predicate	Range
Aircraft	hasSSRReport	SSRReport
SSRReport	hasAirspeed	xsd:integer
SSRReport	hasAircraftCategory	xsd:string
AircraftCategory	hasMaxCategorySpeed	xsd:integer
AircraftCategory	hasMinCategorySpeed	xsd:integer

Table 1: Step 2 - Violation of the Physical Law and Ontology Vocabulary.

5.1.3 Step 3 - The SPARQL Query. The SPAQRL Query in listing 2 is the translation of the FOL query from the previous section. This query is expressed in the same RDF framework as the data mapping that was used to map the application data in network packets to the graph database. This is the first point in time that we can test the query against data from the simulation. This query was successfully run against both clean data from the simulation, and data that contained simulated malicious data.

```
1  SELECT ?assignedTargetID  ?ssrReport
2         ?reprotedSpeed
3  FROM FastInjectedData:
4  WHERE {
5  ?ssrReport ssr:hasTargetID ?assignedTargetID;
6    ssr:hasAircraftCategory ?reportedCategory;
7    ssr:hasAirSpeed ?reprotedSpeed;
8    st:hasMaximumCategorySpeed ?maxCategorySpeed;
9    st:hasMinimumCategorySpeed ?minCategorySpeed.
10 FILTER((?reprotedSpeed < ?minCategorySpeed)
11    || (?reprotedSpeed > ?maxCategorySpeed))
12 }
```

Listing 2: Step 3 - SPARQL Query for the Violation of the Physical Law.

5.1.4 Step 4 - SCL Representation. We transform the SPARQL query to a corresponding SCL constraint. The mapping of RDF elements of the SPAQL query to network fields used in SCL is given in Table 2.

Listing 3 gives a SCL representation of the SPARQL query. This is an extension to the SCL language to allow a logical constraint on the fields of a single single packet. the target packet of the constraint is an SSR Mode S packet. The same packet destroys the instance of the constraint that is created. The second extension to the language is the addition of the domain element that allows the constraint

Ontology Relations	SCL Field Name
ssr:	SSRModeSType
ssr:hasTargetID	SSRModeSTyp.target_id
ssr:hasAirSpeed	SSRModeSTyp.airspeed
ssr:hasAircraftCategory	SSRModeSType.category
st:hasMaximumCategorySpeed	used as a scalar value
st:hasMinimumCategorySpeed	used as a scalar value

Table 2: SCL fields to the ontology relations mapping.

writer to reference elements of the domain. The constraint simply says that the speed must be between minimum and maximum value for the category.

```
1 <constraint>
2  TYPE: SINGLE-PACKET
3  VALID-SEQ: @SSRModeSType, ~SSRModeSType.
4  {(@SSRModeSType.airspeed
5           > Domain.CategoryMinSpeed)||
6   (@SSRModeSType.airspeed
7           < Domain.CategoryMaxSpeed)}
8 </constraint>
```

Listing 3: Step 4 - SCL Representation of the Violation of the Physical Law.

5.1.5 Step 5 - IBED DSL Representation. In the SCL representation of the constraint, it is a single packet constraint and requires comparison of only one value, `airspeed`, for each incoming SSR packet and after comparison it can be destroyed. The IBED DSL is shown in Figure 4.

The constraint starts with the validation tree, that has three values, the speed of a plane, the min and max speed for a category (`categoryMaximumSpeed` and the `categoryMinimumSpeed`).

The DSL requires both an instantiate phase and and an evaluate phase. Nominally these are triggered by different packets and a hash table on values shared between the packets are used to transfer the instance of the constraint tree from one packet to the other. The code for each packet type is generated first for instantiate, bind second, evaluate third and last for destroy. We take advantage of this when generating code to evaluate a predicate on a single packet.

In the instantiate phase on line 6 through line 15 is triggered by an SSRModeSType packet and the values required from the incoming packets are copied to the tree. The notation has been extended with two domain information functions, `DomainLookUpMax`, line 11, and `DomainLookupMin`, that provide external information based on information in the packet. In this case, we use the field `aircarft$category` to find the maximum and minimum speeds of the aircraft. We store the tree instance in the hashtable for use in the evaluate phase.

In evaluate, line 17 through line 24, we recover the tree instance and evaluate it. In destroy we find the tree and destroy it, line 27. As the needed extensions to the constraint engine are in the process of being implemented, a simplified version of the DSL was implemented and after the code was generated, was hand patched to add the needed operators to the evaluation of the tree and in code for the instantiate phase.

```
1  CONSTRAINT AD42
2
3  V( AND( LT(speed, categoryMaximumSpeed),
4         GT(speed, categoryMinimumSpeed) ) )
5
6  INSTANTIATE
7     AppData PDU_AppData.Type is SSRModeSType
8     if not SEARCH Protocol~target$id :Hash=hashIAD42
9         Tree.targetId = Protocol~target$id
10        Tree.category = Protocol~aircraft$category
11        Tree.categoryMaxSpeed = DomainLookUpMax(Tree.
      category)
12        Tree.catgeoryMinSpeed = DomainLookupMin(Tree.
      category)
13        Key = Protocol~target$id
14        HashInstantiate = hashIAD42
15    endif
16
17 EVALUATE
18    AppData PDU_AppData.Type is SSRModeSType
19    HashBind = hashIAD42
20    if SEARCH Protocol~target$id :Hash=hashIAD42
21        Tree.category = Protocol~aircraft$category
22        Tree.speed = Protocol~groundspeed
23        EVAL Protocol~target$id , Protocol~speed
24    endif
25
26 DESTROY
27    if SEARCH Protocol~target$id :Hash=hashIAD42
28        Key=Protocol~target$id
29        HashBind =  hashIAD42
30    endif
31 END
```

Listing 4: Step 5 - IBED DSL Code for Violation of the Physical Law.

6 EVALUATION AND RESULTS

The evaluation shows that constraint engine can be extended to handling not only network constraints but application data constraints as well. It also shows that the ontology and SPARQL can be used to evaluate potential threats in the domain/ It shows that we can implement the SPARQL queries in the constraint engine and enforce them at the network level. In addition to the constraint detecting the violation of physical laws, we applied the process to the other two threats identified in section 3.

6.1 Evaluation of SPARQL

All three threats were expressed as SPARQL queries on our ATC ontology. We generated four data sets. One contains only the clean data from a Euroscope scenario file. We created three scripts that injected malicious data for each of the three threats. Three graph databases were created, each with one set of data. Each query was run against the clean graph database and the malicious data set for that threat. The result of the SPARQL queries is shown in table 3.

In one case, the SPARQL query successfully detected the malicious data, and processed the clean data without incident. In the ghost plane attack scenario, the range of the primary radar was set to the range used by EuroScope. However, this ended up with an edge condition in which an aircraft came into range and broadcast

Threat	Normal Trace	Attack Trace
Physical Law Violation	No alerts	Correct detection
Ghost Plane	1 false alert	Correct detection
Spoofed Location	2 false alerts	Correct detection

Table 3: SPARQL Query Results.

an ADS-B message before the simulated radar detected the aircraft. Revising the query to use a slightly smaller range of radar to ensure that the radar picks up a legitimate aircraft before an ADS-B message is considered malicious.

The third attack scenario is that the position of an existing aircraft is altered by immediately following a legitimate ADS-B message with a malicious ADS-B message with a false position. This query detects this attack by examining the period between ADS-B messages. However, there were two cases in the clean scenario where EuroScope generated legitimate updates that were closer than threshold used in the query.

6.2 Evaluation of IDS

The results of evaluation the application data constraints with the constraint engine are summarized in figure 4. The ghost plane and the violation of the physical law constraints have the same results as the SPARQL queries. The IBED DSL features needed for the spoofed location query were not available, even using the approach of using a placeholder and manually correcting the generated code.

Threat	Normal Trace	Attack Trace
Physical Law Violation	No alerts	Correct detection
Ghost Plane	1 Case	Correct detection
Spoofed Location	Not complete	Not Complete

Table 4: IBED DSL Results.

One of the contributions of this research is to identify the extensions required in the IBED DSL to implement application domain data constraints. The two needed extensions are:

- All three application data constraints rely on external information to be evaluated successfully, such as the 'range of a radar station'. But this information is not available in any packet. We added domain functions such as DomainLookupMax in line 11 of listing4. These allow facts about the real world to be added to constraints.
- The current IBED DSL does not support a constraint on a single packet, as most single packet issues at the network level are handled in the protocol parser. We constraints on single packets that aren't limited to the parsing of the packet.
- The existing IBED DSL implementation has a limited number of logical and relational operators, and no arithmetic operators. These are needed if more general constraints are to be implemented. We generalized the constraint trees to include arithmetic, logical and relational operators.

7 RELATED WORK

There are three areas of related research. The first is redundancy checking and correlation of data. The second is related research in intrusion detection, first order logic and data integrity. The last is work related to the types of threats we investigate.

7.1 Redundancy Checking

Co-relating available information is one of strategies that can be effective in the existing security of any system. This co-relation can be done between different types of data, between data of different systems or between data from different layers of same system. Choo et al. [5] propose that the cyber attacks are 'coordinated' and are 'interconnected'. The main defense of such attacks requires an infrastructure that includes data analytics. Choo et al. suggest that a research challenge is the intelligent analysis of data that is collected from different layers of network security.

Every detection system has the potential to raise false alarm. Ducharme [10] notes that most of the time the consequences of false alarms are resources and time. He notes that to avoid the consequences, it is important to understand the false alarms and be able to co-relate them. Eschelbeck et al. [12] note the importance of the assessment and correlation of data between different systems. They identify the need for correlation of information and used a correlation engine with *Snort IDS* to reduce and validate alerts.

Parnas et al. [28] suggest a "triple redundancy" approach for safety critical systems. The main system of any critical system must perform reliably. Any backup systems must be independent. Parnas et al. suggest that double or triple failure in a disjoint infrastructure is less likely. We do not claim data integrity checking in the IDS is a replacement for data integrity checking in command and control systems such as ATC. Using an IDS to validate application data adds redundancy and more confidence in the overall security of the system.

7.2 Related IDS

Many organizations use security information and event management (SIEM) systems to get an overall view of the information security activity and enforce data integrity [24]. In general, SIEM systems are designed to process security events which are generated by network security solutions [3]. SIEM systems gather a considerable amount of data for analysis from different sources in various formats. SIEM has many advantages, but there are limitations. To make any sense of this data it must be converted into a consistent format [35]. Security reports and dashboards provided by SIEM systems are useful for security staff and management, because they show several security metrics and the general state of information security within organizations [25].

But these reports, logs and alerts contain a significant amount of data. SIEM rules are used to correlate this information. Majeed et al. [20] suggest that many SIEM systems are incapable of giving the status of these rules in real time. Our approach may be adapted to allow critical rules to be validated in real time.

Andrea et al. [4] investigate using an IDS that represent the states of the system using a rules language for Industrial Control Systems (ICS). Like our approach they work with a domain specific network. We focus on data in command and control systems such as ATC. Elfaki et al. [11] also based their intelligent rules on first order logic to better detect inconsistencies. We use FOL for representation of our threats, which are then transformed to IBED DSL constraints.

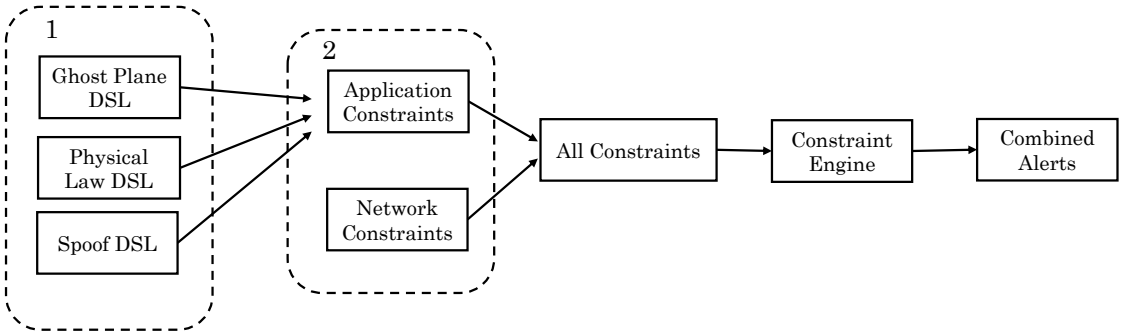

Figure 7: Contributions

7.3 Threats and Attacks

Costin et al. [7] in *Ghost in the Air* identify security issues in ADS-B and show that attacks on the ADS-B are not only possible, but easy. In ATC systems, data provided by ADS-B is trusted and lacks security as a key feature [7] [33]. Costin et al. emphasize adding the most basics authentications. Our physical law violation threat model is a demonstration of the lack of such basic authentication. Abnormal behaviour can be indicative of something that needs to be further investigated.

Ray et al. [32] propose using an ontology for threat models. They suggest starting by familiarizing oneself with the domain by interviewing domain experts before building a threat model. Balduzzi et al. [2] provided a categorization of attacks on AIS. AIS and ADS-B share many of the same vulnerabilities and threats.

8 CONCLUSIONS AND FUTURE WORK

In this research we extend an existing constraint based IDS to identify data integrity at the application level. We demonstrate the extensions in the domain of air traffic control. Figure 7 is a representation of the contributions. We specify a set of transformations from natural language to SPARQL queries to IBED DSL constraints, that can be used to generate a custom IDS which are shown on the left of figure 7. We test our proposed application data constraints with our current IDS framework. The evaluation demonstrates some elements of the DSL and generator that must be extended to fully support application data constraints as shown in region 2 of Figure 7. We show that with the extensions, application data constraints can use the same life-cycle as our network constraints. We propose and present a set of application domain data constraints for the ATC domain, using the same auto-generated framework.

The future work for our research will focus on extensions to the SCL and the IBED DSL. More application data constraints should be evaluated and the work on mutli-packets constraint will be completed. The IDS framework is currently generated semi-automatically. The extensions identified in this research are in the process of being integrated into the constraint engine generator.

The IDS is now capable of working on ensuring integrity in two different aspects of a system, network and data. One interesting dimension would be to explore defining constraints on another aspect or working layer, to see if that adds further security.

In conclusion, some application domain data can be evaluated at the network level. Industrial control systems and command and control applications are often complex, and while security is a critical component, it is one of many components for critical systems. Our approach adds a redundant check of the integrity of application data in the intrusion detection system, where the sole focus is on the system security.

We also provide an example of using the ATOM process to use an Ontology to evaluate application integrity in the air traffic control domain using queries. We then transform them to a low level constraint representation that can be validated in real time.

9 ACKNOWLEDGMENTS

We would like to acknowledge funding from the Department of National Defense.

REFERENCES

[1] ISO/IEC JTC 1. 1994. *ISO/IEC 7498-1:1994 Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model.* International Standards Organization, Geneva, Switzerland.

[2] Marco Balduzzi, Alessandro Pasta, and Kyle Wilhoit. 2014. A Security Evaluation of AIS Automated Identification System. In *Proceedings of the 30th Annual Computer Security Applications Conference* (New Orleans, Louisiana, USA) *(ACSAC '14)*. ACM, New York, NY, USA, 436–445. https://doi.org/10.1145/2664243.2664257

[3] S. Bhatt, P. K. Manadhata, and L. Zomlot. 2014. The operational role of security information and event management systems. *IEEE Security & Privacy* 12 (2014), 35 – 41.

[4] Andrea Carcano, Igor Nai Fovino, Marcelo Masera, and Alberto Trombetta. 2010. State-Based Network Intrusion Detection Systems for SCADA Protocols: A Proof of Concept. In *Critical Information Infrastructures Security*, Erich Rome and Robin Bloomfield (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 138–150.

[5] Kim-Kwang Raymond Choo and Ali Dehghantanha. 2018. Introduction to the Minitrack on Cyber Threat Intelligence and Analytics: A Conceptual Three-Pronged Approach and Future Research Agenda. In *Proceedings of the 51st Hawaii International Conference on System Sciences*. 5521 – 5523. https://doi.org/10.24251/HICSS.2018.688

[6] James R. Cordy. 2006. The TXL source transformation language. *Science of Computer Programming* 61, 3 (2006), 190 – 210. https://doi.org/10.1016/j.scico.2006.04.002 Special Issue on The Fourth Workshop on Language Descriptions, Tools, and Applications (LDTA âĂŹ04).

[7] Andrei Costin and Aurélien Francillon. 2012. Ghost in the Air(Traffic): On insecurity of ADS-B protocol and practical attacks on ADS-B devices. In *BLACKHAT 2012, July 21-26, 2012, Las Vegas, NV, USA*. Las Vegas, UNITED STATES. http://www.eurecom.fr/publication/3788

[8] Gergely Csernak. [n.d.]. EuroScope User Guide, for Version 3.0a. https://www.euroscope.hu/documents/EuroScopeUsersGuide30.pdf. Accessed: 2019-10-29.

[9] R. De Cerchio and C. Riley. 2012. Aircraft systems cyber security. In *2012 Integrated Communications, Navigation and Surveillance Conference*. 1–12. https://doi.org/10.1109/ICNSurv.2012.6218454

[10] É. Ducharme. 2017. *Détection d'intrusion à l'aide d'un système expert basé sur l'ontologie*. Master's thesis. École Polytechnique de Montréal.

[11] Abdelrahman Osman Elfaki, Somnuk Phon-Amnuaisuk, and Chin Kuan Ho. 2009. *Investigating Inconsistency Detection as a Validation Operation in Software Product Line*. Springer Berlin Heidelberg, Berlin, Heidelberg, 159–168. https://doi.org/10.1007/978-3-642-05441-9_14

[12] Gerhard Eschelbeck and Michael Krieger. 2003. Eliminating noise from intrusion detection systems. *Information Security Technical Report* 8 (04 2003), 26 – 33. https://doi.org/10.1016/S1363-4127(03)00004-9

[13] Object Management Group. [n.d.]. The Real-time Publish-Subscribe Protocol (RTPS) DDS Interoperability Wire Protocol Specification. https://www.omg.org/spec/DDSI-RTPS/2.3/Beta1/PDF. Accessed: 2019-05-22.

[14] RDF Working Group. 2014. Resource Description Framework (RDF). https://www.w3.org/RDF/. (2014).

[15] SPARQL Working Group. 2008. SPARQL Query Language for RDF. https://www.w3.org/TR/rdf-sparql-query/. (2008). Accessed: 2020-06-12.

[16] MD Siam Hasan, Thomas Dean, Fahim T. Imam, Francisco Garcia, Sylvain P. Leblanc, and Mohammad Zulkernine. 2017. A Constraint-based Intrusion Detection System. In *Proceedings of the Fifth European Conference on the Engineering of Computer-Based Systems* (Larnaca, Cyprus) *(ECBS '17)*. ACM, New York, NY, USA, Article 12, 10 pages. https://doi.org/10.1145/3123779.3123812

[17] M. S. Hasan, A. ElShakankiry, T. Dean, and M. Zulkernine. 2016. Intrusion detection in a private network by satisfying constraints. In *2016 14th Annual Conference on Privacy, Security and Trust (PST)* (Aukland, New Zealand). 623–628. https://doi.org/10.1109/PST.2016.7906997

[18] Fahim Imam. 2020. *Specifying Constraints in SCL5 for Intrusion Detection*. Technical Report. http://pyxis.ece.queensu.ca/papers/compasstr20-1.pdf/ [Online; Accessed: 2020.02.13].

[19] Legal Information Institute. [n.d.]. Aircraft approach category. https://www.law.cornell.edu/cfr/text/14/97.3 [Online; accessed 14-June-2020].

[20] Abdul Majeed, Raihan ur Rasool, Farooq Ahmad, Masoom Alam, and Nadeem Javaid. 2019. Near-miss situation based visual analysis of SIEM rules for real time network security monitoring. *Journal of Ambient Intelligence and Humanized Computing* 10, 4 (01 Apr 2019), 1509–1526. https://doi.org/10.1007/s12652-018-0936-7

[21] Simon Malenfant-Corriveau. 2017. *PROPOSAL FOR A METHOD OF DEVELOPING ONTOLOGY FOR A SYSTEM EXPERT IN SECURITY*. Master's thesis. École Polytechnique de Montréal.

[22] Mohsen Riahi Manesh and Maima Kaabouch. 2017. Analysis of Vulnerabilities, Attacks, Countermeasures and Overall Risk of the Automatic Dependent Surveillance-Broadcast (ADS-B) System. https://doi.org/10.1016/j.ijcip.2017.10.002. *Int. J. Crit. Infrastruct. Prot.* 19, C (Dec. 2017), 16âĂŞ31. https://doi.org/10.1016/j.ijcip.2017.10.002

[23] Sylvain Marquis, Thomas R. Dean, and Scott Knight. 2005. SCL: A Language for Security Testing of Network Applications. In *Proceedings of the 2005 Conference of the Centre for Advanced Studies on Collaborative Research* (Toranto, Ontario, Canada) *(CASCON âĂŹ05)*. IBM Press, 155âĂŞ164.

[24] Pal Michelberger and Sandor Dombora. 2016. A Possible Tool for Development of Information Security- Siem System. *Ekonomika, Journal for Economic Theory and Practice and Social Issues* 1350-2019-2051 (2016). https://doi.org/10.22004/ag.econ.288703

[25] Raydel Montesino, Stefan Fenz, and Walter Baluja GarcÃŋa. 2012. SIEM-based framework for security controls automation. *Information Management & Computer Security* 20 (10 2012). https://doi.org/10.1108/09685221211267639

[26] L.-P Morel. 2017. *Using Ontologies to Detect Anomalies in the Sky*. Master's thesis.

[27] Ontotext. 2019. What is SPARQL. https://www.ontotext.com/knowledgehub/fundamentals/what-is-sparql/. (2019). Accessed: 2020-02-13.

[28] David Parnas, Jan Madey, and G. Asmis. 1991. Assessment of safety-critical software in nuclear power plants. *Nuclear Safety* 32 (04 1991).

[29] CFR Part. 91. Automatic Dependent Surveillance–Broadcast (ADS–B) Out Performance Requirements to Support Air Traffic Control (ATC) Service. *Final Rule* 91 (91).

[30] Y. Raimond and G. Schreiber. 2014. RDF 1.1 primer. http://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/. (2014).

[31] Mohamed Sami Rakha, Fahim T. Imam, and Thomas R. Dean. 2019. Generating a Real-Time Constraint Engine for Network Protocols. In *12th IFIP International Conference on Information Security Theory and Practice (WISTP) (Information Security Theory and Practice, Vol. LNCS-11469)*, Olivier Blazy and Chan Yeob Yeun (Eds.). Springer International Publishing, Brussels, Belgium, 44–60. https://doi.org/10.1007/978-3-030-20074-9_5 Part 2: Real World.

[32] Cyril Ray, Romain Gallen, Clement Iphar, Aldo Napoli, and Alain Boujou. 2015. DeAIS project: Detection of AIS spoofing and resulting risks. IEEE, OCEANS 2015 - Genova, Genoa, Italy. https://doi.org/10.1109/OCEANS-Genova.2015.7271729

[33] SC-186. 2009. *DO-282B, Minimum Operational Performance Standards for Universal Access Transceiver (UAT) Automatic Dependent Surveillance-Broadcast (ADS-B)*. Technical Report. 1150 18th NW, Suite 910 Washington, DC 20036 USA.

[34] Lucio Vismari and JoÃčo Junior. 2011. A safety assessment methodology applied to CNS/ATM-based air traffic control system. *Reliability Engineering & System Safety - RELIAB ENG SYST SAFETY* 96 (07 2011), 727–738. https://doi.org/10.1016/j.ress.2011.02.007

[35] Peter Zegzhda, Dmitry Zegzhda, Maxim Kalinin, Alexander Pechenkin, Alexander Minin, and Daria Lavrova. 2016. Safe Integration of SIEM Systems with Internet of Things: Data Aggregation, Integrity Control, and Bioinspired Safe Routing. In *Proceedings of the 9th International Conference on Security of Information and Networks* (Newark, NJ, USA) *(SIN '16)*. ACM, New York, NY, USA, 81–87. https://doi.org/10.1145/2947626.2947639