Don't Panic! Analysing the Impact of Attacks on the Safety of Flight Management Systems

Luca Maria Castiglione*, Philipp Stassen†, Cora Perner†, Daniel Patrick Pereira†,
Gustavo de Carvalho Bertoli†, and Emil C. Lupu*
* *Imperial College London*
† *Airbus*

***Abstract*—Increased connectivity in modern aircraft also significantly increases the attack surface available to adversaries and the number of possible attack paths. It is therefore of essence to characterise the attacks that can impact safety. We present Cassandra , a novel methodology combining System Theoretic Process Analysis Security (STPA-Sec) with formal verification to automatically identify safety critical threat scenarios. Unlike previous methodologies for safety and security analysis, Cassandra leverages the integration with the aircraft architecture, together with the set of threats and the privileges required to execute them, to also identify safety critical attack paths. We employ Bayesian inference to compute the probability of success for the safety critical attacks found. We describe how Cassandra can be used in the system early design phase to reason about attack paths leading to safety critical threat scenarios and discuss how it can be further used to evaluate mitigation and assurance cases by reducing threat vectors and increasing safety. In particular, we apply Cassandra to analyse the safe operation of a Flight Management System (FMS) when the adversary tries to access safety critical information by compromising the device used as the Electronic Flight Bag (EFB). We evaluate the probability of successful attacks in three different scenarios: EFB available on pilot owned device, EFB available on airline controlled device with limited connectivity, and EFB available on aircraft only. While the outcome of Cassandra may be intuitive in this case, the example allows us to show how Cassandra improves automation and integration of safety and security analysis for modern avionic architectures, where complexity hinders intuition and manual analysis is laborious and error prone.**

***Index Terms*—safety, security, attack graphs, verification, STPA-Sec.**

I. Introduction

With the increase of connectivity in modern aircraft, the attack surface available to adversaries also grows. Complex attacks combine the exploitation of vulnerabilities with the malicious use of legitimate functionality to induce cascading effects and alter system behaviour. In safety critical systems without appropriate safeguards, such attacks can lead to the violation of safety requirements.

The support of the EPSRC Centre for Doctoral Training in High Performance Embedded and Distributed Systems (HiPEDS, Grant Reference $EP/L016796/1$) is gratefully acknowledged. This work has also received funding, in part, from the European Union's Horizon 2020 / Europe research and innovation programme under grant agreements No 830927 (Concordia) and No 101077071 (Albatros). Views and opinions expressed here are those of the authors alone and do not necessarily reflect those of the funding organisations.

The Flight Management System (FMS) is of utmost importance among the various systems integrated into modern aircraft, primarily due to its pivotal role in significantly reduce the crew workload associated with flight planning, navigation, and guidance. With the rapid advances in technology, FMS has become an indispensable component of both commercial and general aviation aircraft, significantly improving operational efficiency and flight safety [1]. Its functionality relies on seamless integration with numerous other systems and components within the aircraft, thereby increasing the complexity involved in conducting a comprehensive analysis of all potential paths that could lead to safety or security violations. Performing a thorough analysis of aircraft systems such as the FMS during the development phase of the aircraft facilitates the identification of potential architectural improvements and the implementation of necessary mitigations to maintain required levels of safety and security and thus development cost. To enable such early-stage analyses, top-down techniques such as System Theoretic Process Analysis (STPA) need to be employed [2]. In fact, STPA offers the advantage of performing analysis at a functional level, thus eliminating the need of specifications for individual components or item-level definitions of the aircraft. It further enables identify the causal factors leading to hazards by examining the interactions between system components [3], [4]. By considering functional interaction, STPA enables the identification of potential risks and the formulation of recommendations at an earlier stage in the design process. In contrast, traditional methodologies (e.g., fault/attack-trees) for safety and security analysis require well-defined architectures, with the item-level specifications, making it more difficult to identify desirable architectural changes and recommendations. Furthermore, these methodologies are generally performed manually and require a significant amount of effort and highly specific knowledge across several teams of experts in systems, safety and security. Consequently, the results obtained by these analyses are highly subjective and lack reproducibility. Moreover, the methods employed lack a clear link to one another, thus limiting traceability between different analyses.

In this paper, we present Cassandra, a novel methodology that combines System Theoretic Process Analysis for security (STPA-Sec) with formal verification to automatically identify the safety critical threat scenarios. Unlike earlier methodolo-

gies, Cassandra leverages a correspondence between high-level threats and aircraft architecture to identify safety critical attack paths, i.e. sequences of vulnerabilities that the adversary can exploit to cause harm. We analyse the attack paths found to quantitatively compute the probability of success for safety critical attacks and measure the impact of security controls. In contrast to other approaches that combine threat modelling with safety analysis, Cassandra offers an integrated set of tools that enable the automated derivation of safety critical sequences of threats and their respective attack paths. This provides an important step towards making security analyses less subjective, more reproducible and thus more suitable for applications in safety-critical aircraft contexts.

In particular, this work brings the following contributions:

- We perform a semi-automated integrated safety and security analysis of the flight management system and identify *attack paths* leading to the execution of safety critical attacks.
- We model threats against a representative, realistic flight management system.
- We leverage exact Bayesian inference to analyse found paths and evaluate different types of security controls.

The paper is structured as follows. We outline related works in Section II and introduce our aircraft model in Section III. Section IV describes our proposed methodology and the preliminary application of STPA. In Section V we show the application of Cassandra to automatically identify and enumerate the threat scenarios, while in Section VI we leverage the integration with the aircraft architecture to uncover safety critical attack paths. In Section VII we discuss the obtained result. Finally, Section VIII presents the conclusions.

II. Related Works

STPA-Sec [5], [6] extends STPA to consider security threats amongst the causes of unsafe application of control actions. STPA-SafeSec [7] extends STPA-Sec and considers a number of generic threats affecting the architecture of the system. Khan et al. analyse the impact of security threats in conjunction with STAMP in [8]. These approaches require a significant amount of expert knowledge and are error prone as they are traditionally performed manually. In addition, they only provide a set of high level scenarios and recommendations as output. In contrast, our work introduces automation - which allows a systematic and complete (with respect to the threats and the model specified) exploration of all the possible cases. Moreover, through the analysis of attack graphs we can analyse the impact of individual vulnerabilities on the high-level safety properties. Works presented in [9] and [10] propose integrated safety and security analysis using STPA in conjunction with simulations and attack graphs, respectively. Studies [11], [12], and [13] combine STPA with model checking. However, they exclusively focus on safety and testing, and do not consider adversarial threats. Verdict [14] is an AADL annex that includes Model Based Architecture Analysis and Synthesis (MBAAS) and Cyber Resiliency Verification (CRV). MBAAS uses propagation rules manually specified for each component to infer the propagation of the effects of threats, while CRV verifies the reachability of threat scenarios for a limited number of hard-coded threats. In contrast, we rely on the application of systematic threat modelling to systematically synthesise a model of the adversary. SAHARA [15] combines STRIDE threat modelling with the HARA safety methodology to integrate the analysis of safety and security, whilst, more generally [16] and [17] present an overview of other formalisms used to study safety and security. Finally, The use of attack trees for goal oriented analysis [18] and attack graphs for representation of possible attacks has also been reported in the literature [19]. The Boolean Driven Markov Processes (BDMP) approach proposed in [20], allows to model the impact of vulnerabilities on safety in the control system of a pipeline, while [21] presents an interesting argument for the need for standards for security and safety co-analysis. Barrère et al. [22] use an approach based AND/OR graphs to identify mission critical components in CPS.

III. Flight Management System

The Flight Management System (FMS) is part of the auto flight system in conjunction with flight guidance and flight envelope management. The FMS is responsible for managing the navigation, datalink, lateral and vertical functions of an aircraft. It carries out safety critical functions such as following route and fuel management. It exchanges information with other part of aircraft avionics such as Navigation Systems (NS), Flight Control Systems (FCS), as well as other sensors and actuators to carry out its functions. For simplicity, in this paper, we primarily focus on the safety of the operations of the *follow route* function, but our approach can be easily applied to the safety of other functions. In Figure 1, we show the network topology considered in this paper. It is divided in three distinct domains, each of them with its own characteristics and security levels. The aircraft control domain (ACD), depicted in blue, is used by avionics components such as FMS, NS, sensors, etc. to exchange information. This network is safety critical and contains redundant components as well as redundant buses. For simplicity, this redundancy is not shown explicitly in Figure 1 [23]. The airline service domain (ASD) is depicted on a green background, it contains the management, non safety critical, information systems used to communicate with airline services through wireless networks. The ASD also contains the Data Loader System (DLS), used by the pilots to upload flight plans from the Electronic Flight Bag (EFB). We assume that the latter is a portable device where the flight plans are stored. Finally, components for passenger *infotainment* are located in the passenger information and entertainment services domain (PIES).

In this paper, we analyse how the risk of safety critical attacks changes, given the following three different levels of security controls on the Electronic Flight Bags (EFB) enforced by the operator: **(Scenario** 1**)** EFB is provided by the airline and has access to the internet (e.g., available under bring-your-own-device policy). **(Scenario** 2**)** EFB is provided by the airline and cannot access the internet, but is

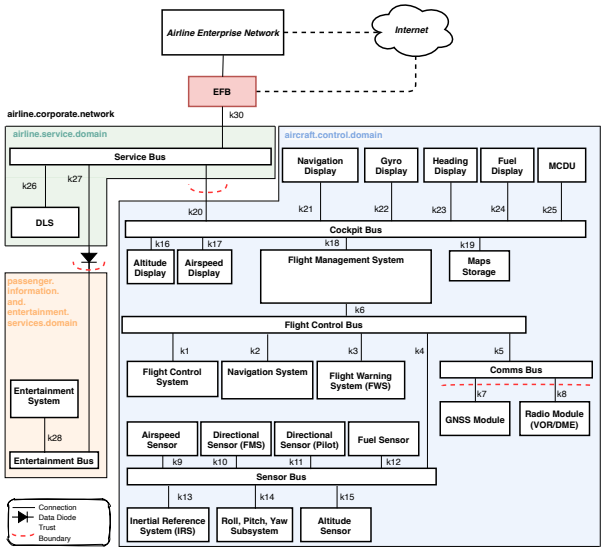

Fig. 1: Aircraft Computer Network

enabled with network access towards airline owned enterprise network. **(Scenario 3)** EFB does not have access to the internet. We assume that the EFB device is vulnerable to `CVE-2022-22620` (Remote Code Execution (RCE)) and `CVE-2022-22675` (Local Privilege Escalation). Similarly, in our scenario, the map server hosts a software component vulnerable to `CVE-2017-0144` (RCE). We use the CVSS score of each vulnerability to measure their likelihood of being exploited, in line with much of the related work in this area [24]. Finally, we assume the presence of the following vulnerabilities `CVE-2023-x` (RCE), `CVE-2023-y` (RCE), and `CVE-2023-z`(RCE) which affect the EFB, the DLS, and the aircraft map storage, respectively. Since `CVE-2023-x`, `CVE-2023-y`, and `CVE-2023-x` affect devices that are not widely available to the public which, among other things, run proprietary software, we assume that each of them, although severe, is *highly unlikely* to be exploited.

IV. Proposed Approach and Preliminaries

Cassandra is organised in *three* stages, shown in Figure 2. The first stage uses the safety model of the system to produce

Fig. 2: Overview of Cassandra

the *Threat Scenarios Tree*, an attack tree (AT) that explains how high-level threats can lead to catastrophic consequences (❶). In this stage, we leverage STAMP to model the safety of the system and STPA and STPA-Sec to discover the dynamics leading to accidents [6], [25]. STPA and STPA-Sec are well accepted methods for safety analysis and are applied across many different sectors [26]. Furthermore, we use STRIDE [27] as the threat modelling approach within STPA-Sec. The derivation of the interleaving sets of malicious actions and system behaviour is achieved through model checking. In doing so, we are able to also consider critical aspects relative to the time and context in which threats occur (e.g., flight phase, autopilot being enabled). In the second stage, we leverage the integration with the system architecture to identify safety critical attack paths, sequences of vulnerability exploitations that the adversary can carry out to execute safety critical threats. This operation is critical in identifying the privileges that the attacker needs to obtain to execute safety critical threat scenarios and cause harm (❷). Finally, we employ exact Bayesian inference on the attack graph to quantitatively compute the probability of success for the attack paths identified. We evaluate the effectiveness of security measures in reducing the probability of success of safety critical attacks (❸).

The first stage of Cassandra is grounded in STPA and uses, as input, part of the results obtained from the execution of the first three steps of STPA. STPA-Sec stems from STPA and uses the same structure as STPA, which develops in four successive steps [2]: *definition of scope of the analysis*, *derivation of the control structure*, *identification of Unsafe Control Actions*, and *identification of threat scenarios*. Like STPA, STPA-Sec is performed manually and the application is driven by expert knowledge. In the next paragraphs we show the application of the first three steps of off-the-shelf (OTS) STPA/STPA-Sec to our representative flight management system.

A. Definition of the scope of the analysis

The first step of STPA/STPA-Sec consists in defining the scope of the analysis. At this stage, we identify *Losses*, *Accidents* and *Hazards*, as defined in [25], that are relevant to the FMS. In particular, we consider the following subset of losses from [28]the Acceptable Means of Compliance (AMC) for Large Aeroplanes[1] [28]; Multiple Fatalities (L_1), Hull Loss (L_2), and Physical discomfort or a significant increase in workload of Flight Crew (L_3). According to [28], L_1 and L_2 are classified as *Catastrophic* (allowable probability $< 10^{-9}$) while L_3 is classified as *Major* (allowable probability $< 10^{-5}$). For this paper, we consider the following reduced subset of events leading to L_1, L_2 and L_3: (A_1) Collision (from Loss of Separation), (A_2), Controlled Flight into Terrain (CFiT), (A_3) Unreliable Avionics Behaviour. We have identified in Table I

TABLE I: Table of Hazards for Flight Management System

#	Hazard	Accidents
H_1	Aircraft violates minimum separation standards in flight	A_1, A_2
H_2	Aircraft does not maintain safe distance from terrain and other obstacles	A_2
H_3	Unable to follow route	A_3

the subset of hazards of concern for the *follow route* functionality of the FMS. If the FMS receives an unreliable position from the Navigation System (NS), it will compute commands that can lead to a loss of separation, and, eventually, a collision, if the aircraft is on autopilot (H_1). Conversely, commands issued by FMS based on an unreliable position or map data can lead to CFiT (H_2) when the aircraft is descending on autopilot. Finally, unreliable feedback from avionics or unreliable data on the databases, can lead to errors in the computation of the flight path and an increase of workload of Flight Crew that are forced to compute corrections by alternate means (H_3).

[1]AMC 25.1309 – System Design and Analysis

B. Derivation of control structure

The second step of STPA/STPA-Sec is the derivation of the safe control structure (SCS). This is part of the STAMP model and captures the dynamics of the interactions between system components [25]. The SCS is a tuple (C, D, K) where C denotes the set of components (controllers, sensors, actuators and physical processes), D the set of *control actions* and *feedback*, and K a set of *functional channels* [9]. Within each controller, the variables of the *process model* retain the representation of the physical process and are used by the *control algorithm*. The SCS for our use case is shown in Figure 3. For simplicity, only a subset of components, control actions and feedback are analysed using Cassandra in the next sections. These have been highlighted in Figure 3 and are summarised in Table II.

TABLE II: Control Actions and Feedback in Figure 3.

CA/FB	Description (Source Component)	CA/FB	Description (Source Component)
CA 1.1	Change Altitude (Climb/Descend) (Pilot)	CA 1.2	Change Yaw (Pilot)
CA 1.3	Change Pitch (Pilot)	CA 1.4	Change Roll (Pilot)
CA 1.5	Change Thrust (Decrease) (Pilot)	CA 2.1	Change Altitude (Climb/Descend) (FMS)
CA 2.2	Change Yaw (FMS)	CA 2.3	Change Pitch (FMS)
CA 2.4	Change Roll (FMS)	CA 2.5	Change Thrust (FMS)
CA 3.1	Change Altitude (Climb/Descend) (FCS)	CA 3.2	Change Yaw (FCS)
CA 3.3	Change Pitch (FCS)	CA 3.4	Change Roll (FCS)
CA 3.5	Change Thrust (FCS)	CA 6.1	Set Route (Pilot)
CA 6.2	Change Route (Pilot)	CA 6.3	Start Follow (Pilot)
CA 6.4	Stop Follow (Pilot)	CA 8	Toggle Alarm (Pilot)
CA 7.1	Set Route (MCDU)	CA 7.2	Change Route (MCDU)
CA 7.3	Start Follow (MCDU)	CA 7.4	Stop Follow (MCDU)
CA 5	Upload Data (Ground Crew)	CA 4	Upload Flight Plan (Pilot)
CA 15	Radio Signal (Radio Source)	CA 16	GNSS Position (GNSS Module)
CA 17	Radio Position (Radio Module)	CA 14	GNSS Signal (GNSS Source)
CA 9	Position (Navigation System)		
F1	Flight Data (MCDU)	F2	Alarm (FWS)
F3	Airspeed (Airspeed Display)	F4	Altitude (Altitude Display)
F5	Fuel (Fuel Display)	F6	Heading (Heading Display)
F7	Roll, Pitch, Yaw (Gyro Display)	F8	VNav (Visual Navigation)
F9	Position (Navigation Display)	F10	Flight Data (FMS)
F11	Alarm (FMS)	F12	Uploaded Data (DLS)
F13	Stored Maps (Map Storage)	F14	Heading (Dir Sensor FMS)
F15	Altitude (Altitude Sensor)	F16	Airspeed (Airspeed Sensor)
F17	Position (Navigation System)	F18	Roll, Pitch, Yaw (RPY Subsystem)
F19	Airspeed (Airspeed Sensor)	F20	Altitude (Altitude Sensor)
F21	Fuel Level (Fuel Sensor)	F22	Heading (Dir Sensor Pilot)
F23	Roll, Pitch, Yaw (RPY Subsystem)	F24	Position (Navigation System)
F25	IMU-1 Heading (Raw)	F26	Airspeed (Raw)
F27	DME Input (Raw)	F28	Fuel Level (Raw)
F29	IMU-2 Heading (Raw)	F30	Roll, Pitch, Yaw (Raw)
F31	Visual Navigation Input (Raw)	F32	Acceleration Forces (Raw)
F33	Position (IRS)	F34	CA 14
F35	CA 15	F36	CA 16
F37	CA 17	F38	Fuel Level (Fuel Sensor)

In this paper, we focus on the safe operation of the FMS which issues control actions Change Altitude ($CA_{2.1}$), Change Yaw ($CA_{2.2}$), Change Pitch ($CA_{2.3}$), Change Roll ($CA_{2.4}$), and Change Thrust ($CA_{2.5}$). We use CA_2 to refer to the entire output of the FMS, hence control actions $CA_{2.1}$ to $CA_{2.5}$. These actions are computed, on the basis of the current state of the FMS, from the input received from the Navigation System (NS) and other sensors such as airspeed, fuel, heading, altitude, etc. The NS computes CA_9 on the basis of the data received from the Inertial Reference System (INSS) (F_{32}), the Global Navigation Satellite System (GNSS) (F_{36}), and the radio (F_{37}) using the algorithm outlined in Section V.

C. Identification of Unsafe Control Actions

The third step of STPA/STPA-Sec is the identification of Unsafe Control Actions (UCA), i.e., control actions that cause hazards if applied in a specific *context* [29]. For example, the *climb* command issued by the FMS is unsafe when its application places the aircraft in a unsafe situation, e.g., loss of separation, steep climb, etc. STPA distinguish between $four$ types of unsafe applications of control actions: *a control action is* ***provided when not required***, *a control action is* ***not provided when required***, *a control action is* ***provided too early (or too late)***, and *a control action is* ***provided for too long or too short*** [25]. The latter only applies to control actions defined in the continuous domain. For simplicity, we report here only the following unsafe applications related to the control action *change altitude*, output of the FMS ($CA_{2.1}$):

$UCA_1 : CA_{2.1}$ (Change altitude) is applied (**climb**) when not required while **autopilot is active** $\leftarrow H_1, H_3$.
$UCA_2 : CA_{2.1}$ (Change altitude) is applied (**descend**) when not required while **autopilot is active** $\leftarrow H_2, H_3$
$UCA_3 : CA_{2.1}$ (Change altitude) is applied (**climb** or **descend**) when not required while **autopilot is not active** H_3
$UCA_4 : CA_{2.1}$ (Change altitude) is not applied (**climb**) when required while **autopilot is active** $\leftarrow H_2, H_3$.
$UCA_5 : CA_{2.1}$ (Change altitude) is not applied (**descend**) when required while **autopilot is active** $\leftarrow H_1, H_3$.
$UCA_6 : CA_{2.1}$ (Change altitude) is not applied (**climb** or **descend**) when required while **autopilot is not active** $\leftarrow H_3$.
$UCA_7 : CA_{2.1}$ (Change altitude) is applied (**climb** or **descend**) too early or too late while **autopilot is active** $\leftarrow H_1, H_2, H_3$.
$UCA_8 : CA_{2.1}$ (Change altitude) is applied (**climb** or **descend**) for too long or too short while **autopilot is active** $\leftarrow H_1, H_2, H_3$.

D. Identification of Threat Scenarios (STPA-Sec)

In the last step of STPA/STPA-Sec, we search for causes behind the application of unsafe control actions in the whole control structure. In particular, STPA-Sec looks into threats as a possible cause, whereas traditional STPA focuses on faults. The identification of causal scenarios is traditionally performed manually, mainly relying on expert knowledge. This process is complex, time consuming and error prone as it requires the analyst to intersect the specific behaviour of individual system components with a high-level system view on how control actions and feedback propagate. In addition to this, STPA-Sec does not provide clear tools to define the scope of the security analysis (e.g., trust boundaries) nor provides guidelines on the nature of threats to consider in this fourth step. In the first step of Cassandra we employ STRIDE [30] to identify threats that are applicable to individual elements of the control structure. Thus, we implement an automatic strategy based on model-checking which allows to systematically enumerate *threat scenarios*.

V. Enumeration of Threat Scenarios

Although highly effective in highlighting control dependencies within the cyber physical system (CPS), STPA alone cannot perform the automatic analysis of the cascading effects leading to the application of unsafe control actions. In fact, STAMP does not formalise the behaviour of system components and, without it, we cannot infer how alterations in components' input reflect on their output. In this paper, we encode input/output relationships of system components in the *safe behavioural model* (SBM) of the CPS. A well-defined correspondence between the STAMP model of the CPS and the SBM enables us to express the safety requirements (previously identified during the application of the third step of STPA) in

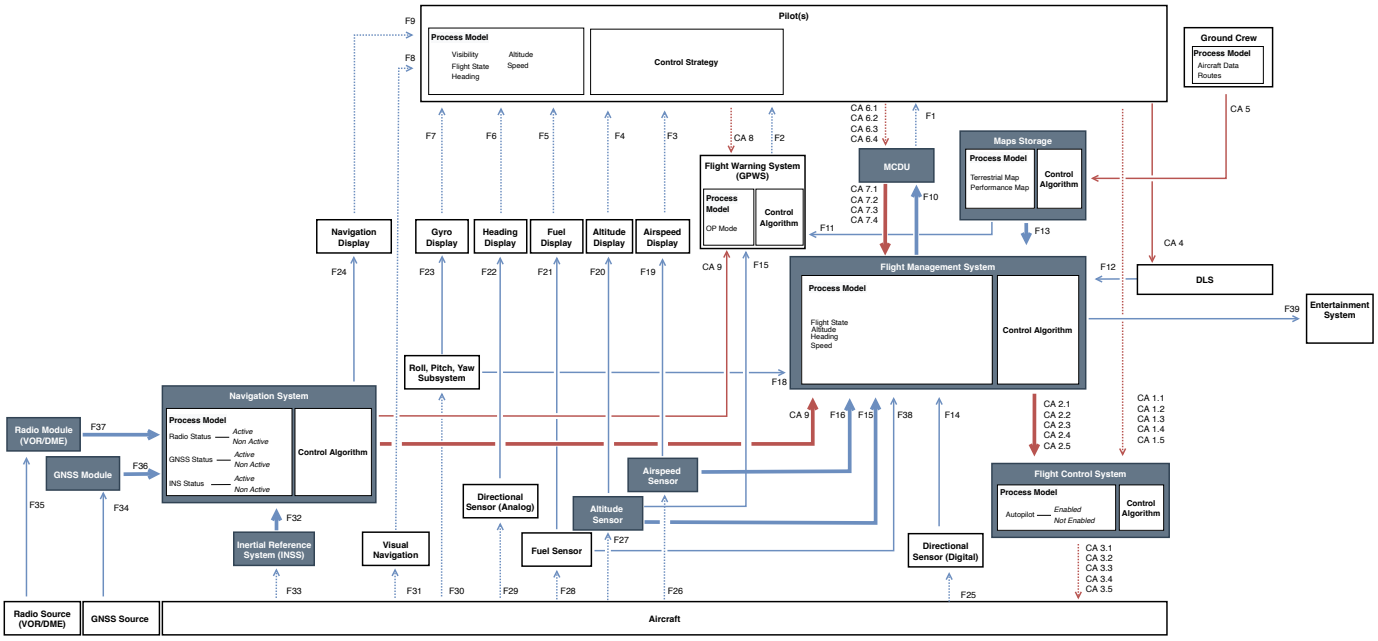

Fig. 3: Safe Control Structure of Avionics - Highlighted elements are analysed in the next Section using Cassandra.

computational tree logic (CTL). Our implementation of the SBM fundamentally differs from those proposed in literature as it is designed to operate in adversarial conditions. In this sense, it allows the integration with a behavioural representation of an adversary who can access and modify information in the CPS. To this extent, we model integrity and availability threats against the components and information flow of the safe control structure and build a synthetic representation of the adversary. We call *instrumented model* of the cyber physical system the parallel composition of the SBM with the produced attacker model [31].

Through formal verification of the reachability of CTL properties against the instrumented model of the CPS, we can *systematically* identify safety critical threat scenarios, i.e., sequences of applications of threats that the adversary can use to cause harm. Finally, we employ a custom algorithm to automatically repeat the formal verification of the properties until no new threat scenario is found. In doing so, we automate the enumeration of a complete set of threat scenarios.

A. Safe Behavioural Model

The *safe behavioural model* (SBM) is a labelled transition system (LTS) which we use to model the propagation of cascading effects across the safe control structure. Our approach is largely based on the propagation model proposed by [14]. For each component in the safe control structure, the safe behavioural model encodes how alterations in its input reflect on the output. As the SBM only captures functional relationships, it is agnostic to specific implementations. In addition, the absence of complex description of behaviours allows to swiftly verify the reachability of unsafe control actions. On the other hand, due to the lack of behavioural details, this approach requires the analysts to manually specify input/output relationships for each component and cannot capture the evolution of the cyber physical system state over time. We use a *network of timed automata* (NTA) [32] to represent the behaviour of the avionics in the safe control structure (Figure 3). For simplicity, we include only a subset of components and information flows (those shown with a darker background in Figure 3). The behaviour of each component, control action and feedback is described used a distinct Timed Automaton (TA) [33]. In particular, the NTA representing our use case includes 17 timed automata, including one for each of the components *altitude sensor*, *inertial reference system* (INSS), *flight management system* (FMS), *map storage*, *multi-function control and display unit* (MCDU), *flight control system* (FCS), and *navigation system*. We model the changes of states in the information flows with the following TAs *MCDU input*, *MCDU output*, *FMS pitch output*[2], *FMS Steering Output*[3], *GNSS position*, *radio position*, *inertial position*, *position (Navigation System Output)*, *map storage data*, and *altitude*. Timed automata in the NTA are synchronised with each other over *synchronisation channels*. The states of the timed automata represent different failure modes for each component or information flow (e.g., loss of availability/reliability of a component or information flow, alternate mode of functioning of a component, etc.). Transitions across locations express dependencies between the failure modes of elements in the SCS, e.g., how a loss of reliability/availability on the input of a controller affects the state of the controller, and/or how, an eventual change of state of the latter affects its output. For each component and information flow we consider two basic failure modes: *unreliable* and *unavailable*.

- If a component becomes unreliable (unavailable), we assume that all of its output become unreliable (unavailable), unless otherwise specified.

[2]For simplicity, this output aggregates changes of *pitch*, *yaw* and *roll*

[3]For simplicity, this output aggregates changes of *thrust* and *altitude*

- If a part of and information flow becomes unreliable (unavailable), we assume that all the components for which the information flow is an input become unreliable (unavailable), unless otherwise specified.

For complex components, such as the controller, we specify a broader set of failure modes where the output is still reliable (e.g., with lower precision) when part of the input has become unreliable (unavailable); an example of this behaviour is given in the description of the navigation system below. Accurately designing failure modes (locations) and the transition between failure modes is crucial to ensure that the consequences of (adversarial) events correctly propagate through the SCS. We assume that complex failure modes (functional input/output relationships) are provided by the design team. In Figure 4 we show the timed automaton that represents the *behaviour* of a simplified flight control system (FCS). For simplicity, the FCS

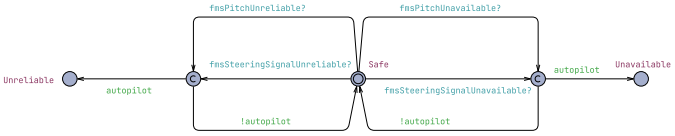

Fig. 4: Flight Control System (TA)

accepts two inputs from the flight management system. *FMS Steering*, aggregates requests of changes of thrust, change of altitude and change of heading, while *FMS Pitch* aggregates pitch, yaw and roll. The FCS starts in a safe location where both its inputs are available and reliable, and so is its output. If one of the input becomes *unavailable* or *unreliable*, the output becomes *unavailable* or *unreliable* respectively - when the autopilot is engaged. In Figure 5 we show the timed automaton

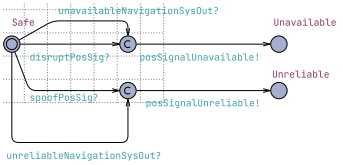

Fig. 5: Position Signal (TA)

that encodes the states of the position signal (information flow) which is an output of the NS and an input of the FMS. The position signal starts in a *Safe* location and becomes *unreliable* (or *unavailable*) as a result of an attack: 1) against the navigation system (NS) (threat to source) or 2) against the position itself (threat to information flow data). In Figure 6, we show the TA that encodes the behavioural model of the navigation system (NS) - a behaviour more complex than that of the FCS shown earlier. The NS uses a weighted voting algorithm of its three inputs to estimate the current position of the aircraft (output). The first input is provided by the GNSS service, which is considered the least reliable as vulnerable to spoofing attacks. The second input comes from the radio receiver. This is considered more precise and reliable than GNSS although its availability is not guaranteed for the entirety of the route. Although more difficult to attack than GNSS, the radio signal comes from an external source and cannot be considered secure. Finally, the third input comes

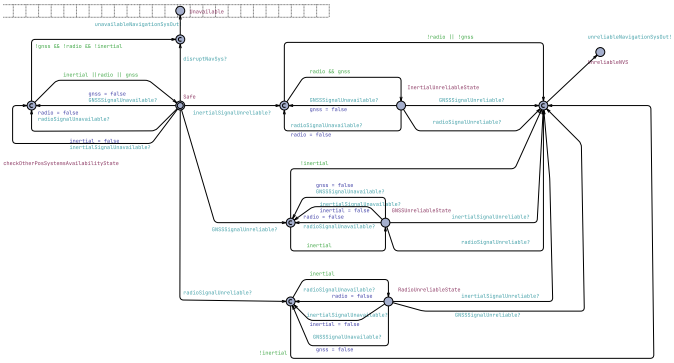

Fig. 6: Navigation System (TA)

from the inertial reference system (INSS) and is considered the most secure as it does not rely on external communications, however its precision is low and degrades over time. The NS starts from a *Safe* location where we assume that the inputs are all *available* and *reliable* and so is its output. 1) When the inputs are all available, but provide different values, the NS trusts the INSS above all. 2) A degraded mode allows the NS to trust GNSS and Radio over the INSS when GNSS and Radio agree. 3) Additional degraded modes allow the system to operate when one or two inputs are not available. In this case, the order of trust is INSS, radio, and GNSS. 4) if all three inputs are unavailable, then the NS becomes unavailable. 5) the output of the NS becomes unreliable when multiple adverse events occur (e.g., radio and INSS are unavailable and GNSS provides unreliable values, or GNSS and radio start providing unreliable values as a result of a maliciously induced fault).

B. Threat Model

To formally verify the reachability of unsafe states in adversarial conditions, we build a synthetic model of the adverary's possible behaviour. We first define the trust boundaries and systematically model integrity and availability threats against the components, control actions and feedback in the SCS. For simplicity, we do not consider threats against *confidentiality* which (generally) do not impact safety. We assume that an attacker can conduct *tampering*, *spoofing*, and *denial of service* (DoS) attacks against the components, control actions and feedback of SCS (Figure 3) which are also included in the SBM. Overall, we consider a total of 34 threats (17 threats to availability and 17 to integrity).We use the threat model to build a synthetic attacker. This is a timed automaton which starts from a *Idle/Ready* location from where any threat can be launched in a non deterministic fashion. When a threat is launched, the target enters a failure mode the effects of which propagate through the safe control structure.

C. Formal verification

The safe behavioural model captures how conditions such as *alternate modes of functioning*, but also loss of *availability* or *reliability* affect control actions and feedback, and how their effects propagate through the control structure. In this paragraph, we build a correspondence between locations of TA representing control actions and the types of unsafe control

actions defined in STPA. In building such correspondence we assume that a *loss of availability* of a control action can lead to: 1) an unsafe control action of the type **non provided**. 2) an unsafe control action of the type **provided too late**, if the loss of availability is temporary. 3) an unsafe control action of the type **provided for too short**, if the loss of availability prematurely ends the application of the control action. Similarly, we assume that a *loss of reliability* of a control action leads to: 1) an unsafe control action of the type **non provided**. If the control action is not applied due to an unreliable information flow. 2) an unsafe control action of the type **provided**. If the control action is provided when it should not due to an unreliable information flow. 3) an unsafe control action of the type **provided too early** or **too late**, if the loss of reliability affects the timing of the control action. 4) an unsafe control action of the type **provided for too short** or provided for **too long**, if the loss of reliability respectively ends the application of the control action before time, or when it is too late.

Safety specifications are expressed in TCTL as reachability statements of the form: $E <> \phi_{UCA}$, i.e., *there is at least one computation where* ϕ_{UCA} *is true* [34]. For example, we define the following two statements to verify UCA_1, UCA_2, UCA_3, UCA_4, UCA_5, UCA_6, UCA_7, and UCA_8:

1) $\phi_1 := E <> fmssteering.Unreliable$
   Explanation: Is there a situation where *FMS Steering Output* becomes unreliable following an attack ($UCA_1 - UCA_8$)?
2) $\phi_2 := E <> fmssteering.Unavailable$
   Explanation: Is there a situation where *FMS Steering Output* becomes unreliable following an attack ($UCA_4 - UCA_8$)?

The formal verification of ϕ_1 and ϕ_2 on the instrumented model allows us to explore all the possible combinations of threats which alone, or through cascading effects lead to safety violations. If the reachability statement is satisfied, the model checker also returns a witness trace as proof of the reachability. The trace includes the ordered sequence of threats leading to the UCA as well as the state of the components when the threats are effective (window of opportunity). In Cassandra, we include a custom algorithm to enumerate all threat scenarios that verify the reachability of the same UCA. This enables us to uncover a set of scenarios that is complete with respect to the threats, component behaviours and bounds imposed on the model checking. We show the outcome of the

TABLE III: Threat scenarios found.

Hazard	Spec	T	CT
H_1	ϕ_1	12	$30.92s$
H_1	ϕ_2	9	$31.387s$

verification of ϕ_1 and ϕ_2 leading to the hazard H_1 in Table III. For each ϕ we also report the total number scenarios found (T), and the cumulative time (CT) taken to verify each ϕ. Cassandra finds 12 threat scenarios potentially leading to the hazards H_1 through ϕ_1, while 9 additional scenarios leading to H_1 are found through the verification of ϕ_2. Figure 7 shows an example of the threat scenario tree derived for H_1 through the verification of ϕ_1. Interestingly, while the last four scenarios are straightforward as the threats directly affect the FMS or its input, scenarios $1-5$, show all the combination of attacks

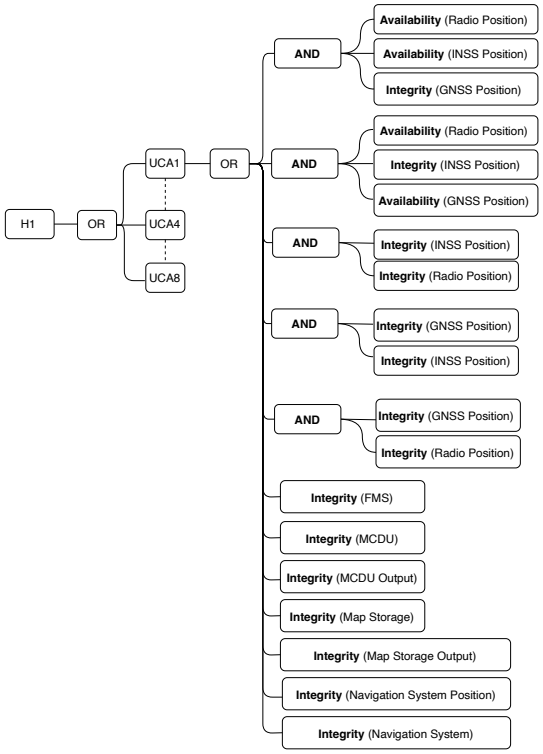

Fig. 7: Threat scenarios tree (UCA_1)

on the INSS, GNSS and Radio subsystems that can affect the decisions taken by the FMS while the aircraft is on autopilot.

VI. Attack Graph Analysis

In the second step of Cassandra, we employ MulVal [35] to derive the safety critical attack paths i.e. the sequences of privileges an attacker needs to obtain to execute a threat scenario and thus cause a hazard. The attack graph of the CPS shows the paths leading to a target threat scenario. For brevity, we focus on two types of privileges within the architecture: *network access* and *code execution*. This step of Cassandra relies on two assumptions. First, we assume that the *architecture of the aircraft is known*. It describes the network topology and also contains information on the hosts, e.g., hardware model, software version, etc.. We also assume that a security assessment has been carried out to identify the vulnerabilities and weaknesses that affect the different components, and the privileges such vulnerabilities would grant to an adversary following a successful exploitation. We leverage the usual traceability requirements - normally adopted within the development life-cycle of safety critical systems - to link threats against elements of the safe control structure to components in architecture, e.g., which privileges enable which threats. In Table IV, we show the subset of threats we have considered in our proposed use case. In essence, an adversary that acquires network access to the most critical domain on the aircraft (*aircraft.control.domain*) poses a threat to the integrity and the availability of most - or all - safety relevant control actions and feedback previously defined in the safe control structure. Among these, for simplicity, we

TABLE IV: Threats and privileges in the use case. Spoofing and Tampering threats affect the integrity whereas DoS affects availability. NA: Network access. Exec: Code execution.

Privilege	Threat
$NA(GNSSNet)$ $NA(control.domain)$	$\{(integrity, GNSSPosition)\}$
$NA(internet)$ $NA(GNSSNet)$	$\{(availability, GNSSPosition)\}$
$NA(control.domain)$	$\{(integrity, INSSPosition)\}$
$NA(control.domain)$	$\{(availability, INSSPosition)\}$
$NA(control.domain)$	$\{(integrity, RadioPosition)\}$
$NA(control.domain)$	$\{(availability, RadioPosition)\}$
$NA(control.domain)$	$\{(integrity, Position(NavSysOut))\}$
$NA(control.domain)$. $Exec(FMS, root)$ $Exec(DLSr, root)$	$\{(integrity, mapDataOut)\}$
$NA(control.domain)$. $Exec(FMS, root)$	$\{(integrity, fmsSteeringOut)\}$
$NA(control.domain)$	$\{(availability, fmsSteeringOut)\}$
$NA(control.domain)$	$\{(integrity, mcduOut)\}$

only consider *GNSS Position*, *Radio Position*, *INSS Position*, *NavSysOut*, *mapDataOut*, *fmsSteeringOut*, and *mcduOut*. Similarly, an adversary who gains network visibility of the GNSS receiver (e.g., via strong satellite signal) poses a threat to the integrity and/or the availability of the *GNSS Position* feedback. Finally, code executions privileges on the *DLSLoader* and *FMS* are a threat to the integrity of *mapDataOut* and *fmsSteeringOut*.

A. Generation of the attack graph

Using MulVal [35] we generate the attack paths leading to the execution of the threat scenarios relevant to UCA_1 in $three$ distinct cases: 1) Electronic Flight Bag (EFB) is provided by the airline and has access to the internet, or is available under bring-your-own-device (BYOD) policy (T_1). 2) EFB is provided by the airline and cannot access the internet, but is enabled with network access towards airline owned enterprise network (T_2). 3) EFB does not have access to the internet (T_3). To perform this analysis we *run MulVal* providing, as input, $three$ variations of the aircraft architecture, each encoding a different case. The attack graph generation process (outlined in [10]) uses the target scenarios as an additional input; we use threat scenarios $TS-1$ to $TS-12$ for this. The attack graph produced by MulVal is a *directed* tripartite graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ linking pre-conditions to attack steps and their post-conditions. Pre-conditions represent privileges required to perform the attack step but also security conditions (e.g., software running on a host, a network service, a vulnerability, etc.). Post-conditions represent the privileges obtained. In the first scenario (EFB with connectivity to the internet), the attack graph produced has 136 vertices and 155 edges, representing all the possible known attack paths that enable the execution of the attack. With the same settings, the attack graph produced when EFB has limited connectivity (no internet access) is slightly smaller (126 vertices and 143 edges). Finally, no attack paths are found when the EFB has no connectivity at all and at least two among GNSS, INSS and Radio are operating correctly. This is expected as we are not considering physical attacks on the EFB. However, if we run Mulval while INSS and Radio are down (e.g., due to a failure), MulVal correctly finds that GNSS spoofing is a valid path leading to UCA_1. Due to the imposed limit of pages, we are not able to show the attack graphs produced by MulVal, however, they are available on our repository [4].

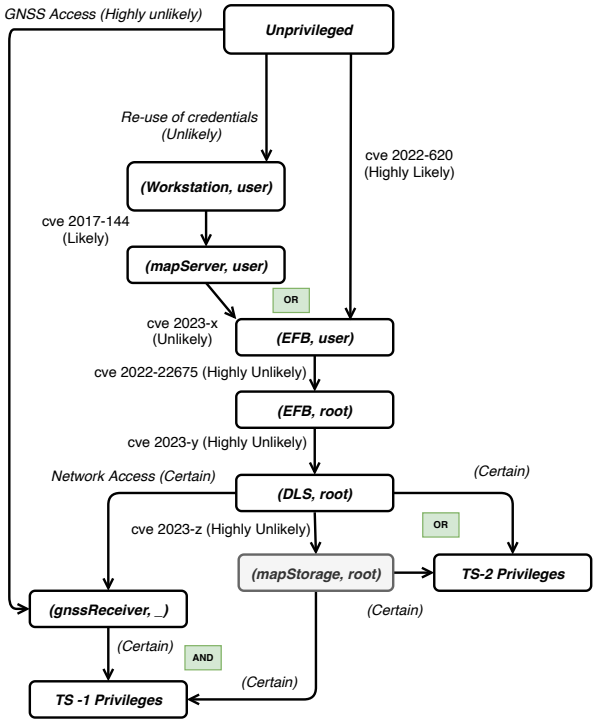

Fig. 8: Attack Graph - (Scenario 1).

B. Likelihood of Threat Scenarios

We employ Bayesian Attack Graphs (BAG) [36] to compute the marginal probabilities of an adversary being successful in a threat scenario. The graph produced by MulVal is a tripartite directed graph which also admits cycles. In order to apply Bayesian inference, we need to transform the graph produced by MulVal into a Directed Acyclic Graph (DAG) where the vertices represent privileges and edges represent the exploitation of a vulnerability (or a weakness). We remove cycles by enforcing the assumption that the attacker never releases privileges once it acquires them. Then, we transform the MulVal graph into a DAG. In essence, privileges in MulVal are treated as privileges in the DAG, and MulVal *derivation rules* are used to re-construct the edges. Figure 8 shows the attack graph produced for the case T_1, with targets $TS-1$ and $TS-2$, with

($TS-1$): *Integrity(GNSS Position)* AND *Availability(Radio Position* AND *Availability(INSS Position)*.
($TS-2$): *Integrity(MapStorageOut)*

We use Bayesian inference as described in [24] on the resulting DAG. Vertices and are modelled using Bernoulli random variables with the probability of the vertex v_i being *compromised* by the adversary being $P(V_i = 1) = p$ [24]. BAG edges model dependencies between vertices. In this case, a dependency

[4]https://github.com/rissgrouphub/dasc-43-submission

represents a vulnerability that the attacker exploits to acquire a new privilege. The weight p_{ei} of the $i-th$ edge reflects the likelihood of the vulnerability i being exploited. We have used the following values for the inference *Highly-Unlikely* $= 0.1$, *Unlikely* $= 0.2$, *Likely* $= 0.5$, *Highly-Likely* $= 0.8$, and *Certain* $= 1$. Table V shows the marginal probabilities of the attacker successfully executing a threat scenario ($TS-1$ or $TS-2$) in the three different cases. Overall, we obtain that,

TABLE V: Marginal Probabilities

Configuration	TS	Marginal Probability
EFB Internet access	$TS-1$	$3*10^{-4}$
EFB Internet access	$TS-2$	$26*10^{-3}$
EFB Airline restricted access	$TS-1$	$<10^{-5}$
EFB Airline restricted access	$TS-2$	$2*10^{-4}$
EFB disconnected	$TS-1$	$\leq 10^{-1}$
EFB disconnected	$TS-2$	NA

in the first configuration the marginal probability of $TS-1$ is $3*10^{-4}$ while that of $TS-2$ is $26*10^{-3}$. This is coherent with $TS-1$ requiring more conditions to be fulfilled than $TS-2$. Marginal probabilities of the two threat scenarios are, respectively, $<10^{-5}$ and $2*10^{-4}$ under the second configuration, when the EFB has limited connectivity. Finally, if connectivity on the EFB is disabled, $TS-2$ cannot be achieved while $TS-1$ is achievable with marginal probability of 0.1 if and only if the NS is working in alternate mode GNSS only (worst case). However, without access to the aircraft through the EFB, the adversary would not be able to directly cause a loss of availability of both radio and INSS systems. Hence, the adversary needs the two systems to have failed independently already for the attack to be successful.

VII. Discussion

Using Cassandra we have evaluated the risk associated with $three$ different policies regarding management and configuration of EFB. The application of the first step of Cassandra on a subset of our aircraft SCS led to the discovery of 23 threat scenarios, ordered sequences of threats that malicious actors can actuate to cause a hazard. We leveraged the integration of the aircraft control structure with its architecture to generate the attack paths leading to the privileges that the attacker needs to engage in 12 of the found threat scenarios. We have identified two main paths of compromise starting from outside the system perimeter when the EFB is equipped with internet connectivity. The first path reaches the ACD through the airline enterprise network which, for simplicity, we have represented as a flat network with two hosts, a workstation, and a map server. By reusing credentials [37], [38] the adversary can access the workstation and then pivot towards the map server. Network visibility on the EFB enables them to exploit `CVE-2023-x` - which we have assumed. A second path leads the adversary from the outside directly to the EFB through the exploitation of `CVE-2022-620`. Assuming that `CVE-2023-x` cannot be exploited from the internet (e.g., traffic blocked), the attacker can exploit `CVE-2022-22675` to acquire root privileges on the EFB. Then, they can exploit `CVE-2023-y` and `CVE-2023-z` ASD and ACD. The edge on the far left of Figure 8 shows that the GNSS service is also reachable from outside system perimeter. As expected, the attack graph generated when the EFB does not have internet access does not contain the edge `CVE-2022-620` Figure 8. The qualitative analysis of the attack graph shows that the EFB is a bottleneck in the attack graphs. It also shows that the feasibility of threat scenarios depends on configuration of the aircraft architecture; by disconnecting the EFB from the internet and then from the airline network, we increasingly limit the number of threat scenarios that can be exploited. Trading off usability - workload imposed on the crew - against the security gains requires a quantitative analysis of the attack graphs produced. We do so by reasoning over the probabilities using Bayesian inference in the third step of Cassandra. The analysis of the marginal probabilities of threat scenarios yields three key results. Firstly, we show that the third configuration (EFB isolated) is the one that carries the least risk, which is intuitive. However, it also increases the workload for the crew, as a limited connectivity also reduces the availability of automation in uploading data to the EFB. Secondly, low level security controls (e.g., fixing of vulnerabilities) applied on airline corporate network and devices are critical to contain the risk of successful attacks targeting the ACD through the EFB. In our use case, we show that, through the containment of vulnerabilities `CVE-2022-22675` and `CVE-2017-144`, we can reduce the marginal probabilities of $TS-2$ below 10^{-5}. To this extent, it is possible to be compliant with certification while also providing the Flight Crew with automated processes. Thirdly, internet connectivity on devices that interact with the aircraft network (e.g., EFB tablet) is associated with a significant risk and is highly discouraged.

VIII. Conclusions

The increased availability of connectivity leads to desirable gains in ease of use and productivity. Unfortunately, this trend also leads to an increased attack surface. In safety critical systems, such as aviation, identifying and mitigating adversarial threats to safety is necessary to avoid hazards and ensure compliance with applicable regulation. Not all threats impact safety, so it is necessary to identify, characterise and mitigate those that do. This requires a combined safety and security analysis with systematic identification of the threats and their corresponding attack paths. Risk based decisions require a quantification of the risks and of the probabilities involved. The complexity of the systems considered, as well as cost pressures, favour the application of automated and semi-automated techniques over human effort. Cassandra offers a methodology combining safety and security analysis. Through the application of formal methods it enables a systematic identification of the safety critical attack scenarios, which are then mapped to attack paths. This enables a qualitative reasoning for mitigation of the attack paths and probabilistic (Bayesian) reasoning for risk quantification and assurance. Together, they enable to evaluate different system design alternatives and make informed choices. We show the application of Cassandra on a contained and perhaps intuitive case study.

However, it can be applied more broadly across the entirety of the system design. Several simplifications have been made to the models analysed to ensure that the case study could be presented within the imposed page limit. Important lessons learnt in the development of Cassandra concern the integration of the different methodologies, tools and design models across safety, security, systems engineering and verification. These require rigorous mappings to be established across slightly different formalism and tools overlapping in scope but independently developed. This is a difficult task and one that could be significantly facilitated in the future by the co-design of security and safety methodologies and improved availability of system design models. Like all systems, Cassandra has a number of limitations. These mainly stem from the capability of existing tools to cope with existing system complexities e.g., model checking and attack graph generation. Furthermore, Cassandra is not yet able to search design spaces, synthesise mitigation or remediation procedures. Our future work, will be focused on novel techniques to remove these limitations and improve Cassandra's capabilities to deal with dynamical systems and run-time security information.

REFERENCES

[1] I. Moir, A. Seabridge, and M. Jukes, "Navigation syst." in *Civil Avionics Syst.* John Wiley & Sons, Ltd, Aug. 2013, pp. 405–447. [Online]. Available: https://doi.org/10.1002/9781118536704.ch11

[2] N. G. Leveson and J. P. Thomas, "STPA handbook," *Cambridge, MA, USA*, 2018.

[3] S. M. Sulaman, A. Beer, M. Felderer, and M. Höst, "Comparison of the FMEA and STPA safety analysis methods–a case study," *Softw. Qual. J.*, vol. 27, no. 1, pp. 349–387, Mar. 2019.

[4] L. Sun, Y.-F. Li, and E. Zio, "Comparison of the HAZOP, FMEA, FRAM, and STPA Methods for the Hazard Analysis of Automatic Emergency Brake Systems," *ASCE-ASME J Risk and Uncert in Engrg Sys Part B Mech Engrg*, vol. 8, no. 3, 10 2021, 031104. [Online]. Available: https://doi.org/10.1115/1.4051940

[5] W. Young and N. Leveson, "Systems thinking for safety and security," in *Proc. 29th Annu. Comput. Secur. Appl. Conf.*, 2013, pp. 1–8.

[6] W. Young and R. Porada, "System-theoretic process analysis for security (STPA-SEC): Cyber security and STPA," in *2017 STAMP Conf.*, 2017.

[7] I. Friedberg, K. McLaughlin, P. Smith, D. Laverty, and S. Sezer, "STPA-SafeSec: Safety and security analysis for cyber-physical systems," *J. Inf. Sec. and Appl.*, vol. 34, pp. 183–196, 2017.

[8] S. Khan and S. E. Madnick, "Cybersafety: A system-theoretic approach to identify cyber-vulnerabilities & mitigation requirements in industrial control systems," *IEEE Trans. Dependable and Secure Comput.*, 2021.

[9] L. M. Castiglione and E. C. Lupu, "Hazard driven threat modelling for cyber physical systems," in *Proc. 2020 Joint Workshop on CPS&IoT Secur. and Privacy*, 2020, pp. 13–24.

[10] L. M. Castiglione, Z. Hau, K. T. Co, L. Muñoz-González, F. Teng, and E. Lupu, "Ha-grid: Security aware hazard analysis for smart grids," in *2022 IEEE Int. Conf. Commun., Control, and Comput. Technol. for Smart Grids (SmartGridComm)*. IEEE, 2022, pp. 446–452.

[11] A. Abdulkhaleq and S. Wagner, "A systematic and semi-automatic safety-based test case generation approach based on systems-theoretic process analysis," *arXiv preprint arXiv:1612.03103*, 2016.

[12] P. Asare, J. Lach, and J. A. Stankovic, "FSTPA-I: A formal approach to hazard identification via system theoretic process analysis," in *Proc. of the ACM/IEEE 4th Int. Conf. on Cyber-Physical Syst.*, 2013, pp. 150–159.

[13] A. L. Dakwat and E. Villani, "System safety assessment based on STPA and model checking," *Saf. science*, vol. 109, pp. 130–143, 2018.

[14] B. Meng, D. Larraz, K. Siu, A. Moitra, J. Interrante, W. Smith, S. Paul, D. Prince, H. Herencia-Zapana, M. F. Arif *et al.*, "Verdict: a language and framework for engineering cyber resilient and safe system," *Syst.*, vol. 9, no. 1, p. 18, 2021.

[15] G. Macher, H. Sporer, R. Berlach, E. Armengaud, and C. Kreiner, "Sahara: A security-aware hazard and risk analysis method," in *2015 Design, Automation & Test in Europe Conf. & Exhib. (DATE)*, 2015, pp. 621–624.

[16] C. Kolb, S. M. Nicoletti, M. Peppelman, and M. Stoelinga, "Model-based safety and security co-analysis: a survey," *arXiv preprint arXiv:2106.06272*, 2021.

[17] L. Pièrre-Cambacédès and M. Bouissou, "Cross-fertilization between safety and security engineering," *Rel. Eng. & System Saf.*, vol. 110, pp. 110–126, 2013.

[18] S. Longari, A. Cannizzo, M. Carminati, and S. Zanero, "A secure-by-design framework for automotive on-board network risk analysis," in *2019 IEEE Vehicular Networking Conf. (VNC)*. IEEE, 2019, pp. 1–8.

[19] I. Stellios, P. Kotzanikolaou, and C. Grigoriadis, "Assessing iot enabled cyber-physical attack paths against critical systems," *Comput. & Secur.*, vol. 107, p. 102316, 2021.

[20] S. Kriaa, M. Bouissou, F. Colin, Y. Halgand, and L. Pietre-Cambacedes, "Safety and security interactions modeling using the bdmp formalism: case study of a pipeline," in *Int. Conf. Comput. Saf., Rel., and Secur.* Springer, 2014, pp. 326–341.

[21] C. Ponsard, J. Grandclaudon, and P. Massonet, "A goal-driven approach for the joint deployment of safety and security standards for operators of essential services," *J. Softw.: Evol. and Process*, p. e2338, 2021.

[22] M. Barrère and C. Hankin, "Analysing mission-critical cyber-physical systems with and/or graphs and maxsat," *ACM Trans. Cyber-Physical Syst.*, vol. 5, no. 3, pp. 1–29, 2021.

[23] M. L. Olive, R. T. Oishi, and S. Arentz, "Commercial aircraft inf. security-an overview of arinc report 811," in *2006 IEEE/AIAA 25th Digit. Avionics Syst. Conf.* IEEE, 2006, pp. 1–12.

[24] L. Muñoz-González, D. Sgandurra, M. Barrère, and E. C. Lupu, "Exact inference techniques for the analysis of bayesian attack graphs," *IEEE Trans. Dependable and Secure Comput.*, vol. 16, no. 2, pp. 231–244, 2017.

[25] N. G. Leveson, *Engineering a safer world: Systems thinking applied to safety*. The MIT Press, 2016.

[26] R. Patriarca, M. Chatzimichailidou, N. Karanikas, and G. Di Gravio, "The past and present of system-theoretic accident model and processes (stamp) and its associated techniques: A scoping review," *Saf. science*, vol. 146, p. 105566, 2022.

[27] A. Shostack, "Experiences threat modeling at microsoft." *MODSEC@ MoDELS*, vol. 2008, 2008.

[28] C. Specifications, "Acceptable means of compliance for large aeroplanes CS-25," *Eur. Aviation Safety Agency, Amendment*, vol. 24, no. 10, 2020.

[29] J. P. Thomas IV, "Extending and automating a systems-theoretic hazard analysis for requirements generation and analysis," Ph.D. dissertation, Massachusetts Institute of Technology, 2013.

[30] R. Khan, K. McLaughlin, D. Laverty, and S. Sezer, "Stride-based threat modeling for cyber-physical systems," in *2017 IEEE PES Innovative Smart Grid Technol. Conf. Eur. (ISGT-Europe)*. IEEE, 2017, pp. 1–6.

[31] S. Hussain, O. Chowdhury, S. Mehnaz, and E. Bertino, "Lteinspector: A systematic approach for adversarial testing of 4G LTE," in *Network and Distrib. Syst. Secur. (NDSS) Symp. 2018*, 2018.

[32] R. Alur, C. Courcoubetis, and D. Dill, "Model-checking for real-time systems," in *[1990] Proc. 5th Annu. IEEE Symp. Logic in Comput. Science*. IEEE, 1990, pp. 414–425.

[33] R. Alur and D. Dill, "Automata for modeling real-time systems," in *Int. Colloq. automata, lang., and programming*. Springer, 1990, pp. 322–335.

[34] C. Zhao, L. Dong, H. Li, and P. Wang, "Safety assessment of the reconfigurable integrated modular avionics based on STPA," *Int. J. Aerosp. Eng.*, vol. 2021, 2021.

[35] X. Ou, S. Govindavajhala, A. W. Appel *et al.*, "Mulval: A logic-based network security analyzer." in *USENIX Secur. Symp.*, vol. 8. Baltimore, MD, 2005, pp. 113–128.

[36] Y. Liu and H. Man, "Network vulnerability assessment using bayesian networks," in *Data mining, intrusion detection, inf. assurance, and data networks Secur. 2005*, vol. 5812. SPIE, 2005, pp. 61–71.

[37] Mitre, "Blackenergy software s0089 mitre att&ck 2022," 2022. [Online]. Available: https://attack.mitre.org/software/S0089/

[38] Dragos, "Chernovite's pipedream malware targeting industrial control systems (ICS)," 2022.