Considering the Aeronautics Cyber-Security Standards for Multi-Core Platforms

Anthony Fernandes Pires, Julien Brunel, Kevin Delmas

Considering the Aeronautics Cyber-Security Standards for Multi-Core Platforms

Anthony Fernandes Pires
ONERA
Toulouse, France
anthony.fernandes_pires@onera.fr

Julien Brunel
ONERA
Toulouse, France
julien.brunel@onera.fr

Kevin Delmas
ONERA
Toulouse, France
kevin.delmas@onera.fr

***Abstract*—New complex functions are emerging for avionic systems. These new functions ask for high performance computing, which mean the need to embed new type of hardware such as hybrid architectures integrating multi or many-core processors. However, these processors are often Commercial Off-The-Shelf and suffer a lack of documentation and predictability. In the all-connected trend of today digital world, these issues can lead to new security vulnerabilities exploitable by malicious people. In the context of the PHYLOG 2 research project aiming at defining a certification framework for multi-core platforms, we study the aeronautics standards ED-202A/DO-326A and ED-203A/DO-356A about airworthiness security. The objective is to take into account these standards at the level of the multi-core processors in order to ensure the compliance of security assessment and development for certification. We present our review and understanding of the standards and their projection at the level of multi-core platforms. In addition, we describe our application on a use case and report our feedback.**

***Index Terms*—cyber-security, multi-core processors, aeronautics, certification**

I. Introduction

New types of hardware are making their way to avionic systems as new complex functions are emerging, such as pilot assistance or flight supervision coupled to machine learning. These new functions are indeed asking for high performance computing. This implies the need to embed hybrid architectures integrating multi or many-core processors and accelerators. However, these processors are mainly COTS (Commercial Off-The-Shelf), so they suffer from low predictability and a significant lack of documentation. This lack of documentation and the complexity of these processors open new vulnerabilities for cyber-attacks. In addition, the avionic systems are becoming more open and connected in the modern digital era (*e.g.* the use of Electronic Flight Bag), leaving these vulnerabilities more accessible to malicious persons.

Standards such as ED-202A/DO-326A and ED-203A/DO-356A offer guidelines, considerations and certification objectives to address airworthiness security. The airworthiness corresponds to the capacity of an aircraft and its systems to operate safely and to carry out their expected function. The airworthiness security consists in the protection of the aircraft against intentional unauthorized electronic interactions. However, these standards have been created to be applicable at the classical levels of aeronautics development: aircraft, system and item. To be able to reason about airworthiness security for the multi or many-core processors, we need to refine these standards to assess cyber-security at the level of the platform, *i.e.* the processors architecture (hardware) and its executive layer. This is not the system level or the item level, but a level in between.

PHYLOG 2 is a research project[1] supported by DGAC, which aims at defining a certification framework for multi- and many-core hybrid architectures. In this context, we study the security standards ED-202A/DO-326A and ED-203A/DO-356A. The objective is to understand the guidelines and considerations expressed in these standards and how to apply them at a platform level to gain confidence on the airworthiness security of multi-core platforms and ensure compliance with certification.

In this paper, we present the work done to achieve this objective in the context of COTS. It is organised as follows. In Section II, we introduce the cyber-security standards by giving a summary of our understanding of the standards contents and our understanding of the application of a part of the Airworthiness Security Process on the use case example given in the ED-203A/DO-356A. In Section III, we present the challenges brought by the use of multi-core architecture in terms of existing cyber-attacks for processors, and in terms of considered level of development for the application of the standards. In Section IV, we present how we interpret the cyber-security standards at the development level of the platform and we apply this interpretation on a revised version of the Air Management System use case based on a simplified version of a Texas Instruments platform. In Section V, we discuss our feedback and the applicability of the standards at platform level. We conclude in Section VI.

II. Understanding the standards

A. Overview

Standard ED-202A/DO-326A (Airworthiness Security Process Specification) [14] and companion document ED-203A/DO-356A (Airworthiness Security Methods and Considerations) [15] describe the process, guidelines and regulatory considerations to address airworthiness security. From our understanding, the standards offer different kind of information. First, a definition of the fundamental concepts required to

[1] https://w3.onera.fr/phylog/

understand and to conduct the Airworthiness Security Process. Second, an overview of the process, activities and suggested methods to be carried out. Third, a list of Security Assurance Objectives to satisfy at the different level of development. Finally, a set of appendixes on methods and examples of application on different use cases. In this section, we focus on giving an overview of the Airworthiness Security Process, introducing the necessary concepts along the way, and we present our understanding of the application of a part of this process on the use case coming from the ED-203A/DO-356A. We finish with an introduction to the Security Assurance Objectives.

B. The Airworthiness Security Process

A simple representation of the recommended process is visible Figure 1.

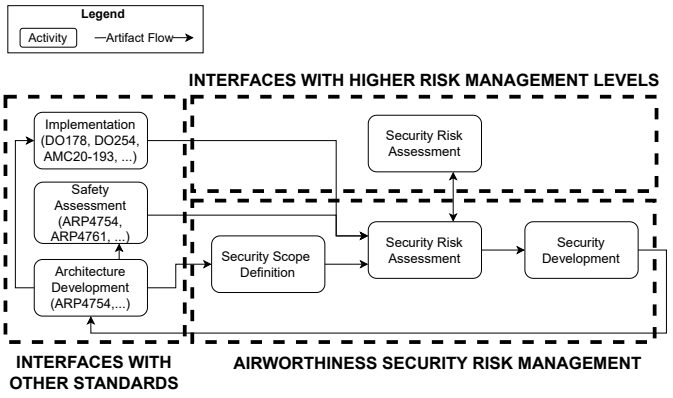

Fig. 1. Airworthiness Security Process preview

It is based on three major activities. First, the security scope definition activity identifies the elements under consideration in the process. The security scope is composed of the assets considered for the airworthiness security, their security perimeter *i.e.* the border between the assets and the external world, and their security environment *i.e.* all the elements external to the security perimeter that can interact with the assets.

The second activity is the security risk assessment that identifies and qualifies the security risks. Based on the definition of the security scope, this activity identifies the threat conditions and their effects, *i.e.* the conditions resulting from potential attacks, the threat scenarios leading to these threat conditions, and the existing security measures. From this information, it allows evaluating the risk for security by calculating the level of threat and evaluating the severity of threat conditions effects. The Level of threat represents the likelihood of a threat condition to occur, while the severity provides a qualitative evaluation of the level of harm of their effect.

The security development activity focuses on the design of security related development (*e.g.*, security measures) adapted to the evaluated security risks and the verification of their effectiveness.

The process always starts by conducting a preliminary Security Risk Assessment at design phase and proceed with one or many iterations of the Security Risk Assessment once the implementation is available. Indeed, once the security risk is evaluated and the security development to mitigate this risk is achieved by modifying the architecture, it is necessary to re-evaluate the risk to find out whether it is acceptable.

The airworthiness security process also interfaces with other activities. It particularly interacts with the airworthiness security process at a higher level. Moreover, the process interfaces with activities linked to other standards. The architecture issued from the Architecture Development following the standard ED-79A/ARP-4754A [17] is necessary to conduct the airworthiness security process. In addition, the failure conditions coming from the Safety Assessment following standard ED-135/ARP-4761 [16] and the implementation following standards ED-12C/DO-178C [13], ED-80/DO-254 [12] and AMC20-193 are required for the Security Risk Assessment. So these interfaces are mainly represented as inputs to the airworthiness security process.

C. Illustration on The Air Management System from ED-203A/DO-356A

The standards illustrate the application of the different activities of the Airworthiness Security process on practical examples. One of this example is the Air Management System (AMS) described in ED-203A/DO-356A. The standard presents the application of the the Security Scope Definition activity and the Security Risk Assessment activity on this particular use case. The Security Development for this use case is not described in the standards and is out of the scope of this paper. The following is our understanding based on the material available in the standard.

1) Description: As described in ED-203A/DO-356A, the AMS of an aircraft fulfils five functions: it provides cabin acclimatization, cabin pressurization, In-Flight information, support for maintenance and support for manufacturing. To manage these functions, the AMS is composed of a Temperature Controller and a Pressurization Controller as depicted in Figure 2.

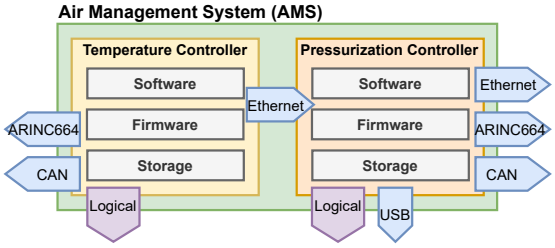

Fig. 2. AMS overview

Each of these controllers contains a software, a firmware and a data storage but also interfaces to the external world. First, there are physical interfaces such as Ethernet interface, ARINC 664 interface, CAN (Controller Area Network) interfaces. The Pressurization Controller is connected to the Temperature Controller via an Ethernet connection. The Pressurization Controller is also the only controller to be equipped with an

USB port. Second, there are logical interfaces that represent digital connections to equipment, such as Aircraft Systems, most of the time via a network of intermediate equipment, e.g. an Ethernet switch. In addition to these descriptive elements, a list of identified Failure Conditions for the AMS is also available in the ED-203A/DO-356A. In this paper, we will primarily focus on the *Loss of pressurization for crew and passengers* which is identified as catastrophic.

2) Definition of the Security Scope: The representation of the complete security scope for the AMS is described in Figure 3 (This figure is based on representations from the ED-203A/DO-356A and [19]). It is composed of the assets under consideration, their security perimeter and their security environment.

a) Assets under consideration: The security scope definition of the AMS starts by the identification of the assets under consideration. Here, the Temperature/Pressurization controllers are considered in their entirety. Each of them includes, as assets: their constituents (e.g. micro-processor), their functions (e.g. *"Provide Cabin pressurization"*), their information (e.g. software, firmware, data storage, etc), their interfaces.

b) Security Perimeter: The Security Perimeter is the border between the assets and the external world. For the AMS, it is composed of the physical and logical interfaces of the two controllers. The physical interfaces include the interfaces to A664 Switch, to ethernet switch, to Maintenance Ground Support Equipment (GSE) and the USB interface. The logical interfaces are composed of the interfaces to Bleed System, to EFB, to QAR, to IFE File Server, to Avionic Systems and to Airline and Manufacturer network.

c) Security Environment: The Security Environment represents the external world of the assets and what can interact with them. It is the place where attacks originate. In the case of the AMS, the security environment includes MRO personnel, pilot, first officer, operator's personnel, operator's maintenance personnel, airline ground infrastructure, manufacturer ground infrastructure and avionic systems. The security environment also covers security assumptions that have been made. One example for the AMS is *"The Pressurization Controller can be updated via GSE or embedded Ethernet switch"*.

3) Security Risk Assessment: A part of the security risk assessment activity on the AMS is described in the ED-203A/DO-356A. Here we summarize the threat conditions identification, the threat scenario identification and the security measure characterisation available in the standard. We then carry out ourselves a level of threat evaluation on a security measure of a scenario in order to illustrate this particular part.

a) Threat condition identification: An example of Threat Conditions identification on the AMS as presented in ED-203A/DO-356A is given in Table I. It describes a threat condition impacting the asset "Logical interface to the Bleed system" and resulting in the loss of pressurization for the crew and passengers. It is is associated to the failure condition coming from the safety assessment, *Loss of pressurization for crew and passengers*. This threat condition considers the corruption of the pressurization controller leading to the dispatch of misleading commands to bleed system. In terms of impact, each threat condition affects a security attribute of an asset. There are typically three security attributes which are considered: confidentiality, integrity and availability. They are referred as CIA. Here, the threat condition is considered as a loss of integrity for the logical interface to the Bleed system and the severity of the effect is identified as catastrophic.

TABLE I
EXAMPLE OF A THREAT CONDITION FOR THE AMS

Threat Cond.	Asset	Attribute (CIA)	Description	Effects	Severity
TC.1	Logical Interface to Bleed system	Loss of Integrity	Misleading commands to bleed due to Pressurisation Controller corruption	Loss of pressure control for crew and occupants	Catastrophic

b) Threat scenario identification: An example of threat scenario identification is given in Table II. It represents a scenario where a criminal, terrorist or insider uses the Wireless Connection to bypass security measures[2], to access the Pressurization Controller Storage and to achieve the threat condition presented in Table I.

TABLE II
EXAMPLE OF A THREAT SCENARIO FOR THE AMS

Threat Sc.	Threat Sources		Attack Path	Security Measures	Threat Cond.
	Attacker	Attack Vector			
TS.1	Criminal, Terrorist, Insider	Wireless connection	Wireless Bridge, Ethernet switch, Pressurization Controller storage	Flight Phase (SR1), Wireless Bridge Access Control	TC.1

c) Security Measures Characterization: In the ED-203A/DO-356A, the Wireless Bridge and its access control is used as example for Security Measures Characterization, as it is on the attack path of the scenario defined in Table II. A summary of this characterization extracted from the ED-203A/DO-356A is presented in Table III. It offers password protection to access the wireless network but its main vulnerability is that it comes with default login and password at the delivery of the aircraft. If the credentials are not changed by the operator, an attacker with knowledge of the default settings can exploit this vulnerability.

d) Level Of Threat Evaluation: Following the examples of methods available in ED-203A/DO-356A, the assessment of the level of threat can be carried out in different ways. Here, we consider the assessment of the level of threat based on the evaluation of the effectiveness of protection. This kind of evaluation depends on three criteria. First, the preparation means *i.e.* is previous knowledge required to conduct the

[2]In the table, SR1 represents the Security Requirement 1 as defined in ED-203A/DO-356A: *"The Pressurization Controller shall only accept external connections routed via Wireless Bridge when the aircraft is on-ground and engine is off."*

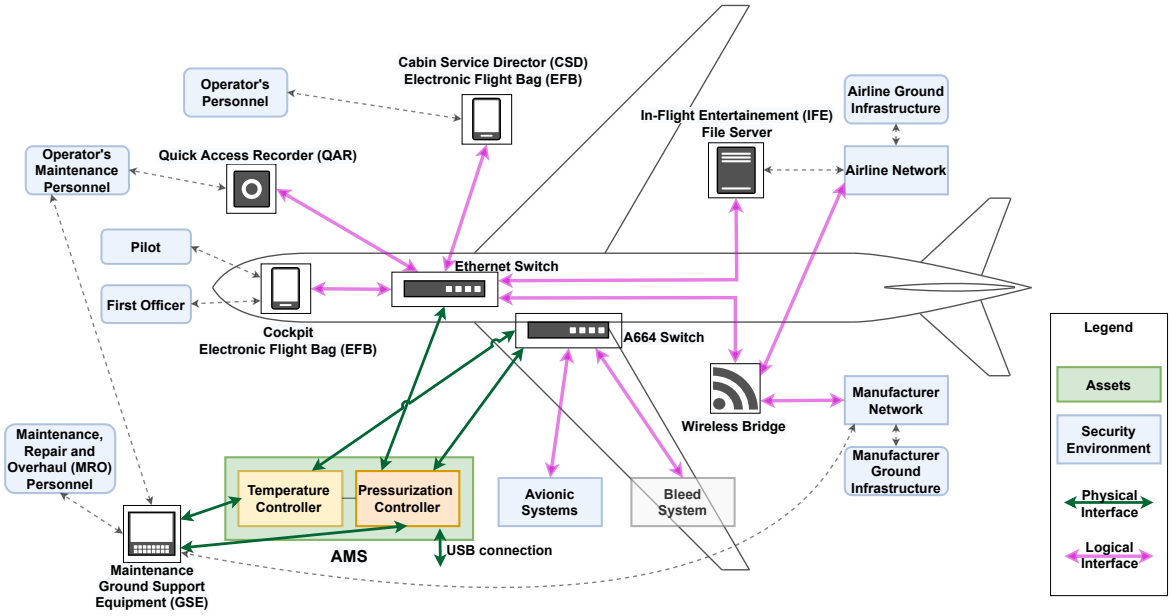

Fig. 3. AMS Security Scope from ED-203A/DO-356A

TABLE III
EXAMPLE OF CHARACTERISATION OF THE SECURITY MEASURE WIRELESS BRIDGE ACCESS CONTROL

Description	Protected Assets	Capability	Type of effect	Position in the architecture	Known vulnerability	Dependencies
Standard wireless access point with optional channel encrypting and access controls	AMS system, functions and data	Provide password protection, interface hardening, etc	Preventive	In the security perimeter border	Delivered with standard user name and password	Need to enforce password definition after delivery or replacement

TABLE IV
LEVEL OF THREAT DEPENDING ON EFFECTIVENESS SCORE

Level of threat	Effectiveness	A
Very High	None	< 7
High	Basic	≥ 7
Moderate	Moderate	≥ 13
Low	High	≥ 19
Extremely Low	Very High	≥ 25

attack? Second, the window of opportunity *i.e.* when will the attack be possible? Finally, the execution means, which prerequisites need to be carried out?

For each of this criterion, a score table is presented in ED-203A/DO-356A in order to give a score depending on predefined answers for the associated question. By combining the three scores, the user obtains an effectiveness score of the protection, denoted A. The level of threat can then be decided following the score table presented in Table IV. It is then up to the user to combine the effectiveness scores of the different security measures, following specific rules defined in ED-203A/DO-356A annex E, to obtain the total score of the effectiveness of protection and so the level of threat of the threat scenario.

For the AMS use case, we tried to carry out this assessment on the Wireless Bridge Access Control in the context of the scenario presented in Table II, as the illustration on this example does not seem to be available in the standards. In terms of preparation means, the Wireless Bridge Access Control corresponds to a standard equipment and uncontrolled information, as the equipment is delivered with standard login and password. The obtained score for this criterion is then 2. In terms of window of opportunity, the access to the wireless bridge is only available on ground engine off which is a reduced window. The evaluated score is then 2. In terms of execution means scale, the attacker needs to be proficient with standard equipment to gain access. The given score is then 4.

The total computed effectiveness score for the Wireless Bridge Access Control is 8. Following Table IV, it corresponds to a basic effectiveness and a high level of threat for this protection. However, other security measures are available for this particular threat scenario and would need to be taken into account to obtain the final level of threat.

D. Assurance Objectives

Finally, the standards also defined security assurance objectives for certification. There are 39 objectives organised following 13 categories (*e.g.* security risk assessment, design, verification, etc). Each objective is tagged with its scope of application, *i.e.* Aircraft development, System development or Item development and its level of application according to the Security Assurance Level (SAL). In the same vein as the DAL, the SAL represents the level of rigor to demonstrate, in terms of security, for a product and its development process. It goes

from level 0, no protective effect, to a maximum level of 3. It is determined based on the severity of threat condition effects to which the product is exposed to and it is assigned to the security measures and assets of the product. SAL is considered out of the scope of this paper, as we reviewed every objective in our work.

To give an example of Assurance Objective, let us consider for the remainder of the paper the objectives O1.1, O1.2 and O1.3 from the Security Risk Assessment category, all applying to the Aircraft and System development level. They are respectively: The security scope is established and validated (O1.1), the Threat Condition Identification and Evaluation is complete and validated (O1.2), the Preliminary Aircraft/System Security Risk Assessments and Aircraft/System Security Risk Assessments are performed and consistent with related aircraft/system safety assessments (O1.3).

III. The security challenges brought by Multi-core Platform

A. Cyber-Attacks at the level of the processors

Many works of the literature have identified cyber-attacks targeting processors, including multi and many-core processors. For instance, [21] presents a sophisticated attack that retrieves information on a secret cipher key by observing the shared Last Level Cache (LLC) of a platform. This attack is called *Flush + Reload* and illustrates a larger category of attacks called side channel attacks. Side channels attack principle is to infer information by observing a phenomenon correlated to the computation (e.g. power consumption, execution time, sound, etc.). Here the attack context is a multi-core platform with shared LLC. On one core, a victim program encrypts a message with its secret key. On another core, an attacker program seeks to retrieve the key to decrypt the message. All encryption programs are considered accessing the same memory location where its instructions are stored. The cache structure of the platform is said to be inclusive, *i.e.*, when a user flushes a line in cache, the line is flushed from all cache levels and for all users. The encryption algorithm used is a non-protected version of RSA. The RSA algorithm is composed of three basic operations: square, multiply, modulo. In a non-protected version of RSA, the order of execution of these operations depends on the bit stream of the secret key. Hence, the order of the operations executed by the victim informs on the secret key. The scenario of the attack is the following. When the victim is encrypting data, only one out of the three operations will have a reduced access time, meaning that it has been loaded in cache by the victim. After the attacker measured the time of access to all the commands and found the one executed by the victim, it flushes all the three operations from the cache and waits for the victim to execute another operation. Repeated during the whole encryption, this process gives the sequence of executed commands, which informs on the secret key used by the victim. In terms of security measures, it is quite easy to protect against some of this type of attacks. However, it is very difficult to protect against all of them. Some security measures may need hardware modification or could be only software. For instance, an easy way of protection for the case described in [21] is to make the algorithm load all the operations at each step whatever the one it executes currently. The observation of the exploitable phenomena is then mitigated.

Other versions exist such as *Prime + Probe* [10] or *Flush + Flush* [5] that are all based on the same concept of obvervable phenomena but with different configurations and efficiency. Additional ways to exploit those phenomena via side channel attacks are described in [22]. From another perspective, work such as [20] studies protection mechanisms against side channels attacks in the context of Simultaneous Multi-Threading (SMT) processors. Indeed, SMT architecture allows multiple threads to compete for shared resources on a single core. Even if it offers performance benefits, it comes at the cost of higher security risks, especially from side channels attacks which exploit shared resources. The protection mechanism described in [20] is based on a spatial and time partitioning of the execution units and ports to prevent side-channel execution on this kind of processor.

Side channel attacks can also be part of more recent sophisticated attacks, such as Meltdown [7] or Spectre [6]. Both of these attacks take advantage of a specific feature that is implemented in modern processors called out-of-order execution. This feature allows a processor to speculate on future operations and schedule them to idle execution units. If the speculation becomes correct, the changes induced by the operations are applied otherwise they are discarded. It increases performance but provides observable side effects which can be exploited. For instance, Spectre attacks trick a victim into speculatively performing operations that would not happen during the nominal program execution and which leak confidential information. The side effects can then be exploited to retrieve this information via side channel attacks.

In a different vein, [9] describes an attack on the Input–Output Memory Management Unit (IOMMU) of a platform that, in a particular configuration, gives a malicious hardware access to the whole memory space. On some processors, Direct Memory Access (DMA) can make direct accesses to the shared memory. It allows to bypass the CPU (and thus the OS if any) so that other devices, such an Field-Programmable Gate Arrays (FPGA), could access in parallel the memory. For example, an FPGA dedicated to cryptographic operations could encrypt and decrypt messages in parallel to the main OS running. But such an unlimited access is not without dangers if the code on the device is malicious or simply flawed. An IOMMU can then be implemented to filter memory access requests. It serves both as a protection against illegal access and as an abstracted interface since it translates virtual addresses to physical addresses. However, at the start of the system, the IOMMU is not immediately configured while the DMA access is enabled immediately after the CPU initialization. The IOMMU is set after some time in the boot process, which leaves a time window for unlimited memory access. This is exploited in this attack. The attack context is

a heterogeneous architecture with one or more CPUs and a Linux OS. The architecture also includes hardware devices like FPGA, external to the OS. These hardware devices are connected to the main memory via DMA and an IOMMU is set to protect the main memory. It is considered that one of the hardware devices is infected by a malicious program at startup. The scenario is the following. On startup, the Linux kernel creates the configuration table of the IOMMU that contains the access policy rules. This table is in the main memory before it is loaded in the IOMMU internal register, where it becomes protected. However, while this table is in the main memory and before it is loaded, the malicious hardware will rewrite it to give itself access to the whole memory space. This scenario is based on the assumption that the DMA access is enabled by default at startup. Nowadays, this is true for many platforms because of legacy reasons, as explained in [9]. Concerning security measures, this work also illustrates the vulnerability that a security mechanism at the level of the platform (the IOMMU) can have and that could be exploited to bypass the protection.

In [4], the authors list a number of attacks targeting Network-on-Chip (NoC). A NoC is a network between System-on-Chips (SoC) which are integrated circuit containing processors. A NoC design combines notions from the network and the hardware domains, including their security vulnerabilities, and creates a completely new set of attack possibilities between processors. For example, we present one of the attacks described in [4], the attack concerning packet corruption at routers. In this attack, a Trojan hardware is inserted in a router of the network and is able to either copy and resend packets, send forged packets or tamper the data inside a transiting packet. The context of the attack is based on two CPU clusters on the network, a sender and a receiver, which need to communicate. A hardware element has been introduced into a router of the network that is on the path between the sender and the receiver. There is no encryption or authentication mechanism implemented on the network and the Trojan is sufficiently stealthy to not be detected by traditional means of network monitoring. The attack scenario is quite simple. The message from the sender to the receiver is intercepted by the Trojan and is modified or additional messages are forged. In terms of security measures, detecting a Trojan hardware is very difficult if not impossible. Current research works are more focused on protecting the network in case of such compromising. Protection mechanisms include what is already done in networks, i.e. error-correcting codes, encryption, authentication, etc. However, this attack requires the installation of hardware on the network, which makes the feasibility of the attack more difficult.

Despite the numerous processor level attacks identified in the literature, we did not find any example of attacks in the aeronautics domain or any practical applications of ED-202A/DO-326A and ED-203A/DO-356A on a multi-core platform.

B. The need to consider a development level of the platform

In this paper, we consider that a platform is composed of several hardware and software items interacting between them. It includes a processor architecture, a platform configuration, which can be hardware and/or software, and an executive layer, i.e. the hypervisor. Cyber-attacks reported in the literature show that it exists many attacks happening at the level of processors and their interactions. Security measures are also defined at this level of details. In this case, it becomes essential to consider the whole content of a platform to carry out the Airworthiness Security process. As seen in Section II, the standards ED-202A/DO-326A and ED-203A/DO-356A consider the traditional development level of an aeronautic development. However, the platform is neither a system nor an item. It is something in-between. It is then necessary to consider an alternative level of development to apply the recommendations of the cyber-security standards. We suggest studying their application at the level of the platform.

IV. Interpretation of the standards at the platform level

Based on our review of the standards ED-202A/DO-326A and ED-203A/DO-356A and the need to reason at platform level, we first extracted the concepts needed to carry out the activities, along with their definitions and relationships. Accordingly, we designed a representation of the security process as we understood it and we outlined it at the level of the platform, along with the Security Assurance Objectives. In addition, we took the AMS example described in Section II-C and we brought it at the platform level to review the activities at this specific level. The work achieved for the process, the assurance objectives and the use case is summed up in this section.

A. Interpretation of the Standards at the Level of Multi-Core Platforms

1) Interpretation of the Airworthiness Security Process: We present our understanding of the Airworthiness Security Process in Figure 4.

In particular, we describe the precise links between the different activities of the Airworthiness Security Risk Management introduced in Figure 1. Three activities are out of the Airworthiness Security Risk Management. The architecture development provides the architecture to the three main activities of the Airworthiness Security Risk Management. The Platform Safety Assessment provides the failure conditions to the Platform Security Risk Assessment and the Implementation activity provides the implementation and derived requirements. From this point, the Security Scope Definition is carried out and supplies the security scope (Assets under consideration, security perimeter, security environment and security assumption) to both the Platform Security Risk Assessment and the Platform Security Development. Based on the provided artefacts, the Platform Security Risk Assessment is first conducted and supplies security requirements, the evaluation of the security risks and their level of threat to enable the

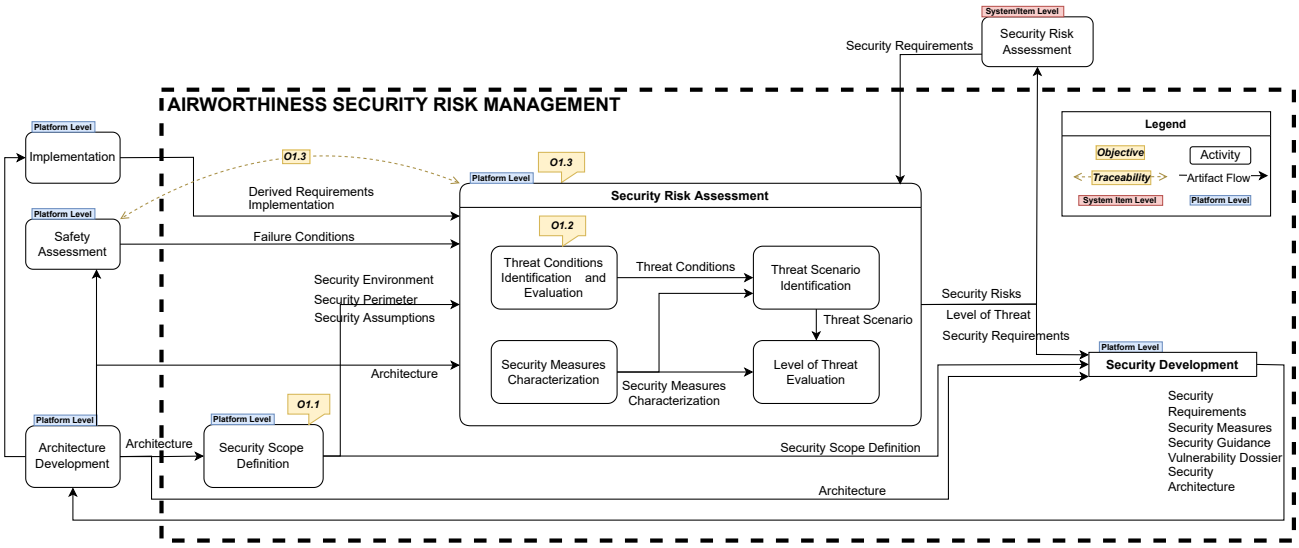

Fig. 4. Extract of our interpretation of the standards of the Airworthiness Security Process

Platform Security Development. The detailed activities of the Security Risk Assessment follow the representation available in ED-203A/DO-356A. Once the Security Development is conducted, it provides all the necessary information to the architecture development in order to take into account the required modifications to mitigate the security risks.

All the mentioned activities happen at platform level. Following Figure 1, there are also links between the activities of the Airworthiness Security Risk Management and the Security Risk Assessment carried out at an upper level. Here we consider that this upper level could be the System level or the Item level.

2) Interpretation of the Assurance Objectives: Assurance objectives have been interpreted for the platform regardless their SAL application. The idea was to obtain an overview of their relevance at the level of the platform. Our review shows that most of them can be interpreted as they are specified at platform level, as long as they now specifically mentioned the platform level. For example for O1.3 mentioned in Section II-D, it becomes *The Preliminary Platform Security Risk Assessments and Platform Security Risk Assessments are performed and consistent with related Platform Safety Assessment*. Only two objectives are deemed not interpretable at the level of platform. They correspond to objectives linked to configuration management process and credentials, which are not specific to the platform, but to the development process of the overall system.

In addition, each objective was mapped to the activity we judged it was related to in our process interpretation. For example, the objectives O1.1, O1.2 and O1.3 mentioned in Section II-D are positioned on the process presented in Figure 4. The three of them are mapped to the respective activities they referred to. In addition, O1.3 mentioned the need of consistency between the security risk assessment and the related safety assessment. This need is interpreted in Figure 4 as a traceability link between both activities.

B. Bringing the AMS use case to the platform level

1) Description: To reason at platform level, the AMS described in Section II is adapted on a simplified representation of the KEYSTONE TCI6630K2L multi-core platform from Texas Instruments composed of only three cores. This representation is presented in Figure 5. It describes the structure of the platform itself and the location of the assets of the AMS. The representation of the structure of the platform follows an initiator-target modelling described in [3]. This modelling is based on three types of components. The initiator component can initiate a transaction, *i.e.* a request for resources. The target component is the final destination of the transaction. The transporter component routes the transaction from the initiator to the target.

In terms of assets, the two controllers composing the AMS are hosted on two different cores. The Pressurization Controller is hosted on a C66 DSP core, namely *CorePac0*. Its software and specific data (computation data, part numbers, certificates, cryptographic keys) are stored in the core SRAM. The Temperature Controller is hosted on an ARM core. Its software is stored in the MSMC SRAM and the regular controller data (Health Monitoring, CAN Messages, ARINC messages, configuration) are stored in the external DDR memory, along the regular controller data for the Pressurization Controller. The firmwares of both controllers are stored in the Boot ROM. In addition to these two cores, only one another core C66 DSP core, namely *CorePac1*, is present in the platform. The core is unused by the AMS but still present for the sake of the example.

2) Security Scope Definition: We conduct the security scope definition on the use case by identifying the assets under consideration, the security perimeter and the security environment.

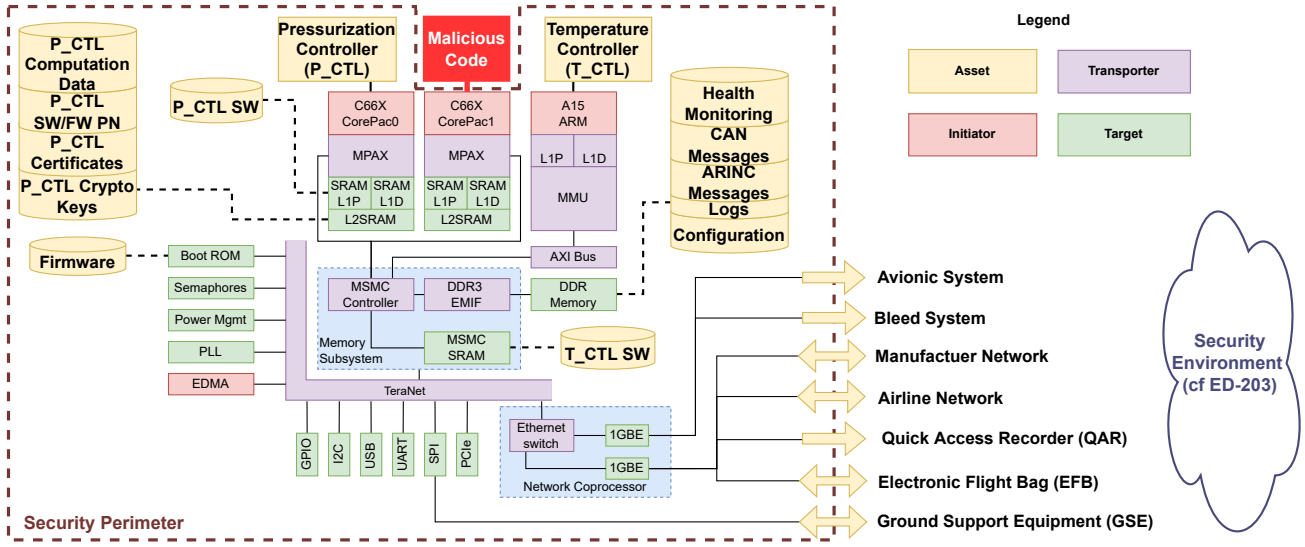

Fig. 5. AMS adapted on a simplified keystone platform

a) Assets under consideration: The assets under consideration correspond to the same assets as in the original AMS example in ED-203A/DO-356A described in Section II, but mapped on different elements (*e.g.* the cores). In this way, we consider both controllers in their entirety (constituents, data and interfaces) as assets.

b) Security Perimeter: Here the security perimeter includes the security perimeter of the AMS defined in ED-203A/DO-356A. It contains the physical and logical interfaces to different equipment, to Bleed System, to IFE File Server, to Avionic Systems and to Airline and Manufacturer network. In addition, at platform level, the security perimeter includes the logical interface with an external code, most likely malicious, present in the unused C66 DSP core CorePac1. This choice is based on an assumption that the external code, which is not part of the AMS, is already uploaded on the platform and executes on CorePac1. This assumption has been made in order to reason about cyber-security at platform level.

c) Security Environment: The security environment is the same as the original AMS example described previously with the difference that it now includes the malicious code.

3) Security Risk Assessment: We conduct the four steps of the Security Risk Assessment on this use case. We start by the identification of threat conditions. We identify two threat conditions representative of our problematic. We continue by the identification of threats scenarios. Finally, we give an example of security measures characterisation and evaluate the level of threat.

a) Threat condition identification: An example of Threat Conditions identification on the AMS platform example is described in Table V. The first threat condition, TC.1, is similar to the one presented in the original AMS example in Section II-C3. It considers the sending of erroneous data to bleed system. It is considered as a loss of integrity for the logical interface to the Bleed system. The second threat condition, TC.2, has been identified especially for this use case. It considers the blocking of data sent by the pressurization controller to the Bleed system. It is considered as a loss of availability for the logical interface to the Bleed system. Both threat conditions can lead to the loss of pressurization for the crew and passengers and are identified as catastrophic.

TABLE V
EXAMPLES OF THREAT CONDITIONS FOR THE AMS PLATFORM EXAMPLE

Threat Cond.	Asset	Attribute (CIA)	Description	Effects	Severity
TC.1	Logical Interface to Bleed system	Loss of Integrity	Pressurization controller sends erroneous data to the Bleed system	Loss of pressure control for crew and occupants	Catastrophic
TC.2	Logical Interface to Bleed system	Loss of Availability	Pressurization controller does not send data to the Bleed system	Loss of pressure control for crew and occupants	Catastrophic

b) Threat scenario identification: Two threat scenarios related to the previously described threat conditions are identified and given in Table VI. The first scenario deals with threat condition TC.2 and considers that the malicious code executing on CorePac1, modifies the MPAX (Memory Protection and Address eXtension) configuration register in order to block CorePac0, *i.e.* the pressurization controller, to access the Ethernet switch of the platform and so to communicate data to the Bleed system. The second threat scenario is associated to threat condition TC.1. It also considers that the malicious code present in CorePac1, modifies the MPAX configuration register but in this case to give itself writing rights on CorePac0 and to corrupt the pressurization controller. In this case, use of side channel attacks can inform the malicious code on where are stored the Pressurization Controller Computation data to corrupt. Note that these scenarios follow the assumption that

TABLE VI
EXAMPLES OF THREAT SCENARIOS FOR THE AMS PLATFORM EXAMPLE

Threat Sc.	Threat Sources		Attack Path	Security Measures	Threat Cond.
	Attacker	Attack Vector			
TS.1	Criminal, Terrorist, Insider	CorePac1	**1-** Malicious code modifies MPAX configuration register to block CorePac0 access to Eternet Switch **2-** Logical Interface to Bleed system is blocked	MPAX, OS Access Control	TC.2
TS.2	Criminal, Terrorist, Insider	CorePac1	**1-** Malicious code modifies MPAX configuration register to give itself writing rights on CorePac0 L2SRAM **2-** Logical Interface to Bleed system is corrupted	MPAX, OS Access Control	TC.1

the malicious code has already been uploaded on the platform. Also note that both the MPAX, which is in charge of enforcing the rules included in the configuration register, and the OS, which is in charge of loading the configuration table into the MPAX configuration register, are countermeasures.

c) Security Measures Characterization: An example of security measure characterization on the platform is conducted on the OS access control. It is a preventive security measure that corresponds to a Linux Operating System (Symmetric Multi Processing). The access control protects the memory storage of the platform but presents a known vulnerability at boot time, as described in Table VII and as seen in Section III.

TABLE VII
EXAMPLE OF CHARACTERISATION OF THE SECURITY MEASURE OS ACCESS CONTROL

Description	Protected Assets	Capability	Type of effect	Position in the architecture	Known vulnerabilities	Dependencies
Linux Operating System (Symmetric Multi Processing)	Memory storage of the platform	Provide memory access protection	Preventive	Inside the security perimeter	During activation of access control mechanisms at boot time, it is possible to modify configuration tables [9]	N/A

d) Level Of Threat Evaluation: For this use case, we used the same type of effectiveness of protection assessment as in the section II-C3, but carried out at a global level on the platform for the considered scenarios. In terms of *preparation means*, the platform can be considered as a special equipment with a specific configuration to host the AMS. So insider knowledge or significant preparation time would be needed for the attack. The obtained score for this criterion is then 6. In terms of *window of opportunity*, the identified attack can only be carried out during a very restricted time slot independent from the flight phase, e.g. during system reboot. The evaluated score is then 8. In terms of *execution means* scale, the attack requires experts in multiple domain in order to be carried out on the platform. The given score is then 12.

The total computed effectiveness score for the platform and the considered scenarios is 26. Following Table IV, it corresponds to a very high effectiveness of protection and a very low level of threat.

V. FEEDBACK AND DISCUSSIONS ON THE CONSIDERATION OF THE INTERNAL PLATFORM FOR SECURITY

We presented our understanding of the standards ED-202A/DO-326A and ED-203A/DO-356A through the AMS use case and we adapted this use case to the platform level in order to reason about cyber-security in the context of the PHYLOG 2 project. The AMS as described in ED-203A/DO-356A lacks some details that made it challenging to understand. The projection at platform level was not an easy task either, example of attacks, or application of the standards, on aeronautic use case were not available in the literature. We had to make assumptions to reason about cyber-security at platform level.

The strongest assumption was made on the attack origin: the malicious code is already uploaded on the platform and executes on CorePac1. The question of how to get to this situation is the most difficult one. The starting point of the attack pathway is in fact external to the platform. We can make the same observation on the attacks on processors described in the literature. As we have seen, the work presented in [9] and [4] consider that the infected hardware device is already present in the architecture. From our point of view, adding a malicious element to the platform would require a lot of effort to bypass a number of existing security measures, e.g. physical access to the platform, ability to load malicious code, etc. This particular problem of loading malicious code challenges the plausibility of the attack as we understand it today. However, cases of more sophisticated attacks on information systems, *i.e.* attacks composed of a series of attacks on different assets to target another specific asset, are appearing today. For instance, the supply-chain attack reported in [11] started by gaining access to the source code of a particular software developed by a company to directly insert malware in it. In this way, the malware was automatically distributed to all the customers in the next update via official channels, allowing the attackers to gain access to customer's information without directly attacking them. The update containing the malware was installed by thousands of customers, including numerous U.S. federal agencies. We could imagine a similar sophisticated scenario happening for the update of a COTS platform. We could also imagine the same scenario for an Open Hardware platform where the shared high-level hardware description of the platform is targeted to impact the final users. This increasing sophistication of cyber-security attacks reinforces our idea of the necessity to study cyber-security at the level of multi-core platforms in aeronautics in order to gain confidence on the protection against attacks reaching this level. However, the security analysis cannot start at the platform level. It needs first to be conducted at an upper level to get a global picture of the origin of the attack and the attack path itself.

Considering the security risk of a multi-core platform comes with challenges. It would first mean study the vulnerability,

threat conditions, threat scenarios and the security measures at this level. In Section III, we reported on existing attacks, scenarios and security measures on processors, including multi-core platforms. In the general case, hardware and software cyber-attacks have been documented in cyber-security databases online. For instance, as mentioned in [8], there are databases dedicated to vulnerabilities found in existing systems such as CVE [2]. This database provides a catalogue of known vulnerabilities in applications, operating systems and hardware products. There are also databases reporting attack patterns as mentioned in [8] and [18], such as *Common Attack Pattern Enumeration and Classification* (CAPEC) [1]. The CAPEC database provides more than 500 known cyber-security attack patterns, including hardware-related ones. While all these databases are not dedicated especially to multi-core platforms, they offer a more global overview of actual known attacks in the cyber-security domain. But vulnerabilities, attacks scenarios and security measures at the level of multi-core platform might be very well dependent of the studied architecture. As we have seen in the example in Section IV, they seem specific to the architecture we used. The use of COTS and their lack of predictability and documentation may also make the task more complex. Further work would be necessary to understand precisely all these challenges by studying the application of the airworthiness security process on a larger panel of multi-core platforms, including Open Hardware solutions.

VI. Conclusion

To conclude, we presented our understanding of the Security Airworthiness process as described in ED-202A/DO-326A and ED-203A/DO-356A and assurance objectives that are relevant at platform level. Our main feedback is that even though the standards provide rich details on the different cyber-security concepts and lay the foundation of a sound assessment process, the process as described in these standards may lack some details that made it challenging to understand for non-insiders. The standards would benefit from a use case which would serve as a common thread for an application of the Security Airworthiness process in order to give an end-to-end example to the future applicant. In addition of our understanding, we applied our interpretation of the Security Assessment process on a platform use case and we opened the discussion on the usefulness of considering the internal platform in the context of cyber-security analysis. Our work is limited to the study of a fictitious use case inspired from ED-203A/DO-356A and to an exploratory overview of the current literature on attacks on processors. Nevertheless, it introduces an example of application of the standards in the case of a multi-core platform. In this sense, it gives a first glance on the difficulties behind an attack inside a multi-core platform, but the increasing sophistication of today cyber-attacks suggests it might be worth starting discussing the application of the standards at platform level.

Acknowledgment

The work presented in this paper is part of the PHYLOG 2 project supported by the Directorate General of Civil Aviation (DGAC). It is funded by the French government through the France Relance program, based on the funding from the European Union through the NextGenerationEU program.

References

[1] CAPEC. http://capec.mitre.org/. Last access: 2024-03-27.

[2] CVE. www.cvedetails.com/vulnerabilities-by-types.php. Last access: 2024-03-27.

[3] F. Boniol, Y. Bouchebaba, J. Brunel, K. Delmas, C. Pagetti, T. Polacsek, and N. Sensfelder. A service-based modelling approach to ease the certification of multi-core COTS processors. In *SAE AEROTECH® Europe*, Bordeaux, France, Sept. 2019.

[4] S. Charles and P. Mishra. A survey of network-on-chip security attacks and countermeasures. *ACM Computing Surveys*, 54(5), 2021.

[5] D. Gruss, C. Maurice, K. Wagner, and S. Mangard. Flush+flush: A fast and stealthy cache attack. In J. Caballero, U. Zurutuza, and R. J. Rodríguez, editors, *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 279–299, Cham, 2016. Springer International Publishing.

[6] P. Kocher, J. Horn, A. Fogh, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom. Spectre attacks: Exploiting speculative execution. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1–19, 2019.

[7] M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, J. Horn, S. Mangard, P. Kocher, D. Genkin, Y. Yarom, M. Hamburg, and R. Strackx. Meltdown: reading kernel memory from user space. *Commun. ACM*, 63(6):46–56, may 2020.

[8] N. Messe, N. Belloir, V. Chiprianov, I. Cherfa, R. Fleurquin, and S. Sadou. Development of secure system of systems needing a rapid deployment. In *2019 14th Annual Conference System of Systems Engineering (SoSE)*, pages 152–157, 2019.

[9] B. Morgan, E. Alata, V. Nicomette, and M. Kaaniche. IOMMU protection against I/O attacks: A vulnerability and a proof-of-concept. *Journal of the Brazilian Computer Society*, 24, 12 2018.

[10] D. A. Osvik, A. Shamir, and E. Tromer. Cache attacks and countermeasures: The case of aes. In D. Pointcheval, editor, *Topics in Cryptology – CT-RSA 2006*, pages 1–20, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[11] S. Peisert, B. Schneier, H. Okhravi, F. Massacci, T. Benzel, C. Landwehr, M. Mannan, J. Mirkovic, A. Prakash, and J. B. Michael. Perspectives on the solarwinds incident. *IEEE Security & Privacy*, 19(2), 2021.

[12] RTCA, Inc / EUROCAE. DO-254 / ED-80 Design Assurance Guidance for Airborne Electronic Hardware, 2005.

[13] RTCA, Inc / EUROCAE. DO-178 / ED-12C - Software Considerations in Airborne Systems and Equipment Certification, 2011.

[14] RTCA, Inc / EUROCAE. DO-326A / ED 202A - Airworthiness Security Process Specification, 2014.

[15] RTCA, Inc / EUROCAE. DO-356A / ED 203A - Airworthiness Security Methods and Considerations, 2018.

[16] SAE / EUROCAE. ARP-4761/ED-135 Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment, 1996.

[17] SAE / EUROCAE. ARP-4754A/ED-79A Guidelines for development of civil aircraft and systems-enhancements, novelties and key topics, 2011.

[18] A. Shaked and Y. Reich. Model-based threat and risk assessment for systems design. In *Proceedings of the 7th International Conference on Information Systems Security and Privacy - Volume 1: ICISSP,*, pages 331–338. INSTICC, SciTePress, 2021.

[19] P. Skaves. Electronic flight bag (EFB) policy & guidance information, paper # 263. In *2011 IEEE/AIAA 30th Digital Avionics Systems Conference*, pages 1–30, 2011.

[20] D. Townley and D. Ponomarev. Smt-cop: Defeating side-channel attacks on execution units in smt processors. In *2019 28th International Conference on Parallel Architectures and Compilation Techniques (PACT)*, pages 43–54, 2019.

[21] Y. Yarom and K. Falkner. FLUSH+RELOAD: A high resolution, low noise, l3 cache Side-Channel attack. In *23rd USENIX Security Symposium*, pages 719–732. USENIX Association, 2014.

[22] Y. Zhou and D. Feng. Side-channel attacks: Ten years after its publication and the impacts on cryptographic module security testing. *IACR Cryptology ePrint Archive*, 2005:388, 01 2005.