Method and framework for security risks analysis guided by safety criteria

Gabriel Pedroza, Guillaume Mockly

Method and Framework for Security Risks Analysis Guided by Safety Criteria

Gabriel Pedroza
Université Paris-Saclay, CEA, List
F-91120, Palaiseau, France
gabriel.pedroza@cea.fr

Guillaume Mockly
Trialog
25 rue du général Foy, 75008, Paris, France
guillaume.mockly@trialog.com

ABSTRACT

As previously discussed [20], the challenges to achieve a consistent intertwining between safety and security are rather diverse and complex. Recent advances in safety and security suggest that risks analyses provide guidance for achieving a comprehensive alignment. However, for many domains, like in aeronautics, security is rather a recent concern whereas aircraft development has been mostly guided by safety criteria for several decades. The referred disparity along with the fact that security is, in many respects, a discipline still in evolution, imposes restrictions for specifying and applying methods to conduct safety and security co-engineering as a unified process. In this paper, we present the progress in the development of a model-based method, a framework and a tool useful to conduct a security risks analysis guided by safety criteria and goals. Among others, the approach relies on know-how found in the state of the art, in standards like ED202, ED203 (EUROCAE)[1], as well as in open knowledge bases like CAPEC and CWE (MITRE)[2]. These sources are integrated which allows the instantiation of patterns of attacks, vulnerabilities, and architectures, which are crucial elements to semi-automate the analysis. A rule-based algorithm for exploring potential attack paths across an architecture is proposed and implemented. The approach is finally demonstrated by analyzing a combined attack-failure path in a Flight Control System which can undermine the safety of a modern aircraft. The framework and tool support seek safety-security by design and aim to facilitate the reuse of case studies and to settle a basis for repeatability and results comparison.

CCS CONCEPTS

• **Applied computing** → **Computer-aided design**; • **Security and privacy** → **Distributed systems security**; • **Computing methodologies** → **Model development and analysis**; • **Software and its engineering** → *Integrated and visual development environments.*

KEYWORDS

Security, safety, risks analysis, co-engineering, attack-failure path, Model Driven Engineering.

ACM Reference Format:
Gabriel Pedroza and Guillaume Mockly. 2020. Method and Framework for Security Risks Analysis Guided by Safety Criteria. In *ACM/IEEE 23rd International Conference on Model Driven Engineering Languages and Systems (MODELS '20 Companion), October 18–23, 2020, Virtual Event, Canada.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3417990.3420047

1 INTRODUCTION

The separate analyses of safety and security during systems design are complex topics and have attracted attention in the research community and industry for several decades. Ongoing discussions and work recently published suggest that safety and security concerns should be addressed following a co-engineering process [15], [16]. Some advantages mentioned in the literature are preventing design flaws due to conflicts between safety and security requirements not identified at early phases of the system development. Many of the approaches proposed to separately conduct safety and security analyses follow a risk-oriented perspective [16]. However, despite the similarities observed in safety and security risks methods, achieving a consistent intertwining is a challenging topic [20]. Indeed, for a comprehensive and unified co-engineering process, the consistency between concepts, metrics, methods and related techniques seems necessary but rather hard to achieve. Among the reasons identified, it can be mentioned the disparity in research and application of techniques: whereas well-known methods (including models and techniques) have been effectively applied to achieve safety, security is, in many respects, a discipline still in evolution, *e.g.* in unmanned aerial systems [14], IoT [10], and vehicular clouds [12]. This assessment is true in particular for the aeronautics domain where stringent certification processes exist and are based upon broad consensus and operable standards. In this paper we present the advances in a method developed to align safety and security risks analyses. The contribution of the method is three-fold. First, the approach consists in to placing in a sequence safety and security analyses as a primary step prior to conduct simultaneous co-engineering. To achieve such alignment, it is proposed to first conduct a safety analysis in order to obtain a set of safety critical events (e.g., critical failures, minimal cut sets). Afterwards, the security analysis considers that the system attackers will target the components whose failure shall lead to the safety critical events. Such alignment allows to enchain safety and security analyses and define metrics for a consistent evaluation of combined failure-attack paths. The second part of the contribution is the integration of open

[1] https://www.eurocae.net/
[2] https://www.mitre.org/

MODELS '20 Companion, October 18–23, 2020, Virtual Event, Canada

ACM ISBN 978-1-4503-8135-2/20/10…$15.00
https://doi.org/10.1145/3417990.3420047

knowledge bases like CAPEC [18] and CWE (MITRE) [17]. It allows to instantiate and reuse patterns of attacks, vulnerabilities, and architecture technology thus easing modeling and design reusage and results comparison. Some model-driven techniques are also leveraged in order to integrate airworthiness domain knowledge from standards like ED202 [5], ED203 [6]. The last part of the contribution is an algorithm for unveiling attack paths across the components model. The algorithm is based upon rules settling pre-conditions and post-conditions for an attack to progress up to the critical components failure. The subsequent failure propagation can be explored via typical methods (e.g., fault trees). The algorithm relies and takes advantage of the knowledge bases CAPEC and CWE already integrated. Overall, the method and framework aim to increase automation of safety-security co-engineering. The rest of the paper is structured as follows. The Section 2 provide a quick survey including methods for co-engineering and formal techniques useful for threats and robustness assessment. The Section 3 includes a description of the proposed co-engineering method. The Section 4 gives an overview of the techniques leveraged to support the method. The approach is finally demonstrated in Section 5 by analyzing a Flight Control System via a critical combined attack-failure path. Some conclusions and perspectives are finally presented in Section 6.

2 RELATED WORKS

As discussed in the previous section, the similarities between safety and security has lead to multiple attempts at reusing techniques from one field to another and integrating risks identified from both points of view as evidenced by Pietre Cambacedes et al. in [23]. The following section gives a brief overview of some of those techniques. More particularly, it focuses on tree-based techniques or tool-supported techniques but more complete comparative surveys like [24] are also available. Tree based formalisms, introduced with fault trees for safety[11] and attack trees for cybersecurity[26] are widely used and can also be combined together. For example the Extended Fault Trees proposed by Fovino et al.[7] describes a way to integrate cybersecurity and safety analysis in one common tree representation. Further extension have also been made to this formalism like the support for describing mitigations, as evidenced by the approaches described by Kordy et al.[13] and Roy et al.[25]. The main advantage of this kind of approach is that quantifying the risk can then rely on reusable assessments where attacks have an initial risk and mitigation reduce this risk. Boolean Driven Markov Processes (BDMP), introduced by Pietre Cambacedes et al.[22], are another kind of tree-based formalism for analysing threats. Originally developed for safety analysis, they also take into account causality relations between branches, certain events being able to enable other parts of the tree. This results in the production of metrics like a Mean Time To Security Failure, similar to the notion of MTTF in safety, but relies on the availability of similar metrics for individual attack steps, which are often not readily available. These tree-based methods can also quickly become difficult to use because, as the size of the system grows, so does the size and number of the trees. Other formalisms can be used to counter this problem, for example graph-based representations like the one described by Sommestad et al. in [28] which has been put in practice in the Cysemol language[27]. This approach derives risks from a description of the system architecture which also helps the systematic identification of scenarios. Cysemol also functions as an expert system to reduce the need of a dedicated cybersecurity expert for assessing threats, however because of the constant evolution of the field, this system still has to be regularly maintained. The STPA-Sec ontology introduced by Pereira et al.[21] is another recent development which aims at facilitating the construction of threat scenarios, however it does not provide a way to manage the expert's knowledge for assessing those scenarios. Our approach aims to complement those approaches by using architectural description and expert knowledge to guide the identification and evaluation of cybersecurity threats and integrating them in safety analyses.

3 PROPOSED SAFETY-SECURITY ANALYSIS METHOD

The proposed approach aims to be complementary to the related works described in Section 2. Its novelty comes from the integration of the following aspects. First, the method integrates standardized domain knowledge in the aim of increasing attacks coverage while still limiting complexity. Secondly, it is based upon standardized model-driven languages what eases modeling and facilitates models re-usability, exchange and results comparison. Last yet not the least, along with the integration of attack and failure paths, some means to automate their identification are given. The most salient features of the method are summarized in four phases and described in the following subsections.

3.1 Context and security perimeter

In this phase, the parameters to conduct the security and safety analyses are selected. Regarding the security part, confidentiality, integrity, authenticity, availability, and non-repudiation are among the criteria to select. Regarding the safety part, reliability, availability, tolerance to failure, and maintainability can be in the scope of the study. The scales and metrics for evaluating the risks associated should be accordingly included and harmonized. Thus, the functions for measuring and evaluating overall likelihoods, severity of impacts, risks and their acceptability should be specified. The analysis relies upon a view of the target system and its context which are captured via a model. Since a reliable security analysis often demands details of the architecture, the model should provide functional and system views including components, subcomponents, ports and connectors. In fact, the analysis of attack likelihood/difficulty requires details about the architecture technology as well as the identification of existing control measures and their protection features. The last yet quite important step of this phase is the definition of a security perimeter, *i.e.*, a border circumventing the target of study and the untrusted context including the hostile activity. Ports and connectors are thus concerned by the definition of the security perimeter which can be settled at aircraft, system, subsystem or component levels.

3.2 Knowledge bases instantiating

The instantiating of knowledge bases aims to facilitate reasoning and integration of security and safety aspects, specially for the non-savvy. To do so, the vulnerabilities of patterns are first instantiated

according to the components technology already specified in the model. Once done, the attack patterns can also be instantiated according to component technology and its vulnerabilities. This is a crucial nonetheless complex step which unfortunately can not be automated and demands expert intervention and validation. Even so, a guidance for the selection and instantiating of patterns can be provided. More concretely, attack and vulnerability patterns can be cross-related in advance by identifying the attributes that match. *E.g.* both attack and vulnerability patterns can include the attribute *OS Architecture*, if a component, and by extension its vulnerabilities, is associated with a specific OS, attack patterns targeting the same or no particular OS will be selected by default. The matched attributes can afterwards be re-used during the pattern instantiating by suggesting suitable candidates according to the specific architecture component. The selected patterns are finally associated to elements in the security perimeter (*e.g.*, exposed ports) thus defining attack vectors.

3.3 Attack scenarios

The attack scenarios are identified considering the set of components whose failure or misbehavior is a condition for the safety events to unfold. The referred components are assumed as potential targets of attacks and are called Threats Conditions. Successful attack scenarios shall finally lead to Threats Conditions having an impact on the overall system safety. Thus, the severity of attacks can be inherited from the impact assessment of safety events. Since different resources, knowledge, skills and also motivations (gain/investment trade-off) are necessary for an attack to be deployed, the attack scenarios should also be associated to the potential sources of threats and attacker profiles. To ease the exploration of paths from exposed ports in the security perimeter, towards the Threats Conditions components, an algorithm is deployed. The algorithm is based upon rules *Pre-Condition* $\Rightarrow$ *Post-Condition*. The *Pre-Conditions* are tuples with the syntax shown in (1).

$$(AttackPattern_i, \{VulnerabilityPattern_j\}, \neg\{Countermeasure_k\}) \quad (1)$$

The associations between attack and vulnerability patterns are inherited from previous phase [3]. The negated *Countermeasures* mean that their effectiveness to prevent the attack is at stake or is limited, *e.g.,* due to partial vulnerability coverage. The *Post-Conditions* take the syntax shown in (2).

$$(\{Vulnerability_m\}, \{PortType_n\}, \{AttackActionEffect_p\}) \quad (2)$$

The *Post-Conditions* contain the exploited *Vulnerabilities* which allow attack progression through specific *PortTypes*. The *Attack Action Effect* become new *Pre-Conditions* useful to re-apply the rule on components connected to compromised ports. The architecture exploration and unveiling of potential attack paths results from iterating over linked components. To ease model exploration, the Algorithm 1 for attack path discovering was implemented. It receives an architecture $Arch := (C, L)$ defined by a set of components C and links $L, L \subset P \times P$, over a set of ports P for inter-component communication. Each component is a tuple $C_i := (T_i, P_i, V_i, CM_i, F_i : I_i \mapsto O_i)$ where T_i specifies the component technology type, P_i its ports, V_i the set of vulnerabilities, CM_i the security countermeasures whereas $F_i : I_i \mapsto O_i$ is a function that maps input ports I_i into output ports O_i. $Att := (T_{at}, V_{at}, A_{at})$ is an attack pattern where T_{at} and V_{at} are respectively the technology and vulnerabilities known by the attacker and A_{at} is the set of allowed actions over the architecture. The algorithm 1 first searches for a match between attacker knowledge and the component technology and vulnerabilities. Once found, it then verifies that the attack is feasible by verifying that an exploitable vulnerability without corresponding countermeasure truly exists $(v_{i,j}, \neg cm_{i,k})$. In such case, the component C_i is thus compromised and added to the *AttPath*. To infer the attack progression, a rule *Pre-Condition* $\Rightarrow$ *Post-Condition* should be selected ensuring that the attack satisfy its pre-conditions. The output ports $O_{i,l}$ satisfying the post-conditions are then identified. The attack can further progress through associated connectors, ports $O_{m,n}$, and respective components C_m.

Data: $Arch := (C, L), L \subset P \times P, C_i \in C,$
$C_i := (T_i, P_i, V_i, CM_i, F_i : I_i \mapsto O_i), I_i \subset P_i, O_i \subset P_i,$
$Att := (T_{at}, V_{at}, A_{at})$
Result: Attack path sequence $AttPath$
Select component: $C_i \in C$, Target components:
$Target := [Ci]$, Explored components: $Exp := \emptyset$;
while $Target \neq \emptyset$ **do**
  $C_i := Target.next()$;
  $Target.remove(C_i)$;
  $Exp.add(C_i)$;
  if $T_i \cap T_{at} \neq \emptyset, V_i \cap V_{at} \neq \emptyset, Access(I_i) \in A_{at}$ **then**
    for all $v_{i,j} \in V_i \cap V_{at}, cm_{i,k} \in CM_i$;
    if $v_{i,j}, \neg cm_{i,k}$ **then**
      Select $Rule = PreCondition \Rightarrow PostCondition \Uparrow$
       $Att \cap PreCondition \neq \emptyset$;
      $AttPath.add(C_i)$;
      for all $O_{i,l} \in O_i, (O_{i,l}, P_{m,n}) \in L$;
      if $O_{i,l} \cap PostCondition \neq \emptyset, C_m \notin Exp$ **then**
        $Target.add(C_m)$;
      else
      end
    else
    end
  else
  end
end

Algorithm 1: Algorithm for attack path computation

3.4 Risks assessment and treatment

Once the attack paths are well identified, the assessment of its potential occurrence is performed. For that, likelihood, difficulty, and probability are among the qualitative and quantitative metrics proposed in the literature. Notice that the tuples (1) and (2) elicited in previous phase are a basis to propose metrics for measuring attack occurrence. The function for risks evaluation is applied to each target component (Threat Condition): *RiskLevel(AttackOccurrence, ImpactSeverity)*. The function for risks acceptability finally determine

[3] For now, the tuples *(AttackPattern, Vulnerability,* $\neg$ *Countermeasure)* should be defined, validated and maintained by an expert.

the unbearable risks for which treatment strategies need to be applied. The risk avoidance strategy can be applied by iterating on the Attack Scenarios by removing/modifying the Threat Sources and/or Attack Patterns. The risk reduction strategy can be applied either by eliciting more effective countermeasures or by removing vulnerabilities (component technology update/upgrade). This finally implies to update *Pre-Condition⇒Post-Condition* rules to reflect more constrained attacker options.

4 LEVERAGED TECHNIQUES FOR SAFETY-SECURITY CO-ENGINEERING

The following subsections describe the salient techniques leveraged to support the method specified in Section 3.

4.1 Module for CAPEC and CWE importing and integration

To support the identification of attack scenarios described in Section 3.3, the CAPEC[18] and CWE[17] knowledge bases, respectively containing attack patterns and software weaknesses, were leveraged. Those knowledge bases are structured in a way that facilitates the specification of potential vulnerabilities associated with an asset and define how to enchain attacks on specific assets to build a scenario. While the weaknesses in CWE have a higher level of abstraction than vulnerabilities, they are still useful in our approach because they describe in detail exploitable patterns without being tied to specific implementations as is the case in CVE [19], a concrete vulnerability repository which allows to take into consideration potentially unknown vulnerabilities. CAPEC and CWE are linked together as attack patterns link to relevant weaknesses and vice-versa. Additionally, both knowledge bases are structured in a way that allows to identify the technology characteristics of potential targets. This structure allows us to provide the user with a list of potential weaknesses and attacks potentially relevant for the system under analysis. Figure 1 shows the specific CAPEC and CWE attributes used to cross-relate attack patterns, vulnerabilities and components technology.

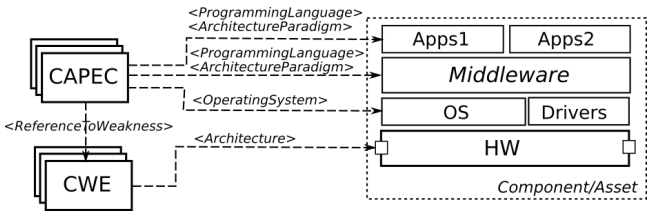


Figure 1: Specific CAPEC and CWE attributes cross-related to the component architecture

Since CAPEC also contains information about attacks' precedence, it is plausible to build more complex attack scenarios by suggesting attacks which may follow or precede another. CWE also provides information on potential mitigations or detection methods which can then be taken into account in the design as a counter measure against the weakness. Both bases also have evaluations for likelihood and impact which can be used when performing risks evaluation. To use these knowledge bases inside the tool, a transformation was implemented from the XML files made available by MITRE [18], [17] which results in the listing shown in Figure 2. This approach allows to support the user in a semi-automated way to handle large systems with more ease. It also leads to a more reproducible analysis, relying on standardised formats and reusable knowledge.

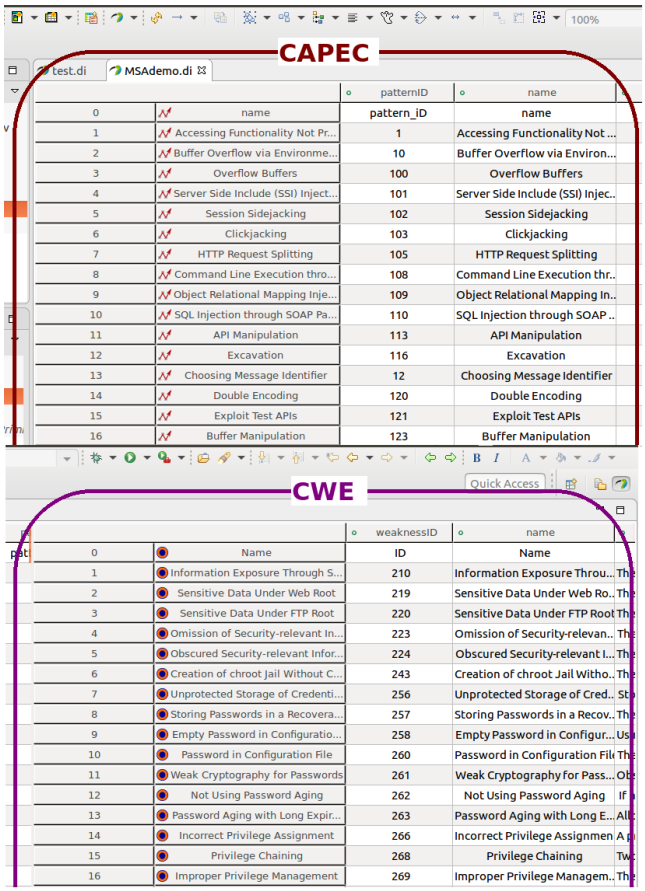


Figure 2: Overview of imported knowledge bases

4.2 UML profile for safety and security engineering

The framework for safety-security co-engineering is inspired by, and based upon Model Driven Engineering (MDE) techniques and built on top of Papyrus [3]. One of the most salient features of Papyrus is the support for defining domain specific languages via UML [9] or SysML [8] profile implementations. A profile is indeed an implementation of a meta-model which is adequate to capture and stereotype fundamental concepts, features, dependencies/relationships associated to a specific domain or concern. The developed profile supports modeling of domain specific knowledge related to aircraft technology and in particular components, communication channels, software architecture, OS, etc. An excerpt of the profile diagram is shown in Figure 3.

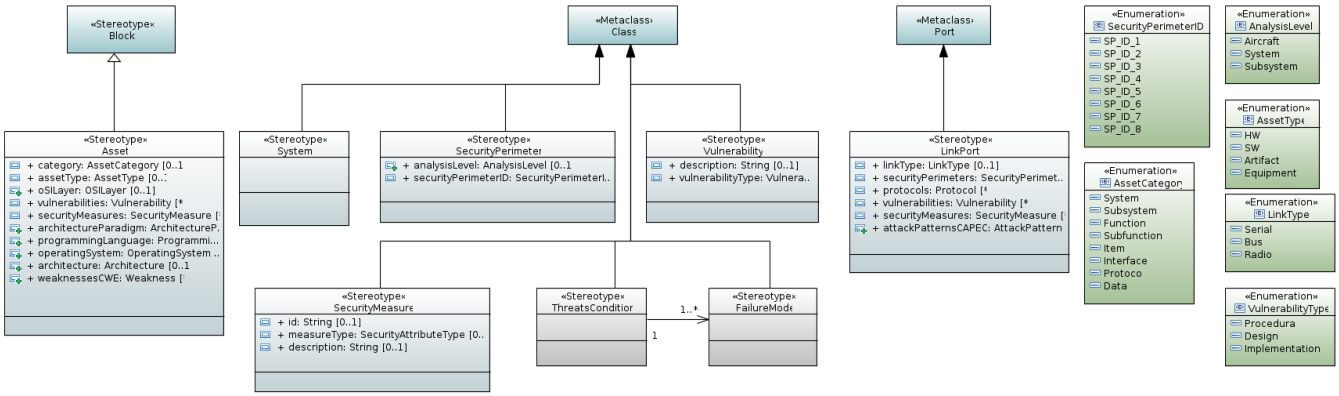


Figure 3: Excerpt of the profile for modeling and analysis of security and safety

The profile is also a mean to deploy the interfaces necessary to import external knowledge sources from CAPEC [18] and CWE [17]. UML has been selected as a basis language to implement our profile in order to keep a good balance between its genericity and its capacity to capture the most salient notions and features of CAPEC and CWE. More concretely, the UML profile facilitates the assignment of weaknesses to components/assets and attack patterns to the ports (as depicted in Figure 1) thus settling potential preconditions for attack continuation troughout its connections. The attack progression inside the system towards its goal thus depends upon the components/assets, their interfaces and the weaknessess in both. These elements are structured to support the identification and exploration of attack-failure paths as shown in Figure 4.

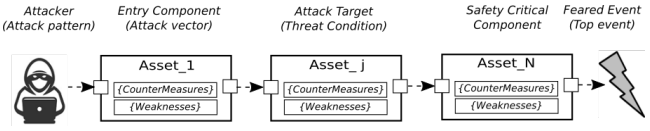


Figure 4: Structure of an attack-failure path model

Along with the referred modeling capabilities, the profile is enriched with elements useful to semi-automate analyses. More specifically, the elements necessary for model analysis like the system context (*e.g.*,security perimeter), the parameters for safety-security criteria, the metrics for the assessment of safety events occurrence and the severity of impact. Overall, the identification of attack scenarios, the assessment and treatment of risks are also integrated and designed via the profile.

4.3 Framework for safety-security design and analysis

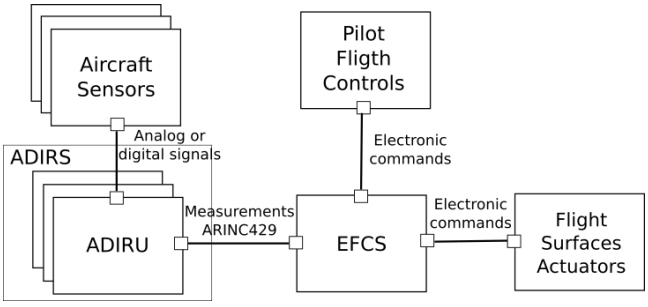
The design and analysis framework covers all the phases of the method introduced in Section 3 and is built on top of Papyrus [3], an Eclipse module for Model Driven Engineering supporting UML2.0 and which is available under the EPL license. The main modeling and analysis features are summarized in the following paragraphs. To support modeling, the user can first define the parameters for the analysis as described in Subsection 3.1. Some default templates are available and can be imported so as to provide guidance in this task. Dedicated diagrams are implemented to separately model vulnerabilities and security countermeasures, as well as to visualize the imported knowledge bases (see Figure 2). These model elements are to be reused during the definition of the target system which is modeled in two steps. In the first step, a functional view is designed including its decomposition. In the second step, an internal view of the architecture is given thus detailing components, subcomponents, ports and connectors. The components view is amenable for refinements and several architecture layers can be created what facilitates the introduction of security perimeters at different levels. The traceability of model elements ensures the consistency in case of cross-layer analyses. As part of the analyses activities supported, the exploration of Attack Scenarios is based upon the model of Threats (Sources and Attack Patterns) and upon the identification of targeted components (Threats Conditions). The attack-failure paths are obtained by an implementation of the algorithm described in Subsection 3.3. Finally, the functions for assessing attack-failure paths occurrence (likelihood, difficulty, probability) and the ones for evaluation and treatment of risks offer method support as described in Subsection 3.4. An overview of the modeling frontend is shown in Figure 6.

5 CASE STUDY: ANALYSIS OF A FLIGHT CONTROL SYSTEM

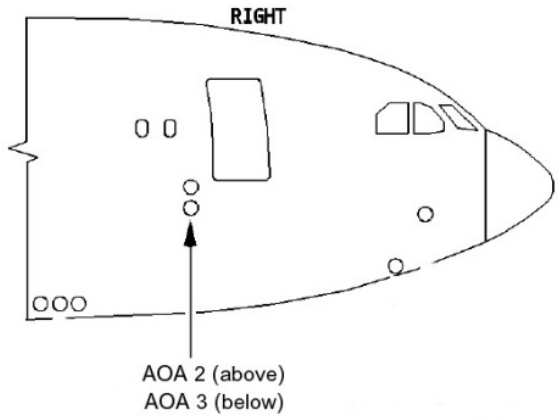
The method and leveraged techniques respectively described in Sections 3 and 4, were applied to analyze an industry-size case study covering all electronic systems of an aircraft (4MB model size). The case study is inspired in the incident report [2] involving a Flight Control subsystem leading to an unexpected aircraft diving and temporary loss of flight control. In the incident, a specific input signal pattern, sent by a sensor of the aircraft, triggered a flight control system misbehaviour leading to the wrong activation of compensation mechanisms to avoid a nonexistent stall condition. Due to lack of space, the approach is shown on the specific subsystem concerned by the incident.

5.1 Target system and safety critical event

The Figure 5a represents the system considered in this study. The two main elements of this system are the Air Data and Inertial Reference System (ADIRS) and the Electrical Flight Control System (EFCS).



(a) Flight Control System Architecture



(b) Location of the Angle of Attack sensors

Figure 5: Internal and external views of the aircraft architecture. Figure (b) is borrowed from the ATSB report [2]

The ADIRS is charged with collecting data from the different sensors in the aircraft and then computing different flight parameters. The parameters are sent to the EFCS where they are compared to inputs coming from the crew flight controls or the autopilot to determine the actual commands to be sent to the flight surfaces actuators. The failure in this event is a misbehaviour of the algorithm charged with filtering erroneous values in the angle of attack (AOA) measurements. Confronted with a particular pattern in the measurements for this parameter,the output of the algorithm was set to an erroneous value for several seconds. This in turn led the EFCS to activate compensation mechanisms as the AOA value was above their activation threshold. The mechanisms are in place to avoid a loss of control of the aircraft due to being unable to generate enough lift (stall condition). This resulted in actions on the elevators which led to a sudden dive of the aircraft (stall recovering). The unnecessary dive caused injuries among the passengers of the aircraft while the crew tried to take control back which was only restored after the erroneous input was finally discarded by the flight control system.

5.2 FCS Threat scenario

The incident described in the previous Subsection 5.2 clearly demonstrates a potential impact on safety. To intertwine the safety assessment with the security analysis, it should be inferred whether an attacker can trigger the incident intentionally and how. The attacker's goal can be take over the control of, or at least be able to have enough influence, on the measurement of the AOA so as to recreate the pattern leading to the -wrong- aircraft dive for stall recovering. By analyzing the Figure 5a, we go from the source of malfunctioning -the EFCS- and trace back to the origin of the measurement. The following potential targets arise: the ARINC 429 bus conveying the measurements from the ADIRS towards the EFCS, the ADIRS, the connection between the AOA sensor and the ADIRS, and the sensor itself. As described in the incident report [2], the connection between the AOA sensor and the ADIRS is a raw wire conveying an analogical value which is particularly vulnerable since neither the integrity nor the authenticity of the data are protected. Figure 5b shows the location of the AOA sensors what suggests that a significant amount of wiring is disposed over the aircraft fuselage towards the ADIRS in the cockpit. According to this hypothesis, an attacker can use a directed electromagnetic impulsion device to recreate the incriminated pattern on the wiring.

5.3 FCS Threat scenario assessment and evaluation

To assess the attack scenario, the tool filters a list of attack and vulnerability patterns according to the technology of the component selected as attack entry. The CAPEC attack pattern shown in Table 1 is selected due to its name (*Fault Injection*) and after inspecting its definition. A vulnerability that corresponds to the attack is also selected from CWE (see Table 2) and both patterns are instantiated.

Table 1: Attack pattern instantiated from CAPEC

Attribute	Value
Pattern ID	624
Name	Fault Injection
Description summary	The adversary uses disruptive signals or events (e.g. electromagnetic pulses, laser pulses, clock glitches, etc.) to cause faulty behavior in electronic devices.
Attacker prerequisites	Physical access to the system:: The adversary must be cognizant of where fault injection vulnerabilities exist in the system in order to leverage them for exploitation.
Resources required	The relevant sensors and tools to detect and analyze fault/side-channel data from a system. A tool capable of injecting fault/side-channel data into a system or application.
Criticality	Typical severity: high. Typical likelihood: low.
Solutions and mitigations	Implement robust physical security countermeasures and monitoring.

Once instantiated, the patterns are leveraged in order to define the rules to perform a semi-automatic attack path exploration. To do so, the tuples *Pre-Conditions* (1) and *Post-Conditions* (2) defined in

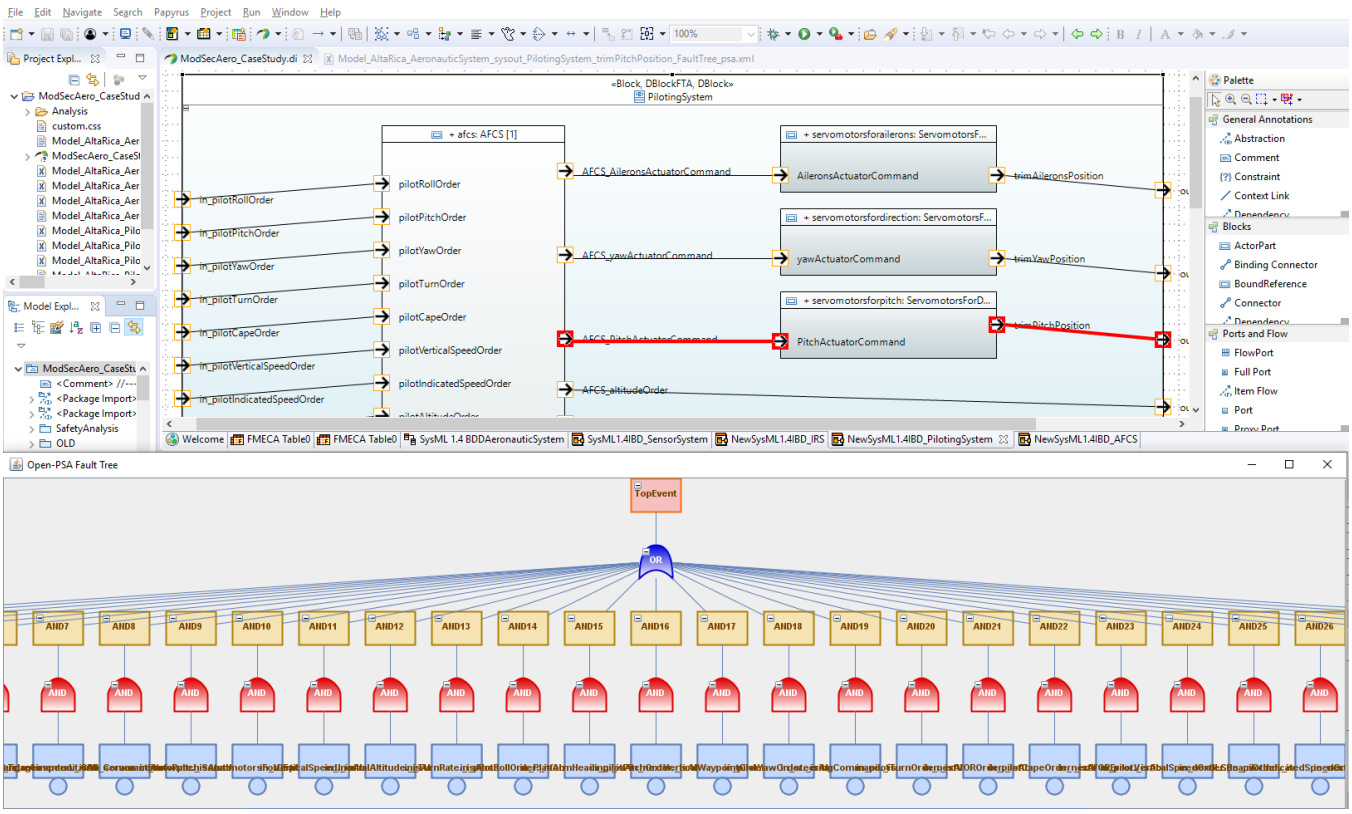


Figure 6: Overview of the attack-failure path in the FCS threat scenario and generated fault tree

Table 2: Vulnerability pattern instantiated from CWE

Attribute	Value
Weakness ID	790
Name	Improper Filtering of Special Elements.
Description	The software receives data from an upstream component, but does not filter or incorrectly filters special elements before sending it to a downstream component.
Modes of introduction	:::PHASE:Implementation:DESCRIPTION:::
Common consequences	::SCOPE:Integrity:TECHNICAL IMPACT:Unexpected State::

Subsection 3.3 are taken as reference. A definition of *Pre-Conditions* is obtained by extracting the contents of the Attack and Vulnerability patterns as follows:

$$(FaultInjection_{ID624}, ImproperFiltering_{ID790}, \neg\{PhysicalProtection_{ID624}\}) \quad (3)$$

To define the *Post-Condition* of the rule, we focus on the *Common consequences* of the Vulnerability in Table 2. We can easily infer that any impact on data or signals integrity can finally lead to an unexpected component state. By cross-relating this fact with the *Description* of the Attack in Table 1, the referred consequence can be expected at every component and port where improper or no filtering of signals is observed. This *Post-Condition* is formalized as follows:

$$(ImproperFiltering_{ID790}, AllPortType, UnexpectedState_{ID624}) \quad (4)$$

Once defined, the rules are entered to the algorithm 1 and applied to explore the target system depicted in Figure 5a. The outcomes show that output ports of the ADIRS are impacted and consequently the attack is propagated via the ARINC 429 towards the EFCS. Since the *Pre-Conditions* (3) are also satisfied at EFCS level, consequently, the *Post-Conditions* (4) are again held implying that the EFCS output ports degrade to unexpected state. The attack finally reaches the target component, *i.e.*, the Flight Surfaces Actuators thus leading to the critical safety event described in Subsection 5.1. An overview of the attack path and respective fault tree are shown in Figure 6.

To conduct the risk assessment, the *Criticality* levels in Table 1 are first considered (*Typical severity: high. Typical likelihood: low.*). However, the criticality of the incident according to ARP-4761 [1] is Catastrophic (Failure may cause a crash), so the severity is increased to *Very high*[4]. The resulting risk becomes unbearable and it needs to be reduced, e.g., by deploying electromagnetic protections or by redesign of filtering functions in the Flight Control System components. Once deployed, a method iteration is conducted to evaluate the residual risk.

[4]It is assumed that the safety expert conducts such assessment or intervenes afterwards to confirm or disapprove it

6 CONCLUSIONS AND PERSPECTIVES

This paper presents a method and framework to conduct a security analysis guided by safety events. The method proposes to place in a sequence security and safety analyses and gives means to harmonize fundamental concepts, metrics and techniques which are typically but separately applied in risks analyses. The approach mainly addresses the identification and evaluation of combined attack-failure paths across an architecture model. To ease modeling and analysis, the framework integrates open knowledge bases containing patterns of attacks and vulnerabilities and allows to instantiate them according to the architecture technology. Some model driven engineering techniques were leveraged in order to ease the integration of knowledge related to the aeronautics and airworthiness domains. An algorithm was also defined to automatically explore the system architecture and to unveil attack-failure paths. The algorithm is based upon a set of predicate rules which settle pre-conditions for the attack actions to occur, and post-conditions declaring the respective impact on components and ports. The overall approach was demonstrated in an industry-size case study in the aeronautics domain which provides a basis for scaling. In particular, models and analysis outcomes are amenable for re-usage, exchange and comparison. As perspectives, probabilistic methods can be applied to quantitatively evaluate the trade-off vulnerability vs. countermeasures. The attack-failure path algorithm and rules can be extended and consolidated by analyzing further case studies from other application domains. The formalization of architecture models and rules via a language and semantics can help to soundly verify properties on attack-failure paths.

ACKNOWLEDGMENTS

Part of the work presented in this paper was conducted in the scope of the project ModSécAéro partially funded by thre French RAPID programme [4].

REFERENCES

[1] SAE ARP4761. 1996. Guidelines and Methods for Conducting the Safety Assessment Process on Airborne Systems and Equipments. *USA: The Engineering Society for Advancing Mobility Land Sea Air and Space* (1996).
[2] Australian Transport Safety Bureau ATSB. 2011. *Aviation Occurrence Investigation. ATSB Transport Safety Report.* Technical Report. https://www.atsb.gov.au/media/3532398/ao2008070.pdf
[3] CEA-LIST. 2020. Papyrus designer. https://www.eclipse.org/papyrus/. Online; accessed 25 July 2020.
[4] Agence de l'Innovation de Défense. 2020. *RAPID (régime d'appui à l'innovation duale).* https://www.defense.gouv.fr/aid/deposez-vos-projets/subventions/rapid
[5] RTCA DO. 2014. 326A/EUROCAE ED-202A,"Airworthiness Security Process Specification.".
[6] EUROCAE. 2018. ED-202A,"Airworthiness Security Methods and Considerations.".
[7] Igor Nai Fovino, Marcelo Masera, and Alessio De Cian. 2009. Integrating cyber attacks within fault trees. *Reliability Engineering & System Safety* 94, 9 (2009), 1394 – 1402. ESREL 2007, the 18th European Safety and Reliability Conference.
[8] Object Management Group. 2015. *Systems Modeling Language V1.4.* https://www.omg.org/spec/SysML/1.4/PDF
[9] Object Management Group. 2017. *Unified Modeling Language V2.5.1.* https://www.omg.org/spec/UML/2.5.1/PDF
[10] A. Hameed and A. Alomary. 2019. Security Issues in IoT: A Survey. In *2019 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT).* 1–5.
[11] AF Hixenbaugh. 1968. *Fault tree for safety.* Technical Report. Boeing Co, Seattle WA, Support Systems Engineering.
[12] J. Kang, D. Lin, E. Bertino, and O. Tonguz. 2019. From Autonomous Vehicles to Vehicular Clouds: Challenges of Management, Security and Dependability. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS).* 1730–1741.
[13] B. Kordy, S. Mauw, S. Radomirovic, and P. Schweitzer. 2012. Attack-defense trees. *Journal of Logic and Computation* 24, 1 (June 2012), 55–87. https://doi.org/10.1093/logcom/exs029
[14] M. Leccadito, T. Bakker, R. Klenke, and C. Elks. 2018. A survey on securing UAS cyber physical systems. *IEEE Aerospace and Electronic Systems Magazine* 33, 10 (2018), 22–32.
[15] M. Śliwiński, E. Piesik, and J. Piesik. 2018. Integrated functional safety and cyber security analysis. *IFAC-PapersOnLine* 51, 24 (2018), 1263 – 1270. 10th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes SAFEPROCESS 2018.
[16] H. Martin, Z. Ma, Ch. Schmittner, B. Winkler, M. Krammer, D. Schneider, T. Amorim, G. Macher, and Ch. Kreiner. 2020. Combined Automotive Safety and Security Pattern Engineering Approach. *Reliability Engineering & System Safety* (2020), 106773.
[17] MITRE. 2006. *Common weakness enumeration.* https://cwe.mitre.org
[18] MITRE. 2008. *Common attack pattern enumeration and classification.* https://capec.mitre.org/
[19] MITRE. 2020. *Common vulnerabilities and exposures.* https://cve.mitre.org/
[20] Gabriel Pedroza. 2019. Towards Safety and Security Co-engineering: Challenging Aspects for a Consistent Intertwining. In *Security and Safety Interplay of Intelligent Software Systems.* Springer International Publishing, Barcelona, Spain, 3–16.
[21] Daniel Patrick Pereira, Celso Hirata, and Simin Nadjm-Tehrani. 2019. A STAMP-based ontology approach to support safety and security analyses. *Journal of Information Security and Applications* 47 (2019), 302–319.
[22] Ludovic Pièrre-Cambacédès and Marc Bouissou. 2010. Modeling safety and security interdependencies with BDMP (Boolean logic Driven Markov Processes). In *2010 IEEE International Conference on Systems, Man and Cybernetics.* IEEE, 2852–2861.
[23] Ludovic Pièrre-Cambacédès and Marc Bouissou. 2013. Cross-fertilization between safety and security engineering. *Reliability Engineering & System Safety* 110 (2013), 110–126.
[24] Christian Raspotnig and Andreas Opdahl. 2013. Comparing risk identification techniques for safety and security requirements. *Journal of Systems and Software* 86, 4 (2013), 1124–1151.
[25] Arpan Roy, Dong Seong Kim, and Kishor S Trivedi. 2010. Cyber security analysis using attack countermeasure trees. In *Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research.* 1–4.
[26] Bruce Schneier. 1999. Attack trees. *Dr. Dobb's journal* 24, 12 (1999), 21–29.
[27] Teodor Sommestad, Mathias Ekstedt, and Hannes Holm. 2012. The cyber security modeling language: A tool for assessing the vulnerability of enterprise system architectures. *IEEE Systems Journal* 7, 3 (2012), 363–373.
[28] Teodor Sommestad, Mathias Ekstedt, and Pontus Johnson. 2009. Cyber security risks assessment with bayesian defense graphs and architectural models. In *2009 42nd Hawaii International Conference on System Sciences.* IEEE, 1–10.