On Computer-Aided Techniques for Supporting Safety and Security Co-Engineering

Antoaneta Kondeva
fortiss GmbH
Munich, Germany
kondeva@fortiss.org

Carmen Carlan
fortiss GmbH
Munich, Germany
carlan@fortiss.org

Harald Ruess
fortiss GmbH
Munich, Germany
ruess@fortiss.org

Vivek Nigam
fortiss GmbH
Munich, Germany
nigam@fortiss.org

***Abstract*—With the increasing system interconnectivity, cyber-attacks on safety-critical systems can lead to catastrophic events. This calls for a better safety and security integration. Indeed, a safety assessment contains security relevant information, such as, key safety hazards, that shall not be triggered by cyber-attacks. Guidelines, such as, SAE J3061 and ED202A, already recommend to exchange information gathered by safety and security engineers during different phases of development. However, these guidelines do not specify exactly how and which information shall be exchanged. We propose a methodology for enabling computer aided techniques for extracting security relevant information from safety analysis. In particular, we propose techniques for automatically constructing Attack Trees from safety artefacts such as fault trees, hazard analysis and safety patterns. Lastly, we illustrate these techniques on an Industry 4.0 application.**

***Index Terms*—safety and security co-engineering, ED202A, attack trees, FTA**

I. Introduction

The past years have witnessed a rapid increase on system inteconnectivity, leading to new exciting services and business models, *e.g.*, Industry 4.0 and autonomous cars. This technological revolution, however, leads to new challenges for safety and security. Cyber-attacks can cause catastrophic events by remotely targeting safety-critical systems [6]–[9]. For example, cyber-attacks exploiting a connection to an autonomous vehicle may remotely disable safety features, such as airbags, thus placing passengers in danger [8].

This calls for a better integration between safety and security. Indeed, standards and guidelines, such as, ED202A [1], for the avionics industry, and SAE J3061 [3], for the automotive industry, already takes steps towards this integration. In particular, they specify interaction points between the two processes, namely when information gathered by safety engineers shall be made available to security engineers and vice versa.

More concretely, ED202A [1] recommends that all safety hazards identified in the *Hazard Analysis Phase* (e.g. Functional Hazard Analysis (FHA)) shall be used as input to both the *Preliminary Safety System Assessment* (PSSA) and the *Preliminary Security System Assessment* (PSA). The safety assessment shall include analysis for each identified hazard evaluating the risk of system failures or development and software errors to trigger this hazard, while the security shall perform threat analysis in order to assess the risk of an attacker to trigger this hazard. For example, a safety hazard could be shut down a factory element due to an erroneous message sent by the controller. This erroneous message can be caused by a software defect, or by an attacker that forces the controller to send such erroneous message.

However, as the main purpose of the guidelines is to inform, they do not propose concrete techniques on how and which information is to be exchanged. This paper is a step towards filling this gap. We present a methodology supporting the safety and security co-engineering process conform to ARP4754 [5], ED202A. In particular, we propose computer-aided methods for extracting security relevant information from safety analysis carried out during the FHA and PSSA.

For example, *Fault Tree Analysis* (FTAs) contains the basic events causing a hazard. This information can be used to refine the Preliminary System Security Assessment, by assessing what is the risk of attackers triggering these basic events. Moreover, the output of safety analysis are safety requirements on the system, such as, the use of specific safety (architectural) patterns, *e.g.*, voters, to control some particular hazard. Such safety patterns can be attacked [25] to trigger hazards.

We start in Section II revisiting some basic methods for safety and security assessments, and in Section III describe how these techniques are used in the development process. Section IV describes a methodology for extracting security relevant information from safety analysis. The application of this methodology on a Industry 4.0 example is described in Section V. We describe related work in Section VI and conclude in Section VII by pointing out to future work.

II. Basic Safety and Security Analysis Techniques

This section briefly reviews the main techniques used during the safety and security co-engineering process.

A. Safety Techniques

a) Hazard Analysis: Hazard Analysis such as HARA [2], FHA [4], is a method for systematic identification of hazards at early system development stages. Hazard is defined by [21] as:

A system state or state of conditions that, together with a particular set of worst-case environmental conditions, will lead to an accident (loss).

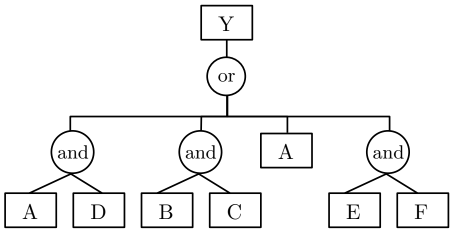


Fig. 1. FTA Example.

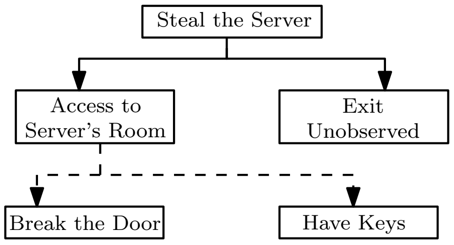


Fig. 2. Attack Tree Example. Full lines denote "And" connections and dashed "Or" connections.

Consequently, system functions are examined to identify their functional failures such as malfunctioning or loss of function. Subsequently, the severity of functional failures under a particular operational condition are assessed and classified. The outcome is a hazard list, that is functional failures evaluated with a safety effect.

For example, consider the aircraft function *decelerate aircraft on the ground*. The functional failure *unexpected loss of deceleration* during operational condition *landing and certain runway length* could have a catastrophic effect since in this case the crew could be unable to decelerate the aircraft and this can result in a high speed overrun, which in turn could harm persons. The same failure would be classified with a lower severity under operational condition *during taxing*, similarly when using other runway length.

b) Fault Tree Analysis (FTA): Fault Tree Analysis is a deductive failure analysis aiming at identifying and modeling the causes of an event leading to a hazard. FTA is based on a graphical model, fault tree, were the root (the top event) is decomposed using boolean algebra into sub-events representing the causes for the top event. FTA can be qualitative and quantitative. Outcome of the qualitative analysis are the *cut sets in a fault tree*, which is a set of sets of events. Each cut set represents the basic events whose (simultaneous) occurrence ensures that the top event occurs. The minimum cut sets can be calculated, by using the inclusion relation, *i.e.*, for two cut-sets $\mathcal{S}_1, \mathcal{S}_2$, $\mathcal{S}_1 \preceq \mathcal{S}_2$ if and only if $\mathcal{S}_1 \subseteq \mathcal{S}_2$.

Consider, for example, the FTA depicted in Figure 1. The undesired event is Y placed at the root of the tree. A safety engineer is interested on the *cut sets of an FTA*, that is, the collections of events, normally component faults, that lead to the undesired event. For this FTA example, the cut sets are {A,D}, {B,C}, {A}, {E,F}, as any of these combinations lead to the event Y. The minimum cut set for the given example is {B,C}, {A}, {E,F}.

Minimal cut sets are used to understand the structural vulnerability of a system. They can also be used to discover single point failures. Safety engineers prioritize minimal cut-sets to be controlled by carrying out a criticality analysis. Higher critical elements are prioritized. For more detailed information about FTA and its application we refer the reader to [22], [4]

c) Safety Requirements and Safety Patterns: The output of hazard and FTA analysis are safety requirements on the artefacts ensuring that top events are not likely to occur, or there are no single point of failure, etc. Some safety patterns for embedded systems is described in [25].

For example, the M out of N Voter (MooN) pattern is a safety mechanism used to reduce the probability of triggering an unintended signal. It combines N independent signals, and it only triggers an event if M signals support the triggering of the event. For another example, the Watchdog pattern is used to ensure the liveness of the system.

Finally, the paper [25] also carries out a security analysis of a number of safety patterns. While we are inspired by [25], there are some differences. Firstly, we use Attack Trees (described in Section II-B) instead of a using the Goal Structure Notation formalism. The use of Attack Trees allows the specification of threat scenarios by using information from FTA and Hazard Analysis, leading to more precise threat scenarios.

d) Guiding Words: A predefined set of guiding words such as general failure modes, $\mathcal{GW}$, specify domain knowledge and serve as technique for hazard and fault identification during safety assessment (FHA, FTA).

Guiding words differ on the development level and concern different topics. For example, on early development phase during the FHA, general functional failures are *loss of function* or *erroneous function* [4]. These general failure conditions can be further specified as adding more detail such as *function react too late*, *too early*.

B. Attack Trees (ATs)

First proposed by Schneier [29], Attack Trees (ATs) and its extensions, Attack Defense Trees [11], [18], are among the main security methods for carrying out threat analysis [30]. An AT specifies how an attacker could pose a threat to a system. It is analogous to FTA but, instead of arguing about the safety of a system, an AT breaks down the possibilities of how an attack could be carried out.

Consider, for example, the AT depicted in Figure 2, taken from [11]. It describes how an attacker can successfully steal a server. He needs to have access to the server's room and be able to exit the building without being noticed. Moreover, in order to access to the server's room, he has to break the door or obtain the keys.

III. The V Development Process

This section describes the safety and security co-engineering process recommended by ED202A. The purpose is to point out which activities and techniques are applied at which stage of

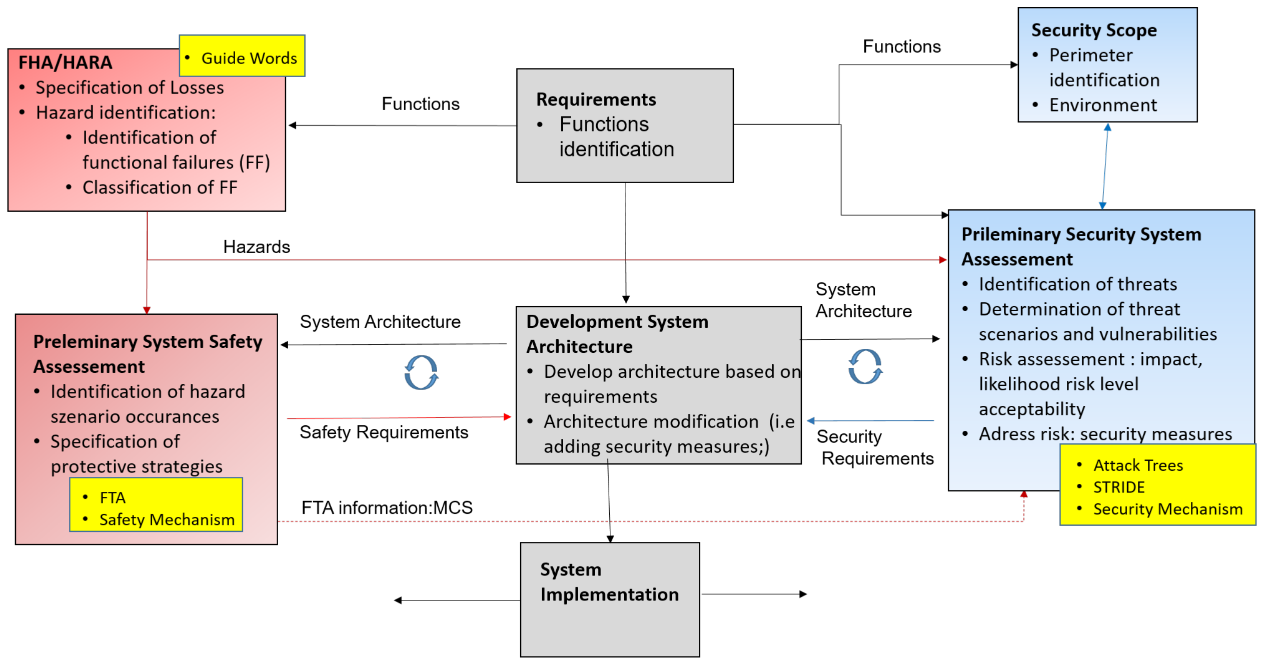

FHA/HARA
Specification of Losses
Hazard identification:
Identification of functional failures (FF)
Classification of FF
Guide Words
Functions
Requirements
Functions identification
Functions
Security Scope
Perimeter identification
Environment
Hazards
Prilemінary Security System Assessement
Identification of threats
Determination of threat scenarios and vulnerabilities
Risk assessement : impact, likelihood risk level acceptability
Adress risk: security measures
Attack Trees
STRIDE
Security Mechanism
System Architecture
Preleminary System Safety Assessement
Identification of hazard szenario occurances
Specification of protective strategies
FTA
Safety Mechanism
Development System Architecture
Develop architecture based on requirements
Architecture modification (i.e adding security measures;)
Safety Requirements
System Architecture
Security Requirements
FTA information:MCS
System Implementation

Fig. 3. Interaction between Safety, Security and Development Processes according to ARP4754 and ED202A. The connection with the dashed line is ours.

development. We emphasize the interrelations between the sub-processes, pointing out the concrete informational exchange. Further, we extend the process by adding additional artefacts on the interaction points from safety to security (dashed line).

The red boxes in Figure 3 define the safety process and the artefacts coming from the safety process. The black boxes in Figure 3 represent the system development process and the activities during the process. The blue boxes Figure 3 represent the security assessment process and the interrelation points to the safety and the development processes. We detail the safety and security sub-processes and then discuss which information can be exchanged, when and how.

a) Safety Sub-Process: The safety process starts with the **Functional Hazard Analisys (FHA)**. The outcome of FHA is a list of hazards which are provided to the Preliminary System Safety Assessment (PSSA) and to the Preliminary Security Assessment activities.

The **PSSA** aims at identifying the causes of the identified hazards and to provide protective solutions to control them. The first goal is accomplished usually by applying FTA, where the system architecture provided by the system development is assessed by building scenarios on how particular failures of the system components can lead to the undesired event, the hazard. Note that, the focus of the analysis is on the causes coming from inside the system such as system failure or error. The safety engineer identifies safety mechanisms such as diversity, voters, redundancy; and applies these by specifying the safety requirements for these design decisions and providing them to the development process. Consequently, the interrelation between PSSA and the System Architecture Development is highly iterative process.

b) The Security Sub-Process: Before starting with the risk assessment, the security scope, namely, what has to be protected against unauthorised interactions, shall be identified. Here, the security perimeter and the security environment are determined. The security perimeter defines the assets that are under system control and sets the boundary to the security environment, elements under external control. Security perimeter descriptions entail the assets that shall be protected from attacks and the entry points to these assets.

The **Preliminary Security Assessment (PSA)** aims at evaluating the security risk of a system exposed on unauthorized interaction. This is accomplished by

1) identifying the threat conditions;
2) developing the threat scenarios leading to those conditions;
3) addressing the risk by developing security measures to mitigate the threats.

Input for the process are hazards identified during the FHA, which serve as main part of the threat condition list, and system architecture. For each threat condition with high severity and by considering the system architecture, the security engineer develops threat scenarios leading to the threat conditions. Applicable techniques for scenario modelling are attack trees. Since it is impossible to handle all identified scenarios, these are assessed against their likelihood and severity, to determine whether risk is acceptable or not and measure the effort. This is part of the risk estimation activity. Then, for all unacceptable risks, counter measures are developed to make them acceptable. The outcome of the PSA are security requirements provided to the development process.

The system is implemented according to the decided system architeture and the verification process starts.

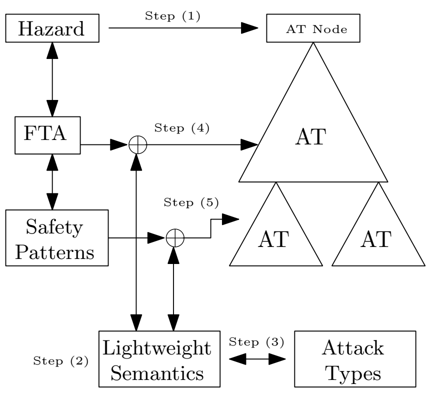


Fig. 4. Methodology for translating safety models to Attack Trees

c) Discussion: The guidelines we investigated only explicitly recommend exchanging the information about Hazards from the Safety to the Security sub-processes.[1] As illustrated in Figure 3, there some more possible candidates for information exchange:

- *FTA Minimal Cut-Sets* can be used by the security sub-process to refine the threat assessments. In particular, the minimal cut-sets specify ways in which an attacker can generate an event leading to a hazard;
- *Safety Requirements* with mechanisms to control hazards, *e.g.*, safety voters, can be also be used to refine threat scenarios. For example, if a voter is used to control a hazard, then attacking it may also lead to the hazard.
- *Security Requirements* with mitigation mechanisms, *e.g.*, the use of encryption, can be used by the safety sub-process to evaluate the risk of hazards due to late messages.

Notice that the latter two are *implicitly* implied by iteration with the *Development System Architecture*, shown in Figure 3.

As we demonstrate in Section IV, we can apply computer-aided methods to exchange this information. Our focus will be in the information exchange from safety to security. We leave the direction from security to safety to future work.

IV. Automated Threat Analysis Generation from Safety

This Section proposes a methodology for extracting security relevant information from safety analysis using computer-aided methods. We then instantiate this methodology, proposing translations from safety models to ATs.

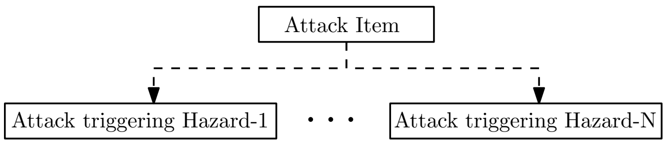
A. Methodology

The overall methodology is depicted in Figure 4 and consists of the steps described below. Following this methodology, one is able to use computer aided methods for extracting security analysis in the form of ATs from safety analysis.

1) **Hazard to Attack Tree** – For each hazard identified in the Hazard analysis, we generate the root node of the attack tree. This follows the guidelines discussed above. The idea is that each attack tree will be expanded using the information collected in the safety analysis.
2) **Lightweight Semantics** – In order to automate the extraction of information from safety analysis, however, we associate to each safety model, *e.g.*, events in FTA, a lightweight semantics. Inspired by the work of [15], this semantics consists of simple annotations, normally already used by safety engineers, *e.g.*, Guide Words, that specify the nature of events involved in the safety analysis and of the types of safety patterns deployed;
3) **Relating Guide Words to Attacks** – We associated with each guide word used in the lightweight semantics the possible attacks that could lead to it. Attacks can be expressed by using the STRIDE notation, for example.
4) **FTA to Attack Tree** – Based on the lightweight semantics associated to the events, an FTA analysis for triggering a hazards is translated to an AT, by reasoning over its minimal cut sets. The AT obtained from the hazards is then expanded using this AT;
5) **Safety Patterns to Attack Trees** – We then further expand the trees by reasoning over the safety patterns, *i.e.*, architectural patterns recommended in security requirements. In particular, new sub-attack trees are constructed enumerating ways for which attackers can trigger a hazard by attacking a safety pattern.

We instantiate each one of the steps in the following.

1) Hazard to Attack Tree: Each hazard identified for an item corresponds to a sub-attack tree for an item:



I.e, an item can be attacked by triggering one of its hazards.

2) Lightweight Semantics: We annotate safety analysis, events and safety patterns, with *lightweight* semantics. In particular, each event, ev, in the Hazard and FTA analysis is associated (manually) to a pair, $\langle \mathsf{cp}, \mathsf{gw} \rangle$, where

- cp is the model element, *e.g.*, component to which the event is associated with;
- gw is a Guide Word specifying the type of fault associated with the event.

In this paper, we will use the following Guide Words to illustrate our translations.

- **Loss of Function:** A hazard occurring when a function is lost. For example, production is stopped if the controller of a factory element stops to function;
- **Erroneous Function:** A hazard occurring when a function behaves incorrectly. For example, a controller that sends the wrong signal (due to, *e.g.*, a software bug), or it sends the expected signal too late or too early.
- **Software Error:**is a further specification of the more general "Erroneous" Guide Word and relates to failure events

[1]The Guidelines mention that other information can be exchanged wihtout entering into the details.

caused by software, i.e. when the function is allocated to software.

A more specific set of guide words can be used, *e.g.*, late response, too early response, etc. The more specific the semantics, the finer can the translations to attack trees be. For example, if an event (leading to a hazard event) is associated with being late, security engineers shall evaluate whether an attacker can cause such delay.

We also associate safety patterns with lightweight semantics. The semantics simply annotates the type of pattern. For example: $\langle \mathsf{MooN}, \mathcal{I}, out, \mathsf{ev}\rangle$ specifies a M out of N voter, taking as input N signals in $\mathcal{I}$, and sends the ev in the channel out.

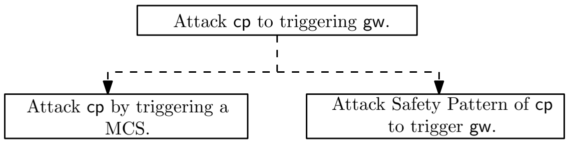
A key advantage of lightweight semantics as opposed to more formal semantics is that it uses terminology used by safety engineers, while formal semantics requires further expertise. It seems possible, however, to use formal semantics to further refine the attack trees generated from the safety analysis.

3) Relating Guide Words to Types of Attacks: Following [14], we associate to each guide word an attack type, *e.g.*, STRIDE attacks [30].

- *loss of function* can be caused by an attacker carrying out a (1) denial of service attack or (2) elevating his privilege to disable the function;
- *erroneous* event can be caused by an attacker (1) tampering with data and generating the erroneous event, or (2) spoofing the device that generates the event.
- *software error* can be caused by an attacker applying a malware and modifying the correct functioning of the system;

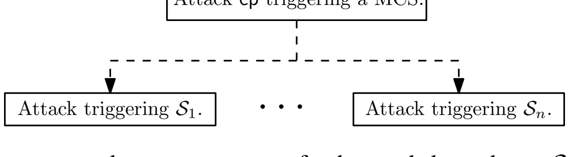
This association of Guide Words to STRIDE notation is done manually. It is a way to correlate the semantics of safety, *i.e.*, Guide Words, to security, *i.e.*, attacks.

4) FTA to Attack Tree: For each one of the identified hazards, an FTA analysis is constructed. There may be many events that trigger such a hazard, for which the minimal cut sets (MCS) are constructed. Each event in the minimal cut sets is associated with lightweight semantics. The attack tree constructed from the FTA for a particular top event starts with the sub-tree:

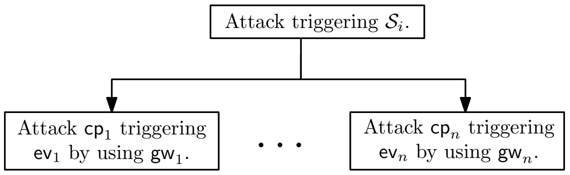

Attack cp to triggering gw.
Attack cp by triggering a MCS.
Attack Safety Pattern of cp to trigger gw.

That is, the attack node associated with the top event hazard is expanded. Either the attacker triggers a minimal cut set or it attacks a safety pattern (described below).

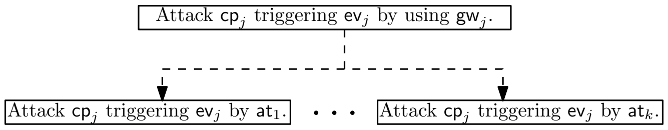
Given the MCS composed by the minimal cut sets $\mathcal{S}_1, \ldots, \mathcal{S}_n$, we expand the tree as shown below:


Attack cp triggering a MCS.
Attack triggering $\mathcal{S}_1$.
. . .
Attack triggering $\mathcal{S}_n$.

Moreover, each event ev_j of the sub-branches $\mathcal{S}_i = \{\mathsf{ev}_1, \ldots, \mathsf{ev}_m\}$, is associated with a lightweight semantics which contain a guide word, gw_j and the model element cp_j. We can further expand the AT above with the sub-trees shown below:


Attack triggering $\mathcal{S}_i$.
Attack cp_1 triggering ev_1 by using gw_1.
. . .
Attack cp_n triggering ev_n by using gw_n.

Finally, since each guide word, gw, is associated with particular attacks, $\mathsf{at}_1, \ldots, \mathsf{at}_k$, we can further expand each one of the sub-branches above with these attacks.


Attack cp_j triggering ev_j by using gw_j.
Attack cp_j triggering ev_j by at_1.
. . .
Attack cp_j triggering ev_j by at_k.

For example, having a fta with specified failure event *Erroneous person detection* of the component e.g. *camera*. This failure event has the specified guide word *erroneous*. The failure type *erroneous* is related to the attack types *tampering* and *spoofing* (see IV-A3). Consequently, the outcome of the transformation in the attack tree will be two more entries, namely, *tampering the camera* and *spoofing the camera*.

a) Discussion: Notice that at the end of the translation, the AT specifies the rational (based on the safety analysis) of how attacks can lead to hazards. Moreover, as the translation depends on the lightweight semantics, more precise lightweight semantics leads to more detailed AT. For example, if guide words detailing the timing of events is used, ATs can also detail attacks that cause such delays.

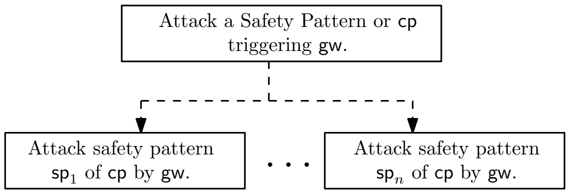
However, it is worth noticing that not all information in safety analysis are necessarily useful for security. For example, criticality analysis of cut sets is carried out without considering an active attacker. Hence, events with low criticality for a safety engineer, such as, door lock fault, does not necessarily correspond to how easy an attacker can trigger that event, such as, the attacker breaking the door lock.

5) Safety Patterns to Attack Trees: Safety patterns in the form of safety requirements are concrete architectural choices in order to control hazards. Attacker can, therefore, also try to trigger a hazard by attacking the implementation of the patterns.

For example, a 2oo2 voter can be used to consider two independent signals in order to reduce the chance of falsely triggering an event, *e.g.*, deploying a car airbag. The attacker can attack such safety mechanism in order to (1) maliciously trigger the event or (2) to maliciously prevent the trigger of such event.

There are a number of safety patterns, normally, deployed by safety engineers. Each one of these patterns can be attacked in different ways as described in [25]. The novelty here is that we can use these analysis and integrate it with the attack tree obtained from the hazard analysis and the FTA analysis. This is done by also using the lightweight semantics associated with these patterns as follows:

1) Collect all the safety patters associated with the element cp. Let $\mathsf{sp}_1, \ldots, \mathsf{sp}_n$ be these design solutions.
2) Construct the AT:


Attack a Safety Pattern or cp triggering gw.
Attack safety pattern sp1 of cp by gw.
. . .
Attack safety pattern spn of cp by gw.

3) Build the AT for each branch with sp_i following the analysis of [25], but refined only considering attacks associated with the guide word gw.

V. Pick and Place

This section illustrates the automatic translation described in the previous section on an use case. Consider a factory element called Pick and Place (PnP).[2] Its purpose is to place a cap on a metallic cylinder. Once an incoming cylinder is detected on the rolling belt, the PnP arm grabs a cap from the cap repository using a sucking device, lifts the cap, positions the cap over the cylinder, and places the cap on the cylinder. The transport capability of the cap (picking, lifting and placing) is realized by a sucking device, which is controlled by an controller. The controller sends a stopSuccking command to position the cap on the cylinder only if there is no person in front (input from a camera installed to provide the person detection function).

Assume lightweight semantics and guide words described in Section IV-A2 and the attacks associated with guide words described in Section IV-A3. We demonstrate the translations only using the PnP use case.

Part of the attack tree is shown in Figure 5. We also modified the node names so that they are not so verbose.

a) Hazard Identification (FHA): First, we identify a list of losses unacceptable to the stakeholders of the PnP. There are two possible losses: **L1:** Loss of life or person injured by the PnP and **L2:** Loss of production. We then conduct a FHA and identify among others the following hazards:

- **Hazard H1**: Person hurt by moving arm, related to L1;
- **Hazard H2**: Release of cap hurting person in falling trajectory, related to L1;
- **Hazard H3**: PnP stops working, related to L2.

b) FTA: For each hazard we conduct a FTA. We describe the FTA analysis for the Hazard H2 shown above. This hazard can occur when the cap is released in an unintended way. This can happen by loss of the sucking function or an erroneous loss of the sucking function. The former can be caused by, for example, *losing the vacuum function* of the cap gripper. The latter can happen by a *software error*, or by error on the sensors detecting when there is someone near the PnP. In particular, there are two sensors in the PnP:

- **Laser Fencing:** Lasers around the PnP to detect if a person is near to the PnP;
- **Camera:** Camera that detects if a person is near the PnP.

A fault in each one this sensors consists a different cut-set.

[2]See https://www.youtube.com/watch?v=Tkcv-mbhYqk starting at time 55 seconds for a very small scale of the PnP.

c) Safety Patterns: Moreover, in order to avoid Hazard H3, these sensors are integrated with a 2oo2 safety pattern, as only when both sensors detect someone near the PnP, the factory moves to a safe mode, where the factory stops. The lightweight semantics for 2oo2 $\langle \mathsf{id}_{2oo2}, \mathsf{2oo2}, \{Camera, Fence\}, stop\rangle$, specifying the input signals used and the event that is triggered.

The information above is incorporated in the attack tree shown in Figure 5. In particular, the FTA analysis leads to different branches and attacks on the 2oo2 pattern can lead to attacks leading to the sucking function to work erroneously. One part of the AT generated that is left open to the user is how an attacker shall proceed in attacking the MooN inputs, marked with three ellipses, *i.e.*, whether it is an “And” or “Or”. This is because the lightweight semantics used does not provide enough information to infer this as it depends on the behavior of the MooN. We conjecture that using more elaborate lightweight semantics, *e.g.*, the Guide Words in [14], will enable this inference.

d) Discussion: As one can observe, the AT generated for the PnP enumerates a large number of attack scenarios that have to be assessed by security engineers. At this point, defenses can be suggested, as security requirements. For example, signing mechanisms in order to mitigate attacks on the MooN installed. Moreover, the lightweight semantics used allows to discard attacks scenarios. For example, Denial of Service attacks were not considered for hazards triggered by erroneous functions.

Indeed, we used the attack tree generated to demonstrate an attack on the PnP, where the intruder spoofed messages from the MooN.

VI. Related Work

We present the state-of-the art regarding co-analysis of safety and security. To this scope, we used the work of Kriaa *et al.*, who present an extensive survey on such analyses [20]. To our knowledge, there is so far no approach that integrates safety and security by allowing safety and security engineers to use their own techniques, thus maintaining a clear separation of concerns. In contrast to the already existing methods, we do not change the already used analysis methods, but provide a translation between them, to enhance them with information from the other analysis.

a) Approaches extending FTAs and FMEAs.: While Kornecki *et al.* report on modeling both safety and security requirements for an air traffic management system via FTAs [19], Fovino *et al.* and Steiner *et al.* propose extensions of FTAs so that ATs can also be represented in the same model, while being associated to events in the FTAs [16], [31]. Schmittner *et al.* propose an extension of the FMEA to also reason about security aspects, such as vulnerabilities and attacks [28].

b) Approaches for safety and security co-analysis.: Given the importance of the topic, there is a plethora of approaches for integrating safety and security analyses. While Reichenbach *et al.*propose Safety Integrity Levels as a factor for risk assesment within the threat vulnerability and risk assessment (TVRA) method from ETSI TS 102 165-1 [26], Friedberg *et al.* propose a methodology for integrating STPA and STPA-sec [17],

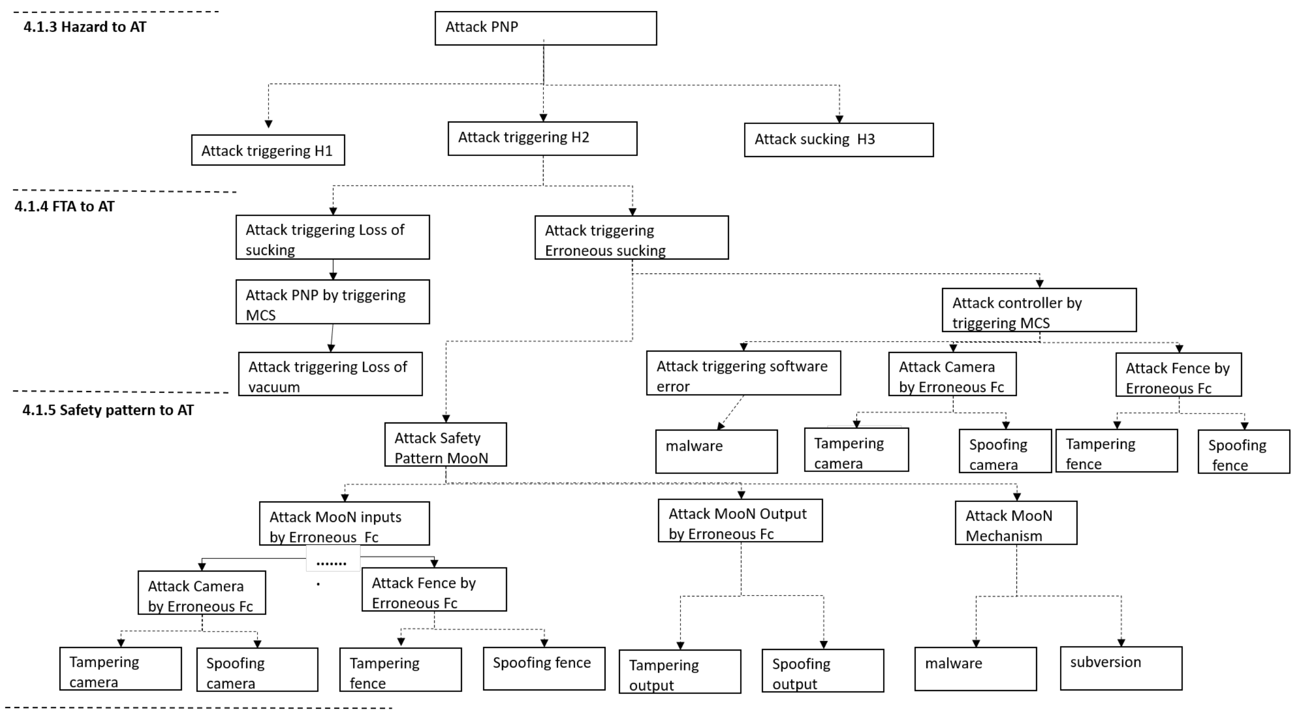

4.1.3 Hazard to AT
Attack PNP
Attack triggering H1
Attack triggering H2
Attack sucking H3
4.1.4 FTA to AT
Attack triggering Loss of sucking
Attack triggering Erroneous sucking
Attack PNP by triggering MCS
Attack controller by triggering MCS
Attack triggering Loss of vacuum
Attack triggering software error
Attack Camera by Erroneous Fc
Attack Fence by Erroneous Fc
4.1.5 Safety pattern to AT
Attack Safety Pattern MooN
malware
Tampering camera
Spoofing camera
Tampering fence
Spoofing fence
Attack MooN inputs by Erroneous Fc
Attack MooN Output by Erroneous Fc
Attack MooN Mechanism
Attack Camera by Erroneous Fc
Attack Fence by Erroneous Fc
Tampering camera
Spoofing camera
Tampering fence
Spoofing fence
Tampering output
Spoofing output
malware
subversion

Fig. 5. Generated Attack Tree for the PNP based on PnP hazard analysis, FTA and safety patterns.

by considering security constraints during the STPA analysis. Several works propose approaches for modeling safety and security requirements and dependencies between them [12], [24], [32], [33]. Also, certain works extend system modeling approaches with an integrated treatment of safety and security requirements [10], [27]. Given the lack of automation in many of these approaches, it seems that they may profit from computer-aided techniques.

VII. Conclusions

This paper investigates how computer-aided methods can help the integration of safety and security. We propose a methodology for building computer aided methods for extracting security relevant information from safety analysis by model transformations. We illustrate this methodology by applying it to an Industry 4.0 application.

These are still the first steps. The set of guide words that we use here is small, thus limiting the types of analysis that can be carried out, *e.g.*, not reasoning about the timing of attacks. We plan to instantiate the proposed methodology with larger sets of guide words, *e.g.*, in [14] for automotive systems.

Another important issue that is left to future work is how to support trade off analysis between safety and security. That is, how security assessment results impact safety architecture decisions and vice versa. For example, applying architectural safety pattern providing a safe state where the system stops is good safety solution but it opens possibility for an attacker to apply denial of service attacks. In this case it should be investigated if another safety solution could better fulfil both safety and security requirements.

Further we investigate, in the White-Paper [23], how security assessments can affect *the confidence on the safety assessments*, expressed using quantitative methods [13]. This interaction shall also take into account trade-off analysis.

References

[1] ED 202A: Airworthiness security process specification. https://standards.globalspec.com/std/9862360/eurocae-ed-202.
[2] ISO 26262, Road vehicles — Functional safety software level. https://www.iso.org/standard/43464.html.
[3] SAE J3061: Cybersecurity guidebook for cyber-physical vehicle systems. https://www.sae.org/standards/content/j3061/.
[4] ARP 4761: Guidelines and methods for conducting the safety assessment. https://www.sae.org/standards/content/arp4761/.
[5] ARP 4754a: Guidelines for development of civil aircraft and systems. https://www.sae.org/standards/content/arp4754a/.
[6] Hackers remotely kill a Jeep on the highway with me in it, 2015. https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/.
[7] Cyberattack on a German steel-mill, 2016. Available at https://www.sentryo.net/cyberattack-on-a-german-steel-mill/.
[8] A deep flaw in your car lets hackers shut dowm safety features, 2018. https://www.wired.com/story/car-hack-shut-down-safety-features/.
[9] Hacks on a plane: Researchers warn it's only 'a matter of time' before aircraft get cyber attacked, 2018. https://tinyurl.com/ycgfa3j8.
[10] Tiago Amorim, Helmut Martin, Zhendong Ma, Christoph Schmittner, Daniel Schneider, Georg Macher, Bernhard Winkler, Martin Krammer, and Christian Kreiner. Systematic pattern approach for safety and security co-engineering in the automotive domain. In SAFECOMP, 2017.
[11] Stefano Bistarelli, Fabio Fioravanti, and Pamela Peretti. Defense tree for economic evaluations of security investment. In *ARES*, 2006.
[12] Julien Delange, Laurent Pautet, and Peter H. Feiler. Validating safety and security requirements for partitioned architectures. In *Ada-Europe* 2009.
[13] Lian Duan, Sanjai Rayadurgam, Mats Heimdahl, Oleg Sokolsky, and Insup Lee. Representation of confidence in assurance cases using the beta distribution. In HASE 2016.
[14] J. Dürrwang, M. Braun, , R. Kriesten, and A. Pretschner. Enhancement of automotive penetration testing with threat analyses results. *SAE Intl. J. of Transportation Cybersecurity and Privacy*, 2018.
[15] Jürgen Dürrwang, Kristian Beckers, and Reiner Kriesten. A lightweight threat analysis approach intertwining safety and security for the automotive domain. In *SAFECOMP* 2017.

[16] Igor Nai Fovino, Marcelo Masera, and Alessio De Cian. Integrating cyber attacks within fault trees. *Rel. Eng. & Sys. Safety*, 94(9):1394–1402, 2009.

[17] Ivo Friedberg, Kieran McLaughlin, Paul Smith, David M. Laverty, and Sakir Sezer. Stpa-safesec: Safety and security analysis for cyber-physical systems. *J. Inf. Sec. Appl.*, 34:183–196, 2017.

[18] Barbara Kordy, Sjouke Mauw, Sasa Radomirovic, and Patrick Schweitzer. Foundations of attack-defense trees. pages 2010.

[19] Andrew J. Kornecki, Nary Subramanian, and Janusz Zalewski. Studying interrelationships of safety and security for software assurance in cyber-physical systems: Approach based on bayesian belief networks. In *FCSIS*, 2013.

[20] Siwar Kriaa, Ludovic Piètre-Cambacédès, Marc Bouissou, and Yoran Halgand. A survey of approaches combining safety and security for industrial control systems. *Rel. Eng. & Sys. Safety*, 2015.

[21] Nancy Leveson. *Engineering a Safer World: Systems Thinking Applied to Safety Engineering systems*. MIT Press.

[22] Clifton A.Ericson. *Hazard Analysis Techniques for Safe Systems*. Wiley.

[23] Vivek Nigam, Alexander Pretschner, and Harald Ruess. Model-based safety and security engineering. White Paper https://arxiv.org/abs/1810.04866, 2018.

[24] Ludovic Piètre-Cambacédès and Marc Bouissou. Modeling safety and security interdependencies with BDMP (boolean logic driven markov processes). In *ICSMC*, 2010.

[25] Christopher Preschern, Nermin Kajtazovic, and Christian Kreiner. Security analysis of safety patterns. PLoP, 2013.

[26] Frank Reichenbach, Jan Endresen, Mohammad M. R. Chowdhury, and Judith E. Y. Rossebø. A pragmatic approach on combined safety and security risk analysis. In *ISSRE* 2012.

[27] Yves Roudier and Ludovic Apvrille. Sysml-sec - A model driven approach for designing safe and secure systems. In *MODELSWARD* 2015.

[28] Christoph Schmittner, Thomas Gruber, Peter P. Puschner, and Erwin Schoitsch. Security application of failure mode and effect analysis (FMEA). In SAFECOMP 2014.

[29] B. Schneier. Attack trees: Modeling security threats. *Dr. Dobb's Journal of Software Tools*, 24:21–29, 1999.

[30] Adam Shostack. *Threat Modeling: Designing for Security*. Wiley.

[31] Max Steiner and Peter Liggesmeyer. Combination of safety and security analysis - finding security problems that threaten the safety of a system. In *SAFECOMP*, 2013.

[32] Nary Subramanian and Janusz Zalewski. Assessment of safety and security of system architectures for cyberphysical systems. In *SysCon* 2013.

[33] Mu Sun, Sibin Mohan, Lui Sha, and Carl Gunter. Addressing safety and security contradictions in cyber-physical systems, 2009.