SESAME
SECURE AND SAFE MULTI-ROBOT SYSTEMS

Project Number 101017258

D4.3 Safety-Security Co-Engineering Framework

Version 1.0
29 June 2022
Final

Public Distribution

University of Hull and FORTH

Project Partners: **Aero41, ATB, AVL, Bonn-Rhein-Sieg University, Cyprus Civil Defence, Domaine Kox, FORTH, Fraunhofer IESE, KIOS, KUKA Assembly & Test, Locomotec, Luxsense, The Open Group, Technology Transfer Systems, University of Hull, University of Luxembourg, University of York**

PROJECT PARTNER CONTACT INFORMATION

Aero41 Frédéric Hemmeler Chemin de Mornex 3 1003 Lausanne Switzerland E-mail: frederic.hemmeler@aero41.ch	**ATB** Sebastian Scholze Wiener Strasse 1 28359 Bremen Germany E-mail: scholze@atb-bremen.de
AVL Martin Weinzerl Hans-List-Platz 1 8020 Graz Austria E-mail: martin.weinzerl@avl.com	**Bonn-Rhein-Sieg University** Nico Hochgeschwender Grantham-Allee 20 53757 Sankt Augustin Germany E-mail: nico.hochgeschwender@h-brs.de
Cyprus Civil Defence Eftychia Stokkou Cyprus Ministry of Interior 1453 Lefkosia Cyprus E-mail: estokkou@cd.moi.gov.cy	**Domaine Kox** Corinne Kox 6 Rue des Prés 5561 Remich Luxembourg E-mail: corinne@domainekox.lu
FORTH Sotiris Ioannidis N Plastira Str 100 70013 Heraklion Greece E-mail: sotiris@ics.forth.gr	**Fraunhofer IESE** Daniel Schneider Fraunhofer-Platz 1 67663 Kaiserslautern Germany E-mail: daniel.schneider@iese.fraunhofer.de
KIOS Maria Michael 1 Panepistimiou Avenue 2109 Aglatzia, Nicosia Cyprus E-mail: mmichael@ucy.ac.cy	**KUKA Assembly & Test** Michael Laackmann Uhthoffstrasse 1 28757 Bremen Germany E-mail: michael.laackmann@kuka.com
Locomotec Sebastian Blumenthal Bergiusstrasse 15 86199 Augsburg Germany E-mail: blumenthal@locomotec.com	**Luxsense** Gilles Rock 85-87 Parc d'Activités 8303 Luxembourg Luxembourg E-mail: gilles.rock@luxsense.lu
The Open Group Scott Hansen Rond Point Schuman 6, 5th Floor 1040 Brussels Belgium E-mail: s.hansen@opengroup.org	**Technology Transfer Systems** Paolo Pedrazzoli Via Francesco d'Ovidio, 3 20131 Milano Italy E-mail: pedrazzoli@ttsnetwork.com
University of Hull Yiannis Papadopoulos Cottingham Road Hull HU6 7TQ United Kingdom E-mail: y.i.papadopoulos@hull.ac.uk	**University of Luxembourg** Miguel Olivares Mendez 2 Avenue de l'Universite 4365 Esch-sur-Alzette Luxembourg E-mail: miguel.olivaresmendez@uni.lu
University of York Simos Gerasimou & Nicholas Matragkas Deramore Lane York YO10 5GH United Kingdom E-mail: simos.gerasimou@york.ac.uk nicholas.matragkas@york.ac.uk	

DOCUMENT CONTROL

Version	Status	Date
0.1	Initial outline	16 June 2022
0.2	Initial (separate) input from Hull & FORTH	22 June 2022
0.4	Combined version	23 June 2022
0.8	Next iteration following input from FORTH and Hull	26 June 2022
1.0	Final version incorporating feedback from York & Luxembourg	29 June 2022

TABLE OF CONTENTS

TABLE OF FIGURES

EXECUTIVE SUMMARY

The advantages of a model-based approach for safety have been clear for many years now. However, security analysis is typically less formal and more ad-hoc; it may involve systematic processes but these are not generally tied into a formal model-based development and analysis process in the same way that safety can be.

Task 4.3 of the SESAME project, Safety/Security Co-Engineering, sets out to remedy this by investigating a holistic co-engineering approach that integrates both of these different concerns.

In this report we therefore present a combined safety/security co-engineering framework based on the ODE, the metamodel that serves as a basis for the EDDI dependability management concept. The ODE acts as a common ontology for both safety and security, establishing equivalencies between key concepts and allowing joint analyses to take place in which failures can be incorporated into security analysis and attacks into safety analysis. The combined results indicate the causes and consequences of hazardous events regardless of whether they originate from safety or security issues, and the same risk estimation applies to them all.

While developed for design-time usage, this framework paves the way for the generation of combined safety/security artefacts at runtime as well. The common approach means that specification of requirements, event monitors, diagnostic engines, and responses/actions can take advantage of both safety and security information stored in the design-time models.

To demonstrate the approach, we apply it to a drone-based case study derived from two of the SESAME use cases.

LIST OF ABBREVIATIONS

ASIL	Automotive Safety Integrity Level
BE	Basic Event (i.e., root cause of a fault tree)
BN	Bayesian Network
CAPEC	Common Attack Pattern Enumeration and Classification
CCF	Common Cause Failure
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
CWE	Common Weakness Enumeration
DDI	Digital Dependability Identity
EDDI	Executable Digital Dependability Identity
FMEA	Failure Modes & Effects Analysis
FMEDA	Failure Modes, Effects, & Diagnostic Analysis
FMVEA	Failure Modes, Vulnerabilities, and Effects Analysis
FTA	Fault Tree Analysis
GCS	Ground Control Station
HARA	Hazard Analysis & Risk Assessment (or Hazard And Risk Analysis)
IMU	Inertial Measurement Unit (for drone navigation)
MAS	Multi-Agent System
MBSA	Model-based Safety Analysis
ML	Machine Learning
MRS	Multi-Robot System
MTTF	Mean Time To Failure (also, MTBF: Mean Time Before Failure or Mean Time Between Failure)
OAS	Open Adaptive System
ODE	Open Dependability Exchange

RPN	Risk Priority Number
RVD	Robot Vulnerability Database
SACM	Structured Assurance Case Metamodel
SAHARA	Security-Aware Hazard and Risk Analysis
SIL	Safety Integrity Level
TARA	Threat Analysis & Risk Assessment
UAS	Unmanned Aircraft/Airborne System (also, UAV: Unmanned Aerial Vehicle)

1. INTRODUCTION

1.1 MODEL-BASED SAFETY & SECURITY ANALYSIS

As the complexity of safety-critical systems has increased, new techniques have had to be developed to keep pace in an attempt to ensure their safety. Traditionally, safety analysis & engineering were intensive, largely manual processes which are too slow and too costly to apply to modern systems, where more agile approaches are required.

The field of Model-based Safety Analysis (MBSA) has emerged as one of the key technologies to address this issue. MBSA leverages the power of computer-aided tools and sophisticated metamodelling to create a more holistic approach to safety, one which uses the same design artefacts as the nominal design. This offers important advantages in terms of keeping the safety analyses in sync with the current state of the design, preventing discrepancies between the models, and better supports a tightly integrated, iterative safety engineering process. In MBSA, system models are extended with dependability-related information, ranging from failure data to propagation logic to safety requirements and more. Because this data is embedded in the same model, changes to the design can be rapidly reflected in the safety analysis results. The use of standardised models also helps encourage reuse and common exchange between different tools.

However, as described in **D4.1: Safety Analysis Concept and Methodology**, even MBSA is not immune to the challenges posed by modern systems as they become increasingly dynamic, complex, and distributed — thereby rendering them vulnerable to security attacks and software issues as well as hardware failures.

This deliverable deals with this issue of security and how it intersects with safety, which constitutes Task 4.3 of the SESAME project: Safety-Security Co-Engineering. As safety-critical systems make increasing use of computational resources and distributed networks, their vulnerability to security threats also increases. As such, it is important that security be considered as part of the overall dependability assessment process to ensure that these vulnerabilities are identified, their effects on safety evaluated, and mitigations or countermeasures put in place accordingly.

Although systematic security analysis processes exist, like those discussed in **D5.1: Security Analysis Concept**, these are generally based on different models than those used in safety and are often not integrated into wider dependability processes. They also tend to be either design-time or run-time only, rather than spanning the full lifecycle of the system from design to operation.

All of these issues are particularly magnified when applied to multi-robot systems (MRS) and other multi-agent systems. Security plays a much larger role in such systems, where the different robots necessarily communicate often with each other and any controllers in order to achieve their tasks. Furthermore, not only are such systems more vulnerable, but the consequences of security breaches are often more severe as well — particularly when dealing with autonomous robots or unmanned aerial vehicles (UAVs) like drones.

1.2 COMBINING SAFETY AND SECURITY

To address these issues, in SESAME we have been developing a combined model-based approach to safety & security, one that is intended to enable a holistic analysis of multiple dependability properties (e.g. safety, security, reliability, perhaps even maintenance) over the entire system life cycle, from design time to runtime. The basis for this process is a set of executable models — the EDDIs (Executable Digital Dependability Identities) — and the common metamodel that underlies them, the ODE or Open Dependability Exchange.

The first step, therefore, is to ensure that the ODE is capable of representing both safety and security issues as part of the same model, as well as being able to establish casual relationships between them. This has been achieved on several levels:

- A *SecurityViolation* is modelled as a sub-type of *Failure* and can thus appear wherever a *Failure* might, e.g. as the cause of a hazard or as an interface failure at a component output.
- *IDSEvents* and *IDSEventMonitors* provide scope for modelling security-specific events (and event monitors) at runtime.
- Attack Trees are modelled using the same structures as Fault Trees, making them compatible with each other;
- The ODE also contains an entire TARA (Threat Analysis & Risk Assessment) package which has been expanded to support the various vulnerability databases in use (e.g. RVD, CWE, CAPEC etc. — see Section 2.2).

Further information about the ODE and its support for safety and security can be found in **D4.2/D5.2: Safety/Security ODE and EDDI Specification**.

In addition to the metamodel, however, it is also useful to harmonise the methodologies involved. Risk assessment in the safety domain is generally a more formal, structured process designed to identify possible hazards and evaluate the risks involved in a systematic fashion. When incorporated as part of an MBSA approach, it also benefits from formal models and parity with the design artefacts.

Although there are of course also structured processes for security analysis (such as those described in **D5.1**), they are typically not part of a formal model-based approach as with modern safety-related risk assessments. To that end, the approach we propose in this deliverable attempts to harmonise both safety and security risk assessment as part of a common model-based process. This reduces the risk of discrepancies emerging between one or the other form of analysis, improves interoperability, and better allows the safety implications of security threats to be captured and addressed.

Later, this combined process can be extended to lead into the runtime aspects of EDDI generation and execution. In this work package (WP4), however, we focus primarily on the design-time considerations.

Finally, to demonstrate the approach, we have applied it to a drone-based use case similar to the SESAME use cases from KIOS/Cyprus Civil Defence (power station

inspection) and Domaine Kox/LuxSense/Aero41 (pest control in viticulture using drones).

1.3 EXISTING APPROACHES

There are various existing standards for both safety-based risk assessment and security-based risk assessment. Among those standards, ISO/IEC 27005:2018 [1], IEC/TS 62443–1–1 [2], IEC 62443–3–2 [3], and NIST SP 800–30 [4] have almost the same steps for security risk assessment, namely:

1. Identifying Assets
2. Identifying Vulnerabilities
3. Identifying Threats
4. Determine consequences and Impacts
5. Assess incident likelihood
6. Assess risk level

Although some additional steps are present in a subset of the standards — such as "Determine Security Level Targets", which is present only in IEC/TS 62443–1–1 and IEC 62443–3–2 — the convergence on this common set of steps across several standards suggests broad consensus on these as being the most important. Therefore any attempt to combine them with standard safety workflows should ensure they are all included in some form.

Prior such attempts to harmonise safety and security risk assessment methodologies do exist. In 2018, Skoglund *et al.* [5] investigated possible synergies in the automotive domain when dealing with the major automotive functional safety standard ISO 26262 [6] and a cybersecurity standard, ISO/SAE 21434 [7]. One of the results was the V-lifecycle reproduced in Figure 1, which shows the synergies between safety and security based on those standards.

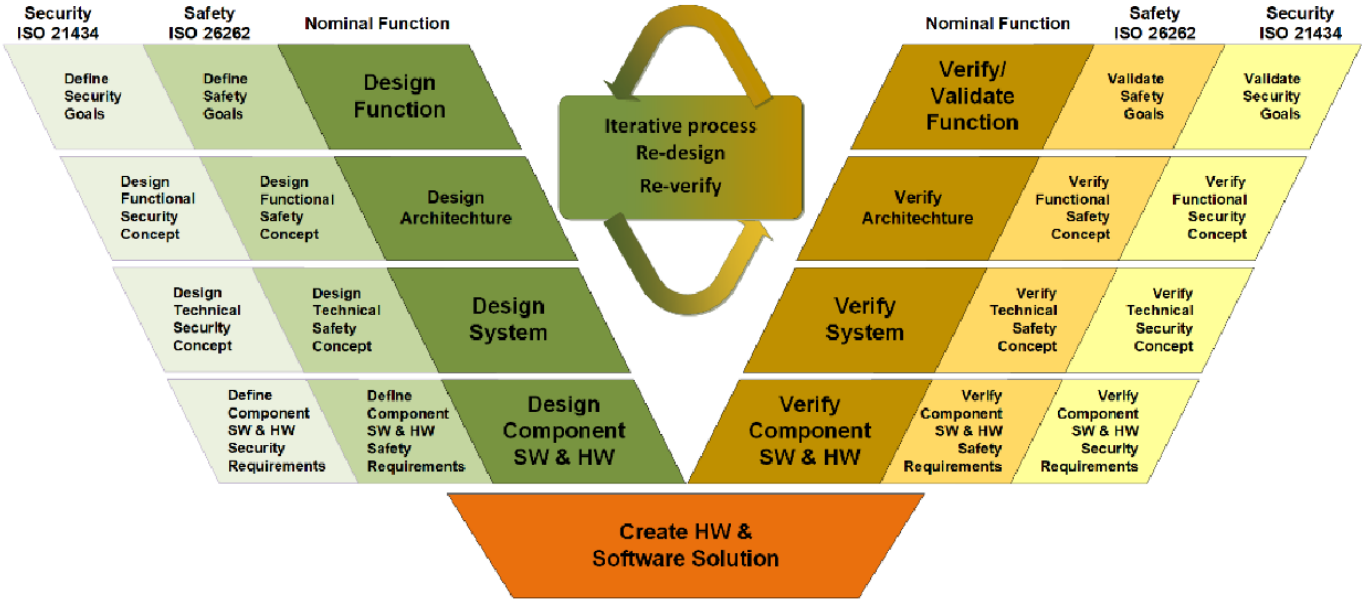


Figure 1: V-model for Safety and Security Co-engineering (from [5])

This takes a very high-level view of the entire lifecycle, however. For a more practical co-engineering perspective, Dobaj *et al.* [8] focused on integrating the quantitative aspects of safety and security risk assessment. They combine SAHARA (Security-Aware Hazard and Risk Analysis), FMVEA (Failure Modes, Vulnerabilities, and Effects Analysis), and Diamond models into one common model. Their combined model revealed that risk analysis methods do not consider all relevant aspects that are studied in security incident analysis, and do not take under consideration the victim and mitigation strategy models required for the system resistance calculation. They propose the model could be extended to i) include active security mitigation measures, ii) be applicable to a whole attack graph instead of just one conducted attack, and iii) include the aspect of time, being able to assess the attacker's capabilities and the target system's resistance over time.

Lyu *et al.* [9] compared existing risk assessment approaches in the light of safety and security. The comparison results are presented along with the gap between what is needed and what is currently available. Among the identified gaps are the lack of i) algorithms to resolve safety and security conflicts, ii) unified safety and security measurements, and iii) dynamic risk assessment.

A novel framework for providing systematic pattern-based safety and security co-engineering is proposed by Martin *et al.* in [10]. This framework is capable of generating guidance for the selection and combination of both types of patterns in the context of system engineering. In [11], the authors analysed state-of-the-art methods for integrated security, safety, and reliability engineering. The framework they propose takes advantage of SAHARA, FMVEA, and attack tree analysis (ATA) and served as a partial inspiration for the work of Dobaj *et al.* mentioned above.

Schwarzl *et al.* [12] summarized well-known safety and security methods in the automotive sector. In this paper, highly automated vehicles (HAVs) were targeted and a framework for safety and cybersecurity co-analysis and co-design was introduced. Witte *et al.* [13] continued this by emphasising self-adaptation and proposed a

solution that considers various system attributes and its adaptation mechanisms as part of both safety and security analysis.

Alanen *et al.* [14] focused on industrial control systems and proposed a hybrid ontology for safety-security risk assessment and co-engineering, with the intention of forming the basis for model-based approaches for risk evaluation of mission-critical systems. The study was the extension of the idea that has been proposed in [15], adding a case study of a nuclear power plant. However, the paper considered only design-time risk evaluation, omitting consideration of runtime risk evaluation, and focused more on the combined ontology (or metamodel, effectively) than the application of that ontology in a combined safety/security risk assessment methodology.

In the robotic domain, Gleirscher *et al.* [16] studied the socio-technical challenges such systems pose for risk assessment, including ethics, risk prioritisation, risk representation and communication, coordinating and integrating risk analyses, ensuring quality and availability of third-party evidence, handling dynamic threat and impact landscapes, resourcing safety and security interactions, and increasing automation of assurance, amongst others.

Similar to security-based risk assessment, safety-based risk assessment has equivalent series of recommended steps for risk evaluation. More information can be found in Section 5 of **D4.1: Safety Analysis Concept and Methodology**, which discusses the topic in greater detail, but in summary the aim of these steps is identify what can go wrong in a system, investigate how it can go wrong, and assess what the consequences are for safety if it does go wrong (typically by basing it on the worst-case outcome).

The exact details can vary from standard to standard**,** but generally these steps include:

1. Hazard identification
2. Determining causes in terms of degradation, faults, and failures
3. Estimating failure likelihood
4. Qualitative risk level assessment
5. Quantitative risk evaluation

These general steps are commonly found in relevant safety standards like ISO 12100:2010 [17] and ISO 26262. However, as with security, this focuses only on the design-time aspects; if the risk evaluation process is to be extended to incorporate the runtime portion of the system lifecycle, consideration must also be given to situation-aware dynamic risk evaluation — such as the approach pursued in SESAME via the EDDI concept.

2. SESAME SAFETY & SECURITY CO-ENGINEERING FRAMEWORK

The concept underpinning the combined safety & security "co-engineering framework" — henceforth SSCOF for brevity — being developed in SESAME is illustrated in the figure below, which is based on the key steps common to multiple security standards, as described in the previous section, as well as the common steps for safety-based risk assessment (likewise described previously).

The framework defines an ontological equivalency between safety and security, like the approach in [14], but also supports practical use of combined analysis techniques like FMVEA (used in [8]) and merged fault trees & attack trees (similar to [11]). However, SSCOF is not directly derived from any of them and is distinct in that it is also intended to support the transition to joint runtime management of safety & security via EDDIs. This means it includes elements that would otherwise be superfluous at design time, but which will become more valuable when the models created are used to generate runtime event monitors and diagnostic engines etc.

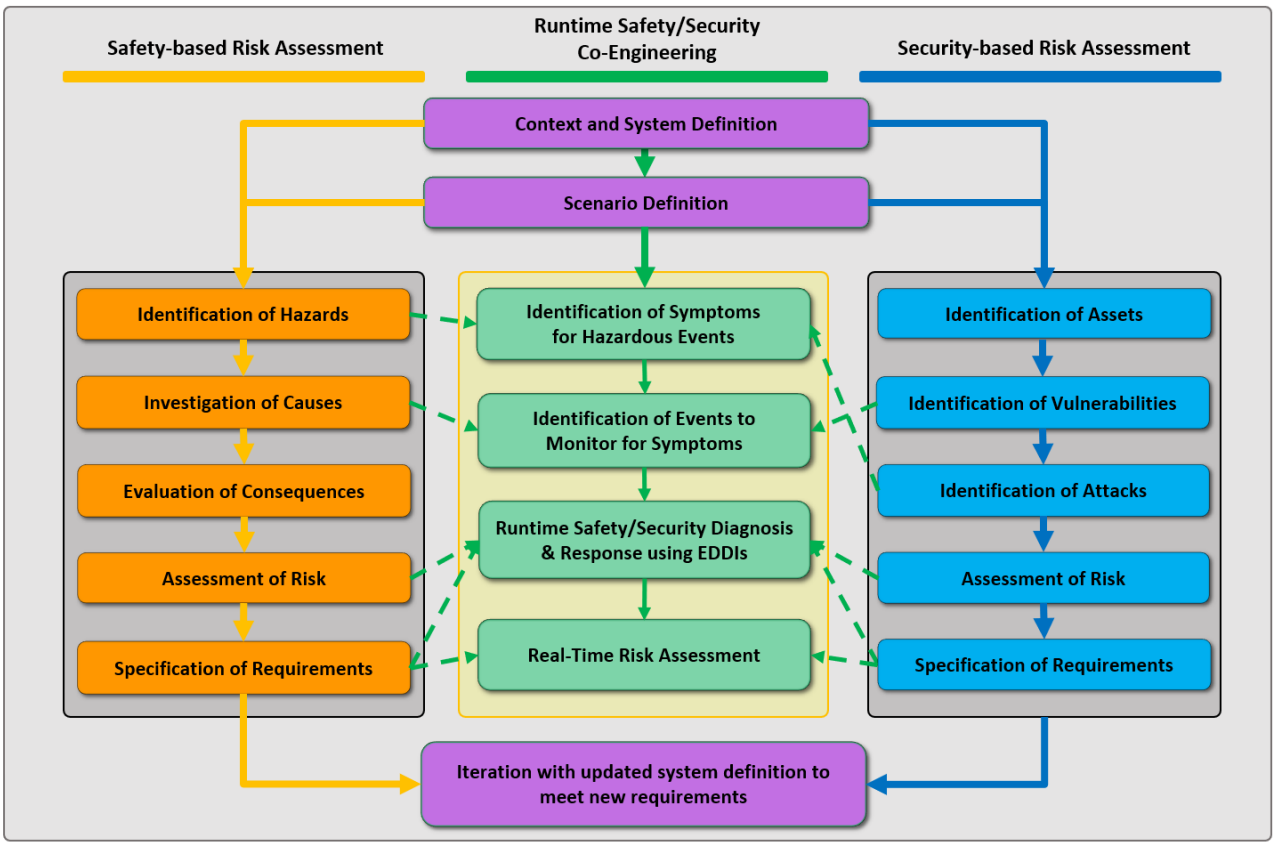


Figure 2: The SESAME Safety/Security Co-Engineering Framework (SSCOF)

At the top are two prerequisites necessary to everything else: definition of the system itself and definition of scenarios (i.e., consideration of the situations and environments in which the system is to operate). These inform both safety and security, as well as later generation of runtime risk evaluation artefacts.

The six security steps mentioned earlier are condensed here into five, though the tasks involved remain unchanged. Identification of assets, vulnerabilities, and attacks (threats) remain unchanged, but we have combined the assessment of

consequences/impacts, likelihood, and risk level into a single step and added a requirements step to the end.

The requirements step is included to reflect the fact that during system design, risk assessment is typically a key driver of the requirements specification process. Risk assessment is not carried out in isolation and must fit into the wider design lifecycle; in this case, the requirements inform the next iteration of the design, where these requirements must be met (and then validated and verified, as in the common V-lifecycle like that in Figure 1). This is made explicit by the box at the very bottom.

On the safety side, the general steps are easy to see once more, though again we have condensed both qualitative and quantitative risk assessment into a single step and added a requirements step afterwards.

The framework encompasses both safety and security at design time — highlighting their common aspects — and is designed to facilitate the production of common elements required for the later generation of runtime EDDI artefacts, such as event monitors, diagnostic engines, and real-time risk evaluators. Note however that these runtime artefacts would likely only be produced after the final design iteration.

The left-most column of the diagram indicates the (design-time) safety-based stream. This focuses primarily on HARA — hazard analysis and risk assessment. The right-most column is the security-based stream and focuses on TARA — threat analysis and risk assessment. The middle column indicates how both of these contribute to the preparatory steps necessary to generate runtime risk evaluation artefacts like EDDIs: identification of possible symptoms of hazards/attacks (such as loss of functionality), consideration of how these symptoms might be detected via event monitors, generation of a suitable diagnostic engine and specification of failure/attack response, and finally the creation of the executable real-time risk assessment models themselves, which are necessarily informed by the safety/security requirements.

2.1 SAFETY: HAZARD ANALYSIS & RISK ASSESSMENT

As discussed in **D4.1**, there is a variety of techniques that can be utilised in MBSA and safety engineering in general to identify hazards, determine their causes, and assess their consequences in order to arrive at an evaluation of the risk they pose. HAZOP (Hazard & Operability Studies) [18], FMEA (Failure Modes And Effects Analysis) [19] and its derivatives, and Fault Tree Analysis (FTA) [20] are amongst the most common safety analysis techniques used in this context, although they each work in different ways and offer different pros and cons.

For the purposes of the SSCOF, we do not mandate any particular analysis technique but rather a series of generic steps that match across both security and safety fields. Any of the aforementioned techniques would be capable of fulfilling these steps, or they could be employed in conjunction with each other to provide greater coverage and higher confidence in the findings.

The steps from the safety side are described below. Note that terms in italics refer specifically to ODE model elements.

A. Context and System Definition

Before anything else, it is necessary to define the system and its operating context — particularly the bounds of the system. What tasks is the system intended to carry out? What functions does it need in order to be able to do this? The result of this stage is typically a form of functional design architecture, outlining the system in abstract terms. This makes it clearer what the system is and thus what needs to be analysed further.

In terms of EDDIs and the ODE, system definition is supported by the ODE::Design package[1], which contains elements for different parts of the system at varying levels of abstractness. For instance, early designs may use generic *Functions* and *Systems*, while later design architectures may evolve these into more concrete *PhysicalComponents* or *LogicalComponents* (i.e., software). The *Context* provides additional information about the system context, e.g. estimated lifetime/mission time and other information about system usage. The *SystemBoundary* also allows boundaries to be defined explicitly.

B. Scenario Definition

Safety standards like ISO 26262 mandate not only the definition of the system (or item) but also the consideration of its likely operating environment(s) and functional scenarios. In the case of a car, for example, combinations of different road environments (e.g. urban, rural, highway), different trip types (e.g. commute, long-distance), and different weather conditions (e.g. hot, rain, snow, ice) may all be considered.

This is important because the risk of a particular hazard depends greatly upon the context. A loss of braking power, for instance, has very different consequences at high speeds on a wet or icy surface than it does at slow speeds on a dry surface.

ISO 26262 imaginatively calls this combination of hazard and scenario the "hazardous scenario" and typically the worst case is assumed for the purposes of safety requirement specification.

1. Identification of Hazards

The earlier stages are applicable to both security and safety; both need to know what the system is as well as the context in which it may operate. The identification of hazards is the first safety-specific step and is the first stage of a normal HARA. The goal here is to enumerate the possible hazards: conditions or events that may lead to an undesirable outcome. For example, this may involve a loss of necessary functionality (e.g. propulsion for a drone, navigation or perception for a vision-aware robot etc.) or an unintended behaviour of a functionality (such as a mis-perception of the environment, e.g. failing to recognise a nearby person who may be at risk of harm as a result).

In ODE terms, the ODE::HARA package provides the most relevant element: the *Hazard*. It contains multiple fields which will become useful in the next couple of stages.

[1] See **D4.2/D5.2: Safety/Security ODE and EDDI Specification** for further information on the ODE packages.

2. Investigation of Causes

With hazards identified, we can investigate possible causes. This may be done deductively via an FTA (starting with the hazard and working back step by step to determine immediate causes or combinations of causes) or inductively via an FMEA (considering each possible failure and determining its effects, i.e., which hazards it may cause). Either way, this establishes causal relationships between hazards and particular faults or failures of the system components.

As part of this, the likelihood of each cause is often considered as well. Early in the design process, this is most likely a simple qualitative measure, e.g. a score out of 5 or 10 or a scale such as "Very Low", "Low", "Medium", "High", and "Very High". Later, this might be a probability or other quantitative measure, e.g. a failure rate.

Causes can be modelled in the ODE in various ways. Failure models, including FMEA and FTA, can be linked directly from the ODE design model to provide evidence of how the system failures were determined, as well as to record the causal relationships and outcomes of the analysis. Causes themselves are typically modelled as *Failures*, part of the ODE::FailureLogic package, with associated details such as failure class and probability distribution(s). The *Hazard*, mentioned above, contains fields to define their causes in terms of *Failures*.

3. Evaluation of Consequences

Having considered the possible causes of hazards, we also need to evaluate what the consequences might be. As mentioned earlier, this involves consideration of the operating scenario at the time the hazard occurs. A qualitative severity score is typically assigned, though again this may be numeric (1-5, 0-10 etc.) or textual ("Minor", "Major", "Dangerous", "Catastrophic etc.). Criteria are often attached to these scores; for example, ISO 26262 defines the following 4-point scale for severity of hazardous scenarios:

- S0: No injuries
- S1: Light or moderate injuries
- S2: Severe or life-threatening injuries (but survival is probable)
- S3: Life-threatening injuries (but survival is uncertain) or fatal injuries

Sometimes an additional score is also determined. This can have different forms; e.g. ISO 26262 uses "controllability", a measure of how easy it is to contain or mitigate the hazard. An FMEDA would use "detectability", a measure of how easy it is to detect or diagnose a particular failure (and thus mitigate it).

In the ODE, consequences of *Hazards* are modelled as *Accidents*, i.e., the impact of a hazard in a given scenario (or *Situation* in ODE terminology).

4. Assessment of Risk

Based on the identified hazards and their consequences, along with the scores determined at each stage, the overall risk of each hazard can be assessed. Typically this is done multiplicatively, with each score being multiplied to obtain a Risk Priority Number (RPN). For example, a hazard with a severity of 3 and a likelihood of 4 would have an RPN of 12 (i.e., 3 * 4). If detectability is used then it is also combined (or rather its inverse is used, i.e., undetectability); e.g. a hazard that is difficult to detect may have a score of 4, yielding a total RPN of 48 (i.e., 3 * 4 * 4).

The RPN allows designers to estimate and compare the criticality of different hazards, which informs decision-making about where to prioritise effort. Depending on the safety standard applicable in the domain in question, it may also serve as input into the specification of the safety requirement, i.e., a higher RPN means a more stringent safety requirement (see below).

The ODE provides a *RiskAssessment* element to encapsulate this information. Input may come from the *Accident* (which has a severity rating), *Situation* (which has a likelihood), and *Failure* itself (which has a variety of fields relating to probability and effects). The *Hazard* may also include *Measures* taken to address or mitigate the consequences.

5. Specification of Safety Requirements

Now that the risk has been assessed for each hazard, safety requirements can be specified to try to limit or mitigate that risk. The higher the risk, the more stringent the requirement. Modern safety standards typically quantify this using a system of safety integrity levels (SILs), though the actual term used differs from domain to domain. As with the constituents of the risk calculation, SILs typically represent a qualitative scale of safety priority and stringency. In some cases, as with ISO 26262, the SIL is determined directly from the RPN components:

		C1	C2	C3
S1	E1	QM	QM	QM
	E2	QM	QM	QM
	E3	QM	QM	A
	E4	QM	A	B
S2	E1	QM	QM	QM
	E2	QM	QM	A
	E3	QM	A	B
	E4	A	B	C
S3	E1	QM	QM	A
	E2	QM	A	B
	E3	A	B	C
	E4	B	C	D

Table 1: ASIL determination according to ISO 26262

Here, S1-S3 are the severity rankings (S0 means a hazard is assumed to have no safety-specific impact and is thus ignored), E1-E4 are the exposure (or likelihood) ratings, and C1-C3 are the controllability ratings. The result is an ASIL — an Automotive SIL — that ranges from QM (Quality Measures only) at the lowest, least stringent level to ASIL D at the highest, most strict level.

The ODE provides elements to model safety requirements (and dependability requirements in general) both as part of its ODE::Dependability::Requirement subpackage as well as via its SACM elements. Most relevant here is the *SafetyRequirement*, which details a variety of properties, including maintenance, mitigation measures, the related *Hazards* and *Failures*, any required *Actions*, as well as required detection and reaction times etc.

Since it is dependent on the standard(s) being applied, the SIL associated with the requirement is defined as part of the ODE::Dependability::Domain subpackage. This allows explicit definition of the *Standard* as part of the ODE model and defines the corresponding *AssuranceLevels*.

6. Iteration and Decomposition

Although not explicitly shown in the diagram, as mentioned, the design process of safety/security-critical systems — especially if MBSA is being applied — is generally iterative in nature. The output of one risk assessment informs the next design iteration by identifying flaws and highlighting potential solutions.

Ideally, therefore, the HARA process is repeated throughout the design lifecycle, becoming more detailed at each stage as the initial, abstract functions evolve into more detailed components with more data available. In the process, top-level requirements are decomposed into sub-requirements that apply to those system elements that may be responsible for causing the hazard that motivated the requirement, and these elements are allocated SILs accordingly.

The ODE facilitates this process by allowing any requirement to be decomposed into "refined" requirements, creating a hierarchical requirements tree. Related safety argumentation to indicate the safety claims being made and record the evidence supporting them is provided by the SACM elements.

2.2 SECURITY: THREAT ANALYSIS & RISK ASSESSMENT

The ultimate goal of the security analysis, conducted in the context of SESAME, is the identification of all the potential attacks that an adversary could initiate against the target system. Since an attacker exploits a known vulnerability of one or more system components to launch their attack, the security analysis drives us to the discovered attacks though the identified system vulnerabilities.

The SESAME security methodology, described in **D5.1: Security Analysis Concept**, includes a set of processes that aim to take advantage of the threat modelling process by adopting many of its well-defined steps. The major stages of this methodology are described below.

Note that as above, these steps are reliant on the prerequisite stages A and B. Any terms in italics refer to the corresponding ODE model elements.

1. Identification of assets

The very first step of the SESAME security analysis is the description of the system in question, in terms of the software and hardware involved. Additional information may include entry points and trust boundaries of the system. Much of this is formulated during the initial Context and System Definition step, described earlier, but there are unique demands in terms of security that require an additional process: the identification of assets. An asset is a specific system element with security implications, e.g. a software component or a hardware component with networking/computing capabilities.

Security analysis can only proceed when the design is sufficiently mature that concrete components/software have been selected, since the vulnerabilities depend on the precise component being used. There are two different approaches to gathering this system information.

Firstly, the system designer can provide such information by filling in forms and answering corresponding questionnaires. This way of gathering system information is useful for systems that are not fully implemented yet, i.e. still at the design stage.

Secondly, for operational systems (either deployed or at the prototype stage), the system description can also be obtained automatically by scanning it with a set of automated tools. These scanners can scan a given network and/or subnetworks for available services and then use open vulnerability databases to discover known vulnerabilities. OpenVAS[2] and w3af[3] are two such scanners.

In terms of the ODE, the ODE::Design package defines elements for basic system description, but the ODE::TARA package provides additional security-specific elements to record further information. Of these, the most relevant here is the *Asset*, which provides a linkage point for other security-related elements.

2. Identification of vulnerabilities

The next step is the identification of vulnerabilities. In this context, a vulnerability is a flaw in a component that makes it vulnerable to a security attack. During this step, the target system is described and analysed in terms of architecture components, assets, entry points and trust boundaries. This description enumerates all of the deployed programs, libraries and services, which can then be checked against catalogues and databases of known vulnerabilities for security issues. Such databases include:

- CVE — Common Vulnerabilities and Exposures
- RVD — Robot Vulnerability Database

The corresponding ODE element is the *Vulnerability*, which encapsulates information about the vulnerability derived from the catalogues/databases. Such information may

[2] https://openvas.org/
[3] https://w3af.org/

include identifiers that cross-reference the catalogue entry (e.g. CVE or RVD ID) as well as a description of the vulnerability, potentially including a severity score.

3. Identification of potential attacks

The list of identified vulnerabilities discovered serves as input for the next step: the identification of potential attacks. An attack in this context is an attempt by a hostile actor to take advantage of a vulnerability to perform a malicious act, e.g. gaining access to the system, causing errors or malfunctions, or even gaining control over it.

Again, databases of known weakness and attack types are utilised here:

- CWE — Common Weakness Enumeration
- CAPEC — Common Attack Pattern Enumeration and Classification

The CWE is a list of software and hardware weakness types. Each vulnerability is associated with specific weaknesses. “Weakness Enumeration” is one of the fields in the description of a given vulnerability in the CVE repository. In this field a list of all the weaknesses related to a specific vulnerability is provided, in the form of CWE-IDs.

Additionally, the description of every weakness type includes a field called "Related Attack Patterns", presenting the attack patterns used for the exploitation of the corresponding weakness, in the form of CAPEC-IDs. In this way, it is possible to trace a list of CAPEC-IDs from a single CVE-ID.

The ODE provides several elements to model these. The *Cwe* element models the weakness as defined in the CWE list, the *Capec* element models the CAPEC entry (i.e., the attack itself), and the generic *Attack* element, which records information about the nature and consequences of the attack (e.g. time needed, difficulty of the attack, detectability, financial impact etc). The *Cwe* and *Capec* elements provide further information from their respective databases.

Each of the discovered attacks is characterized by a level of severity. Among the information provided for each of the attack entries in the CAPEC repository of known attacks, there is a field titled "typical-severity". This element is used to capture an overall typical average value for each type of attack with the understanding that it will not be completely accurate for all attacks, since it is affected by the overall system context. As a result, the typical severity of an attack is reflected on a qualitative scale of Very Low, Low, Medium, High, and Very High.

Since our intention is to make use of these values in the calculation of the Risk Priority Number, as described earlier, the textual values that are available in the CAPEC repository need to be mapped to numeric ones. Inspired by the Net Promoter Score[4] (NPS) key performance indicator, used in Customer Loyalty Management[5] (CLM), we need to map the severity qualitative values to a set of quantitative numbers that have an order. Such a mapping is depicted below:

[4] https://www.netpromoter.com/know/
[5] https://www.claruscommerce.com/blog/what-is-loyalty-management/

- Very Low → 1
- Low → 2
- Medium → 3
- High → 4
- Very high → 5

Importantly, an Attack also links to a *SecurityViolation*, which is essentially the relationship between a possible attack and an actual attack. SecurityViolations are modelled in the ODE as a subtype of *Failure*, allowing them to be incorporated as causes in failure models such as fault trees.

To analyse more sophisticated multi-stage attacks, attack trees may be constructed. Some more complex attacks may be conducted in sequence, with one opening the way for the next, and attack trees can better capture this chain of events.

The basis for the attack trees is often the hierarchical classification information contained in the CAPEC database, or more specifically the "canFollow" and "canPrecede" relationships between two attack patterns and which define a possible causal chain accordingly (see Figure 3 overleaf). However, they could also be based on expert knowledge of the system architecture in question and its vulnerabilities.

Attack trees are modelled in the ODE using fault trees. This allows them to be completely integrated with wider failure models and means they also benefit from all the same links as normal fault trees, including the ability to trigger *Actions* or to be triggered in turn by monitored *Events* at runtime.

4. Assessment of Risk

With the preceding steps complete, it is possible to arrive at an assessment of the risk posed by a security attack. With likelihood and severity values derived from databases like CAPEC, it becomes possible to generate an RPN in the same fashion as in safety. This provides a comparative measure to assess and compare the risk posed by different security threats, thus allowing the designers to prioritise.

Note that the values in CAPEC, as mentioned earlier, are generalised. For a more accurate and meaningful risk assessment, consideration must also be given to the likely operating environment. Just as in safety, proper evaluation of risk involves assessing what the consequences may be if the security attack occurs in the worst possible scenario. An attack might be locally severe in the sense that it might disable a component or destroy data rather than merely interfere with its function, but of more importance is the propagated effect of that attack on the wider system context. Thus the same security attack applied to both a smartphone and the control system of a nuclear power plant would likely have very different severity ratings.

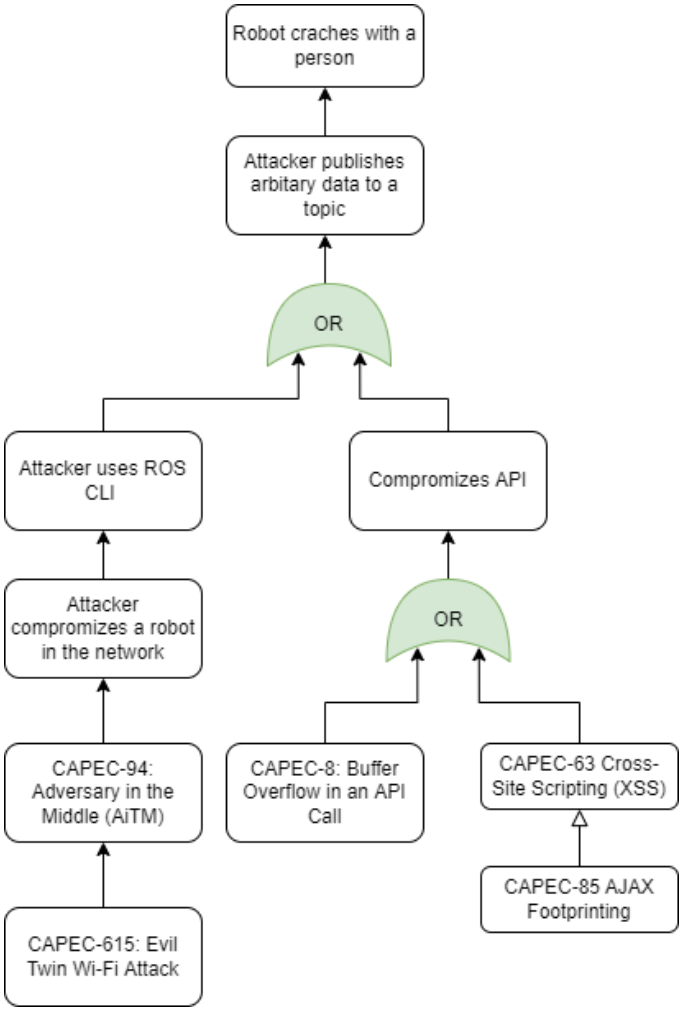


Figure 3: Example attack tree showing tracing of CAPEC attacks

5. Specification of Security Requirements

As with safety, the security risk evaluation process is used as an input into the specification of security requirements. Analogous to SILs, Security Levels (SLs) like those defined by IEC 62443-3-2:2020 may be assigned, which define the level of security that should be applied to prevent (or mitigate) security attacks — or alternatively, that the areas of the system subject to those requirements are vulnerable to attacks of that Security Level, and should thus be protected accordingly. The SL may also be 0, which means no specific requirement or security protection is necessary for the associated element.

The same ODE elements used for safety (e.g. *Standard* and *AssuranceLevel*) can apply here as well to security standards and SLs.

The CAPEC database also includes possible mitigations that can be applied to improve the resilience of the system and prevent, hinder, or mitigate the effects of any potential attack.

In particular, a protection mechanism may be specified, which is responsible for mitigating the attack. The protection mechanism may also form the basis of a security

requirement, modelled in the ODE as a *SecurityRequirement*, which specifies relevant security capabilities that must be provided.

Note however that while there may be equivalency between the abstract concepts involved, in practice the mitigations/protections mandated by a safety requirement tend to differ in nature from those mandated by security requirements. For example, to mitigate failures and improve safety, two common strategies are to increase robustness (i.e., replace the component with a more reliable one or one that does not suffer from the same failure) or increase redundancy (i.e., have backup components or have parallel components that must all fail for the functionality to be lost). The principle in both cases is to improve availability.

For security, however, availability is less of a priority; for instance, duplicating a component vulnerable to security attack just doubles the vulnerability rather than reducing it. Instead, the strategy is more often to impede potential attacks by introducing cryptography, adding diversity, and improving authentication techniques.

2.3 COMBINED DEPENDABILITY EVALUATION

Although the safety and security processes may take place at different stages in the design lifecycle (security notably needing a more mature design with concrete assets specified), and may execute steps in a different order, there are nevertheless many commonalities between them as, Figure 2 shows.

Taking inspiration from safety/security ontologies like those discussed in Section 1.3, we can draw general equivalencies between key concepts:

Table 2: Conceptual Equivalency

Concept	Safety	Security	SSCOF
A situation or event that poses a risk if it arises	Hazard	Attack (or Threat)	Hazardous Event (HE)
A potential cause of a risky situation/event	Failure or Fault	Vulnerability	HE Cause
A system element that is host to a potential cause	Item, Component, Function, System etc	Asset	HE Source
A measure taken to mitigate a risk	Safety Mechanism or Preventative Measure	Countermeasure or Mitigation	HE Mitigation
Requirement	Safety Requirement	Security Requirement	Dependability Requirement
Qualitative measure of requirement stringency	SIL, ASIL, DAL etc	Security Level (SL)	Assurance Level

Here we can see various common concepts and the terms used to describe them in the safety and security domains, as well as the generic term used in the SSCOF. It is also important to note that elements exist in the ODE for all of these generic terms:

- A Hazardous Event can be modelled either as a *Hazard* or as a *Failure* (or both), which also includes *SecurityViolation*; ODE *Failures* are very flexible and can represent different types of failure (see **D4.2/D5.2** for more information).

- A HE Cause is a *Cause*, applicable both to safety (via fault trees) and security (via attack trees). *Causes* have links to *States*, *Events*, and *Actions* as well for runtime purposes.

- A HE Source is a system element. The ODE provides several of these to reflect different levels of abstraction, but the most generic is *System*.

- A HE Mitigation is most likely also a system element, but it can be explicitly identified as a mitigating measure inside the *DependabilityRequirement*.

- A Dependability Requirement is readily modelled as an ODE *Dependability Requirement*, already the parent of both Safety and Security Requirements.

- Finally, the generic Assurance Level also already exists in the ODE as part of the Dependability::Doman subpackage, as described earlier.

This equivalency means that, to a significant degree, safety and security models are interchangeable or at least cross-referenceable within a given ODE model. For instance, a safety hazard may list a security vulnerability as a contributing cause in addition to failures; similarly, an attack tree could also incorporate failures as part of the causal logic leading to the attack goal, just as a fault tree could incorporate a security violation as part of its logic. Requirements for both safety and security can be modelled and a common assurance level applied.

This equivalency — along with translation between domain-specific scales, like that described earlier — makes it possible to perform a risk assessment that takes into account both safety and security. Both hazards and attacks may have severity values, while causes such as failures and vulnerabilities can both have likelihood/probability scores. Difficulty (in security) and detectability (in safety) can also form a common third factor.

By combining these, we can arrive at a generic Risk Priority Number that becomes comparable across both domains (indeed, it could itself be generated using a combination of safety and security values). This obviously requires careful thought to ensure that the values are indeed comparable and judged on the same scale, e.g. that a hazard with a severity of 5 is as severe as an attack with a severity of 5, but this is a necessary process in any case since we must ensure that two hazards or two attacks with the same score are comparably severe (or likely, or detectable etc).

By the same token, safety analyses such as FMEAs or FTAs or indeed security analysis via attack trees may include elements from both domains. While it may be difficult to combine detailed quantitative failure data (e.g. failure rates, MTTF) with qualitative security scores, earlier analyses that only deal with qualitative measures should be

comparable, meaning that we can produce a kind of joint safety/security FMEA, termed an "FMVEA" (Failure Modes, Vulnerabilities, & Effects Analysis). Note that the ODE already supports this, since security violations are just another type of failure as far as its FMEAs are concerned.

These initial qualitative models can then be incorporated into later quantitative analysis models as well. The FMVEA may form the basis for later generation of probabilistic models such as Markov models or Bayesian models, assuming such probabilistic information can be derived for the safety & security events in question. Dynamic models such as dynamic fault trees or state machines can also incorporate them to define sequences and state transitions; in this way, for example, both safety failures and security attacks could be used to trigger a change of state (e.g. moving the system to a higher alertness level that requires stricter authentication, or simply changing operational mode, such as the robot returning to base and shutting down rather than risk further damage/attack).

Section 3 demonstrates this in practice by performing a kind of FMVEA for a drone-based use case example.

2.4 PREPARATION FOR GENERATION OF RUNTIME EDDI

As discussed in the introduction, SESAME is not limited solely to design-time activities. Indeed, the goal is to be able to support dependability activities that span both design and runtime via the EDDI concept.

With that goal in mind, we can also see how the equivalencies identified above can be used as input into the generation of generic specifications for safety/security runtime artefacts, such as diagnostics and event monitoring. Although the nature of the concrete instances of these artefacts is dependent on the requirements of the platform and domain, the information required to create either a safety or security EDDI contains many commonalities.

Thus while a security intrusion detection system (IDS) may require very different technologies and implementation from a safety-specific fault diagnostic system based on e.g. Bayesian networks, both require similar inputs:

- Identification of which assets/components need to be monitored;
- Specification of which events may occur and how they may be detected;
- Relating these detected symptoms to probable underlying causes (whether a failure or a security violation);
- Specification of Actions to take in response to any detected/diagnosed Hazardous Event (perhaps accompanied by a particular confidence level, if e.g. assessed by an ML component).

A detailed description of how these runtime artefacts are generated is out of the scope of this document, being more the domain of WP7 and its associated deliverables (see

D7.1: Runtime Safety/Security Concept and **D7.2: Tools for Generation of Runtime EDDIs**), but it is nevertheless worthwhile noting that these necessary prerequisites are equally applicable to both safety and security artefacts at design time.

For instance, the ODE provides elements for both *Events* and *EventMonitors* (including various safety- or security-specific subtypes). An *EventMonitor* is generic and refers to a particular component/asset as the source of its *Event*. The associated *Event* may in turn have a specified trigger condition which refers to variables originating from those sensors (or other sources of perception/detection) regardless of whether they are symptoms of failures or security violations.

In this way, we can specify clear-cut cases, e.g. an *Event* that may be triggered if a safety-critical component is overheating or an *Event* that may be triggered by an IDS if a particular security violation is detected. However, we can also specify *Events* that may be symptoms of either safety or security issues; for example, an Event may be triggered by network communications failure, the condition for which may be unacceptable delays in packet delivery, and this *Event* may be the symptom of a failure (e.g. a fault in the network router or, for wireless, in the WiFi radio) or the symptom of a security attack (e.g. a denial of service attack on the network).

This makes it possible to generate joint safety/security EDDIs that use probabilistic models like Bayesian networks or Markov chains or logical models like combined fault/attack trees attempt to perform diagnosis to determine whether the cause of the detected Hazardous Event is either a security or safety issue and react accordingly. If the goal is mitigation of symptoms rather than diagnosis, then it may not even matter whether the cause is a security attack or a failure; what matters is the host platform's response, such as an emergency landing or return-to-base in the case of a drone.

3. CASE STUDY DESCRIPTION

To illustrate the application of the SESAME safety/security co-engineering framework (SSCOF), we apply it to a drone-based case study based on elements of both the KIOS/Cyprus Civil Defence use case (Power Infrastructure Inspection with Drones) and the DomaineKox/LuxSense/Aero41 use case (Viticulture Pest Control with Drones). In this section, we describe the case study system before proceeding to analyse it in Section 4.

However, please note that for the purposes of illustration, some details have been changed from those use cases and this example is not (yet) applied to the real-world systems of those use cases.

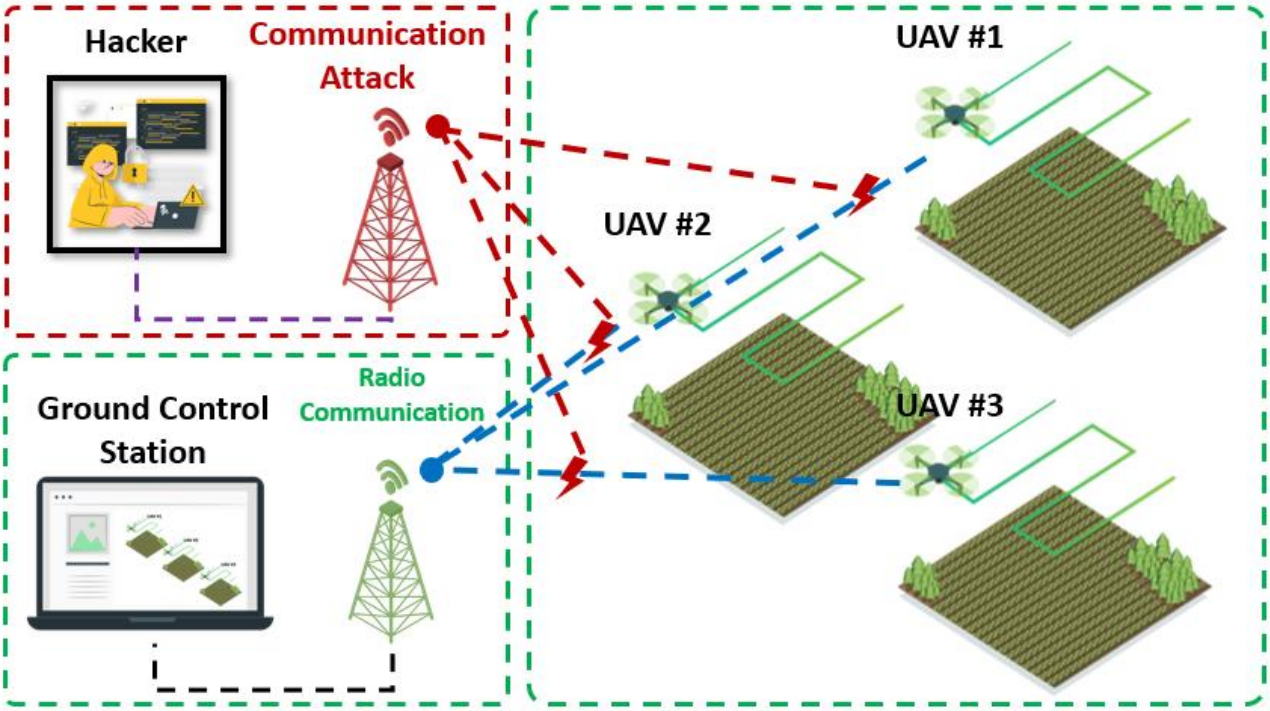


Figure 4: Overview of case study

Figure 4 above provides an overview of the case study. One or more drones are performing tasks — in this case, spraying crops with pesticides — while in communication with a ground station. The ground station is responsible for informing the drones of human presence in the fields and recalling the affected drones back to base. The drones may also have onboard person-detection capabilities as a fallback, but due to the difficulty of distinguishing a person amongst the foliage, this is not regarded as robust enough to rely on by itself.

The Hazardous Event being analysed in this case is the spraying of a human with harmful chemicals, which as we will see could be caused both by safety failures or by a security attack. This means that either the drones failed to receive the warning about human presence and then failed to detect the person themselves, or they received the warning but for some reason failed to act on it correctly.

The analysis conducted here will be qualitative in nature and take place as a form of FMVEA (Failure Modes, Vulnerabilities, & Effects Analysis). This allows the

equivalencies between safety and security in the SSCOF approach to be better illustrated than if quantitative analysis using probabilistic data was used.

Though here we focus primarily on design-time EDDI analysis, we also note in the next section the relevant aspects that could be used as input into the generation of runtime EDDIs.

3.1 SAFETY DIMENSION

There are two major system elements in this example: the UAVs (which are assumed to be identical models of drone) and the ground control station or GCS. To keep the scope of the example analysis manageable, we will focus primarily on those aspects of the two subsystems which have direct bearing on the hazardous event being investigated (namely, the spraying of a hapless person with a faceful of pesticide).

3.1.1 UAV Safety

To explore the potential failure modes of the UAVs, a very basic fault tree was constructed, shown below.

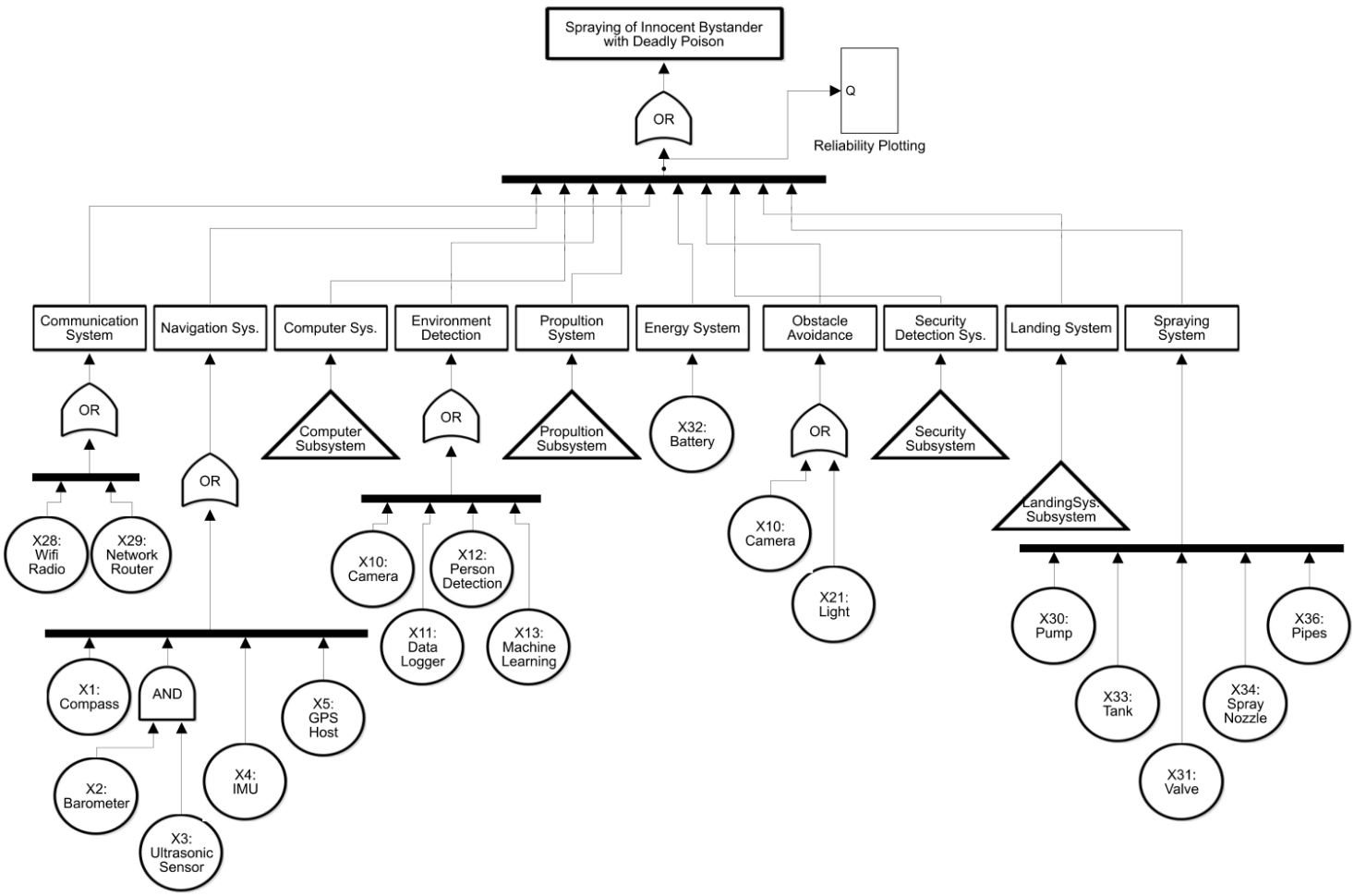


Figure 5: Overall fault tree for a drone

This generic fault tree encompasses many generic failures of the drone, from crashing (e.g. due to loss of propulsion) to task error (e.g. straying off course due to navigational failure) and more, but the one we are interested in is Spraying of Innocent Bystander With Deadly Poison. For this hazard, certain subsystems are less relevant, e.g. the propulsion and energy (battery) subsystems, and consequently the details of their failure have been elided from the tree for clarity purposes.

The key subsystems to consider are:

- The Communications system. If the drone does not receive warning of people in the vicinity, it may inadvertently spray them. The Communications system consists of a WiFi radio and a network router connected to the computer system.

- The Navigation system. The drone may receive the warning but fail to navigate away from the person. The Navigation system consists of a GPS receiver, compass, and various onboard sensors, e.g. accelerometers, attitude sensors, and other IMU (Inertial Measurement Unit) instruments.

- The Computer system. A software error may mean the drone either ignores the warning or reacts incorrectly (e.g. by not shutting off the spray while returning home). The computer system is treated as a single black box unit; we do not intend to investigate its subcomponents (e.g. RAM, CPU etc).

- The Environment Detection system, or specifically in this case the Person Detection subsystem. Although intended as a fallback mechanism, the drone may be able of detecting the person by itself; failure to do so in conjunction with a failure to receive or obey the warning means potentially spraying the person with chemicals. In this case the Person Detection subsystem involves a camera and an ML-based algorithm for detecting people running on the onboard computer.

- The Spraying system, which is responsible for delivering the pesticide to the crops. This consists of a tank, a valve, the spray nozzle, a pump, and the pipes between them.

3.1.2 Ground Control Station

The Ground Station is comparatively simple by contrast. It consists of a Communications system (WiFi router + radio), a Computer system (which processes data from the drones and issues instructions), and finally a User Console. The User Console allows the user to interact with both the GCS and to issue manual instructions to the drones. It is also how the warning of people in the fields is initially issued.

3.2 SECURITY DIMENSION

The key asset in this case study is the communications protocol in use. The Micro Aerial Vehicle Link (MAVLink) protocol is a communication protocol that is used for information exchange between UAVs and their corresponding GCS. UAVs typically communicate through a wireless connection with a GCS that controls their actions and receives updates about their status, as is the case in this example.

MAVLink is part of the ArduPilot system, a very popular open-source autopilot software. It is a lightweight messaging protocol for communicating with unmanned systems including drones. Its design pattern is a combination of publish-subscribe and point-to-point for data streams and mission/parameter communication respectively. Lorenz Meier released MAVLink in 2009 under the LGPL license. It serializes the exchanged state and command messages into a specific binary format (i.e., a stream of bytes) that is platform-independent.

The MAVLink protocol defines the structure of messages and their serialization at the application layer (i.e., the topmost layer of the network stack; see [21]). After the definition of the structure and serialization, the messages are forwarded to the transport (using a network protocol like TCP) and physical layers (e.g., digital transmission over radio), to be transmitted to the rest of the network. Since MAVLink is lightweight it supports different types of transport layers and physical mediums, including WiFi, Ethernet, TCP/IP, or serial telemetry low bandwidth channels operating at subGHz frequencies, namely 433 MHz, 868 MHz or 915 MHz [22].

The primary purpose of the communications in this example is the issuing of commands to the drones (e.g. areas to spray, orders to return home etc.) and the receipt of status updates in return (e.g. pesticide levels, battery status, current position, task completion %). Although drones may also send updates to each other, e.g. to prevent collision or to warn of detected persons or obstacles, there is no direct peer-to-peer communications; inter-drone messages are routed through the GCS.

4. CASE STUDY ANALYSIS

4.1 PRELIMINARY SAFETY ANALYSIS

4.1.1 Identification of Hazards

In effect we are already starting with the hazard we are interested in: inadvertent spraying of a person with pesticide. However, in truth this is a *hazardous event*, or in other terms, a hazardous scenario: a combination of hazard and scenario. It is worth therefore exploring the contributing hazards and scenarios in a little more detail through a process of decomposition.

Firstly, as mentioned, this hazardous event can only occur in scenarios where there is a person to be sprayed and the drone is still present and still spraying. There are multiple possible contributing factors involved in this case:

- The hazard "loss of communication with GCS" could lead to the drone never receiving the warning about the presence of the person.
- The hazard "failure to detect person in the vicinity" means the backup safety mechanism also fails (though due to the uncertainty in the person detection process, there is a probability of this happening even if the subsystem is functioning normally).
- The hazard "sprayer fails to cut off flow" could also lead to this event if the drone received the warning and was attempting to return home, but failed to cease spraying.
- The hazard "navigation error" could be a contributing factor in certain circumstances, but this also assumes that the spray continues even while returning to base, so we disregard this one as it is effectively a specialised case of the above.

Note that all of these hazards may, in other scenarios, have little to no effect on safety. A loss of communication is not in itself a serious problem, especially if it is temporary; however, in this case it leads to the drone's failure to receive a safety-critical warning. Similarly, if the sprayer fails to stop spraying, this is more of an inconvenience as long as a person is not standing underneath at the time.

The most serious is probably failure to detect a person in the vicinity, since this already implies a failure to receive advance warning. However, this only becomes critical if the drone is actively spraying at the time. If it is simply travelling from point to point, e.g. to return home to recharge or to travel to its destination field, then passing over a person is not a problem.

The above are all drone-based hazards. We must also consider possible hazards of the GCS:

- The hazard "GCS refuses user input" obviously leads to the inability to report the presence of humans in the vicinity, though again this may only be an

annoyance under other circumstances (and may well be fixed through the time-honoured remedy of "turning it off and on again").

- The hazard "loss of communication with drones" again means that the warning does not get passed on, at least in a timely fashion. Once more, however, this is only critical in the case that the loss of communication occurs in a scenario when a person is in the vicinity of the drone.

Further speculation, however, is the realm of investigation of causes.

4.1.2 Investigation of Causes

First we consider the GCS as the simpler of the two main system elements. When investigating the cause of a hazard, we consider only the hazard itself, not the scenario; thus, we are investigating the cause of e.g. loss of communication regardless of whether or not a person is present.

Since we will be using FMEAs later, for this purpose we illustrate a deductive approach instead using a simple qualitative FTA. In the case of the GCS, we have two hazards to investigate. In each case, the hazard becomes the top event of the fault tree and we work backwards to investigate the immediate causes at each stage until we reach the limits of the analysis (e.g. a system boundary).

Since fault trees take up a lot of space, for clarity we present them here in text format. OR and AND express Boolean logic gates, while BE refers to a basic event (i.e., a root failure not investigated further, such as a component fault). CC refers to a common cause (a basic failure external to the component that may also affect other systems/components). Finally, SV refers to a basic event that is a security violation.

```
Hazard: GCS Refuses User Input
OR
+------ AND: Physical damage to all user input devices
|       +---- BE: Physical damage to keyboard
|       +---- BE: Physical damage to mouse
+------ AND: All user input devices are disconnected
|       +---- BE: Keyboard disconnected
|       +---- BE: Mouse disconnected
+------ OR: Monitor is non-functional
|       +---- OR: Monitor has no video output
|       |       +---- BE: Video cable disconnected
|       |       +---- BE: Computer failure (GPU)
|       +---- BE: Monitor is damaged
|       +---- OR: No power to monitor
|               +---- BE: Power cable disconnected
|               +---- BE: Computer failure (PSU)
|               +---- CC: Power cut
+------ BE: Software error
+------ BE: Computer failure
```

Here we can see that there are five upper-level causes, three of which can be broken down further: damage to all input devices, disconnection of input devices, a non-functional monitor, software error, or some computer (hardware) failure that we do not investigate in further detail.

```
Hazard: GCS loses communication with drones
OR
+------ OR: Router failure
|       +---- BE: Router hardware failure
|       +---- BE: Router software failure
|       +---- OR: No power to router
|               +---- BE: Power cable disconnected
|               +---- CC: Power cut
+------ BE: WiFi radio failure
|       +---- OR: No power to radio
|               +---- BE: Power cable disconnected
|               +---- CC: Power cut
+------ CC: Radio interference
+------ BE: Drone comms system failure
+------ BE: Network congestion
+------ BE: Computer failure
+------ SV: DoS attack on network
```

The two primary causes here are router failure or WiFi failure (with power cut being a common cause external to the system itself). Radio interference is also another common cause, the cause of which is outside of the scope of the system but the effects of which must nonetheless be considered. Finally, here we also have a security violation (DoS attack).

Next we investigate the three drone hazards:

```
Hazard: Drone fails to detect person in vicinity
OR
+------ OR: Camera failure
|       +---- BE: Camera occluded
|       +---- OR: Camera has no power
|       |       +---- BE: Battery failure
|       |       +---- BE: Power cable disconnected
|       +---- BE: Camera hardware failure
|       +---- BE: Camera software error
+------ OR: Person detection failure
        +---- BE: Person not recognised when present
        +---- BE: Software error
        +---- OR: Computer failure
                +---- BE: Battery failure
                +---- BE: Hardware failure
```

Here, broadly speaking the failure is of either the camera or the person detection system. The basic event "Person not recognised when present" is difficult to diagnose further at this stage, since it represents a classification failure of the ML component.

```
Hazard: Unintentional spraying of pesticide
OR
+------ OR: Flow continues incorrectly
|       +---- AND: Incorrect flow from tank
|       |       +---- OR: Pump stays on incorrectly
|       |       |        +---- BE: Pump hardware failure
|       |       |        +---- BE: Software error
|       |       |        +---- BE: Short circuit on signal
|       |       +---- OR: Valve stuck open
|       |                +---- BE: Valve physically jammed
|       |                +---- BE: Control wire disconnected
|       |                +---- BE: Software error
|       |                +---- BE: Computer failure
|       +---- OR: Drone erroneously instructed to keep spraying
|               +---- BE: Incorrect command from GCS
|               +---- SV: Unauthorised command from attacker
+------ OR: Leakage of pesticide
        +---- BE: Leakage from tank
        +---- BE: Leakage from pipes
```

For this fault tree, the causes are primarily mechanical/hardware in nature and consist of either the pump failing to shut off, the valve remaining open, or leakage from the fluid system (e.g. the pesticide tank).

```
Hazard: Loss of communication with GCS
OR
+------ BE: Failure of GCS comms system
+------ OR: Router failure
|       +---- BE: Router hardware failure
|       +---- BE: Router software failure
|       +---- OR: No power to router
|               +---- BE: Power cable disconnected
|               +---- BE: Battery failure
+------ BE: WiFi radio failure
|       +---- OR: No power to radio
|               +---- BE: Power cable disconnected
|               +---- BE: Battery failure
+------ CC: Radio interference
+------ BE: Network congestion
+------ BE: Computer failure
+------ SV: DoS attack on network
```

Finally, here we look at the loss of communication. It is very similar to the equivalent GCS tree since both have similar implementations; and indeed the failure of the other

system is a root cause here too, leading to a cyclic/hierarchical fault tree if that basic event were to be expanded further.

Ordinarily the minimal cut sets — the smallest combinations of failures necessary to cause the top event — and the likelihood would also be considered here. However, we will defer these until the FMVEA rather than repeating the same information twice.

Note that there are a couple of security violations (DoS attack, Unauthorised command from attacker) as well as some common causes (e.g. radio interference). The latter, radio interference, means that the operating environment is experiencing sufficient EM radiation interference to inadvertently jam communications (or at least render them unreliable); this applies equally to all drones and the GCS, hence a common cause failure.

The security violations are not investigated here but rather considered further as part of the security analysis in Section 4.2. These violations may themselves be the head of their own attack trees. Since these are compatible with fault trees in the ODE, they would effectively form nested fault trees in this case.

4.1.3 Evaluation of Consequences

For evaluating the consequences, we choose the worst case scenario for each hazard — namely, spraying in the vicinity of a person when the hazard occurs — and assess its severity. To ensure equivalency across all measures, we are using a simple scale across the board:

1. Very low (minimal harm / inconvenience, e.g. wet clothing)
2. Low (minor harm possible, e.g. skin irritation)
3. Medium (moderate harm possible, e.g. skin blisters)
4. High (serious harm possible, e.g. eye damage, chemical burns)
5. Very high (life-threatening injuries possible)

The severity score for each of the hazards is listed below. For the purposes of this example, we are disregarding detectability (it is assumed the victim knows when they are being sprayed with pesticide).

Table 3: Severity Scores

Hazard	Severity Score
GCS Loss of communication	4
GCS Refuses user input	4
UAV Loss of communication	4
UAV Failure to detect nearby person(s)	4

UAV Unintentional spraying of pesticide	4

Since all of the hazards can lead to the same worst-case hazardous event (spraying someone with pesticide), they all have the same severity.

4.1.4 Assessment of Risk

In this step, we assess the risk by combining likelihood of causes with the severity of the consequences for each hazard. Typically this is performed via an FMEA, and so we present a simple FMEA below. Note that in this case the FMEA also contains the effects of multiple failures, since it is derived from the fault trees presented earlier.

Again, likelihood is a simple 5-point scale from very unlikely, unlikely, medium, likely, and very likely. The risk estimate (RPN) is therefore a number from 1 (lowest) to 25 (highest priority).

Note also that since communications loss is bi-directional — loss of GCS comms causes loss of drone comms, and vice versa, since each is an input into each other's fault tree — these are represented in the FMEA table as a combined failure (Communications Loss) to save space and improve clarity.

COMPONENT	FAILURE MODE	LIKELI-HOOD	EFFECT (HAZARD)	SEVE-RITY	RISK (L*S)
User interface peripherals	Physical damage to keyboard	2	AND: GCS refuses user input *Likelihood of both occurring assumed to be 1 (very unlikely)*	4	4
	Physical damage to mouse	2			
	Keyboard disconnected	3	AND: GCS refuses user input *Likelihood of both occurring assumed to be 1 (very unlikely)*	4	4
	Mouse disconnected	3			
GCS Monitor	Video cable disconnected	2	GCS refuses user input	4	8
	Monitor damaged	1	GCS refuses user input	4	4
	Monitor power cable disconnected	1	GCS refuses user input	4	4
GCS Power	Power cut	1	GCS refuses user input	4	4

COMPONENT	FAILURE MODE	LIKELI-HOOD	EFFECT (HAZARD)	SEVE-RITY	RISK (L*S)
	(common cause)		Communications loss	4	4
GCS Computer	Computer failure	1	GCS refuses user input	4	4
			Communications loss	4	4
	Software error	2	GCS refuses user input	4	8
GCS Router	Router hardware failure	1	Communications loss	4	4
	Router software (driver) error	1	Communications loss	4	4
	Power cable disconnected	1	Communications loss	4	4
	Network congestion	1	Communications loss	4	4
GCS WiFi	WiFI hardware failure	1	Communications loss	4	4
	WiFi software (driver) error	2	Communications loss	4	8
	Power cable disconnected	1	Communications loss	4	4
	DoS attack on network	(SV)	Communications loss	4	?
External	EM interference	1	GCS Communications loss	4	4
	(common cause)	1	UAV communication loss	4	4
UAV Camera	Camera occluded	2	UAV Failure to detect person	4	8
	Power cable disconnected	1	UAV Failure to detect person	4	4
	Hardware failure	1	UAV Failure to detect person	4	4
	Driver software error	1	UAV Failure to detect person	4	4

COMPONENT	FAILURE MODE	LIKELI-HOOD	EFFECT (HAZARD)	SEVE-RITY	RISK (L*S)
UAV Person Detection System	Failure to recognise a person when present	3	UAV Failure to detect person	4	12
	Software error	2	UAV Failure to detect person	4	8
UAV Computer	Computer failure	1	UAV Failure to detect person	4	4
	Software failure	1	Communications loss	4	4
		1	UAV Unintentional spraying	4	4
	Unauthorised command	(SV)	UAV Unintentional spraying	4	?
UAV Battery	Battery failure	2	UAV Failure to detect person	4	8
			Communications loss	4	8
UAV Pump	Hardware failure	1	UAV Unintentional spraying	4	4
	Short circuit on 'on' signal	1	UAV Unintentional spraying	4	4
UAV Valve	Jammed open	2	UAV Unintentional spraying	4	8
	Control wire came loose	2	UAV Unintentional spraying	4	8
UAV Tank	Leakage	1	UAV Unintentional spraying	4	4
UAV Pipes	Leakage	1	UAV Unintentional spraying	4	4
UAV Router	Router hardware failure	1	Communications loss	4	4
	Router software (driver) error	1	Communications loss	4	4

COMPONENT	FAILURE MODE	LIKELI-HOOD	EFFECT (HAZARD)	SEVE-RITY	RISK (L*S)
	Power cable disconnected	2	Communications loss	4	8
	Network congestion	1	Communications loss	4	4
GCS WiFi	WiFI hardware failure	1	Communications loss	4	4
	WiFi software (driver) error	2	Communications loss	4	8
	Power cable disconnected	2	Communications loss	4	8
	DoS attack on network	(SV)	Communications loss	4	?

The highest risk in the table comes from the failure to detect a person when present. However, as explained previously, this requires both the onboard person detection to fail AND that the drone never received warning of nearby people from the GCS, thereby reducing the likelihood of this in practice.

Security violations in the table have been marked and risk values left incomplete; this is because their values will be determined as part of the security analysis below.

4.2 PRELIMINARY SECURITY ANALYSIS

4.2.1 Identification of Assets

As mentioned in Section 3.2, the primary asset of interest here is the communications system of both the drones and the GCS and in particular the protocol being used, MAVLink. The other relevant assets are the computer systems in use, but as we are treating those as a black box we disregard them for the purposes of this example.

Although MAVLink is a protocol with widespread usage, it faces security problems due to lack of confidentiality and authentication mechanisms. As a result, when MAVLink is used, UAVs exchange state and control messages with the GCS through an unauthenticated, unencrypted channel. The nature of the communication, via a wireless channel, make it open to various attacks that could compromise data confidentiality, integrity, availability, and authenticity. Figure 6 brings together the potential attacks on the MAVLink protocol categorized under different security threats and security requirements.

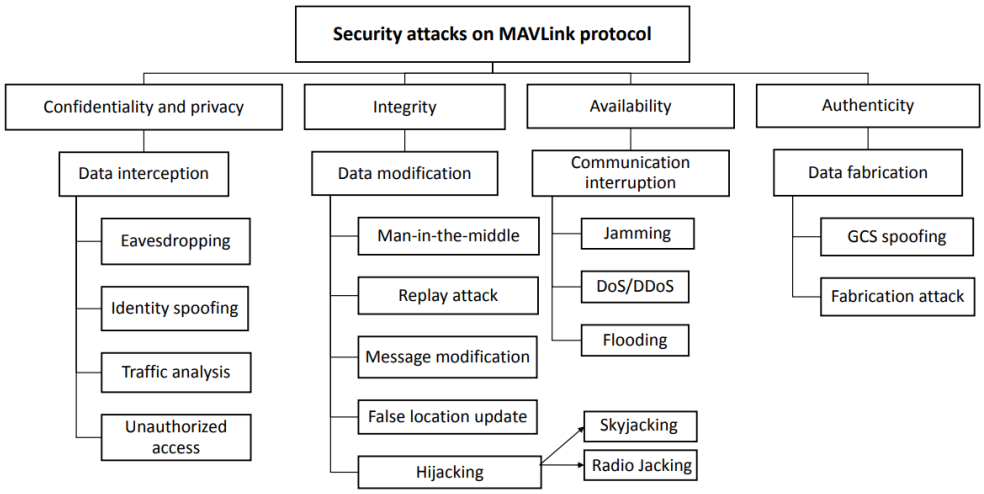


Figure 6: Security attacks on MAVLink protocol [22]

4.2.2 Identification of Vulnerabilities

The widespread popularity of MAVLink makes it a strong candidate for the communication that takes place between the UAVs and the GCS of our case study. In that case, the RVD#3316 vulnerability of the Robot Vulnerability Database (RVD) would be identified as one of the system vulnerabilities during the security assessment process. The whole information available in the corresponding RVD entry is depicted in Listing 1. As can be seen, in addition to the ID of the vulnerability, a lot more information is also available, including: title, description, related Common Weakness Enumeration (CWE), corresponding Common Vulnerabilities and Exposures (CVE), keywords, affected system, system vendor, Common Vulnerability Scoring System (CVSS) score, Robot Vulnerability Scoring System (RVSS) score, related references, and mitigation suggestions.

```
id: 3316
title: 'RVD#3316: No authentication in MAVLink protocol'
type: vulnerability
description:  The Micro Air Vehicle Link (MAVLink) protocol presents no authentication
  mechanism on its version 1.0 (nor authorization) which leads to a variety of attacks
  including identity spoofing, unauthorized access, PITM attacks and more.
  According to literature, version 2.0 optionally allows for package signing which
  mitigates this flaw. Another source mentions that MAVLink 2.0  only provides a simple
  authentication system based on HMAC. This implies that the flying system overall
  should add the same symmetric key into all devices of network. If not the case, this
may cause a security issue, that if one of the devices and its symmetric key are compro-
mised, the whole authentication system is not reliable.
cwe: CWE-306
cve: CVE-2020-10282
keywords:
- MAVLink
- v1.0
- v2.0
- PX4
- Ardupilot
system: "MAVLink: v1.0"
vendor: "PX4"
severity:
  rvss-score: 9.6
  rvss-vector: RVSS:1.0/AV:AN/AC:L/PR:N/UI:N/S:U/Y:T/C:H/I:H/A:H/H:U
  severity-description: critical
  cvss-score: 9.8
  cvss-vector: CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
```

```
links:
- https://arxiv.org/abs/1906.10641
- https://arxiv.org/abs/1905.00265
- https://ieeexplore.ieee.org/document/8425627
-
https://www.researchgate.net/publication/335973981_Assessing_and_Exploiting_Security_Vul
nerabilities_of_Unmanned_Aerial_Vehicles
- https://link.springer.com/chapter/10.1007/978-981-13-8406-6_66
- https://www.esat.kuleuven.be/cosic/publications/article-2667.pdf
- https://www.usenix.org/conference/usenixsecurity19/presentation/kim
- https://docs.google.com/document/d/1ETle6qQRcaNWAmpG2wz0oOpFKSF_bcTmYMQvtTGI8ns/edit
- https://docs.google.com/document/d/1upZ_KnEgK3Hk1j0DfSHl9AdKFMoSqkAQVeK8LsngvEU/edit
- https://docs.google.com/document/d/1XtbD0ORNkhZ8eKrsbSIZNLyg9sFRXMXbsR2mp37KbIg/edit
- https://github.com/PX4/Firmware/issues/13538#issuecomment-574281772
- https://github.com/rligocki/Diploma_thesis_px4
flaw:
  phase: unknown
  specificity: subject-specific
  architectural-location: platform code
  application: Flying vehicles and/or others using MAVLink protocol.
  subsystem: communication
  package: N/A
  languages: C, C++
  date-detected:
  detected-by:
  detected-by-method: testing
  date-reported: '2020-06-30'
  reported-by: "Victor Mayoral Vilches (Alias Robotics)"
  reported-by-relationship: security researcher
  issue: https://github.com/aliasrobotics/RVD/issues/3316
  reproducibility: always
  trace: N/A
  reproduction: N/A
  reproduction-image: N/A
exploitation:
  description: Not available
  exploitation-image: Not available
  exploitation-vector: Not available
  exploitation-recipe: ''
mitigation:
  description: MAVLink 2.0 includes signing capabilities which mitigate this issue. Sig-
natures seem to be optional for backwards compatibility and
https://arxiv.org/abs/1906.10641 confirms this matter. Proper mitigation should enforce
signatures.
  pull-request: N/A
  date-mitigation: null
```

Listing 1: RVD#3316 vulnerability

According to the vulnerability description, “The Micro Air Vehicle Link (MAVLink) protocol presents no authentication mechanism on its version 1.0 (nor authorization) which leads to a variety of attacks including identity spoofing, unauthorized access, PITM attacks and more…”. The vulnerability has a high severity score (rvss-score: 9.6 and cvss-score: 9.8) with the corresponding severity description to be characterized as critical. The related CVE-ID and CWE-ID are CVE-2020-10282 and CWE-306 respectively. Finally, the proposed mitigation includes the usage of signatures.

4.2.3 Identification of Attacks

The fact that the “RVD#3316: No authentication in MAVLink protocol” vulnerability is related to the weakness CWE-306 can lead us to potential attacks. CWE-306 weakness with the title “Missing authentication for critical Function” is associated with four potential attacks with CAPEC IDs 12, 166, 36, 62. The aforementioned attacks are presented as class instances in Figure 7 and Figure 8.

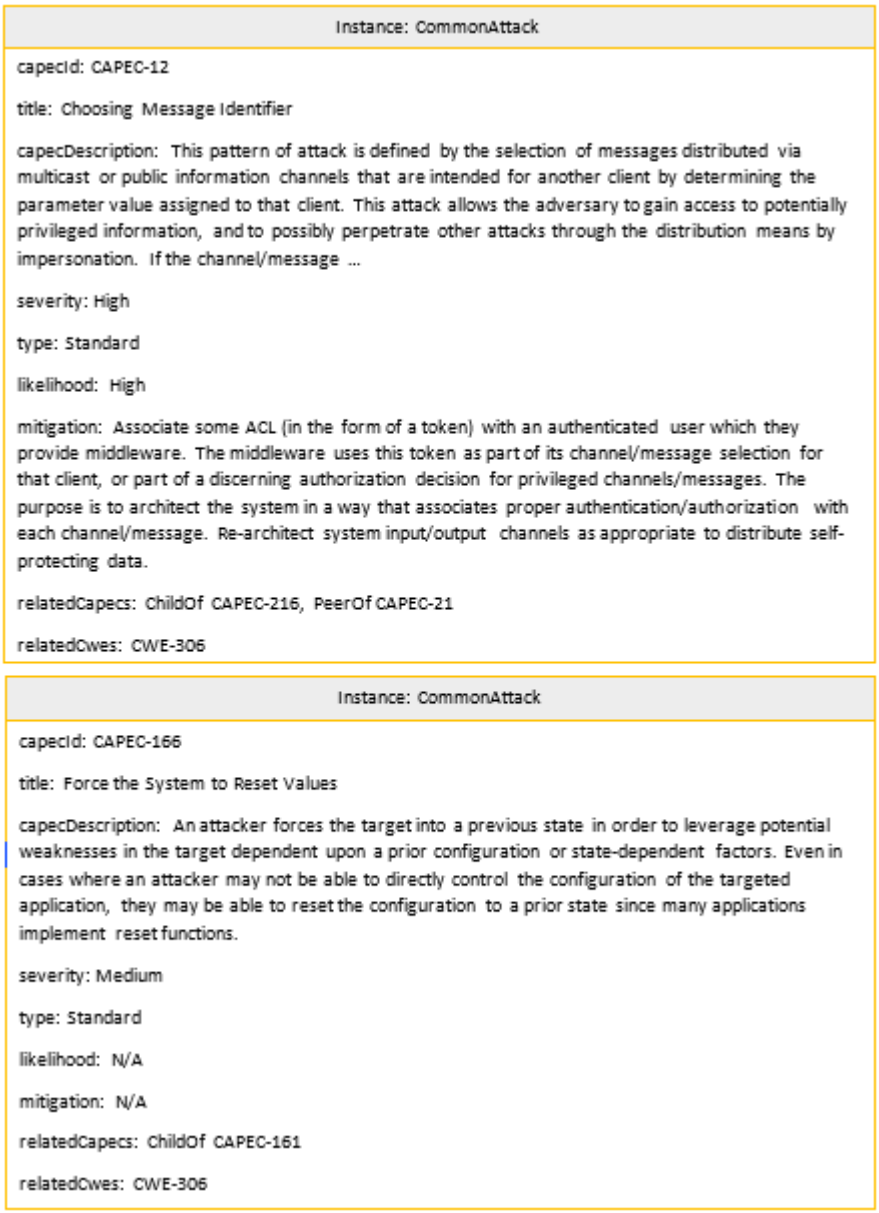
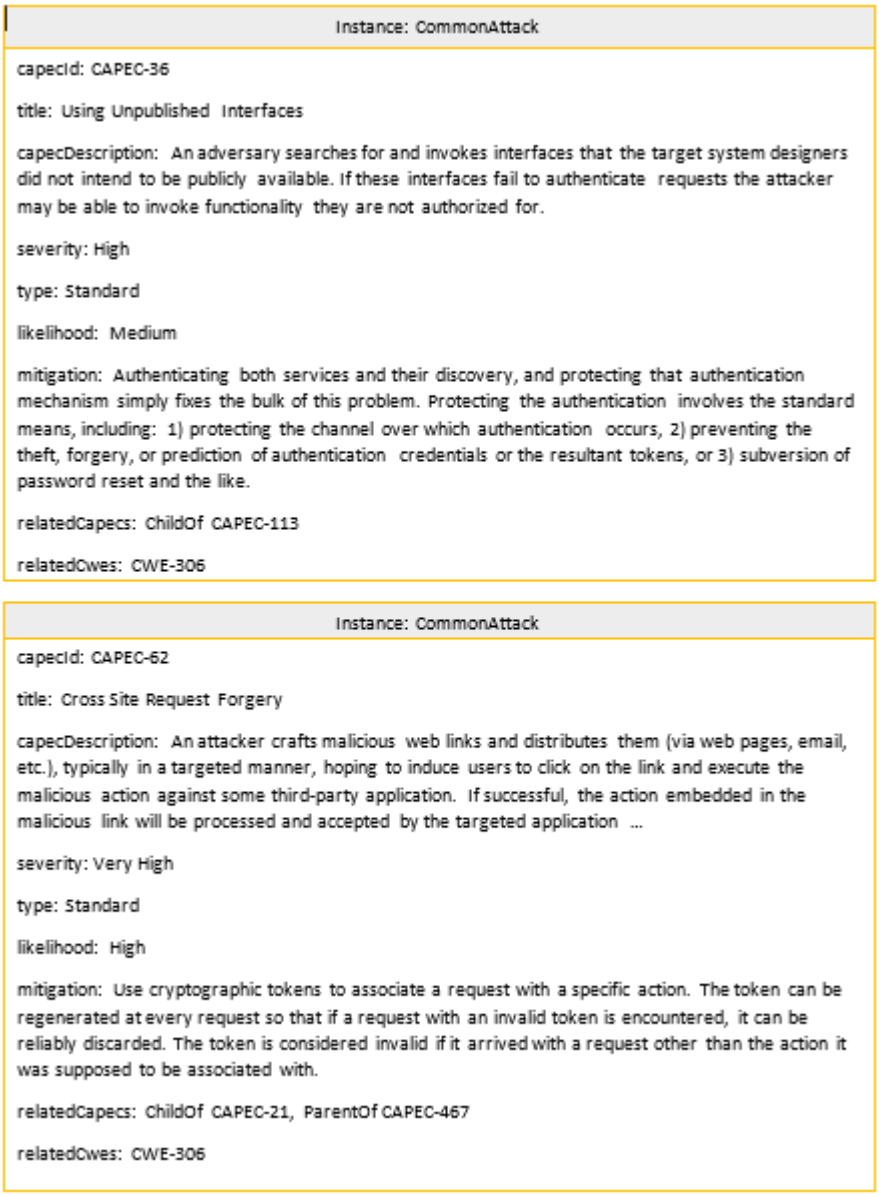


Figure 7: Potential attacks related to CWE-306

All these instances are linked with the CWE-306 weakness through the *relatedCwes* attribute. As it can be seen, the severity of one of the potential attacks is characterized as Very High, two are mentioned as High, and one as Medium. Mitigation actions are given for three of them. The attack titles are “Choosing Message Identifier”, “Force the System to Reset Values”, “Using Unpublished Interfaces”, and “Cross Site Request Forgery”.



Instance: CommonAttack

capecId: CAPEC-36

title: Using Unpublished Interfaces

capecDescription: An adversary searches for and invokes interfaces that the target system designers did not intend to be publicly available. If these interfaces fail to authenticate requests the attacker may be able to invoke functionality they are not authorized for.

severity: High

type: Standard

likelihood: Medium

mitigation: Authenticating both services and their discovery, and protecting that authentication mechanism simply fixes the bulk of this problem. Protecting the authentication involves the standard means, including: 1) protecting the channel over which authentication occurs, 2) preventing the theft, forgery, or prediction of authentication credentials or the resultant tokens, or 3) subversion of password reset and the like.

relatedCapecs: ChildOf CAPEC-113

relatedCwes: CWE-306

Instance: CommonAttack

capecId: CAPEC-62

title: Cross Site Request Forgery

capecDescription: An attacker crafts malicious web links and distributes them (via web pages, email, etc.), typically in a targeted manner, hoping to induce users to click on the link and execute the malicious action against some third-party application. If successful, the action embedded in the malicious link will be processed and accepted by the targeted application ...

severity: Very High

type: Standard

likelihood: High

mitigation: Use cryptographic tokens to associate a request with a specific action. The token can be regenerated at every request so that if a request with an invalid token is encountered, it can be reliably discarded. The token is considered invalid if it arrived with a request other than the action it was supposed to be associated with.

relatedCapecs: ChildOf CAPEC-21, ParentOf CAPEC-467

relatedCwes: CWE-306

Figure 8: Potential attacks related to CWE-306 (part b)

According to the SESAME security methodology described in **D5.1**, the detected potential attacks of a given target system are placed in attack trees making use of attack tree templates. If there is one path from the leaves of such an attack tree template to its root that can be confirmed based on the results of the security assessment (=detected potential attacks), the attack tree template is adopted and presented as a description of a possible attack scenario targeting the system in question. The detected CAPEC-12 attack confirms such a path in the corresponding attack tree template, depicted in Figure 9.

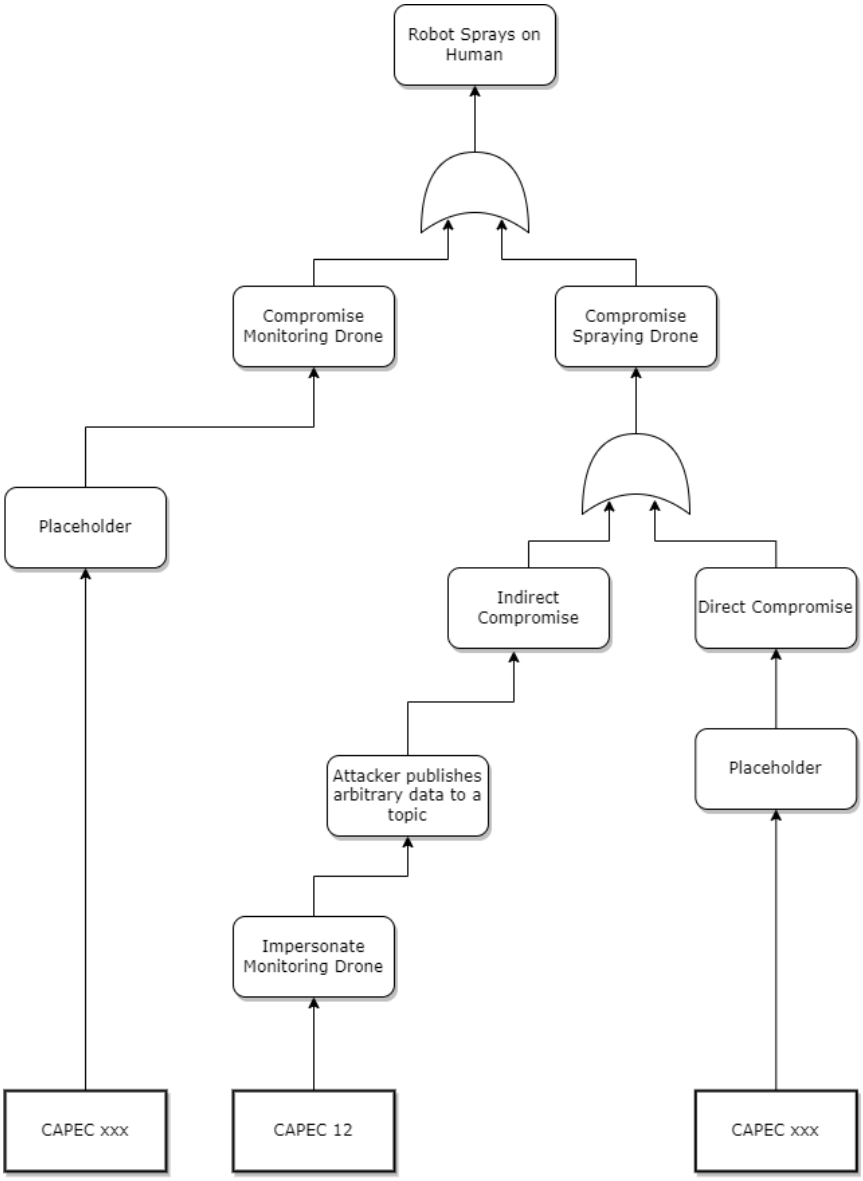


Figure 9: Attack tree (Fault tree) created due to CAPEC-12 potential attack

The presented attack tree corresponds to an attack scenario that could be conducted in the context of our case study. The root of the attack tree, which is the final goal of the attacker, is the Spraying Robot to continue the spraying while a human is detected in the close proximity. The sequence of steps from the CAPEC-12 attack towards the final goal of the attacker, which is the leaves-to-root path that was confirmed, is as follows:

1. CAPEC-12 - The attacker discovers a public information channel and sniffs the exchanged packets. This channel is the wireless communication with the spraying drone.

2. Due to the MAVLink protocol and the lack of authentication, the attacker is able to determine which packets are sent from the monitoring drone regarding the proximity of humans, and reproduce them with arbitrary values.

3. In that way, the attacker compromises the spraying drone by feeding it with wrong information.

4. The spraying drone, ignores that a human is present in the field and continues spraying.

Note that there are additional leaves-to-root paths in the depicted attack tree template. The presence of these paths shows that there are alternative ways for the attacker to reach their final goal, which may involve different vulnerabilities and potential attacks. However, as they are out of the scope of this system, there is no need for these additional paths to be described in detail, and hence there are no specific CAPEC IDs in the other leaves (CAPEC xxx) and a lot of the intermediate goals towards the final goals are described with the use of just one “Placeholder” cause box.

4.2.4 Safety & Security Risk Assessment

With the security analysis now complete, we can combine this with the earlier safety analysis. This can be used to produce more comprehensive fault trees that incorporate the attack trees as input, e.g.:

```
UAV: Unintentional spraying of pesticide
OR
+------ OR: Flow continues incorrectly
|       +---- AND: Incorrect flow from tank
|       |       +---- OR: Pump stays on incorrectly
|       |       |       +---- BE: Pump hardware failure
|       |       |       +---- BE: Software error
|       |       |       +---- BE: Short circuit on pump
|       |       +---- OR: Valve stuck open
|       |               +---- BE: Valve physically jammed
|       |               +---- BE: Control wire disconnected
|       |               +---- BE: Software error
|       |               +---- BE: Computer failure
|       +---- OR: Drone erroneously instructed to keep spraying
|               +---- BE: Incorrect command from GCS
|               +---- SV: OR: Unauthorised command from attacker
|                     +---- CAPEC-12: Message identifier
|                     +---- CAPEC-36: Unpublished interfaces
|                     +---- CAPEC-62: Cross-site forgery
|                     +---- CAPEC-166: Force value reset
+------ OR: Leakage of pesticide
        +---- BE: Leakage from tank
        +---- BE: Leakage from pipes
```

It can also be used to generate a combined FMVEA, an FMEA that combines both safety and security values. The portion of the FMVEA relevant to the UAV and featuring the "unauthorised command" security violation is shown below. The maximum likelihood value from the four possible CAPEC attacks is "high", i.e., 4.

COMPONENT	FAILURE MODE / SECURITY VIOLATION	LIKELI-HOOD	EFFECT (HAZARD)	SEVE-RITY	RISK (L*S)
UAV Computer	Computer failure	1	UAV Failure to detect person	4	4
	Software failure	1	Communications loss	4	4
		1	UAV Unintentional spraying	4	4
	Unauthorised command	4 (CAPEC)	UAV Unintentional spraying	4	16

Given the likelihood of the security attack combined with the severity of the possible effect, we can see that the resulting RPN indicates high priority.

The next step in both safety and security processes would be to use this information to inform the specification of the dependability requirements, which in turn may lead to the inclusion of safety mechanisms or countermeasures, or indeed changes to the design itself. In this case, for example, it may be that a different protocol is chosen in place of MAVLink, or that an additional encryption or authorisation protocol is added as a protective layer. From the safety side, a higher SIL is likely to be issued to the person detection and spraying systems, which may lead to e.g. a valve that closes as a failsafe in the absence of power or loss of control signal, and perhaps a second camera or alternative perception device to improve the accuracy and robustness of the person detection.

4.2.5 Preparation for Runtime Quantitative Dependability Monitoring

Although the scope of this deliverable is mainly restricted to design-time safety & security co-engineering, as explained earlier, it is also important to recognise that the information derived from this process may then go on to be used to generate runtime artefacts such as EDDIs. Therefore, while we are not including a full description of possible additional annotations that could be made in this regard, we will describe a few examples for the UAV specifically.

Firstly, from a security perspective, ideally the UAVs (and the GCS for that matter) would have onboard IDS. This would enable them to snoop the network traffic and attempt to detect any security violations in real-time. As we have two possible security violations, this means two symptoms:

- Detection of a command with incorrect (or absent) authorisation
- Detection of abnormal network congestion

These two Events would make use of the IDS as an event monitor, which in turn would be linked in the ODE model to the UAV's WiFi comms system as its source.

Mitigating Actions could then be defined so that the EDDI can either recommend them as responses or execute them directly if they are internal (e.g. to switch internal state in the EDDI itself). Here, potential responses could be:

- For the unauthorised command:
  - Disregard the command
  - Report the intrusion to the GCS
  - Perform emergency landing and shutdown if it is believed there is risk of further attack
- For the DoS attack:
  - To pause all activity until network activity returns to normal
  - Return to base
  - Perform an emergency landing and shutdown as above

Note that some degree of diagnosis is most likely necessary here for the second symptom. The Event "abnormally high network congestion" could also be caused by other causes, e.g. a software error, interference, or a hardware failure of some sort. In which case, while the Event provides a possible symptom, it would have to go through a diagnostic engine (e.g. a Bayesian network or a form of extended fault tree) to try to evaluate the actual cause of the symptom and thus what the correct response should be.

The advantage of having such capabilities at runtime is that risk can be managed more efficiently. While at design time we must assume the worst-case scenario, at runtime a more nuanced approach is possible on the basis of the additional information available. For example, we can use information from the GCS — such as warnings of people in the area — to increase confidence in any identification of nearby people from the person detection system (i.e., if we're told people are nearby, then that person we're seeing is probably not a false positive).

On a more prosaic level, environmental factors can also be taken into account in other areas. Detection of high wind, for instance, may lead to a cancellation of the task. Similarly, overheating or abnormal battery readings can be evaluated at runtime and used to generate a dynamic risk estimate, which can then be used to e.g. abort the task or perform an emergency landing. This type of behaviour is the aim of SafeDrones, a prototype of the runtime EDDI concept; more information can be found in **D7.1** and a report on SafeDrones experiments with SESAME partners has also been submitted as the following paper.

Project Publication
Koorosh Aslansefat, Panagiota Nikolaou, Martin Walker, Mohammed Naveed Akram, Ioannis Sorokos, Jan Reich, Panayiotis Kolios, Maria K. Michael, Theocharis Theocharides, Georgios Ellinas, Daniel Schneider and Yiannis Papadopoulos, (2022) SafeDrones: Real-Time Reliability Evaluation of UAVs using Executable Digital Dependable Identities. (Submitted for the 8th International Symposium on Model-Based Safety Assessment (IMBSA 2022)).

5. DISCUSSION OF RESULTS & CONCLUSION

5.1 CASE STUDY RESULTS

The case study presented in this report provides an example of safety and security engineering can be combined as part of the same framework. The example illustrates the parallel processes of identifying hazards/attacks, their causes, and evaluating their consequences, all of which can use common artefacts and result in a combined risk estimation. This estimate can then be used to inform dependability requirements that apply to either safety, security, or both.

In this particular example, we saw how one of the possible causes of a hazardous event identified during safety analysis could be traced back to a security violation. The security analysis then investigated this attack further, identifying the vulnerabilities that may give rise to the attack and leveraging open databases to derive likelihood and severity values.

The case study also provided examples of instances where the design-time EDDI information could be extended for use in subsequent generation of runtime EDDIs, e.g. for event monitoring, diagnosis, and response.

5.2 CONCLUSION

The advantages of a model-based approach for safety have been clear for many years now. However, security analysis is typically less formal and more ad-hoc; it may involve systematic processes but these are not generally tied into a formal model-based development and analysis process in the same way that safety can be.

Therefore, here we present a combined safety/security co-engineering framework based on the ODE, the metamodel that serves as basis for the EDDI concept. The ODE acts as a common ontology for both safety and security, establishing equivalencies between key concepts and allowing joint analyses to take place in which failures can be incorporated into security analysis and attacks into safety analysis. The combined results indicate the causes and consequences of hazardous events regardless of whether they originate from safety or security issues, and the same risk estimation applies to them all.

While developed for design-time usage, this framework paves the way for generation of combined safety/security artefacts at runtime as well. As mentioned, the common approach means that specification of requirements, event monitors, diagnostic engines, and responses/actions can take advantage of both safety and security information stored in the design-time models.

Although development continues, we hope this provides a good foundation for robust, combined dependability management of safety critical multi-robot systems.

6. REFERENCES

[1] ISO, "ISO/IEC 27005:2018 - Information technology — Security techniques — Information security risk management," 07 2018. [Online]. Available: https://www.iso.org/standard/75281.html.

[2] IEC, "IEC/TS 62443-1-1 — Industrial communication networks – Network and system security – Part 1-1: Terminology, concepts and models," 01 07 2009. [Online]. Available: https://standards.globalspec.com/std/1184691/iec-ts-62443-1-1.

[3] IEC, "IEC/TS 62443-3-2 — Security for industrial automation and control systems - Part 3-2: Security risk assessment for system design," 1 7 2009. [Online]. Available: https://webstore.iec.ch/publication/30727.

[4] NIST, "NIST SP 800-30: Guide for Conducting Risk Assessments," 16 01 2020. [Online]. Available: https://www.nist.gov/privacy-framework/nist-sp-800-30.

[5] M. Skoglund, F. Warg and B. Sangchoolie, "In Search of Synergies in a Multi-concern Development Lifecycle: Safety and Cybersecurity," in *International Conference on Computer Safety, Reliability, and Security* , Springer, 2018.

[6] ISO, "ISO 26262-1:2018 Road vehicles — Functional safety," 12 2018. [Online]. Available: https://www.iso.org/standard/68383.html.

[7] ISO/SAE, "ISO/SAE 21434:2021: Road Vehicles - Cybersecurity engineering," 08 2021. [Online]. Available: https://www.iso.org/standard/70918.html.

[8] J. Dobaj, Schmittner, K. M. C. and G. Macher, "Towards Integrated Quantitative Security and Safety Risk Assessment," in *International Conference on Computer Safety, Reliability, and Security*, Turku, Finland, 2019.

[9] X. Lyu, Y. Ding and S. H. Yang, "Safety and Security Risk Assessment in Cyber- physical Systems," *IET Cyber-Physical Systems: Theory & Applications,* vol. 4, no. 3, pp. 221-232, 2019.

[10] H. Martin, Z. Ma, C. Schmittner, B. Winkler, M. Krammer, D. Schneider, T. Amorim, G. Macher and C. Kreiner, "Combined Automotive Safety and Security Pattern Engineering Approach," *Reliability Engineering & System Safety,* vol. 198, no. 1, p. 106773, 2020.

[11] G. Macher, E. Armengaud, C. Kreiner, E. Brenner, C. Schmittner, Z. Ma, H. Martin and M. Krammer, "Integration of Security in the Development Lifecycle of Dependable Automotive CPS," in *Research Anthology on Artificial Intelligence Applications in Security*, Pennsylvania, United States, IGI Global, 2021, pp. 101-142.

[12] C. Schwarzl, N. Marko, H. Martin, V. Expósito Jiménez, J. Castella Triginer, B. Winkler and R. Bramberger, "Safety and Security Co-engineering for Highly Automated Vehicles," *E&I Elektrotechnik und Informationstechnik,* vol. 138, no. 7, pp. 469-479, 2021.

[13] T. Witte, R. Groner, A. Raschke, M. Tichy, I. Pekaric and M. Felderer, "Towards Model Co-Evolution Across Self-Adaptation Steps for Combined Safety and Security Analysis," EasyChair Pre-Print, 2022.

[14] J. Alanen, J. Linnosmaa, T. Malm, N. Papakonstantinou, T. Ahonen, E. Heikkilä and R. Tiusanen, "Hybrid ontology for safety, security, and dependability risk assessments and security threat analysis (STA) method for Industrial Control," *Systems. Reliability Engineering & System Safety,,* vol. 220, no. 1, p. 108270, 2022.

[15] T. Tommila and J. Alanen, "Conceptual Model for Safety Requirements Specification and Management in Nuclear Power Plants," VTT-VTT Technology, 2015.

[16] M. Gleirscher, N. Johnson, P. Karachristou, R. Calinescu, J. Law and J. Clark, "Challenges in the Safety-Security Co-Assurance of Collaborative Industrial Robots," in *The 21st Century Industrial Robot: When Tools Become Collaborators*, Springer, 2022.

[17] ISO, "ISO 12100:2010 — Safety of machinery: General principles for design; Risk assessment and risk reduction," 11 2010. [Online]. Available: https://www.iso.org/standard/51528.html.

[18] British Standards Institution, "BS EN 61882:2016 - Hazard and operability studies (HAZOP studies) Application Guide," British Standards Institution, 2016.

[19] US Department of Defense, "MIL-STD-1629A: Procedure for Performing a Failure Mode, Effects and Criticality Analysis," US Department of Defense, Washington DC, USA, 1980.

[20] W. E. Vesely, M. Stamatelatos, J. Dugan, J. Fragola, J. Minarick and J. Railsback, "Fault Tree Handbook with Aerospace Applications," NASA Office of Safety and Mission Assurance, USA, 2002.

[21] Cloudlfare Inc, "What is the OSI Model?," [Online]. Available: https://www.cloudflare.com/learning/ddos/glossary/open-systems-interconnection-model-osi/. [Accessed 29 06 2022].

[22] A. Koubaa, A. Allouch, M. Alajlan, Y. Javed, A. Belghith and M. Khalgui, "Micro Air Vehicle Link (MAVLink)

in a nutshell: A survey," *IEEE Access*, no. 7, 2019.