Project Number 101017258

D5.1 Security Analysis Concept and Methodology for EDDI development

Version 1.0
23 December 2021
Final

Public Distribution

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 draft with outline and first content	21 October 2021
0.2	First draft	7 December 2021
0.3	Ready for internal review	20 December 2021
0.4	Reviewed version	22 December 2021
1.0	Final QA version	23 December 2021

TABLE OF CONTENTS

TABLE OF FIGURES

EXECUTIVE SUMMARY

This deliverable describes the proposed security assessment concept and methodology that is going to be used in the SESAME project. The aim of identifying the security flaws of multi-robot systems has been proven to be challenging due to the increased connectivity of the robots, the fact that they operate in proximity with humans, and the low realization of risks that robotic systems face.

The state-of-the-art techniques, tools and repositories used for conducting security assessment are presented, investigating how they can be utilized for the definition of the SESAME security assessment concept and methodology. Security assessment in robotic systems is reviewed, trying to identify patterns in the used methodology.

The steps of SESAME security assessment are listed, and a proof-of-concept application of the methodology based on a high-level description of one of the use-case robotic systems is presented.

LIST OF ABBREVIATIONS

AiTB	Adversary in the Browser
AiTM	Adversary in the Middle
CAPEC	Common Attack Pattern Enumeration and Classification
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
CWE	Common Weakness Enumeration
XSS	Cross-Site Scripting
CAN	CVE Numbering Authorities
CPS	Cyber Physical Systems
DFDs	Data Flow Diagrams
DoS	Denial-of-Service
EM	ElectroMagnetic
EDDI	Executable Digital Dependability Identity
IPS	Intrusion Prevention System
MX	Mail Exchange
NVD	National Vulnerability Database
NGFW	next-generation
ODE	Open Dependability Exchange
OSSTM	Open Source Security Testing Methodology
PUF	Physically Unclonable Function
ROS	Robot Operating System
RVD	Robot Vulnerability Database
RTT	Round-Trip Time
SSI	Server Side Include
SACM	Structured Assurance Case Meta-Model
UTM	Unified Threat Management
URL	Uniform Resource Locator
UAV	Unmanned Aerial Vehicle
VDP	Velocity-Dependent Path

1. INTRODUCTION

1.1 OVERVIEW

Software and hardware vulnerabilities in the context of robotic systems are a fact with very serious potential consequences. Such vulnerabilities can trigger attacks that cause financial damage, exposure of sensitive data, loss of the customers' trust, negative effects to critical goods, even human injuries and losses. Robotic systems have an active part in many industry sectors such as automotive, energy (traditional and alternative), food, pharmaceutical, aerospace, etc. All industry sectors are extremely important for a nation, and that is the reason why they can become targets of adversaries.

The need for securing robotic systems is indisputable, however, the burden of materializing such a task should be carried out not only by the robot designers and operators but the standards creators, software developers, robot vendors, and security experts. Their goal is to make the process of exploiting robot vulnerabilities challenging and resource demanding

1.2 SECURITY CHALLENGE

Robotic systems of today face a whole new category of threats due to a number of newly adopted characteristics. They have been part of our daily life, integrated into cars, appliances, surveillance platforms, medical equipment, etc., operating in close proximity to humans. Moreover, they make use of software that does not incorporate security mechanisms for protection to malicious threats. Said robotic systems need to be connected to the outside world for monitoring and maintenance, creating APIs and introducing new attack surfaces. Finally, the administrators of such systems do not seem to be aware of the new risks due to the traditional industrial robot environment that used to be closed and trusted. Due to all these reasons, security assessment of robotic systems has become necessary but challenging.

The rest of the deliverable is structured as follows. In the Challenge of Security Assessment section, the definition of the problem of security threats in robotic systems is described, listing the main reasons why such systems become attack targets. Moreover, the state-of-the-art techniques of conducting security assessment are presented. Different kinds of attacks, protection mechanisms and the most common robot specific attacks are mentioned. The threat modelling process and different threat modelling models are described. Works found in the literature that present security assessment approaches on robotic systems are referenced. The last part of this section includes security knowledge repositories that are used in the proposed methodology. Section 3 presents the steps of the SESAME security methodology along with a proof-of-concept application. Finally, in section 4 we present our concluding remarks.

2. THE CHALLENGE OF SECURITY ASSESSMENT

2.1 DEFINING THE PROBLEM

One dimension of the cybersecurity problem regarding robotic systems is their growing pervasiveness to our daily life. Some of the robotic domains that blend in with society are autonomous vehicles, surveillance platforms, robot-assisted surgery, home service robots, industrial automation, providing surface for attacks with real-world consequences and risks. For example, attacks to industrial robot arms can cause injuries, compromised driverless vehicles can even bring death to passengers and pedestrians, while an attack to a personal robot making use of IoT devices, could potentially become a threat for privacy and cause identity theft [1].

Another aspect of the problem is the Robot Operating System (ROS), a robotic middle-ware standard that allows for the creation of heterogeneous clusters of robots by offering communication among its members [2]. The great adoption of this standard due to its strengths such as the active community and the offered code reuse not only by researchers but also by the industry reveals its drawbacks such as network security, authorization and resource permissions.

ROS as described in [1] was created based on a set of goals such as peer-to-peer communication, tools-based building, multi-lingual supporting, thin ideology, free and open source distribution. ROS is a meta-operating system for robots. Services provided by ROS include hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. One of its strongest features is the fact that it offers tools and independent libraries making its code reusable across multiple systems.

ROS is expected to play an increasingly important role in robotics, especially now that the digitization of industrial systems is taking place. Most probably, ROS will be used in real-life tasks and not so much in pure research-oriented scenarios. These new potentials of ROS bring to the surface its serious security issues. Said issues must be tackled before the creation and distribution of commercial products that are based on ROS. In this section we are presenting the most well-known vulnerabilities of ROS that is it proven that can be exploited.

A set of ROS drawbacks are presented in [1] including secure communication, access control, and process profiles. The XML-RPC protocol is used in ROS for the intercommunication of nodes. Messages are serialized with the help of libraries such as ROSTCP or ROSUDP and travel through IP sockets. However, this communication infrastructure lacks data encryption and integrity checking. As a result, ROS becomes an ideal target for attacks such as packet sniffing and man-in-the-middle. Content of packets can be altered and sensitive information can be stolen violating traditional security properties such as confidentiality and integrity. As far as access control is concerned, name-spaces are used for the definition of topics, services, names of nodes, and other locations in ROS' graph. No access control is provided regarding actions that a particular node is allowed to perform such as i) to what topics to publish or subscribe, ii) what parameters to read or write, iii) what services to invoke, or iv) which ROS API to make use of. Consequently, security problems such as compromised nodes, mode unavailability and registration to unallowed name-spaces may rise. The last-mentioned drawback of ROS is the fact that its packages are created from different sources, creating a large set of dependencies regarding vulnerabilities. Using MAC for ROS node processes could help protect from zero-day exploits.

Several vulnerabilities known to the authors of [3] are presented in their work, including communications in plain-text and unprotected TCP ports. ROS node-to-node communication includes exchange of messages in plain-text allowing for easy interpretation of the message form by a malicious user. This can result in spoofing of fake messages. Another vulnerability of ROS is the fact that its internal communication is based on TCP ports, to allow for connected robots to be placed in a distributed way. The downside of this approach is that TCP ports are exposed offering, at the same time, little authentication.

The need for security assessment of robotic systems is highlighted [4], mentioning some observations that the authors have made regarding industrial robots. The first observation is the increased connectivity of robotic systems that creates attack surfaces. Industrial robots used to work isolated inside controlled environments but due to their integration with ICT ecosystems, they are now connected to external networks and even to the Internet. This connectivity of industrial robots, as a way for controlling, monitoring and maintenance, is also included in ISO standards for robot systems integration [5]. There is a movement towards the creation of robot APIs that offer endpoints where user-defined requests end up and allow for the control of the robots. Administration and supervision of robots can be done even from portable devices such as smartphones [6].

Moreover, there is a trend to implement safety mechanisms creating programs and libraries moving away from hardware implementations of the past. This new kind of implementation, creates the severity of potential security incidents. If we combine this with the new generation of industrial robots that work closely to humans, we end up with a broader impact of security attacks on robotic systems that can easily threaten humans.

Another mentioned observation is the low realization of risks that the robotic systems are exposed to. Authors in [4] conducted a survey and some of the results showed that: i) default safety measures are changed due to the introduced limitation (60% of the survey respondents), ii) access control is not applied to robots and robot-controllers (28% of the survey respondents), and iii) security assessment is not utilized as a security tool (76% of the survey respondents).

It seems that the previous way that industrial robots used to offer their services, in environments that were closed and trusted, drove the robot manufactures to neglect necessary security mechanisms [7].

The security problem becomes denser in distributed MRSs. In such setups if one robot becomes the target of an attack, it can potentially affect other robots or even the whole system. The compromised robot can act as bad robot aka bad bot, in the sense that will perform malicious automated tasks at the adversary’s will, starting attacks to other system components including robots or robot-controllers. A representative example is mentioned [8], where 100 drones crashed into a building while a light show was taking place in Chongqing, China. The problem started from the mainframe control [9].

Security assessment of robotic systems, meaning the process of identifying, assessing and treating security risks to be compliant with the system security requirements, seems to be a necessity. Said process includes recognition of assets and security holes, identification of corresponding threats able to compromise the system assets, and discovery of protection means taking under consideration the calculated risk.

2.1.1 Definitions

In this section we list a few of the most common terms along with their definitions that we encounter in security assessment [10].

Threat: Any event with the potential to impact operations and assets through a system via unauthorized access, destruction, disclosure, or modification of information, and/or denial of service.

Vulnerability: A weakness in a system that could be exploited by a threat source.

Impact: The magnitude of harm that can be expected from an unauthorized disclosure, modification or destruction of information or system availability.

Likelihood: A weighted risk factor of the probability that a given threat source is capable of exploiting a given vulnerability or set of vulnerabilities.

Intrusion: “A security event, or a combination of multiple security events, that constitutes a security incident in which an intruder gains, or attempts to gain, access to a system or system resource without having authorization to do so.”[1]

Breach: “The loss of control, compromise, unauthorized disclosure, unauthorized acquisition, or any similar occurrence where: a person other than an authorized user accesses or potentially accesses personally identifiable information; or an authorized user accesses personally identifiable information for another than authorized purpose.”[2]

System Characterization: Identify applications, hardware, operating systems and endpoint devices.

Threat Source Identification: Identify sources of potential threats. Threats are categorized in:

- Human threats e.g. malware, data breaches
- Environmental threats e.g. power failures
- Natural threats e.g. storms, fires.

Vulnerability Identification: Identify exploitable weaknesses such as unpatched systems, weak security policies, poor password practises etc.

Control Analysis: Identify security controls such as firewalls, antivirus tools as well as alarms and locks.

[1] https://csrc.nist.gov/glossary/term/intrusion
[2] https://csrc.nist.gov/glossary/term/breach

Likelihood Determination: Assess the probability of a security breach based on the identified threats, vulnerabilities and security controls. There are three tiers:

- High
- Medium
- Low

Impact Analysis: Estimate the damage that will occur from a security breach in terms of value of hardware or data, reputation damage, loss of confidentiality, cost of repair.

Risk Determination: Quantify risk based on the likelihood of a threat combined with the vulnerability and the value of a specific asset.

2.1.2 Attacks

There is an unofficial methodology (can be considered a presumed model) that the attackers follow in order to achieve a successful attack that will lead them to their final goal regarding the manipulation of the target system. The first step of this methodology is the one called **reconnaissance** including gathering of information. Social engineering and the use of some automated tools, such as searchers, lead to the extraction of desirable information such as IP addresses or URLs (Uniform Resource Locator) [11] [12] [13] [14].

Continuing with the second step, gathered information is used for the **discovery** of hosts in a network and any available information such as operating system, active ports, services, and applications in a procedure is called scanning. A port scan will reveal ports of a network that are open or closed and services that are available. The discovered services are potentially having vulnerabilities that an attacker could take advantage of and start an **attack** [15]. The third and final step is the launching of the attack itself.

One way to categorize the attacks that target robotic systems is the one presented in [7]. The type of the attack (digital, physical) and its location (local, remote) create four combinations (local-digital, remote-digital, local-physical and remote-physical) where each attack can be assigned.

Local-digital attack examples are the installation of malware via a USB drive that can be attached physically on a robot. Another example is a DoS attack that is launched from a node hosted locally. There is a plethora of remote-digital attacks, since they can be conducted easier, including remote DoS, remote compromise of a system vulnerability, phishing attempts, malware sent via emails or remote services, etc. Local-physical attacks include theft and vandalism of the system in question or even hostile hardware installation. Finally, remote-physical attacks can be conducted on-site with the help of remote-controlled devices, such as drones.

2.1.2.1 Description of attacks

Due to the very nature of robotic systems, the fact that a large number of robotic nodes are interconnected, allows for a large number of different attacks. The most common ones will be presented in this section.

DoS/DDoS attack: A Denial-of-Service (DoS) attack aims to shut down a system or a network making its services unavailable to its users. This is performed by overloading the network with a great number of automatically produced service requests or by making it crash. Said requests are not to be served but their aim is to flood the target, consume its resources slowing it down or even making it inaccessible. A DoS attack makes use of only one attacker but if more of them cooperate then another type of attack is conducted, a Distributed Denial-of-Service (DDoS) attack. The victim is common and it is attacked by several locations simultaneously. The created number of attacking locations enables the attacker to execute a really troublemaking attack and hide its true location.

Spoofing attack: During a spoofing attack the attacker pretends to be an authorized user or device and aims to bypass the access control system and get access to the system in question, steal data or money, or spread malware. Email spoofing, URL spoofing, IP spoofing and DNS spoofing are some examples. Regarding robotic systems, such an attack could make a robot behave unexpectedly. For example, a GPS spoofing attack, can send fake GPS coordinates to the control unit of a drone and force it to change its original trajectory [16].

Man-in-the-middle attack: A man-in-the-middle attack is conducted when the attacker positions themselves between a user and an application, allowing them to manipulate the exchanged traffic. As we have already discussed, ROS, probably the most widespread operating system for robots, does not offer encrypted and secure communications. The information included in these communications can be control instructions, software updates or applications. Manipulating traffic in such ways could allow for insertion of malicious commands that could be executed by the target-robot, creating even safety issues.

Tampering attack: A tampering attack modifies parameters that are exchanged between a client and a server. In that way, different kind of data, such as credentials, permissions or quantity of ordered products can be manipulated. In the context of robotics, manipulated data can be calibration parameters of a robot or production logic [4].

Replay attack: A replay attack occurs when a secure network communication is intercepted and then it is delayed or resent to the receiver. The advantage of this type of attack is that there is no need for decryption of the message itself. The attack is performed by just resending the original message. The replay attack can be performed asynchronously even after the end of the original communication. A successful Replay attack allows adversaries to imitate authenticated users and try to take over their accounts.

Fault injection attack: A fault injection attack, regarding categorization, falls into the physical attacks. Its aim is to inject a fault in a system in order to bypass security mechanisms, change the normal behavior of the system or extract sensitive information. The injection of the fault has to be precise and it can be done using techniques such as voltage glitching, clock glitching, laser injection, electromagnetic (EM) injection. Such an attack can be performed via the hardware or the software.

Sybil attack: A Sybil attack is an attack observed in peer-to-peer networks, where a node pretends to be multiple regular nodes at the same time, creating fake identities. In a peer-to-peer network, that can lead to the manipulation of the whole network, by gaining the majority of the influence.

Jamming attack: A jamming attack is a type of DoS attack that is observed in wireless sensor networks (WSNs) where IEEE 802.15.4 standard is used. During such an attack, the attacker interferes traffic into the communication channel degrading the performance of the network and disrupting the communication of the regular users.

HW Backdoor attack: One way of accomplishing a Hardware backdoor attack is by physically code to hardware. Sometimes it can be done during the manufacturing process. Said code is triggered by an event and can be used for bypassing authentication mechanisms or encryption processes. Hardware backdoor’s impact is considered very severe since it cannot be detected by conventional security mechanisms.

Remote access Trojan attack: Such an attack includes a malware program that is downloaded at the target computer in disguise, as part of another program or as an email attachment. This malware makes use of a backdoor to take administrative rights to compromise the target. One common tactic then is to distribute RATs to other computers and create a botnet of compromised hosts.

Stealthy attack: An attack is characterized as stealthy if the attack process remains hidden. This can be achieved by manipulating the target system in such a way that its original behavior is not altered. Authors in [17] claim that these attacks present a regularity and can be detected.

SQL Injection attack: The target of such an attack is web applications and SQL databases that can be accessed via the Internet. Known vulnerabilities of said applications are explored to inject code in the form of SQL statements. These statements run on the database and can steal, alter or delete the database contents. Except for the loss of the actual data, the loss of customer trust should also be taken under consideration.

Cross-Site Scripting (XSS): XSS is another code injection attack that targets web applications. XSS includes the injection of malicious content to an application (JavaScript HTML), targeting cookies or other information regarding session. The usual aim is the redirection of the user to a malicious website, where the attacker can steal sensitive information such as usernames, passwords, bank credentials etc.

2.1.2.2 Protection mechanisms

The large variety of attacks dictates the existence of protection mechanisms that are able to protect a system against known attacks. In this section, we briefly present such state-of-the-art security solutions.

Network segmentation: According to this protection mechanism, a network is divided into a set of smaller and isolated subnetworks. This division has multiple advantages since it i) minimizes the attack surface exposing only the dedicated to services subnetwork, ii) makes harder for the attacker to locate resources that are scattered in many different subnetworks, iii) isolates architecture components that are considered vulnerable such as outdated ones, iv) makes it easier to enforce access policies to targeted subnetworks, and v) minimizes the damage in case of a successful attack. A very popular way to achieve network segmentation is by creating rules that dictate communications among hosts and services. Authors in [18] propose MilSeg, an architecture that segregates military networks in the SDN environment in order to minimize various attack vectors and spread of damage from the attacks.

Firewall: Firewalls is a security mechanism that monitors inbound and outbound network traffic. A set of rules allows for deciding which traffic is blocked or not. A common position for a firewall is the gateway of a system where the traffic that is exchanged with other networks is filtered, fencing the internal trusted network from the external untrusted ones. A firewall can be implemented as hardware or software. There are several types of firewall including proxy firewalls, stateful inspection firewalls, Unified threat management (UTM) firewalls, next-generation (NGFW) firewalls, threat-focused NGFW, and virtual firewall [19].

Intrusion Detection System (IDS): IDS can also be implemented as software or hardware and filters traffic trying to identify malicious packets. There are three main types of IDS: host based, network based and application based [20]. Host based IDSs are deployed on hosts and monitor incoming and outgoing traffic. Such hosts are systems that carry sensitive data, cannot be patched or have other reasons for extra security measures. Network based IDSs are placed in key points in networks, such as the gateway, and filter the traffic that is exchanged among the different devices of the network. Application based IDSs try to identify intrusions by filtering traffic on application specific protocols, such as SQL protocol. In all of the above, and in case an intrusion is detected, the administrator is alerted. Moreover, there are two different types of detection that is used by IDS: signature based and anomaly detection based. The former has an advantage regarding the known attacks, since they can be detected with great precision. The way it is done is by recognizing specific patterns in the headers or body of traffic packets. On the other hand, anomaly detection based IDSs are better at discovering unknown attacks. Said attacks alter the traffic making it different from the norm. Machine learning techniques, such as Tree classifiers, Bayesian Clustering, Deep Learning are used for the detection of unknown attacks.

Intrusion Prevention System (IPS): IPSs extend in a way IDSs since they incorporate an additional functionality, the prevention. They are in place to prevent detected intrusions and to do so, they need to be deployed in line. Actions that are taken include packet drop, traffic blockage, and connection reset.

Anti-Virus: Anti-Viruses are programs that are installed in hosts aiming to detect and remove malware. The technique that is used is the comparison of malware signatures with the installed software in the host. Additional types of anti-Viruses are able to check incoming to host documents such as mail, attachments, etc. trying to identify unusual properties due to known viruses [21].

Breach Detection System (BDS): BDSs are able to detect breaches or side-channel attacks that are not found by any other security mechanism, by focusing on the traffic that is exchanged inside a given network. A set of techniques are used for the detection of breaches such as traffic analysis, risk assessment, safe marked traffic, data policy understanding and violation reporting. There are three main way for the BDS to be deployed: i) out of band, where the traffic is mirrored to the BDS for scanning, ii) in line, where the IDS is deployed between the network in question and the WAN interface, and iii) deployment on endpoint machines.

Anti-phishing: Such a security solution protects from phishing attempts by detecting and blocking them in internet content. In that way, unauthorized access to sensitive data is avoided. Additional services include analysis of how data has been stolen, data recovery and protection from additional hacking.

Unified Threat Management (UTM): UTM can be a hardware or software solution that combines different security mechanisms at a single point. In that way, the end client does not need to be supplied with several security tools that each provides one security function.

Encryption: During encryption, a corresponding algorithm, called cipher, takes data, also called plaintext, as input and creates encoded data that is called ciphertext. The produced encrypted data can be decrypted only by authorized parties. There are two types of encryption, symmetric and asymmetric. The former uses the same key for encryption and decryption of the plain text. In case said key can be kept safe or there is no need to be transferred somewhere else, a symmetric type of encryption can be used. On the other hand, for encryption of communication between server and client, the asymmetric encryption is used, where the encryption is used with a key that can be revealed, public key, while the decryption is done with private keys that only authorized parties have.

Penetration Testing: Penetration testing is the process of launching simulated cyberattacks making use of strategies and tools that are meant for exploitation. The aim of penetration testing is the identification of flaws that could be the starting point of attacks. In that way, the security protection mechanisms and policies can be defined. Another way that penetration testing is useful is for the testing of the security policy effectiveness, the compliance to regulations, and the overall security awareness. The ultimate goal is to discover the weaknesses of a given system before they are discovered by adversaries that could take advantage of them.

In case of a network, a penetration testing process could reveal unused ports, firewall rules that need to be corrected or fine-tuned, and other security flaws. In case of web applications, buffer overflow, SQL injection, cross-site scripting, and other vulnerabilities may be revealed. Other types of attacks that can be performed during a penetration testing are those that try to steal or alter sensitive information from a system.

There are different strategies that are used, including external, internal, blind, double blind, and targeted testing. Types of tools that are used during such a process, include port scanners, vulnerability scanners, application scanners and web application assessment proxies.

2.1.2.3 *Robot-specific attacks*

There are some types of attacks that seem to be common as far as industrial robot systems are concerned. The frequency of occurrence of said types of attack is due to the existence of architectural commonalities and standards. Based on 4 desired robot properties, named sensor reading, control logic execution, movement precision, and human safety, authors in [22] created corresponding classes of attacks:

Altering the control-loop parameters. During such an attack, kinematics and configuration parameters are modified in a way that the robot's reference signal, position and speed, center of gravity and mass, and breaks are changed causing unexpected movements. The result of this attack could lead to faulty products.

Tampering calibration parameters. This attack can change the copy of the calibration parameters of a robot that is stored in the controller. The outcome can be very similar to the previous attack or if the controller checks the position and speed of the robot, the whole procedure will be put in a halt. In that way, an attack that tampers calibration parameters can have the effects of a DoS attack.

Tampering with the production logic. An attacker could take advantage of known vulnerabilities and change different program tasks including the whole process of product manufacturing.

Altering user-perceived robot state. In the case where a UI acts as an intermediate for the user of the robot to be informed about the robot's state, an attack that modifies said UI could be critical. During such an attack, the user is not aware about the true state of the robot and may make wrong decisions regarding safety issues, causing even human injuries.

Altering the robot state. Such an attack may alter the true state of a robot without the controller or the operator noticing it. This is due to the fact that some features, such as the mode the robot works on, can be changed via software. For example, an operator that believes that a robot works on automatic mode may decide to walk really close to it. At the same time an attacker changes the mode to manual and makes the robot move in a way that injures the operator.

We should mention here that all the aforementioned attacks can lead to safety implications due to the close proximity of the robots in contemporary robotic systems to the human operators.

2.2 STATE OF THE ART IN SECURITY ASSESSMENT

2.2.1 Threat modeling and security assessment

The selection of security measures is not just a collection of security technologies at random. A security designer must take under consideration the design of the whole system in question. Ideally, the security concerns should be tackled as soon as possible, incorporating the design of the security in the system design process.

Threat modeling is the process of identifying, communicating and understanding the threats of a given system, and then defining countermeasures to mitigate the effects of said threats that bring to the system [23]. Threat modeling investigates a system through the adversary's perspective and helps the designers to predict potential attacks, answering questions like what the system needs to protect and from whom. The benefits of threat model are of great value regardless of the stage of development of the system [24].

The main goals are to reveal critical issues and challenges during implementing security and to define and document the security requirements of a given system. The identification of assets, vulnerabilities, potential attacks, and mitigations are very important. Threat modeling is a structured process and includes steps such as system description, architecture dataflow, architecture components and their trust boundaries, identification of system entry points, threat analysis and determination of countermeasures.

The high-level threat modeling steps mentioned by the Open Web Application Security Project (OWASP) are the following [23]:

- Decompose the application. Information about the application is gathered and documented including dependencies, entry/exit points, assets, trust levels, and data flow.

- Determine and rank threats. Threats are categorized and STRIDE[3]/DREAD[4] threat models are used for ranking and risk estimation.

- Determine Countermeasures and Mitigation. Countermeasures are identified based on the categorization of threats with STRIDE and Application Security Frame (ASF).

- Complementing code review. The outcome of the threat modeling process allows code analysis to be focused on components with higher risk.

Moreover, the corresponding steps followed to perform threat analysis to ROS2 robotic systems are: [25]

- System description. The actors, assets and entry points are defined.

- Architecture dataflow diagram. The communications between all the components of the robot are represented in dataflow diagram.

- Robot application components and trust boundaries. Trusted and untrusted components are identified.

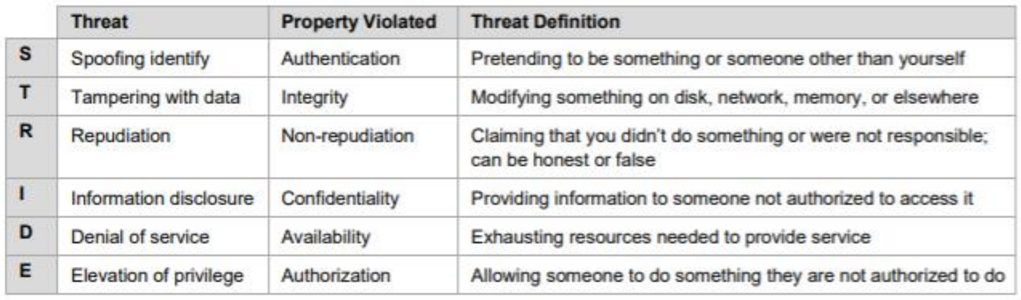
- Threat analysis and modeling. STRIDE and DREAD are used for the creation of a table with the whole produced information.

Currently there is a large number of threat modeling methods with different characteristics. Some of them are more abstract promoting granularity, while others are people-centric. Of course, they can be combined during a threat modeling process to produce a more comprehensive understanding of the potential threats. Some of these methods are described below:

- **STRIDE:** STRIDE is considered the most mature among the threat modeling methods. It was invented in 1999 and adopted by Microsoft in 2002, while variants have been created since then [26] [27] [28]. Initially, the modeling of the system in question is taking place, while data flow diagrams are used for

[3] Spoofing identity, Tampering with data, Repudiation threats, Information disclosure, Denial of service and Elevation of privileges (STRIDE)
[4] Damage Potential, Reproducibility, Exploitability, Affected Users, Discoverability (DREAD)

depiction of entities and boundaries. The next step is the identification of threats. The name of the method is an acronym meaning Spoofing identity, Tampering with data, Repudiation, Information disclosure, Denial of service, and Elevation of privilege, and is used as a mnemonic for threat discovery. Figure 1 includes the definition of the corresponding threats and the properties that each threat category violates. Available threat checklists and tables help this procedure [29].



	Threat	Property Violated	Threat Definition
S	Spoofing identify	Authentication	Pretending to be something or someone other than yourself
T	Tampering with data	Integrity	Modifying something on disk, network, memory, or elsewhere
R	Repudiation	Non-repudiation	Claiming that you didn't do something or were not responsible; can be honest or false
I	Information disclosure	Confidentiality	Providing information to someone not authorized to access it
D	Denial of service	Availability	Exhausting resources needed to provide service
E	Elevation of privilege	Authorization	Allowing someone to do something they are not authorized to do

Figure 1: Threat categories of STRIDE from [33]

According to [29], STRIDE has a low rate of false positives and a high rate of false negatives. DREAD, a similar method developed by Microsoft, stands for Damage potential, Reproducibility, Exploitability, Affected users, Discoverability. Values are assigned to the five categories, while an average value is calculated representing the overall risk of the system. However, DREAD is not used by Microsoft since 2008 due to inconsistent ratings.

- **PASTA**: PASTA stands for Process for Attack Simulation and Threat Analysis and is a threat model that incorporates a risk and impact analysis defining threats, application vulnerabilities and priorities for countermeasures. It includes seven stages of assessment, each one building on the top of the previous, presented in Figure 2.

  As it is described in [30], PASTA allows threat modeling to create security output that is taken under consideration in many aspects of a business, such as architecture, development, operations, even governance. In that way security is put at the center of the entire business.

- **LINDDUN**: LINDDUN stands for Linkability, Identifiability, Non-Repudiation, Detectability, Disclosure of Information, Unawareness, Non-Compliance, it works as a mnemonic and the focus of this method is privacy. The followed steps are depicted in Figure 3, below.

  As it can be seen, the first step includes a DFD that illustrates the system data flow, while the second step allows for mapping specific threats to the system components. The third step of the "problem space" part of the method includes the identification of scenarios where the discovered threats can occur. Steps 2 and 3 are conducted based on questionnaires that help with the identification of threats and scenarios. At the "solution space" part the mitigation specification takes place [31].

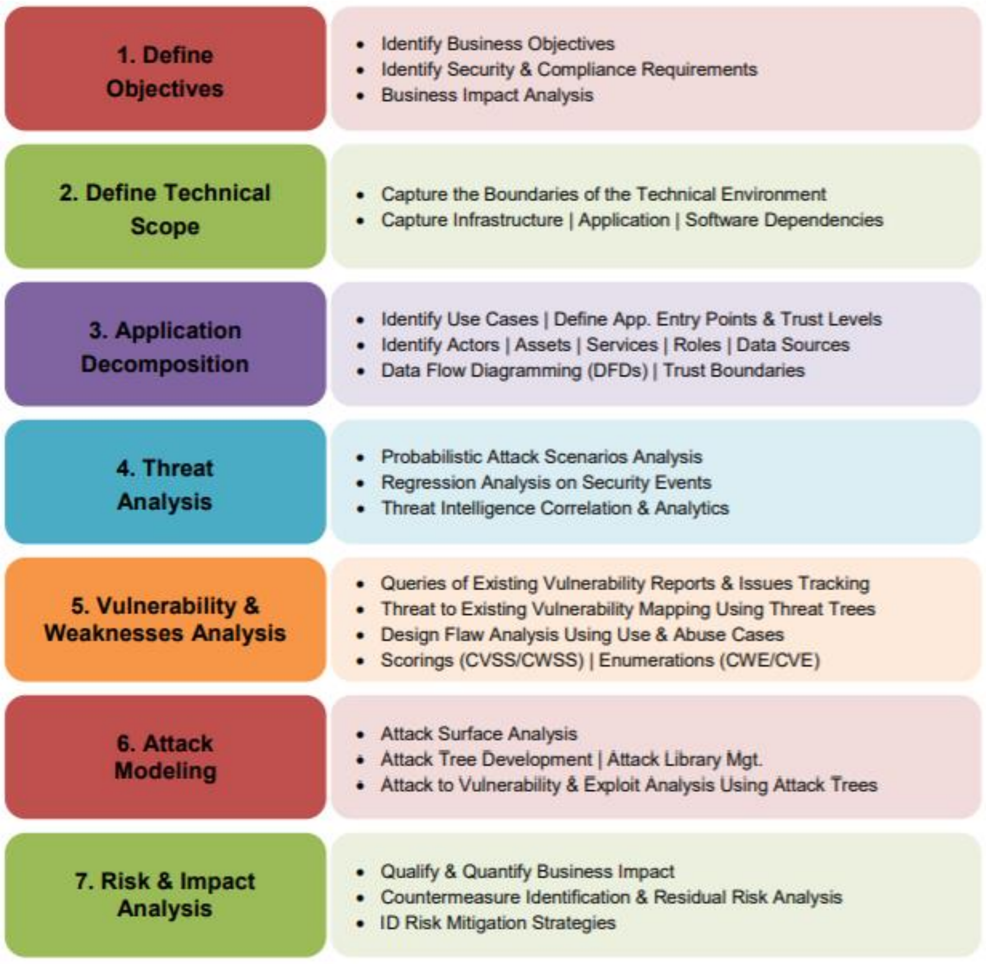

1. Define Objectives
Identify Business Objectives
Identify Security & Compliance Requirements
Business Impact Analysis
2. Define Technical Scope
Capture the Boundaries of the Technical Environment
Capture Infrastructure | Application | Software Dependencies
3. Application Decomposition
Identify Use Cases | Define App. Entry Points & Trust Levels
Identify Actors | Assets | Services | Roles | Data Sources
Data Flow Diagramming (DFDs) | Trust Boundaries
4. Threat Analysis
Probabilistic Attack Scenarios Analysis
Regression Analysis on Security Events
Threat Intelligence Correlation & Analytics
5. Vulnerability & Weaknesses Analysis
Queries of Existing Vulnerability Reports & Issues Tracking
Threat to Existing Vulnerability Mapping Using Threat Trees
Design Flaw Analysis Using Use & Abuse Cases
Scorings (CVSS/CWSS) | Enumerations (CWE/CVE)
6. Attack Modeling
Attack Surface Analysis
Attack Tree Development | Attack Library Mgt.
Attack to Vulnerability & Exploit Analysis Using Attack Trees
7. Risk & Impact Analysis
Qualify & Quantify Business Impact
Countermeasure Identification & Residual Risk Analysis
ID Risk Mitigation Strategies

Figure 2: Stages of assessment in PASTA from [33]

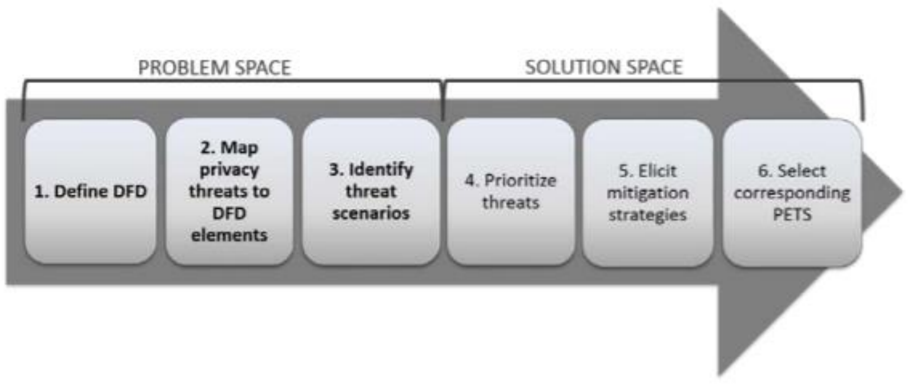

PROBLEM SPACE
SOLUTION SPACE
1. Define DFD
2. Map privacy threats to DFD elements
3. Identify threat scenarios
4. Prioritize threats
5. Elicit mitigation strategies
6. Select corresponding PETS

Figure 3: Process steps in LINDDUN from [31]

- **CVSS**: CVSS is an acronym meaning Common Vulnerability Scoring System. It is a framework that comprises information about software vulnerabilities, such as characteristics that are constant or evolve over time and their severity. CVSS consists of three metric groups, named Base, Temporal and Environmental. Based on the former, a score is produced ranging from 0 to 10. The other groups can modify this score [32]. The three metric groups are depicted in Figure 4, below.

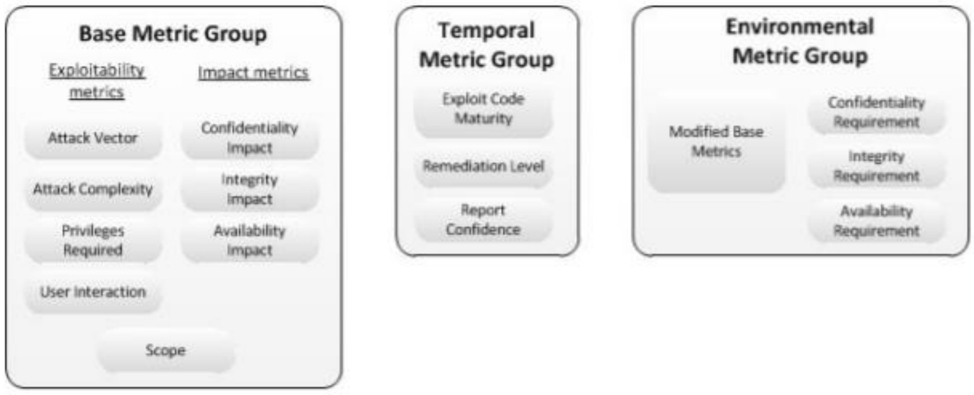


Figure 4: Metric groups of CVSS from [32]

The score produced by the Base metric group represents those characteristics of a vulnerability that are constant, having in mind the worst possible impact that said vulnerability could have among different deployments. Temporal metrics adjust the output of the Base group based on characteristics that may change such as availability. Finally, this assessment may also change based on the environmental metrics that are based on the very specific characteristics of a given environment.

There is a plethora of models that are available out there, each focusing on different aspects, depending on the involved placeholders, the available resources, the previous experience, the desirable outcome. Some additional models that are worthy to be mentioned are Persona non Grata, Security Cards, hTMM, Quantitative TMM, Trike, VAST Modelling, and OCTAVE [33].

Some worth mentioning, open-source threat modelling tools are:

- **Cairis:** Cairis [34] is an open-source web-based tool. The tool allows for creation of attackers' profiles. It identifies attack patterns and defines mitigation actions.

- **Microsoft Threat Modelling Tool:** Microsoft Threat Modelling Tool [35] uses the STRIDE methodology. Flow diagrams can be created, while threats and corresponding mitigations are offered. There is a focus on Azure and Windows services.

- **OWASP Threat Dragon:** OWASP Threat Dragon is an open-source web-based tool. Its output include flow diagrams, lists of potential threats and mitigations. The main advantage of this tool is its rule engine.

- **Threagile:** Threagile [36] is an open-sourced tool kit that is code-based. The output is available in many formats. It is YAML-based, which makes it easy for the threat model to be manipulated.

- **Tutamantic:** Tutamantic [37] is a flexible tool that allows for changes when the design of the system changes. There is a Beta version that is free. It uses taxonomies that are considered common, such as STRIDE, CWE and CAPEC, and its output is available in different consumable forms.

2.2.2 Security assessment in robotic systems

As we have already mentioned, robots are invading many aspects of our daily life including transportation, surveillance systems, home assistance, remote medical services, goods production, energy networks, etc. The fact that robots couple different types of sensors and actuators, human to machine interfaces, information processing and mobility, introduces new vulnerabilities that can be exploited and cause economic damage or even safety problems.

There are many works in the literature that present security analysis of different kinds of robotic systems, trying to identify cyber-attacks and their impacts. In this section we present some of these works.

Exploring attacks to robotic systems

Authors in [3] employ a cyber-physical honeypot that makes use of ROS and features sensors and actuators allowing for a new set of vulnerabilities and exploits to be discovered. The exploitation of the honeypot vulnerabilities was a contest challenge for the attendees of the DEF CON 20 conference. The conducted security exploits included false ROS messages injection. One of the learned lessons of this work is that the security of cyber-physical systems demands knowledge in cyber and physical security, and in robotics.

The work in [38] demonstrates hacking a modern automobile affecting systems such as digital dash, door locks, brakes, and engine control component. Additionally, it was shown in [39] that based on the Precision Immobilization Technique, the physical security of a car-like sensor node can be compromised in a relatively simple way.

Authors in [40] present an evaluation of DoS attacks in two different Unmanned Aerial Vehicles (UAVs), produced by three DoS attacks Tools. They perform a comparison between the vulnerabilities of the two UAVs and also introduce a tool, able to run inside ROS, for mitigating availability issues such as losing the control of the drone. Their methodology for conducting the attacks included the following steps: a) Establish a connection between the pilot and UAVs (AR.Drone and SOLO); b) Pilot sends legitimate commands to UAVs under normal conditions; c) Establish a connection between attacker and UAVs; d) The attacker makes reconnaissance attacks on UAVs; e) Attacker launches a DoS attack towards UAVs. The proposed tool calculates the Euclidean distance between the starting position (takeoff position) and the current position of the UAV and when it reaches the 10-meter limit, the base station emits a beep sound. According to their evaluation, the attacks resulted in lower average frame rate of the UAVs’ cameras (dropping from 30 to almost 5) and larger average network latency. Moreover, the use of a more powerful UAV showed that better hardware and software configuration is not a solution to said attacks. Distributed Denial of Service attacks could be utilized in future work. Additionally, the introduced solution for mitigating availability issues affects the efficiency of the UAVs.

A model to represent the performance of multi-robot systems introducing the Velocity-Dependent Path (VDP) that affects the workload completion times is introduced in [41]. In that way, critical execution paths and function nodes that determine the performance of an MRS are determined. Their focus is on cloud-robotic systems, where the computational tasks are migrated to the cloud to preserve task execution time and battery life of the robots. Moreover, they identify possible attack strategies and design three novel DoS attacks (Network Contention, Micro-architecture Contention, Direct Delay), where just one malicious node can compromise the entire cloud-robotic platform. The evaluation showed that during a Network contention attack: a) the Round-Trip Time (RTT) of each message is significantly increased. When the flooding rate is 25 Hz, RTT of the victim message becomes 7.32ms (normal RTT is 2.24ms), with maximum RTT 193ms; b) about 35% messages are dropped due to the network contention in the wireless router (flooding rate 20 Hz). A micro-architecture contention attack, where a malicious publisher floods dummy messages to subscriber nodes in the same machine, increases the CPU utilization of the subscriber nodes making them deprived of the CPU of other nodes. Finally, the performed Parameter manipulation attack increased the processing time. The focus of the paper is on ROS since this is still the most popular choice for products and companies. However, the newly introduced ROS2 could possible mitigate said attacks. As future work, the authors mention their intention to extend their evaluation to ROS2.

Security assessment methodologies

Authors in [42], present a structured methodology to conduct a security assessment over Pepper, a commercial human-shaped social robot. Pepper is designed to infer basic human emotions and react accordingly. It can be operated through ROS, one of the most widespread middleware in robotics, and is equipped with a number of sensors such as microphones, HD cameras, 3D depth and touch sensors. According to the presented methodology, security assessment is conducted in two phases, i) an automated one including a port scan and vulnerability scanning with OpenVAS and OWASP ZAP, and ii) a manual one including traffic analysis with Wireshark, brute force attack with Hydra and investigation of uncommon open ports. Said assessment revealed a number of security flaws that may enable credentials spoofing, stored data steal, hacking of connected devices. The authors propose countermeasures for every detected vulnerability and point out the general trend of shallowness regarding the security status of robots and IoT devices, especially those that are meant to interact with people.

A security assessment for the Franka Emika Panda is performed in [7]. The authors analyse potential attack surfaces along with possible impacts on safety-relevant parameters. Systematic penetration test is conducted based on the Open Source Security Testing Methodology (OSSTM) and the OWASP testing guide. Countermeasures are proposed for each of the found vulnerability to prevent corresponding cyber-attacks. Although security tools were used (Nmap, Wireshark, Nessus, Burp Suite Pro), they majority of the identified vulnerabilities was not pinpointed by said tools automatically. The findings revealed that the web application of the Franka Emika Panda was the attack surface with the most vulnerabilities. Moreover, the authors showed that a security attack can affect both human safety and manufacturing process.

An analysis of the security issues of Cyber Physical Systems (CPS) is presented in [43]. The authors make use of a three-layer architecture: perception, transition and application layer. Different security issues in each layer leads to different threats and the need for different security solutions. According to the presented analysis, it can be concluded that the most common security targets at the perception layer are the sensors and the actuators; data leakage, DoS, control or destruction at the transmission level; while privacy disclosure and unauthorized access at the application level. Although

CPSs have security needs of their own due to heterogeneous involved technologies, general IT security techniques could be used to some extent. Suggested CPS focused solution include Physically Unclonable Function (PUF) for the unique identification of heterogeneous connected devices, dealing with integrity and authenticity. On the other hand, PUF cannot be widely adopted since not all devices can implement PUF technology. Moreover, an Identity-Based encryption technique that uses small key sizes could be a solution for privacy issues, without burdening devices with limited resources. Other suggested solutions are a unified data processing standard that allows for data compression and data fusion techniques, and utilization of the cloud for computation processes.

Authors in [4] present an experimental security analysis of an industrial robot controller. They define an attacker model to express attackers of industrial robots based on their goals, their level of access to the system, and their capabilities. Regarding the former, four potential goals are mentioned: Production Outcome Altering, Physical Damage, Production Plant Halting and Unauthorized Access. Access to the system can be gained either from a network attacker, since controllers of robots are sometimes internet exposed or remotely accessible from vendors, or from a physical attacker, which can plug a device into the robot controller's openly accessible ports. As far as the attacker capabilities are concerned, attackers: a) can be insiders or general cybercriminals; b) must be able to become familiar with the structure of the target robot; and c) access just target firmware or both firmware and hardware (full-fledged deployment), in order to discover vulnerabilities and test exploits. Moreover, robot-specific attack classes are identified including Control Loop Alteration, User-perceived Robot State Alteration, Robot State Alteration, Production Logic Tampering, Calibration Parameters Tampering. In order to present the feasibility of said attacks a reference robot was used. The range of the attacks were limited due to costs and robot security and safety regulations. Additionally, only standard features of the reference robot were taken under consideration without considering optional equipment that potentially increases the attack surface.

2.2.3 Security knowledge repositories

There are many repositories, lists, directories that enclose information about vulnerabilities, weaknesses, bugs, etc. This section includes the description of those that are taking part in the SESAME security assessment process.

Common Vulnerabilities and Exposures (CVE). CVE [44] is actually a list of computer security flaws, cybersecurity vulnerabilities, and can be used for searching or incorporated into products and services for free. Each of these flaws is assigned an identifier called CVE-ID, which is used as a dependable way to uniquely recognise vulnerabilities. CVE-IDs are issued by CVE Numbering Authorities (CNA), group of IT vendors, security companies and research organizations. Well-known vendors that are considered CNAs are Adobe, Apple, Cisco, Linux, Google, HP, IBM, Microsoft, Mozilla, Oracle, and Red Hat. Reports of CVEs can be done by almost anyone, even a simple user of a product. In any case, when information about a vulnerability reaches a CNA, a CVE-ID is assigned and a description is created. The final step is the vulnerability to be posted on the CVE website. The sequence of steps in the lifecycle of a CVE record are depicted in Figure 5.

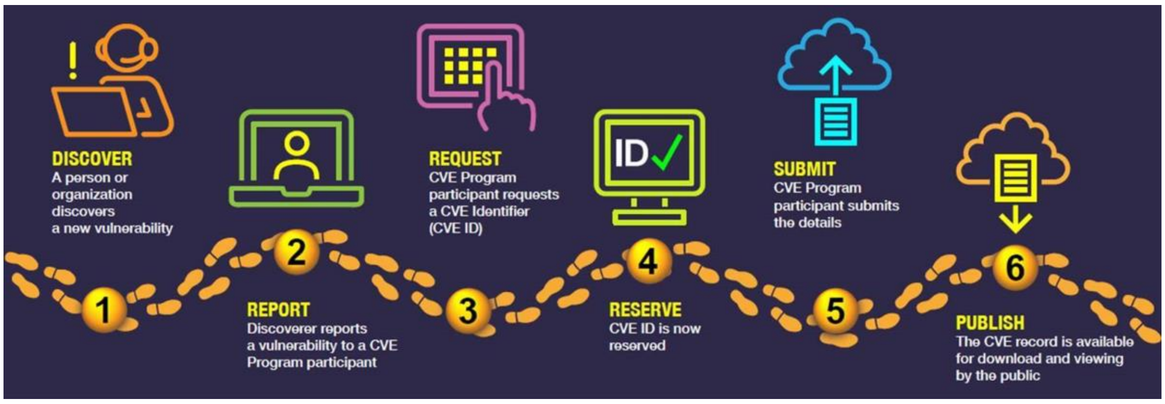

DISCOVER
A person or organization discovers a new vulnerability
1
2
REPORT
Discoverer reports a vulnerability to a CVE Program participant
REQUEST
CVE Program participant requests a CVE Identifier (CVE ID)
3
ID
4
RESERVE
CVE ID is now reserved
SUBMIT
CVE Program participant submits the details
5
6
PUBLISH
The CVE record is available for download and viewing by the public

Figure 5: CVE record lifecycle[5]

The information that is disclosed for every vulnerability includes: description, references to code repositories, assigning CNA, record date, etc. There is no information about risks, impacts, countermeasures, since this part of other databases such as National Vulnerability Database. CVE has the role of a baseline used among security advisories, bug trackers and databases to communicate with each other.

However, not all vulnerabilities are reported to CVE. The WhiteSource's 2018 Annual Report on the State of Open Source Vulnerabilities Management, reported that only 86% of existing vulnerabilities are added in the CVE list. According to that, 14% of vulnerabilities are reported somewhere else. Even though CVE list does not contain all the possible vulnerabilities, it is the most popular repository used for vulnerability documentation.

National Vulnerability Database (NVD). NVD [45] was created in 2000 with a different name (Internet categorization of Attacks Toolkit) and later got its present form. Its goal is to offer additional information about a CVE-ID of the CVE list. Said information includes a description, a severity score according to CVSS (Critical, High, Medium, Low), references to countermeasures, a list of relevant Common Weakness Enumerations, known affected software configurations and change history. NVD is fully synchronized with the CVE list and the provided information can be used for detection and fix of published known vulnerabilities.

Common Weakness Enumeration (CWE). CWE [46] is essentially a list of weakness types regarding software and hardware. This list is the output of a community effort that creates very specific definitions for each of these types, trying to distinct each one from the others while describing it in an adequate way. The members of the community are able to submit software and hardware weaknesses in the CWE Research Discussion List.

The information that is disclosed for every weakness includes: short and extended description, alternate terms, relationships with other weaknesses (ChildOf, ParentOf, CanFollow, MemberOf), modes of introduction, application platforms, common consequences with specific scope (confidentiality, integrity, availability), likelihood of exploit, demonstrative examples, observed examples, potential mitigation, detection

[5] https://www.cve.org/About/Process

methods, membership in specific CWE views, references in publications, and content history (submission, modifications).

CWE offers a number of different representations of the available list of weaknesses, based on concepts that are used in software development, hardware design, and research. Moreover, there are more views such as “CWE top 25 (year)”, “OWASP top ten (year)”, “Architectural concepts”. These view offer subsets of the list that are connected to a specific factor. It is also worthy of mentioning that CWE calculations are strongly biased in favour of frequency over severity as pointed out by authors in [47]

Common Attack Pattern Enumeration and Classification (CAPEC). CAPEC [48] is a classification and dictionary of known attacks. Common attack patterns are described helping the understanding of how weaknesses of systems can be exploited. New content is constantly added by enterprises and public participation aiming the disclosure of attack patterns to the security community. This information can be valuable for anyone who wants to strength their defences.

Attack patterns are organized hierarchically. On the top of the hierarchy a number of different categories can be found. Under these categories, meta patterns follow, and then we find standard patterns. Finally, patterns with more details may be the children of the standard patterns. Said hierarchy can be seen in Figure 6 for the Cross-Site Tracing attack. In this specific example, *Domains of attack* is the view, *Software* is the top category, *Exploitation of Trusted Identifiers* is the meta pattern, *Session Hijacking* is the standard pattern, and *Cross Site Tracing* is the detailed one. As a result, the hierarchy of the detailed pattern *Cross Site Tracing*, looks like this:

(V) Domains of attack → *(C) Software* → *(M) Exploitation of Trusted Identifiers* → *(S) Session Hijacking* → *(D) Cross Site Tracing*

The usage of this hierarchy and naming convention, easies the discovery of attack patterns and the definition of mitigation actions.

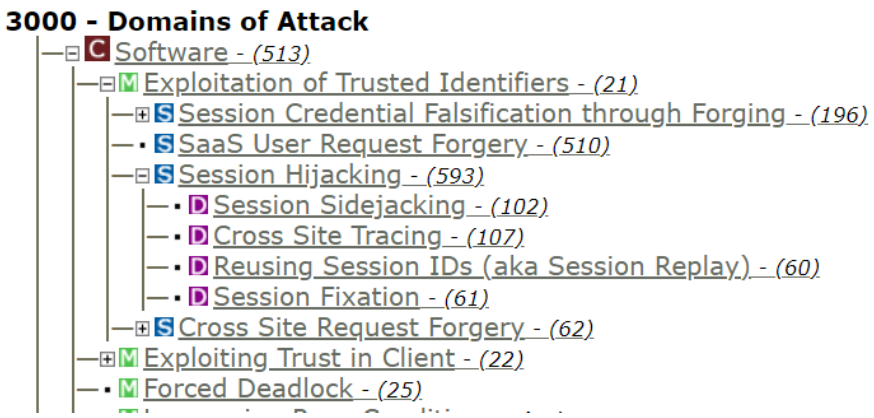

```
3000 - Domains of Attack
 ─⊟ C Software - (513)
    ─⊟ M Exploitation of Trusted Identifiers - (21)
       ─⊞ S Session Credential Falsification through Forging - (196)
       ─▪ S SaaS User Request Forgery - (510)
       ─⊟ S Session Hijacking - (593)
          ─▪ D Session Sidejacking - (102)
          ─▪ D Cross Site Tracing - (107)
          ─▪ D Reusing Session IDs (aka Session Replay) - (60)
          ─▪ D Session Fixation - (61)
       ─⊞ S Cross Site Request Forgery - (62)
    ─⊞ M Exploiting Trust in Client - (22)
    ─▪ M Forced Deadlock - (25)
```

Figure 6: Hierarchy of Cross Site Tracing attack in CAPEC

The information that is disclosed for each attack includes: description, likelihood of attack, severity, relationships with other attacks (ChildOf, ParentOf, etc.), execution flow of the attack, prerequisites, required skills regarding the attacker, required resources, indicators that the attack has been performed, consequences, mitigations, example instances, weaknesses that are related to the attack, possible mappings to other taxonomies, and content history.

Robot Vulnerability Database (RVD). RVD includes robot related vulnerabilities and bugs that are referred to software and hardware. The aim is to record and categorize robot related flaws. RVD is available at GitHub and offers tools that ease its management. Robot Vulnerability Scoring System (RVSS) is used for the rating of the included vulnerabilities.

3. THE SESAME SECURITY METHODOLOGY

3.1 PROCESSES OF THE SESAME SECURITY METHODOLOGY

As it is already described in section 2.2.1, the threat modeling process allows for the specification of the security design and collection of security technologies that will be adopted by a system, taking under consideration the system itself and its own security requirements. The security assessment that will be conducted in the context of SESAME is strongly influenced by the threat modeling process and adopts its main principles. More specifically, SESAME security assessment follows a structured series of steps, which overlaps in many cases with the highly structured process of threat modeling and its clearly specified steps, based on the chosen model. Diagram in Figure 7 depicts the steps of our methodology, including inputs, outputs, additional external resources, and processes.

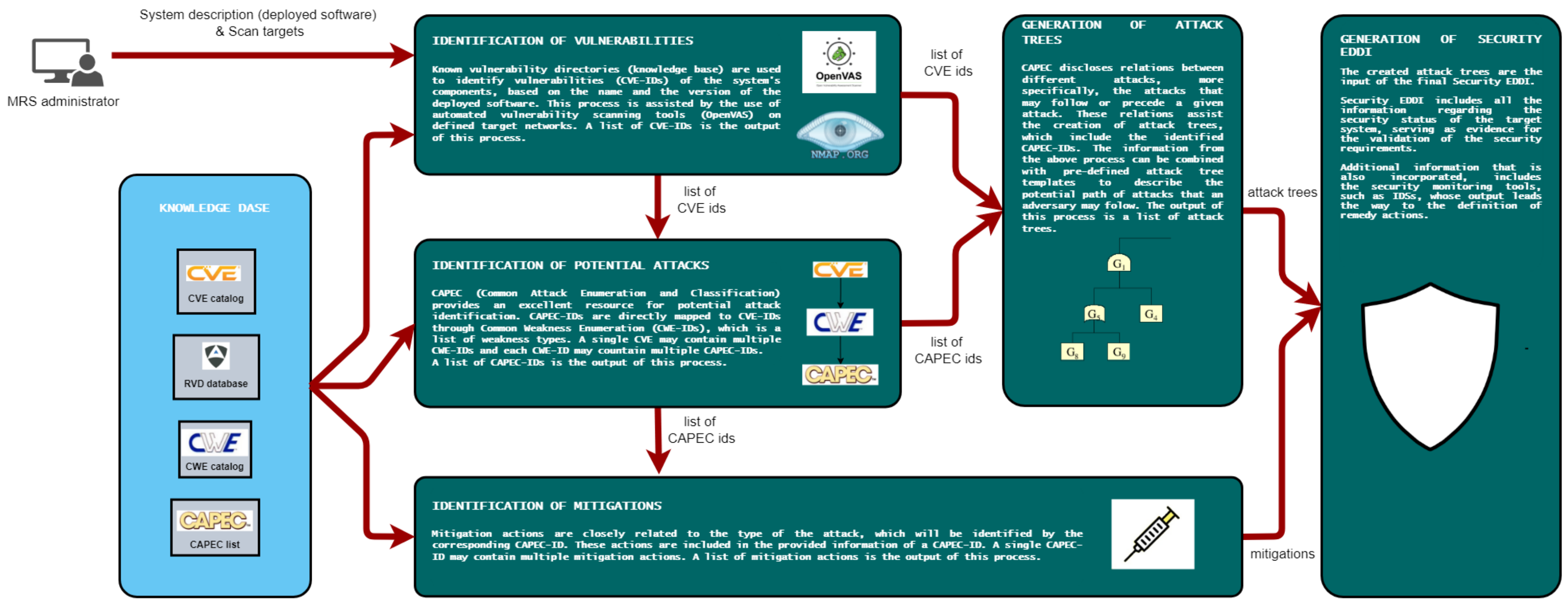
Although the high-level steps described in the next sections constitute a general approach, the key for a successful application of the proposed methodology to individual MRSs with specific requirements is the detailed description of the system in question. Concepts such as the assets importance and the trust boundaries that contain assets, can describe the unique security requirements of a system. Moreover, the identified vulnerabilities and their combination makes each system a different use case, making the SESAME security methodology to produce varying outputs (potential attack scenarios along with mitigations).

3.1.1 Identification of vulnerabilities

The security assessment starts with the description of the system. As it is indicated by the threat modeling process, the description of the system in question, in terms of architecture components, assets, entry points and trust boundaries, is necessary. This information is gathered by the system administrator. The intention in this step of the security assessment is a UI to be created where said information is collected by filling in forms and answering corresponding questionnaires.

Based on the system description, all the deployed programs, libraries and services are pinpointed. User’s input triggers the process during which free databases for the known vulnerabilities of the recognized software are used, with CVE catalog to be the first choice for this purpose. However, since our target systems are MRSs, RVD directory, dedicated to disclosure of bugs, weaknesses and vulnerabilities in robots, is considered a necessary addition. In that way, the complexity and special characteristics of robots, not reflected in other vulnerability lists, is taken under consideration. RVD aspires to add to the vulnerability disclosure with robotics specific information [49]. A parser searches said vulnerability directories and, using the name and version of each software present in the system in question, spots the associated vulnerabilities. Each of those vulnerabilities are uniquely identified by the CVE identifiers (CVE-IDs). A list of such CVE-IDs is the output of this process. The said vulnerability directories are constantly updated with information regarding newly discovered vulnerabilities with a method described in section 2.2.3. Inherently the approach followed here is also not static as it will be in sync with the updated directories.

The process that was just described is also offered as functionality by a set of automated tools, called vulnerability scanners. Following the same principle; they are scanning a given network and/or subnetworks for available services and then use open vulnerability databases to discover known vulnerabilities. OpenVAS, OPENSCAP, OWASP ZAP are some of the open-source options. For the sake of completeness, the SESAME security assessment includes the use of such scanning tools, since the provider of the system information may not be aware of some services that are running in devices, which are part of the system, and have some known vulnerabilities. Of course, the prerequisite in this case is that the system must be up and running, otherwise the vulnerability scanning tools cannot produce an output.



IDENTIFICATION OF VULNERABILITIES

Known vulnerability directories (knowledge base) are used to identify vulnerabilities (CVE-IDs) of the system's components, based on the name and the version of the deployed software. This process is assisted by the use of automated vulnerability scanning tools (OpenVAS) on defined target networks. A list of CVE-IDs is the output of this process.

IDENTIFICATION OF POTENTIAL ATTACKS

CAPEC (Common Attack Enumeration and Classification) provides an excellent resource for potential attack identification. CAPEC-IDs are directly mapped to CVE-IDs through Common Weakness Enumeration (CWE-IDs), which is a list of weakness types. A single CVE may contain multiple CWE-IDs and each CWE-ID may countain multiple CAPEC-IDs. A list of CAPEC-IDs is the output of this process.

IDENTIFICATION OF MITIGATIONS

Mitigation actions are closely related to the type of the attack, which will be identified by the corresponding CAPEC-ID. These actions are included in the provided information of a CAPEC-ID. A single CAPEC-ID may contain multiple mitigation actions. A list of mitigation actions is the output of this process.

GENERATION OF ATTACK TREES

CAPEC discloses relations between different attacks, more specifically, the attacks that may follow or precede a given attack. These relations assist the creation of attack trees, which include the identified CAPEC-IDs. The information from the above process can be combined with pre-defined attack tree templates to describe the potential path of attacks that an adversary may folow. The output of this process is a list of attack trees.

GENERATION OF SECURITY EDDI

The created attack trees are the input of the final Security EDDI.

Security EDDI includes all the information regarding the security status of the target system, serving as evidence for the validation of the security requirements.

Additional information that is also incorporated, includes the security monitoring tools, such as IDSs, whose output leads the way to the definition of remedy actions.

Figure 7: SESAME security methodology

3.1.2 Identification of potential attacks

The output of both processes of identification of vulnerabilities is a set of CVE-IDs, which serves as input to the next step of the security assessment, the identification of the potential attacks to the system. CWE catalog, a list of software and hardware weakness types, is an additional input for the process during this step. CWE plays the role of common language for security tools. Due to the wider acceptance of CWE it is used as a stepping stone between the spotted vulnerabilities and the potential attacks. Finally, CAPEC, a dictionary of identifiers for attack patterns that are used by attackers to take advantage of weaknesses, is used as input.

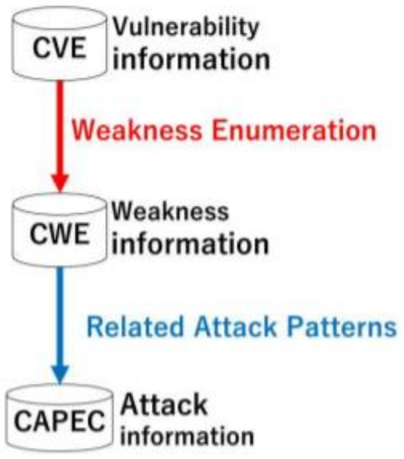


Figure 8: Discovering potential attacks from known vulnerabilities – from [50]

Figure 8 depicts the route from the pinpointed vulnerabilities of a system software to the information of the corresponding potential attacks, showing how all the aforementioned directories are connected with each other.

CWE, as we already mentioned, is used as a connection point between CVE and CAPEC. “Weakness Enumeration” is one of the fields in the description of a given vulnerability. In this field a list of all the weakness types, in the form of CWE-IDs, that are related with the specific vulnerability is provided. Moreover, the description of every weakness type (CWE-ID) includes a field called "Related Attack Patterns", presenting the attack patterns (CAPEC-ID) used for the exploitation of the corresponding weakness. In this way, it is possible to trace a list of CAPEC-IDs from a single CVE-ID. This process is repeated for every of the discovered system vulnerabilities, meaning that its output is a number of different sets of CAPEC-IDs, one for each discovered CVE-ID.

Information related to a CAPEC-ID that is highly valuable to us, includes a description in natural language, relationship with other attacks, prerequisites for the attack to be performed, and mitigation actions.

3.1.3 Identification of mitigations

In subsection 2.2.3 we have already described the hierarchical classification of CAPEC and the corresponding levels. CAPEC-IDs in standard and detailed levels usually include mitigation actions. More specifically, especially for the detailed level, a very specific protection mechanism is required to mitigate the actual attacks and this mechanism is mentioned under the *Mitigations* section. During the identification of mitigations process the information under *Mitigation* tab is collected for every defined CAPEC-ID. If the CAPEC-ID in question is of standard and detailed level, the information is directly available. On the other hand, a meta level attack pattern is more abstract, avoiding information about specific methodologies, techniques, implementations and protection mechanisms. Said attack pattern serves as generalization of a more well-defined group of standard level attack patterns. In such a case, mitigations that are mentioned in the corresponding standard level attack patterns will be utilized for gathering the mitigation actions.

3.1.4 Generation of attack trees

The next process in the SESAME security methodology is the generation of attack trees, including two different steps. During the first step the information that is disclosed by the CAPEC repository is once again utilized. The CAPEC hierarchical classification includes different types of relationships between two attack patterns, including *CanFollow* and *CanPrecede*. The former gives information about the attacks that may follow a given attack according to a specific attack pattern. The latter reveals attacks that could have been conducted before a given attack, opening the way for it. Following these relationships, we can create different graphs/trees, one for each possible known attack pattern. An example of such a graph can be seen in Figure 9.

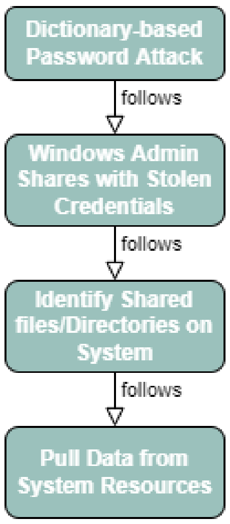

Dictionary-based Password Attack
follows
Windows Admin Shares with Stolen Credentials
follows
Identify Shared files/Directories on System
follows
Pull Data from System Resources

Figure 9: Example graph that can be produced utilizing the CanFollow relationship of CAPEC

Prerequisite for the creation of such a graph is the identification of each of the included attacks due to vulnerabilities that have been previously found in the system in question. The existence of specific vulnerabilities allows the identification of potential attacks. Supposedly all the attacks mentioned in Figure 9 are already identified as potential attacks, the first step of the generation of attack trees process with create the corresponding graph based on the CanFollow relationship.

During a *Dictionary-based Password Attack*, an attacker tries all the words of a dictionary as passwords of a specific user account. If the chosen password is in the dictionary, the attack is successful and the attacker gains access. In case the broken account is a Windows administrator account, the attacker could conduct a *Windows Admin Shares with Stolen Credentials* attack. During such an attack, the attacker gets access to Windows Admin Shares, which allow administrators to access all disk volumes on a network-connected system and copy, write and execute files. This opens the way for another attack, the *Identify Shared Files/Directories on System*. During this attack, the adversary may locate and collect sensitive data through the use of shared folders or drives between systems or system parts. Another possible usage of required information is the design of routes in the network that serve other attacks. An attack that can follow is the so-called *Pull Data from System Resources*. During this attack, an adversary pulls data from resources that has access, such as files or memory. The attacker does not need to know what the information that they pull is. The scanning of the information can be done afterwards.

When this first step of generation of attack trees process finishes, a number of graphs, showing how identified attacks can be combined, has been created.

During the second step utilization of attack tree templates is taking place. The attack tree template are predefined attack trees. According to the general form of an attack tree, the root of said trees is called goal of the attacker and the leaves are ways to achieve the goal. Since there might be several attacker goals, there would be different attack trees. Between the goal and the leaves, one can find sub-goals, which describe achievements of the attacker that bring them closer to their goal. Such goals could include the following:

- Control the movement of the robot
- Make the robot unresponsive (loss of availability)
- Steal/Change sensitive information (loss of integrity)
- Make a robot not to achieve a business goal

An example attack tree template not closely related to MRS, is depicted in Figure 10 for demonstrative purposes. As it can be seen, the goal of the attacker in this case is the loss of availability of a specific service. On the other hand, the leaves represent vulnerabilities of the system that can be exploited and attacks that can be performed based on these vulnerabilities.

CVE-2021-41450 corresponds to a vulnerability of certain HTTP/2 implementations. According to this vulnerability, a flood of empty frames can be performed, leading to a DoS attack. During such an attack, frames are sent with empty body and no end-of-stream flag, while the receiver tries to process them consuming CPU. The actual attacking machine that can be used for this attack can be a machine inside the system in question that is taken under control due to another attack.

Additionally, CVE-2021-44026 corresponds to a vulnerability of the certain versions of the Roundcube open-source webmail software. The email client is prone to SQL injection attacks. Such an attack could be utilized by an adversary to take control of the host in which Roundcube is deployed. Such an attack could lead to the convertion of the host to a harmful bot, programmed to perform attacks to other machines in the system.

The two aforementioned vulnerabilities can be combined and create an attack tree template. Such a template could be applied to a system where both these vulnerabilities are present. Moreover, attack tree templates can be used for merging together graphs that have been created in the first step of this process. If the vulnerabilities mentioned in the leaves of a template are included in a graph, that graph could substitute the leaf. In that way, more than one graph could replace leaves and be merged in an attack tree template.

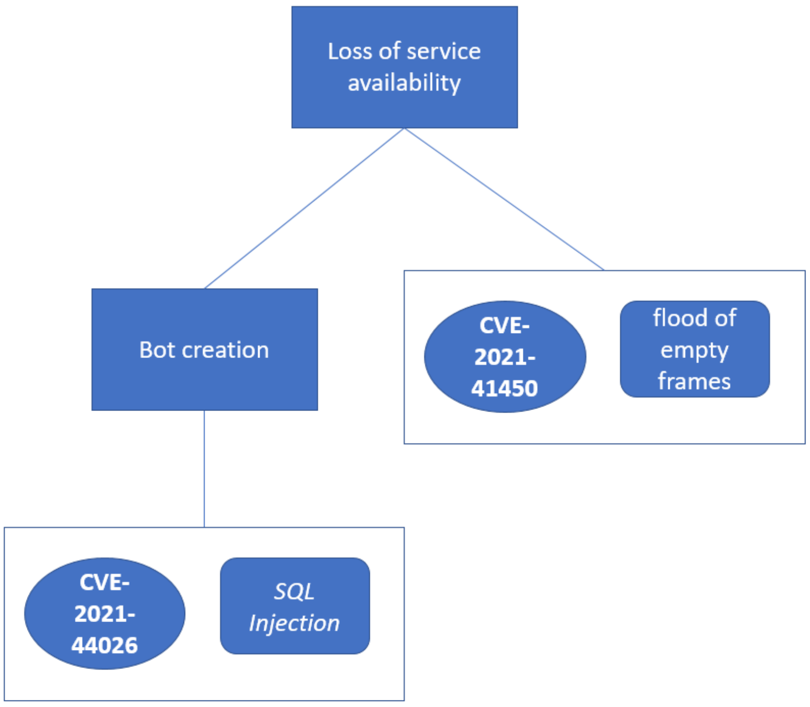


Figure 10: Example Attack Tree Template

The final attack tree that constitutes the output of this process, could include additional information, such as the severity of the attack, its likelihood, the overall risk, monitoring mechanisms for the detection of an attack, etc.

3.1.5 Generation of security EDDIs

The output of the *generation of attack trees* process serves as input in the *generation of security EDDIs* process. During the latter, all the produced information is used for the creation of the security EDDI.

The EDDI solution is the evolution of the DDI concept. It is an extended version, made to include properties necessary for runtime deployment and addressing MRS related issues. EDDI serves, at the same time, as a design-time dependability artefact, and as a dynamic dependability management tool. The overall role of an EDDI is twofold including i) online monitoring that observes and manages the system’s safety and security, and ii) distributed communications among the different system components for managing the dependability of a wider MRS system.

The EDDI’s features include the following:

- Event monitoring to monitor dependability-related inputs from the system;

- Runtime diagnostics to determine probable causes and possible consequences of detected failure events;
- Dynamic risk prediction, to update design-time risk estimates with new information based on the current system state;
- Mitigating actions and recovery planning, such as recommending the system enter a safe failure state or a degraded mode to continue operation.
- Intercommunication with other connected EDDIs to both assure them of the system dependability status and respond to errors reported by other EDDIs.

More details about the EDDI and the DDI concepts can be found in D4.1 "Safety analysis Concept", where an elaborated description of both can be found along with their architectures.

At this stage, the produced EDDI serves as a repository of information about the system in question regarding security. It can be used as evidence that the security requirements for the individual parts of the system are being met.

The information that is produced by the security assessment must be conformed to the ODE metamodel. Structured Assurance Case Meta-Model (SACM), a metamodel specialised for the creation of structured system assurance cases, provides the ODE with assurance case support. In our case, an assurance case incorporates the arguments and evidence that support the claim that a given system or service is able to satisfy safety and security requirements. The form such an assurance case can be expressed in is a machine-readable model carrying information such as the scope of the system, the operational context and the safety and/or security arguments [51].

The ODE includes a security-oriented package called Threat Analysis and Risk Assessment (TARA). This package captures *Risk Assessment* that is based on *Threat Agents*, which perform *Attacks* taking advantage of *Assets* with identified *Vulnerabilities*. The performed attacks can be addressed by Security Capabilities of the system, which are implemented by Security Controls [52]. The TARA package in the form of a class diagram is depicted in Figure 11.

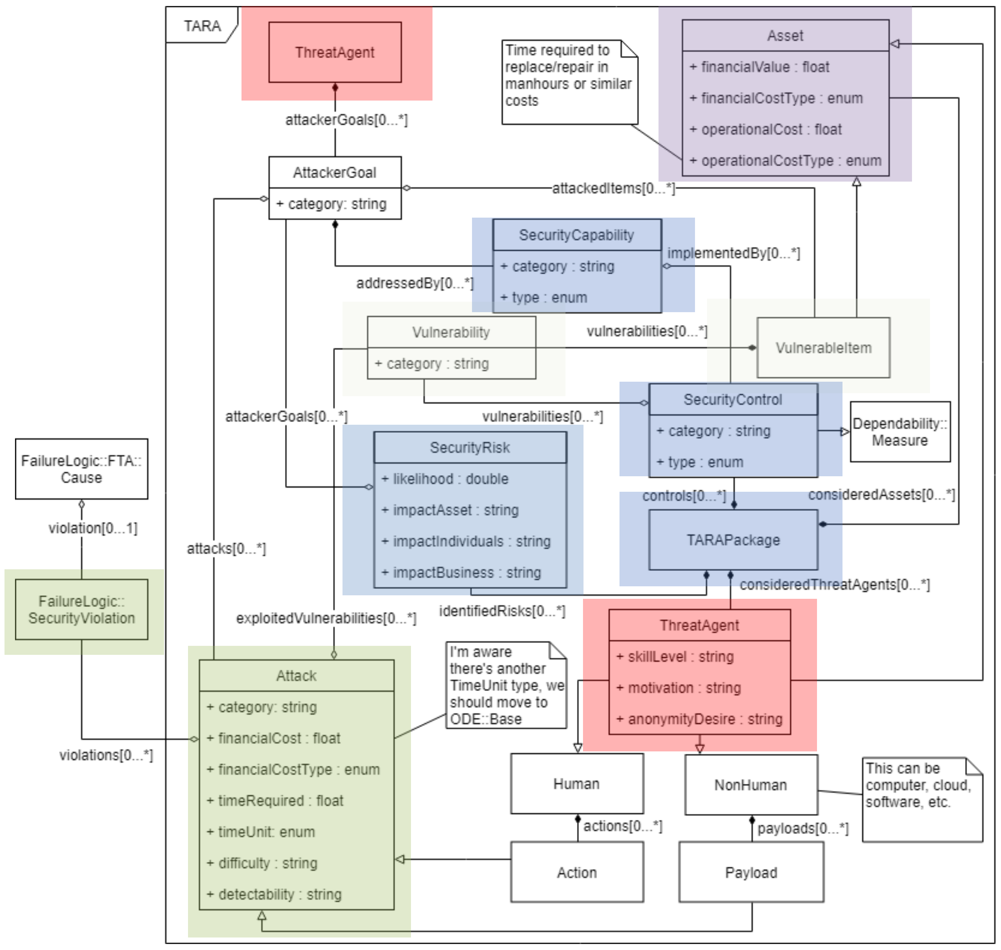


Figure 11: ODE TARA package

Information that is useful to the SESAME security assessment methodology includes the CVSS score of a vulnerability, the importance of a system asset and the type of a threat/attack. CVSS score depicts the severity of a vulnerability, which is one of the indicators that determine how the vulnerability itself or attacks based on it, will be confronted. The importance of a system asset specifies if its vulnerabilities are worth threating. If the role of a specific component is not major regarding the overall functionality of the system, then its unavailability, due to a security attack, may not be a problem. Finally, the type of a threat/attack allows for the determination of the mechanism that is going to be used in runtime to monitor it. This is very valuable information as far as safety is concerned.

EDDIs can also incorporate information for communicating with runtime security monitoring tools. Valuable input from tools such as IDSs, Anti-Viruses and BDSs can be included in the security part of an EDDI, towards to definition of remedy actions. In subsection 2.1.2.2, such monitoring tools have been already mentioned. IDSs are used for the identification of malicious packets. In case of protection of known attacks, attack signatures as used for the creation of rules that recognize specific patterns in the header or body of the traffic packets. As far as the unknown attacks are concerned, anomaly detection technique is used, detecting alteration in the traffic from the normal one. In both cases, the detection of an attack creates alerts for the system administrator. Moreover, Anti-Viruses are another type of protection that detects and removes malware from the host. Additional functionality is the detection of unusual properties due to known viruses, in incoming documents, such as emails, attachments, etc. Breaches and side-channel attacks are detected by BDSs. Finally, anti-phishing solutions protect from phishing attempts.

As it is described in D4.1 “Safety Analysis Concept and Methodology for EDDI development (Initial Version)”, the deployment of an EDDI could be done two ways. It could be synthesized into code and run on the target platform or, in a virtual machine-style approach, the EDDI is executed by a target-specific native program.

3.1.6 Runtime security

As it is mentioned already, the produced information from the security assessment process is incorporated in the security EDDI, transferring in this way, said information to the runtime, to be used for the mitigation of threats. The prerequisite, in this case is the monitoring of the security events that need to take place also during runtime. In subsection 3.1.5 a number of potential security monitoring tools have been mentioned, such as IDS, Anti-Virus and BDS. Our intention is to use an IDS to monitor the network incoming malicious packets. Regarding the type of detection, a signature based IDS is going to be used, where specific patterns are recognized in the headers or body of traffic packets. There are some candidates that could be used, such as Snort, Suricata, Zeek, OSSEC, etc. It is possible that our choice will be Snort due to our familiarity with the tool, and its usage for the creation of components in other systems. Such a tool will allow for the detection of attacks towards the system in question and the creation of corresponding alerts.

The created IDS alerts then, can serve as input to a Business Rule Management System (BRMS). Drools is such a BRMS that allows for the construction, maintenance, and enforcement of business policies in an organization, application, or service. A Drools production rule has the following generic structure:

rule name <attributes>*

when <conditional element>*

then <action>* end

The when part of the rule specifies a set of conditions, and the part then of the rule, a list of actions. When a rule is applied, the Drools rule engine checks whether the rule conditions (defined within the <conditional element> above) match with the facts in the Drools Knowledge Base (KB), and if they do, it executes the actions of the rule. Rule actions are typically used to modify the KB by inserting, retracting, or updating the objects (facts) in it through the standard Drools actions “insert”, “retract”, and “update”, respectively. The desired mitigation actions that fit better to the detection of an attack can be expressed as Drools rules. A very simple example would be the detection of a DoS attack that is launched from a specific IP (condition element), which leads to the drop of all the incoming packets from that specific IP (action). These rules could also serve as user-defined security policies. Such policies can set a security level and Drools can indicate if the level is reached or not, during runtime.

3.1.7 Safety and security

According to the definitions of security and safety given in [53], security refers to "the protection of a plant or machinery from unauthorised access from outside as well as the protection of sensitive data from corruption, loss and unauthorised access from within", while safety "denotes the functional safety of plants, meaning the protection of people and the environment against foreseeable threats that can stem from machinery". Although these two concepts have different meanings, there is a strong correlation between them regarding the robotic systems.

As we saw in the previous section with the common attacks to robotic systems, an attack that ends up with manipulation of the robot parameters (control or calibration) can cause human injuries during a human-machine interaction. On the other hand, the existence of external safety sensors that need to be integrated into a robotic system, creates additional attack surfaces to the system [54]. This complex relationship between security and safety makes it crucial to take care both these aspects to avoid faulty and unexpected robot behaviour.

As far as the EDDI implementation is concerned, information about types of threats with safety implications should be included in the safety EDDI. Moreover, information about the way said threats are monitored during runtime should be included. In that way, safety runtime mechanisms could take under consideration security events, such as detected ongoing attacks, and end up with some safety adaptation actions.

The security assessment process described herein, can create input for the safety reasoning model. Security and safety are both important regarding the dependability goal. Combing both security and safety assessment the dependability of a robotic system can be achieved.

3.2 APPLYING SESAME METHODOLOGY

In this section, we will present how the SESAME security assessment can be applied to the use case 2: *Disinfecting hospital environments using robotic teams*. This is an initial approach that serves as proof-of-concept for the applicability of the proposed approach. A more complete application of the SESAME security assessment, on all five use cases of the project, will be presented in future deliverables such as D5.3 and D5.6 *Tools for Automated Security Analysis of MRS and for Production of EDDIs*, initial and final version, respectively.

As it is already presented in D1.1 Project Requirements, the SESAME use case partner Locomotec uses a fully autonomous fleet of robots for disinfection of contact surfaces and aerosols using UV-C as a known method to disinfect surfaces. In order to ensure that persons are not over exposed with the maximum daily dose, a person detection system is installed. Once a person is detected, the lamps on the UV-C robot turn off.

The main components of a robot are depicted in below.

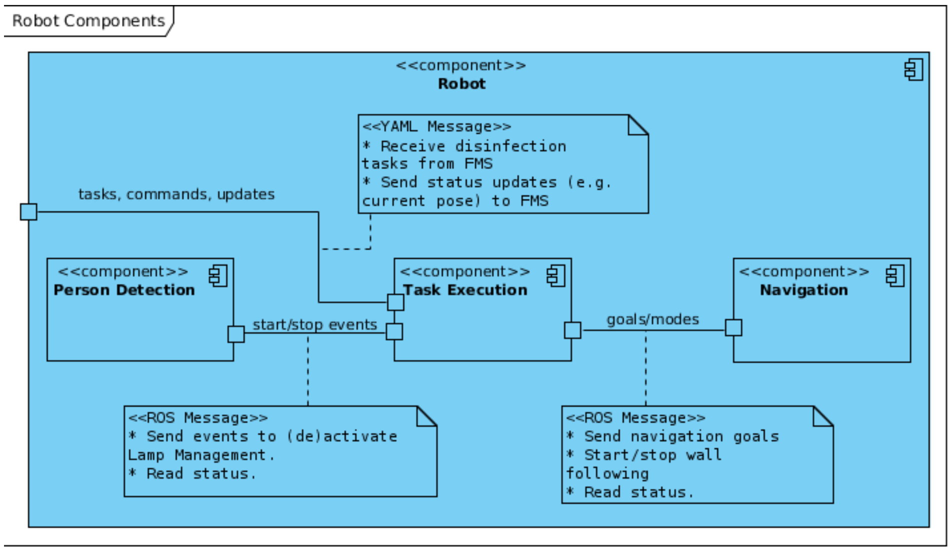


Figure 12: Main software components of a Robot

As it can be seen in Figure 12, ROS is used on the robots as the build system and communication middleware. Since this is just a test for the applicability of our methodology, ROS is going to be used as the initial input from the system administrator. If we wanted to perform the security assessment throughout, we should have more details about all the deployed software.

Searching using the keyword “ROS” in the two security knowledge repositories, CVE and NVD, 39 different entries come out. However, not all of them are related to the actual robot operating system. After the removal of vulnerabilities of other systems, such as RuggedCom Rugged Operating System (whose acronym is also ROS), that misleadingly appear in the search results, the remaining vulnerabilities are presented in Table 1.

Table 1: Identified ROS-related vulnerabilities in CVE and NVD repositories

CVE-IDs	CWE-IDs	CWE Name
CVE-2016-10681	CWE-300	Channel Accessible by Non-Endpoint
CVE-2016-10681	CWE-310	Cryptographic Issues
CVE-2019-13445	CWE-190	Integer Overflow or Wraparound
CVE-2019-13465	CWE-noinfo	NA
CVE-2019-13566	CWE-120	Buffer Copy without Checking Size of Input ('Classic Buff-er Overflow')
CVE-2019-19625	CWE-200	Exposure of Sensitive Information to an Unauthorized Actor
CVE-2019-19627	CWE-200	Exposure of Sensitive Information to an Unauthorized Actor
CVE-2020-10271	CWE-668	Exposure of Resource to Wrong Sphere
CVE-2020-10272	CWE-306	Missing Authentication for Critical Function
CVE-2020-10289	CWE-20	Improper Input Validation
CVE-2020-16124	CWE-190	Integer Overflow or Wraparound

As it can be seen, the CVE-IDs along with the corresponding CWE-IDs and the CWE names are presented. The identified vulnerabilities (CVE-IDs) are coupled with weaknesses (CWE-IDs), such as buffer overflow, missing authentication, exposure of sensitive information, etc. Moreover, each of these weaknesses is related to specific attack patterns, attacks that an adversary can conduct taking advantage of the said weakness. As a result, the defined weaknesses lead to a number of potential attacks, presented in Table 2.

Table 2: Identified ROS-related attack patterns in CAPEC repository

CWE-IDs	CAPEC-IDs	CAPEC Name
CWE-300	CAPEC-466	Leveraging Active Adversary in the Middle Attacks to Bypass Same Origin Policy
	CAPEC-57	Utilizing REST's Trust in the System Resource to Obtain Sen-sitive Data
	CAPEC-589	DNS Blocking
	CAPEC-590	IP Address Blocking
	CAPEC-612	WiFi MAC Address Tracking
	CAPEC-613	WiFi SSID Tracking
	CAPEC-615	Evil Twin Wi-Fi Attack
	CAPEC-662	Adversary in the Browser (AiTB)
	CAPEC-94	Adversary in the Middle (AiTM)
CWE-310	-	
CWE-190	CAPEC-92	Forced Integer Overflow
CWE-120	CAPEC-10	Buffer Overflow via Environment Variables
	CAPEC-100	Overflow Buffers
	CAPEC-14	Client-side Injection-induced Buffer Overflow
	CAPEC-24	Filter Failure through Buffer Overflow
	CAPEC-42	MIME Conversion
	CAPEC-44	Overflow Binary Resource File
	CAPEC-45	Buffer Overflow via Symbolic Links
	CAPEC-46	Overflow Variables and Tags
	CAPEC-47	Buffer Overflow via Parameter Expansion
	CAPEC-67	String Format Overflow in syslog()
	CAPEC-8	Buffer Overflow in an API Call
	CAPEC-9	Buffer Overflow in Local Command-Line Utilities
	CAPEC-92	Forced Integer Over
CWE-200	CAPEC-116	Excavation
	CAPEC-13	Subverting Environment Variable Values
	CAPEC-169	Footprinting
	CAPEC-22	Exploiting Trust in Client
	CAPEC-224	Fingerprinting
	CAPEC-285	ICMP Echo Request Ping
	CAPEC-287	TCP SYN Scan
	CAPEC-290	Enumerate Mail Exchange (MX) Records
	CAPEC-291	DNS Zone Transfers
	CAPEC-292	Host Discovery

CWE-IDs	CAPEC-IDs	CAPEC Name
	CAPEC-293	Traceroute Route Enumeration
	CAPEC-294	ICMP Address Mask Request
	CAPEC-295	Timestamp Request
	CAPEC-296	ICMP Information Request
	CAPEC-297	TCP ACK Ping
	CAPEC-298	UDP Ping
	CAPEC-299	TCP SYN Ping
	CAPEC-300	Port Scanning
	CAPEC-301	TCP Connect Scan
	CAPEC-302	TCP FIN Scan
	CAPEC-303	TCP Xmas Scan
	CAPEC-304	TCP Null Scan
	CAPEC-305	TCP ACK Scan
	CAPEC-306	TCP Window Scan
	CAPEC-307	TCP RPC Scan
	CAPEC-308	UDP Scan
	CAPEC-309	Network Topology Mapping
	CAPEC-310	Scanning for Vulnerable Software
	CAPEC-312	Active OS Fingerprinting
	CAPEC-313	Passive OS Fingerprinting
	CAPEC-317	IP ID Sequencing Probe
	CAPEC-318	IP 'ID' Echoed Byte-Order Probe
	CAPEC-319	IP (DF) 'Don't Fragment Bit' Echoing Probe
	CAPEC-320	TCP Timestamp Probe
	CAPEC-321	TCP Sequence Number Probe
	CAPEC-322	TCP (ISN) Greatest Common Divisor Probe
	CAPEC-323	TCP (ISN) Counter Rate Probe
	CAPEC-324	TCP (ISN) Sequence Predictability Probe
	CAPEC-325	TCP Congestion Control Flag (ECN) Probe
	CAPEC-326	TCP Initial Window Size Probe
	CAPEC-327	TCP Options Probe
	CAPEC-328	TCP 'RST' Flag Checksum Probe
	CAPEC-329	ICMP Error Message Quoting Probe
	CAPEC-330	ICMP Error Message Echoing Integrity Probe
	CAPEC-472	Browser Fingerprinting
	CAPEC-497	File Discovery
	CAPEC-508	Shoulder Surfing
	CAPEC-573	Process Footprinting
	CAPEC-574	Services Footprinting
	CAPEC-575	Account Footprinting
	CAPEC-576	Group Permission Footprinting
	CAPEC-577	Owner Footprinting
	CAPEC-59	Session Credential Falsification through Prediction
	CAPEC-60	Reusing Session IDs (aka Session Replay)
	CAPEC-616	Establish Rogue Location

CWE-IDs	CAPEC-IDs	CAPEC Name
	CAPEC-643	Identify Shared Files/Directories on System
	CAPEC-646	Peripheral Footprinting
	CAPEC-651	Eavesdropping
	CAPEC-79	Using Slashes in Alternate Encoding
CWE-668	-	
CWE-306	CAPEC-12	Choosing Message Identifier
	CAPEC-166	Force the System to Reset Values
	CAPEC-36	Using Unpublished Interfaces
	CAPEC-62	Cross Site Request Forgery
CWE-20	CAPEC-10	Buffer Overflow via Environment Variables
	CAPEC-101	Server Side Include (SSI) Injection
	CAPEC-104	Cross Zone Scripting
	CAPEC-108	Command Line Execution through SQL Injection
	CAPEC-109	Object Relational Mapping Injection
	CAPEC-110	SQL Injection through SOAP Parameter Tampering
	CAPEC-120	Double Encoding
	CAPEC-13	Subverting Environment Variable Values
	CAPEC-135	Format String Injection
	CAPEC-136	LDAP Injection
	CAPEC-14	Client-side Injection-induced Buffer Overflow
	CAPEC-153	Input Data Manipulation
	CAPEC-182	Flash Injection
	CAPEC-209	XSS Using MIME Type Mismatch
	CAPEC-22	Exploiting Trust in Client
	CAPEC-23	File Content Injection
	CAPEC-230	XML Nested Payloads
	CAPEC-231	Oversized Serialized Data Payloads
	CAPEC-24	Filter Failure through Buffer Overflow
	CAPEC-250	XML Injection
	CAPEC-261	Fuzzing for garnering other adjacent user/sensitive data
	CAPEC-267	Leverage Alternate Encoding
	CAPEC-28	Fuzzing
	CAPEC-3	Using Leading 'Ghost' Character Sequences to Bypass Input Filters
	CAPEC-31	Accessing/Intercepting/Modifying HTTP Cookies
	CAPEC-42	MIME Conversion
	CAPEC-43	Exploiting Multiple Input Interpretation Layers
	CAPEC-45	Buffer Overflow via Symbolic Links
	CAPEC-46	Overflow Variables and Tags
	CAPEC-47	Buffer Overflow via Parameter Expansion
	CAPEC-473	Signature Spoof
	CAPEC-52	Embedding NULL Bytes
	CAPEC-53	Postfix, Null Terminate, and Backslash
	CAPEC-588	DOM-Based XSS
	CAPEC-63	Cross-Site Scripting (XSS)

CWE-IDs	CAPEC-IDs	CAPEC Name
	CAPEC-64	Using Slashes and URL Encoding Combined to Bypass Validation Logic
	CAPEC-664	Server Side Request Forgery
	CAPEC-67	String Format Overflow in syslog()
	CAPEC-7	Blind SQL Injection
	CAPEC-71	Using Unicode Encoding to Bypass Validation Logic
	CAPEC-72	URL Encoding
	CAPEC-73	User-Controlled Filename
	CAPEC-78	Using Escaped Slashes in Alternate Encoding
	CAPEC-79	Using Slashes in Alternate Encoding
	CAPEC-8	Buffer Overflow in an API Call
	CAPEC-80	Using UTF-8 Encoding to Bypass Validation Logic
	CAPEC-81	Web Logs Tampering
	CAPEC-83	XPath Injection
	CAPEC-85	AJAX Footprinting
	CAPEC-88	OS Command Injection
	CAPEC-9	Buffer Overflow in Local Command-Line Utilities

The first column of the table includes the IDs of the weaknesses mentioned in Table 1. The other two columns present the IDs and the names of the attack patterns that are related to each of the weaknesses. Each of the included attacks are attacks that could be conducted to our system. Some of these attacks are more abstract, while others provide low level of details. As we described in subsection 2.2.3, attack patterns have different types, with meta attack patterns being the most abstract without mentioning used technologies or attack implementation, and detailed attack patterns being the most specific with information of the attack techniques and the protection mechanisms. This attack pattern classification makes the detailed attack patterns more valuable to this process, especially for the identification of the mitigations part. Some of the mentioned attacks may require the presence of additional technologies, other than just ROS. However, since this is an example application of our approach and we do not have the full picture of the testing system, we consider all the identified attacks as valid.

Although each of the attacks included in the table above is a threat for our system on its own, some of them can be combined and create more complex attacks. Using the CanFollow and CanPrecede relationships between the attack patterns four small graphs are created, presented in Figure 13.

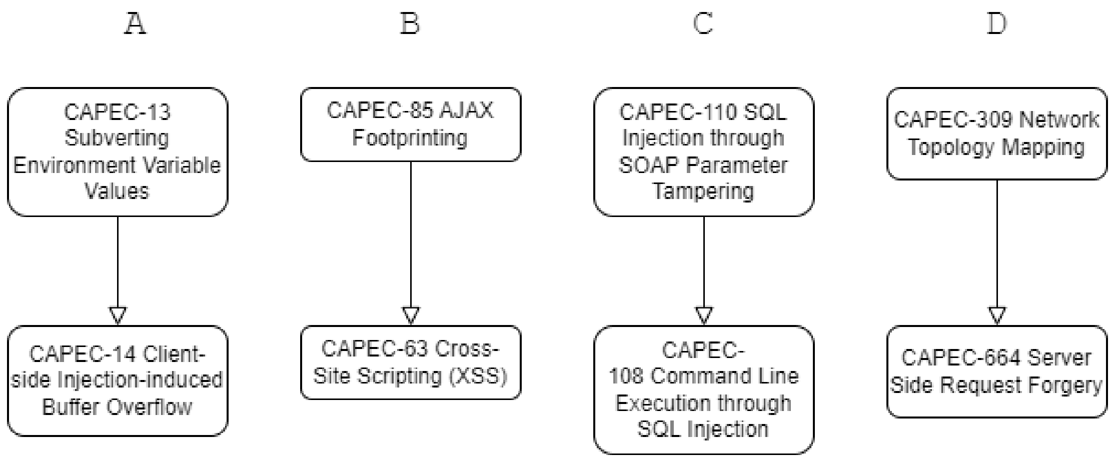

A
B
C
D
CAPEC-13 Subverting Environment Variable Values
CAPEC-14 Client-side Injection-induced Buffer Overflow
CAPEC-85 AJAX Footprinting
CAPEC-63 Cross-Site Scripting (XSS)
CAPEC-110 SQL Injection through SOAP Parameter Tampering
CAPEC-108 Command Line Execution through SQL Injection
CAPEC-309 Network Topology Mapping
CAPEC-664 Server Side Request Forgery

Figure 13: Combined attack patterns based on the CanFollow and CanPrecede relationships

These graphs group together attacks that can be conducted in sequence. One attack can create the necessary prerequisites for another attack to happen.

The aforementioned graphs along with the individual identified attacks can be combined in larger attack graphs that are created based on attack tree templates, described in subsection 3.1.4. Such a tree that combines a subset of the attacks that we spotted based on the vulnerabilities of ROS is depicted in Figure 14.

The tree includes five of the attacks identified in the previous steps of the security assessment. If all of them or some of them are conducted, the final goal of the attacker will be reached. The final goal of the attacker, regarding this attack tree, is for the target robot to crash with a person. Following the opposite direction than the one depicted by the arrows, an adversary has to publish to a specific ROS topic, responsible for movement of the robot. There are two paths that the attacker can follow to be able to do so. This is indicated in the attack tree with the OR node.

According to the first path, the one depicted in the left side, they can use the ROS CLI tool. It can be done if the attacker manages to compromise a robot in the same network with the target robot. CAPEC-615 and CAPEC-94 are two attacks that can lead to exactly that. CAPEC-615 is the identifier of the Evil Twin Wi-Fi Attack. During such an attack, the attacker installs Wi-Fi equipment that acts as a legitimate Wi-Fi network access point. As soon as the target node connects, the exchanged traffic is intercepted, captured, and analyzed. This attack leads the way for the CAPEC-94 attack. CAPEC-94 is the identifier for the Adversary in the Middle attack. During such an attack, the attacker is placed between two communicating components, observing and possibly altering the data before reaching their intended receiver. We should mention here, that the presence of the former attack is not necessary for the attacker to accomplish their goal. Of course, the latter attack has to be conducted successfully.

The second path, which is depicted on the right side of the tree, can be materialized if the target robot exposes an API. As we already discussed in subsection 2.1, such APIs allow for the control of a robot. In this case, the attacker can publish to the topic that controls the movement of the target robot by compromising the exposed API. At this point the attack tree is once again divided into two paths. According to the first path, the CAPEC-8, Buffer Overflow in an API Call, is sufficient for achieving such a goal.

During such an attack, the adversary takes advantage of libraries or code vulnerable to buffer overflow attacks and targets software that includes them. According to the second path, CAPEC-85 and CAPEC-63, conducted in sequence, can have the same outcome. CAPEC-85 is the identifier for the AJAX Footprinting attack. The most common first step of an attacker is to try to find information about the targeted environment to design their attack. Such a process is eased by the frequent communication that is taking place during an Ajax conversation. The knowledge that is gained is then used for conducting other attacks. CAPEC-63 refers to the Cross-Site Scripting attack. During such an attack, an adversary incorporates scripts in content that is going to be consumed by a browser, aiming the execution of the said script with the target user's privileges. Once again, CAPEC-63 could be conducted without CAPEC-85 preceding it. However, the combination of the two is a valid scenario.

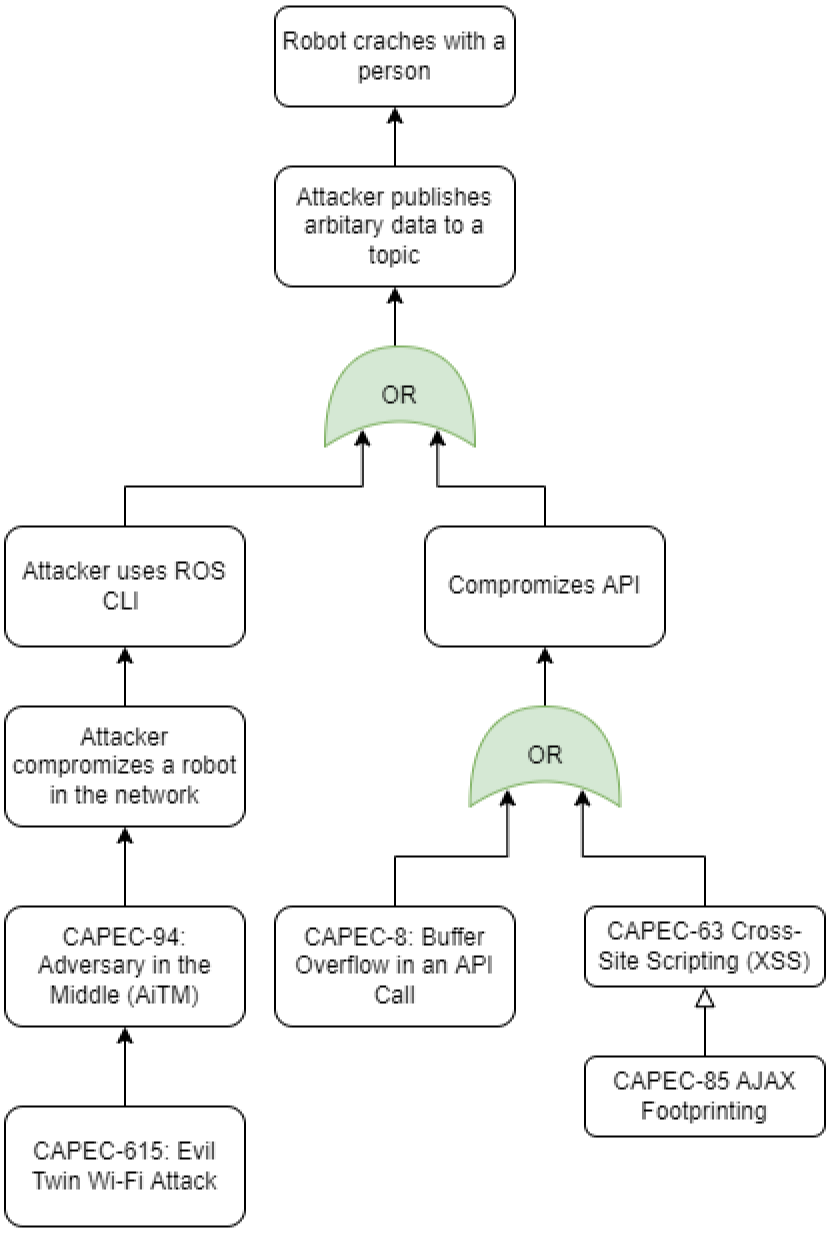

Robot craches with a person
Attacker publishes arbitary data to a topic
OR
Attacker uses ROS CLI
Compromizes API
Attacker compromizes a robot in the network
OR
CAPEC-94: Adversary in the Middle (AiTM)
CAPEC-8: Buffer Overflow in an API Call
CAPEC-63 Cross-Site Scripting (XSS)
CAPEC-85 AJAX Footprinting
CAPEC-615: Evil Twin Wi-Fi Attack

Figure 14: Attack tree created by a template attack tree and identified attacks for the Locomotec use case

The attack tree presented here, manages to combine attacks that were previously identified as potential, based on the vulnerabilities of the target system. Among the attacks included in the tree we have individual attacks along with one of the simple trees that are automatically generated based on the CanFollow and CanPrecede relationships of the CAPEC-IDs.

As it is mentioned in subsection 3.1.6, what follows the creation of the attack trees is the transfer of the included information to the run time in the form of security EDDI. At runtime security monitoring tools provide their output allowing for the initialization of mitigation actions.

4. CONCLUSIONS

This deliverable introduces to the reader the problem of performing security assessment to robotics systems. What makes it challenging is the fact that robotic systems of today operate in a whole new environment, opened to the external world, communicating with other systems, devices and services of doubtful confidence, operating in close proximity to humans or even interacting with them. Said environment is in contrast with the traditional industrial robot environment that used to be closed and trusted.

What follows is a presentation of the state-of-the-art techniques for security assessment, mentioning security assessment approaches that have been conducted on robotic systems, and security knowledge repositories that are used from these approaches. Among those repositories RVD is mentioned as a dedicated database for robotic-specific vulnerabilities.

Based on the above, trying to incorporate the state-of-the-art techniques and tools, the SESAME security assessment methodology is introduced. To prove the applicability of the proposed methodology, some preliminary outputs of the different methodology processes are presented, taking under consideration the Robot Operating System, key software for the Locomotec use case.

5. REFERENCES

[1] Ruffin White, Dr Christensen, I Henrik, Dr Quigley, et al. SROS: Securing ROS over the wire, in the graph, and through the kernel. *arXiv preprint arXiv:1611.07060*, 2016.

[2] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. ROS: an open-source Robot Operating System. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[3] David D Mascarenas, Jarrod McClean, Christopher J Stull, and Charles R Farrar. A Preliminary Cyber-Physical Security Assessment of the Robot Operating System (ROS). Technical report, Los Alamos National Lab (LANL), Los Alamos, NM (United States), 2013.

[4] Davide Quarta, Marcello Pogliani, Mario Polino, Federico Maggi, Andrea Maria Zanchettin, and Stefano Zanero. An Experimental Security Analysis of an Industrial Robot Controller. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 268–286. IEEE, 2017.

[5] International Organization for Standardization (ISO). Robots and robotic devices—safety requirements for industrial robots—part 2: Robot systems and integration, 2011.

[6] https://www.youtube.com/watch?v=yFi7UL70zTo&ab_channel=TUBerlin-IndustrielleAutomatisierungstechnik, (accessed December 18, 2021).

[7] Siegfried Hollerer, Clara Fischer, Bernhard Brenner, Maximilian Papa, Sebastian Schlund, Wolfgang Kastner, Joachim Fabini, and Tanja Zseby. Cobot attack: a security assessment exemplified by a specific collaborative robot. *Procedia Manufacturing*, 54:191–196, 2021.

[8] Gelei Deng, Yuan Zhou, Yuan Xu, Tianwei Zhang, and Yang Liu. An investigation of byzantine threats in multi-robot systems. In *24th International Symposium on Research in Attacks, Intrusions and Defenses*, pages 17–32, 2021.

[9] Global Times. Mainframe malfunction causes dozens of drones to crash into building in SW China. https://www.globaltimes.cn/page/202101/1214165.shtml, 2021 (accessed December 18, 2021).

[10] RS Ross. Guide for conducting risk assessments NIST special publication 800-30 revision 1. *US Dept. Commerce, NIST, Gaithersburg, MD, USA, Tech. Rep*, 2012.

[11] Patrick Engebretson. *The basics of hacking and penetration testing: ethical hacking and penetration testing made easy*. Elsevier, 2013.

[12] Fabián Cuzme-Rodríguez, Marcelo León-Gudiño, Luis Suárez-Zambrano, and Mauricio Domínguez Limaico. Offensive security: Ethical hacking methodology on the web. *In Conference on Information Technologies and Communication of Ecuador*, pages 127–140. Springer, 2018.

[13] Gurpreet K Juneja. Ethical hacking: A technique to enhance information security. *International Journal of Innovative Research in Science, Engineering and Technology*, 2(12):7575–7580, 2013

[14] Hilary Berger and Andrew Jones. Cyber security & ethical hacking for SMEs. *In Proceedings of the The 11th International Knowledge Management in Organizations Conference on The changing face of Knowledge Management Impacting Society*, pages 1–6, 2016

[15] Susmit Panjwani, Stephanie Tan, Keith M Jarrin, and Michel Cukier. An Experimental Evaluation to Detemine if Port Scans are Precursors to an Attack. In *2005 International Conference on Dependable Systems and Networks (DSN'05)*, pages 602–611. IEEE, 2005.

[16] Edwin Vattapparamban, Ismail Güvenç, Ali I Yurekli, Kemal Akkaya, and Selçuk Uluağaç. Drones for smart cities: Issues in cybersecurity, privacy, and public safety. In *2016 international wireless*

communi- cations and mobile computing conference (IWCMC), pages 216–221. IEEE, 2016.

[17] Yan Hu, Hong Li, Hong Yang, Yuyan Sun, Limin Sun, and Zhiliang Wang. Detecting stealthy attacks against industrial control systems based on residual skewness analysis. *EURASIP Journal on Wireless Communications and Networking*, 2019(1):1–14, 2019.

[18] Kwang Joon Yang, Jinho Choi, Seungsoo Lee, and Seungwon Shin. MilSeg: SDN-based Military Net- work Segregation Architecture. Technical report, EasyChair, 2019.

[19] CISCO. What is a firewall? https://www.cisco.com/c/en/us/products/security/firewalls/what-is-a-firewall.html, (accessed December 18, 2021)

[20] Syeda Fatimav, Shahid Abdul Sattar, Syeda Fatima, and Syed Adil. A study on intrusion detection. *International Journal of Advanced Research in Engineering & Technology*, 10, 03 2019.

[21] Gerald Post and Albert Kagan. Management tradeoffs in anti-virus strategies. *Information & Management*, 37(1):13–24, 2000

[22] Federico Maggi, Davide Quarta, Marcello Pogliani, Mario Polino, Andrea M Zanchettin, and Stefano Zanero. Rogue robots: Testing the limits of an industrial robot's security. *Trend Micro, Politecnico di Milano, Tech. Rep*, 2017.

[23] Victoria Drake. Threat Modeling. https://owasp.org/www-community/Threat_Modeling, (accessed December 18, 2021)

[24] Suvda Myagmar, Adam J Lee, and William Yurcik. Threat modeling as a basis for security requirements. In *Symposium on requirements engineering for information security (SREIS)*, volume 2005, pages 1–8. Citeseer, 2005.

[25] Thomas Moulard, Juan Hortala, Xabi Perez, Gorka Olalde, Borja Erice, Odei Olalde, and David May- oral. ROS 2 Robotic Systems Threat Model. https://design.ros2.org/articles/ros2_threat_model.html, 2021 (accessed December 18, 2021).

[26] Adam Shostack. *Threat modeling: Designing for security*. John Wiley & Sons, 2014.

[27] Adi Karahasanovic, Pierre Kleberger, and Magnus Almgren. Adapting threat modeling methods for the automotive industry. In *Proceedings of the 15th ESCAR Conference*, pages 1–10, 2017.

[28] Rafiullah Khan, Kieran McLaughlin, David Laverty, and Sakir Sezer. STRIDE-based threat modeling for cyber-physical systems. In *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, pages 1–6. IEEE, 2017.

[29] Riccardo Scandariato, Kim Wuyts, and Wouter Joosen. A descriptive study of Microsoft's threat modeling technique. *Requirements Engineering*, 20(2):163–180, 2015.

[30] S Simeonova. Threat modeling in the enterprise, part 2: Understanding the process. *Security Intelligence*, 2016.

[31] Kim Wuyts and Wouter Joosen. Linddun Privacy Threat Modeling: a tutorial. *CW Reports*, 2015.

[32] FIRST. Common Vulnerability Scoring System version 3.1. https://www.first.org/cvss/ v3-1/cvss-v31-specification_r1.pdf, (accessed December 18, 2021).

[33] Nataliya Shevchenko, Timothy A Chick, Paige O'Riordan, Thomas P Scanlon, and Carol Woody. Threat modeling: a summary of available methods. Technical report, Carnegie Mellon University Software Engineering Institute Pittsburgh United . . . , 2018.

[34] CAIRIS. https://cairis.org/, (accessed December 18, 2021).

[35] Microsoft security development lifecycle threat modeling. https://www.microsoft.com/en-us/securityengineering/sdl/threatmodeling, (accessed December 18, 2021).

[36] Christian Schneider. Threatgile. https://threagile.io/, (accessed December 18, 2021).

[37] Tutamantic. https://www.tutamantic.com/, (accessed December 18, 2021).

[38] Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, Tadayoshi Kohno, Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, et al. Experimental security analysis of a modern automobile. In *2010 IEEE symposium on security and privacy*, pages 447–462. IEEE, 2010.

[39] David Mascarenas, Christopher Stull, and Charles Farrar. Escape and evade control policies for ensuring the physical security of nonholonomic, ground-based, unattended mobile sensor nodes. In *Unattended Ground, Sea, and Air Sensor Technologies and Applications XIII*, volume 8046, page 80460G. International Society for Optics and Photonics, 2011.

[40] Gabriel Vasconcelos, Rodrigo Miani, Vitor Guizilini, and Jefferson Souza. Evaluation of DoS attacks on commercial Wi-Fi-based UAVs. *International Journal of Computer Network and Information Security*, 11:212, 04 2019.

[41] Yuan Xu, Gelei Deng, Tianwei Zhang, Han Qiu, and Yungang Bao. Novel denial-of-service attacks against cloud-based multi-robot systems. *Information Sciences*, 576:329–344, 2021.

[42] Alberto Giaretta, Michele De Donno, and Nicola Dragoni. Adding salt to pepper: A structured security assessment over a humanoid robot. In *Proceedings of the 13th International Conference on Availability, Reliability and Security*, pages 1–8, 2018

[43] Yosef Ashibani and Qusay H Mahmoud. Cyber physical systems security: Analysis, challenges and solutions. *Computers & Security*, 68:81–97, 2017.

[44] MITRE. Common Vulnerabilities and Exposures. https://cve.mitre.org/cve/, (accessed December 18, 2021).

[45] NIST. National vulnerability database. https://nvd.nist.gov/, (accessed December 18, 2021).

[46] MITRE. Common Weakness Enumeration. https://cwe.mitre.org/, (accessed December 18, 2021).

[47] Carlos Cardoso Galhardo, Peter Mell, Irena Bojanova, and Assane Gueye. Measurements of the Most Significant Software Security Weaknesses. In *Annual Computer Security Applications Conference*, pages 154–164, 2020

[48] MITRE. Common Attack Pattern Enumerations and Classifications. https://capec.mitre.org/, (accessed December 18, 2021).

[49] Víctor Mayoral Vilches, Lander Usategui San Juan, Bernhard Dieber, Unai Ayucar Carbajo, and Endika Gil-Uriarte. Introducing the robot vulnerability database (rvd), 2021.

[50] Kenta Kanakogi, Hironori Washizaki, Yoshiaki Fukazawa, Shinpei Ogata, Takao Okubo, Takehisa Kato, Hideyuki Kanuka, Atsuo Hazeyama, and Nobukazu Yoshioka. Tracing CAPEC attack patterns from CVE vulnerability information using natural language processing technique. *In Proceedings of the 54th Hawaii International Conference on System Sciences*, page 6996, 2021.

[51] Jan Reich, Daniel Schneider, Ran Wei Rasmus Adler, Marc Zeller Tim Kelly, Ioannis Sorokos, Joe Guo, Georg Macher Christof Kaukewitsch, and Eric Armengaud. Digital Dependability Identities and the Open Dependability Exchange Meta-Model. https://deis-project.eu/fileadmin/user_upload/DEIS_D3.1_Specification_of_the_ODE_meta-model_and_documentation_of_the_fundamental_concept_of_DDI_PU.pdf, (accessed December 18, 2021).

[52] Jackson Wynn. Threat assessment and remediation analysis (TARA). https://www.mitre.org/sites/default/files/publications/pr-2359-threat-assessment-and-remediation-analysis.pdf, 2014 (accessed December 18, 2021).

[53] PILZ. White paper security. https://www.pilz.com/mam/pilz/content/uploads/wp_ security_en_2018_10.pdf, 2018 (accessed December 18, 2021).

[54] Gilbert Tang and Phil Webb. Human–robot shared workspace in aerospace factories. In *Human–Robot Interaction*, pages 72–79. Chapman and Hall/CRC, 2019.