



UNIVERSITÀ
DEGLI STUDI
FIRENZE



UNIVERSITÀ
DEGLI STUDI
DI PERUGIA

[iNSdAM]
Istituto Nazionale
di Alta Matematica

Università di Firenze, Università di Perugia, INdAM consorziate nel CIAFM

**DOTTORATO DI RICERCA
IN MATEMATICA, INFORMATICA, STATISTICA
CURRICULUM IN INFORMATICA
CICLO XXXVII**

Sede amministrativa Università degli Studi di Firenze
Coordinatrice Prof.ssa Alessandra Sestini

**Cybersecurity Assessment:
Challenges and Advancements around
the ADVISE Meta Framework**

Settore Scientifico Disciplinare INFO-01/A

Dottorando:
Francesco Mariotti

Tutore:
Prof. Paolo Lollini

Coordinatrice:
Prof.ssa Alessandra Sestini

Francesco Mariotti: *Cybersecurity Assessment: Challenges and Advancements around the ADVISE Meta Framework*, Dottorato di Ricerca in Matematica, Informatica e Statistica (Curriculum in Informatica), Ciclo XXXVII

ABSTRACT

A Cyber-Physical System (CPS) is a system consisting of a computer system (the cyber system), a controlled object (the physical system) and possibly interacting humans. In the past, CPSs were mainly designed to work in isolation. Nowadays, due to the increasing complexity and need for automation, such systems are becoming highly interconnected and remotely accessible for monitoring and control. Therefore, they are potentially exposed to malicious threats which could lead to catastrophic consequences. For this reason, cybersecurity is becoming extremely important for CPSs.

Cybersecurity assessment is fundamental for identifying possible threats that could maliciously cause damage to a system. In industry, there is an increasing interest in assessing the security of CPSs; see as an example the emergence of cybersecurity standards like ISA/IEC 62443 for industrial automation control systems, CENELEC CLC/TS 50701 for railway, and ISO/SAE 21434 for automotive. From an industry viewpoint, there are several needs that a cybersecurity assessment approach should take into account like the management of the complexity of the process, the possibility to apply the assessment from the initial stages of system development, and the reusability of the approach to perform analysis of different but similar systems.

Due to the complexity of such systems, experts need methodologies and tools to face this not-easy task. To help them evaluate the system's security, it is needed to have i) a (semi-)formal approach to represent and compare different system components and architectures; ii) a well-founded and reasonably large catalog of attacks; iii) a standardized collection of adversaries; iv) a way to represent and analyse possible defense mechanism and to capture system's safety and security interdependencies.

Among other cybersecurity assessment approaches, like penetration testing and threat analysis, model-based evaluation is well-known and used to analyse system's properties, such as safety, availability, reliability, and security. Thanks to its high level of abstraction it brings benefits like complexity management, early-stage application, and reusability. In this thesis, we rely on an existing modeling framework, ADVISE Meta, which, starting from an ontology (specifying system components, attacks, and adversaries) and a user-defined architectural model of the system, allows to automatically derive stochastic security models, specified in the ADVISE formalism). These models can be used to evaluate the attack paths that can be carried out by adversaries, analysing, e.g., the probability that an adversary can reach a specific security goal, and compare different architectural solutions.

Concerning the overmentioned needs, we identified the following research challenges around the ADVISE Meta framework: i) the attacks and adversaries available in the ontology of the framework are few and generic; ii) only the adversary point of view is considered, without taking into account possible defense mechanisms and system dynamics.

In this thesis, we propose some contributions to face these challenges. First, we define a methodology to extend the ADVISE Meta ontology with a well-founded collection of attacks and adversaries' profiles coming, respectively, from the CAPEC database and the Threat Agent Library (TAL). Moreover, since the manual integration of the CAPEC database is a time-consuming task, we propose

to use artificial intelligence as a support tool for the semi-automated integration of CAPEC attacks into the ADVISE Meta ontology. Then, to overcome the fact that the ADVISE formalism only takes into account the viewpoint of the adversary, we rely on a more flexible modeling formalism, that is Stochastic Activity Networks (SAN). Hence, we perform a model-to-model transformation from ADVISE to SAN formalism. This facilitates the composition with other SAN models, which can be used to represent defense mechanisms and system aspects, e.g., to analyse the interdependencies between safety and security. Moreover, with SAN, compared to ADVISE, we can represent additional attack scenarios, like simulating the execution of multiple attacks in parallel.

ACKNOWLEDGEMENTS

First of all, special thanks go to my supervisor, Prof. Paolo Lollini, for wisely guiding me during this three-year journey.

Then I want to thank all the other members of the Resilient Computing Lab research group (past and present) for their cooperation: Prof. Andrea Bondavalli, Prof. Andrea Ceccarelli, Prof. Tommaso Zoppi, Prof. Leonardo Montecchi, Lorenzo Sarti, Tommaso Puccetti, Francesco Terrosi, Marzieh Kordi, Muhammad Atif and Fahad Ahmed Khokhar.

My gratitude also goes to the PERFORM Group at the University of Illinois Urbana-Champaign, for providing me access to the ADVISE Meta framework, which constitutes the core of this thesis.

I thank all the bachelor's and master's students who collaborated with me on the research topics presented in this thesis, particularly Matteo Tavanti, Roberto Magrini, and Matteo Luddi.

Last but not least, I want to thank my family and friends for supporting and encouraging me during this adventure.

CONTENTS

List of Figures	7
List of Tables	9
1 Introduction	13
1.1 Motivation	13
1.2 Approaches for cybersecurity assessment	14
1.2.1 Vulnerability identification and analysis	15
1.2.2 Penetration testing	15
1.2.3 Threat analysis	15
1.2.4 Model-based evaluation	16
1.2.5 Reviews	16
1.3 Research direction: ADVISE Meta and its challenges	16
1.4 Contributions of the thesis	17
2 Background	21
2.1 ADVISE	21
2.2 ADVISE Meta Framework	22
2.3 SAN	26
2.4 Common Attack Pattern Enumeration and Classification (CAPEC) .	27
2.5 Threat Agent Library (TAL)	29
3 An extension of the ADVISE Meta Modeling Framework	31
3.1 Introduction	31
3.2 Related Work	32
3.3 Methodology: Linking CAPEC, TAL and ADVISE Meta Elements .	34
3.3.1 From CAPEC Sections to TAL Attributes	34
3.3.2 From TAL Attributes to ADVISE Meta Elements	36
3.3.3 From CAPEC Sections to ADVISE Meta Elements	37
3.4 Extension of ADVISE Meta Ontology	37
3.4.1 Extension of ADVISE Meta with TAL Adversaries' Profiles .	38
3.4.2 Extension of ADVISE Meta with CAPEC Attacks	39
3.5 Case Study: SPaCe System	42
3.6 Security Analysis of the SPaCe System	43
3.6.1 Objectives of the Analysis	44
3.6.2 Modeling Process	44
3.7 Analysis Scenarios and Results	50
3.7.1 Analysis at Varying of Adversaries' Profiles	50
3.7.2 Analysis at Varying of the Attack Step Success Probability .	53
3.7.3 Analysis at Varying of Implementation Characteristics of	
the System	53
3.7.4 Analysis of Most Probable Attack Paths	54
3.7.5 Analysis of the Exposure Level of System's Components . .	55
3.8 Conclusions	56
4 Semi-automated integration of CAPEC attacks into ADVISE Meta ontology	59
4.1 Introduction	59

4.2	Related work	60
4.3	Methodology for the integration of CAPEC attacks	61
4.3.1	Step 1: Preliminary Operations	63
4.3.2	Step 2: Data Extraction	63
4.3.3	Step 3: Creation of the Attack Steps	69
4.3.4	Step 4: Assigning the Attack Steps to the Architectural Components	73
4.3.5	Step 5: Final Insertion	76
4.4	Updating the Ontology	76
4.4.1	Check for updates	76
4.4.2	Perform the update	77
4.5	Discussion	77
4.5.1	Usage of AI	77
4.5.2	Validation of the results	77
4.5.3	Using other attack databases	78
4.6	Conclusions	78
5	Model-to-Model Transformation: from ADVISE to SAN	79
5.1	Introduction	79
5.2	Related work	80
5.2.1	Model transformation	80
5.2.2	XML parsing and transformation	80
5.3	Mapping between ADVISE and SAN elements	81
5.3.1	Skill, Access, Knowledge, and Goal	82
5.3.2	Attack Step	82
5.3.3	Arc	83
5.3.4	Execution algorithm	83
5.4	Correspondence between XML representations	84
5.5	Implementation of the transformation	87
5.5.1	Transformation algorithm	87
5.5.2	Python implementation	88
5.6	Conclusions	91
6	Case study: the 2015 Ukraine power outage	93
6.1	Description of the cyber attack	93
6.2	Modeling the Ukraine attack using ADVISE Meta	94
6.2.1	Selection and modification of the CAPEC attacks	95
6.2.2	Models definition	100
6.3	Transformation of the generated ADVISE model to SAN	101
6.3.1	Comparative analysis	102
6.3.2	Defense mechanisms	104
6.4	Conclusions	108
7	Conclusions and Future Work	109
7.1	Summary of the contributions	109
7.2	Open challenges	110
7.3	Future research directions	111
	Bibliography	113

LIST OF FIGURES

Figure 1	Schematic view of the main contributions of the thesis. . .	19
Figure 2	A simple ADVISE model.	22
Figure 3	TAL adversaries profiles.	30
Figure 4	Relationships between CAPEC sections, TAL attributes, and ADVISE Meta elements.	35
Figure 5	TAL “Terrorist” adversary’s profile visualized in the ADVISE Meta framework.	39
Figure 6	“Flooding” attack step added to the ontology of ADVISE Meta.	41
Figure 7	DMZ architecture for the SPaCe system	44
Figure 8	Architectural model (SID) of SPaCe system created with ADVISE Meta framework.	46
Figure 9	ADVISE model generated by the tool for the Terrorist adversary.	49
Figure 10	Probability of successfully achieving the <i>LANUseIntercepted-Data</i> goal for the Terrorist adversary.	52
Figure 11	Probability of successfully achieving goals for the Terrorist adversary as the user authentication level varies.	54
Figure 12	Simplified ADVISE model for the Terrorist adversary with the <i>ServerCILogicalAccess</i>	55
Figure 13	Exposure level of SPaCe components for Vandal and Terrorist adversaries.	56
Figure 14	Methodological steps for the semi-automated integration of CAPEC attacks into ADVISE Meta ontology.	62
Figure 15	Mapping between ADVISE and SAN elements.	82
Figure 16	Simplified view of the architecture of the Ukrainian power grid control system.	94
Figure 17	The InterfaceManipulation attack step details as visualized into the ADVISE Meta framework.	99
Figure 18	System Instance Diagram (SID) of the Ukraine power grid control system.	100
Figure 19	Generated ADVISE model for the Ukraine attack.	101
Figure 20	Transformed SAN model for the Ukraine attack.	102
Figure 21	Probability of successfully reaching the adversaries’ goals for the Ukraine attack, simulated in ADVISE and SAN. . .	103
Figure 22	SID of the Ukraine power grid with three redundant UPSs.	106
Figure 23	SAN results for the probability of successfully disabling the UPS service: configuration with one, two, and three UPSs.	106
Figure 24	SAN model of a generic system composed of three redundant servers, using the proactive obfuscation technique. . .	107

LIST OF TABLES

Table 1	Comparison of the main cybersecurity assessment approaches.	14
Table 2	Assignment of numerical values to TAL attributes.	38
Table 3	Setting of the adversaries used in the experiments.	50
Table 4	Probability of successfully achieving goals as the adversaries' profiles vary.	51
Table 5	Comparison between different XML parsing approaches.	82
Table 6	Comparison of the simulation results of ADVISE and SAN models for the Footprinting attack step	104

LISTINGS

Listing 1	Prompt for the definition and values assignment of TAL attributes and attack steps' fields.	65
Listing 2	Answer of ChatGPT to the description and assignment of TAL attributes and attack steps' fields.	66
Listing 3	Prompt for the interpretation of "CAPEC-125: Flooding" attack pattern.	68
Listing 4	ChatGPT interpretation of "CAPEC-125: Flooding" attack pattern.	68
Listing 5	Attack step template.	70
Listing 6	Prompt with the instructions for placeholders' replacement in the attack step template.	71
Listing 7	Example of corrupted ChatGPT answer concerning the attack step creation.	72
Listing 8	Prompt for the creation of attack steps.	72
Listing 9	Example of generated attack step.	72
Listing 10	Prompt for the definition of the architectural components.	74
Listing 11	Prompt for the the assignment of the attacks to the architectural components and creation of the labels.	75
Listing 12	An example of ChatGPT answer concerning the assignment of attacks to the components	75
Listing 13	Example of XML representation of an Attack Step in ADVISE.	84
Listing 14	Example of XML representation of an Attack Step in SAN (input gate).	85
Listing 15	Example of XML representation of an Attack Step in SAN (timed activity).	85
Listing 16	Example of XML representation of an Attack Step in SAN (output gate).	86
Listing 17	Python code for parsing of the ADVISE XML file.	88
Listing 18	Python code for the creation of the Input Gate.	88
Listing 19	Python code for the creation of the Timed Activity.	89
Listing 20	Python code for the creation of the Output Gates.	90

INTRODUCTION

In this chapter, we introduce the motivations that push industry and research to invest in the assessment of cybersecurity. We also briefly analyze some of the possible approaches for cybersecurity assessment. Finally, we present a research direction around model-based cybersecurity evaluation and illustrate the main contributions of the thesis. Part of the chapter is based on [1].

1.1 MOTIVATION

A Cyber-Physical System (CPS) [2] is a system consisting of a computer system (the cyber system), a controlled object (the physical system) and possibly interacting humans. Examples of CPSs are transportation systems, power grids, or industrial systems. Since such systems interact with the physical world, their failure can result in catastrophic consequences. Historically, CPSs were mainly designed to work in very isolated environments, relied on simple components, e.g., PLCs, and communications with the external world were limited and scarce. Nowadays, in the context of Industry 4.0, there is a great interest in automation [3]; remotely controlled or autonomous vehicles, smart grids, and industrial automation control systems (IACS) are just examples of the new generation of CPSs.

Such systems need a large number of connected cyber components to be automated or remotely controlled. In such complex environments, adversaries can exploit internal and external malicious threats to damage the system, potentially leading to catastrophic consequences. For this reason, cybersecurity is becoming extremely important even for CPSs. The industry is starting to perceive the need to define standardized processes and approaches for the design, implementation, and maintenance of secure CPSs [4]. As an example, note the emergence in the last years of cybersecurity standards like the ISA/IEC 62443 series of [5] for the design and assessment of IACS, the ISO/SAE 21434 [6] for automotive, and the CLC/TS 50701 [7] for railway.

Cybersecurity assessment can be generally seen as the process of determining the present cybersecurity posture of an information system and evaluating the fulfillment of security objectives [8]. This is done by identifying and analysing the system's vulnerabilities, the possible adversaries that could menace the system, the threats and attacks that can be exploited by the adversaries, the consequences of such threats over the system, and the countermeasures or security defenses against those threats. Since such systems are often complex, usually it is not possible to demonstrate that a system is completely secure, but it is possible to say that it is reasonably secure by collecting evidence and performing detailed analysis.

From an industry point of view, there are several corporate needs that a cybersecurity assessment approach should take into account. First, the approach should *manage the complexity* of the cybersecurity assessment process. Cost and time are fundamental aspects for companies, thus it is important to use approaches as efficiently as possible. Then, Security by Design is increasingly becoming the mainstream development approach to ensure the security and privacy of software systems. For this reason, it is important to consider approaches that can be used since the *initial stages of system development*. Lastly, the *reusability* of the approach should be taken into account, since it might be necessary to perform the assessment of different but similar systems.

Security design and assessment are especially difficult during the initial system development phases. In the early stage, the only thing that is known is the possible architecture of the system. The specific vulnerabilities, adversaries, threats, and attacks are not known. Due to the lack of details, experts need methodologies and tools to help them with this not-easy task. From an industry viewpoint, the following aspects would provide fundamental support for early-stage security assessment:

- an approach to represent and compare different system components and architectures, e.g., to understand which is the most secure architectural solution;
- a well-founded and reasonably large catalog of attacks, that could be used to perform broad-range analysis, compare and evaluate different attack paths, and understand which of them are the most dangerous for the specific system;
- a standardized collection of adversaries, for comparing and evaluating different profiles which could attempt to attack the system;
- a way to represent and analyse countermeasures and defense mechanisms, and to capture possible interdependencies between safety and security.

1.2 APPROACHES FOR CYBERSECURITY ASSESSMENT

Cybersecurity assessment is commonly performed following different approaches [8]. Among others, we can mention vulnerability identification and analysis, penetration testing, threat analysis, model-based evaluation, and reviews. In [Table 1](#) these approaches are compared according to the main industry needs, which are discussed in the subsequent sections.

Table 1: Comparison of the main cybersecurity assessment approaches.

Cybersecurity assessment approach	Management of complexity	System development stage	Reusability
Vulnerability identification and analysis	Low	Late	Low
Penetration testing	Low	Late	Low
Threat analysis	Low-Medium	Any (Initial)	Medium
Model-based evaluation	Medium-High	Any (Initial)	High
Reviews	Low	Any (Late)	Low

1.2.1 *Vulnerability identification and analysis*

Vulnerability identification and analysis aims at discovering well-known or brand-new flaws, i.e., vulnerabilities, which can lead to security incidents. This activity is usually supported by automated tools that can scan hardware and software components and return a list of vulnerabilities. Known lists of vulnerabilities are contained in catalogs like the Common Vulnerabilities and Exposures (CVE) [9]. The Common Vulnerability Scoring System (CVSS) [10], instead, provides a way to capture the principal characteristics of a vulnerability and produce a numerical score reflecting its severity, thus can be used also to evaluate newly discovered vulnerabilities.

Identifying and analyzing the vulnerabilities of a complex system, like a CPS, can become quite a tough activity. Since vulnerabilities differ from component to component, it is often difficult to reuse such evaluations. Unless the specific components are known from the beginning, vulnerability analysis can be done only in advanced stages of development.

1.2.2 *Penetration testing*

Penetration testing [11] is a comprehensive method to test the complete, integrated, operational, and trusted environment that consists of hardware, software, and people. The process, which can be a white box or black box, involves an active analysis of the system for any potential vulnerabilities, including hardware and software flaws, operational weaknesses, and poor or improper system configurations. The penetration testing activity in part simulates the concrete actions that an adversary would perform in order to carry out an attack. Penetration testing activities involve the usage of tools such as packet sniffers, port scanners, password crackers, and vulnerability and exploitation frameworks.

The same considerations that we made for vulnerability identification on complexity management, reusability, and development stage hold for penetration testing.

1.2.3 *Threat analysis*

Threat analysis is the practice of identifying possible threats that can menace the system and determining their consequences and impact on the system, as well as the probability that those threats will happen. In the literature, some methodologies exist that encourage experts to reason, identify, and analyze possible threats to the system according to different categories. Examples of such methods are STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege) [12] and DREAD (Damage, Reproducibility, Exploitability, Affected users, Discoverability) [13]. Once the threats are identified, severity and likelihood can be associated to them.

When considering specific threats, the complexity of the approach can quickly explode. However, if threats are grouped into categories, the approach is more manageable and allows for some reusability. Threat analysis can be performed

from the early stages of system development, but the more information available, the more accurate the analysis will be.

1.2.4 *Model-based evaluation*

Model-based evaluation makes use of models to obtain predicted measures by calculation or simulation [14]. Models are useful for explaining why and how certain features of the system actually occur and help in developing insights on the operation of the system and in understanding the directions of the influence that certain input parameters may have on the results [15]. In the past models were mainly used to measure dependability properties like safety, availability, and reliability [16]. More recently, model-based evaluation has started to be used to analyse security problems [17]. For attack and defense modeling, graph-based models are often used [18]. For a deeper analysis of model-based security evaluation approaches, we defer to [section 3.2](#).

Modeling allows to possibly represent and analyse aspects of the system at different abstraction levels. This means that model-based approaches can be used from the initial phases of development and allow the representation of complex systems. However, the more details are considered, i.e., the more the abstraction level is low, the more the approach will be complex. Depending on the abstraction level, models are often reusable for representing and analysing different but similar systems or configurations.

1.2.5 *Reviews*

Reviews involve passive, usually manual, examinations of documentation related to the assessed object [8]. The documentation commonly reviewed during security assessments includes technical specifications, logs, rules, and configuration files.

The amount of documentation produced during the late stages of development and during the operation of complex systems is very high, so their examination is often a time-consuming task. The documentation is also often tailored for the specific system, so it is difficult to reuse it.

1.3 RESEARCH DIRECTION: ADVISE META AND ITS CHALLENGES

As discussed in [section 1.2](#), model-based evaluation is a high-abstraction approach that brings benefits like complexity management, early-stage application, and reusability. The ADversary VView Security Evaluation (ADVISE) [19, 20] is a model-based evaluation formalism developed and integrated into the Möbius modeling tool [21]. It allows to define security models that can represent the attacks carried out by an adversary on a system. The ADVISE modeling formalism is constituted of an attack execution graph (AEG), defining the attack steps that can be followed to reach defined adversaries' goals and an adversary profile that describes an attacker's profile according to some attributes. A Markov Decision Process algorithm [19], called ADVISE execution algorithm, is then used to simulate the behavior of the adversary in deciding what attack path to follow.

Security experts and modelers have to manually define ADVISE models, which can be a complicated and time-consuming process, especially for large systems. To address this challenge, the ADVISE Meta framework has been introduced in [22]. This framework presents a high-level model, called System Instance Diagram (SID), which can be used to represent the architecture of a system based on some built-in blocks defined in the ontology of the framework. By leveraging the ontology, which encompasses attack steps, adversaries' profiles, and metrics, a series of ADVISE models can be generated automatically.

The approach adopted by the ADVISE Meta framework poses the following challenges:

- The attacks and adversaries' profiles available in the base ontology of the framework are quite generic and limited in number. Hence, this ontology does not fully cover the broad range of attacks and attackers that we can find in real-world scenarios.
- The simulation of the adversary's behavior is constrained by the execution algorithm integrated into ADVISE formalism, which brings several assumptions on the possible attacker's behavior. For instance, the algorithm forces to pursue the attack path that minimizes cost for the adversary. In reality, the adversary might choose the attack path to follow based on various factors, such as choosing the shortest available attack path. Another assumption linked to the ADVISE execution algorithm is that an adversary can try to exploit only one attack path at a time, while in a real scenario, an adversary could also trigger more parallel attacks at the same time.
- Only the adversary viewpoint is considered in ADVISE Meta framework and ADVISE formalism without considering security defense mechanisms, and the potential consequences of attacks on the system.

1.4 CONTRIBUTIONS OF THE THESIS

To face the challenges around the ADVISE Meta framework enumerated in [section 1.3](#), in this thesis we propose the following main contributions (also depicted in [Figure 1](#)):

1. **ADVISE Meta ontology extension.** In [chapter 3](#), we propose a methodology for the extension of the ADVISE Meta framework to integrate the ontology with (i) the attack patterns from the Common Attack Pattern Enumerations and Classifications (CAPEC) [23] by MITRE, which is a database publicly available with more than five hundred common attack patterns; and (ii) adversaries' profiles from the Threat Agent Library (TAL) [24], developed by Intel, which offers a description of the human agents capable of posing threats to IT systems. In this first step, we manually integrate into the framework all the twenty-one TAL adversaries' profiles and a few representative CAPEC attacks.
2. **Semi-automated integration of CAPEC attacks.** In [chapter 4](#), since the manual integration of the attacks covered by the first point is a time-consuming

activity, we propose a semi-automated approach supported by Artificial Intelligence (AI) to integrate the CAPEC attack patterns into the ADVISE Meta ontology.

3. **Model-to-model transformation from ADVISE to SAN.** To overcome the assumptions made by the ADVISE formalism, we decided to rely on a more flexible modeling formalism, that is Stochastic Activity Networks (SANs) [25, 26]. Hence, in [chapter 5](#), we propose a model-to-model transformation from ADVISE to SAN models. This transformation provides the following benefits:
 - We get free from the assumptions of the ADVISE execution algorithm, e.g., we can simulate how the adversary behaves while pursuing multiple attack paths simultaneously, without the constraint of choosing according to the decision parameters, e.g., the attack’s cost.
 - We facilitate the composition of the attack model with other models representing defense mechanisms and/or the system itself, to capture safety and security interdependencies.

A first application of the extended ontology is provided in [chapter 3](#). In particular, we consider a supervision system in the public transport domain and we perform broad-range evaluations like analysis at the varying of adversaries’ profiles, analysis at the varying of implementation characteristics of the system, or analysis of the most probable attack paths.

Then, an application of the modeling approach to the 2015 Ukraine power outage case study is provided in [chapter 6](#). We use this case study to show the benefits of i) the semi-automated integration of the CAPEC attacks and ii) the transformation from ADVISE models to SAN models. The first part allows for a well-founded base of attacks that can be used in the models. The second part, instead, allows to capture some behaviors that are not considered by the ADVISE formalism, like simulating the parallel execution of attacks, and to possibly take into account defense mechanisms.

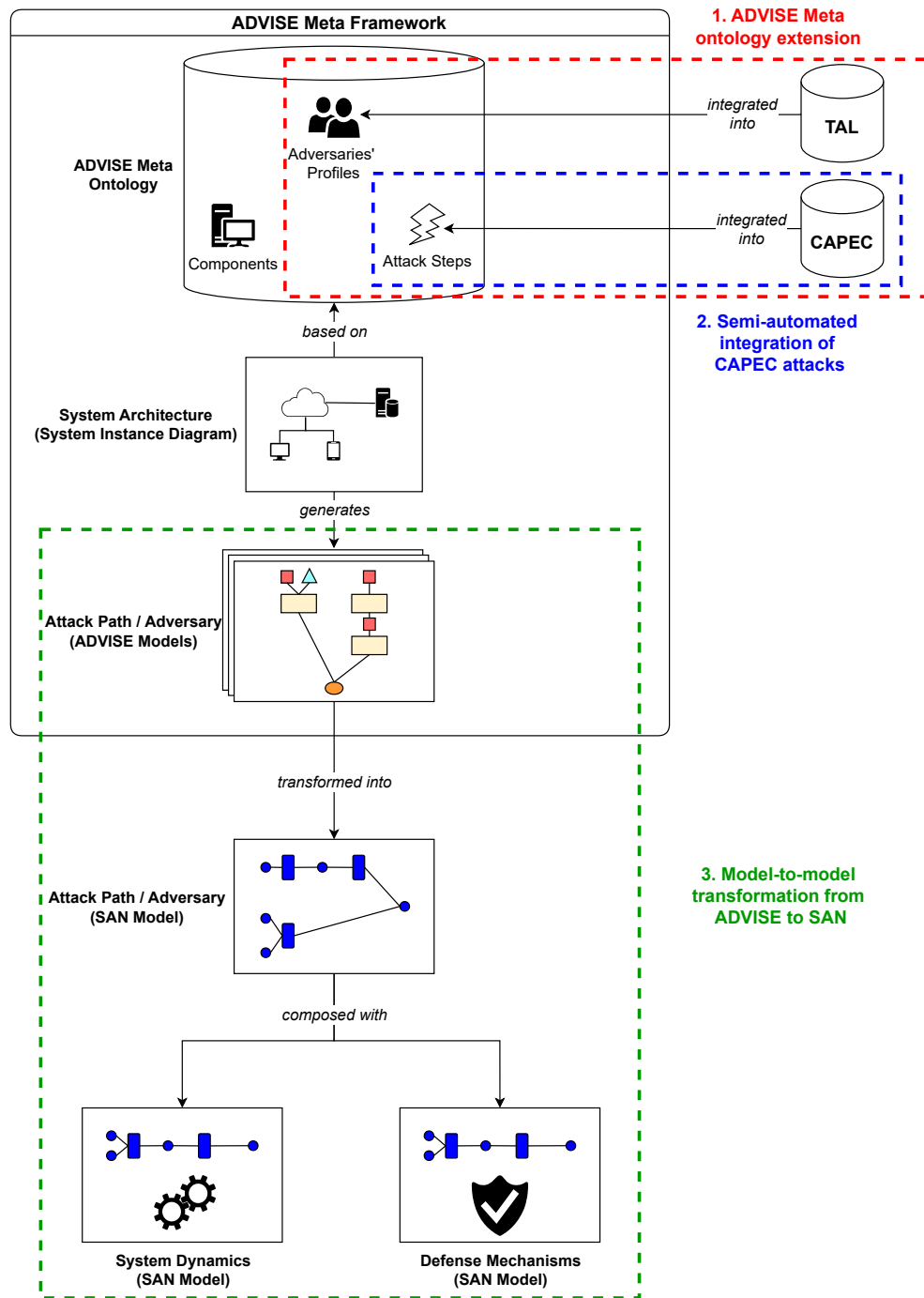


Figure 1: Schematic view of the main contributions of the thesis.

BACKGROUND

In this chapter, we give a brief description of the core elements used in the rest of the thesis, namely ADVISE, ADVISE Meta, SAN, CAPEC, and TAL. Part of this chapter is based on [27, 28].

2.1 ADVISE

ADVISE (ADversary View Security Evaluation) [19][20] is a stochastic security modeling formalism that allows to model adversaries and attack steps, aiming to analyse the probability that an adversary can achieve a certain goal, and the required effort. A model developed with ADVISE consists of two parts: the *Attack Execution Graph* (AEG) and the *Adversary Profile*.

The AEG describes the actions that an adversary has to follow to reach a certain goal, and it consists of items of five basic types. The *Skill*, *Knowledge*, and *Access* items play the role of requirements (or prerequisites) for executing an attack, i.e., items that must be held by an adversary for enabling the execution of a specific attack. These items can also be gained as a result of a successful execution of an attack step. An *Attack Step* item represents a single step of an attack that can have different *Outcomes* (e.g., success or failure), with the associated *Outcome Probability*. An attack step also has an *Attack Execution Time*, a *Probability of Detection*, and a *Precondition Expression*. A *Goal* is an objective that the adversary wants to reach (like the logical access to a server). These elements are represented graphically in the ADVISE Editor as depicted in [Figure 2](#).

In the Adversary Profile, we can define the profile of an attacker through the specification of a set of attributes, whose values determine if a particular adversary can reach a specific goal. Those attributes are: *name*, *decision parameters* (planning horizon, attack preference weights), *skills*, *initial access*, *initial knowledge*, and *goals*.

ADVISE is implemented as an atomic formalism in the Möbius framework [21][29], which integrates the ADVISE execution algorithm [19] for simulating the adversary's behavior. The algorithm is based on Markov Decision Processes (MDPs), and consists of two steps repeated cyclically: i) selection of the optimal attack step to be attempted next, and ii) simulation of its outcome.

The adversary starts from an initial state with a defined set of skills, access, and knowledge. The attack execution graph determines possible attack paths. The adversary selects the most attractive next attack step by evaluating based on a weighted sum of the *Cost of execution*, *Payoff* (success potential,) and *Probability of Detection*. When the adversary attempts an attack step for which it has the prerequisites (in terms of skills, knowledge, and access), the outcome (success or

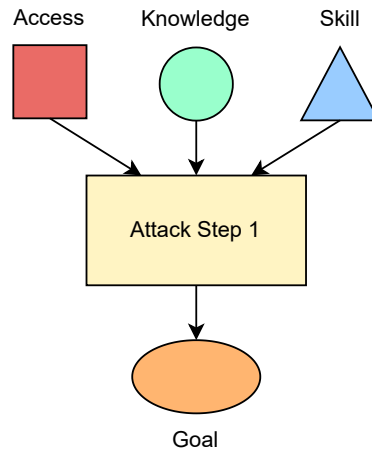


Figure 2: A simple ADVISE model.

failure) is determined stochastically. A long-range-planning adversary considers multiple attack steps in advance before making a decision. This is modeled using a State Look-Ahead Tree (SLAT), which explores all possible future attack paths up to a *Planning Horizon* N . The adversary computes attractiveness recursively for each path of the SLAT over N steps. Once the tree is fully built, branches with non-optimal attack sequences are pruned. The adversary then selects the first attack step of the best long-term attack path. The process repeats until the adversary reaches a goal or decides to stop.

The ADVISE execution algorithm makes use of several assumptions concerning the models' parameters, like the cost of execution of an attack and the probability of detection. This is something that is not always true in the real world, as an adversary might ignore such constraints. Moreover, ADVISE evaluates the possible different attack paths but the attacks are executed in sequence. In a real attack scenario, different attacks could be potentially carried on in parallel. These are some of the motivations that led us to implement the model-to-model transformation from ADVISE to SAN described in [chapter 5](#).

2.2 ADVISE META FRAMEWORK

ADVISE Meta [22, 30] is a meta-level modeling framework that has been proposed to mitigate the complexity of building ADVISE models by hand, which can be a very difficult and time-consuming task. In the framework, the architectural model of the system is built at a higher abstraction level (i.e., at the meta-level), and the low-level ADVISE models are derived automatically. The system is described using generic built-in blocks and relationships that constitute the ontology of the framework, and that embed information on possible attacks in their definition.

The ontology consists of the following elements; those marked with an asterisk (*) are also present in the plain ADVISE formalism:

- **Component:** it defines a base category of elements that can be part of a system. Some examples: *Device, OperatingSystem, Network*.
- **Relationship:** it defines a kind of relation that may exist between two components. Note that a relationship only applies to the specific kinds of components for which it is defined. Some examples: *onNetwork, storageDevice, canDamage*.
- **Attribute:** it represents a characteristic of a component, and can be used as a parameter of the attack steps attached to the component. Some examples: *dataIntegrityControl, mediaPortEnabled, userAuthenticationType*.
- **Access*:** it defines an access that an adversary may have at the beginning of an attack, or which may be gained during the attack. Some examples: *InsiderAccess, LogicalAccess, PhysicalAccess*.
- **Skill*:** it defines a skill that an adversary may have in varying degrees of proficiency. Some examples: *BasicCyberOffense, Cryptanalysis, NetworkPenetration*.
- **Knowledge*:** it defines something that the adversaries may know beforehand, or that it may be acquired during the attack. The ADVISE Meta ontology from [22] includes only one knowledge: *FirewallRulesetKnowledge*.
- **System State Variable:** it can be used to define state variables related to system components, which are also typically used to define adversary attack goals. Some examples: *Damaged, Disabled, MalwareInstalledOn*.
- **Attack Step*:** it defines a step of an attack that can be performed by an adversary. Some examples: *PhysicalDisable, GainUserCredentials, ModifyDataLocally*.
- **Adversary*:** it defines an adversary's profile with several characteristics. Some examples of built-in adversary templates: *ForeignGovernment, HackerGroup, OrganizedCrime*.
- **Metric:** only one metric (*goalAchieved*) is defined in the base ontology from [22], but other metrics can be added.

The modeling process using ADVISE Meta consists of: i) adding the components that are part of the system into the System Instance Diagram (SID) and setting the corresponding attributes; and ii) connecting the components through the available relationships, based on the system architecture.

The components available in the base ontology of the framework are the following:

- **Application:** software designed for the end user that performs a function in the system.

- **Data:** information in digital format that is created, stored or processed by the system. Examples are substation equipment settings, patient records, and bank account information.
- **Device:** physical thing that performs a function in the system. These may be typical system components such as workstations and servers, or cyber-physical components such as pacemakers or weapons.
- **Host:** device that hosts a general-purpose operating system.
- **Firewall:** device that controls communication between networks. It must reside on at least two networks in a system model. A firewall may be an appliance or a hosted firewall.
- **FirewallAppliance:** firewall that is purchased on its operating hardware and does not support other uses of an underlying general-purpose operating system. This is distinguished from FirewallHosted because appliances are typically less susceptible to common operating system attacks.
- **FirewallHosted:** firewall that is purchased as software and installed on a general-purpose operating system.
- **Network:** physical and logical infrastructure used by devices to communicate. May designate any type of network, such as local area network or wide area network. Often the networks specified in a system model represent local area networks, and the Internet. A network may be wired or wireless.
- **OperatingSystem:** software that supports a computer's basic functions. Examples would be a specific installation of Windows 10 or Linux.
- **Server:** a host that provides a centralized resource for data or functions, accessed over a network on which it resides.
- **Software:** a placeholder term not envisioned to be used to create instances in system models. Software is the parent of Application and OperatingSystem, which inherit attributes and attack steps from their parent.
- **PhysicalThing:** entity with a location in the space dimension, and subject to physical damage. Examples are devices, networks, backup tapes, and building facilities.
- **System:** a placeholder term not envisioned to be used in system models, for the parent of Software and Device. However, these sub-classes inherit attack steps and attributes from their parent System. Note that although the term "system" is also used to refer to an entire "system model," there is no connection between these two uses of the term.
- **WiredNetwork:** network implemented by using wires or cables to transmit communications between devices on the network.

- **WirelessNetwork** network implemented without using wires or cables for communication between devices on the network. Usually implemented using radio waves.
- **Workstation**: host supporting the interaction of a human user with application functions.

Once the model has been defined, the low-level model (i.e., ADVISE model) is generated, based on the definition of the particular configuration of the system to be analyzed, i.e., one system diagram, one adversary, and a subset of available metrics [22]. The transformation algorithm loops through each model fragment instance in the SID, identifying the component instance associated with each model fragment. For each variable macro in a model fragment, the algorithm finds the relevant state variables by traversing the SID and explores the model structure based on path expressions that describe relationships between components. Once the required state variables are identified, the algorithm applies modifications specific to the modeling formalism being used (e.g., ADVISE). These modifications might involve adding new arcs or updating model parameters. Other modeling elements are also automatically generated, including performance variable reward models, set studies, and the simulator.

The attacks generated in the ADVISE model depend on the components (and their attributes) and the relationships between them, so if a particular component is not present in the SID model, the related attacks will not be derived.

The ADVISE Meta framework (and the generated ADVISE models) relies on some properties that are embedded in the definition of each attack step and in each adversary's profile. Each attack step has the following properties:

- *Attack Cost*, which determines the cost of the attack (e.g., expressed in dollars);
- *Attack Execution Time*, which expresses the time needed to perform the attack;
- *Precondition Expression*, which determines the prerequisites needed by the adversary to perform the attack (a few examples are given in [subsection 3.4.2](#));
- *Success Probability* (and *Failure Probability*), which defines the probability that an attack step succeeds (or not);
- *Detection Probability*, which provides the probability that the adversary will be detected during the attack.
- *Effect*, which identifies the ADVISE elements (e.g., Access, Knowledge) that will be obtained by an adversary after attempting an attack step, both in case of success or failure.

When one defines the SID model, several attributes are associated with each Component, inherited from the ADVISE Meta ontology. Components of type "Device" have a total of sixteen attributes (e.g., *componentAnomalyDetectionStrength*, *resistanceToPhysicalDisable* and *userCyberSecurityAwareness*), while "Network" components have a total of thirteen attributes (e.g., *limitedIncomingProtocols*, *networkEncryptionStrength* and *networkWhiteList*). Some of these attributes are common to both types of components, e.g., *strengthOfUserAuthentication*.

The values assigned to the properties of each attack step, to the properties defined in the adversary's profile (i.e., *Planning Horizon*, *Payoff* and *Cost of Detection*, as explained at the end of [section 2.1](#)), and to the attributes associated to each Component can have an impact on the adversary's behavior and on the probability of successfully reaching a goal.

2.3 SAN

Möbius supports various formalisms, including Stochastic Activity Networks (SANs) [25, 26]. SANs are stochastic extensions of Petri nets [31], and make use of graphical primitives to offer a high-level modeling formalism. This allows for the specification of detailed models for performance, dependability, and performability analysis.

The SAN formalism consists of the following four elements [32]:

- **Place:** represents the state of the modeled system and is depicted graphically as a circle. Each place holds a certain number of tokens, which indicate its marking. The collection of all place markings constitutes the overall marking of the stochastic activity network. The tokens within a place are homogeneous, meaning that only the amount of tokens is known, without distinguishing between different tokens within the same place.
- **Activity:** represents actions within the modeled system and is categorized as either timed or instantaneous. Timed activities have a duration and are graphically depicted as thick vertical lines. Each timed activity is associated with a time distribution function. Instantaneous activities, on the other hand, represent actions that occur immediately when enabled and are shown as thin vertical lines. Case probabilities, illustrated as circles on the right side of an activity, are used to model uncertainty in the completion of an activity, with each case representing a possible outcome. These activities have an associated case probability distribution. An activity becomes enabled when the predicates of all connected input gates are true, and all places linked by incoming arcs contain tokens. Once enabled, the activity samples its delay distribution function to determine the time until it fires. When the activity fires, the system's state is updated by removing tokens from places connected by incoming arcs, adding tokens to places connected by outgoing arcs, and executing functions in both input and output gates. The order of state updates follows this sequence: input gates, input arcs, output gates, and finally output arcs.
- **Input gate:** regulates the enabling of activities and specifies the marking changes that apply upon an activity's completion. Input gates are depicted graphically as red triangles, with arcs extending from them to the places they depend on, known as input places. Each input gate is defined by an enabling predicate and a function. The enabling predicate is a boolean function that determines whether the connected activity is enabled, based on the markings of the input places. The input gate function outlines the marking changes that occur when the associated activity completes.

- **Output gate:** specifies the marking changes that occur when activities complete, similar to input gates. The key distinction is that an output gate is associated with a single case. Output gates are depicted graphically as black triangles, with the flat side connected to an activity or case, and arcs extending from the other side to the places affected by the marking changes. Each output gate is defined solely by a function, which determines the marking changes upon activity completion.

SANs can be solved through either analysis or simulation, depending on the characteristics of the system [25]. Analytic methods can be applied when all activity time distributions are exponential, and activities are reactivated frequently enough so that their rates depend only on the current state. Under these conditions, stochastic processes can be used to derive analytic solutions for a wide range of reward variables related to both activity and marking behavior. If these conditions are not met, system behavior can be evaluated through simulation instead.

SANs support several distributions that can be used for firing times of activities, such as deterministic, exponential, or normal. In performability analysis, some distributions of particular characteristics of a system are known. For example, the exponential distribution is frequently used to model the failure of electronic components that usually do not wear out until long after the expected life of the product in which they are installed. When dealing with cybersecurity it is extremely difficult to identify specific distributions for attacks as the time needed for executing an attack relies on many different factors like adversary capabilities and motivations, system characteristics, and attack complexity. For this reason, the model-based cybersecurity analysis should not focus on a strict quantitative evaluation. Instead, such kind of analyses should be used as a support in design phases for comparing different attack scenarios, considering different adversaries, architectural solutions, and countermeasures.

2.4 COMMON ATTACK PATTERN ENUMERATION AND CLASSIFICATION (CAPEC)

Throughout the history of IT security, it has become increasingly necessary to have reference lists of possible threats to IT systems, like the “OWASP Top 10” for web applications [33], or the MITRE Common Weakness Enumeration (CWE) [34]. MITRE ATT&CK [35] is a wide-adopted and modern knowledge base for attacks, but the detailed textual and descriptive content of its entries does not help the integration with structured frameworks like ADVISE Meta.

The MITRE Common Attack Pattern Enumeration and Classification (CAPEC) [23] is a large online catalog containing more than five hundred entries of common *attack patterns*. An attack pattern is a description of the common attributes and approaches used by adversaries to exploit known weaknesses in IT systems.

Each entry in the database describes a particular attack pattern and is organised in sections. This well-structured organization of the CAPEC entries facilitates the mapping with other taxonomies, like, in our case, the ADVISE modeling elements, as we show in [chapter 3](#). Each CAPEC entry contains, among others, the following sections:

- *Description:* represents a detailed description of an attack pattern. Content may include a summary and a list of steps taken by the attacker. This element

can be used to capture a range of descriptive information. Comprehensive descriptions might include attack trees, exploit graphs, etc., to more clearly elaborate this type of attack.

- *Prerequisites*: represents a container of one or more attack prerequisites. An attack prerequisite is a condition that must exist in order for an attack of this type to succeed.
- *Likelihood*: reflects the likelihood of attack success on a scale of Very Low, Low, Medium, High, Very High, in consideration of the attack prerequisites, targeted weakness, attack surface, skills and resources required, as well as the effectiveness of likely implemented blocking solutions.
- *Resources Required*: describes the resources (CPU cycles, IP addresses, tools, etc.) required by an attacker to effectively execute the specific type of attack.
- *Skills Required*: represents a container of one or more attacker skills required. Attacker skill required describes the level of skills or specific knowledge needed by an attacker to execute the particular type of attack.
- *Consequences*: represents the desired technical results that could be achieved/leveraged by this attack pattern, represented as an enumerated list of defined adversary motivations/consequences.

CAPEC offers three main ways of navigating the various attack patterns:

- By Mechanisms of Attack, which organizes attack patterns based on the mechanisms frequently employed by adversaries to exploit vulnerabilities.
- By Domains of Attack, which organizes attack patterns by the domain, i.e., the type of component attacked, such as software or hardware.
- By Abstractions, which organize the attack patterns according to the abstraction possessed by each attack pattern

The possible levels of abstraction are the following:

- *Meta*: these attack patterns are an abstract characterization of a methodology or technique used in an attack, often reported without a specific implementation. Most often they are a generalization of a related group of "Standard" abstraction attack patterns. They are also particularly useful for threat modeling applications at the architectural and design level.
- *Standard*: these attack patterns are focused on a specific methodology or technique used in an attack, often considered as a single step of complete attacks.
- *Detailed*: these attack patterns provide a low level of detail, usually targeting a specific technique or technology, and expressing a complete execution flow. They also often combine several "Standard" abstraction attack patterns chained together to achieve a certain goal.

2.5 THREAT AGENT LIBRARY (TAL)

When considering the possible threats that can threaten a system, it is also fundamental to identify which are the adversaries' profiles that can attempt the attack. Naming an adversary with generic terms like "hacker" or "spy" can be misleading. In order to have a detailed description of the adversaries that might be involved in attacks, Intel developed the Threat Agent Library (TAL) [24], a standardized reference that provides a description of the human agents that can threaten IT systems and other information assets.

TAL relies on a common set of characteristics, or attributes, to define each adversary, or "threat agent", in a unique way. To the best of our knowledge, this is the only work that classifies adversaries in such a structured way, therefore facilitating the integration with formalisms like ADVISE. In particular, the following eight attributes are defined [24]:

- *Access*: defines the extent of the agent's access to the company's assets;
- *Intent*: defines whether the agent intends to cause harm;
- *Limits*: the legal and ethical limits that may constrain the agent. This characteristic also defines the extent to which the agent may be prepared to break the law;
- *Objective*: the action that the agent intends to take in order to achieve a desired outcome;
- *Outcome*: usually defines the agent's primary goal and what the agent hopes to accomplish with a typical attack;
- *Resources*: defines the organizational level at which an agent typically works, which in turn determines the resources available to that agent for use in an attack;
- *Skill Level*: the special training or expertise an agent typically possesses;
- *Visibility*: the extent to which the agent intends to conceal or reveal his or her identity.

Each of these attributes can have different predefined values. For instance, the *Access* can be "internal" or "external", while the *Skill Level* can be "none", "minimal", "operational" or "adept". In TAL, the combination of these attributes' values results in a total of twenty-one different adversaries (e.g., Vandal, Employee Disgruntled, and Terrorist), as shown in [Figure 3](#).

Intent	NON-HOSTILE						HOSTILE															
	Employee Reckless	Employee Untrained	Info Partner	Anarchist	Civil Activist	Competitor	Corrupt Government Official	Data Miner	Employee Disgruntled	Government Cyberwarrior	Government Spy	Internal Spy	Irrational Individual	Legal Adversary	Mobster	Radical Activist	Sensationalist	Terrorist	Thief	Vandal	Vendor	
Internal Access (1)																						
External																						
Acquisition/Theft																						
Business Advantage																						
Damage																						
Embarrassment																						
Tech Advantage																						
Code of Conduct																						
Legal																						
Extra-legal, minor																						
Extra-legal, major																						
Individual																						
Club																						
Contest																						
Team																						
Organization																						
Government																						
None																						
Minimal																						
Operational																						
Adept																						
Copy																						
Deny																						
Destroy																						
Damage																						
Take																						
All of the Above/Don't Care																						
Overt																						
Covert																						
Candestine																						
Multiple/Don't Care																						

Figure 3: TAL adversaries profiles. Figure extracted from [24].

AN EXTENSION OF THE ADVISE META MODELING FRAMEWORK

In this chapter, we present a methodology that enables the extension of the ADVISE Meta ontology with CAPEC attack patterns and TAL adversaries' profiles. We apply the extended ontology to a case study in the public transportation domain. The content of the chapter is based on the works in [27, 28].

3.1 INTRODUCTION

Models for early-stage security analysis can be used for a preliminary assessment of the most critical architectural components of a system, allowing to identify those that should be more protected. Such analysis is performed having a very preliminary knowledge of the system, without knowing which are the vulnerabilities of the components, which are the possible involved attacks, which are the adversaries' profiles that could potentially perform such attacks, and the consequences of such attacks.

In the literature several formalisms have been proposed for helping with such challenging activity [36][19]. However, for complex systems, the application of such formalisms is often a time-consuming and error-prone activity, and information from experts in the application domain is required for properly capturing key elements (e.g., attack patterns or adversaries), which are rarely formalized. Model-Driven Engineering frameworks [37] can thus play a fundamental role and, starting from an high-level architectural description of the system, they can be used to derive complex low-level analyzable models.

In this chapter, we propose a methodology for the extension of the security-oriented modeling framework called ADVISE Meta [22]. The methodology aims at integrating into the ontology: i) attack patterns from the CAPEC (Common Attack Pattern Enumerations and Classifications) database [23], which is a publicly available catalog of common attack patterns; and ii) adversaries' profiles from the Threat Agent Library (TAL) [24] by Intel, that provides a description of the human agents that can threaten IT systems and other information assets.

In this chapter we also make use of the extended framework for an early-stage security analysis of a public transport supervision system called Smart Passenger Center (SPaCe) [38]. ADVISE Meta allows computing the probability that a given adversary can successfully reach a specific goal, and analysing the most probable attack path that the adversary will follow to reach the goal. With the proposed extension we enable additional analysis scenarios, ranging from broad security analysis at varying of TAL adversaries, where CAPEC attacks are involved, to the

identification of the components of the system that are most exposed to threats. Such kind of analysis can be used to guide the system development process.

The rest of this chapter is organized as follows. In [section 3.2](#) we discuss some related works. The methodology to map CAPEC and TAL elements into the ontology of ADVISE Meta is presented in [section 3.3](#), and it is applied in [section 3.4](#) for the extension of the ADVISE Meta ontology with TAL adversaries' profiles and CAPEC attack patterns. In [section 3.5](#) we introduce the SPaCe system and its high-level architecture, which is then modeled in [section 3.6](#), and analyzed in [section 3.7](#) considering different analysis scenarios.

3.2 RELATED WORK

Security analysis has widely relied on models, especially qualitative models, as a means to organize the information on the system under analysis [36].

The survey in [18] proposes an extensive overview on attack and defense modeling techniques based on Directed Acyclic Graphs (DAGs). The authors analyse more than thirty formalisms and group them according to two main dimensions, which are i) attack and/or defense modeling, where attack modeling focuses on attackers' actions while defense modeling focuses on defensive aspects, and ii) static or sequential approaches, where sequential formalisms can model temporal aspects, while static approaches cannot. Static modeling of attacks includes, among others, attack trees, while sequential attack modeling includes Bayesian networks. Among the static formalisms that include defense aspects are security-activities graphs, while for sequential approaches for defenses include, e.g., attack-response trees. The authors also briefly illustrate a few formalisms which are not based on DAGs, like Petri nets and attack graphs.

Attack trees [39, 40] originated from adapting the idea behind fault trees to security analysis: basic attacks are combined in a tree-like structure, until reaching a top event, which represents a system-level security violation. Attack graphs allow for a more detailed modeling of the possible paths an adversary can follow, as they are not restricted to a tree structure. The ADVISE formalism [19] is a quantitative extension of attack graphs, in which the time required to perform attacks and their outcomes are determined by probability distributions. Further, ADVISE introduces specialized features to describe different attackers.

The ADVISE Meta ontology framework [22] builds on the ADVISE formalism to increase its abstraction level. To the best of our knowledge, that is the only attempt in the literature to automatically generate detailed, stochastic security models from a collection of system meta-components and a concrete system configuration. For a more detailed discussion on the peculiarities of ADVISE Meta, and of its positioning with respect to other works sharing the same objective, we refer to [22].

In the following, we discuss the work in the literature from a perspective that is closer to ours. That is, we focus on work whose objective is to derive detailed security analysis models from higher-level representations of the system, and we discuss the extent to which they consider a variety of adversaries' profiles and attack patterns.

The authors of [41] propose a tool that generates and simulates attack scenarios based on CAPEC. The input of the tool includes a detailed configuration of the

network, information on the hosts, the profiles of the adversaries, and the CAPEC patterns. Their work shares some objectives with ours, most importantly the evaluation of possible attacks against with the inclusion of adversaries' profiles. However, the work in [41] is more tailored to advanced stages of system development, when details on the system are known; conversely, our approach can be applied since the early phases of the systems development life-cycle. Furthermore, we provide a more detailed characterization of adversaries, adopting the categorization defined in the TAL library.

The authors of [42] have analyzed and evaluated several existing conceptualizations on the topic of cyberthreat analysis, including different taxonomies, sharing standards, and ontologies. Like our work, they also covered both TAL and CAPEC. Their analysis concludes that no single taxonomy covers all the aspects and abstraction layers that are needed to perform an effective security analysis. We believe that such result confirms the need for cross-taxonomy mappings, like those we have presented in this paper. The work in [43] proposes a framework for assigning security scores to domain-specific Cyber-Physical Systems. The work also includes different attack types and adversaries' profiles. However, differently from our work, the attacks to the system are not simulated, but instead a scoring algorithm is used. Besides that, the approach adopts more general categories, both for attack types and for the capabilities of adversaries. In our work we specify more detailed adversaries' profiles (in terms of accesses, knowledge, and skills), as well as more detailed attack patterns.

The authors of [44] propose a meta-language for modeling threats and simulating attacks. The approach is based on a textual meta-language that is used to specify domain-specific models, from which Java code for simulating the system is automatically derived. However, common attack patterns and adversaries' profiles are not included in the proposed meta-language. The work in [45] proposes extensions to UML for the specification and modeling of security aspects of critical infrastructures. Based on such specification, models for security analysis can be automatically generated. While UML is relatively widespread, building detailed UML models using customized profiles requires advanced modeling skills; conversely, the ADVISE Meta approach and our methodology focus on even higher abstraction and ease of use. Besides that, the work in [45] does not consider different types of adversaries' profiles.

SecuriDN [46] is a graphical tool based on Draw-Net [47] designed to allow security analysts to create architectural models and to automatically derive attack paths and Dynamic Bayesian Networks (DBNs). By defining the network architecture of interest, users can combine predefined assets and specify their relationships based on the existing architecture. Each asset includes information about potential attacks targeting that resource, and the relationships between assets indicate how those attacks could spread to others. Once the architecture is completed, SecuriDN integrates the information from the various nodes into an attack graph, illustrating the potential attack processes available to the adversary. The generated DBN permits the quantification of the system security level through evidence-based and time-driven probabilistic analysis.

The General Agent Model for the Evaluation of Security (GAMES) [48] is a modeling formalism that generalizes the ADVISE formalism. Besides the adversary,

GAMES allows the representation of the defender and user counterparts, which can be composed together to build and analyze comprehensive security scenarios.

To summarize, we believe our work is one of the first exploring the connections between: i) quantitative security analysis formalisms at early-design stage, ii) established taxonomies of adversaries' profiles (like the one defined in TAL), iii) established taxonomies of attack patterns (like those defined in CAPEC).

In a very recent work [49] a computational environment based on the ADVISE formalism to model attack paths on CPSs has been developed, using a generalized stochastic optimization framework that allows to implement attacker agents based on different techniques, including approximate dynamic programming, reinforcement learning, or stochastic programming. While such work also builds on top of ADVISE, the focus is on its mathematical formulation on the evaluation algorithm. We focus, instead, on how to build complex models and how to map them to real security taxonomies.

3.3 METHODOLOGY: LINKING CAPEC, TAL AND ADVISE META ELEMENTS

In this thesis, we are interested in preliminary security evaluation conducted at an early stage of development. By definition, this type of analysis is subject to significant uncertainty about the effective attacks likely to be seen in operation. Thus, having a wide selection of attacks and adversaries would allow to conduct broad security analyses. After observing the ontology provided by the ADVISE Meta framework, two main aspects can be noted:

1. There are only a few attack steps in the ontology, categorized into just five attack types (i.e., damage and disable, malware, gain access, compromise data integrity, and compromise data confidentiality). The CAPEC database, on the other hand, provides more than five hundred attack patterns.
2. The adversaries' templates provided by the ontology are still too generic. For example, considering the Hacker Group, any individual with enough IT skills could be potentially classified as "hacker". Conversely, the adversaries' profiles proposed by TAL are more specialized (e.g., Government Cyberwarrior, Thief, Civil Activist).

In this section, we present a methodology that is based on the definition of relationships between the properties identified in the CAPEC, TAL, and ADVISE Meta domains. The identified relationships can be used to extend the ADVISE Meta framework with CAPEC attacks and TAL adversaries (see [section 3.4](#)), and thus use them to build ADVISE models for security analysis.

3.3.1 *From CAPEC Sections to TAL Attributes*

We have first identified the relationships between CAPEC and TAL, to understand how the information found in the CAPEC sections are linked to TAL attributes. This mapping allows CAPEC attack patterns to be described in terms of attributes of TAL adversaries. We have identified the following relationships between CAPEC and TAL, which are also summarized in the left part of [Figure 4](#):

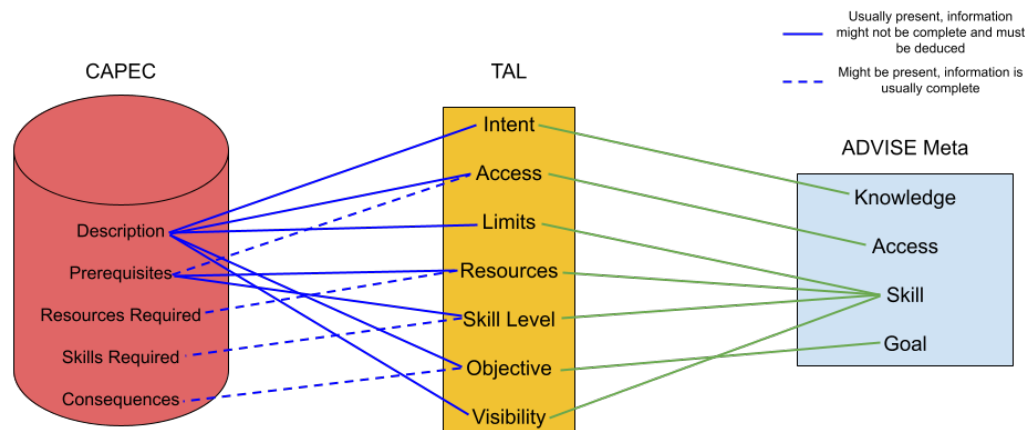


Figure 4: Relationships between CAPEC sections, TAL attributes, and ADVISE Meta elements. Figure extracted from [28].

- According to the TAL's *Intent*, which defines the adversary's intention to cause harm, an adversary can be "Hostile" or "Non-Hostile". The "Description" section of a CAPEC entry usually provides some inferable information related to this TAL attribute.
- In TAL, the *Access* attribute can have two values, "Internal" or "External", denoting the extent of the adversary's access to the system's assets. A list of prerequisites that an adversary must satisfy in order to execute the attack (including the access to assets) is usually given in the "Prerequisites" section of CAPEC. If this section is not detailed enough, additional information can be derived from the "Description" section.
- The TAL's *Limits* attribute defines the ethical and legal limits of an adversary, and how much the adversary is prepared to break the law. Four different values are allowed ("Code of Conduct", "Legal", "Extra-Legal Minor" and "Extra-Legal Major"). No specific section containing information about ethical and legal limits exists in CAPEC, so one must infer such information from the "Description" section.
- The type of organization and the amount of resources owned by the adversary to run attacks are defined by the TAL's *Resource* attribute, which can have six different values ("Individual", "Club", "Contest", "Team", "Organization", and "Government"). The "Resources Required" CAPEC section provides a little information about specific equipment, software, and other kinds of resources needed in order to perform the attack. If this section is not present, this information must be inferred from the "Prerequisites" section.
- The expertise of an adversary is determined by the TAL's *Skill Level*. Four different values are available ("None", "Minimal", "Operational" and "Adept"). The "Skills Required" CAPEC section describes the the skill level needed to execute the attack. Additional information can be found in the "Prerequisites" section.

- In TAL, the *Objective* attribute defines the goal that an adversary wants to achieve. Five different values are allowed (“Copy”, “Destroy”, “Injure”, “Take” and “Don’t Care”). This attribute can be associated to the “Consequences” CAPEC section. Here the scope (e.g., confidentiality, integrity, or availability), and the impact of the attack are described. If this section is not present, the information must be inferred from the “Description” section.
- The extent to which the adversary intends to hide/reveal her or his identity is described by the *Visibility* attribute, which can have one of the following values: “Overt”, “Covert”, “Clandestine” and “Don’t Care”. No dedicated section is available in CAPEC, so the “Description” section should be checked.

Note that these relationships do not depend on the underlying modeling framework. On the other hand, the relationships are not always easy to identify, as the CAPEC database sometimes does not provide all the necessary information. For example, in CAPEC entries, while the “Description” and “Prerequisites” sections can always be found, other useful sections like “Resources Required” and “Skills Required” might be missing. In such cases, the missing information must be inferred and interpreted from the available sections, when possible.

3.3.2 From TAL Attributes to ADVISE Meta Elements

As a second step, we have identified the relationships between TAL and ADVISE Meta elements, to represent TAL adversaries’ profiles in the ADVISE Meta framework. In the right part of [Figure 4](#) we can see how the TAL attributes are related to the ADVISE Meta elements:

- The TAL’s *Intent* attribute can be associated to the *Knowledge* concept of ADVISE Meta, as the adversary knows if she/he has an hostile intent or not. To execute a malicious attack, a malicious intent is usually necessary, so we have added a new Knowledge element named *Intent* to the ontology, to be associated to malicious adversaries.
- The *Access* attribute from TAL can be intuitively related to the *Access* element (ADVISE concept) already present in the ontology, called *InsiderAccess*.
- The TAL’s *Limits* attribute represents the ethical and legal limits of the adversary. This concept has been associated to a Skill in ADVISE Meta, which can be interpreted as the adversary’s ability to act at different levels of legality. We have therefore added a new Skill element called *Limits* to the ontology, to model such attribute in ADVISE Meta.
- Also the *Resources* TAL’s attribute can be associated to a Skill in ADVISE Meta, because it can be seen as the adversary’s ability to gain the appropriate resources required for attacks. A new Skill element named *Resources* has been added to the ontology.
- The TAL’s *Skill Level* represents the expertise level of an adversary. We have related this attribute to its counterpart in ADVISE Meta, i.e., *Skill*. A new Skill element called *SkillLevel* has been added to the ontology of the framework.

- No ADVISE Meta element can be easily associated to TAL's *Objective* attribute. *Goal* is the closest concept in the ADVISE formalism, which is however associated to attack steps. Therefore, one should not add this element to the ontology when modeling adversaries' profiles, but when modeling attack steps.
- The degree of importance for the adversary to remain hidden is represented by TAL's *Visibility* attribute. Conceptually it is something that the adversary knows, but because it has more than two possible values, it cannot be associated to a Knowledge element, which is instead a Boolean property. Therefore, a new Skill element named *Visibility* has been added to the ontology, which can be interpreted as the extent to which the adversary intends to hide/reveal her or his identity.

In ADVISE, a Skill element is defined as an integer which can have integer values between 0 and 1000. The numerical thresholds used to represent the different TAL attributes' values are shown in [Table 2](#). This should only be interpreted as a translation of the TAL attributes' values from a qualitative to a quantitative point of view, needed to practically represent these attributes in the framework. As a default setting we have equally distributed the qualitative values of each TAL attribute in a quantitative range between 0 and 1000. This setting can be anyway modified according to the modeler's need and could be made parametric using global variables.

3.3.3 From CAPEC Sections to ADVISE Meta Elements

The extension of the ADVISE Meta ontology with CAPEC attacks may require further information that can be found in CAPEC sections. In particular, when a new attack step is added to the ontology, the target of the attack (i.e., the involved system's component) should be specified. In ADVISE Meta each attack step is associated with a single component, but the outcome of the execution of an attack step can have an impact on other components according to the relationships involved (the "Relationship" element is defined in the ontology, see [section 2.2](#)).

Such kind of information can be derived from the CAPEC "Description" section. Moreover, the "Description" and "Precondition" sections can be used to derive additional information on attacks, like preconditions related to the existence of particular architectural components.

3.4 EXTENSION OF ADVISE META ONTOLOGY

The methodology presented in [section 3.3](#) has been applied to extend the ontology of the ADVISE Meta framework, to include some representative CAPEC attack patterns and all the adversaries' profiles provided by TAL.

Table 2: Assignment of numerical values to TAL attributes. Table extracted from [28].

TAL attribute	TAL attribute value	Numerical value
Intent	Not Hostile	0
	Hostile	1
InsiderAccess	Outsider	0
	Insider	1
Limits	Code of Conduct	250
	Legal	500
	Extra-legal minor	750
	Extra-legal major	1000
Resources	Individual	0
	Club	200
	Contest	400
	Team	600
	Organization	800
	Government	1000
SkillLevel	None	0
	Minimal	250
	Operative	750
	Adept	1000
Visibility	Overt	1000
	Covert	500
	Clandestine	250
	Don't Care	0

3.4.1 Extension of ADVISE Meta with TAL Adversaries' Profiles

Following the TAL/ADVISE Meta relationships and the attributes' values specified in the TAL library, we translated the TAL adversaries' profiles to ADVISE Meta adversaries.

To exemplify the application of the methodology, we discuss how to extend the ontology with the *Terrorist* profile. In TAL this adversary has "External" Access, "Hostile" Intent, "Adept" Skill Level, "Extra-Legal Major" Limits, "Organization" Resources and "Covert" Visibility. To add this profile to the ontology we have created a new ADVISE Meta adversary called "Terrorist".

The screenshot shows a user interface for configuring an adversary profile. The 'Name' field is set to 'Terrorist'. A checkbox 'Use default code name:' is checked, and the 'Code Name' field also contains 'Terrorist'. Under 'Decision Parameters', 'Planning Horizon' is 5 and 'Cost of Detection' is 1000. The 'Access' section is currently empty. The 'Knowledge' section contains one entry: 'Intent' with an 'Init Value' of 1. The 'Skills' section contains a table with the following data:

Name	Proficiency
Limits	1000
Visibility	500
Resources	800
SkillLevel	1000

Figure 5: TAL “Terrorist” adversary’s profile visualized in the ADVISE Meta framework. Figure extracted from [28].

According to Table 2, the Terrorist profile has the following attributes values (Figure 5):

- *InsiderAccess* with value 0 (“External”), i.e., the Access is not added to the profile;
- *Intent* with value 1 (“Hostile”);
- *Limits* with value 1000 (“Extra-Legal Major”);
- *Visibility* with value 500 (“Covert”);
- *Resources* with value 800 (“Organization”);
- *SkillLevel* with value 1000 (“Adept”).

3.4.2 Extension of ADVISE Meta with CAPEC Attacks

To show how it is possible to extend the ontology of the framework with new attacks by using the proposed methodology, we have chosen some representative attack patterns involving the core components of the CIA triad, i.e., Confidentiality, Integrity and Availability [16]. We have focused on the following CAPEC attacks (the affected security properties are specified in brackets):

- CAPEC-94: Adversary in the Middle (Confidentiality and Integrity);

- CAPEC-125: Flooding (Availability);
- CAPEC-153: Input Data Manipulation (Integrity);
- CAPEC-248: Command Injection (Confidentiality, Integrity, and Availability);
- CAPEC-549: Local Execution Of Code (Confidentiality, Integrity and Availability).

Although we have considered only a few representative CAPEC attacks, the methodology illustrated in [section 3.3](#) is applicable to all the attack patterns defined in the CAPEC database. However, in handling the largeness of the CAPEC database (more than five hundred attacks), two main problems arise: i) adding manually all the CAPEC attacks to the ontology would be an error-prone and very time-consuming activity, and ii) the complexity of the derived ADVISE models will become unmanageable due to an exponential growth in the number of states to explore.

The identification of the relationships between CAPEC, TAL, and ADVISE Meta (described in [section 3.3](#)) and the subsequent extension of the framework ontology have been done entirely manually. In [chapter 4](#) we present an approach for the semi-automated integration of CAPEC attack into the ADVISE Meta ontology.

By adding more attack patterns to the ontology, the generated ADVISE models will become more complex and the number of states to explore will grow exponentially. However, as already explained in [section 2.2](#), the automatic generation of attacks depends on some modeling constraints that can (at least partially) mitigate the problem. In particular, an attack will not be generated if the related component is not present in the SID model. Moreover, to limit the number of generated attack steps, it could be helpful to disable one or more attacks on one or more specific components' instances in the SID model. For example, the modeler may believe that a specific workstation cannot be the target of flooding attacks, because of how the system is deployed. In this case, one could add some "flag" attributes to specific component elements in the ontology, and then use them as prerequisites of the attacks, so that some specific attacks can be disabled.

Flooding

Here we describe the application of the methodology to the Flooding attack. Flooding is a Denial of Service attack, in which the adversary wants to deplete the resources of a target system to deny the access to users. In CAPEC this attack is classified under the "Software" and "Communication" domains, and under the "Abuse Existing Functionality" attack mechanism.

This attack has been modeled by adding a new attack step to the ontology ([Figure 6](#)). The targets of this attack are those system components that are classified as Device (according to the "Description" section of the CAPEC entry). Preconditions for this attacks are at least "Extra-Legal Minor" Limits, and the adversary's intent to cause damage to the system (from the "Description" section of the CAPEC entry). Furthermore, at least "Club" Resources are required, because a script or a network able to generate an high number of requests is needed (from "Resources Required" section of the CAPEC entry).



Figure 6: “Flooding” attack step added to the ontology of ADVISE Meta. Figure extracted from [28].

Considering the default setting for the values of the TAL attributes presented in Table 2, the above preconditions are met if the value of the Limits TAL attribute is greater or equal than 750 (value of the “Extra-Legal Minor” Limits), and the value of the Resources TAL attribute is greater or equal than 200 (value of the “Club” Resources). Another precondition is the access to a network (still from the “Description” CAPEC section). Thus, the *Precondition Expression* of the Flooding attack step, i.e. the Boolean expression that has to be satisfied to perform the attack, is the following:

```
return((${Limits}>=750) //at least 'Extra-legal minor' Limits
&& (${Intent}) //has 'Hostile' Intent
&& (${Resources}>=200) //at least 'Club' Resources
&& (${AccessToNetwork})); //has access to the targeted network
```

To better model this attack, the *handlingRate* attribute has been added to the ontology for the Device element, denoting the rate at which the device is able to handle requests. This attribute can have values in the range between 0 and 10: low (0–3), medium (4–6), and high (7–10). In the following we show the expression of the *Success Probability* for the Flooding attack:

```
if (${handlingRate}>=7) return 0.2;
else if (${handlingRate}>=4) return 0.5;
else return 0.9;
```

The *handlingRate* attribute has an impact on the probability that an adversary can successfully perform an attack: the lower its value, the higher the probability to

perform a successful attack. When creating the SID, the modeler can adjust the value of the attribute, according to the characteristics of the device to be represented in the model.

Adversary in the Middle

In the CAPEC-94 Adversary in the Middle, also known as Man in the Middle, the adversary takes position between two network nodes, to retrieve or modify the messages exchanged between the two victims before forwarding them to the other node. In the CAPEC entry, the “Description” section reports that the possible targets of this kind of attack are components classified as Device.

To model this attack in ADVISE Meta framework, we have added three different attack steps to the ontology, following the “Execution Flow” section of the CAPEC entry. In the first step (*MITMDetermineCommunicationMechanism*) the adversary identifies the mechanism used for the communication between the two nodes. Preconditions for this step are “Extra-Legal Minor” Limits (i.e., with value of at least 750), “Hostile” Intent (i.e., the Intent element is present), and “Team” Resources (i.e., with value of at least 600). We retrieved this information from the “Description” section of the CAPEC entry. Moreover, according to the “Prerequisites” section of CAPEC, two components communicating through a network must be present in the system model (in this case a client and a server). The *Preconditions Expression* of this attack step is the following:

```
return((${Limits}>=750)           //at least 'Extra-legal minor' Limits
&&(${Intent})                    //has 'Hostile' Intent
&&(${commAccessAlready}==0)     //adversary doesn't have access to the communication
&&(${Resources}>=600)           //at least 'Team' Resources
&&(${ClientExists})             //a client device is connected to the network
&&(${ServerExists}));          //a server device is connected to the network
```

After successfully performing the attack, the *CommunicationAccess* Access is gained by the adversary. In the subsequent attack step (*MITMPositionBetweenTargets*) the adversary takes position inside the network to intercept the messages exchanged between the two victims. The preconditions of this attack are the same as the previous ones, with the addition of the *CommunicationAccess* Access. If this attack is successfully completed, the adversary gains the *MonitoredNetworkAccess* Access.

In the last attack step (*MITMMonitoringNetworkAccess*) the adversary attempts to read or modify the intercepted data. For the previous attack steps no major law breaches were required, while a major illegal action is required to carry out this final step. Thus, additional preconditions for this attack step are “Extra-Legal Major” Limits, along with the *MonitoredNetworkAccess* Access obtained in the previous step. If the attack step is successful, the attacker finally obtains the *UseInterceptedData* Access.

3.5 CASE STUDY: SPACE SYSTEM

The Smart Passenger Center (SPaCe) system [38] is designed for the orchestration and supervision of the mobility of public transport, improving the passenger

experience, trying to prevent security breaches and facilitating investigations after a system violation.

The SPaCe system receives the data collected by sensors (mainly video cameras) installed on board of vehicles (trains and buses), and processes them to infer the status of vehicles in real time. Through the analysis of the collected data, the system is able to obtain information on the occupancy level of vehicles, informing users and administrators and thus enabling the optimization of passengers flow. The system can also identify possible dangerous situations on board, such as the presence of suspicious objects or damaged equipment.

The network architecture adopted for the SPaCe system is based on the DMZ (DeMilitarized Zone) model, which generally aims to protect a private network from external threats. A DMZ is a subnetwork that exposes services to external networks that are not considered secure (e.g., the Internet), and it is located between the internal network and the external one, ensuring that if a machine inside the DMZ is attacked, it does not directly affect the private network. A typical DMZ architecture consists of a private LAN network, a first firewall that acts as a filter between it and the DMZ network, and a second firewall with the same purpose between the DMZ network and the external network. DMZ hosts have the dual task of i) receiving information and requests from the external network and forwarding them to the private network hosts, and ii) receiving messages from the private network hosts and forwarding them to the external network.

[Figure 7](#) shows the system-level architecture of the SPaCe system and its components. The main components are the following:

- User Configuration Item (CI), located on the external network, as it is installed on the devices (e.g., PCs or smartphones) owned by users of the system, who can use them to retrieve information about on-board conditions (e.g., the occupancy level of the vehicles).
- Vehicle Configuration Item (CI), positioned on the external network, as it is installed on board of the vehicles. It supplies the data coming from vehicle sensors.
- Server Configuration Item (CI), located within the private network. Its objective is to provide the main functionalities of the system and to protect the stored data. A sub-component, called View Interface is located in the DMZ and it is responsible for retrieving, organizing and delivering information to be displayed to final users.

3.6 SECURITY ANALYSIS OF THE SPACE SYSTEM

In this section we describe the security analysis performed on the SPaCe system, using the methodology and the framework extension described in [section 3.3](#) and [section 3.4](#), respectively.

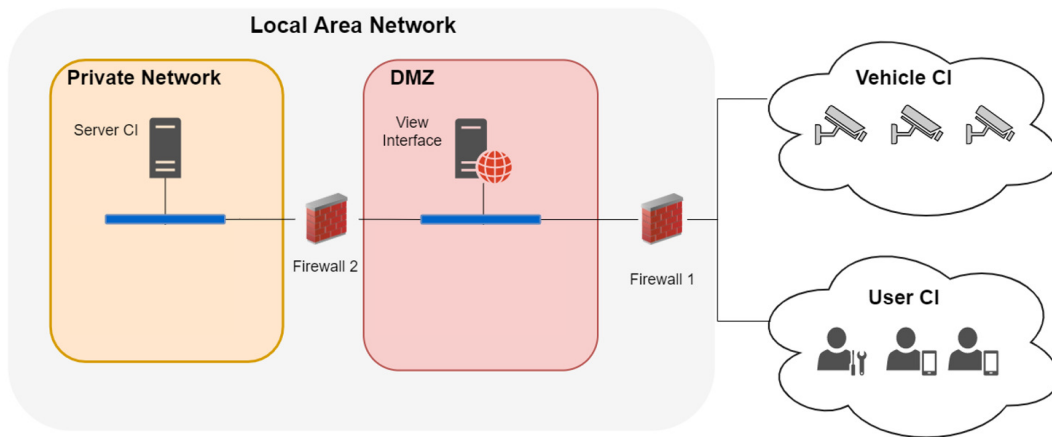


Figure 7: DMZ architecture for the SPaCe system. Figure extracted from [28].

3.6.1 Objectives of the Analysis

Thanks to the extensions to the ADVISE Meta modeling framework we are able to perform broader security analyses with respect to the base ontology of the framework. In particular, the extended framework allows us to: i) compute the probability that an adversary can successfully reach a goal in a given time window; ii) derive the attack path (i.e., the sequence of attack steps) that allows the adversary to reach a goal; iii) perform sensitivity analysis at varying of attack patterns and adversaries' profiles; iv) compare different implementation solutions; and v) identify the system's components that can be more probably attacked by the adversaries.

These analyses are carried out during an initial phase of system development, in which only the main architectural aspects of the system are known and the exact security mechanisms that will be adopted are not yet defined.

Given the early-stage nature of the analyses, besides offering preliminary quantitative indications on the targeted security aspects, the derived results can provide qualitative indications to guide the design process of the system. For example, analysis results can be used to support architectural decisions, to identify the most vulnerable components with respect to different CAPEC attacks and TAL adversaries, and to compare the impact of adopting different implementations and security mechanisms.

3.6.2 Modeling Process

In this section we provide the steps required for the security analysis of the SPaCe system using the extended ADVISE Meta framework. We first show the architectural model of SPaCe, then we provide instruction on how to define goals, adversaries and metrics in ADVISE Meta framework. Finally, we give some details about the security models generation and simulation.

Definition of the Architectural Model

The first step is the creation of the architectural model, also called System Instance Diagram (SID), of the SPaCe system by using ADVISE Meta framework. This model, shown in [Figure 8](#), has been created following the architecture described in [section 3.5](#). The architectural components have been added to the model using the elements from the ADVISE Meta ontology, and the components have been connected to each other through relationships. The main relationship involved in the model is the *onNetwork* relationship, which connects components of type Device (e.g., the ServerCI) to components of type Network (e.g., the LAN), to indicate that the device is on the network. Another relationship that appears in the model is the *readData* relationship, which connects a Data element to a Sensor component, meaning that the sensor collects those data from the environment.

The main architectural components introduced in [section 3.5](#) are represented in the model as it follows:

- User CI is represented in the model by a Device of type Workstation. In fact, the Workstation type defines a host supporting the interaction of a human user with application functions, thus including PCs, tablets and smartphones.
- Vehicle CI is represented by a Device of type Sensor.
- Server CI is represented by two different Devices of type Server, called ViewInterface and ServerCI.

Several attributes, inherited from the ontology of the framework, are associated to each component of the SID.

Referring to the analyses shown in the following sections, we focus on the *strengthOfUserAuthentication* attribute: it determines the level of security measures used to authenticate users for that particular component. The attribute can have the following values: 0 (no authentication or authentication with short or poorly validated passwords); 4 (long and strongly validated passwords, e.g., using Bloom Filters); 6 (use of a Primary Key Infrastructure, PKI); 7 (two-factor authentication); 9 (biometrics). This attribute is associated to each component of type Device and Network (i.e., every component of the SID model in [Figure 8](#), with the exception of InputData).

Definition of the Goals

Once the architectural model of the system is defined, the objectives (Goals) of the adversaries must be specified. The modeler must associate each goal with an element related to a specific component in the SID (e.g., the components' accesses, like Logical Access). For each component of the system, one or more representative goals involving that component have been defined. The added goals have been chosen among the CAPEC attacks added to the ontology. In particular:

- For each component of Network type, two objectives have been defined: one regarding access to the network (*NetworkAccess*), and another regarding the use of intercepted data in a potential Man-In-The-Middle attack (*UseInterceptedData*).

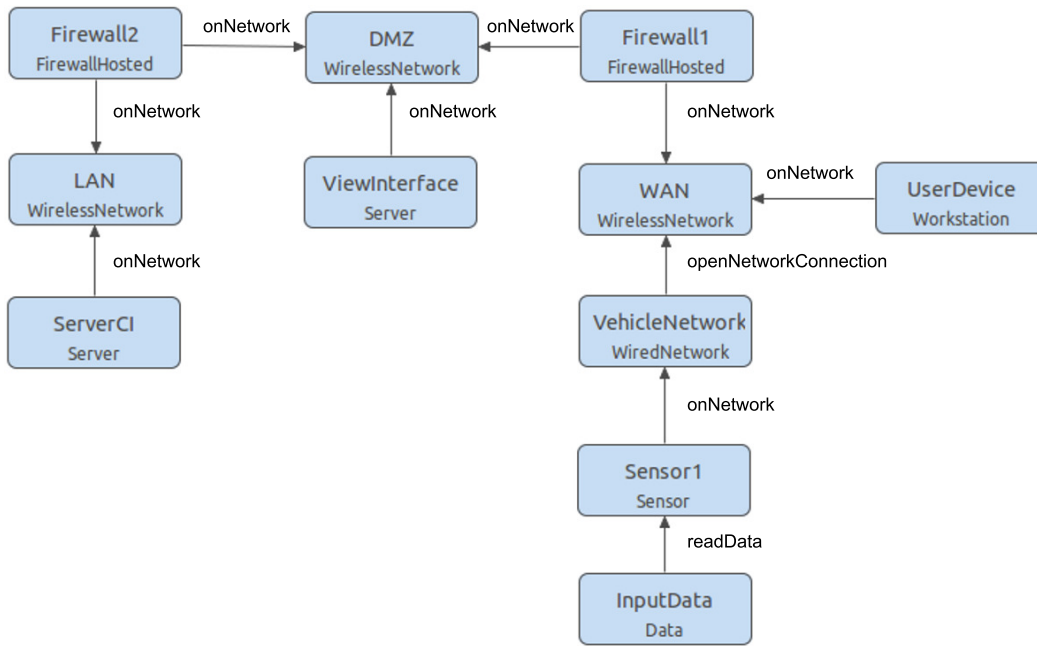


Figure 8: Architectural model (SID) of SPaCe system created with ADVISE Meta framework. Figure extracted from [28].

- For each component of Device type (in this case the Firewall and Server type components), three objectives have been defined: logical access to the device (*LogicalAccess*), installation of a ransomware software (*RansomwareInstalled*) and unavailability of the device (*UnableToService*).
- For components of Sensor type, the goal *ManipulatedInputData* has been defined. Achievement of this goal can be obtained, for example, with attacks that attempt to deceive the images classification algorithms by exposing an *ad hoc* crafted sign (Adversarial Patch) to a video camera. In the analyzed scenario we have only one sensor (Sensor1).

In summary, considering all the components of the architectural model (Figure 8), the following twenty-one different goals have been defined (the names are derived from the goal names and the names of the involved element in the SID):

- DMZNetworkAccess
- DMZUseInterceptedData
- Firewall1LogicalAccess
- Firewall1RansomwareInstalled
- Firewall1UnableToService
- Firewall2LogicalAccess
- Firewall2RansomwareInstalled

- Firewall2UnableToService
- LANNetworkAccess
- LANUseInterceptedData
- Sensor1ManipulatedInputData
- ServerCILogicalAccess
- ServerCIRansomwareInstalled
- ServerCIUnableToService
- VehicleNetworkNetworkAccess
- VehicleNetworkUseInterceptedData
- ViewInterfaceLogicalAccess
- ViewInterfaceRansomwareInstalled
- ViewInterfaceUnableToService
- WANNetworkAccess
- WANUseInterceptedData

Definition of the Adversaries

The next step is to define the adversaries' profiles to be used in the analysis. To have the broadest possible view on the threats represented by the adversaries, all twenty-one adversaries' profiles of the TAL library added to the ontology during the extension of the framework have been used. All twenty-one previously defined goals have been associated with each adversary. This implies that each adversary can potentially aim to achieve all twenty-one defined goals, allowing for a broad comparison against various attacks and adversaries' profiles. Concerning the initial access to the system, it has been assumed that the adversary: owns a device (UserDevice) connected to the WAN, and has physical access to the sensor (Sensor1) located on the vehicle.

Definition of the Metrics

In order to evaluate the probability of achieving goals by the adversaries, a *goalAchieved* metric has been associated with each goal. By running the simulation, this set of metrics provides the probability that, as time varies, the adversary reaches each of the goals.

To define such metrics we must select the goal to observe, the first and the last observation time-point and the length of time between the observations. One or more metrics can be assigned to each adversary. Through the model generator, ADVISE Meta will automatically create a reward model associated to each metric. In particular, the metrics are computed by an instant of time rate reward variable

that returns the number of tokens in the goal element at the different observation times.

Generation and Simulation of the Models

The ADVISE models associated with each adversary's profile have been derived using the model generator (the framework generates a different ADVISE model for each adversary). Creating such models manually would have been a time-consuming and error-prone task. This complexity is clearly visible in [Figure 9](#) (left part), which shows the generated ADVISE model for the Terrorist adversary. A detail of the model is highlighted in the red box of [Figure 9](#) (right part), showing the attack steps related to Man In the Middle attack on the LAN component. This is one of the CAPEC attack patterns that we added to the ontology.

Each attack step has some prerequisites that the adversary must fulfill in order to successfully complete it, i.e., elements like Skills, Accesses, Knowledge and State variables. For example, the *LAN_MITMPositionBetweenTargets* attack step requires the *LAN_CommunicationAccess*, which is obtained after successfully completing the *LAN_MITMDetermineCommunicationMechanism* attack step. Indeed, the completion of each attack step can lead to the adversary gaining one or more elements (e.g., Accesses). Then, the attack path followed by the adversary can lead to one goal. As an example, a goal can be associated with the *UseInterceptedData* Access obtained after completing the *LAN_MITMMonitoringNetworkAccess* attack step. The names of these attack steps have been derived from the "Execution Flow" section of the CAPEC-94 entry.

Once the ADVISE models are derived, it is possible to simulate them using the simulator integrated in the Möbius tool, and to observe the values associated with the previously defined metrics. The tool automatically generates one simulator for each ADVISE model.

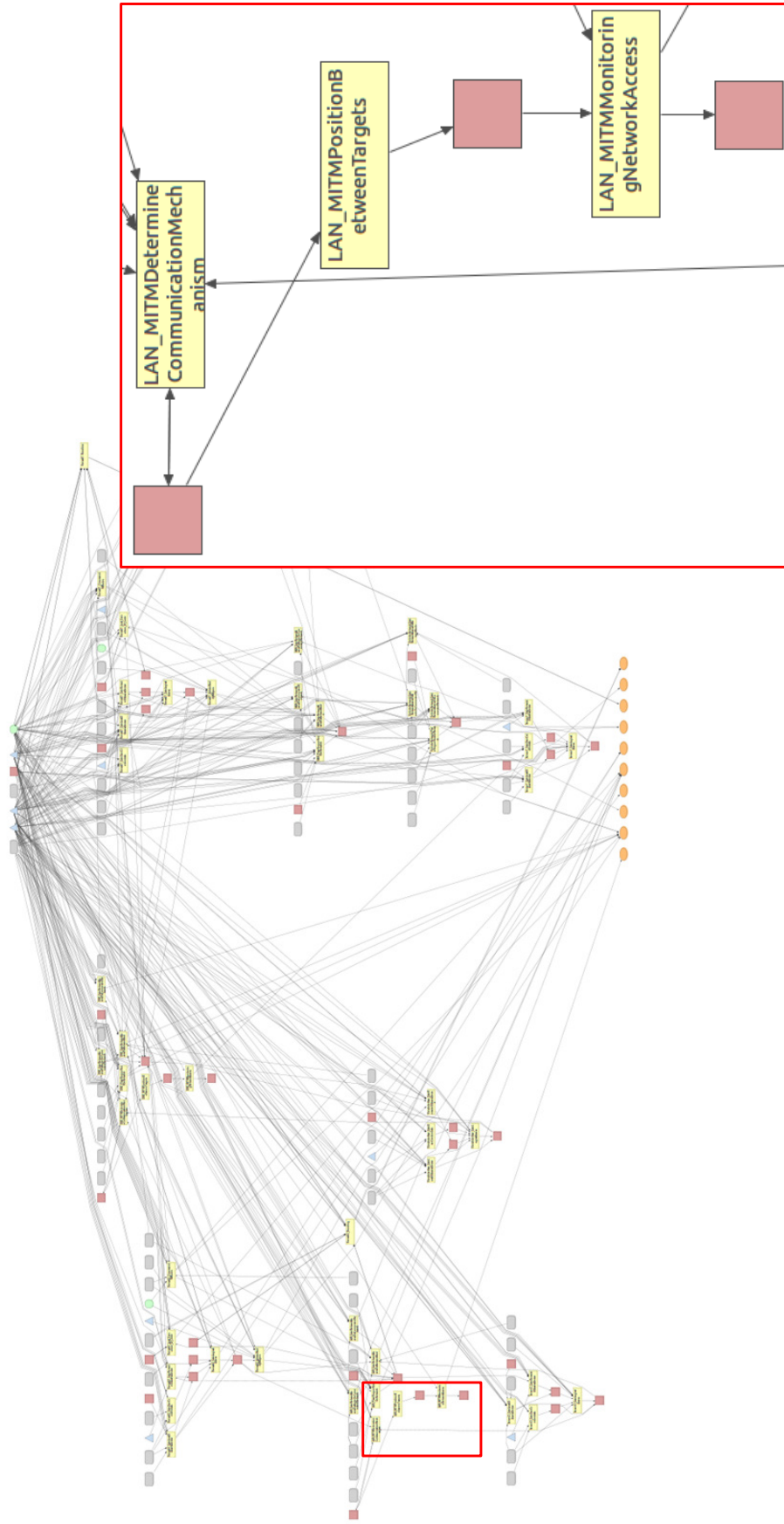


Figure 9: ADVISE model generated by the tool for the Terrorist adversary. In the red box, details of the Man in the Middle attack steps on the LAN are highlighted. Figure extracted from [28].

3.7 ANALYSIS SCENARIOS AND RESULTS

The analyses we carried out are described in this section. Unless differently specified, we computed the probability of achieving the different goals within 96 time units (hours). The adopted interval of time allow us to observe significant variations in the probability of completing the attacks and successfully reaching the goal.

We ran each simulation with, respectively, 1000 and 10000 minimum and maximum number of batches (simulation runs), converging within 95% probability in a 0.1 relative interval.

As part of the the extension of the framework, we assigned numerical values to the properties of attack steps and adversaries' profiles. For the attack steps, the values have been derived from CAPEC, inferring them from the entries of the database (as an example, we have already discussed the Precondition Expression for the Flooding attack in [subsubsection 3.4.2](#)). For the attacks that were already present in the ontology, time distributions and success/failure probabilities of the attack steps have been left as they were defined. For the CAPEC attacks that we have added to the ontology, instead, due to the lack of information, we used deterministic time distributions, and the success/failure probability was set according to the indications provided by the "Attack Likelihood" CAPEC section. To have long-sighted and motivated adversaries, we have used the following setting: *Planning Horizon*=5, *Payoff*=500000, and *Cost of Detection*=1000. The remaining setting of the adversaries used in the experiments is reported in [Table 3](#).

Table 3: Setting of the adversaries used in the experiments.

Attribute	Vandal	Employee Untrained	Employee Disgruntled	Terrorist
Insider Access	0	1	1	0
Intent	1	0	1	1
Skill Level	750	250	750	1000
Limits	750	250	1000	1000
Resources	400	0	0	800
Visibility	500	1000	1000	500

If not differently specified, the properties of the model (e.g., the components' attributes in the SID model) have been set to the default values as they are defined in the original ADVISE Meta framework. Specific sensitivity analyses should be performed at varying of the values of these parameters to evaluate their impact on the targeted metrics (see [subsection 3.7.2](#) for a concrete example).

3.7.1 Analysis at Varying of Adversaries' Profiles

[Table 4](#) shows the results related to the probability of achieving the goals defined in [subsubsection 3.6.2](#), at varying of some adversaries' profiles. The adversaries

Table 4: Probability of successfully achieving goals as the adversaries' profiles vary.
Table extracted from [28].

Goal	Vandal	Employee Untrained	Employee Disgruntled	Terrorist
DMZNetworkAccess	1	0	0	1
DMZNetworkAccess	1	0	0	1
DMZUseInterceptedData	0	0	0	0.8191
Firewall1LogicalAccess	0.9667	0	0	0.9999
Firewall1RansomwareInstalled	0	0	0	1
Firewall1UnableToService	0.9834	0	0	0.9548
Firewall2LogicalAccess	1	0	0	1
Firewall2RansomwareInstalled	0	0	0	1
Firewall2UnableToService	0.8264	0	0	0.0551
LANNetworkAccess	0.9647	0	0	0.8058
LANUseInterceptedData	0	0	0	0.3617
Sensor1ManipulatedInputData	1	0	1	1
ServerCILogicalAccess	0.8279	0	0	0.7756
ServerCIRansomwareInstalled	0	0	0	0.8978
ServerCIUnableToService	0.522	0	0	0.0615
VehicleNetworkNetworkAccess	1	0	0	1
VehicleNetworkUseInterceptedData	0	0	0	0
ViewInterfaceLogicalAccess	0.9995	0	0	0.3462
ViewInterfaceRansomwareInstalled	0	0	0	1
ViewInterfaceUnableToService	0.9698	0	0	0.9788
WANNetworkAccess	1	0	0	1
WANUseInterceptedData	0	0	0	0.3657

have been chosen among some of the most representative for the SPaCe system: Vandal, Employee Untrained, Employee Disgruntled and Terrorist. For this analysis, the system authentication level has been configured as low. In particular, the *strengthOfUserAuthentication* parameter defined in each element of the SID model has been set to 0. This means that authentication is either absent or of an extremely low level (e.g., short passwords are used, with no constraints on characters).

The first observation that can be made is that, for the Employee Untrained adversary, the probability of achieving each objective is always 0. This happens because all the involved attacks require the adversary's hostility, i.e. the adversary's profile must have the Intent attribute of type Knowledge with value equal to 1. The Intent attribute of the Employee Untrained adversary has value 0, therefore he can never reach any goal.

The Employee Disgruntled adversary is instead able to achieve the sensor data manipulation goal (*Sensor1ManipulatedInputData*), but not any other goal. The values of the attributes associated with this adversary are in fact inadequate

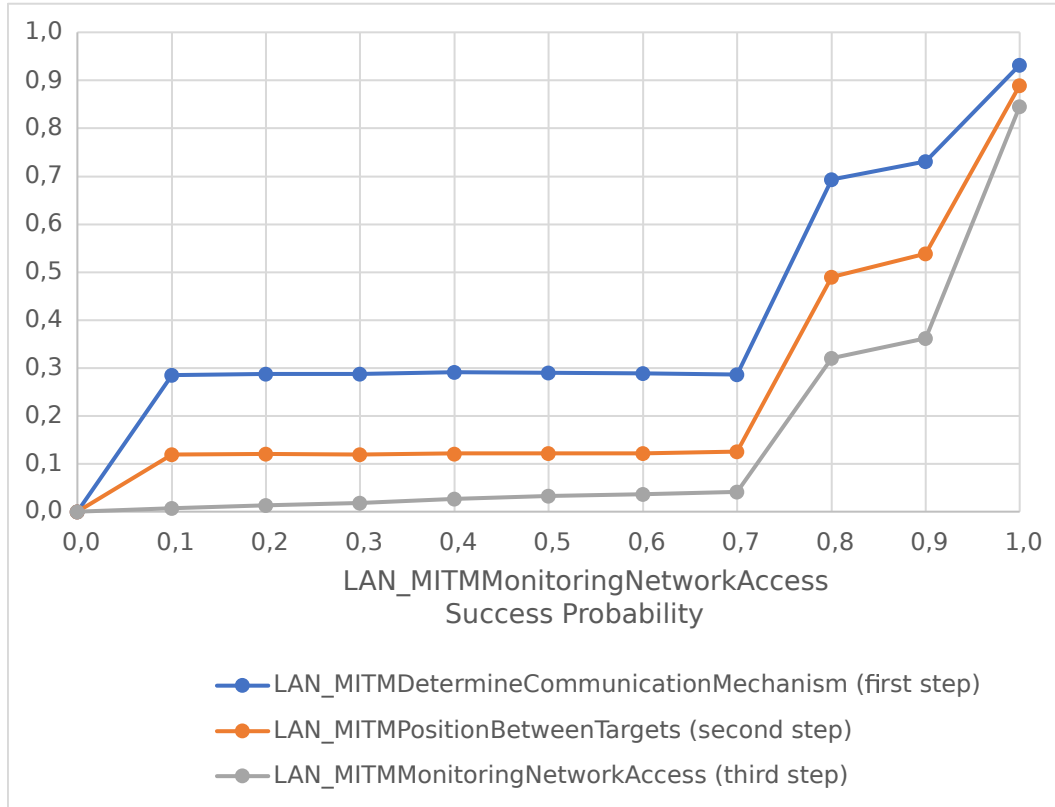


Figure 10: Probability of successfully achieving the *LANUseInterceptedData* goal for the Terrorist adversary, as the *Success Probability* parameter of the *LAN_MITMMonitoringNetworkAccess* attack step varies. The probabilities of successfully completing each of the three attack steps composing the MITM attack are shown. Figure extracted from [28].

to reach the other goals. In particular, the Resources attribute of this adversary is of type Individual, therefore the adversary's profile in ADVISE Meta has the Skill Resources with value 0. Most of the involved attacks instead require that the adversary has at least some club-level resources (e.g., the Flooding attack, see the Precondition Expression in [subsection 3.4.2](#)), meaning that in ADVISE Meta the Skill Resources must have value greater or equal to 200 (see [Table 2](#)).

Comparing the results obtained for the Vandal and Terrorist adversaries, we can see that the probabilities that the Terrorist can achieve each goal is usually higher than those of the Vandal (in some cases for the Vandal it is even equal to 0). However, on some goals (e.g., *Firewall2UnableToService*), the probability of success obtained by the Vandal is instead higher than what is achieved by the Terrorist.

In fact, some goals (e.g., *Firewall2UnableToService*) are actually much more attractive for a Vandal adversary than for a Terrorist, since the latter will have other exploitable attack paths that are more attractive and more convenient to attempt.

3.7.2 Analysis at Varying of the Attack Step Success Probability

As mentioned in [section 2.2](#), there are several parameters embedded in the ADVISE Meta framework (and in the generated ADVISE model) whose setting can have an impact on the adversary's behavior and on the probability of successfully reaching a goal.

In this section we perform a sensitivity analysis considering one of these parameters, the *Success Probability* property defined for each attack step.

The analysis has been performed considering the Man In the Middle attack pattern (already discussed in [Figure 9](#)) and the Terrorist profile. For successfully completing the Man In the Middle attack and reaching the *UseIntercepted-Data* goal, the adversary must successfully execute three sequential attack steps: *LAN_MITMDetermineCommunicationAccess*, *LAN_MITMPositionBetweenTargets* and *LAN_MITMMonitoringNetworkAccess*.

The three plots in [Figure 10](#) show the probabilities of successfully completing the three attack steps at varying of the *Success Probability* of the *LAN_MITMMonitoringNetworkAccess* attack step (the last step before achieving the goal). It is interesting to note that when increasing this probability within the interval [0.1; 0.7], the computed metrics remain almost constant: this means that in the identified interval this parameter is actually not affecting the adversary's behavior. On the contrary, when the *Success Probability* becomes greater than 0.7 we can note a steepening of all the three curves: not only the one directly describing the *LAN_MITMMonitoringNetworkAccess* attack step (the bottom one, in gray), but also the ones of the two preceding attack steps. In fact, for values greater than 0.7, the Man In the Middle attack becomes more appealing for the adversary, thus increasing the attack attempts and, in turn, increasing the probability of successfully completing the attack in the considered time interval (96 hours).

3.7.3 Analysis at Varying of Implementation Characteristics of the System

[Figure 11](#) shows the results of a simulation in which two different system configurations are compared. The first uses a low authentication level (as in the previous scenario), while the second was configured with a high system's authentication level, i.e., more severe restrictions are placed on passwords. In the architectural model, the second configuration is represented by setting the *strengthOfUserAuthentication* parameter to 4, on each component of the system.

The results shown in the plot refer to a subset of the goals considered in [subsubsection 3.6.2](#), evaluated only for the Terrorist adversary, on the two configurations. As it can be immediately noted, for most goals the success probability with the "strong" authentication configuration is significantly lower than with the "weak" authentication configuration, and in most cases it is even zero.

Conversely, for some goals the probability of the attacker achieving them *increases* with the stronger authentication method. It is the case, for example, of the *LanNetworkAccess* goal: with a strong user authentication, the adversary (following the execution algorithm mentioned in [section 2.1](#)) "realizes" that reaching some goals (e.g., the *Firewall1LogicalAccess* goal) would be too costly, and then she/he focuses on other goals that are more rewarding (e.g., the *LanNetworkAccess* goal).

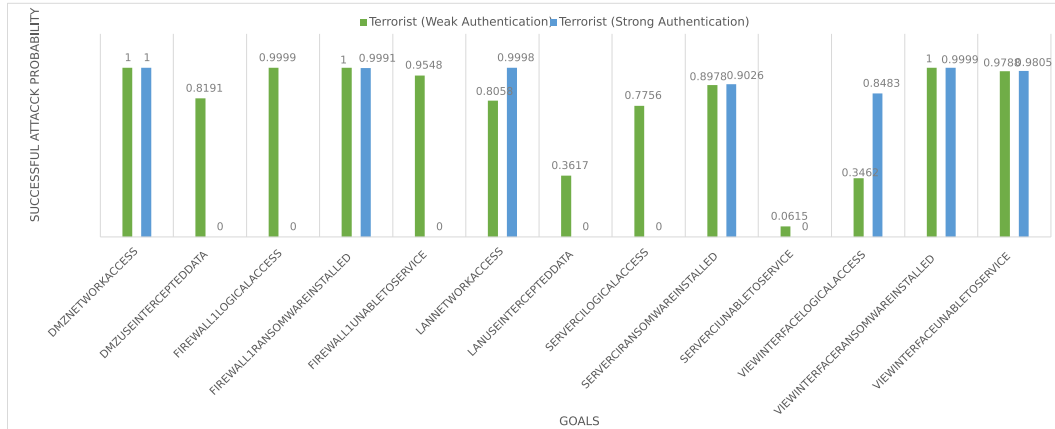


Figure 11: Probability of successfully achieving goals for the Terrorist adversary as the user authentication level varies. Figure extracted from [28].

Note that this does not mean that the adversary is able to cause more damage, but only that it is more likely to access certain parts of the system.

3.7.4 Analysis of Most Probable Attack Paths

As explained at the end of [section 2.1](#), in order to reach a goal, the adversary will follow the attack path that is more attractive depending on the adversary's skills and on the characteristics of the target system. It is thus possible to observe how the attack path followed by an adversary changes as some characteristics of the system vary.

In [Figure 12](#) a simplified representation (ADVISE style) of a combination of attack paths for the Terrorist adversary is shown. In this example the goal considered by the adversary is the *ServerCILogicalAccess* goal. Inside each yellow box representing an attack step, its success probability derived from the simulation is represented in square brackets. For this particular example, we computed the probability of completing the attack up to 24 time units (hours). Similarly to what have been done in [subsection 3.7.3](#), we ran two different simulations, one where the *strengthOfUserAuthentication* parameters of the two firewalls are set to "weak", and one where they are set to "strong".

It is possible to observe that, after the second attack step, two different attack paths are followed by the adversary. In the case of "weak" firewalls' authentication, the adversary decides to try three different attack steps: the adversary first tries to gain the logical access to the second firewall (i.e., she/he does not have the necessary preconditions), then to gain access to the LAN network, and finally to gain access to the Server CI. In the other case, when the firewalls' authentication is "strong", the adversary is not able to gain logical access to the second firewall, so she/he will try to circumvent the firewall rules. Even if the attack path is shorter (i.e., fewer attack steps) than the other, the success probabilities are significantly lower.

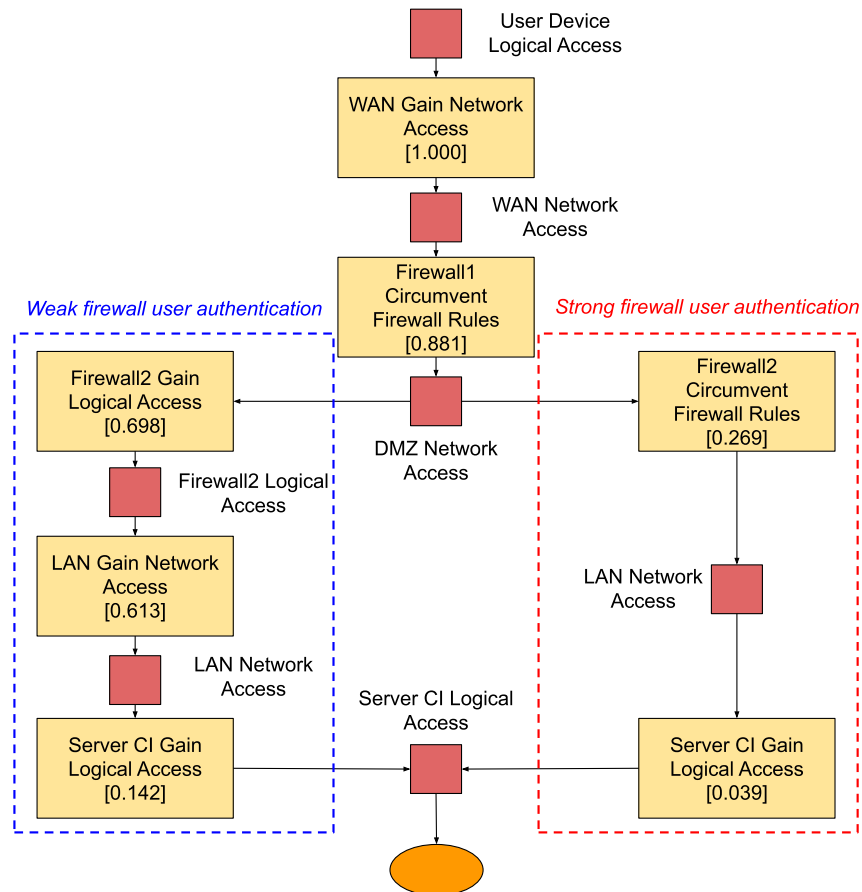


Figure 12: Simplified ADVISE model for the Terrorist adversary with the *ServerCI-LogicalAccess* goal. Two different attack paths are taken by the adversary, depending on whether weak or strong firewall authentication is implemented in the system. For each attack step, its success probability is represented in square brackets. Figure extracted from [28].

3.7.5 Analysis of the Exposure Level of System's Components

On the basis of broad analyses such as the one shown in subsection 3.7.1, it is also possible to obtain qualitative indications on the level of exposure of the system components. As we have already shown (e.g., in subsection 3.7.1), we are able to compute the probability that an adversary can successfully reach a specific goal associated to a component. Now we define the exposure level of a component as the average probability that an adversary successfully achieves all the goals associated to the component (i.e., she/he successfully attacks the component, thus achieving the associated goals), multiplied by 10. This average could also be weighted according to particular needs. Once an adversary is fixed, the exposure level of a certain component is therefore a score ranging from 0 to 10, which indicates how vulnerable the component is to a set of attacks carried out by the specified adversary, in the considered time window.

Figure 13 shows the exposure level of the SPaCe system components for the Vandal and Terrorist adversaries. The considered system configuration is the same

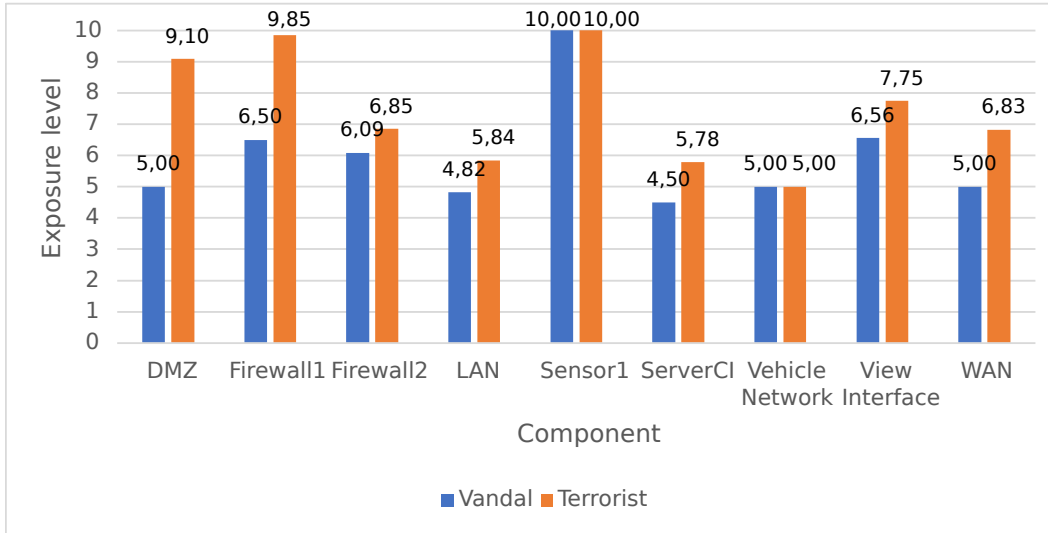


Figure 13: Exposure level of SPaCe components for Vandal and Terrorist adversaries. Figure extracted from [28].

used in subsection 3.7.1, i.e., with the low authentication level. Note that the exposure level of components changes against different adversaries. In this case, for all the components, the exposure level against the Terrorist adversary is always equal to or higher than against the Vandal adversary. For the considered attacks and the target system, the Terrorist is therefore the most dangerous adversary. Besides Sensor1, which is the most exposed component for both adversaries, we have that i) for the Terrorist, the second and third most exposed components are Firewall1 and DMZ, but, ii) for the Vandal, the second and third most exposed components are instead View Interface, and Firewall1.

It is not surprising that these are among the most easily accessible components to adversaries: the sensors are available on board of vehicles, while the Firewall1 and the DMZ (and the View Interface) can be reached by users connected to the WAN. Paying more attention to these components is certainly a first step in strengthening system security because, on one side, the sensors represent the entry point for sensitive data coming from vehicles and, on the other side, the Firewall1 and the DMZ represent a first point of defense for the entire network infrastructure.

3.8 CONCLUSIONS

In this chapter, we have shown a methodology for the extension of the ADVISE Meta ontology, in order to consider the attack patterns from the CAPEC catalog and the adversaries' profile from the TAL. Thanks to a mapping between ADVISE, CAPEC, and TAL taxonomies, we were able to integrate into the ADVISE Meta ontology all the TAL adversaries and some representative CAPEC attacks.

This extension enabled us to perform a broad-range security analysis on a case study, that is the SPaCe system. After defining the architectural model of the system inside the ADVISE Meta framework, we were able to generate the low-level security models. Thanks to the nature of the framework and the contribution of

the extension, several kind of analysis were performed, both targeting quantitative metrics, like the probability that a given adversary can successfully reach a particular goal, and qualitative metrics, like ranking the system's components based on their estimated exposure level. We compared different implementation solutions, we analysed different system's scenarios at varying of the adversaries' profiles, and we analysed the most probable attack path that can be followed by the adversary to reach the goal.

The extension shown in this chapter was performed manually. If we consider the vastness of the CAPEC catalog, its manual integration into the ADVISE Meta ontology is a time-consuming process. To understand which are the prerequisites of each attack, we have to interpret the descriptive sections contained in each CAPEC entry. Then, we have to manually add to the ontology each attack step using the UI of the framework.

To overcome this issue, in [chapter 4](#) we will illustrate a methodology for the semi-automated integration of the CAPEC attacks into the ADVISE Meta ontology with the support of AI.

SEMI-AUTOMATED INTEGRATION OF CAPEC ATTACKS INTO ADVISE META ONTOLOGY

In this chapter we present a semi-automated approach for the integration of the CAPEC attack patterns into the ADVISE Meta ontology, using ChatGPT as a support tool. This chapter is based on [50], and partially on [1].

4.1 INTRODUCTION

In [chapter 3](#) we presented a methodology for the integration of TAL adversaries and CAPEC attack patterns into the ontology of the ADVISE Meta framework. We integrated all the twenty-one TAL adversaries, and a few representative CAPEC attacks, which however did not cover the vastness of the CAPEC repository. This extension was done manually.

The manual incorporation of CAPEC attack patterns into ADVISE Meta ontology is a time-consuming process. The modeler should carefully read the descriptive sections of the CAPEC entry to retrieve the meaningful information to be inserted into the ontology. This operation should be repeated for all the five hundred attack patterns of the CAPEC database. To solve this concern we propose to rely on artificial intelligence as a supportive means for the semi-automated integration of CAPEC attacks into ADVISE Meta ontology.

ChatGPT [51] is a large language model (LLM) suitable for a variety of uses thanks to its scalability, adaptability, and efficiency [52]. Despite some drawbacks like potential bias, absence of emotional understanding, and a finite knowledge base, these issues can be addressed through thoughtful selection of training data and additional programming. ChatGPT can also be used for identifying security vulnerabilities and generating proof-of-concepts [53]. It can assess code for security weaknesses, offering clear explanations on how it identifies them. Moreover, it can facilitate functional proof-of-concept demonstrations of exploits that leverage the identified weaknesses.

The proposal of utilizing ChatGPT as a support tool for the semi-automated integration CAPEC attacks into the ADVISE Meta ontology came after an investigation conducted on other AI chat systems, e.g., Microsoft Copilot [54] or Google Gemini [55]. In our experience, the other AIs seem to have shortcomings in both answers' format and accuracy compared to ChatGPT. Additionally, it was discovered that other AIs offer no advantage in terms of maximum input length, as they can handle equal or even shorter inputs compared to ChatGPT.

We used GPT-3.5 since it is free to use and easily accessible to all users. An alternative is GPT-2, which is free and open-source [56] but is less powerful and

requires a manual configuration by the user. GPT-4 is the newest powerful version made available but is not free to use.

Both CAPEC database and ADVISE Meta ontology are available in XML format, so they can be easily processed by ChatGPT. The idea is to ask ChatGPT to interpret the CAPEC attack patterns according to some information that is provided as input:

- the XML file of CAPEC database [57];
- the CAPEC, TAL, and ADVISE Meta elements and their definitions;
- the mapping between CAPEC, TAL, and ADVISE Meta elements, as suggested by [Figure 4](#) in [chapter 3](#).

Afterward, we can ask ChatGPT to insert the processed CAPEC attacks into the XML file of the ADVISE Meta ontology.

Note that in our approach we use the AI as a support tool for the interpretation of the textual CAPEC sections and to insert the interpreted data into the ADVISE Meta ontology to create the attack steps. This is done by providing the AI with the rules on how to do that. This means that we do not train the AI model.

Through this extension, the ADVISE Meta tool will be equipped with a plethora of adversaries' profiles and attacks. The user who aims to model a system and analyze it from a security perspective could then select the adversaries and attacks of interest from those available in the extended ontology, for example, considering those related to a specific application domain.

The rest of the chapter is organized as follows. In [section 4.2](#) we briefly overview works that use LLMs for interpreting cybersecurity catalogs. In [section 4.3](#) we present the details of the methodology for the semi-automated integration of CAPEC attacks into the ADVISE Meta ontology, using ChatGPT. In [section 4.4](#) we show how the ontology can be modified in case of updates of the CAPEC database entries. Lastly in [section 4.5](#) we discuss some aspects of the approach which require more clarification.

4.2 RELATED WORK

In the literature, other authors have proposed to rely on LLMs to interpret the textual content of attack databases like CAPEC.

In [58] the authors proposed a LLM framework for mapping CVE vulnerabilities [9] to ATT&CK techniques. During the process, the models also identify possible related CWEs and CAPEC attack patterns.

The work in [59] evaluated two different usages of ChatGPT: map CVE to CWE and map CVE to ATT&CK techniques. In the tests performed, ChatGPT accomplished better results in the first activity, rather than the second.

The authors of [60] consider the problem of classifying a textual description of a tactic according to the ATT&CK techniques and CAPEC descriptions. They compare the usage of LLMs (like GPT) against fine-tuned small-scale LLMs. The second group seems to obtain better results.

The work in [61] is a survey providing an overview of the usage of LLMs in cybersecurity. It includes a section encompassing vulnerabilities and cybersecurity threats, including text-to-text problems, like mapping CWEs to CAPEC.

In [62] different algorithms are used, such as term frequency-inverse document frequency (TF-IDF), Universal Sentence Encoder (USE), and Sentence-BERT (SBERT), to identify common links between CAPEC-ID and CVE-ID. The quality and completeness of the CVE and CAPEC datasets depend on the provided algorithms.

The authors of [63] provided an evaluation of the capability of LLMs in solving CTF (Capture The Flag) to fully automated workflow process in a way to pave the approach for systematic evaluation of cyber security capabilities in LLMs.

The work presented in [64] explored the feasibility of using LLMs like ChatGPT to create attack trees for specific scenarios. In this study, ChatGPT is asked to generate an attack tree based on a provided scenario. The process involves presenting the scenarios first, followed by the prompts used to guide ChatGPT in creating the trees.

4.3 METHODOLOGY FOR THE INTEGRATION OF CAPEC ATTACKS

Figure 14 shows the methodological steps used to integrate the CAPEC attack patterns into the ADVISE Meta ontology. The operations are grouped in the following macro-steps:

1. Preliminary operations: preparatory operations are performed to filter the relevant data from the CAPEC attack patterns that will be used in the subsequent steps.
2. Data extraction: the data are extracted from the CAPEC attack patterns, and values are assigned according to the mapping between CAPEC, TAL, and ADVISE Meta elements, which is provided to ChatGPT as data extraction and interpretation rules.
3. Creation of the attack steps: the attack steps in XML format are created using an attack step template and the values extrapolated in the previous macro-step.
4. Assignment of the attack steps to the components of the ontology: the attacks are assigned to the architectural components available in the ontology.
5. Final insertion: the attack steps are finally added to the XML file of the ADVISE Meta ontology.

The attack patterns that were integrated into the ontology all belong to the "Meta" abstraction level. This was done because the ADVISE Meta ontology is characterized by a rather high level of abstraction. Therefore, the integration of "Standard" attack patterns would limit their use to specific types of attack scenarios and increase the number of duplicates within the ontology. Attacks with "Meta" abstraction, instead, are reusable in multiple scenarios and can be employed abstracting from the application domain.

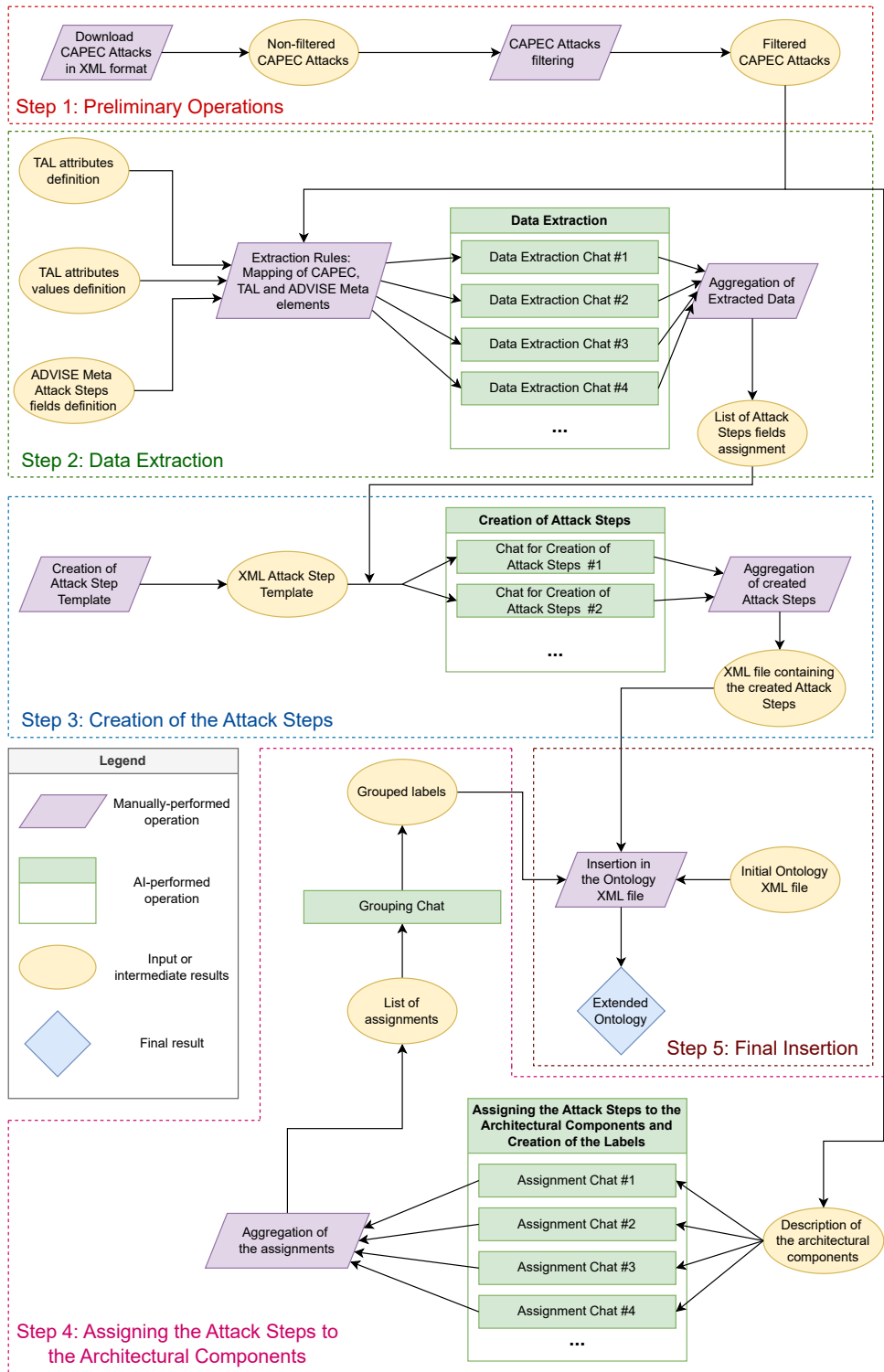


Figure 14: Methodological steps for the semi-automated integration of CAPEC attacks into ADVISE Meta ontology.

4.3.1 Step 1: Preliminary Operations

In this section, we will explain which are the operations needed to prepare data and chats in order to use them in the process.

Download of CAPEC database in XML format

The data concerning the CAPEC attack patterns are extrapolated from the XML file containing all the attack patterns present in the CAPEC database called "1000.xml" [57]. The reason for using this XML file as a reference, instead of directly using the data present on the web pages of the CAPEC website, is given by the inability of ChatGPT to directly access the pages of the attacks (this limitation was also observed for the other tested AIs).

Filtering of the attack patterns

A few filtering operations are performed over the XML file to create a file containing only the CAPEC attack patterns with "Meta" abstraction and containing only the fields actually useful for the analysis.

Note that these filtering operations must be carried out manually, due to the size of the file which prevents the use of ChatGPT (at least the free version). In the first step, all the attack patterns which do not have "Meta" abstraction are removed. Afterward, the attack patterns that do not contain enough information to allow the correct construction of the attack steps will be eliminated. These attacks are in "Draft" status, i.e., incomplete and subject to future changes, containing an insufficient amount of fields for the correct extrapolation of the data (for example they contain only the Description section with a very abstract description of the attack). The next step is to eliminate sections of the CAPEC attack patterns that are not used in the analysis, which are:

- Content_History;
- Taxonomy_Mappings;
- Related_Weaknesses;
- Related_Attack_Patterns;
- Mitigations.

4.3.2 Step 2: Data Extraction

In this section, we discuss the mapping and assignment of values to the fields of the attack steps, and how to extract the field values from the resulting XML file after performing the filtering described in [subsubsection 4.3.1](#).

From the tests carried out on data extraction and attack steps construction, we experienced that it is necessary to use different chats for the several steps that constitute the methodology. This is because the AI's accuracy in understanding and executing instructions degrades as the chat gets longer. This effect is especially

noticeable during the data extraction phase. In fact, since the filtered file is too long to be passed entirely to the chat, it will have to be divided into several parts that will be passed and processed sequentially by the artificial intelligence chat. If, for example, the first ten attack patterns are passed, the data extraction will be performed correctly and the returned result will be in the correct format of a list containing the fields of the attack steps and the values assigned to them. However, if the data extraction operation is continued in the same chat, once the new attack patterns have been passed, the result returned will no longer be in an acceptable format, returning a verbose description of the attacks instead of the list of fields with the associated values. The only experienced way to overcome this problem is to use a chat for each group of attack patterns from which to extract the data.

Mapping of CAPEC, TAL, and ADVISE Meta elements

In order to properly construct attack steps to be used in the ADVISE Meta framework, we have to understand from which CAPEC sections we can extract the attack step fields. The ADVISE Meta attack steps' fields considered are the following:

- *ID*: is the identifier of the attack step. We use the "ID" attribute of the CAPEC attack pattern. For example, the attack step corresponding to the Flooding attack pattern will have an ID equal to 125.
- *Name*: is the name of the attack step. We use the "Name" attribute of the CAPEC attack pattern.
- *Precondition Expression*: determines which are the requirements that an adversary must have in order to attempt the attack. In our context, it is expressed in terms of TAL attributes, following the methodology presented in [section 3.3](#).
- *Attack Cost*: represents the cost that the adversary will have to pay in order to execute the attack step. We use the same value as the TAL Resources attribute since the attribute that represents the level of resources needed to execute an attack is also the one that most describes the cost of that attack. The value of the Resources attribute can have a value of 0, but an attack will never have zero cost. Therefore, it was decided to put the value 1 as the lower limit.
- *Outcome Probability (Success or Failure)*: the probability that the adversary will successfully (or not) complete the attack step. The "Likelihood of Attack" section of the CAPEC attack pattern is used, which indicates, using a qualitative level, the possibility that the attack can be successfully performed. Since the probability is represented in ADVISE Meta by a decimal value, for the Success Probability we can set the following mapping, that can be adapted according to the users' needs (the Failure Probability is given by 1 - Success Probability):
 - Very Low: 0.1
 - Low: 0.3
 - Medium: 0.5
 - High: 0.7

– Very High: 0.9

- *Effect*: represents the outcome in terms of ADVISE elements (e.g., Access) that will be obtained by the adversary after attempting an attack step, both in case of success or failure. We use the "ID" attribute of the CAPEC attack pattern to build the name of the affected element, by appending it to the "ACCESS_" string. For example, the effect of the Flooding attack will be named "ACCESS_125".

Data extraction process

First, we have to instruct the AI so that it can interpret the text contained in the various CAPEC sections, extract the data and assign the values correctly to the various fields of the attack step. The task of the first prompt is therefore to define and describe the fields and values that the AI will have to assign to the fields, extracting them from the texts of the attack patterns analyzed. The definitions of the TAL attributes, and their descriptions, are taken from the official TAL documentation [24], while the attack step fields are described and derived as in [subsection 4.3.2](#).

The prompt used is defined in [Listing 1](#).

Listing 1: Prompt for the definition and values assignment of TAL attributes and attack steps' fields.

Learn these attributes definitions:

Intent:

This defines whether the agent intends to cause harm. Agents fall into two categories depending on their intent:

Hostile: The agent starts with the intent to harm or inappropriately use Intel assets, and the agent takes deliberate actions to achieve that result.

Non-Hostile: The agent is friendly and intends to protect Intel assets, but accidentally or mistakenly takes actions that result in harm.

Access:

This defines the extent of the agent's access to the company's assets.

There are two options:

Internal: Agent has internal access.

External: Agent has only external access.

[Continue for the rest of the TAL attributes and attack steps fields]

Then assign the following values to the attributes of the definitions:

For Intent, if "Hostile" then 1 else if "Non-Hostile" then 0. The value of this attribute is deduced using the Description and Extended Description fields' text.

For Access, if "Insider" then 1 else if "Outsider" then 0. The value of this attribute is deduced using the Prerequisites, Description, Extended Description and Consequences fields' text.

[Continue for the rest of the TAL attributes and attack steps fields]

The AI's response, shown in [Listing 2](#), confirms that ChatGPT correctly memorized the description and how to assign values to the TAL attributes and attack steps' fields. It can happen that the AI response is not formatted like in [Listing 2](#). When this occurs, if we continue with the data extraction, the data will not be extracted or returned correctly, leading to precision errors or unusable response formats. For example, it might happen that the AI does not extract numerical data but returns verbose descriptions of the attacks, or it might assign to the fields some numerical values different from those defined.

Listing 2: Answer of ChatGPT to the description and assignment of TAL attributes and attack steps' fields.

Here is the assignment of values to the attributes of the definitions according to the rules you provided:

1. Intent

Hostile: 1

Non-Hostile: 0

Source fields: Description, Extended Description

2. Access

Insider: 1

Outsider: 0

Source fields: Prerequisites, Description, Extended Description, Consequences

3. Limits

Code of Conduct: 250

Legal: 500

Extra-legal, minor: 750

Extra-legal, major: 1000

Source fields: Typical Severity, Description, Extended Description

Typical Severity values:

Very Low: 250

Low: 500

Medium: 750

High/Very High: 1000

4. Resource

Individual: 0

Club: 200

Contest: 400

Team: 600

Organization: 800

Government: 1000

Source fields: Resources Required, Description, Extended Description

5. Skill Level

None: 0

Minimal: 250

Operative: 750

Adept: 1000

Source fields: Skill and its Level attribute's value, Description, Extended Description

6. Visibility

Overt: 1000

Covert: 500

Clandestine: 250

Don't Care: 0

Source fields: Description, Extended Description

7. Chance of Success

Very Low: 0.1

Low: 0.3

Medium: 0.5

High: 0.7

Very High: 0.9

Source fields: Likelihood of Attack (default to "Medium" if not present)

8. Effect

Format: "ACCESS_" + CAPEC id

9. Name

Format: Name of the attack without spaces

10. Id

Format: CAPEC id of the attack

11. Attack Cost

Value: Equal to the Resource attribute

Special case: If Resource attribute value is 0, then Attack Cost is 1

If the format of the AI response is correct, it is possible to pass to the chat the group of attack patterns from which the data can be extracted. The attack patterns are passed to the AI in the filtered XML format described in [subsubsection 4.3.1](#).

The amount of attack patterns that can be passed to the AI depends on the length and complexity of the latter, but a general rule, empirically derived from the tests carried out, is to pass a maximum of ten attack patterns per chat. It is possible to check that the length of the input is manageable by ChatGPT, using a tokenizer made available by OpenAI [65]. The site calculates the number of tokens that a text will request, so it will be possible to check that this text is manageable by the AI without running the risk of compromising the results of the chat.

[Listing 3](#) shows the prompt used to ask ChatGPT to interpret the CAPEC attack patterns, passed in XML format, in particular an example for the "CAPEC-125: Flooding" attack pattern is reported. [Listing 4](#), instead, shows the interpretation of ChatGPT for the Flooding attack.

Listing 3: Prompt for the interpretation of "CAPEC-125: Flooding" attack pattern.

Now, using the attributes and the values that we defined, describe the following attack patterns:

```
<Attack_Pattern xmlns:capec="http://capec.mitre.org/capec-3"
ID="125" Name="Flooding" Abstraction="Meta" Status="Stable">
<Description>An adversary consumes the resources of a target
by rapidly engaging in a large number of interactions with the target
[...]</Description>
<...> ... <...>
```

Listing 4: ChatGPT interpretation of "CAPEC-125: Flooding" attack pattern.

Here is the detailed assignment of values to the attributes for the attack pattern "Flooding" with ID "125":

Intent

Value: 1 (Hostile)

Source: Description indicates malicious intent to exhaust resources and prevent legitimate access.

Access

Value: 0 (Outsider)

Source: Description and Prerequisites indicate an external attacker.

[...]

Summary of Assigned Values

Intent: 1

Access: 0
Limits: 750
Resource: 600
Skill Level: 250
Visibility: 1000
Chance of Success: 0.7
Effect: "ACCESS_125"
Name: "Flooding"
Id: "125"
Attack Cost: 600
This mapping provides a structured representation of the attack pattern attributes based on the provided rules.

Once the AI has returned a result, we have to check that this is correct and well-formatted. The result shall be returned in the shape of a list where the fields of the various attacks and their values are enumerated. If this does not happen, we will have to instruct the AI with the correct formatting without the need to reprocess the data in a new chat.

In the tests carried out, it was noted that the chat might assign the same values to the fields of different attacks. When this happens it can be assumed that the data extraction is incorrect and therefore a new chat will have to be used and the extraction will have to be executed again. The correct format of the answers, as mentioned above, shall be a list. The formatting details may vary slightly between different chats, but this will not affect the correctness of the data assignment in the next step.

The lists obtained from the various extraction chats are the data that shall be passed to the attack steps creation chats.

4.3.3 Step 3: Creation of the Attack Steps

In this section we illustrate how to create an attack step template in XML format and how to use it for the creation of the attack steps that will be included in the ontology.

Attack step template

As a first step, we create an attack step template so that, using placeholders, ChatGPT can populate it with the data obtained from the data extraction chat.

The XML skeleton of the attack step template can be obtained by creating an empty attack step through the ADVISE Meta framework and retrieving it from the XML file of the ontology.

Concerning the fields that constitute the attack step template in XML format, we use the following placeholders:

- [name] for the *name* field.
- [id] for the *id* field.

- return ([preconditions]); for the *preconditionExpression* field.
- return [cost]; for the *costExpression*.
- the value 0 is used for the *timingDistribution*. This is because, from all the descriptive fields of an attack pattern, it is impossible to extract the actual duration of an attack. Therefore it has been decided to assign it a default value, which will have a placeholder function and which will have to be changed by the user.
- [successChance] for the *probabilityExpression* of the "Success" Outcome.
- [effect] for the *effectsExpression* of the "Success" Outcome.
- [failureChance] for the *probabilityExpression* of the "Failure" Outcome.

Listing 5 shows the produced attack step template, which can be passed as input to ChatGPT.

Listing 5: Attack step template.

```

<classes xsi:type="advisem:0AttackStepClass" id="[id]" name="[name]"
  preconditionExpression=" return([preconditions]);"
  costExpression="return [cost];">
<preconditionElement label="theLimits">
<pathRoot xsi:type="ontology:PCGlobal" elementClass="//@classes.125"/>
</preconditionElement>
<preconditionElement label="theResources">
<pathRoot xsi:type="ontology:PCGlobal" elementClass="//@classes.126"/>
</preconditionElement>
<preconditionElement label="Intent">
<pathRoot xsi:type="ontology:PCGlobal" elementClass="//@classes.122"/>
</preconditionElement>
<preconditionElement label="theVisibility">
<pathRoot xsi:type="ontology:PCGlobal" elementClass="//@classes.127"/>
</preconditionElement>
<preconditionElement label="theSkills">
<pathRoot xsi:type="ontology:PCGlobal" elementClass="//@classes.129"/>
</preconditionElement>
<preconditionElement label="theAccess">
<pathRoot xsi:type="ontology:PCGlobal" elementClass="//@classes.92"/>
</preconditionElement>
<affectedElement label="[effect]"/>
<timingDistribution distributionType="Deterministic">
<parameters>return 0;</parameters>
</timingDistribution>
<outcomes name="Success" probabilityExpression="return [successChance];"
  detectionExpression="return 0.3;" effectsExpression="$ {[effect]}=1;"/>
<outcomes name="Failure" probabilityExpression="return [failureChance];"
  detectionExpression="return 0.7;" effectsExpression=""/>
</classes>

```

Attack steps' construction

The next step is to instruct ChatGPT with the way to correctly replace the placeholders in the attack step template. [Listing 6](#) shows the instructions used.

Listing 6: Prompt with the instructions for placeholders' replacement in the attack step template.

Now, when I'll pass you the data, I want you to replace the placeholders in the template with the data passed using the following instructions:

```
[name] with Name.
[preconditions] with the string
"({theResources}>=Resource)&&({theLimits}>=Limits)
&&({theSkills}>=Skill Required)&&
({Intent}>=theIntent)&&({theVisibility}>=Visibility)
&&({theAccess}>=Access)", where Resource, Skill Required,
Visibility, Limits, Access are substituted by the values contained
in the fields of the same name and theIntent is substituted by the
values contained in the Intent field.
[cost] with Attack Cost.
[successChance] with Chance of Success.
[effect] with the value of the Effect field.
[failureChance] with the result of
1 - (the value of Chance of Success field).
```

Concerning the precondition expression, we experienced that it is difficult with ChatGPT to correctly define preconditions different from those that depend on the TAL attributes, e.g., preconditions regarding the architectural components. Therefore, only prerequisites that use TAL attributes will be included, i.e. attributes' levels that an adversary will need to satisfy in order to carry out the attack. The definition of additional preconditions is left to the final users, who can perform this operation using the ADVISE Meta framework.

Similarly, to define the effects of an attack in case of success, an element named "ACCESS_[CAPEC attack id]" is created and left empty, since the information available is not enough to establish which is the real effect of the attack, that is system-dependent. The final user can then specify the effect that is most suitable for the scenario to represent.

Once the response from the AI has been obtained, the construction of the attack steps can begin.

The problem also encountered in the data extraction step ([subsection 4.3.2](#)) recurs during the construction of the attack steps. This implies that if we use a single chat, the results will be corrupted. As it is possible to observe in [Listing 7](#), the labels contained in *elementClass* have completely lost their meaning, the *affectedElement* has not been created, and a wrong value has been entered in the *timingDistribution*.

Listing 7: Example of corrupted ChatGPT answer concerning the attack step creation.

```
<classes xsi:type="advisem:0AttackStepClass" name="AuthenticationAbuse"
  precondition="true">
  <preconditionElement label="theLimits">
    <pathRoot xsi:type="ontology:PCGlobal" elementClass="//@classes.500"/>
  </preconditionElement>
  <preconditionElement label="theResources">
    <pathRoot xsi:type="ontology:PCGlobal" elementClass="//@classes.0"/>
  </preconditionElement>
  <preconditionElement label="Intent">
    <pathRoot xsi:type="ontology:PCGlobal" elementClass="//@classes.1"/>
  </preconditionElement>
  <preconditionElement label="theVisibility">
    <pathRoot xsi:type="ontology:PCGlobal" elementClass="//@classes.1000"/>
  </preconditionElement>
  <preconditionElement label="theSkills">
    <pathRoot xsi:type="ontology:PCGlobal" elementClass="//@classes.750"/>
  </preconditionElement>
  <preconditionElement label="theAccess">
    <pathRoot xsi:type="ontology:PCGlobal" elementClass="//@classes.0"/>
  </preconditionElement>
  <timingDistribution distributionType="Deterministic">
    <parameters>return 24;</parameters>
  </timingDistribution>
  <outcomes name="Success" probabilityExpression="return 0.5;"
    detectionExpression=""/>
  <outcomes name="Failure" probabilityExpression="return 0.5;"
    detectionExpression=""/>
</classes>
```

The solution also in this case is to use a different chat for each group of data passed as input. From the tests carried out, to maintain the correctness of the answers, it is enough to divide the list of data into two parts of twenty-five attack patterns each and pass each part to a different chat. Hence, to create an ontology containing CAPEC attacks with "Meta" abstraction we just need two chats for the creation of attack steps.

The prompt for the creation of the attack steps is shown in [Listing 8](#), while [Listing 9](#) reports an example of a correctly constructed attack step.

Listing 8: Prompt for the creation of attack steps.

Now, using the following attack steps data, create a new template for each attack and replace the placeholders with the value of said attack. The attack data are:

[Data copied from the data extraction chat]

Listing 9: Example of generated attack step.

```

<classes xsi:type="advisem:OAttackStepClass" id="549"
  name="LocalExecutionofCode"
  preconditionExpression="return((${theResources}>=800) &&&
  ${theLimits}>=500) &&& (${theSkills}>=750) &&&
  (${Intent}>=1) &&& (${theVisibility}>=500) &&&
  (${theAccess}>=0);" costExpression="return 800;">
<preconditionElement label="theLimits">
<pathRoot xsi:type="ontology:PCGlobal" elementClass="//@classes.125"/>
</preconditionElement>
<preconditionElement label="theResources">
<pathRoot xsi:type="ontology:PCGlobal" elementClass="//@classes.126"/>
</preconditionElement>
<preconditionElement label="Intent">
<pathRoot xsi:type="ontology:PCGlobal" elementClass="//@classes.122"/>
</preconditionElement>
<preconditionElement label="theVisibility">
<pathRoot xsi:type="ontology:PCGlobal" elementClass="//@classes.127"/>
</preconditionElement>
<preconditionElement label="theSkills">
<pathRoot xsi:type="ontology:PCGlobal" elementClass="//@classes.129"/>
</preconditionElement>
<preconditionElement label="theAccess">
<pathRoot xsi:type="ontology:PCGlobal" elementClass="//@classes.92"/>
</preconditionElement>
<timingDistribution distributionType="Deterministic">
<parameters>return 0;</parameters>
</timingDistribution>
<outcomes name="Success" probabilityExpression="return 0.5;"
  detectionExpression="return
0.3;" effectsExpression="${ACCESS_549}=1;"/>
<outcomes name="Failure" probabilityExpression="return 0.5;"
  detectionExpression="return
0.7;" effectsExpression=""/>
</classes>

```

4.3.4 Step 4: Assigning the Attack Steps to the Architectural Components

In this section, we show how to assign the CAPEC attack steps generated to the architectural system components available in the base ontology of the ADVISE Meta framework. This operation is necessary to allow the framework to generate the attack steps in the AEG. In fact, once the SID has been defined, the transformation engine looks to the attack steps enumerated in the *Dependants* field of the component in order to include them in the AEG.

The architectural components used for this purpose are those available in the base ontology of ADVISE Meta, described in [section 2.2](#).

Note that this operation is done to have just a first proposal of assignment. The final user can always decide to assign the attack steps to other components, even to brand newly defined ones, according to his or her needs (as done in [subsection 6.2.1](#)).

Representation inside the XML ontology file

First, we need to understand how the *Dependants* field is represented in the XML file of the ADVISE Meta ontology. Analyzing the file obtained from the export of the ontology, it is noticeable that the *Dependants* attribute is represented through a field in the component tag called *dependentElements*: in fact, labels will be inserted in this field, representing the elements (e.g., attack steps) related to the component (i.e., present in the *Dependants* field). These labels are in the form "`//@classes.[Appearance No.]`" where "`//@`" is the label prefix, "classes" is the type of tag involved, and "[Appearance No.]" is the number of the location where tag appears (e.g. "`//@classes.150`" indicates the 151st classes tag that appears in the ontology, since class tags are counted starting from 0).

Therefore, in order to assign attack steps to components, we assign a label to each attack step that will then be inserted in the *dependentElements* attribute of the component to which that attack step is assigned. During the final insertion of the attacks in the base ontology, particular attention must be paid to the order of insertion of the attack steps, established with the creation of the labels. If this order is not respected, there might be assignment errors or errors in the import of the ontology.

Mapping between attack steps and components

We use ChatGPT to assign a component and a label to each created attack step. To do so, we use two separate prompts: in the first prompt, the various architectural components and their definitions (those described in [section 2.2](#)) are provided to ChatGPT; in the second prompt, ChatGPT is instructed on how to create the labels and will perform the assignment using the attack steps passed. As for other previous steps, we need to use multiple chats, since the results will lose meaning after some assignments.

The prompt used to define and describe the components is given in [Listing 10](#).

Listing 10: Prompt for the definition of the architectural components.

Learn the following definitions of the component classes for an ADVISE Meta ontology:

- Application: software designed for the end user that performs a function in the system.
- Data: information in digital format that is created, stored or pro-cessed by the system. Examples are substation equipment settings, patient records, and bank account information.

[Continue for the rest of the architectural components]

The AI's response should be formatted as a list, where the components and their descriptions are present. After this, we perform the assignment of the attacks to the architectural components and the creation of the labels using the prompt given in [Listing 11](#).

Listing 11: Prompt for the the assignment of the attacks to the architectural components and creation of the labels.

Now assign the following attack patterns to the components. Please return a list with the name of the attack pattern, the component to which it is assigned and a string formatted like "`///@classes.[i]", where [i] is a counter that starts at 172 and increments by 1 for every attack pattern:`

[attack patterns in XML format]

In the ontology used, 173 classes are already present, including components, access, relationships, attack steps, and other kind of elements. To ensure that the ontology works correctly, we need to add each new element to the ontology in sequence. Therefore, the counter will start from the 172nd instance (since the count starts from 0), and it is necessary that, for each subsequent chat, the start of the counter is changed to the position after the last one assigned in the previous chat.

[Listing 12](#) shows the optimal answer of ChatGPT, that is, a list containing only the name of the attack, the name of the component to which it is assigned, and finally the assigned label. This formatting in fact simplifies the aggregation of the assignments so that they can be passed to the assignment chat.

Listing 12: An example of ChatGPT answer concerning the assignment of attacks to the components

Here's the list of attack patterns along with the components which they are assigned in the format you requested:

1. Interface Manipulation
Assigned to: 'Software'
String format: '`///@classes.172'`
 2. Authentication Abuse
Assigned to: 'Software'
String format: '`///@classes.173'`
 3. Authentication Bypass
Assigned to: 'Software'
String format: '`///@classes.174'`
-

Therefore, the obtained assignment can be inserted manually into the XML file of the ontology. In particular, the labels are added to the *dependentElements* attribute of the components. For example, concerning the assignment shown in [Listing 12](#), all the attack labels should be assigned to the *dependentElements* of the "Software" component.

4.3.5 Step 5: Final Insertion

To conclude the integration of attacks into the ontology it is necessary to insert the attack steps obtained from the creation chats (subsubsection 4.3.3) into the XML file of the ontology. This action is done manually, due to the inability of AI to read and process such an amount of text.

Particular attention must be paid to the position in which the attacks are inserted inside the XML file of the ontology. Each insertion must respect the position criterion used in the part of the assignment of the attack steps to the components (subsubsection 4.3.4). If the order is not respected, there will be assignment errors, i.e., attacks linked to the wrong components, or errors during the import of the ontology, i.e., the ontology will not be imported). Additional insertions must always be made at the end of the file.

To effectively use the attacks integrated into the ontology, it is necessary to manually assign one or more concrete elements (Access, Skills, Knowledge, etc.) associated with the components present in the ontology to the "affected" elements of the attacks. Moreover, the preconditions of the attack steps might be dependent on additional elements, e.g., on the relationships between specific architectural components.

Therefore, during the integration of the attack steps into the ontology, it is not possible to know a priori what the concrete effects of the attacks and the further preconditions will be. Hence, this last step is left to the user.

4.4 UPDATING THE ONTOLOGY

Many of the attack patterns in the CAPEC database are not yet fully studied or implemented and therefore are susceptible to future changes. To differentiate finished attack patterns from updatable ones, CAPEC uses an attribute called "Status": if the attack is completely studied the attribute "Stable" will be assigned to it, otherwise "Draft" will be assigned to it; for example, attacks deleted during filtering operations, since they do not contain enough information, are in a "Draft" status and therefore subject to future changes. This implies that the ontology created may become obsolete and therefore it is necessary to have a methodological approach to update it.

4.4.1 Check for updates

First of all, it is necessary to understand if there has been any update to the list of CAPEC attacks. To do this, check the "News & Events" page of the CAPEC website, where it is possible to find the news of possible updates. Otherwise, it is possible to check the current version of the list by accessing the "Latest Version" page and checking that the version shown at the top left is different from the used one.

Once we have understood that there has been an update of the list, to identify which attacks have been updated, or added, we have to access the "Report" section, that is located on the "CAPEC List" page, and select the "Content Difference Report" option. Using the summary we can easily see if there are any new attack patterns and which attacks have been updated.

4.4.2 *Perform the update*

Once the attacks to be updated have been identified, we can proceed with the actual update phase.

To carry out the update or integration of additional attacks, the operations seen in [section 4.3](#) will be performed, with the difference of being able to use a single chat for data extraction ([subsection 4.3.2](#)) and the creation of the attack step ([subsection 4.3.3](#)) (since the number of attack patterns added or modified will not be large). If the updates concern attacks already present in the ontology, then it will be enough to replace the attack step with its updated version without having to do any further operations. On the other hand, if the attack step is a new attack to be added, the assignment of the attack step to the architectural components ([subsection 4.3.4](#)) must also be carried out; once the label has been obtained, we just have to add the attacks at the end of the ontology and insert the label to the assigned component.

4.5 DISCUSSION

In this section, we discuss some aspects of the methodology which require further clarification. Open points and future work are also discussed in [chapter 7](#).

4.5.1 *Usage of AI*

In the proposed approach we use the ChatGPT AI as a support tool for the interpretation of the textual sections of the CAPEC attack patterns and, afterwards, to insert the attack steps in the ADVISE Meta ontology, properly filling the XML file with the interpreted information. Hence, we directly provide ChatGPT with the rules for the interpretation of the CAPEC sections, which is based on the mapping in [subsection 4.3.2](#).

An alternative approach could be to let the LLM learn the rules for interpreting the CAPEC attacks and for the insertion of the attacks in the ontology. In this case, we would need a sufficiently large amount of attack pattern samples to train the AI. Moreover, GPT-3.5 (and GPT-4) are not open source, so we would need to use GPT-2 which is however less powerful.

4.5.2 *Validation of the results*

One delicate and important aspect is the validation of the results provided by the AI. We mainly use ChatGPT to interpret the text of CAPEC sections and TAL attributes. This activity, even if performed by humans would be very subjective, possibly resulting in different interpretations. Hence, to validate the results, one could evaluate if the interpretation given by the AI is comparable to the interpretation that a human could reasonably give. For future work, we should perform some tests with security experts and, by sampling some of the interpretations given by ChatGPT, check if these interpretations reasonably match those provided by the experts. For example, suppose the AI reports that a prerequisite for the attack is to have an "Internal" *InsiderAccess* but the CAPEC attack pattern clearly states that

the attack can be also started from the outside. In this case, the AI has wrongly interpreted the attack pattern.

During the description of the methodology, we have already discussed other cases where it is more easily noticeable that the AI incorrectly interprets the CAPEC sections, e.g., when ChatGPT returns always the same interpretation to a set of attack patterns. This is the reason why we use separate chats, as explained in [subsection 4.3.2](#).

4.5.3 *Using other attack databases*

Following the approach presented in this chapter, the ontology of ADVISE Meta was extended by integrating the attacks contained in the CAPEC database. However, one might want to use a different database to obtain descriptions of various attacks, e.g., the MITRE ATT&CK [35]. In this case, it would be necessary to make some changes to some steps of methodology.

First, one would need to understand in what format the attacks are represented in the database. Then, it would be necessary to modify the filtering steps to take into account the formatting of the attacks. Next, one would have to modify the prompt in [Listing 1](#), adjusting the fields from which extract the attack's information. Once this step is complete, subsequent operations would remain the same as illustrated in the methodology.

4.6 CONCLUSIONS

In this chapter we have presented a methodology for the semi-automated integration of CAPEC attacks in the ADVISE Meta ontology, using ChatGPT as a support tool. The methodology allowed us to extend the ontology with all the CAPEC attack patterns having a "Meta" abstraction. This extension equips the ADVISE Meta framework with a well-founded collection of attacks that can be used to perform security analysis considering a broad spectrum of threats. In [chapter 6](#) we will apply this extension to a case study in the power grid domain.

The improvement implemented up to this point concerned the limitations of the base ontology of the framework in terms of attacks and adversaries. Still, the challenges regarding the assumptions of the ADVISE formalism, i.e., its execution algorithm and the fact that only the viewpoint of the adversary is considered (see [section 1.3](#)), are not yet addressed. In [chapter 5](#) we will propose a way to resolve this challenges, by implementing a model-to-model transformation from ADVISE to SAN.

MODEL-TO-MODEL TRANSFORMATION: FROM ADVISE TO SAN

In this chapter, we illustrate the model-to-model transformation from the ADVISE models generated by the ADVISE Meta framework to the SAN formalism.

5.1 INTRODUCTION

As mentioned in [section 1.3](#), the ADVISE formalism poses some challenges. First, the simulation of adversary behavior is constrained by the execution algorithm used in the ADVISE formalism, which imposes certain assumptions about the adversary's actions. For example, the algorithm forces the adversary to follow the attack path that minimizes cost. However, in reality, an adversary may select an attack path based on different considerations, such as choosing the shortest route. Another assumption made by the ADVISE execution algorithm is that the adversary can only exploit one attack path at a time, whereas in real-world scenarios, an adversary might launch multiple parallel attacks simultaneously. Lastly, the ADVISE Meta framework and ADVISE formalism focus solely on the adversary's perspective, without taking into account security defense mechanisms or the potential impact of attacks on the system.

To address these challenges, we decided to rely on a more flexible modeling formalism, that is SAN. To achieve this, we have to transform ADVISE models into SAN models. The first benefit is that we overcome the assumptions of the ADVISE execution algorithm, allowing us to simulate the adversary behavior across multiple attack paths simultaneously, without being restricted by decision parameters such as the attack cost.

Moreover, we enable the composition of the attack model with other models representing defense mechanisms and/or the system, in order to capture the interdependencies between safety and security. In principle, Möbius allows the composition of different atomic formalisms, like ADVISE and SAN. However, we experienced a few issues when composing the ADVISE models generated from the ADVISE Meta framework with SAN models. Furthermore, the assumptions made by the ADVISE formalism still would hold even if the composition with SAN was possible.

Note that we do not want to substitute the ADVISE approach. The ADVISE execution algorithm allows to obtain interesting insights on the adversary behavior if elements like the attack cost are known or justifiably estimated. However, we would like to explore situations where such information is not available and enable

different attack scenarios, like simulating the execution of multiple attacks in parallel.

To obtain a SAN model from an ADVISE one, we perform a model-to-model transformation. In Möbius, the atomic models are represented using XML files. This means that ADVISE models have their own XML syntax, as well as the SAN models. Therefore, we implement in Python such a transformation by exploiting the XML representation of ADVISE and SAN models.

The rest of the chapter is organized as follows. In [section 5.2](#) we provide a discussion on related works and alternatives to our approach used for the transformation. In [section 5.3](#) the mapping from ADVISE to SAN elements is provided. The rules for transforming the XML of an ADVISE model into the XML of a SAN model are given in [section 5.4](#). Finally, in [section 5.5](#) we provide some details of the Python implementation of the transformation.

5.2 RELATED WORK

5.2.1 *Model transformation*

Model transformation is known in model-driven engineering as the process of transforming a source model into another entity, which could be a model (model-to-model transformation, M2M) or another kind of artifact, e.g., source code (model-to-text transformation, M2T).

The Eclipse Modeling Framework (EMF) [66] is a widely adopted tool used in model-driven engineering processes. Viatra [67] is an open-source tool available in EMF for an event-driven, reactive model transformation engine built on top of highly scalable incremental graph queries. EMF is equipped with several other tools for model transformation ranging from pure M2M transformations, like ATL [68], to model-to-text (M2T) generation, like Acceleo [69]. Such tools can be used to handle XML files and can be fundamental in managing complex transformation problems, but require several efforts in order to properly configure them.

Refinery [70] is a novel open-source software framework to automatically synthesize a diverse set of consistent domain-specific graph models. To do this, a high-level language is used, allowing to partially define the initial models. Then, through transformation rules, it completes these partial models and transforms them into a variety of complete models in the form of a graph. This type of transformation is very useful in situations where you need to move from abstract and incomplete concepts to operational and usable models. Refinery is a powerful tool, but its complexity is high and it is not appropriate to directly work with XML representations.

5.2.2 *XML parsing and transformation*

Since both ADVISE and SAN models are represented in Möbius using XML, we can directly work on the XML file and transpose the model from the ADVISE syntax to the SAN syntax. There are several tools that allow to perform XML parsing.

The Document Object Model (DOM) [71], originally defined by the World Wide Web Consortium (W3C), is one of the oldest and most widely used XML parsing models. This model creates an abstract representation of the entire XML file as an in-memory tree. The DOM is very versatile and easy to use, making it suitable for small XML files where memory is not an issue. However, the construction of the DOM tree can be slow and requires the entire file to be loaded into memory, making it unusable with very large XML files.

The Simple API for XML (SAX) [72] is an alternative XML parsing model to DOM developed by the Java community. Unlike the DOM, SAX uses an event-based model that processes the XML file sequentially, triggering user-defined callbacks when it finds specific XML elements. This approach is known as "push" parsing, as the parser sends elements to the user's functions as it finds them. SAX is much more memory efficient than the DOM and can handle XML files of any size, but it is more difficult to use, especially when it comes to finding or modifying specific nodes within the XML tree.

The Streaming API for XML (StAX) [73] is another XML parsing model that is based on SAX but uses a "pull" parsing approach instead of a "push" one. With StAX, the user has more control over the parsing process, as elements are only processed when required.

BeautifulSoup [74] is a Python library that is primarily used for HTML data extraction, but it can also be used for XML parsing. It offers an API for extracting information, which can be very useful when working with malformed or error-containing XML files, due to its tolerance for invalid content.

lxml [75] is an advanced extension of the ElementTree Python library. While ElementTree offers a simple and lightweight interface for working with XML, lxml provides more powerful and advanced features. lxml is a Pythonic binding for the C libraries libxml2 and libxslt, hence it is highly performing. The library offers considerable flexibility and functionality, allowing to not only analyze and edit XML files but also create them from scratch. In addition, lxml allows the validation of XML files based on defined schemas, which is not supported by standard Python parsers.

Table 5 summarizes the pros and cons of the aforementioned approaches. lxml was the optimal choice for the transformation from ADVISE to SAN XML representations, due to the prior knowledge of the XML templates, the need for complex changes, and the capability of the library to navigate the entire XML document efficiently. This allows the transformation of the original ADVISE XML file into another format, i.e., the SAN XML file, with precision and efficiency.

5.3 MAPPING BETWEEN ADVISE AND SAN ELEMENTS

Before implementing the transformation from ADVISE to SAN, we have to understand which is the conceptual correspondence between the elements of the two formalisms. Figure 15 shows the mapping between ADVISE and SAN elements, which is discussed in the following subsections.

Table 5: Comparison between different XML parsing approaches.

XML parsing approach	Easiness of use	Quick elements navigation	Memory optimization	Read	Write	High performance	Python implementation
DOM	X	X		X	X		X
SAX			X	X		X	X
StAX	X		X	X	X		
BeautifulSoup	X			X	X		X
lxml	X	X		X	X	X	X

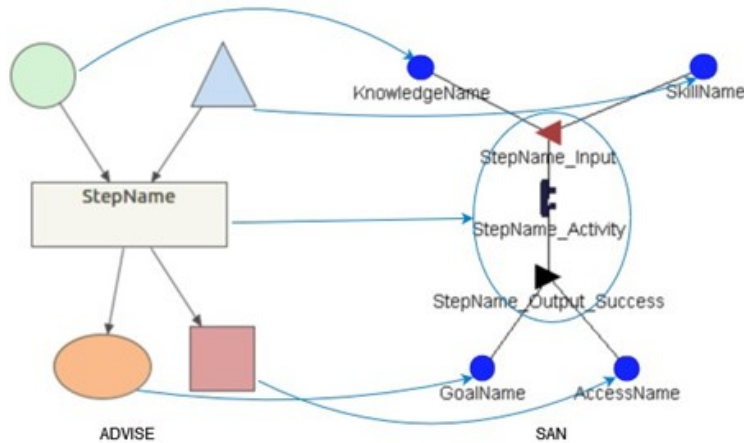


Figure 15: Mapping between ADVISE and SAN elements.

5.3.1 Skill, Access, Knowledge, and Goal

In ADVISE a Skill represents an ability or competence that an adversary possesses, and it is expressed with a value that can go from 0 to 1000, depending on the actual level of competence possessed by the adversary. The most suitable SAN element that can be used to represent a Skill is the Place, which can contain a non-negative integer number of tokens. Hence a Skill element is represented using a Place containing a number of tokens that corresponds to the Skill value.

Similar reasoning can be used for the Access and Knowledge ADVISE elements. In that case, the values admitted are only two: 0 (the element is not possessed by the adversary) and 1 (the element is possessed by the adversary). Therefore, those elements are represented in SAN by a Place, which contains 0 or 1 tokens depending on the element's value.

The same is done for the Goal, where, in this case, 0 and 1 tokens in the Place represent the fact that the adversary has not or has reached the goal.

5.3.2 Attack Step

Translating an Attack Step from ADVISE to SAN is a more complex process since a single Attack Step in ADVISE is composed of several elements. This requires splitting the Attack Step into multiple components in the SAN model, to accurately

capture the logic and dynamics of the attack. The various steps that an Attack Step performs are the following:

1. Check the preconditions: the Precondition Expression is verified in order to understand if the adversary can attempt the attack.
2. Determine the outcome: the outcome of the attack (Success or Failure) is determined according to the probability assigned to the various outcomes.
3. Attack execution: the attack is executed following the Attack Execution Time, and, according to the outcome, the corresponding Effect is performed.

To translate the ADVISE Attack Step in SAN we use the following elements:

- Input Gate: verifies the conditions necessary to enable the Timed Activity. Hence, the Precondition Expression of the Attack Step is mapped to the Enabling Predicate of the Input Gate. The Input Function of the Input Gate is left empty.
- Timed Activity: it represents the execution of the attack itself. The Timed Activity has its own Time Distribution, which we map to the Attack Execution Time of the Attack Step. Concerning the Outcomes of the Attack Step, in SAN we use the Activity Cases. The Probability of each ADVISE Outcome is used for the Probability of each SAN Activity Case.
- Output Gate: the Output Function of the Output Gate is mapped to the Effect of each Outcome of an Attack Step. Hence, for each Outcome of the ADVISE Attack Step, an Output Gate shall be defined.

5.3.3 *Arc*

The transformation of the arcs requires special attention, in particular, for arcs entering and exiting an Attack Step. The incoming arcs of the latter will correspond to the incoming arcs of the Input Gate, while the outgoing arcs will correspond to the outgoing arc of the Output Gate. In addition, new arcs must be generated corresponding to the links between Input Gates and Timed Activities, and between Timed Activities Cases and their corresponding Output Gates.

5.3.4 *Execution algorithm*

As explained in [chapter 2](#), the ADVISE execution algorithm is used in ADVISE models to simulate the adversary's behavior in deciding which attack to attempt. SAN does not implement such an algorithm. The execution of a SAN model depends on the enabling of its activities (according to the marking of connected places and input gates predicates) and firing time distributions. In the way we transform ADVISE to SAN models, we are not adding any additional rules for deciding priorities between attacks. In this way, when more than one attack is enabled at the same time, they will be executed in parallel.

SAN formalism is flexible and provides the possibility to write custom functions using C++ code. In this way, one could potentially implement a custom execution algorithm to explore attack strategies different from those used in ADVISE.

5.4 CORRESPONDENCE BETWEEN XML REPRESENTATIONS

In the previous section, we identified the mapping between ADVISE and SAN elements. After defining the methodology with which we would like to translate the ADVISE components into the respective SAN components, we have to understand how it is possible to apply this theoretical mapping to the actual representations in XML. As an example, we consider the most complex component, which is the Attack Step.

Listing 13 shows an example of XML representation of an Attack Step in ADVISE. The attributes that we have to consider for the transformation are *xpos*, *ypos*, *preconditions*, and *Success* and *Failure* outcomes, with their *probability*, *effects*, *distributionType* and value.

Listing 13: Example of XML representation of an Attack Step in ADVISE.

```
<stepElements key="StepName">
  <value name="StepName" codename="StepName" xpos="170" ypos="180"
    sourceArcs="//@aeg/@arcElements.2 //@aeg/@arcElements.3"
    targetArcs="//@aeg/@arcElements.0 //@aeg/@arcElements.1"
    attackExecutionGraph="//@aeg" width="150" preconditions="return
    (SkillName->Mark())>=100 && KnowledgeName->Mark()>=1;" cost="return
    1;">
    <outcomes key="Failure">
      <value name="Failure" probability="return 0.3;" detection="return
        0.7;" effects=""/>
    </outcomes>
    <outcomes key="Success">
      <value name="Success" probability="return 0.7;" detection="return
        0.3;" effects="AccessName->Mark()=1"/>
    </outcomes>
    <timing distributionType="Deterministic">
      <parameters>return 1;</parameters>
    </timing>
  </value>
</stepElements>
```

Listing 14 shows the the XML representation of the translation of the Attack Step in the first corresponding SAN element, that is the input gate. The attributes that have been taken into account to generate the first component are the following:

- value name="StepName" is associated with <string id="Name"> StepName_Input </string> (the word "Input" is appended to the end of the name);
- value ... xpos="170" is associated with <int id="x">170</int>;
- value ... ypos="180" is associated with <int id="y">130</int> (note that since we have to insert three components where before there was only one, the value of the "y" is decremented by 50);
- value ... precondition is assigned to <string id="Predicate"> return (SkillName->Mark() >= 100 && KnowledgeName->Mark() >= 1); </string>.

Listing 14: Example of XML representation of an Attack Step in SAN (input gate).

```

<class id="Mobius.AtomicModels.San.InputGatePanelObject">
  <int id="ClassVersion">1</int>
  <class id="Mobius.BaseClasses.BaseVertexClass">
    <int id="ClassVersion">3</int>
    <class id="Mobius.BaseClasses.BasePanelObjectClass">
      <int id="ClassVersion">5</int>
      <string id="Name">StepName_Input</string>
      <boolean id="ContainsLabel">true</boolean>
      <boolean id="IsLabelShown">true</boolean>
    </class>
    <point id="CenterPoint">
      <int id="x">170</int>
      <int id="y">130</int>
    </point>
  </class>
  <boolean id="MirrorImage">>false</boolean>
  <string id="Predicate"> return (SkillName->Mark())>=100 &&
    KnowledgeName->Mark()>=1);</string>
  <string id="Function">;</string>
</class>

```

The transposition of the Attack Step into the second corresponding SAN element (timed activity) is shown in Listing 15. The attributes considered are the following:

- value name="StepName" is associated with <string id="Name">StepName_Activity</string> (the word Activity is added to the end of the name);
- value ... xpos="170" is associated with <int id="x">170</int>;
- value ... ypos="180" is associated with <int id="y">180</int>;
- the <outcomes key="Success"> and <outcomes key="Failure"> are assigned to the respective cases in the input gate;
- the <timing distributionType="Deterministic"> and its respective parameter are assigned to <vector id = "Parameters"> <string id="0"> return1;</string></vector>.

Listing 15: Example of XML representation of an Attack Step in SAN (timed activity).

```

<class id="Mobius.AtomicModels.San.TimedActivityPanelObject">
  <int id="ClassVersion">2</int>
  <class id="Mobius.AtomicModels.San.ActivityPanelObject">
    <int id="ClassVersion">1</int>
    <class id="Mobius.BaseClasses.ExtendedBaseVertexClass">
      <int id="ClassVersion">1</int>
      <class id="Mobius.BaseClasses.BaseVertexClass">

```

```

    <int id="ClassVersion">3</int>
    <class id="Mobius.BaseClasses.BasePanelObjectClass">
      <int id="ClassVersion">5</int>
      <string id="Name">StepName_Activity</string>
      <boolean id="ContainsLabel">>true</boolean>
      <boolean id="IsLabelShown">>true</boolean>
    </class>
    <point id="CenterPoint">
      <int id="x">170</int>
      <int id="y">180</int>
    </point>
  </class>
</class>
<int id="NumberOfCases">2</int>
<vector id="CaseDistributions">
  <string id="0">return 0.3;</string>
  <string id="1">return 0.7;</string>
</vector>
</class>
<class id="Mobius.BaseClasses.Distributions.DistributionFunction">
  <int id="ClassVersion">1</int>
  <string id="Name">Deterministic</string>
  <vector id="MetaParameters">
  </vector>
  <vector id="Parameters">
    <string id="0">return 1;</string>
  </vector>
</class>
<string id="Activation"></string>
<string id="Reactivation"></string>
</class>

```

Listing 16 shows the the XML of the translation of the Attack Step in the last corresponding SAN element (output gate). The attributes assignment is the following:

- value name="StepName" is associated with <string id="Name"> StepName_Output_Success </string> (the word Output_Success is added to the end of the name, in case the output gate for failure is generated we add Output_Fail);
- value ... xpos="170" is associated with <int id="x">145</int> (we move left by 25);
- value ... ypos="180" is associated with <int id="y">230</int> (we move down by 50);
- the <timing distributionType="Deterministic"> and its parameter is assigned to <string id="Function"> StepName->Mark(); </string>.

Listing 16: Example of XML representation of an Attack Step in SAN (output gate).

```

<class id="Mobius.AtomicModels.San.OutputGatePanelObject">
  <int id="ClassVersion">1</int>
  <class id="Mobius.BaseClasses.BaseVertexClass">
    <int id="ClassVersion">3</int>
    <class id="Mobius.BaseClasses.BasePanelObjectClass">
      <int id="ClassVersion">5</int>
      <string id="Name">StepName_Output_Success</string>
      <boolean id="ContainsLabel">true</boolean>
      <boolean id="IsLabelShown">true</boolean>
    </class>
    <point id="CenterPoint">
      <int id="x">145</int>
      <int id="y">230</int>
    </point>
  </class>
  <boolean id="MirrorImage">>false</boolean>
  <string id="Function">AccessName->Mark()=1;</string>
</class>

```

5.5 IMPLEMENTATION OF THE TRANSFORMATION

In the previous sections we have seen how we can logically map the elements of the ADVISE and SAN formalisms and how we can translate the XML syntax of an ADVISE model into a SAN one. In the following we describe the algorithm that perform the transformation and a part of its implementation in Python.

5.5.1 Transformation algorithm

The main steps of the algorithm can be summarized as follows (note that steps 1, 12, and 13 must be performed manually):

1. Load the ADVISE model XML file;
2. Create an empty SAN model, i.e., an XML file with extension .san, with the same name as the ADVISE model. It will serve as the base for then adding the transformed elements from the ADVISE model;
3. Extract and convert Access elements to Place;
4. Extract and convert Knowledge elements to Place;
5. Extract and convert Goals elements to Place;
6. Extract and convert Skills elements to Place;
7. Extract and convert Attack Steps elements to Input Gate, Timed Activity, and Output Gate;
8. Insert the converted items into the SAN model;

9. Extract and convert Arcs;
10. Generate any missing Arcs;
11. Insert Arcs into the SAN model;
12. Create a Möbius project containing an empty SAN atomic model with the same name as the ADVISE atomic model;
13. Override the empty SAN model in the project folder with the SAN file built up through the transformation process.

5.5.2 Python implementation

In the following we show part of the implementation of the transformation algorithm using the lxml Python library, in particular of step 7, i.e., the transformation of the attack steps. The process of transforming an attack step involves the creation of three distinct components of the SAN model: an Input Gate, a Timed Activity, and one or one or two Output Gates, depending on whether the effect for failure is defined or not. In the following, we describe how the respective conversion functions work.

Parsing of the ADVISE XML file

The code begins by loading and parsing the input ADVISE XML file using the ElementTree (ET) library provided by lxml. A tree structure is created, allowing to navigate and edit the XML elements ([Listing 17](#)).

Listing 17: Python code for parsing of the ADVISE XML file.

```
tree = ET.parse(input_filename)
root = tree.getroot()
```

Creation of the Input Gate

A function called *convert_step_elements_to_inputgate* ([Listing 18](#)) creates an input gate for each attack step. For each of them, the *name* (to which the word "Input" is appended), the *xpos* and *ypos* coordinates, and the *precondition* expression are extracted. Sometimes there may be code within the precondition expression of the ADVISE model that might contain constructs that are not compatible with SANs, e.g., arrays of Access elements. For this reason, the *transform_cpp_code* function is called. This function has the purpose of recognizing incompatible patterns in order to modify them in a syntax compatible with the SAN model, e.g., transform arrays of Access elements into arrays of Place elements.

Next, we create the XML structure for the Input Gate object, including the name, location, and preconditions.

Listing 18: Python code for the creation of the Input Gate.

```

name = value_elem.get("name") + "_Input"
xpos = value_elem.get("xpos")
ypos = value_elem.get("ypos")
preconditions = transform_cpp_code(preconditions)

class_elem = ET.Element("class",
    id="Mobius.AtomicModels.San.InputGatePanelObject")
name_elem = ET.SubElement(base_panel_object_class, "string", id="Name")
name_elem.text = name
...
x_elem = ET.SubElement(center_point, "int", id="x")
x_elem.text = xpos
y_elem = ET.SubElement(center_point, "int", id="y")
y_elem.text = str(int(ypos) - 50)
predicate_elem = ET.SubElement(class_elem, "string", id="Predicate")
predicate_elem.text = preconditions

```

Creation of the Timed Activity

The *convert_step_elements_to_activity* function (Listing 19) takes care of transforming a step into a Timed Activity. Initially, the name (appending to it the string "Activity"), the position of the attack step, and the value of the time distribution are saved.

Then we proceed to create the XML structure for the Timed Activity and define its name and location.

The type of time distribution and the related parameters of the outcomes are then extracted from the ADVISE model and are then inserted into the XML structure of the Timed Activity.

Listing 19: Python code for the creation of the Timed Activity.

```

name = value_elem.get("name") + "_Activity"
xpos = value_elem.find("../layout//xpos").text
ypos = value_elem.find("../layout//ypos").text
parameter_value_element = timing_element.find("parameters")
parameter_value = parameter_value_element.text

class_elem = ET.Element("class",
    id="Mobius.AtomicModels.San.TimedActivityPanelObject")
name_elem = ET.SubElement(base_panel_object_class, "string", id="Name")
name_elem.text = name
...
xpos_elem = ET.SubElement(layout_elem, "xpos")
xpos_elem.text = xpos
ypos_elem = ET.SubElement(layout_elem, "ypos")
ypos_elem.text = ypos

```

```

timing_element = value_elem.find("./timing")
distribution_type = timing_element.attrib.get("distributionType", "")
...
failure_outcome = value_elem.find("./outcomes[@key='Failure']/value")
success_outcome = value_elem.find("./outcomes[@key='Success']/value")

failure_elem.text = failure_probability
success_elem.text = success_probability
parameter_elem = ET.SubElement(parameters, "string", id="0")
parameter_elem.text = parameter_value

```

Creation of the Output Gates

The *convert_step_element_to_outputgate* (Listing 20) function manages the creation of one or two Output Gates for each attack step, depending on whether the success and failure outcomes are defined.

The name of the two Output Gates is created by appending the right suffix to the names extracted from the ADVISE model.

Afterward, we save the x and y positions, as we did for the generation of the Input Gate and Timed Activity, taking care to change the coordinates in a way that does not overlap the components. Both Output Gates are moved down, and the "success" Output Gate is moved left while the "failure" Output Gate is moved right.

Lastly, the *transform_cpp_code()* function is used to transform C++ code for effects defined in the ADVISE model which are not compatible with SAN.

Listing 20: Python code for the creation of the Output Gates.

```

success_name = name + "_Output_Success"
failure_name = name + "_Output_Fail"

success_x_elem = ET.SubElement(success_center_point, "int", id="x")
success_x_elem.text = str(xpos - 25)
success_y_elem = ET.SubElement(success_center_point, "int", id="y")
success_y_elem.text = str(ypos + 50)
...
failure_x_elem = ET.SubElement(failure_center_point, "int", id="x")
failure_x_elem.text = str(xpos + 25)
failure_y_elem = ET.SubElement(failure_center_point, "int", id="y")
failure_y_elem.text = str(ypos + 50)

success_function_elem.text = transform_cpp_code(success_effects)
failure_function_elem.text = transform_cpp_code(failure_effects)

```

5.6 CONCLUSIONS

In this chapter, we have proposed a mode-to-model transformation from ADVISE to SAN, in order to address the challenges around the ADVISE formalism. We showed the logical mapping between the elements of the two formalisms and some excerpts of the implementation of the transformation.

In [chapter 6](#) we will apply the transformation to a case study, showing the additional behaviors that we are able to capture using the SAN formalism, like the possibility of simulating parallel attacks.

CASE STUDY: THE 2015 UKRAINE POWER OUTAGE

In this chapter, we apply the extensions of the ADVISE Meta framework illustrated in [chapter 3](#) and [chapter 4](#), and the model-to-model transformation described in [chapter 5](#), to the 2015 cyber-induced power outage in Ukraine. We use this case study to show the benefits of i) the integration of the CAPEC attacks and ii) the transformation from ADVISE models to SAN models. The first part equips the framework with a well-founded base of attacks that can be used in the models. The second part, instead, allows to capture some behaviors that are not considered by the ADVISE formalism, like simulating the parallel execution of attacks, and to possibly take into account defense mechanisms.

6.1 DESCRIPTION OF THE CYBER ATTACK

On December 23, 2015, three different Ukrainian power grids simultaneously experienced the opening and closing of circuit breakers, remotely controlled by attackers. This resulted in a power outage for many users in Ukraine.

After investigations, experts for the U.S. Department of Homeland Security produced a report outlining the attack steps that the attackers carried out [76]. Other authors have further analyzed the attack and proposed possible mitigations [77, 78].

[Figure 16](#) shows the architecture of the Ukraine power plant, highlighting the locations where the attack steps took place. The main attack steps can be enumerated as follows [77]:

1. Spear phishing emails were sent to the corporate workstations of the power plants, in order to install the Black Energy 3 (BE3) malware.
2. The successfully installed BE3 malware was used for the reconnaissance and enumeration of the corporate network.
3. The attackers found and used the corporate user accounts and credentials.
4. An encrypted tunnel from an external network to the control network was created using the credentials found, in order to establish a presence in the network.
5. The attackers accessed the SCADA HMI workstations located inside the control network. This was possible due to the misconfigured firewall between the corporate and the control networks.

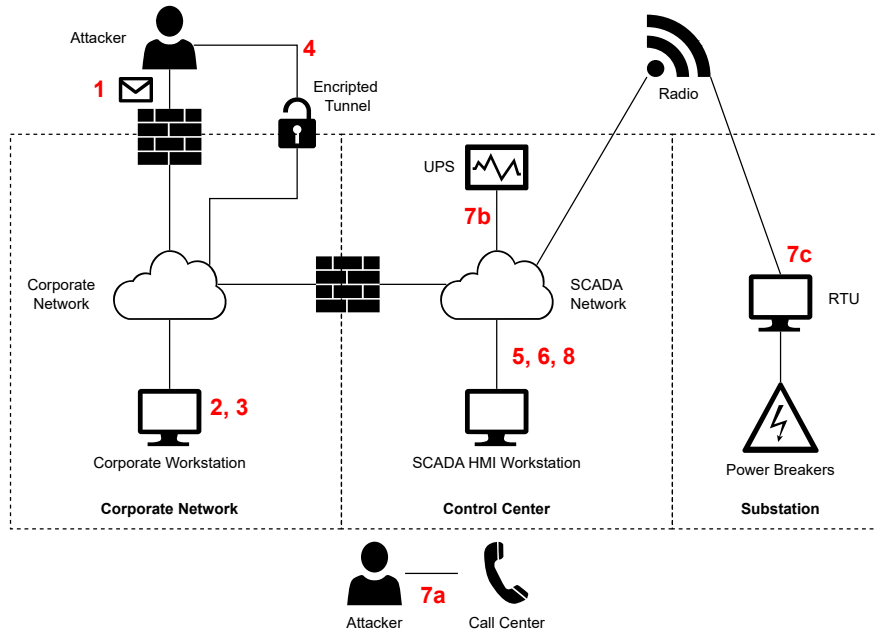


Figure 16: Simplified view of the architecture of the Ukrainian power grid control system. The attack steps sequence is reported using numbers. The figure is adapted from [77].

6. The attackers started to open and close the breakers through the SCADA HMI workstations, causing the power outage. This was the main attack carried out, which started on December 23, 2015, at 3:30 p.m.
7. While the main attack was ongoing, in order to hinder the response actions of the operators, the attackers carried out some additional malicious actions:
 - 7a. A telephony DoS attack was launched on the telephone line to inhibit communication between the operators.
 - 7b. The uninterruptible power supplies (UPSs) inside the control center were disabled.
 - 7c. The firmware of the remote terminal units (RTUs) was corrupted to prevent the operators from using the terminals.
8. The KillDisk malware was finally executed to wipe out the workstations' data.

6.2 MODELING THE UKRAINE ATTACK USING ADVISE META

In the following, we model the Ukraine power outage attack using the ADVISE Meta framework, relying on the extensions of its ontology shown in [chapter 3](#) and [chapter 4](#).

6.2.1 Selection and modification of the CAPEC attacks

As a first step, we select among the Meta CAPEC attack patterns those which can be used to represent the attack steps of the Ukraine power grid attack described in [section 6.1](#). The selected attacks are the following (the ADVISE Meta elements marked with an asterisk have been added to the ontology):

- **Attack step #: 1**

CAPEC attack: CAPEC-151: Identity Spoofing. It refers to the action of assuming (i.e., taking on) the identity of some other entity (human or non-human) and then using that identity to accomplish a goal. Spear phishing emails to the corporate workstations is a form of identity spoofing. In fact, the Identity Spoofing meta attack is a parent of the Spear Phishing detailed attack in the CAPEC hierarchy.

Additional prerequisites: CorporateWorkstation_EmailAddress* (Knowledge). The adversary must know the email address of the workstation in order to send spear phishing emails.

Effect: CorporateWorkstation_LogicalAccess (Access). After opening the phishing email, the BE3 malware is executed and takes control of the workstation (logical access).

System component type: Workstation

- **Attack step #: 2**

CAPEC attack: CAPEC-169: Footprinting. Footprinting is a general term to describe a variety of information-gathering techniques, often used by attackers in preparation for some attack. These actions can be compared to the reconnaissance and enumeration of the corporate network.

Additional prerequisites: CorporateWorkstation_LogicalAccess (Access). Access to the workstation is necessary as a starting point for network reconnaissance and enumeration.

Effect: CorporateNetwork_NetworkAccess (Access). After the attack, the adversary gets access to the corporate network.

System component type: WiredNetwork

- **Attack step #: 3**

CAPEC attack: CAPEC-560: Use of Known Domain Credentials. With this attack an adversary guesses or obtains (i.e. steals or purchases) legitimate credentials (e.g. userID/password) to achieve authentication and to perform authorized actions under the guise of an authenticated user or service. This resembles the actions performed by the adversaries to obtain the corporate users' credentials.

Additional prerequisites: CorporateWorkstation_LogicalAccess (Access). The adversary must possess access to the nodes where the credentials are stored.

Effect: CorporateNetwork_Credentials* (Knowledge). After this attack, the credentials of the corporate network are gained.

System component type: WiredNetwork

- **Attack step #:** 4-5

CAPEC attack: CAPEC-554: Functionality Bypass. With this CAPEC attack an adversary attacks a system by bypassing some or all functionality intended to protect it. The exploitation of the misconfigured firewall between the corporate and the control networks can be seen as a form of functionality bypass.

Additional prerequisites: CorporateNetwork_NetworkAccess (Access) and CorporateNetwork_Credentials* (Knowledge). To bypass the firewall rules, access to the corporate network and corporate network credentials are needed.

Effect: SCADANetwork_NetworkAccess (Access) And MainAttackOngoing* (Global Knowledge). After bypassing the firewall, access to the SCADA network is obtained and the main attack (the power outage attack) is ready to be started.

System component type: Firewall

- **Attack step #:** 6

CAPEC attack: CAPEC-113: Interface Manipulation. It refers to an adversary that manipulates the use or processing of an interface (e.g. Application Programming Interface (API) or System-on-Chip (SoC)) resulting in an adverse impact upon the security of the system implementing the interface. We think this attack suits the actions performed by the adversary in taking control of the SCADA HMI workstations to open and close the breakers.

Additional prerequisites: SCADANetwork_NetworkAccess (Access) and MainAttackOngoing* (Global Knowledge). This attack can be performed when access to the SCADA network is possessed and when the main attack is ongoing.

Effect: PowerOutage* (Global Access). With this attack, the outage of the power grid is achieved.

System component type: Workstation

- **Attack step #:** 7a

CAPEC attack: CAPEC-125: Flooding. With this attack, an adversary consumes the resources of a target by rapidly engaging in a large number of interactions with the target. This corresponds to the objective of the DoS attack on the telephone line.

Additional prerequisites: MainAttackOngoing* (Global Knowledge). This attack is performed as an additional malicious action when the main attack is ongoing.

Effect: TelephoneLine_UnableToService (Access). This attack saturates the telephone line.

System component type: TelephoneLine (Parent Component: Network)

- **Attack step #:** 7b

CAPEC attack: CAPEC-607: Obstruction. In this CAPEC attack an adversary obstructs the interactions between system components. By interrupting or disabling these interactions, an adversary can often force the system into a degraded state or cause the system to stop working as intended. This can cause the system components to be unavailable until the obstruction is mitigated. The disruption of the UPS can be seen as a form of obstruction.

Additional prerequisites: SCADANetwork_NetworkAccess and MainAttackOngoing* (Global Knowledge). This attack is performed as an additional malicious action when the main attack is ongoing. Access to the SCADA network is needed, as the UPS is connected to it.

Effect: UPS_UnableToService* (Access). With this attack the UPS is disabled.

System component type: UPS (Parent Component: Device)

- **Attack step #:** 7c

CAPEC attack: CAPEC-184: Software Integrity Attack. With this attack an adversary initiates a series of events designed to cause a user, program, server, or device to perform actions which undermine the integrity of software code, device data structures, or device firmware, achieving the modification of the target's integrity to achieve an insecure state. The corruption of the RTU firmware falls under this attack.

Additional prerequisites: SCADANetwork_NetworkAccess And MainAttackOngoing* (Global Knowledge). This attack is performed as an additional malicious action when the main attack is ongoing. Access to the SCADA network is needed to reach the RTU through the radio connection.

Effect: RTU_UnableToService (Access). This attack corrupts the RTU firmware, disabling it.

System component type: RTU (Parent Component: Device)

- **Attack step #:** 8

CAPEC attack: CAPEC-549: Local Execution of Code. It refers to an adversary that installs and executes malicious code on the target system in an effort to achieve a negative technical impact. Examples include rootkits, ransomware, spyware, adware, and others. The erasing of the workstation's data was achieved with the KillDisk malware.

Additional prerequisites: PowerOutage* (Global Access). This attack is attempted only after the main attack, i.e., the power outage, has been successfully performed.

Effect: SCADAWorkstation_DiskWiped* (Access). The attack erases the data contained in the SCADA workstations.

System component type: Workstation

The *attack step #* is the number of the attack step according to the description in [section 6.1](#). The *additional prerequisites* are system-dependent prerequisites in terms of ADVISE elements needed to attempt the attack, in addition to the prerequisites in terms of TAL attributes added into the Precondition Expression of each attack through the semi-automated integration of the CAPEC attacks ([subsection 4.3.2](#)). These prerequisites are dependent on the specific case study and have to be added to the ontology. The *effect* is the ADVISE element that is obtained after successfully completing the attack. During the semi-automated integration of the CAPEC attacks, the effect was left empty ([subsubsection 4.3.3](#)), therefore has to be specified here. The *system component type* is the component that is associated with the attack. In the semi-automated integration of the CAPEC attack, we proposed a way to automatically assign the CAPEC attacks to the ADVISE Meta architectural components ([subsection 4.3.4](#)). Here, instead, the assignment is made manually to better suit the case study. Moreover, additional components are added to the ontology, i.e., TelephoneLine, UPS, and RTU.

As mentioned before, in order to effectively use the CAPEC attacks integrated into the ontology of the framework in the context of the case study, we have to modify them. The main modifications involve the Precondition Expression, the Attack Execution Time and the Effect of the attack steps, and the Dependants attribute of the architectural components. We performed these modifications on a copy of the ontology obtained after the semi-automated integration of the CAPEC attacks through ChatGPT.

Let us consider, as an example, the CAPEC-113 Interface Manipulation. The attack step, as a result of the semi-automated integration into the ADVISE Meta ontology using ChatGPT, has the following characteristics:

- **Name:** InterfaceManipulation;
- **Id:** 113;
- **Attack Cost:** return 400;
- **Attack Execution Time:** return 0;
- **Precondition Expression:** return ((\$theResources>=400) && (\$theLimits>=750) && (\$theSkills>=750) && (\$Intent>=1) && (\$theVisibility>=250) && (\$theAccess>=0));
- **Success:**
 - **Probability Expression:** return 0.5;
 - **Effect:** \$ACCESS_113 = 1;
- **Failure:**
 - **Probability Expression:** return 0.5;
 - **Effect:** N/A;

The derived Precondition Expression just takes into account the TAL attributes, while the Attack Execution Time and the Effect for the Success case are used as

placeholders. Moreover, the InterfaceManipulation attack was assigned by ChatGPT to the Software component.

To instantiate the attack in the context of the Ukraine power outage we make the following modifications:

- We change the assignment of the attack to the Workstation component type (for us the involved component will be the SCADA workstation) instead of Software.
- For the Precondition Expression we add the following prerequisites: the adversary must possess the Access to the network where the workstation is located (in our case, the access to the SCADA network) and the main attack, i.e., that started on December 23, 2015, has to be ongoing (*MainAttackOngoing* Knowledge is possessed).
- The work in [78] provides some indications of the time windows in which the different attacks may have occurred, but the actual duration is not known. Therefore, we assume that they all have the same duration, i.e., we assign 1 (unit of time) to the Attack Execution Time (with a deterministic time distribution). In this way, in the analysis, we focus on the probability of attack rather than the time. The same is done for all the other attacks.
- We instantiate the $\$ACCESS_113$ variable in the Effect, by assigning to it the global Access variable *PowerOutage*, which indicates that the power outage has been accomplished.

Figure 17 shows the details of the InterfaceManipulation visualized using the ADVISE Meta framework after making the abovementioned modifications.

The screenshot displays the ADVISE Meta framework configuration for the `InterfaceManipulation` attack. On the left, a tree view shows the attack's structure, including preconditions like `ACCESS_113`, `theLimits`, `theResources`, `Intent`, `theVisibility`, `theSkills`, `theAccess`, `MainAttackOngoing`, `NetworkAccess`, and `onNetwork`. The central panel shows the attack's configuration:

- Name:** `InterfaceManipulation`
- Global:**
- Cost Expression:** `return 400;`
- Attack Execution Time:** `Deterministic` distribution with a value of `return 1;`
- Precondition Expression:**

```
bool atLeastOnePossessed=false;
for (auto const &n : ${NetworkAccess})
{
    if(n->Mark())
        atLeastOnePossessed=true;
}
return (($MainAttackOngoing)&&atLeastOnePossessed
&&(${theResources}>=400)&&(${theLimits}>=750)&&(${theSkills}>=750)
&&(${Intent}>=1)&&(${theVisibility}>=250)&&(${theAccess}>=0));
```

The right-hand panel shows the configuration for the `Success` and `Failure` outcomes:

- Success:**
 - Outcome Probability:** `return 0.5;`
 - Detection Probability:** `return 0.3;`
 - Effects:** ``${ACCESS_113}=1;`
- Failure:**
 - Outcome Probability:** `return 0.5;`
 - Detection Probability:** `return 0.7;`
 - Effects:** (empty)

Figure 17: The InterfaceManipulation attack step details as visualized into the ADVISE Meta framework.

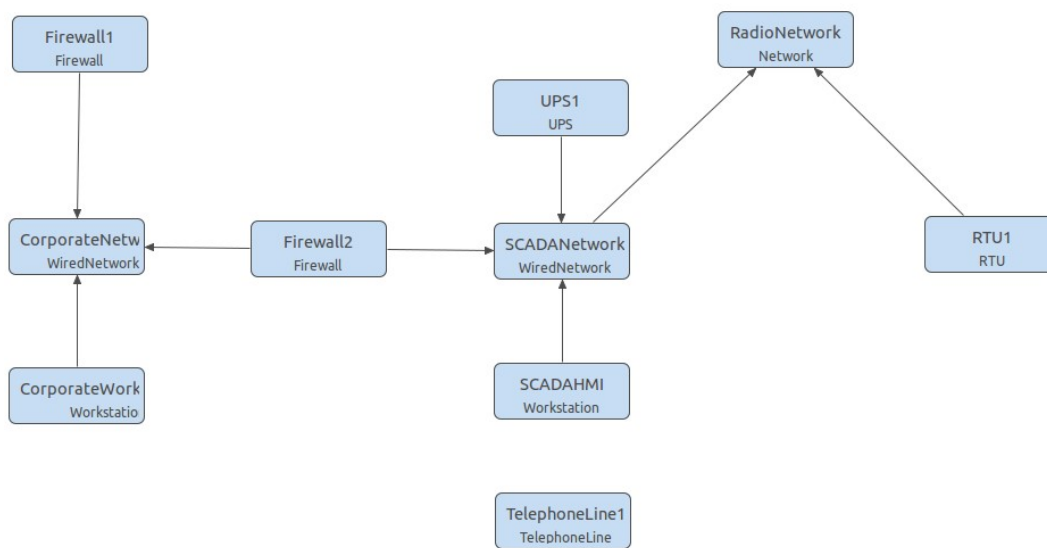


Figure 18: System Instance Diagram (SID) of the Ukraine power grid control system.

6.2.2 Models definition

In order to generate the ADVISE models, we have first to create the System Instance Diagram (SID) model using the ADVISE Meta framework and the architectural components defined in its extended ontology, including the newly added components, i.e., TelephoneLine, UPS, and RTU. Figure 18 shows the SID created reflecting the architecture of the Ukraine power grid control system described in section 6.1.

Among all the TAL adversaries' profiles integrated into the extended ontology, the one chosen for this case study is the Government Cyberwarrior. In fact, the attack on the Ukraine power grid has been attributed by experts to the advanced persistent threat team known as Sandworm [79].

In order to have a comprehensive view of the attack, we define a goal for each element that is obtained as an effect of successfully completing each attack step identified in subsection 6.2.1:

- *CorporateWorkstationLogicalAccess* as result of Identity Spoofing;
- *CorporateNetworkNetworkAccess* as result of Footprinting;
- *CorporateNetworkCredentials* as result of Use of Known Domain Credentials;
- *SCADANetworkNetworkAccess* as result of Functionality Bypass;
- *GridPowerOutage* as result of Interface Manipulation;
- *TelephoneLine1UnableToService* as result of Flooding;
- *UPS1UnableToService* as result of Obstruction;
- *RTU1UnableToService* as result of Software Integrity Attack;

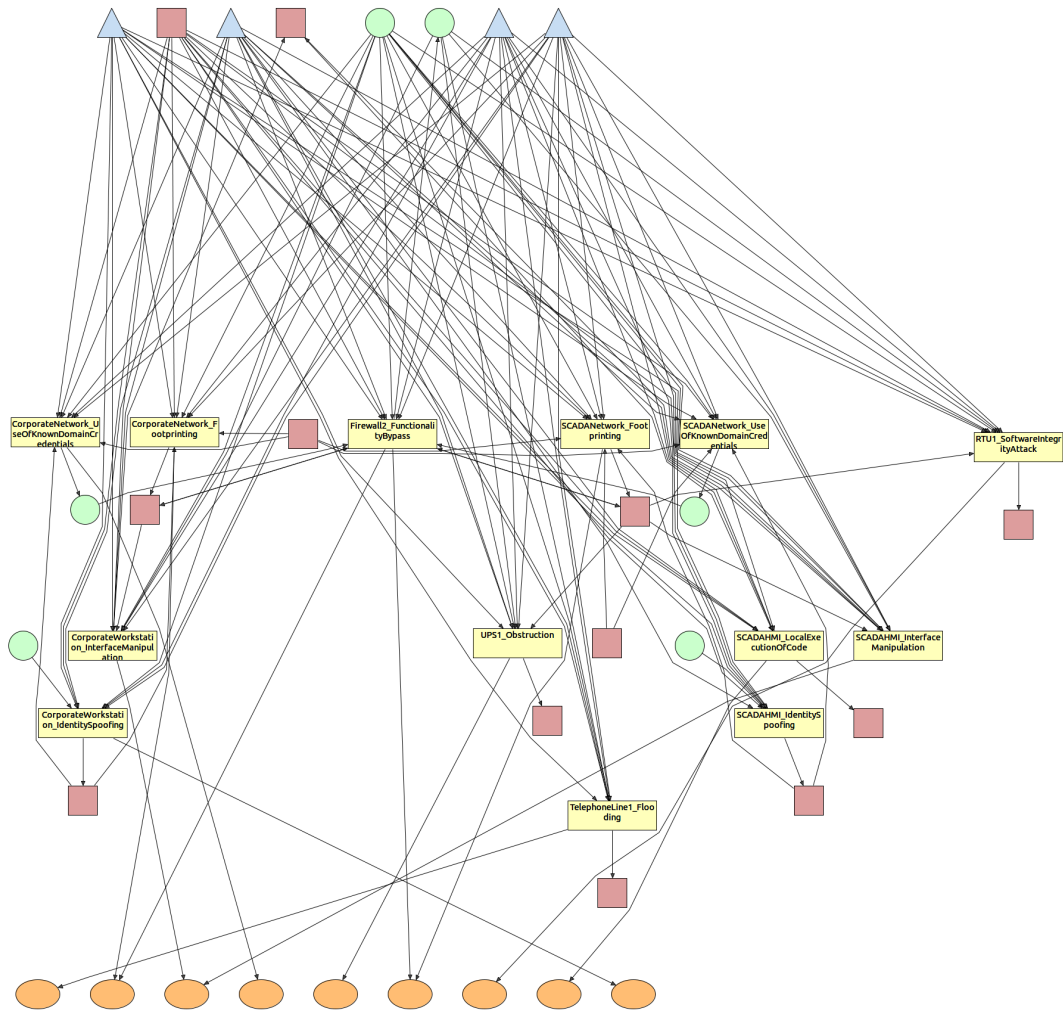


Figure 19: Generated ADVISE model for the Ukraine attack.

- *SCADAHMI_DiskWiped* as result of Local Execution of Code.

After assigning the goals to the adversary profile, we can generate the ADVISE model. Figure 19 shows the resulting ADVISE model for the Ukraine power outage attack.

6.3 TRANSFORMATION OF THE GENERATED ADVISE MODEL TO SAN

We now transform the ADVISE model generated by ADVISE Meta into a SAN model using the model-to-model transformation illustrated in chapter 5.

Figure 20 shows the transformed SAN model. Despite its complexity, it is possible to notice that the structure of the ADVISE model is preserved by the transformation.

Currently, the transformation process does not include the derivation of the reward variables, the studies, and the simulator generated by ADVISE Meta. For this reason, we manually re-created those elements inside the Möbius project, to

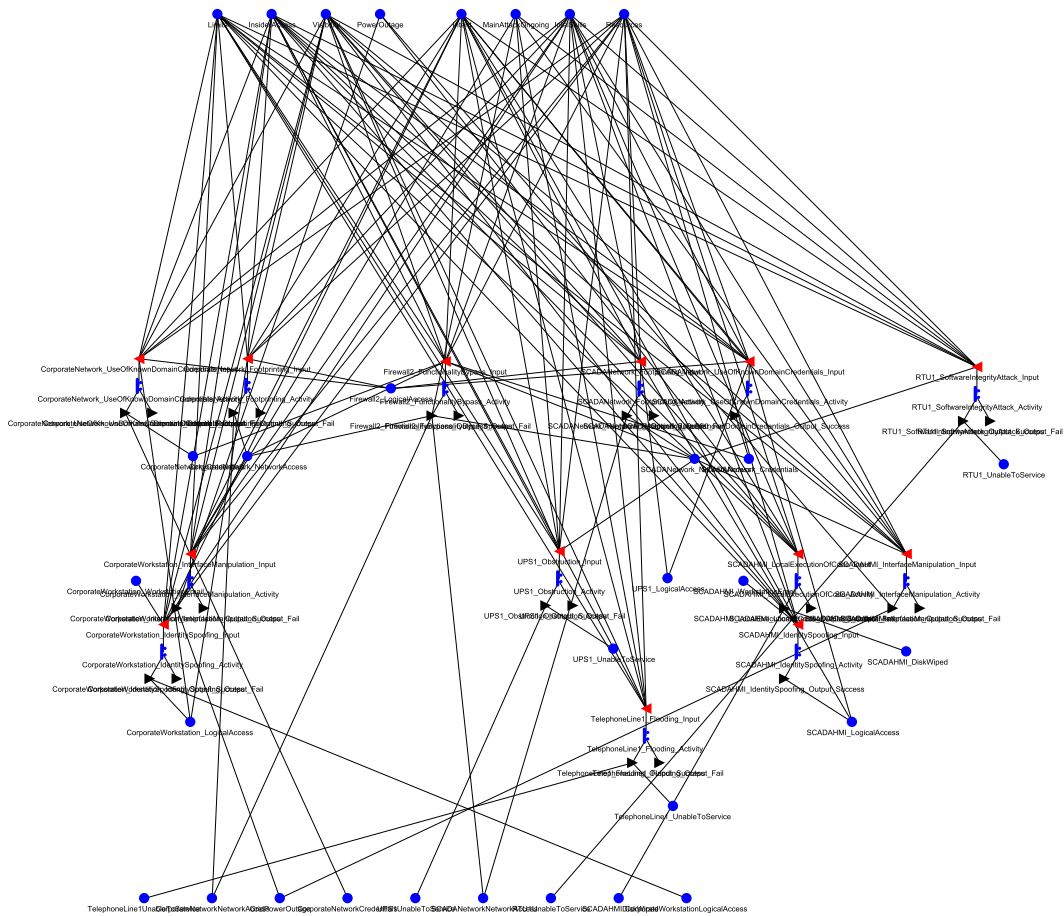


Figure 20: Transformed SAN model for the Ukraine attack.

use them with the SAN model. As an example, the rate reward for the GridPower-
Outage goal is the following:

```
return AttackModel->GridPowerOutage->Mark();
```

6.3.1 Comparative analysis

In the following, we compare the results produced by simulating the two ADVISE and SAN models. In both cases, we simulated the behavior of the TAL Government Cyberwarrior adversary profile in reaching the defined goals. We ran the simulations with, respectively, 1000 and 10000 minimum and maximum number of batches (simulation runs), measuring the first ten time units (hours) through an instant of time reward variable converging within 95% probability in a 0.1 relative interval.

Figure 21 shows the adversary probability of achieving the defined goals, for both ADVISE and SAN models. For ADVISE, we can observe that the main outage attack (*GridPowerOutage*) starts after the adversary has obtained access to the SCADA Network. Here the adversary prioritizes the additional malicious actions, i.e., the telephone line DoS, and the disruption of the UPS and RTU, before

attempting the main power outage attack. Remember that the adversary execution algorithm embedded in ADVISE does not allow parallel exploitation of the attacks.

Regarding SAN, instead, we can see that some of the attacks are executed in parallel, e.g., the attacks on the *CorporateNetworkAccess* and *CorporateNetworkCredentials*. Also, the additional malicious actions are executed in parallel with the *GridPowerOutage* attack. This demonstrates that the SAN formalism allows to get

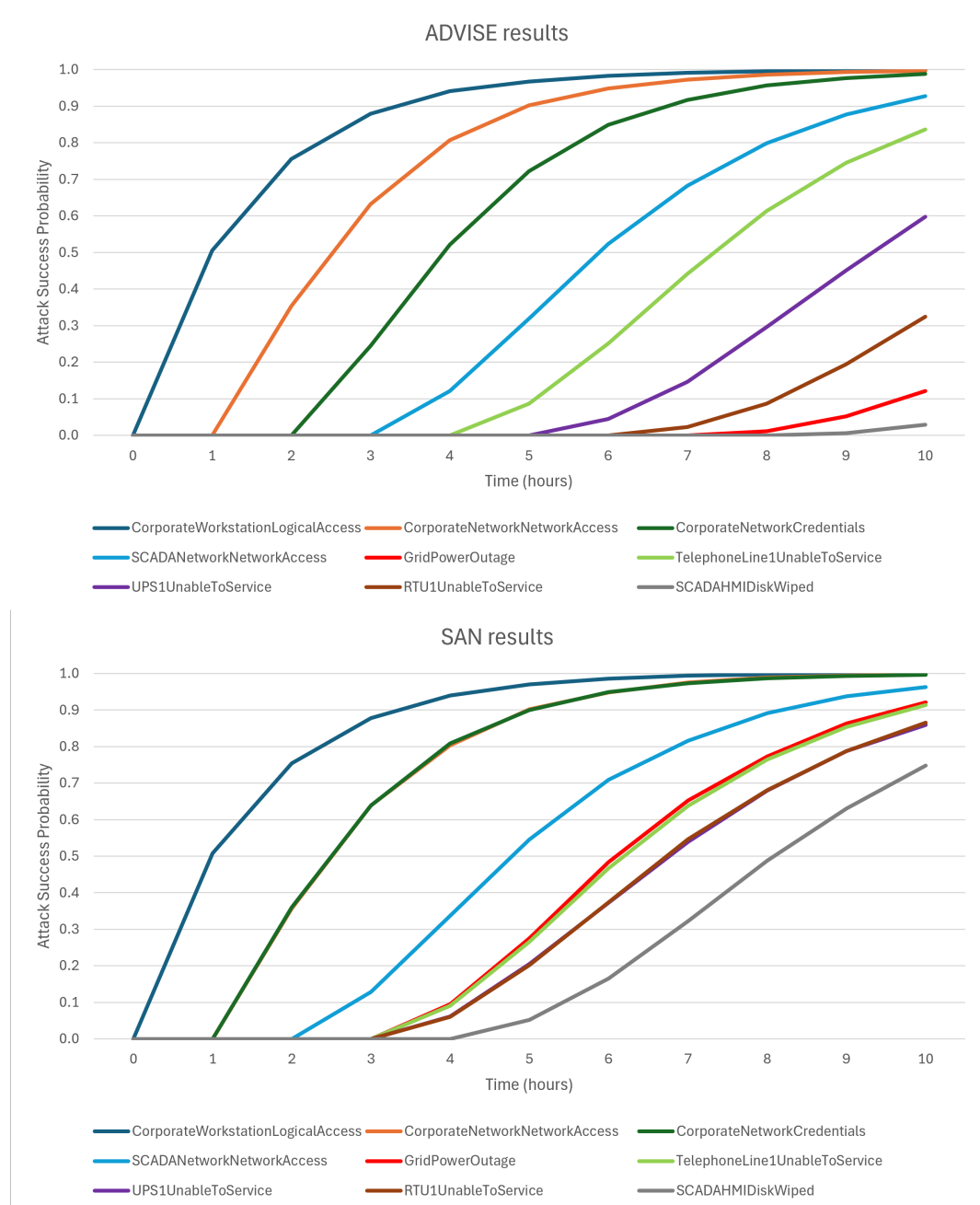


Figure 21: Probability of successfully reaching the adversaries' goals for the Ukraine attack, simulated in ADVISE and SAN.

released from some of the assumptions made by the ADVISE execution algorithm, like the impossibility of performing attacks in parallel.

Despite here we are showing all the goals that are present in the attack model, we have to point out that they have different purposes. The first five goals (from *CorporateWorkstationLogicalAccess* to *GridPowerOutage*) are part of the main attack whose final aim is the outage of the power grid. The remaining goals (*TelephoneLine1UnableToService*, *UPS1UnableToService*, *RTU1UnableToService* and *SCADAHMIDiskWiped*) are made by the attacker to hinder the response to the attack and the restore process. Concerning these goals, according to the simulation, we can state that the disruption of the telephone line is the most probable attack, while the wiping of the workstation's disk is the least probable, as the power outage attack has to be successfully completed in order to perform this attack.

Table 6 reports the comparison of the simulation results of ADVISE and SAN models concerning the second modeled attack step of the Ukraine power grid attack, which is Footprinting. It is possible to notice that the results of both ADVISE and SAN overlap and remain within the confidence intervals. This demonstrates that for attack steps that are executed in sequence, the behavior captured by SAN models is the same as in ADVISE.

Table 6: Comparison of the simulation results of ADVISE and SAN models for the Footprinting attack step (*CorporateNetworkNetworkAccess* goal).

Time	ADVISE	ADVISE Confidence Interval	SAN	SAN Confidence Interval
0	0	0	0	0
1	0	0	0	0
2	0.3536	0.009370971	0.357	0.010499722
3	0.6322	0.009451721	0.63925	0.010523901
4	0.8067	0.007740157	0.8035	0.00870788
5	0.9028	0.005806397	0.9015	0.006530391
6	0.9488	0.004320163	0.947875	0.004871209
7	0.9724	0.003211107	0.975125	0.003413107
8	0.9861	0.002294804	0.988875	0.002298576
9	0.9931	0.001622553	0.994	0.001692416
10	0.9962	0.001205988	0.99775	0.001038342

6.3.2 Defense mechanisms

In the following, we provide a preliminary discussion and possible future research directions around the development of models representing countermeasures and defense mechanisms. Thanks to the flexibility of SAN formalism, we can potentially represent the details of even complex defense mechanisms. However, for the moment, the level of abstraction of the architectural components in the ADVISE Meta ontology and the Meta CAPEC attacks that we have integrated is high. For example, with the analysis, we can observe which is the probability of successfully executing the KillDisk malware (CAPEC-549: Local Execution of Code), but we are

not representing the details of how the attack is performed. Hence, if one wants to consider very specific and detailed countermeasures and defense mechanisms, the system components and the attacks should be adapted and specialized to be at the same abstraction level.

Parameterized countermeasures

The studies around the Ukraine power outage attack provide detailed discussions on possible countermeasures and defenses that should have been implemented in the system to thwart the malicious actions carried out by the adversaries [77, 78]. Just as an example, for preventing and mitigating credentials theft, the usage of IDSs, password rotation, strong firewall rules, and antiviruses is recommended [77].

The presence of such defense techniques could be modeled inside the SAN through parameters that could impact the attack steps, like their execution time or success probability. For example, if we consider as a countermeasure the adoption of strong rules for the firewall, then we can decrease the success probability for the Functionality Bypass, that is the CAPEC attack that in our case study models the bypassing of the firewall rules. In such a way, the techniques would be modeled in a "passive" manner, according to different levels of implemented countermeasures, without actually modeling a state-based defense.

Different architectural solutions and redundancy

With the flexibility of ADVISE Meta and our improvements to the modeling approach, we can evaluate and compare different architectural solutions. A possible defense strategy could consist of the adoption of redundant components.

Let's consider as an example the UPS placed in the Control Center (see [Figure 16](#)). If this is deactivated by an adversary, then the Control Center will be exposed to power failures. Instead, if more redundant UPSs are placed, the adversary has to disable all of them in order to accomplish its goal (assuming that these disruption events are independent). In order to analyse such scenarios we i) place additional UPS components in the SID (see [Figure 22](#)), ii) derive the ADVISE model, iii) perform the model-to-model transformation to SAN.

Using the derived SAN models, [Figure 23](#) shows the probability for the Government Cyberwarrior adversary in successfully disabling the UPS service. Three different configurations are shown: one UPS (no redundancy), two redundant UPSs, and three redundant UPSs. It is possible to observe that the probability of successfully disabling the service decreases as the number of redundant UPSs increases. Moreover, we can state that the time needed for the adversary to disable all the UPSs increases as the number of UPSs increases. This means that also the time for detecting the attack and responding to it increases when more UPSs are used.

Dynamic defense strategies

In the literature exists a class of defense techniques known as Moving Target Defenses (MTD) [80]. These aim at breaking the static nature of systems' configu-

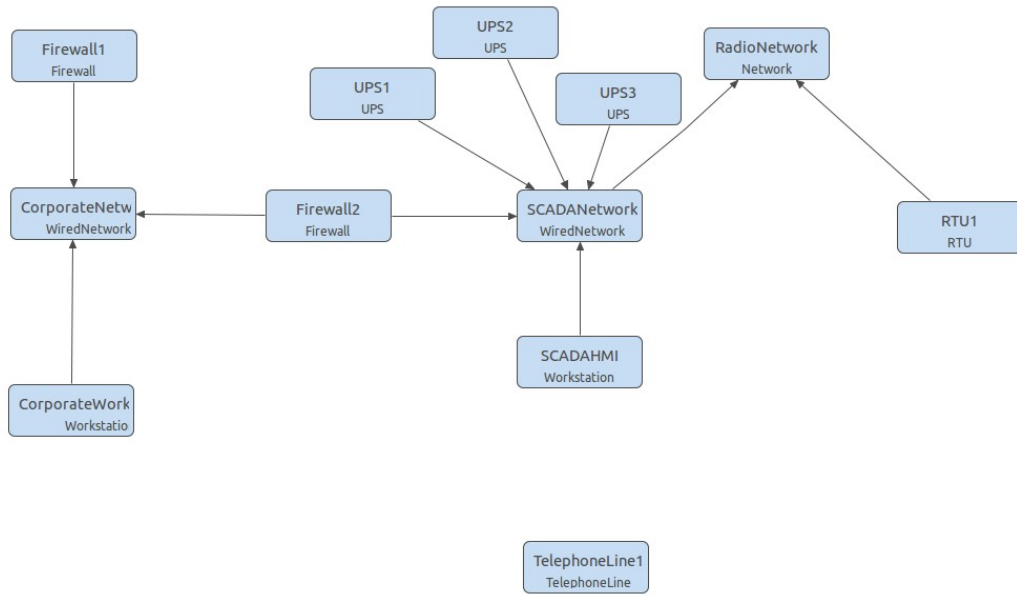


Figure 22: SID of the Ukraine power grid with three redundant UPSs.

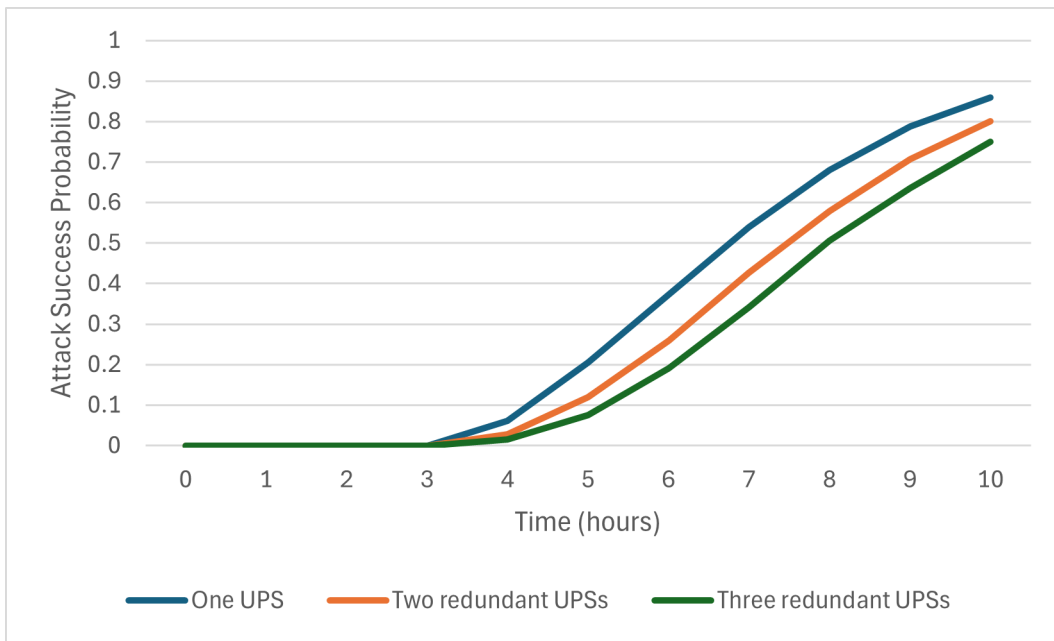


Figure 23: SAN results for the probability of successfully disabling the UPS service: configuration with one, two, and three UPSs.

rations, which often advantage the adversary in exploitation. MTDs often consist of state-based strategies involving diversity, randomness, multi-candidate, limited timeliness, and attack surface reduction.

In a preliminary work [81], we considered one specific MTD strategy, called Proactive Obfuscation [82]. This is a technique that protects and diversifies the executable code of some redundant execution units (e.g., servers), called replicas, using an obfuscator. The obfuscator uses program transformations to automatically

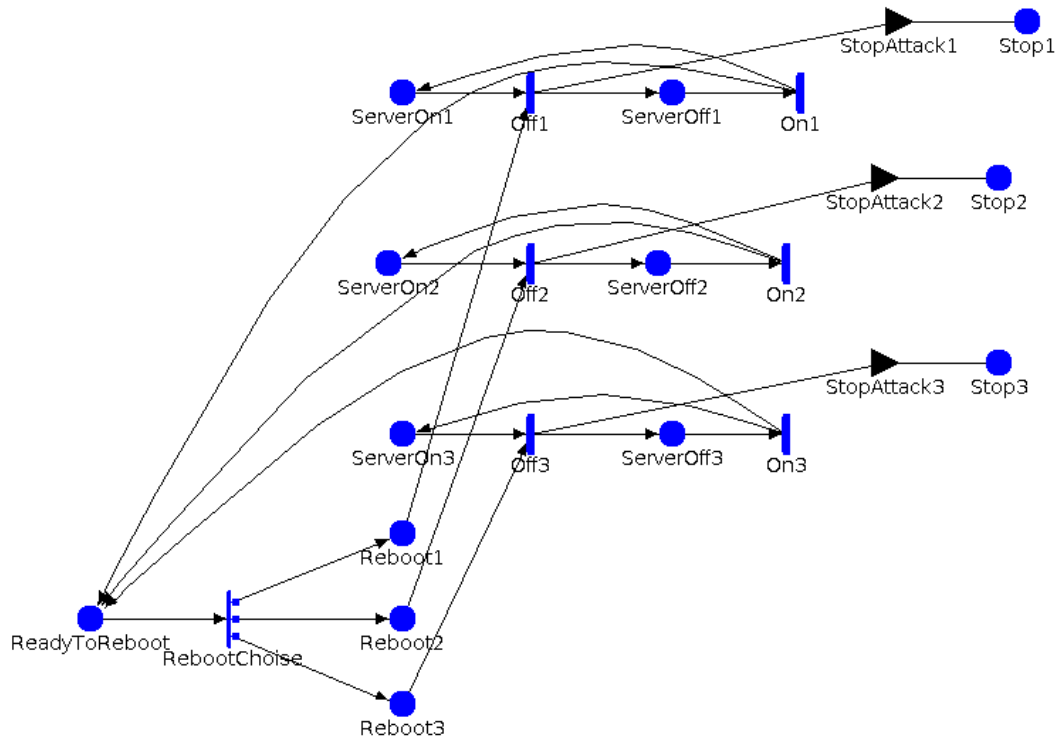


Figure 24: SAN model of a generic system composed of three redundant servers, using the proactive obfuscation technique. Figure extracted from [81].

create diverse executable code on each replica, which is structurally different from the others, but with the same semantics. Periodic restarts are also applied to the replicas. After a restart, a replica is reset (totally or partially) using the obfuscator, nullifying the progress made by the adversary in trying to exploit the executable code. Hence, in order to be successful, the attack carried on by the adversary should be completed within the time window used to restart the replica.

In that work, we modeled the Proactive Obfuscation technique using SAN and composed together with a simple and generic ADVISE model. Figure 24 illustrates the SAN model of the proactive obfuscation technique applied to a generic system utilizing three redundant servers. These servers can be either in the "on" state (*ServerOn*) or "off" state (*ServerOff*). Initially, all servers are active, and the marking of *ReadyToReboot* is set to 1. The server to undergo reboot is randomly selected by the instantaneous activity *RebootChoice*, with equal probability assigned to each case. The selected server is then shut down through the timed activity *Off*, with a deterministic rate. Once the server is off, it is assumed to have completed its service and initiated its obfuscation process. At this point, any attacks targeting that specific server are interrupted, indicated by the marking of *Stop* being set to 1.

Such kinds of MTD models could be composed together with the transformed SAN model representing the attack path, in order to analyse the effectiveness of state-based defenses.

6.4 CONCLUSIONS

In this chapter, we have illustrated an application of the modeling approach to the 2015 Ukraine power outage attack. We have shown that we can rely on CAPEC attacks integrated into the ADVISE Meta ontology using ChatGPT to model the steps of the cyber attack. However, some manual modifications to the ontology are needed in order to instantiate the attacks to the case study, since additional information related to the Precondition Expression, Attack Execution Time, and Effect is dependent on the specific case study and is not known at the moment of the semi-automated integration.

With the transformation from ADVISE to SAN formalism, we have shown that SAN is capable of capturing the same behavior as ADVISE in case of sequential attacks. Moreover, SAN can simulate additional scenarios, like attacks executed in parallel.

We have briefly discussed possible representations of defense mechanisms, like parameterization of countermeasures levels, components redundancy, and dynamic strategies. SAN formalism is flexible enough to allow the representation of even complex solutions. We have listed possible research directions around these topics, but the actual implementation of such models is delegated to future works.

CONCLUSIONS AND FUTURE WORK

In this chapter, we summarize the thesis contributions, discuss the main open challenges, and provide some research directions for future work.

7.1 SUMMARY OF THE CONTRIBUTIONS

Due to the increasing degree of connectivity and complexity of Cyber-Physical Systems, cybersecurity is becoming a concern even for this kind of system. Model-based evaluation can play a fundamental role in assessing cybersecurity from the very initial stages of system development, providing benefits like complexity management and reusability.

In this thesis, we have proposed some advancements around some aspects that, from the industry viewpoint, are crucial for cybersecurity assessment which are i) an approach to represent and compare different system components and architectures, ii) a well-founded and reasonably large catalog of attacks, iii) a standardized collection of adversaries, and iv) a way to represent and analyse countermeasures and defense mechanisms, and to capture possible interdependencies between safety and security.

ADVISE Meta is an existing ontology-based security modeling framework that allows to generate attack models starting from an architectural representation of a system. This framework poses some challenges like the limited number of adversaries and attacks available in its ontology, the assumptions made by the adversary's execution algorithm, and the fact that only the adversary viewpoint is considered.

First, we have illustrated a methodology for the extension of the ADVISE Meta ontology in order to include the attack patterns from the CAPEC catalog and the adversaries from TAL. This extension equips the framework with a well-founded collection of attacks and adversaries, which can be used to perform broad-range security analysis, as demonstrated by the application to a case study of a public transport supervision system.

The aforementioned extension, encompassing all the TAL adversaries and a few representative CAPEC attacks, has been performed manually. If we consider the vastness of the CAPEC catalog, its manual integration into the ADVISE Meta ontology is a time-consuming process. To overcome this issue, we have proposed a methodology for the semi-automated integration of the CAPEC attacks into the ADVISE Meta ontology with the support of AI. We have used ChatGPT to interpret the textual content of the CAPEC sections, create new attack steps, and insert them

in the ontology of the framework, exploiting the XML representations of both the CAPEC database and ADVISE Meta ontology.

To overcome the fact that only the adversary viewpoint is taken into account by the ADVISE formalism and to address the challenges related to the ADVISE execution algorithm, we decided to rely on a more flexible modeling formalism, in particular SAN. Therefore, a model-to-model transformation from ADVISE to SAN models has been illustrated. We have first identified the mapping between the various elements of the two formalisms, also taking into account their XML representation in Möbius. We have implemented such a transformation using the `lxml` Python library for XML parsing. Through an application to a case study in the power grid domain, we have shown that with SAN we can capture the same behavior of ADVISE models in case of sequential attacks, but also to simulate different scenarios, like the parallel execution of attacks.

With the transformation to SAN, we facilitate the composition of the attack model with other models capturing different aspects. We have already briefly discussed possible representations of countermeasures and defense mechanisms, like redundant components or dynamic strategies, e.g., Moving Target Defenses.

7.2 OPEN CHALLENGES

As discussed in this thesis, the proposed modeling approach brings several benefits to cybersecurity assessment. However, there are still some open challenges to be addressed.

First, we have to note that, even if we are using quantitative tools, we mainly use them to provide qualitative observations, e.g., for identifying the most critical attack paths, or the most dangerous attacks and/or adversaries. In the approach, there are several assumptions around quantitative parameters, e.g., success/failure probability of attacks and time distributions. For example, the attack success probability is based on the likelihood qualitative levels provided by CAPEC. This is done because in cybersecurity it is extremely difficult to provide precise values for probabilities and times since they depend on a lot of aspects that can differ from one situation to another.

Regarding the semi-automated integration of CAPEC attacks into the ADVISE Meta ontology, the validation of the results is an important part. In the proposed approach we have already identified and discussed situations where it is easily understandable that the AI is committing mistakes, e.g., wrong formatted XML outputs. However, we have not yet validated all the actual interpretations of the CAPEC sections provided by ChatGPT. This is a very delicate part since this interpretation is an activity that is possibly subject to different outcomes, even if it were carried out by humans. For the moment, we have just verified that the interpretations of the subset of attacks used for the case study are reasonable.

We have briefly discussed possible implementations of models representing countermeasures and defense mechanisms. Such proposals took into account the high abstraction level related to the architectural components present in the ADVISE Meta ontology and the Meta CAPEC attacks that we have integrated. Hence, if one wants to consider more specific and detailed countermeasures and

defense mechanisms, the system components and the attacks should be adapted and specialized to be at the same abstraction level.

7.3 FUTURE RESEARCH DIRECTIONS

Concerning the semi-automated integration of CAPEC attacks, we should identify a way to validate the interpretation of the textual content of the CAPEC sections. One could evaluate if the interpretation given by the AI is comparable to the interpretation that a human could reasonably give. Hence, we should perform some tests with security experts and, by sampling some of the interpretations given by ChatGPT, check if these interpretations reasonably match those provided by the experts. The usage of advanced LLMs like OpenAI o3-mini [83] or DeepSeek-R1 [84], which have some reasoning capabilities, could help in understanding which are the decisions made by the AI that lead to a specific result, and therefore to evaluate it. Then, another future contribution could be to directly train open-source LLM models for the interpretation and integration of attack databases, e.g., CAPEC but also other catalogs like MITRE ATT&CK.

The present and future of AI are looking to dedicated solutions to specific problems and fields, providing an alternative to generic LLMs. For our context, the rapid convergence between ontology and knowledge graph engineering seems promising. A primary benefit is the availability of dedicated platforms, e.g., Graphwise [85], thus solving many pure technical and manual steps. The direct integration between ontologies and AI may help to lighten the transformation process presented in this thesis.

Future works also concern further research around defense mechanisms and countermeasures, the actual development of such models, and their integration with the models automatically generated with the approach presented in this thesis, in order to understand which are the best defenses to hinder the adversaries' actions and to evaluate their effectiveness.

An aspect that we have not directly faced in this thesis is the possible consequences of the attacks on the functions provided by the analysed system. For example, considering the smart grid domain, the work in [86] analyses the consequences of DoS attacks on telecommunication services occurring while the electrical grid is already facing an electrical failure. In the future, additional models capturing the behavior of the system could be developed, e.g., taking inspiration from [87] that considers relationships between the electric infrastructure and its control system. Then, such models could be combined with the attack model in order to evaluate possible interdependencies between safety and security.

Another important aspect is the compatibility with emerging security standards like ISA/IEC 62443, CLC/TS 50701, and ISO/SAE 21434. In a recent work [88], we have proposed a cybersecurity risk assessment methodology for IACS in compliance with the ISA/IEC 62443. In such work, the criticalities of the system's assets were identified through threat categories, and qualitative levels for severity and likelihood of exploitation were assigned to each asset. Attack paths were enumerated by considering all the logical and physical connections between the different assets, without considering the specific types of attack. Future develop-

ments could aim to instantiate the modeling approach proposed in this thesis into the cybersecurity design and assessment process indicated by such standards.

The work in [89] proposes a model-based system engineering (MBSE) solution for the preliminary risk and mitigation assessment in CPS. The framework is based on TOGAF Archimate modeling language [90], allowing to describe at a general level the components of the system, their types, and the relations between them. Error propagation analysis (EPA) is adapted in the context of cybersecurity to evaluate the propagation of attacks among the system components [91]. The proposed approach includes the identification of threat scenarios (involved components, threat actors, potential loss), the quantification of risk, the identification of mitigations, and a cost-benefit estimation and optimization. Future works could explore the possibility of joining this approach with the one proposed in this thesis, to build an environment for the early-stage cybersecurity risk assessment of CPSs, aiming at compliance with the main standards in the field.

BIBLIOGRAPHY

- [1] Marzieh Kordi, Francesco Mariotti, Paolo Lollini, and Andrea Bondavalli. Security Modeling Challenges and Research Directions Around the ADVISE Meta Framework. In Andrea Ceccarelli, Mario Trapp, Andrea Bondavalli, Erwin Schoitsch, Barbara Gallina, and Friedemann Bitsch, editors, *Computer Safety, Reliability, and Security. SAFECOMP 2024 Workshops*, pages 275–283, Cham, 2024. Springer Nature Switzerland. ISBN 978-3-031-68738-9.
- [2] Andrea Bondavalli, Sara Bouchenak, and Hermann Kopetz. *Cyber-Physical Systems of Systems: Foundations - A Conceptual Model and Some Derivations: The AMADEOS Legacy*, volume 10099. Springer Cham, 12 2016. ISBN 978-3-319-47589-9. doi: 10.1007/978-3-319-47590-5.
- [3] Sascha Julian Oks, Max Jalowski, Michael Lechner, Stefan Mirschberger, Marion Merklein, Birgit Vogel-Heuser, and Kathrin Moeslein. Cyber-Physical Systems in the Context of Industry 4.0: A Review, Categorization and Outlook. *Information Systems Frontiers*, 04 2022. doi: 10.1007/s10796-022-10252-x.
- [4] Janusz Pochmara and Aleksandra Swietlicka. Cybersecurity of Industrial Systems - A 2023 Report. *Electronics*, 13(7), 2024. ISSN 2079-9292. doi: 10.3390/electronics13071191. URL <https://www.mdpi.com/2079-9292/13/7/1191>.
- [5] International Standards on Auditing (ISA) and International Electrotechnical Commission (IEC). ISA/IEC 62443, security for industrial automation and control systems, 2020.
- [6] International Organization for Standardization (ISO) and Society of Automotive Engineers (SAE). ISO/SAE 21434:2021 - Road vehicles - Cybersecurity engineering, 2021.
- [7] European Committee for Electrotechnical Standardization (CENELEC). CLC/TS 50701 - Railway applications - Cybersecurity, 2021.
- [8] Rafal Leszczyna. Review of cybersecurity assessment methods: Applicability perspective. *Computers & Security*, 108:102376, 2021. ISSN 0167-4048. doi: <https://doi.org/10.1016/j.cose.2021.102376>. URL <https://www.sciencedirect.com/science/article/pii/S0167404821002005>.
- [9] MITRE. Common Vulnerabilities and Exposures. <https://www.cve.org/>, .
- [10] Forum of Incident Response and Security Teams (FIRST). Common Vulnerability Scoring System. <https://www.first.org/cvss/>.
- [11] Aileen Bacudio, Xiaohong Yuan, Bill Chu, and Monique Jones. An Overview of Penetration Testing. *International Journal of Network Security & Its Applications*, 3:19–38, 11 2011. doi: 10.5121/ijnsa.2011.3602.
- [12] Michael Howard and Steve Lipner. *The Security Development Lifecycle*. Microsoft Press, USA, 2006. ISBN 0735622140.
- [13] Michael Howard and David E. Leblanc. *Writing Secure Code*. Microsoft Press, USA, 2nd edition, 2002. ISBN 0735617228.
- [14] David Kieras. *Model-Based Evaluation*, page 1299. CRC Press, Taylor & Francis, 01 2012. ISBN 978-1439829431.
- [15] Gianfranco Balbo. *Introduction to Stochastic Petri Nets*, pages 84–155. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001. ISBN 978-3-540-44667-5. doi: 10.1007/3-540-44667-2_3. URL https://doi.org/10.1007/3-540-44667-2_3.
- [16] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, and Carl Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, 2004. doi: 10.1109/TDSC.2004.2.
- [17] D.M. Nicol, W.H. Sanders, and K.S. Trivedi. Model-based evaluation: from dependability to security. *IEEE Transactions on Dependable and Secure Computing*, 1(1):48–65, 2004. doi: 10.1109/TDSC.2004.11.

- [18] Barbara Kordy, Ludovic Piètre-Cambacédès, and Patrick Schweitzer. DAG-based attack and defense modeling: Don't miss the forest for the attack trees. *Computer Science Review*, 13-14:1–38, 2014. ISSN 1574-0137. doi: <https://doi.org/10.1016/j.cosrev.2014.07.001>.
- [19] Elizabeth LeMay, Michael D. Ford, Ken Keefe, William H. Sanders, and Carol Muehrcke. Model-based Security Metrics Using ADversary Vlew Security Evaluation (ADVISE). In *2011 Eighth International Conference on Quantitative Evaluation of SysTems*, pages 191–200, 2011. doi: 10.1109/QEST.2011.34.
- [20] Michael D. Ford, Ken Keefe, Elizabeth LeMay, William H. Sanders, and Carol Muehrcke. Implementing the ADVISE security modeling formalism in Möbius. In *2013 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 1–8, 2013. doi: 10.1109/DSN.2013.6575362.
- [21] Tod Courtney, Shravan Gaonkar, Ken Keefe, Eric W. D. Rozier, and William H. Sanders. Möbius 2.3: An extensible tool for dependability, security, and performance evaluation of large and complex system models. In *2009 IEEE/IFIP International Conference on Dependable Systems Networks*, pages 353–358, 2009. doi: 10.1109/DSN.2009.5270318.
- [22] Ken Keefe, Brett Feddersen, Michael Rausch, Ronald Wright, and William H. Sanders. An Ontology Framework for Generating Discrete-Event Stochastic Models. In Anne Remke, Paolo Ballarini, Benoît Barbot, Rena Bakhshi, and Hind Castel-Taleb, editors, *Computer Performance Engineering - 15th European Workshop, EPEW 2018, Proceedings*, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), pages 173–189, Germany, 2018. Springer. ISBN 9783030022266. doi: 10.1007/978-3-030-02227-3_12.
- [23] MITRE. Common Attack Pattern Enumeration and Classification. <https://capec.mitre.org>, .
- [24] Timothy Casey. Threat Agent Library Helps Identify Information Security Risks. *Intel White Paper*, 2007. doi: 10.13140/RG.2.2.30094.46406.
- [25] John F. Meyer, A. Movaghar, and William H. Sanders. Stochastic Activity Networks: Structure, Behavior, and Application. In *International Workshop on Timed Petri Nets*, page 106–115, USA, 1985. IEEE Computer Society. ISBN 0818606746.
- [26] William H. Sanders and John F. Meyer. Stochastic activity networks: Formal definitions and concepts. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2090:315–343, 2001. doi: 10.1007/3-540-44667-2_9.
- [27] Francesco Mariotti, Matteo Tavanti, Leonardo Montecchi, and Paolo Lollini. Extending a security ontology framework to model CAPEC attack paths and TAL adversary profiles. In *2022 18th European Dependable Computing Conference (EDCC)*, pages 25–32, 2022. doi: 10.1109/EDCC57035.2022.00016.
- [28] Francesco Mariotti, Andrea Bondavalli, Paolo Lollini, Leonardo Montecchi, and Simone Nardi. An extension of the ADVISE Meta modeling framework and its application for an early-stage security analysis of a public transport supervision system. *Journal of Reliable Intelligent Environments*, 2023. doi: 10.1007/s40860-023-00209-5.
- [29] PERFORM Performability Engineering Research Group. Möbius Website. <https://www.mobius.illinois.edu>, .
- [30] PERFORM Performability Engineering Research Group. ADVISE Meta Workshop 2016. https://www.mobius.illinois.edu/wiki/index.php/ADVISE_Meta_Workshop_2016, .
- [31] Falko Bause and Pieter S. Kritzinger. Stochastic Petri Nets: An Introduction to the Theory. *SIGMETRICS Perform. Eval. Rev.*, 26(2):2–3, August 1998. ISSN 0163-5999. doi: 10.1145/288197.581194.
- [32] PERFORM Performability Engineering Research Group. SAN Atomic Formalism, Möbius Wiki. https://www.mobius.illinois.edu/wiki/index.php/SAN_Atomic_Formalism#cite_note-Meyer:TPN:85-1, .
- [33] OWASP. OWASP Top Ten. <https://owasp.org/www-project-top-ten>.
- [34] MITRE. Common Weakness Enumeration. <https://cwe.mitre.org>, .

- [35] MITRE. MITRE ATT&CK. <https://attack.mitre.org/>, .
- [36] Egil Trygve Baadshaug, Gencer Erdogan, and Per Håkon Meland. Security modeling and tool support advantages. In *2010 International Conference on Availability, Reliability and Security*, pages 537–542, 2010. doi: 10.1109/ARES.2010.11.
- [37] D.C. Schmidt. Guest editor’s introduction: Model-driven engineering. *Computer*, 39(2):25–31, 2006. doi: 10.1109/MC.2006.58.
- [38] Giuseppe Riccardo Leone, Andrea Carboni, Simone Nardi, and Davide Moroni. Toward Pervasive Computer Vision for Intelligent Transport System. In *2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, pages 26–29, 2022. doi: 10.1109/PerComWorkshops53856.2022.9767376.
- [39] Chee-Wooi Ten, Chen-Ching Liu, and Govindarasu Manimaran. Vulnerability Assessment of Cybersecurity for SCADA Systems. *IEEE Transactions on Power Systems*, 23(4):1836–1846, 2008. doi: 10.1109/TPWRS.2008.2002298.
- [40] Sjouke Mauw and Martijn Oostdijk. Foundations of Attack Trees. In *Proceedings of the 8th International Conference on Information Security and Cryptology, ICISC’05*, pages 186–198, Berlin, Heidelberg, 2005. Springer-Verlag. ISBN 3540333541. doi: 10.1007/11734727_17.
- [41] Igor Kotenko and Elena Doynikova. The CAPEC based generator of attack scenarios for network security evaluation. In *2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, volume 1, pages 436–441, 2015. doi: 10.1109/IDAACS.2015.7340774.
- [42] Vasileios Mavroeidis and Siri Bromander. Cyber threat intelligence model: An evaluation of taxonomies, sharing standards, and ontologies within cyber threat intelligence. In *2017 European Intelligence and Security Informatics Conference (EISIC)*. IEEE, sep 2017. doi: 10.1109/eisic.2017.20.
- [43] Andreas Aigner and Abdelmajid Khelil. A Security Scoring Framework to Quantify Security in Cyber-Physical Systems. In *2021 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS)*, pages 199–206, 2021. doi: 10.1109/ICPS49255.2021.9468168.
- [44] Pontus Johnson, Robert Lagerström, and Mathias Ekstedt. A Meta Language for Threat Modeling and Attack Simulations. In *Proceedings of the 13th International Conference on Availability, Reliability and Security, ARES 2018, New York, NY, USA, 2018*. Association for Computing Machinery. ISBN 9781450364485. doi: 10.1145/3230833.3232799.
- [45] Ricardo J. Rodríguez, José Merseguer, and Simona Bernardi. Modelling Security of Critical Infrastructures: A Survivability Assessment. *The Computer Journal*, 58(10):2313–2327, 2015. doi: 10.1093/comjnl/bxu096.
- [46] Davide Cerotti, Daniele Codetta Raiteri, Giovanna Dondossola, Lavinia Egidi, Giuliana Franceschinis, Luigi Portinale, Davide Savarro, and Roberta Terruggia. SecuriDN: A Modeling Tool Supporting the Early Detection of Cyberattacks to Smart Energy Systems. *Energies*, 17(16), 2024. ISSN 1996-1073. doi: 10.3390/en17163882. URL <https://www.mdpi.com/1996-1073/17/16/3882>.
- [47] Marco Gribaudo, Daniele Codetta-Raiteri, and Giuliana Franceschinis. Draw-Net, a customizable multi-formalism, multi-solution tool for the quantitative evaluation of systems. In *Second International Conference on the Quantitative Evaluation of Systems (QEST’05)*, pages 257–258, 2005. doi: 10.1109/QEST.2005.10.
- [48] Michael Rausch, Ahmed Fawaz, Ken Keefe, and William H. Sanders. Modeling Humans: A General Agent Model for the Evaluation of Security. In Annabelle McIver and Andras Horvath, editors, *Quantitative Evaluation of Systems*, pages 373–388, Cham, 2018. Springer International Publishing. ISBN 978-3-319-99154-2. doi: https://doi.org/10.1007/978-3-319-99154-2_23.
- [49] Samuel Rodriguez Gonzalez, Juan Betancourt Osorio, Germán Pardo González, Daniel Cuellar, Camilo Gomez, Francesco Mariotti, Leonardo Montecchi, and Paolo Lollini. Modeling Attacker Behavior in Cyber-Physical-Systems. In *Proceedings of the 11th Latin-American Symposium on Dependable Computing, LADC ’22*, pages 117–124, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450397377. doi: 10.1145/3569902.3570188.
- [50] Marzieh Kordi, Francesco Mariotti, Roberto Magrini, Andrea Bondavalli, and Paolo Lollini. On the Usage of ChatGPT for Integrating CAPEC Attacks into ADVISE Meta Ontology. In *Proceedings of the Italian Conference on Cyber Security (ITASEC 2025)*, 2025.

- [51] OpenAI. ChatGPT. <https://chat.openai.com>, .
- [52] Dinesh Kalla and Nathan Smith. Study and analysis of Chat GPT and its impact on different fields of study. *International Journal of Innovative Science and Research Technology*, 8(3):827–833, 2023.
- [53] S. Guru Prasad, V. Ceronmani Sharmila, and M.K Badrinarayanan. Role of artificial intelligence based chat generative pre-trained transformer (ChatGPT) in cyber security. In *2023 2nd International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*, pages 107–114. IEEE, 2023.
- [54] Microsoft. Copilot. <https://copilot.microsoft.com>.
- [55] Google. Gemini. <https://gemini.google.com>.
- [56] OpenAI. GPT-2. <https://github.com/openai/gpt-2>, .
- [57] MITRE. Common Attack Pattern Enumeration and Classification Downloads. <https://capec.mitre.org/data/downloads.html>, .
- [58] Chenhui Zhang, Le Wang, Dunqiu Fan, Junyi Zhu, Tang Zhou, Liyi Zeng, and Zhaohua Li. VTT-LLM: Advancing Vulnerability-to-Tactic-and-Technique Mapping through Fine-Tuning of Large Language Model. *Mathematics*, 12(9), 2024. ISSN 2227-7390. doi: 10.3390/math12091286. URL <https://www.mdpi.com/2227-7390/12/9/1286>.
- [59] Xin Liu, Yuan Tan, Zhenghang Xiao, Jianwei Zhuge, and Rui Zhou. Not the end of story: An evaluation of ChatGPT-driven vulnerability description mappings. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 3724–3731, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.229. URL <https://aclanthology.org/2023.findings-acl.229>.
- [60] Reza Fayyazi and Shanchieh Jay Yang. On the Uses of Large Language Models to Interpret Ambiguous Cyberattack Descriptions, 2023. URL <https://arxiv.org/abs/2306.14062>.
- [61] Seyed Mohammad Taghavi and Farid Feyzi. Using Large Language Models to Better Detect and Handle Software Vulnerabilities and Cyber Security Threats, 2024. URL <https://doi.org/10.21203/rs.3.rs-4387414/v1>.
- [62] Kenta Kanakogi, Hironori Washizaki, Yoshiaki Fukazawa, Shinpei Ogata, Takao Okubo, Takehisa Kato, Hideyuki Kanuka, Atsuo Hazeyama, and Nobukazu Yoshioka. Comparative evaluation of NLP-based approaches for linking CAPEC attack patterns from CVE vulnerability information. *Applied Sciences*, 12(7):3400, 2022.
- [63] Minghao Shao, Boyuan Chen, Sofija Jancheska, Brendan Dolan-Gavitt, Siddharth Garg, Ramesh Karri, and Muhammad Shafique. An empirical evaluation of llms for solving offensive security challenges. *arXiv preprint arXiv:2402.11814*, 2024.
- [64] Olga Gadyatskaya and Dalia Papuc. Chatgpt knows your attacks: Synthesizing attack trees using llms. In *International Conference on Data Science and Artificial Intelligence*, pages 245–260. Springer, 2023.
- [65] OpenAI. GPT Tokenizer. <https://platform.openai.com/tokenizer>, .
- [66] The Eclipse Foundation. Eclipse Modeling Framework. <https://eclipse.dev/modeling/emf/>, .
- [67] Daniel Varro, Gábor Bergmann, Abel Hegedus, Akos Horvath, István Ráth, and Zoltán Ujhelyi. Road to a reactive and incremental model transformation platform: three generations of the VIATRA framework. *Software & Systems Modeling*, 15, 2016. doi: 10.1007/s10270-016-0530-4.
- [68] Frédéric Jouault, Freddy Allilaire, Jean Bézivin, and Ivan Kurtev. ATL: A model transformation tool. *Science of Computer Programming*, 72(1):31–39, 2008. ISSN 0167-6423. doi: <https://doi.org/10.1016/j.scico.2007.08.002>. Special Issue on Second issue of experimental software and toolkits (EST).
- [69] The Eclipse Foundation. Acceleo. <https://eclipse.dev/acceleo/>, .

- [70] Kristóf Marussy, Attila Ficsor, Oszkár Semeráth, and Dániel Varró. Refinery: Graph Solver as a Service: Refinement-based Generation and Analysis of Consistent Models. In *Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings, ICSE-Companion '24*, pages 64–68, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400705021. doi: 10.1145/3639478.3640045. URL <https://doi.org/10.1145/3639478.3640045>.
- [71] Joe Marini. *Document Object Model*. McGraw-Hill, Inc., USA, 1 edition, 2002. ISBN 0072224363.
- [72] Scott Means and Michael Body. *The Book of SAX*. No Starch Press, USA, 2002. ISBN 1886411778.
- [73] Xin Ren, Donglei Cao, and Beihong Jin. An efficient STaX based XML parser. In *Proceedings of the 11th Joint International Computer Conference*, pages 203–207, 2005. doi: 10.1142/9789812701534_0046. URL https://www.worldscientific.com/doi/abs/10.1142/9789812701534_0046.
- [74] Gábor László Hajba. Using Beautiful Soup. In *Website Scraping with Python: Using BeautifulSoup and Scrapy*, pages 41–96, Berkeley, CA, 2018. Apress. ISBN 978-1-4842-3925-4. doi: 10.1007/978-1-4842-3925-4_3. URL https://doi.org/10.1007/978-1-4842-3925-4_3.
- [75] lxml dev team. lxml. <https://lxml.de/>.
- [76] Department of Homeland Security. Alert (IR-ALERT-H-16-056-01AP): Cyber-Attack Against Ukrainian Critical Infrastructure, 2016.
- [77] David E. Whitehead, Kevin Owens, Dennis Gammel, and Jess Smith. Ukraine cyber-induced power outage: Analysis and practical mitigation strategies. In *2017 70th Annual Conference for Protective Relay Engineers (CPRE)*, pages 1–8, 2017. doi: 10.1109/CPRE.2017.8090056.
- [78] Jake Styczynski and Nate Beach-Westmoreland. When the lights went out, a Comprehensive Review of the 2015 attacks on Ukrainian critical infrastructure. Technical report, Booz Allen Hamilton, 2019.
- [79] MITRE. 2015 Ukraine Electric Power Attack, MITRE ATT&CK. <https://attack.mitre.org/campaigns/C0028/>, .
- [80] Gui-lin Cai, Bao-sheng Wang, Wei Hu, and Tian-zuo Wang. Moving target defense: state of the art and characteristics. *Frontiers of Information Technology and Electronic Engineering*, 17(11): 1122–1153, 2016. doi: 10.1631/FITEE.1601321.
- [81] Francesco Mariotti, Lorenzo Manetti, and Paolo Lollini. Modeling Moving Target Defense strategies and attacks with SAN and ADVISE. In *Proceedings - 2023 IEEE 34th International Symposium on Software Reliability Engineering Workshop, ISSREW 2023*, pages 160–161, 2023. doi: 10.1109/ISSREW60843.2023.00066.
- [82] Tom Roeder and Fred B. Schneider. Proactive obfuscation. *ACM Transactions on Computer Systems*, 28(2), 2010. doi: 10.1145/1813654.1813655.
- [83] OpenAI. o3-mini. <https://openai.com/index/openai-o3-mini/>.
- [84] DeepSeek. DeepSeek-R1. <https://www.deepseek.com/>.
- [85] Ontotext, Semantic Web Company, PoolParty and GraphD. Graphwise. <https://graphwise.ai/>.
- [86] Davide Cerotti, Daniele Codetta-Raiteri, Giovanna Dondossola, Lavinia Egidi, Giuliana Franceschini, Luigi Portinale, and Roberta Terruggia. Evidence-based analysis of cyber attacks to security monitored distributed energy resources. *Applied Sciences (Switzerland)*, 10(14), 2020. doi: 10.3390/app10144725.
- [87] Silvano Chiaradonna, Paolo Lollini, and Felicita Di Giandomenico. On a Modeling Framework for the Analysis of Interdependencies in Electric Power Systems. In *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07)*, pages 185–195, 2007. doi: 10.1109/DSN.2007.68.
- [88] Francesco Brancati, Diamantea Mongelli, Francesco Mariotti, and Paolo Lollini. A Cybersecurity Risk Assessment Methodology for Industrial Automation Control Systems. *International Journal of Information Security*, 24(76), 2025. doi: 10.1007/s10207-025-00990-9.

- [89] András Földvári, Francesco Brancati, and András Pataricza. Preliminary Risk and Mitigation Assessment in Cyber-Physical Systems. In *2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pages 267–274, 2023. doi: 10.1109/DSN-W58399.2023.00067.
- [90] Henk Jonkers, Henderik Proper, and M Turner. TOGAF and ArchiMate: A Future Together. 11 2009.
- [91] András Földvári, Gergely Biczók, Imre Kocsis, László Gönczy, and András Pataricza. Impact Assessment of IT Security Breaches in Cyber-Physical Systems: Short paper. In *2021 10th Latin-American Symposium on Dependable Computing (LADC)*, pages 1–4, 2021. doi: 10.1109/LADC53747.2021.9672582.