Probabilistic Mathematical Modelling for Security Risk Assessment

Denis Kolev

Submitted in fulfilment of the requirements for the degree of
Doctor of Philosophy

School of Computing Science

College of Science and Engineering
University of Glasgow

08/2021

Abstract

This thesis presents a novel framework for security risk assessment (SRA) and identification, comprising mathematical algorithms and a family of models. Recently, due to the growing incidence of cyber-attacks and cyber-fraud, as well as terrorist attacks and other adversarial activities, qualitative and comprehensive SRA and security risk management have become increasingly important. For large-scale systems, SRA and related data processing tasks are challenging due to the large amount of available information, as well as the diversity and complexity of data sources. However, SRA relies mainly on procedures that, despite being well-formalised, are manual, which introduces the “human factor" as early as the system design stages. The existence of multiple possible threats, along with the variability of the information received from different sensors, has increased the complexity of situation awareness analysis, which often results in scenarios where security officers (operators) are overwhelmed with data and, in certain cases, a high false positive rate. The primary motivation behind this work was to develop a general mathematical approach to SRA based on statistical data processing, data fusion techniques, and game theoretic models.

The proposed framework is based on a slight adjustment of the existing SRA methodology for threat modelling, augmented by additional mathematical formalisations. In general, two primary models are presented as the main contribution:

- “Static Model" for SRA, which is applicable at the stage of designing the protection of the considered system.
- “Dynamic Model" for the processing of generic security-related data, which is applied when the system is in operation.

Both models use graph theory as a basis. The static model uses game theory for optimal protection design, while the dynamic model applies Bayesian inference techniques for “online" data processing.

Extended Abstract

The static model proposed in this study relies on an extension of network security games. It is used to leverage the advantages of the “standard" expert-based security risk assessment (SRA) procedures, as well as to provide a robust formalisation for general large-scale infrastructure protection issues. The proposed model’s instantiation procedure is designed with a strong link to standard SRA methods. In this way, the resulting model provides a mathematical formalisation that shares the entities and terminology of standard “manual" methods. The security control selection problem is modelled as a multi-objective optimisation problem.

The static model is built upon two interwoven models, the asset and attack models, thus establishing an analogy with classical SRA methods. The asset model describes the system and its parameters, while the attack model is used to formalise possible threat scenarios. Both models are used to formulate a multi-objective optimisation problem. A specific solver for this problem is described in detail with a theoretically-grounded justification for its correctness. The model is validated using an airport case study, where some of the methodology’s essential building blocks are discussed. The work reported in this paper demonstrates the feasibility of a generalised mathematically-founded approach to SRA in large-scale systems engineering. Proposed formalism provides means for significant automatisation of some of the SRA phases, namely risk treatment (selection of security controls) and risk acceptance (evaluation of the risk after deployment of security controls), automatic evaluation of different scenarios, and “what-if" simulations.

Compared to the static model, the dynamic model shares the same asset and threat model, but it also introduces the concept of “event detectors". The model performs online information fusion and analysis in a manner similar to the use of Bayesian tracking algorithms. It involves Kalman filter-like inference methodology combined with variational distribution approximation methods. Theoretical properties of the model are proven and discussed, resulting in an exact algorithm for attack detection. Similar to the static model, an event detector allocation is considered an optimisation problem. The model is validated based on a cybersecurity dataset, which demonstrates that the proposed framework is not limited to infrastructure protection problems. The model shows significant improvement on standard machine learning cyber event detection dataset, compared to stand-alone event detectors.

The two models described in the thesis suffer from design limitations, including:

- Model construction still requires expert judgment, thereby transferring the manual analysis to another domain of operation. Heuristic and “automated" model construction

methods are given and demonstrated in the thesis, but these do not cover all possible problems.

- Both the static and dynamic models focus on the case of a single attack (possibly multi-stage) occurring at the same time, executed by a single attacker. While there are mechanisms for a multi-attack analysis for the static model, the dynamic model does not present such a possibility.

- Attacks executed in several phases, which are separated by long time intervals, are challenging for both models. A heuristic for the dynamic model exists, which is used and demonstrated in the experimental phase of this thesis. However, the current formulation of the static model does not consider such attacks.

- The interaction of the security controls and event detectors with the attacker considered in the models is over-simplified and should be extended. The same argument is valid for the adversary's motivation since only zero-sum games are considered, despite the fact that this selection is justified.

None of the abovementioned limitations are major, and so this thesis did not seek to address them. However, they will be addressed in future work.

Acknowledgments

I would like to thank my supervisors: professors Christopher Johnson and Alice Miller, who offered invaluable guidance throughout my research. Special thanks also go to my family, who supported me, as well as my research colleagues, who were a source of great inspiration.

I dedicate this dissertation to my family.

Table of Contents

List of Tables

List of Figures

Part I

Research Overview

Chapter 1

Introduction

1.1 Motivation

Historically, critical infrastructures relied on bespoke software, and because they were loosely integrated, this limited the potential impact of knock-on effects between different systems. Those days are gone. Now, most industries make extensive use of commercial “off-the-shelf" systems [69], which are not specifically intended for security-related applications. These systems include Linux, Voice over Internet Protocol (VoIP), and augmentation systems for GPS, the strengths and weaknesses of which are known by a growing number of attackers. At the same time, a number of large-scale projects are increasing interdependencies both within and between infrastructures.

Recent large-scale programmes in air traffic management (ATM), including the Single European Sky ATM Research (SESAR) programme in Europe, along with the Next Generation Air Transportation System (NextGen) in the United States, illustrate the abovementioned points. In both cases, these programmes are readying novel operational concepts and technological enablers [49, 39, 40]. These concepts rely on large-scale data transfers across national boundaries and between a host of integrated systems. Regulatory organisations are gaining an awareness of potential threats and vulnerabilities [34], as reflected in the attempts of the Federal Aviation Administration and the European Aviation Safety Agency to establish ATM security as an integral part of the system engineering process.

This thesis addresses the challenge of supporting security risk assessment (SRA), as well as the selection of control and event detectors, in large-scale engineering programmes. Current practices and standards for security risk management involve the identification of security risks and the implementation of associated controls at a system- or component-level. Risk assessment is typically performed by experts, and it relies on the combined use of qualitative and quantitative methods. However, the higher levels of interconnectivity across infrastructure components require the analysis of threat propagation within and across the associated supply chains. Various security risk management methodologies exist, but few are specifically tailored to the design and development process.

At the same time, in recent years, methods for mathematical modelling in infrastructure protection have been proposed [5, 15, 16]. Several of these methods have been applied successfully [63], the most prominent among which are presented in this thesis, and discussed in terms of their methods and functionality. Different types of models deal with various problems. For example, game theoretic methods are applied to randomised patrol scheduling problems, while data mining and computer vision models are used for intrusion detection (including in the cyber-domain). However, each model solves only a narrow scope of specific problems. Even generic security-control allocation tasks deal with diverse cases, which are not generalisable.

With the above considerations in mind, the goal of this research is to develop a mathematically-founded method to support SRA in the general case, which involves incorporating the main abstractions defined by the standards . In particular, the approach evaluates the perceived optimality of the security controls and event detectors, analysing the cumulative effect of the application of these security components to the system at the stages of system development and operation. For different security system designs, an evaluation of the “protection" of the system is given, as well as the quality level of “situation awareness". It also assists in the development of “what-if" simulations in support of risk mitigation decision-making processes. The latter allow for the identification of appropriate controls and detectors, as well as their placement within the (sub-)system context. Additionally, knowledge of the joint properties of different components of the security system allows a high-level aggregated analysis to be performed of the alerts that are produced during system operation, considering the temporal development of the received data. This is valuable in lowering the rate of false positives in attack detection, or increasing the true positive rate.

Model development in this thesis is explicitly linked to the SRA process for the purpose of ensuring the generality of the models. The framework is designed in a way that allows a combination of lower-level models (e.g., data mining-driven detectors), as shown in the validation sections of this thesis.

1.2 Research Contribution

This thesis seeks to target SRA at two stages of the condition of the system:

- Design Stage: In this stage, decisions are made about the nature, functionality, and deployment of protection mechanisms.

- Operation Stage: The system is deployed and launched in this stage. The data produced by different components and security detectors can be analysed for attack prevention, detection, and mitigation. In the thesis, the focus is given to attack detection.

Within the scope of this thesis, both stages are considered. The following two corresponding research contributions are presented:

- Generic framework for security risk analysis domain, which is linked to the existing SRA framework. The framework strictly translates the main entities used within the Security Risk Assessment Methodology (SecRAM) into mathematical formalisms, and provides a basis for building a graph-based model. Less strict correspondence is built with ISO 27005 standard. The construction methodology for the graph-based model is presented, which allows the modelling of complex, conditional, multi-step attacks with multiple targets. The process of defining and selecting security controls is represented as a multi-objective zero-sum security game. The solution is found using standard methodology. For this, the game is converted to a sequence of pairs of mixed-integer linear programming problems and proven to be consistent with the requirements of the framework. In this research, the security game is referred to as a “static model". The main scientific contribution is related to the designed correspondence between the model and SecRAM (ISO 27005) and the derivation of the Pareto-optimal search algorithm. The approach is functionally compared with several competing methods, advantages and disadvantages of the “static model" are listed.
- Over the framework designed in the first chapter of the thesis, a method for security awareness analysis is presented, which may be regarded as a logical continuation of the static model. Problems relating to system design are further translated into system operational problems. The proposed “dynamic model" is an “in-operation" online data stream processing generic model, which is proposed and derived using standard Bayesian inference. It does not provide a direct link to SRA, but it is based on the framework of the static model, and can be understood as an additional security control. Additionally, as shown in the corresponding chapter of the thesis, the dynamic model formalises several system security design-related problems.

Both proposed models are validated in the thesis using real and synthetic datasets.

1.3 Publication Summary

- Kolev, D. and Johnson, C., “An Extension of Network Security Games for Large-Scale Infrastructure Protection", Workshops at the 31st AAAI Conference on Artificial

Intelligence. This paper summarises the model and the results presented in this thesis' chapter “Static Security Model Description and Evaluation".

- Markarian, G. and Kolev, D., “Fully Automatic Electro-Optical Drone Detection System", MSG-SET-183. This paper presents an example of a security detector integrated within a sensor fusion platform, which can be analysed using the methodology described in “Dynamic Security Model".

1.4 Thesis Outline

The remainder of this thesis is organised as follows: Chapter 2 presents notable existing approaches for addressing SRA problems using mathematical tools, mostly based on game theory or Markov decision processes. This chapter also considers the state-of-the-art approaches for Bayesian inference, as well as their applications in SRA and security control design. It outlines the attack trees modelling method and aligns it with some of the game theory formalisms. Chapter 3 describes the developed approach for security domain modelling, building a parallel between the mathematical formalism and procedures of currently existing methods for SRA. The two models that form the main contribution of this thesis are given in this chapter. The following chapter, Chapter 4, states a generic security control selection problem using the formalism defined in the previous chapter, and it describes a solver for a specific case of the general problem statement. The developed approach is applied to an airport model example, performing an experiment-based analysis of the model's theoretical properties. The model is functionally compared with several competing methods. Chapter 5 presents a generic description of the “online" security awareness model, which relies both on the Bayesian inference approaches and threat graph formalism. The description is followed by model validation over a cyber-security dataset, showing the advantages of the proposed method against the “per-detector" judgment. The closing chapter of the thesis presents conclusions and potential areas for future research.

Chapter 2

Existing Mathematical Models for Security Risk Assessment

This chapter presents state-of-the-art methodologies and models for security risk assessment (SRA). A high-level review of the diverse approaches to SRA is given in order to contextualise the models presented in this thesis. Special attention is paid to game theoretic and probabilistic graphical methods, as well as the link between the corresponding approaches and classical SRA methods, which are presented later in this chapter.
The purpose of a risk assessment methodology is to establish rules of the assessment, the targets for actors involved, the terminology used to describe risk, and the quantifying and qualifying criteria [36]. In addition, a risk assessment methodology allows degrees of risk to be compared, and it specifies the documentation that must be collected and produced based on the results of the assessments and the follow-on activities. The goal of a risk assessment framework is to establish an objective measurement of the risk level, which allows organisations to understand business risks related to critical information and assets, both qualitatively and quantitatively. Ultimately, risk assessment frameworks provide the tools needed to make business decisions regarding investments in people, processes, and technology such that risk is reduced to an acceptable level [2].
The first part of this chapter, using examples, presents a description of the “standard" SRA approach. Based on this description, a set of requirements for the proposed model is defined. Thereafter, a high-level overview of families of models is given, which is used to narrow the search for proper formalisation. Separately, an overview of the “attack trees" threat modelling technique is presented and compared against graphical game-theoretical models. The main concepts and most prominent results for each of the selected families of models are then presented.

2.1 Classical Methodologies for Security Risk Assessment

In this section, a brief overview of SRA methodologies is given, using examples. Various SRA and security risk management methodologies have been proposed in the literature. The standards and best practices that have been identified vary substantially in terms of their methods, rigour, and process steps. Nevertheless, the core artefacts of SRA include assets, vulnerabilities, threat scenarios, and security controls.
Hereafter, the thesis considers several SRA methods, identifying the main similarities and general structure of the approaches. This section concludes with an identification of the main possible ways for automatisation of the methods and derivation of the requirements to the proposed methodology.

2.1.1 ISO 31000

ISO 31000 is a security risk assessment standard based on Austrian, Australian, and New Zealand's national approaches. This method has evolved into a global and widely accepted leading standard. Due to the generality of the approach, it can be applied to a wide range of organisations and systems, regardless of their type or size [24].
ISO 31000 process consists of several high-level steps presented in the following list.

- Context establishment. During this phase, internal and external contexts are specified. Internal context defines governance, organisational structure, processes, roles and responsibilities, policies, objectives of projects, assets, and other characteristics of the organisation. External context includes social, cultural, political, legal, regulatory, financial, technological, economic, and natural aspects. They are used to build an understanding of the stakeholders' interests within the organisation and externally, define the purpose of risk assessment, and identify risk criteria (which risks should be considered and how to evaluate them).

- Risk assessment. This phase subsumes the complete process of risk identification, risk analysis, and risk evaluation. The first part - risk identification - identifies the sources of risk, areas of impact, risk events, and their consequences. The main output of this part is a list of all possible risks. The next stage is risk analysis. For each item in the list, the corresponding likelihood of occurrence and potential consequences (impact) is defined. Risk analysis focusses on determining the sources of the risks, communicating with the stakeholders for collecting information relevant for decision making, evaluating the risks against the risk criteria. Risk analysis can be

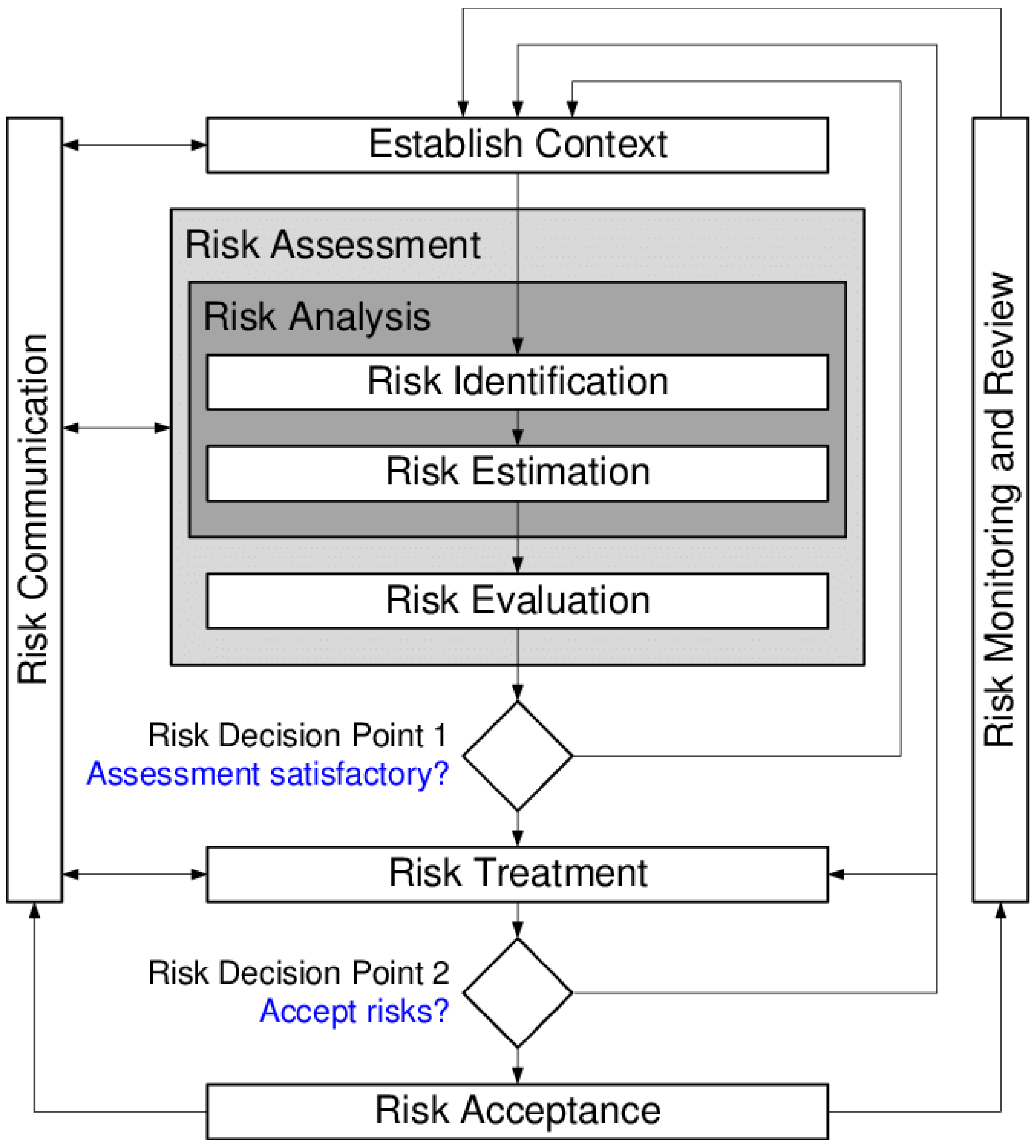


Figure 2.1: ISO 27005 process scheme.

of a quantitative, semi-quantitative, or qualitative nature. The risk evaluation stage ranks the risks for further treatment.

- Risk treatment. This last phase of the analysis deals with the selection and implementation of one or more options for modifying (mitigating) risks. ISO 31000 defines a number of possible strategies that include deploying additional security controls, moving the responsibility for risk mitigation to other organisations, changing the specifics of the organisation's activity, and accepting the risk as it is.

Each stage of the assessment should be linked with communication with the stakeholders and experts in the field. The approach is considered to be iterative and should be repeated periodically.

ISO 31000 suggests a high-level description of the process. This level of precision allows it to be adopted for a large variety of different systems/organisations. However, most steps of the process are not detailed and rely purely on the stakeholders' decision-making and expert judgement. At each stage, supportive information may be used presented in the form of charts, graphs, statistics, and others. However, the standard does not provide any explicit suggestions for possible automation of the decision-making process at each step and lacks pre-defined ways for quantitative evaluation of the risks. The standard suggests that relevant interconnections between systems and cumulative effect of the attacks should be taken into account during risk assessment, but no explicit procedure or algorithm to be followed is given. This may be explained by the requirement for the standard to be on a high level and be applicable in various range of domains.

2.1.2 ISO 27005

ISO 27005 [36] belongs to the ISO 27000 family of standards and represents an extended risk management process, which is specifically adapted to the requirements of information security. The standard itself contains the description of the SRA process, which is still applicable to information and infrastructure security. On a high-level, the ISO 27005 process is represented in figure 2.1.

The first step of the analysis is context establishment. It defines the following subjects of the analysis:

- Impact criteria definition, for example, financial, human, or reputation losses. Impact criteria could vary depending on the specifics of the protected system. Impact criteria are defined as real-valued parameters, characterising the level of caused damage. The usual range of impact criteria is from 0 to 5, where the scale is determined by the security officers conducting the analysis.
- Scope identification, which represents the general view of the organisation on the assets to be protected. The assets could be both tangible (buildings, computers, or personnel) and intangible (data, functions).
- In case the assessment is planned to be a part of some standard process within the organisation, it is important to identify the unit responsible for carrying that out.

The next step of the analysis is risk assessment, which is split into three parts.

- **Risk Identification** examines what kind of incidents can happen and which of them lead to a loss. This includes both the incidents which are under the control of the organisation and those that are based on external influences. Identification of possible threats involves listing critical assets of the system (that could be both tangible and intangible); definition of main vulnerabilities and ways of their exploitation; creation of a set of possible “controls", i.e. means for mitigation or prevention of the attack; setting different types of consequences of the attack, for example, money, reputation, human losses, or any other relevant criteria.

- **Risk Analysis** analyses the threats in terms of the likelihood of their appearance and numerically evaluates them against the set of criteria identified at the previous step. Risk analysis can either be of qualitative or quantitative nature, depending on the type of considered criterion. For example, some of the criteria could be defined as exact numerical value or range of values (financial losses), some of the criteria could be characterised using descriptive scales (like "low", "medium", "high"). Likelihood assessment considers how often a specific threat might occur and how easily related vulnerabilities can be exploited. This can be assessed by the experts in the field and using historical data.

- **Risk Evaluation** re-considers the list of threats, given the evaluated criteria and likelihoods. Based on expert judgement, risks are ranked, and a list of prioritised threats is produced.

The next phase of the process is risk treatment, described as a complex step consisting of four possible options: risk modification, risk retention, risk avoidance, and risk sharing. Those options are not mutually exclusive; a combination of strategies is possible. Risk modification strategy assumes a selection of (new) controls to mitigate the risk. This decision should be taken with respect to the existing resources. The rest of the strategies define different measures for transferring the responsibility of the risk handling to a different organisation or accepting the low-impact risks as they are.

The last phase of the analysis is risk acceptance, when all identified risks are re-evaluated given the treatment strategies selected and accepted or not accepted based on the *residual* impact levels (estimated impact levels after treatment). ISO 27005 provides a high-level algorithm for risk assessment, defines the vocabulary and main entities of the SRA process. It is more detailed compared to ISO 31000, specifying exact types of entities to be analysed during the context establishment phase. However, it does not specify a constructive approach for quantitative evaluation of the risks, relying significantly on expert judgement at each step of the analysis. The standard does not provide any suggestions and instructions for a detailed formalisation of the attacks and any means for automation of the decision-making during risk identification and risk analysis.

2.1.3 Eurocontrol's Security Risk Assessment Methodology

Within the SESAR context, work package 16.02 devised a “Security Risk Assessment Methodology" (SecRAM) based on ISO 27005 [2]. SecRAM is used in the present thesis to build a correspondence between the proposed formalism and SRA standards. SecRAM was selected as a baseline from the side of classical risk assessment for the following reasons:

- The methodology was developed using international SRA methods as a baseline, and it is applied in practice in the scope of SESAR for some of the operational focus areas (OFAs) (i.e. large-scale system development projects with several sub-components).
- Various papers [34, 42] have explained how to apply SecRAM both to infrastructure and cybersecurity problems. Additionally, the methodology has been used in several Horizon 2020 projects, including GAMMA, which demonstrates the applicability of the method and its usage within the European Union.
- The author consulted with security specialists who are responsible for the development of SecRAM. This fact is important for the validation of the thesis results, particularly those relating to the static model, as well as the establishment of the correspondence between the mathematical formalisms and SecRAM.

This SRA method has been extensively used within SESAR programme in a number of OFAs. SecRAM was applied in a number of OFAs, some of them are listed below.

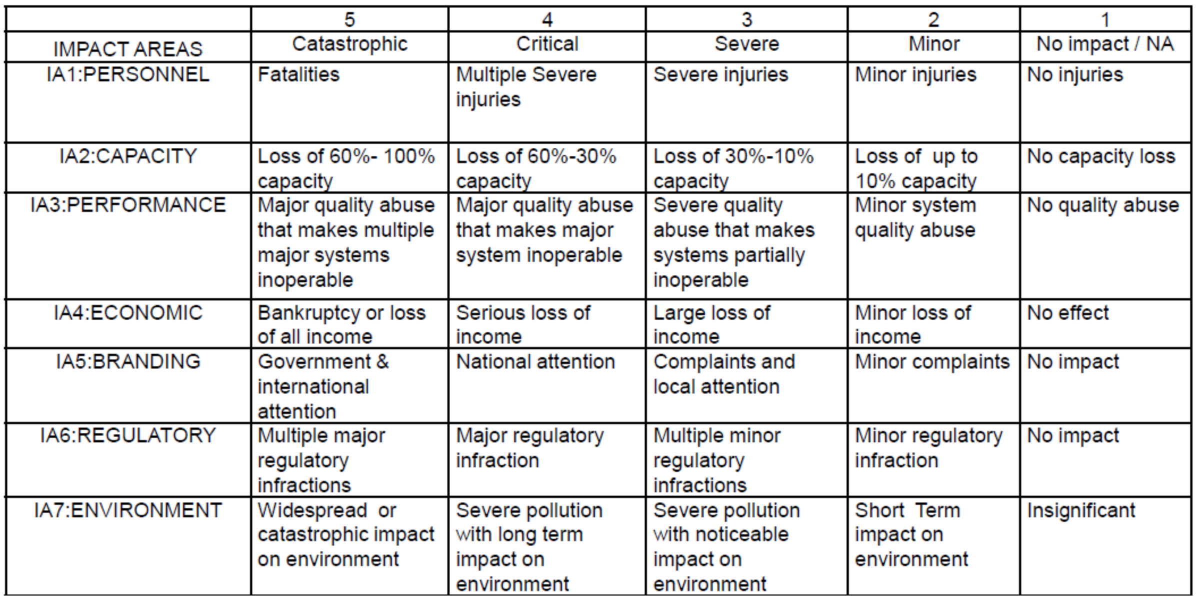
- OFA 01.01.01: Low visibility procedures using ground-based augmentation system (for satellite positioning augmentation).
- OFA 01.03.04 Extended Flight Plan targets to develop enhanced aircraft routing procedures.
- OFA 03.03.01 Ground-Based Separation Provision in En-Route targets to develop systems and processes to increase airspace capacity, reduce flight variability, optimise the costs of operation.
- OFA 04.01.02 Enhanced Arrival and Departure Management. This project aims to increase the capacity of airports, reduce the environmental impact, and reinforce the safety of air transportation by using different decision-support techniques and enhanced processes.
- OFA 05.03.07 Network Operations Planning. The project aims to deploy integrated airport collaborative planning processes, including surface routing planning activity. The goal is to optimise airport processes and airline operations on the ground.

- OFA 05.06.02, Continuous Descent Operation. In this project, a set of means for an improvement of the aircraft descend procedures.

In this section, in parallel with the description of the methodology, a mathematical formalisation for some of the steps is given. This formalisation is further used for the alignment of the proposed model instantiation process and the main steps involved in SecRAM.
The first step in SecRAM defines the scope of the analysis (i.e., impact assessment) and distinguishes between primary and supporting assets. The principal idea is that the protected system is modelled as an information system. The term “primary asset" (PA) is used to denote the intangible services and information elements. A formal definition of PA is given as follows:
Primary Asset: An intangible function, service, process, or piece of information that is an element of the system within the scope of the project, and which has value to the system.
These services and information are instantiated and enabled by "supporting assets" (SAs), as defined below:
Supporting Asset: An enabling entity for the PA. SAs relate to operational, organisational, and technical means that possess vulnerabilities, and which might be susceptible to security attacks.
As shown in due course, SAs can be considered to define the set of targets for adversaries. Each SA is related to one or more PA, which establishes the relationship between the services and components of the system.
For each PA, three high-level criteria are considered, often referred to as comprising the CIA triad:

- **Confidentiality**: Property of information that is not made available or disclosed to unauthorised individuals, entities, or processes.
- **Integrity**: Property of safeguarding the accuracy and completeness of assets.
- **Availability**: Property of being accessible and usable on demand by an authorised entity.

In turn, it is necessary to provide an account of the impact. It is defined for seven impact areas (or types of impact), and this occurs for each PA. The value of the impact ranges from 0 to 5, and this value is evaluated as shown in Figure 2.2 [43]. In this way, a value is defined for each PA, each impact area, and each element of the CIA triad. As shown in Figure 2.2, the impact measures not only tangible damage to personnel and the system, but also it measures the qualitative decrease in system operation (e.g., capacity and performance), reputation losses (e.g., branding), and others. Therefore, the result of the impact definition



	5	4	3	2	1
IMPACT AREAS	Catastrophic	Critical	Severe	Minor	No impact / NA
IA1:PERSONNEL	Fatalities	Multiple Severe injuries	Severe injuries	Minor injuries	No injuries
IA2:CAPACITY	Loss of 60%- 100% capacity	Loss of 60%-30% capacity	Loss of 30%-10% capacity	Loss of up to 10% capacity	No capacity loss
IA3:PERFORMANCE	Major quality abuse that makes multiple major systems inoperable	Major quality abuse that makes major system inoperable	Severe quality abuse that makes systems partially inoperable	Minor system quality abuse	No quality abuse
IA4:ECONOMIC	Bankruptcy or loss of all income	Serious loss of income	Large loss of income	Minor loss of income	No effect
IA5:BRANDING	Government & international attention	National attention	Complaints and local attention	Minor complaints	No impact
IA6:REGULATORY	Multiple major regulatory infractions	Major regulatory infraction	Multiple minor regulatory infractions	Minor regulatory infraction	No impact
IA7:ENVIRONMENT	Widespread or catastrophic impact on environment	Severe pollution with long term impact on environment	Severe pollution with noticeable impact on environment	Short Term impact on environment	Insignificant

Figure 2.2: Correspondence between impact area and value of evaluated impact

stage is a 3D table. From a mathematical perspective, the impact can be defined as a function of three variables: $I(p,i,c)$, where p is one of the PAs, i is one of the seven impact areas, and $c \in \{C,I,A\}$. The values of this function are defined by security experts.

The next step involved in SecRAM is to define the impact for each element of the CIA triad for each PA, which is independent from the impact area. This is accomplished by considering the maximal impact among all impact areas for a given PA and for each function. Finally, a 2D table of impact values is obtained, which is defined for all PAs and, furthermore, for all of the elements in the CIA triad.

Formally, the impact for PA, p, and criterion c is defined as follows:

$$I_{PA}(p,c) = \max_i I(p,i,c) \tag{2.1}$$

The function I_{PA} is the result of the asset analysis of the system.

The next step involves defining the threat scenarios against SA. Each threat scenario compromises a subset of $\{C,I,A\}$ functions of the corresponding supporting assets. To evaluate the impact of the threat t against supporting asset s, a list of criteria compromised by the threat for each PA must be defined, which highlights the relevance of the table produced in the previous step. Therefore, for every threat t, a set of relevant “service-criterion" pairs is designed by security experts. Formally, we can define these pairs as a function $r(t) = \{(p,c)\}, p \in PA, c \in \{C,I,A\}$.

After defining the service-criterion pairs, the impact of threat t on each SA is defined as a maximal impact among all PAs linked to the current SA, and among all the criteria in the set.

Formally, an impact for each SA and for each threat is defined as follows:

$$I_{SA}(s,t) = \max_{(p,c)\in r(t), p\in PA(s)} I_{PA}(p,c) \tag{2.2}$$

Here, $PA(s)$ defines the list of PAs linked to the SA s.
Using the procedures defined above, it is possible to obtain the lists of threats, SAs, and corresponding threats. Additionally, a corresponding likelihood value is assigned for each threat. Depending on the likelihood and impact values, the final risk score is given using a predefined "impact $\times$ likelihood" matrix, where risk varies from low to high.
The next step involves selecting security controls, where – for each threat – one of the following actions should be chosen:

- Accept or tolerate the threat when no additional action is needed.
- Reduce or treat the threat by installing security controls.
- Avoid or terminate the threat by stopping the activity.
- Transfer the threat to another party.

Expert judgement is primarily used to select the actions and, potentially, the security controls, and the decision-making process is informed by system-specific details. The way that security controls or selected actions mitigate the risk is not formally given by the standard, and so should also be considered on the basis of expert judgement.
The formal functions and definitions introduced in this subsection are further used to describe the proposed model, and to compare it to SecRAM.
In summary, it is clear that SecRAM provides a more detailed and strict formalisation of high-level management procedures compared to ISO 31000 and ISO 27005, all the while leaving room for expert judgement. In the following chapters, mathematical properties of the SecRAM "solution" are considered on the basis of this type of analysis. Despite the fact that some of the steps of SecRAM support partial automation (impact evaluation, for example), the decision-making at the threat mitigation phase is done manually.

2.1.4 Security Risk Model Criteria

After introducing 3 standard methodologies, we discuss their distinctive features that are important for incorporation to the proposed mathematical model. These features can be understood as required conditions that must be satisfied so that the security risk experts can use the proposed formalism.

1. System Description Generality: The approach should be applicable to any type of system, and so it should deal with high-level entities that can be adjusted to match the scope of the interest of the security expert.
2. Attack Description Generality: The approach should allow the user to describe an arbitrary threat scenario and diverse types of attack. The attacks can be interdependent, and the scenario could include multiple steps.
3. Protection Description Generality: The approach should be able to introduce diverse security controls and protection means.

It is important to recognise that the list given above is not necessarily sufficient for a qualitative SRA. However, meeting these requirements would allow the model to be technically applied to SRA problems. It is possible to add numerous further restrictions and requirements, such as the necessity to model sequential counter-actions explicitly, to capture simultaneous attack effects, and to consider different types of impact. It is also obvious that meeting the requirements listed above would require a level of simplification compared to the models that deal with specific attacks against a specific system.
In addition to the definition of the modelling criteria, it is essential to outline the application specifics of the developed model (i.e., the group of end-users and required expertise). As stated in the motivation section of this thesis given in Chapter 1, the models were developed to provide support for the SRA process. Therefore, the model's end-users are security officers and specialists who perform the analysis and observe the system in operation. It is clear that the lower the instantiation complexity, the less mathematical skill that is required from the user. Therefore, it is preferable to use “low-complexity" abstractions such as lists, tables, graphs, and other visual elements, which are well-aligned with the entities used in SecRAM.

2.2 Mathematical Methods in Security Risk Assessment

The mathematical methods used for SRA are diverse. They cover a broad spectrum of methods, including formalised methodologies [15], crossing the intermediate expert-based assessments with supporting formal structures (e.g., attack trees and correlation plots)[57], and converging to fully expert-based decision tree methods used to classify or give a quantitative assessment of the security level.
In this thesis, automatisation and artificial decision support techniques were the focus, as discussed in the motivation section. Therefore, a wide collection of families of methods is

considered in the next subsection, which is given to define further a more detailed model selection.

2.2.1 Main Families of Models

In recent years, the following areas of applied mathematics have been successfully exploited in the area of SRA for similar applications: data mining and statistical learning, operation research-based algorithms for security problems, and statistical graphical models.

- Data Mining and Statistical Learning: These areas have been successfully applied in various security problems, especially in intrusion detection (both in cybersecurity and infrastructure security). The designed methods are usually formalised as anomaly detection tasks, solved using techniques such as principal component analysis (or any other subspace projection method) [47], one-class support vector machines (SVMs) [68], or kernel density estimations [18]. Recent advances in deep learning have also been successfully used in applications such as malware detection [56]. For infrastructure protection, object tracking techniques are commonly applied, for example, in automatic video surveillance [62] or radar systems [53]. The advantage of these methods is that it is unnecessary to develop a detailed model of the system, as the training procedure aims to adapt a parametric model to the specifics of the target domain. However, these algorithms are used for specific tasks with a narrow domain, and exploitation is performed at the level of systems components using dedicated equipment (e.g., network analysers, protocol scanners, and malicious code or botnet detectors). Statistically-based algorithms depend on training data and on the structure of the selected algorithm, which significantly narrows the scope of the application. In the case, one exception may be online self-learning models, which can adjust to novel data. Nevertheless, this family of models still requires that the data are presented in the standard format of generic security-descriptive “feature-vectors", which form a separate problem to be solved. Another typical problem associated with statistical methods, especially more complex examples (e.g., deep neural networks), is poor interpretability of the output and complex error analysis/debugging. For this reason, despite the fact that data mining models are used in the experimental part of the thesis as sub-components, they will not form a basis for the development of the main research contributions.

- Operation research-based algorithms for security problems: These are usually designed for large-scale applications, and they solve tasks such as patrol randomisation and security control allocation. As an input, these methods take a formal description of the system, a set of variable assets to be optimised (e.g., security control allocation), and a single – potentially several – target “gain" function. The gain function should be optimised subject to the model’s variables. The adversary is usually presented explicitly as an actor, and their targets may be explicitly modelled by the formalisation means of game or decision theory. This family of models usually operates on a higher, system-wide level. Existing methodologies may differ depending on the modelled variable assets and system, but at the same time, the operation research framework (and game theory in particular) provides a high-level formalism, which may be utilised in order to support the decision making during SRA for large-scale systems. Additionally, an explicit definition of the gain function simplifies the interpretation of the results. Operation research serves as a basis for the development of the static model in this thesis.

- Family of statistical graphical models [13]: This can be understood as a mixture of statistical inference and operation research, despite being formally related to the first one. They do not suffer from the limitations of data mining models, and they are widely applied in safety tasks for the identification of sub-system failures [4], and well as to promote cybersecurity [72]. In contrast to data mining models, statistical graphical models have a preliminary design phase, which seeks to capture the specifics of the system described. The models state an explicit probabilistic dependency between the data observed during operation and a high-level descriptive system state (e.g., a failure of specific modules or possible attack presence). Therefore, graphical models make it possible, for instance, to derive the probability distribution of the system-state variables depending on the data observed. Graphical models, including the Bayesian filter model, were used for the definition of the dynamic model in this thesis.

It is worth mentioning that, in recent years, the abovementioned separation between operation research and data mining has become increasingly flexible. One of the main reasons for this relates to the emerging area of generative adversarial networks (GANs) [31]. GANs can be regarded as a combination of operation research methods and data mining. Some theoretical setups model a zero-sum game between the generator and the classifier [51], the intention being to teach the generator to produce objects that are indistinguishable from the “real" training dataset. However, this does not set an explicit target to model a behaviour between the attacker and defender, and it does not solve the problem of poor interpretability.
In addition to the methods already discussed, there exist a variety of approaches for addressing generic security problems, including generic attack detection methods, as presented in [52]. The paper proposes a framework for the development of a dynamic model for generic attack detection. However, the proposed model is noted linked to the standard “manual" SRA methods. The generic optimisation problem for security control

selection is stated in [11], but on a high level and with no details of the design of the system. A generic data mining framework was proposed in [60] for cyber-attack detection, but it is not linked to the SRA and is limited to a specific set of attacks.

2.2.2 Model Family Selection

Presented families of methods are applied in operation along with classical methods for security risk analysis. The data mining models are usually used at a low level, enhancing security on a per-component basis, and usually excluding the higher-level picture. Considering that the present thesis sought to explore decision support for SRA in large-scale systems, operation research and statistical graph models were selected as the main frameworks. Game theory is used, in particular, because it is a common mechanism for attacker-defender models, as discussed in greater depth in due course. Additionally, game theory provides a mechanism that allows the problem to be analysed, and which permits the creation of strict derivation of the model to specific mathematical problems (usually discrete). In turn, these can be further optimised using well-known techniques, ranging from classical gradient-based algorithms [14] to swarm optimisation or neural network-based solution inference [26]. Graph models and game theory could be used as a tool for in-operation analysis and a high-level aggregation mechanism for the captured generic data stream produced by the system. In this thesis some statistical learning elements are applied, but only as a composite part of a general model, and mostly due to the specifics of the selected dataset.

2.3 Game Theory

Game theory is a sub-field in operation research, which involves the study of mathematical models of conflict and cooperation between intelligent, rational decision-makers. Game theory has a wide range of applications in economics, political science, psychology, computer science, and – as is relevant for the present thesis – the domain of security.
A classical two-player game is defined by a tuple $\Gamma = \langle X_1, X_2, F_1(x_1, x_2), F_2(x_1, x_2) \rangle$, where X_1, X_2 are the strategy domains for the first and second player, and F_1, F_2 are real-valued gain functions. Each of the players aims, for instance, to maximise their gain. The multiplayer game is formalised as an extension of the 2-player game definition. The classical problem in game theory is a Nash equilibrium search. A tuple of strategies (x_1^*, x_2^*) is called a Nash

equilibrium if and only if:

$$\begin{aligned} \max_{x_1} F_1(x_1, x_2^*) = F_1(x_1^*, x_2^*) \\ \max_{x_2} F_2(x_1^*, x_2) = F_2(x_1^*, x_2^*) \end{aligned} \qquad (2.3)$$

An intuitive understanding of the Nash equilibrium strategies is that all the players cannot select a better strategy in case the strategy of the rest of the players is fixed. In a general case, it is possible to have several equilibria for one game. When $F_1 = -F_2$, the Nash equilibrium is an antagonistic equilibrium, and Γ is called an antagonistic (or zero-sum) game. This can be interpreted as that the gain of one of the players is considered as a loss of another or that the main intention is to cause maximal damage.

Another important definition in game theory is multi-objective optimisation and Pareto-optimality. These definitions are essential for the case when the gain functions are vector-valued

$$F_i : X_1 \times X_2 \times ... \times X_n \to \mathbb{R}^k, k \geq 2 \qquad (2.4)$$

where n is the number of players. A simple example of a vector-valued gain or impact function may be given by SecRAM impact areas and the CIA triad, which are further accumulated into a single value.

For the cases of vector-valued gain functions, the ranking should be explicitly defined. A key concept in multi-objective optimisation is Pareto-optimality, which allows for partial ranking of vectors. The definition of a Pareto-optimal point is given below.

Definition: Let $S \subset \mathbb{R}^k, k \geq 2, \forall x \in S, \quad ||x|| < \infty$, where $||x||$ is any norm defined in $\mathbb{R}^k$. Then $p \in S$ is a Pareto-optimal vector if and only if $\nexists x \in S : x \neq p, \quad \forall i \leq k \quad x_i \geq p_i$.

Depending on the specifics of the problem, inequalities in the definition could be changed. The definition given above is relevant for maximisation problems, while for minimisation problem the condition may be changed to $\nexists x \in S : x \neq p, \forall i \leq k \quad x_i \leq p_i$.

As will be shown, some examples of game theory models set up a task to find the set of all Pareto-optimal solutions, which is sometimes defined as a Pareto-optimal frontier. An illustrative example of Pareto-optimal points in two-dimensional space is given in Figure 2.3.

2.3.1 Game Theory Models in Security

Game theory modelling has grown rapidly within the domain of security. Game theoretic models represent interactions between opposing agents. Typically, one player is a defender while the other agents are adversaries. The assumption is made that the defender will select an optimal strategy (e.g., they will implement security controls), which is considered to show optimality in terms of a variety of possible solution concepts: antagonistic equilibrium, Nash equilibrium, Pareto-optimality, and others. [16, 63].

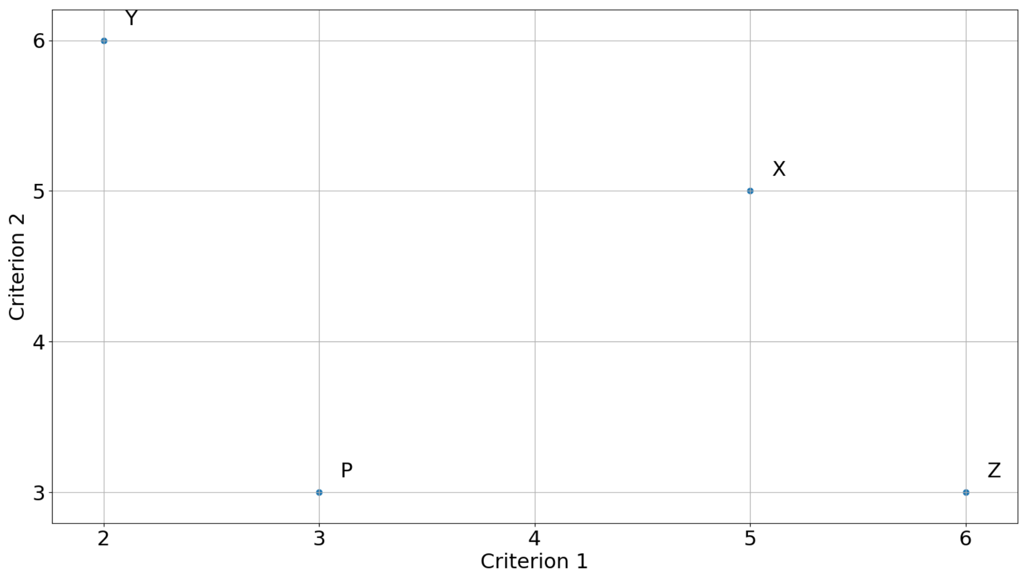


Figure 2.3: Example of the Pareto-optimal frontier in 2-dimensional space of criteria 1 and 2 for a set of four points. Points X, Y, and Z are Pareto-optimal according to the definition; they form a Pareto-optimal frontier. Point P is not Pareto-optimal, as point X is greater at each component.

A classic example of an application of mathematical modelling grounded on a Bayesian Stackelberg game is the ARMOR family of systems [54]. One system is deployed at Los Angeles airport for solving the randomised patrol scheduling problem with limited resources. Furthermore, game theoretic models have been applied in various areas by the US coastguard [59], air traffic security [55], and police patrol [17].

The domain of cybersecurity is more challenging for the application of game theoretic models. Normally, security mechanisms in the cyber-domain are based on techniques from fields such as encryption [61], statistics [18], and others. Some papers have explicitly justified why some of the standard approaches are not applicable to this domain. For example, [32] described game theoretic models that are relevant to information warfare. Their paper analysed several situations and outlined different courses of action with predicted outcomes and what-if scenarios. In their work, the researchers analysed standard tree-based decision-making, and they justified why – for the case of information security – the zero-sum game might not be an appropriate solution. Based on this fact, as well as other reasons, the paper concluded that standard tree-based game models are not directly applicable to the domain of cybersecurity.

At the same time, [23] provided a comprehensive survey of different applications of game theory in security for specific tasks, which include methodologies for secure routing, addressing denial-of-service (DoS) attacks (i.e., detection and counter-action strategy), and so on. In [19], the researchers considered the Internet and its infrastructure as the basis for exploring attacks and security issues. While the majority of studies have focused on securing data transfers, the researchers discussed attacks on infrastructure, which can lead to significant destruction due to the fact that different components of existing Internet infrastructure have various trusting relationships with one another. The study of [46] presented an ontology for distributed DoS attacks and protective methods. The mechanisms classify the attack and defence strategies. Furthermore, the researchers highlighted common attack features and important types of attack strategies, thus putting additional structure to the attack and defence description.

However, each of the models focuses on case-specific details and attack types, with no possibility to support generic SRA-type decision-making. SRA methodologies are well-established and provide a level of abstraction that is suitable for general-case risk assessment. However, existing standards do not provide any mechanism for strict mathematical quantitative assessment procedures. This fact restricts the possibility for automatic “what-if" studies, and it limits decision-making processes to expert-based judgements.

In the next subsections, the most successful game theory-based approaches to SRA are detailed, and the network security game framework is introduced.

2.3.2 Examples of Game Theoretic Models in Security

One definition of game theory states that it is a mathematical theory for decision-making in conflict situations [67]. As previously noted, game theory is viewed as a sub-field in operation research, which intuitively leads to the assumption that it should be applicable to the domain of security by modelling two opposing agents. Some trivial models (which are not related to game theory) describe processes relating to the issue of how decision-makers select actions from predefined sets of actions in order to optimise a given cost function, which depends on the selected strategy. However, such models do not describe conflict situations. The conflict situation differs because the decision is taken by multiple decision-makers (or players). Each player’s cost function may depend on the strategies selected by other players as well. Such a mathematical model is known as a game.

The standard formalisation for a multiplayer game is given as follows. Let A denote a set consisting of players. Each player $a \in A$ operates according to a set of strategies S^a. Each player selects a strategy without any knowledge about the strategies selected by the other players (this statement may vary depending on the model). Thus, it is possible to define a strategy tuple $s = (s^a | a \in A) \in S = (S^a, a \in A)$, from the Cartesian product of all strategy sets for all players. For each player, a gain or cost function is defined $u^a(s) : S \to \mathbb{R}$. Players select their strategies to optimise their corresponding gain functions. The game is defined as a tuple:

$$\Gamma = (A; S; u^a(s), a \in A) \tag{2.5}$$

One of the most well-studied cases is the “game of two players" (i.e., the case where $|A| = 2$). Application of game theory to the domain of security usually operates for games with two players, where one of the players is the adversary (or attacker) and the other is the defender. Furthermore, for purposes of unification, the following notation is used for security-related games with two players: $A = a, d$ is a set of players; d is the defender, referred to from this point onwards as “she”; and a is the attacker, referred to as “he”. An important type of game is the “zero-sum” game involving two players, where $u^a(s) = -u^d(s)$, and where each player is trying to maximise their corresponding cost function. Zero-sum games are often reduced to settle point search problems. For these types of games, it is possible to assume that the attacker and the defender have the same cost function, but one of the players tries to maximise it while the other tries to minimise it.
Several basic theoretical models were described in [5]. This work gives a security domain interpretation of the main theoretical properties of the games. The models define a cost function, its dependence on the adversary’s adopted strategies, and the defender’s response to the attack. The associated set of strategies and systems performance are represented as a set of equations and constraints for the corresponding optimisation problem. Next three examples demonstrate the the application of the game theory to a simplified pipeline protection problem. They show several important ways for defining the characteristics of the game like sequence of actions of the agents, protection formalisation, gain function definition. All examples consider a pipeline structure, which is transferring oil from the source to the sink. This pipeline structure is a subject of an attack from the attacker. The goal of the defender is to select a counter-measure that would optimise the cost the damage caused by the attacker.

Attacker-Defender Model. The adversary attacks first, the defender receives full information about the ongoing attack, and – accordingly – the defender responds. The attacker initially has full knowledge about the defender’s optimal response to every attack strategy. The attacker’s strategy is to disable some subset of the pipes and the defender’s response is to reorganise the oil flow within the rest of the pipes. The following optimisation problem is stated in order to calculate the resulting cost function for the defender (assuming that both players act rationally):

$$\begin{cases} \max_{x \in X} \min_{y \geq 0} \langle c, y \rangle \\ Ay = b \\ y \leq U(1 - x) \\ \langle x, 1 \rangle \leq M \end{cases} \tag{2.6}$$

Here, y represents the vector of flows in every pipeline (i.e., the defender's strategy); U is a diagonal matrix where each element u_{ii} represents the capacity of pipe i, amount of oil that could be transferred at a moment of time; x is a Boolean vector of enemy strategies, indicating which pipes he is going to disable; M limits the number of pipes that the attacker can disable; $\langle x, 1 \rangle$ defines a dot product of the vector x with a vector with all elements equal to 1, resulting into a sum of all elements of x; structure limitations are given by $Ay = b$; and c represents the vector of relative costs of passing a unit of the flow through a given pipe. Thus, the term $\langle c, y \rangle$ estimates the cost of flow distribution y. Some of the structural limitations are defined as inequalities, for example, the term $\langle x, 1 \rangle \leq M$ defines the limitations for the number of non-zero entries to the vector x, which is interpreted as the maximal number of pipes damaged by the attack. Inequality $y \leq U(1 - x)$ states that for each pipe i the flow y_i does not exceed the pipe capacity $u_i i$ if the pipe is not attacked, and is equal to 0 if the pipe is attacked. Equality $Ay = b$ defines the limitations such as that the inflow of each pipe should be equal to the outflow. In a general case, structural limitations could be augmented by inequalities, for example, if there is a lower bound for the inflow for some of the pipes. The defender tries to minimise the cost for each given adversary strategy x. This is done by selecting a y that minimises the cost function. Thus, the inner "min" operator represents the defender's response for a given attack x. The attacker's optimal strategy involves selecting such an attack so that the cost will be maximized, given the optimal response of the defender. The outer "max" is the selection of the optimal x with a known response, where the attacker's aim is to maximise the cost in case of an optimal response. Despite being a zero-sum game, no attempt is made to search for any type of equilibrium. This is done due to the fixed order of the agents' actions in the game. The model gives a good representation of how the sequence of "min/max" operators is linked to the sequence of actions, as well as demonstrating an example of optimal counter-action selection. In addition, the model shows how it is possible to define the structural limitations and attack impacts.

Defender-Attacker Model. For this model, the action sequence is exchanged. The defender prepares using known information about all possible attacks, after which the attacker strikes. The strategy of the defender is to put a protection for some of the pipes, so that they can not be disabled by the attacker. The strategy of the attacker remains the same as in the previous example. A formal description of the model is given below. In terms of what the main notations are, these are listed as follows:

- The defender's assets are indexed by k.
- c_k denotes the gain of the attacker in case of an attack against unprotected asset k.
- p_k denotes a reduction of the attacker's pay-off derived from attacking asset k if that asset is defended (i.e., the adversary receives a pay-off $c_k + p_k, p_k \leq 0$).

- P denotes a diagonal matrix with p_k at position (k,k)

The strategy encoding is given as follows:

$$y_k = \begin{cases} 1 & \text{if asset } k \text{is protected} \\ 0 & \text{otherwise} \end{cases} \tag{2.7}$$

$$x_k = \begin{cases} 1 & \text{if the asset } k \text{is attacked} \\ 0 & \text{otherwise} \end{cases} \tag{2.8}$$

Formulation of the cost evaluation in the case where both players select the optimal strategy is given as follows:

$$\begin{cases} \min_{y} \max_{x} \langle c + Py, x \rangle \\ Ay = b \\ Gy \leq f \\ Cx \leq c \end{cases} \tag{2.9}$$

where x corresponds to the attacker's strategy, and y corresponds to the defender's strategy. Equality $Ay = b$ represents structural limitations of the pipeline, $Gy \leq f$ represents the defender's constraints in terms of resource limitations, while $Cx \leq b$ is the corresponding set of constraints for the attacker. Constraint inequalities can be considered as generalised version of the limitations of form $\langle x, 1 \rangle \leq M$ from the previous model example.
In contrast to the previous model, the current model considers preliminary protection of the system, where the concept of security controls is implicitly introduced. In addition to that, the model demonstrates the sequence of actions considered in SRA, where often a static protection is selected before an attack happens.

Defender-Attacker-Defender Model. The game considers a three-step action sequence: the defender prepares, the attacker strikes, and the defender responds. Using the same notation as in the defender-attacker example, the underlying optimisation problem is stated as follows:

$$\begin{cases} \min_{w} \max_{x} \min_{y} \langle c, y \rangle \\ Ay = b \\ y \leq U(1 - (x - w)_+) \\ \langle x, 1 \rangle \leq M \end{cases} \tag{2.10}$$

where w is a Boolean vector related to the protection, placed by the defender at the first step of the game (outer 'min' operator). If some pipe is protected then the corresponding attack against it is blocked. After that, Attacker-Defender game is performed. A pair (w, y)

represents the defender's strategy, while x is the attacker's strategy. The operator used at the term $(x-w)_+$ is defined as follows:

$$(a)_+ = \begin{cases} a & \text{if } a > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.11}$$

This example shows how game theory allows the modelling of multi-step dynamic situation development.

The three models described above give an important yet preliminary understanding of how game theory allows some parts of the SRA to be performed in certain systems. Additionally, it shows how differences in the sequence of actions undertaken by the attacker and defender can be formalised as sequential per-component optimisation problems. The number of steps performed by the adversary may increase in a similar manner, as shown in the defender-attacker-defender model. It is a commonly used approach, and it is typically used in multi-step games such as chess. An example of a security application of multi-step games is described in [41]. Usually, in such cases, the number of strategies for both of the players is finite and not large, which means that dynamic programming can be used to solve the corresponding problems.

An interesting consequence of the game theory formalism is one that proves that, in case of a zero-sum game with a gain function, it is always better to respond to a known type of attack rather than adopting a general “worst-case" action. This is formulated in an implicit manner in the following inequality, which holds for any function F:

$$\max_d \min_a F(d,a) \le \min_a \max_d F(d,a) \tag{2.12}$$

where d denotes the defender's strategy, and a denotes the attacker's strategy. However, in the defender-attacker action sequence, this is better for the provision of the general case “static" protection, while the attacker-defender strategy is the best choice in terms of counter-response.

2.3.3 Los Angeles Airport Randomised Scheduling

The ARMOR system is a real-world application of a game theoretic model in the field of security [54]. The system was deployed at Los Angeles airport, and it is still used for security patrol schedule randomisation.

The Bayesian Stackelberg game (BSG) is used as a basis modelling scheme. A classical Stackelberg game (SG) is conducted between two players: the leader, the first player, and the follower. The leader selects a strategy first a_l, after which the follower performs an action a_f, having full knowledge about a_l. In the scope of security games, the leader (defender)

defines her strategy (security system), and then the adversary attacks (under the assumption that he knows everything about the security system) by selecting an optimal attack strategy. The corresponding mutual state is known as Stackelberg equilibrium. BSG differs from SG in the way that the leader is allowed to select *mixed* strategies.
Mixed strategy for a player is defined as a probability distribution over the set of (pure) strategies of the corresponding player. The gain function for mixed strategies is defined as the expected value of the gain over the strategies of the player.
The main difference between ARMOR and the models that were described in the previous section is that multiple types of attackers are taken into consideration (i.e., there are multiple types of followers). Additionally, the pay-off functions of the attacker and the defender are not opposite, and so the game is not a zero-sum, antagonistic game. These details give rise to a far more complex motivation model for the adversary than is the case in any of the previously explored models. However, the defender knows about the attacker's pay-off functions, and vice versa, which implies a full-information security game. Prior probabilities over the attacker type are introduced in order to calculate the defender's expected pay-off.
Security configuration (i.e., the system) is represented as a placement of a limited number of patrols at checkpoints. To introduce uncertainty, the patrol schedule is randomised (i.e., stochastic strategy selection). The defender's strategy is represented as a probability distribution over the subsets of checkpoints which are to be protected. Therefore, a probability distribution over subsets of checkpoints is optimised in the ARMOR system. The scheduling plan is made by the machine, which selects a random configuration of checkpoints with a given probability. The defender's strategy is a probability distribution over the checkpoints, which is a *mixed strategy*.
The strategies are solved by using the dual formulation of the game optimisation problem. Consequently, the following mixed-integer quadratic optimisation problem must be solved:

$$\begin{cases} \max\limits_{q,x,a} \sum_{x \in X} \sum_{j \in Q} R_{ij} q_j x_i \\ \sum_i x_i = 1 \\ \sum_j q_j = 1 \\ 0 \leq a - \sum_i C_{ij} * x_i \leq (1 - q_j) M \\ x_i \in \left[0, 1\right] \\ q_j \in \{0, 1\} \\ a \in \mathbb{R} \end{cases} \tag{2.13}$$

where x and q represent a probability distribution vector over the set of checkpoints for the defender and adversary, respectively; R_{ij} is the defender's reward function in the case where the adversary selects a checkpoint j to attack, while the defender has allocated a patrol to i.

It is also notable that pay-off functions for the attackers and defender are created by experts. The version described above encounters only one type of follower (attacker). However, in the next versions of ARMOR, multiple types of followers were considered with different prior probabilities, thereby transforming the optimisation problem into the following:

$$\max_{q,x,a} \sum_{l \in L} \sum_{x \in X} \sum_{j \in Q} p^l R^l_{ij} q^l_j x_i \tag{2.14}$$

where L denotes the set of follower types, and the strategies for each attacker are marked by corresponding l. Prior probabilities p^l are defined by experts. In order to generate a randomised checkpoint schedule for k patrols, a tuple with no repeating samples is generated from the distribution defined by x.
The ARMOR example is important not only because it shows how game theoretic models can be applied to real-world problems (in this case, in the area of air traffic management (ATM) security), but also because – from a theoretical perspective – it introduces an important concept of attacker types, as well as a specific way to manage them. As such, this partially addresses the point about attack generality mentioned in Section 2.1.4.

2.3.4 Multi-objective Security Models

Another notable family of models in the context of security games is known as multi-objective (MO) models. The MO formalism allows an evaluation to be made of the different types of impact, or the different types of attacks, that exist against the system simultaneously. This approach makes it possible to avoid the process of “averaging" diverse types of impact (e.g., economic, reputational, or human losses) into a single pay-off function, and instead considers these values independently.
In several articles [54, 59], various types of adversaries are introduced, and different pay-off functions are created to specify the corresponding types of adversaries. To avoid averaging the pay-off (or impact) based on the type of adversary in the SG, each type of is considered independently. The following entities are defined:

- $U^d_i(c,a)$ – The pay-off of a defender in case of a game with attacker type i, security strategy c, and attack a.
- $U^a_i(c,a)$ – The pay-off of an attacker for the same configuration.

Therefore, in case of SG, the following definition can be given:

$$A_i(c) = Argmax_a U^a_i(c,a) \tag{2.15}$$

where $A_i(c)$ is a set of optimal strategies of the adversary type i in case of security system c. Let us consider the following MO optimization problem for k different types of attackers:

$$\max_c \big[\max_{a_1 \in A_1(c)} U_1^d(c, a_1), \ldots, \max_{a_k \in A_k(c)} U_k^d(c, a_k) \big] \tag{2.16}$$

Here, $\max_c$ should be understood as Pareto-optimisation (i.e., the search of all Pareto-optimal strategies). Several algorithms have been proposed in the literature to solve problems of this kind [15].

The outcome of MO optimisation is usually defined as a set of optimal solutions, which belong to a Pareto-optimal (or Slaiter-optimal) frontier. The complexity of the merging process for multiple functions is transferred to the process of selecting the optimal solution among the set of PO solutions, as well as their corresponding pay-off functions. It can be seen that the proposed methodology builds a model resembling the defender-attacker approach, as it is assumed that the adversary acts after the defender strategy is selected.

MO security games are important because they provide the following: firstly, a regular approach for merging different types of impact, which is frequently undertaken in an artificial manner (as in SecRAM, where only a maximum is taken); and secondly, an implicit assumption of multiple adversary types, as each of the impact types is optimised independently and each defender solution a corresponds to a set of adversary solutions c.

2.3.5 Network Security Games

The domain in a network security game (NSG) is defined by a graph, which encodes the set of actions and strategies of the players. Research into NSG covers a broad domain encompassing applications in nuclear smuggling interdiction [48], assignment of checkpoints to urban road networks [37], fare evasion minimisation in public transport systems [22], and others. In addition, the NSG may present a specific interest as it provides a flexible framework for threat definition, which can help to address the problem of attack generality noted in Section 2.1.4.

NSG is modelled using a graph, denoted as $G = (N, E)$. The adversary begins an attack at any of the source nodes $s \in S \subset N$, and selects a path in the graph to any one of the targets $t \in T \subset N$. The set consisting of an attacker's pure strategies are all the possible s-t paths from any source $s \in S$ to any reachable target $t \in T$. The defender attempts to block the attacker by placing k available (homogeneous) controls on edges (or nodes) in the graph. Each of the resources blocks the adversary if his path passes through the node. Therefore, the pure strategies of the defender consist of all possible allocations of k controls within the edges, amounting to $\frac{|E|!}{k!(|E|-k)!}$ in total. In the case of sufficiently large graphs, this number may be too large for a brute-force solution to be viable. Assuming that the defender plays allocation $X_i \subset E$, and also that the attacker chooses path $A_j \subset E$, the attacker succeeds if

and only if $X_i \cap A_j = \emptyset$. Each of the targets t has an associated pay-off $I(t)$, such that the attacker receives $I(t)$ for a successful attack on t, and 0 otherwise. NSG is zero-sum, and so the defender receives $-I(t)$ in case of a successful attack on t, and 0 otherwise. The NSG domain is modelled as complete information game. Thus, the set S of sources, T of targets, and the pay-offs I for all targets and the number of defender's resources k are assumed to be known for both players.
The problem may be formulated as mixed-integer linear programming in the following way.

$$\begin{cases} \min_{D,p} z \\ \forall (n_i, n_j) \in E \quad p_j - p_i \leq D_{ij} \\ \forall t \in T \quad \forall (n,t) \in E \quad z \geq (1 - p_t) I(t) \\ \forall s \in S \quad p_s = 0 \\ \sum_{i,j} D_{ij} \leq k \end{cases} \tag{2.17}$$

The variables have the following meaning and range of values:

- $D \in \{0,1\}$ – Variable for each edge in E, which is related to the controls placed at the edges.
- $p \in \{0,1\}$ – Variable for each node in N, which indicates whether the node is “protected" (i.e., not reachable) from any of the source nodes (0 corresponds to unprotected, and 1 corresponds to protected).

2.3.6 General Remarks on Game Theoretic Models

The set of models presented throughout this section shows the possibility of formalising the actions of the adversary and the defender for a specific domain. The order of actions, restrictions of the domain, and the agents' abilities are encoded in the final optimisation problem. The order of actions between the defender and attacker are shown in an elegant way by changing the order of the optimisation operators. In terms of the analysis of the relationship between SecRAM and game theory, the security controls are selected based on threat analysis, which itself relies on an understanding of the calculated type of impact. Therefore, it may be assumed that SecRAM considers the defender-attacker type of actions. This may not be the case if some of the controls are assuming any type of counter-action that depends on the attack type. However, the type of counter-action could be encoded into the calculation of the mitigated impact.
Another important point to recognise is that, in ARMOR, the types of adversaries are separated, which provides greater flexibility and scalability for the model in terms of adding new “threat scenarios". To take a novel type of threat into account, new impact and gain

matrices should be defined, as well as the corresponding prior probabilities. However, the domain in which ARMOR operates does not provide any flexibility because the only possible output is a mixed strategy for the set of endpoints.
MO optimisation encapsulates two important real-world features: one is the type of adversary, as mentioned before, and the second is the diversity of the types of impact. It is notable that the second point is explicitly used in SecRAM.
Finally, specific attention should be paid to the NSG game type because the model of the scenario is significantly more complex and flexible compared to the previously mentioned models. Graph-based representations are used to represent attack actions in several domains. Additionally, graphical representation is efficient for discrete-scenario multi-step security games, especially in the context of information security. Therefore, special attention will be paid in the next sections of this thesis to the graph-based representation of the scenarios.

2.4 Probabilistic Graphical Models

A probabilistic graphical model (PGM) is a probabilistic model for which a graph defines the conditional dependence structure between random variables. PGMs use a graph representation as a descriptor in order to encode a probability distribution over a multi-dimensional space, where the graph nodes define the random variables and the graph arcs define the dependencies between subsets of variables. Two different types of graphical models are commonly used: Bayesian networks and Markov random fields. Both families encompass the graphical representation of interdependencies, but they differ in terms of the way they define the corresponding probability distribution. Given the graph $G = (N, A)$, the corresponding rules for the definition of the probability distribution include the following:

- Bayesian networks are defined by a directed and acyclic graph, where the directed edges illustrate causal relationships between variables in the network. The probability for a set of variables (or nodes) follows the following factorisation rule: $P(x_1, \dots, x_k) = \prod_{x_i} p(x_i | Pa(x_i))$, where the operator $Pa(x_i)$ defines the set of direct “parents" of the node x_i (i.e., the nodes that have edges ending in x_i).

- Markov random fields are defined by an arbitrary graph, and the interdependency is defined in such a way that, for a set of variables (or nodes) $\{x_1, \dots, x_k\}$, the following holds:

$$p(x_i | x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_k) = p(x_i | x_j, j : (i, j) \in A \quad or (j, i) \in A) \tag{2.18}$$

  Each variable is conditionally independent of the non-adjusted nodes, given the adjusted nodes. No explicit causal relation is given.

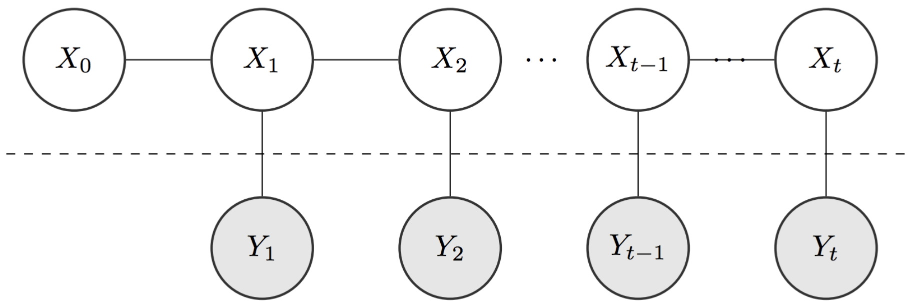


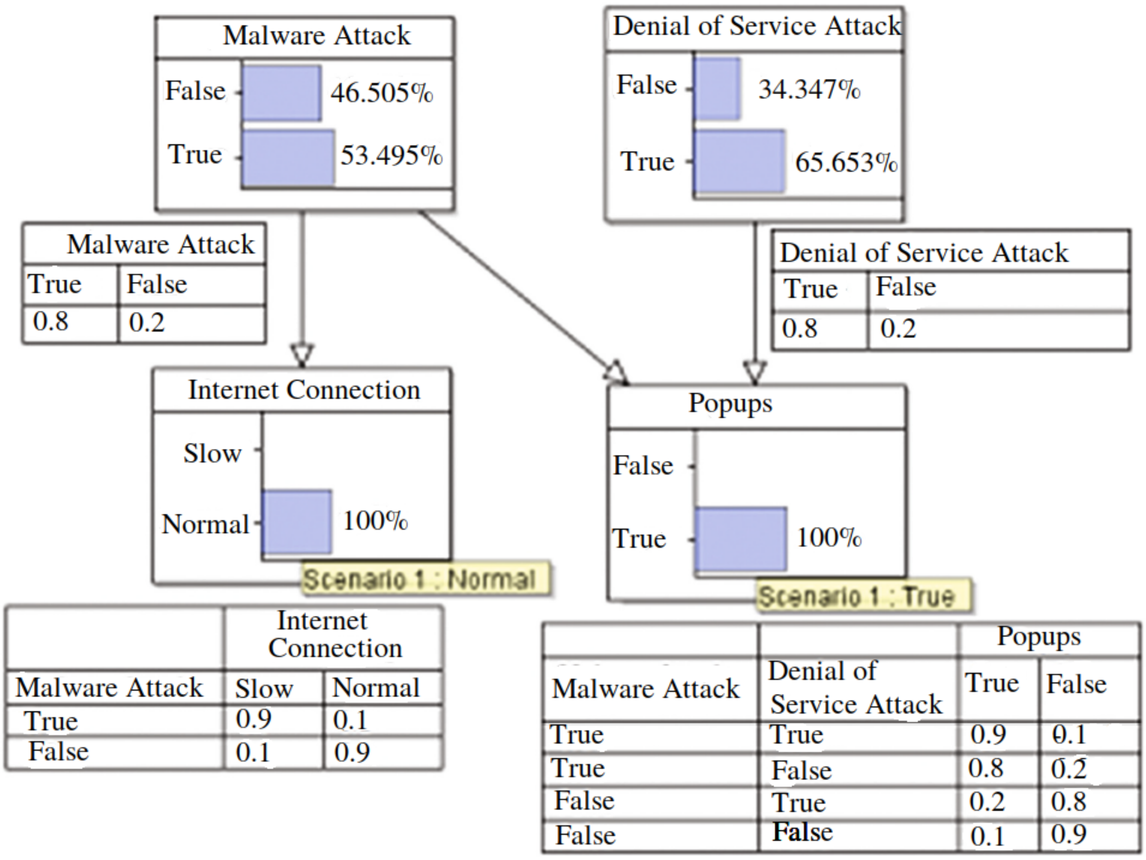
Figure 2.4: Probabilistic graphical model of Bayesian filters, where variables X_i are hidden variables andY_i are visible variables.

As mentioned in the motivation section of this thesis, the proposed framework intends to address the SRA process not only at the stage of system design but also at operation, where temporal data processing is required. From this perspective, Bayesian filters (BF) present an interesting topic for consideration. BFs are a broad family of Bayesian network models, which are applicable in the context of temporal data processing. This family includes the well-known Kalman filters (KF), extended KF, particle filters, unscented filters, hidden Markov models, and others. Each of these models relies on a different computational methodology, but the following two properties are shared across all:

- The models incorporate two types of variables: “visible" variables, which are directly observed by the user; and “hidden" variables, which are dynamically and temporally connected, and which condition the visible variables. The main problems stated for BF models are converged to an inference of the hidden variables, given some part of the visible variables.
- The causal scheme between the hidden and visible variables always follows the graphical representation shown in Figure 2.4. The hidden variables form a Markov Chain time series (depending on the discrete variable t), while the visible variables are connected to the corresponding hidden variable.

2.4.1 Applications in the Domain of Security

This subsection focuses on the application of PGMs in the domain of security. As previously noted, an essential property of PGMs is the possibility they afford to estimate the internal, hidden variables of the system, given the observed data. A toy example of the application of Bayesian networks (BNs) to cybersecurity is given in Figure 2.5, as originally presented in [21]. In the example in Figure 2.5, conditional probabilities are defined for observing “poor connection" and “pop-up" events depending on the presence of denial of service (DoS) and



Malware Attack	
True	False
0.8	0.2

Denial of Service Attack	
True	False
0.8	0.2

	Internet Connection	
Malware Attack	Slow	Normal
True	0.9	0.1
False	0.1	0.9

		Popups	
Malware Attack	Denial of Service Attack	True	False
True	True	0.9	0.1
True	False	0.8	0.2
False	True	0.2	0.8
False	False	0.1	0.9

Figure 2.5: Example of a Bayesian network model.

malware attacks. Given the observations and Bayes' theorem, it is possible to derive the conditional probabilities of the presence of any of the given attacks.

It is possible to derive and apply models that are significantly more complex, as described in [21]. Furthermore, these models can be applied in various areas, including data breach detection, evaluation of critical software vulnerability exploitation, and quantitative assessments of network security attacks, DoS attacks, or intrusion detection. The software exploitation model [35] defines a model that assesses the time required for a vulnerability of a software package to become exploitable given some set of parameters (e.g., the language used, availability of source code, and development practices in place). Proposed quantitative assessments of network security often rely on compliance with given best-practices and organisation's rules for the mitigation of specific attacks, which limits the generality of the application of the model and does not take into account temporal development during online operation.

In [27], the authors introduced the concept of a dynamic Bayesian attack network, wherein the temporal component is considered in a Markovian manner, similar to the BF domain. The model relies on the concept of attack graphs, as discussed in [7], to provide a description of the threat (albeit one that is narrowed to the domain of cybersecurity), and it builds a BN upon this model. The model considers how the threat and possible exploits evolve over time, thus presenting an evolving mechanism for the assessment of the network security. Noteworthily, the paper uses a generic mechanism for threat modelling, which is designed specifically for the domain of cybersecurity. However, the paper does not suggest a method for in-operation data analysis.

The set of models dealing with streaming data processing and attack detection are mostly limited to a narrow domain of attacks. BFs are widely used in security operations, but mostly as sub-components of radar or video-tracking systems [70], and not as generic methods for streaming data analysis. In [52], a dynamic model, similar to the Kalman filter, was used to solve the generic cyber-attack detection problem given the observed system-related data.

2.5 Attack Trees

Attack trees [57, 44] are graphs that represent how an asset might be attacked. Originating from cybersecurity, they are used to describe threats on computer systems and sequences of actions required to realise those threats. They are applied in various range of different domains, including aviation and military.

Attack trees are directed tree graphs consisting of one designated root. Each node of the graph has its own label, which defines some condition that might be in a “true" or “false" state. In order for a condition to become “true", at least one of the conditions of the child nodes must be “true". When the root condition is satisfied, the attack is complete.

As it follows from the definition of the attack tree models, the formalism models the “or" behaviour: parent node (condition) is expressed as a disjunction of the conditions of the child nodes. It is possible to model “and" conditions as well bu re-labelling the nodes. For example, if a parent node is satisfied when conditions $c1$ and $c2$ are present, it is possible to add a child node with a condition $c1\&c2$. Another possible option is to explicitly describe the “behaviour" of the connections for each node. The connectivity of the child nodes could be denoted as “and" explicitly. By default, an “or" statement is assumed. An example of an attack tree is given in Figure 2.6.

An attack within attack trees methodology is presented as a sequence of satisfied conditions, which makes a direct link to SRA methodologies. Attack trees could be used within SRA standards as explanatory models, for example ISO 31000 explicitly suggests the utilisation of such models for different threat scenarios.

Attack trees are often used to quantify or qualify the existing threats. For example, all leaves of the graphs are quantified with the guess of the “cost" of the condition for the adversary. This allows for the estimation of the “cheapest" attack. It is possible to assign labels to the nodes, for example, if the satisfaction of the node requires special equipment. In addition, the concept of security controls can be introduced to the attack trees. Controls are

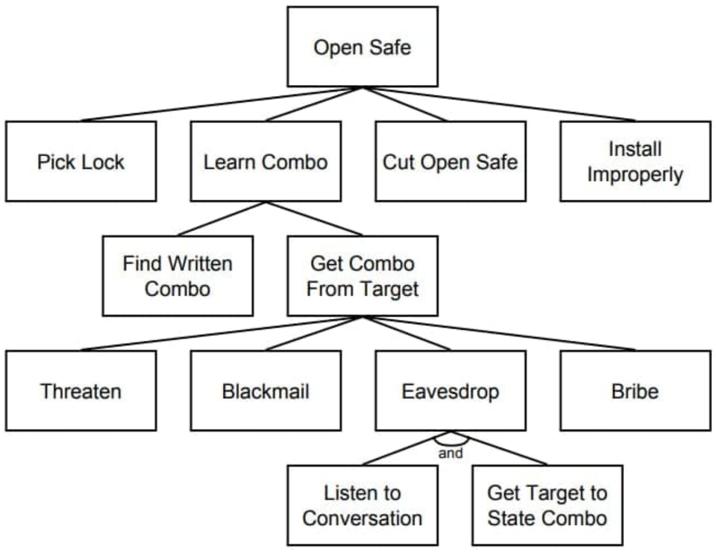


Figure 2.6: A simple attack tree against a physical safe. To open the safe, the attacker can pick the lock, learn the combination, cut open the safe, or install the safe improperly so that they can easily open it later [57]. Examples of “or" and “and" conditions are given.

breaking the dependency of a subset of nodes with a subset of their child nodes. This may be expressed as “blocking" of some of the edges in the graph. Often, an introduction of a security control might require an introduction of additional nodes into the tree. For example, password authentication on a computer might require an introduction of a node with a “break password" condition.

One of the important features that make attack trees useful is that they capture knowledge in a reusable form. It is possible to re-use the completed trees in different circumstances, augment them, and apply them again at different iterations of the SRA process.

Attack trees are going into deep details of the system as this is needed to create a decent description of the sequence of conditions required for an attack to happen. It is possible to automatise the decision-making using attack trees, for example, protection selection. Automatisation is done by using databases of pre-developed attack trees with best-practice control suggestions. At the same time, the scope of the analysis of the attack trees is limited to the event of the attack, with no consideration of the impacts, main assets, and other relevant parts of the SRA process. For this reason, attack trees can be used as a supportive mechanism for decision-making at the threat (risk) analysis parts of SRA.

2.5.1 Comparison with Network Security Games

Attack trees and NSGs are sharing the instantiation methodology because they are based on graphs. Attack trees with “or" links may be topologically translated to NSGs in a straightforward way using the following steps.

- Each leaf is considered as a source node.
- Root node is considered as a sink.

In this case, each “parent" node is accessible if and only if the attacker managed to reach one of the “child" nodes. This directly translates to the attacker’s movements within the graph, as it is encoded in NSGs. However, translation of an attack tree with “and" condition is not straight-forward and involves modification of the original NSG. An approach for the introduction of an “and" statement analogue to NSGs is covered in the next chapter of the thesis.

However, NSGs are missing several important aspects of the attack trees and their development process. NSGs do not explicitly specify for each node of the graph its physical meaning, relation to the actions of the adversary, cost of each movement for the adversary. In general, it is possible to conclude that NSGs lack “annotation" for their nodes and edges. Of course, an introduction of this information does not contradict the instantiation of the model. However, this type of annotation of the graph is not considered as part of the standard NSG model, and the process of the development of the NSGs is not formalised in any manner, opposite to attack trees.

At the same time, it is impossible to map the NSGs to attack trees, as topologically, NSGs are much more reach. NSGs may contain multiple sinks, corresponding to various attacks against different assets. They may represent the situation when a single condition may serve as an enabler for a number of possible threats. The underlying graph is not restricted to be a tree; it may contain cycles. However, NSGs may not be used for the evaluation of the threat cost in the same manner as attack trees. NSGs are less illustrative but may capture a broader scope of threats.

The concept of security controls is the same for NSGs and attack trees. NSG concept is slightly different and is trivialising the problem as if all security controls were presented as a “block" of a single edge.

As a conclusion, it is possible to state that NSGs present means for a broader scope of modelling by capturing multiple threats at a single moment of time. Attack trees are much better elaborated in terms of rules for construction, graph annotation, visualisation.

2.6 Graph Models for Cybersecurity Risk Assessment

Graphs are a common mechanism in cybersecurity risk assessment methods. In [20] an extensive overview and categorisation of various techniques for cyber SRA are presented. Some of these methods have certain similarities to the models proposed in this thesis. For this reason, they are described further in this section and compared with the presented method

in terms of context establishment and risk evaluation/mitigation.
The paper describes more than twenty different techniques for cyber SRA, but only a few of them are tailored for system-scale security risk assessment, operating with high-level generic terms. These methods are referred to as “guidelines" or “elaborated guidelines". “Elaborated guidelines" provide a detailed scheme for end-to-end cyber-SRA, usually supported by a graph-based mathematical formalism. These methods are listed below.

- In [8] a technique is proposed for semi-automatic graph-based cyber SRA. The method is based on a pair of graphs: infrastructure graph and (threat) evolution graph. For each of the graphs, a clear construction procedure is given. The attacker’s strategy is presented as a path within the evolution graph, which encodes the evolvement of his “access rights" from the “initial rights" to “target rights". Security controls are able to prune some of the edges in the evolution graph and block some of the attacks. From that sense, controls are implemented in a way similar to NSGs.

- A technique for quantitative risk assessment in supervisory control and data acquisition systems is proposed in [45]. The described method is based on a weighted graph that describes the main actions required to perform the attack. The weights of the edges encode the time required for the attacker to pass from one node to other. Graph construction procedure has an informal, conceptual description. The attack is presented as a path from a “start" node to a “target" node. Risk mitigation means are considered to increase the weights of the edges, thus increasing the time required to perform the attack.

- The network security risk model [33] proposes a method for an assessment and quantification of the risks in process control networks (extendable to cyber SRA in critical infrastructure). The method describes the system by attack trees with and/or junctions and barriers (controls). The system is decomposed into a subset of components, each of the components may be in two different states: compromised by the attacker and non-compromised. The attacker’s action space is presented as a weighted system status graph. The system status is defined as a vector of states of each sub-system, resulting into 2^n states for n subsystems. Each edge in the state graph represents an action of the adversary, associated weight presents the probability of that action. The system risk level is evaluated by solving a stationary equilibrium equation for the attacker’s set of actions. Security controls are changing the parameters of the graph, thus changing the equilibrium.

- The technique suggested in [64] is a cybersecurity level quantification method that uses and/or attack trees and a risk measure called “vulnerability index" (VI). VI computation formulas are derived from the attack tree structure and linked to specific controls considered: port scans and password length policy.

Each of the methods is described in more detail in the following chapters of the thesis and is compared with the proposed methodology.

2.7 Conclusion

After undertaking a comprehensive review of the extant and related literature, it is important to consider the models observed and select the base for further development of the methodology. Classical SRA relies on a high-level analysis of the system considered. However, each of the methods observed has certain common features, which are listed below.

- A stage of the instantiation of the analysis or establishment of the context. During that stage, the main purpose of the analysis is identified and the main properties of the system observed are stated.
- A stage of threat identification. During that stage, possible attacks are described; consequences are analysed and quantified.
- Based on the previous steps, strategies for threat mitigation are selected.
- Optionally, the SRA should be repeated and ensure the monitoring of the system considered.

The first two stages may be mapped to the model instantiation phases of the mathematical methods considered: definition of the gain functions and the set of strategies for game theory, development of the graphical model and joint probabilities definition for PGMs, design of the attack tree, and quantification of the leaves. The third part of the SRA may be mapped to the process of game solution; attack tree analysis and controls selection. PGMs are focussed on implementing a specific mitigation strategy related to attack detection and prediction, which can be considered as a support for the third and fourth stage of the SRA.
Existing mathematical formalisms considered are dealing with the automation of the following stages.

- Security control selection.
- Online data processing and attack detection.

For this reason, the proposed model for general SRA support should be designed in a way that it provides means for automatisation at the “third" and “fourth" part of the general SRA process.
Next, each of the model families is considered against every criterion from 2.1.4.

System description generality. This criterion corresponds to the way how the model is instantiated. Main entities for the system description defined by SRA usually include vulnerable parts of the organisation/system (targets), their quantisation in terms of possible loss (impacts for SecRAM and ISO 27005), estimation of the business and resource limitations for the defender (stakeholder commitment in ISO 31000).
Game theory models define explicitly the set of targets, which could or could not be logically separated from the set of strategies. Consequences estimation is encoded into the gain function of the defender.
Attack trees have limited capabilities for system description, as their target is to describe the threats. Any cost estimation is linked to the particular attack described by the tree. Security controls selection process can be modelled and automatised within attack trees paradigm, where resource limitations can be encoded.
PGMs may encode targets and impacts for the system modelled. Modelling of the resource limitation depends on the context as PGMs do not deal with any direct optimisation (possibly, only within the Bayesian inference process).

Attack description generality. This criterion corresponds to risk identification and description parts of the SRA. The most important property here is that targets within the system are logically separated from the threat description. The standard models for attack descriptions are attack trees, as they are widely used in different fields.
In game theory, attack scenarios correspond to the attacker’s set of strategies. In addition, they provide means for modelling the information available for the adversary by exchanging the order of “min-max" operators, which is an important part of the stage of the definition of the intentions of the adversary. The main problem within that context is that game theory models may not explicitly separate the targets from the attack description. For example, the set of targets in ARMOR is equivalent to the set of strategies of the attacker, as the attack description does not include a logically consistent sequence of conditions that enabled the attack (as for attack trees). At the same time, NSG family of models has this separation and may be related to the attack trees, as shown in the previous section. NSGs may require additional augmentation and annotation to match the descriptive capabilities of the attack trees. The properties of NSGs are relevant for PGMs as well since the task of attack block may be trivially converted to the task of attack detection (as shown in the next chapters of

the thesis).

Protection description generality. This part is related to the risk mitigation phase of the SRA process. It significantly relies on the way how context establishment and risk identification was carried out. From the scope of game theory, NSGs are considered the most promising model that may be extended to fit the previous criteria. Protection may be modelled in the manner that breaks the subset of links in the graph, which is equivalent to the attack trees.
Based on the criteria 4.4.1, model requirements, and considerations above, it was possible to draw the following conclusions:

1. An extended version of NSG model is used for generic attack modelling.
2. The model for SRA support at the system design stage is based on game theory and PGM (security control selection automation requirement).
3. The model for generic online data processing for attack detection is based on PGM (attack detection requirement).

Part II

Proposed Threat Graph Model

The purpose of this part of the thesis is to provide a detailed description of the basis of the proposed threat graph framework, which is one of the main contributions of the thesis. The main target, as previously noted, is to define a generic modelling framework that can be used for diverse problems in the domain of security. A description of the proposed modelling scheme is given below.

- Definition of the graph-based modelling approach for generic systems and threats (as required in Section 2.1.4).
- A model, based on the proposed graph approach, supports design-related decision-making in security risk assessment (SRA). The model is based on game theory, and it is evaluated on real-world systems.
- A model, based on the proposed graph approach, which supports security-related decision-making during system operation. The model is based on Bayesian filtering, and it is evaluated on a standard cybersecurity dataset.

Chapter 3

Definition of Formal Threat Graph

The aim of this chapter is to present in parallel the standard steps involved in SRA, using SecRAM as an example. Additionally, this chapter outlines the main augmentations to the network security game (NSG) model that are performed in order to extend the generalisation capability by adding the features that are necessary for risk assessment. Detailed motivation for the selection of NSG as a base model is given in the previous chapter. The main reasons are the flexibility of the model and relative topological similarity with attack trees that are a well-established formalism for threat description.

3.1 Structure and Procedure of Security Risk Assessment

Since the graph model is intended to support decision-making in SRA, it is important to understand which steps in the SRA procedures can be adequately represented this way. SRA considers a system-level architecture with different impacts at the service level (safety, capacity, and efficiency), asset level (investment and replacement costs), and operational level (communication bandwidth) [1]. While the main function and services are intangible, impact conditions and vulnerability conditions are modelled for the assets implementing the functions, including threat propagation paths within the set of susceptible (sub-)system components [40]. The defender must act within the scope constraints (e.g., in terms of business and cost), which are implied by the domain of the system to be protected, which leads to a limited decision-making problem.
The threat identification part of SRA standards often aims to define a list of options with a formal step-wise description of the actions to be performed by an adversary and the required conditions for the attack. For instance, the SecRAM standard aims to develop a list of threat scenarios, which are presented as a sequence of actions and prerequisites for the attacker.

As it was discussed in the previous chapter, attack trees develop a directed graph structure, which encodes the main steps involved in the attack.
Taking into account the facts given above, it is reasonable to consider the graph domain as suitable for asset and threat modelling. In order to capture the complex cases with several potential targets, NSG is used as a basis, which will be augmented with the following:

- Introduction of the “context establishment" phase, or asset model, which encodes the main vulnerabilities of the system.
- Approach for graph development to encode possible threats, or “threat model".

3.2 Proposed Modelling Scheme

As identified in the previous chapter, a graph-based formalisation for generic threat analysis is adequate for the current work, considering a high-level system view as it is done in SRA standards. An attack scenario is represented as a directed path linking the targeted supporting asset with the exploited vulnerability, as it is usually done in different graph-based attack representations.
The proposed modelling scheme can be regarded as an extension of the domain used in NSG [9]. The approach described in this thesis defines a regular way to extend the classical NSG to an MO game with multiple types of attack against multiple targets, demonstrating alignment with basic SRA procedures.
The SecRAM methodology defines four main steps, each of which is modelled in the next subsections: primary asset (PA) and supporting asset (SA) identification, threat scenario definition, risk evaluation, and selection of security controls.
For systematisation reasons and to achieve better alignment with existing SRA procedures, the proposed approach is split into two sub-models: an asset model and a threat model. These are described as follows:

- Asset model: Represents the structure of the system to be protected, its main assets, and its functions. As shown in due course, instantiation of the asset model is closely related to the asset identification phase of SRA.
- Threat model: Related to the definition of the main threats and the sequence of actions undertaken by an adversary.

Additionally, this thesis offers a detailed discussion of how both of the models can be defined. In the experimental section of this thesis an example for the development of the TPG is provided for the Remote Tower system [1], which focuses on high-level infrastructure

protection. Remote Tower was selected because a SecRAM process was partially debugged over this project. For this reason, it is possible to re-use parts of the performed SRA in order to demonstrate that the proposed framework is applicable as a support instrument. Remote Tower analysis is performed on a high-level, and it involves both infrastructure security and cybersecurity.
In Chapter 5, the TPG is applied to a more specific case related to a network security problem, which was initially defined in [58]. The process of developing the asset and threat models is presented for a small-scale system “from scratch", without the usage of any preliminary performed SRA analysis like in Chapter 4, showing that the model may be applied in a stand-alone fashion.

3.2.1 Asset Model

It is possible to interpret the first step as a formalised context definition phase, as in ISO 27005. This involves defining the main processes and functions of the system (or PAs, following the SecRAM definition) and SAs that implement them. All the supporting assets may be characterised by a set of parameters that describe the performance of the system. Thus, the system can be considered as a set of assets that provide values for some specific system-wide parameters W. The set of SAs is further denoted as follows:

$$SA = \{SA_1, \dots, SA_M\} \tag{3.1}$$

For example, a system that transmits a video stream has a SA “transmission channel". It is characterised by its bandwidth, which is a parameter of the channel.
A PA is defined as a set of (in-)equations that employ specific limitations over W, and which identify the limits for different subsets of system functions to be operational, as shown below.

$$\begin{aligned} f(W) &\leq 0 \\ g(W) &= 0 \end{aligned} \tag{3.2}$$

Let us return to the video transmission example. The transmission channel is operational if the communication bandwidth is greater than a given threshold in Mbit/sec. Certain parameters W can also be defined based on the system's structure (for example, power consumption in operation), while others may be directly controlled by the defender (such as video compression rate). So, for video transmission example, the following limitations can be identified.

- Transmission channel bandwidth not lower than X Mbit/sec.

- Power supply for the video source is not lower than Y Watts.
- Compression rate is lower than Z % (for better visibility, for example).

These limitations demonstrate examples of inequalities over the system parameters, f-term from the 3.2. An example of a g-term from 3.2 may be given by a set of equations describing the oil-flow in a pipeline, like examples in the previous chapter.
Identification of the main parameters that ensure the operations of the system and assessing the limitations f and g over these parameters is a part of the process that corresponds to the PA identification in SecRAM.
Similar to the process involved in SRA, SA defines collections of targets for the adversary. Every attack is performed against a given subset of supporting assets, affecting in a particular manner the system's parameters, and so the equations may be violated. Similar to the SecRAM methodology, a set of attack types indexed by $1,\ldots,L$ was used (e.g., physical attack, extortion attack, denial of service, and others). An attack of type j against supporting asset(s) $v \in SA$ is encoded as A_{vj}.
The types of attacks discussed in this thesis are different depending on the application. This thesis considers two main examples: remote tower security risk assessment and intrusion detection dataset, IDS2017. Attack types definition varies depending on the nature of the attacks considered against the assets. For the remote tower scenario, the types of attacks are described in section 4.4.1. Attacks considered there are the physical destruction of the asset, compromise of information and electronic interference. The last two attacks are relevant for information exchange equipment. IDS2017 was considering a different type of system; the scope of the analysis is biased to cyber threats. For this reason, the set of attack types was different and is linked to non-authorised manipulation of the data or parameters of the computational nodes of the network described, listed below.

- Data manipulation.
- Computational capacity reduction.
- Machine control.

For example, an SSH brute force attack scenario against a given host may target all three types of attacks against the supporting asset. Denial of service attack targets only the computational capacity.
The main target of the asset model is to define an explicit approach to calculating the impact functions. The central assumption is that it is possible to evaluate the main types of impact based on the values of the constraints and the parameters affected by the attack. Denoting the changed parameters due to the performed attack by $A_{vj}(W)$, the pure impact function can

be expressed as follows:

$$a(f,g,A_{vj}(W)) \rightarrow \mathbb{R}^m \tag{3.3}$$

The term 3.3 presents a general method for defining the impact as a function of the primary assets, expressed in the form of functional constraints 3.2; initial parameters of the system W; and the attack operator A_{vj}. In this context, the term "pure" means that the impact function is evaluated under the assumption that no (additional) security controls are enforced. In case of MO optimisation, $m > 1$. A formal definition of the impact function with security controls is given in the threat model description. The definition of such a function may be a subject for the consideration of a security expert.

A simple example we may consider, which is similar to the defender-attacker model described in section 2.3.2, involves a pipeline infrastructure between a set of sinks and sources, where the parameter to consider is the throughput of each pipe. The next paragraphs demonstrate a formal application of the proposed approach for the construction of the asset model. As a natural PA, we can consider the required minimal throughput for each sink. The main entities are straightforwardly formalised as T,S for the sink and source sets, respectively, as well as the weighted directed graph $G = (N,L,W), S \subset N, T \subset N$ for the pipeline infrastructure. N defines a set of nodes of the pipeline structure, L defines the corresponding set pipes (links), and the weights W correspond to the capacity of each pipe. Denote the flow sequence for pipe $n_{ij} \in L$ by f_{ij}, and the maximal capacity of this pipe by w_{ij}. The overall flow structure is defined by graph G and a set of variables f. When an attack occurs against a subset of pipes the values of w_{ij} are modified (reduced). Therefore, the following limitations can be stated for the PAs under attack A_{vk}:

$$\begin{cases} \forall s \in T \quad \sum_{n_{is} \in L} f_{is} \geq R_s \\ \forall (i,j) \in L \quad f_{ij} \leq A_{vk}(w_{ij}) \end{cases} \tag{3.4}$$

The first inequality states that the total inflow for each sink s should not be lower than R_s. The second inequality states that the flow at pipe (i,j) does not exceed the maximal capacity w_{ij} adjusted by the attack operator A_{vk}, which represents an attack of type k against SA v.

Therefore, the set of the system parameters can be defined using the pair (F,W), where W represents the set of maximal capacities of each pipe (which can be modified by an attacker), and F is controlled by the defender.

If the equation system (3.4) has a solution $F = \{f_{ij} | (i,j) \in L\}$, there is no impact on the system (according to the stated PA). In case the system has no solution, the impact function may be defined as proportional to the lack of resources in the sinks. To give an example:

$$a(F,R,A_{vk}(W)) = \sum_{s \in T} C_s\Big(\sum_{n_{is} \in A} f_{is} - R_s\Big)_+ \quad \forall (i,j) \in L \quad f_{ij} \leq A_{vk}(w_{ij}) \tag{3.5}$$

Here $C_s \geq 0$ stays for the relative cost of sink s. Overall, the example measures the shortage of inflow for each sink. Less straightforward definitions of the impact functions may be used for different problems. For the specific case, it is important to notice that the function is not defined as a solution of an optimisation problem, which means that the user does not adapt the flow inside the pipelines in order to minimise the impact. However, the adaptation of the flow δF may be regarded as a security control and, as such, defined at the later stage.

3.2.2 Threat Model

The threat scenario definition phase is used to determine the main vulnerabilities of the system, and to identify a constructive approach for their exploitation. The term “threat scenario" in SecRAM and ISO 27005 is defined as “chain or series of events that is initiated by a threat and that may lead to an unwanted incident." These chains of events are formalised in the proposed methodology as a directed graph.

Considering the literature, apart from the works that are related in a direct way to mathematical modelling, graph-based attack description is explicitly used in the attack tree standard [57, 25]. This describes a sequence of actions that must be undertaken to perform the attack. Every action is assigned to an edge or node in the tree, which is related to the state of the adversary. Another similar example is a cyber-focused attack graph, which encodes a sequence of exploited vulnerabilities. Thus, an NSG family of models provides an adequate set of means by which a definition can be given of the threat scenarios according to the existing SRA standards.

The description of the attack scenario in the scope of the proposed method is further denoted as an attack model or a threat path graph (TPG). As a concise description of the approach, the attacker’s strategy is to select a pair of SAs and a type of attack against the target. Following the NSG model, the adversary selects a path in the graph $G = (N, E)$ from an entry node to the target. This graph encodes the main steps that the attacker must undertake to perform an attack against $v \in SA$. For instance, this may include compromising physical security by opening doors or accessing a computer. Each node of the graph corresponds to some status of the adversary within the system, i.e. physical position, level of control over a machine, availability of information. Each edge corresponds to an action of the adversary that targets changing the status. A subset of nodes of G is mapped to the set SA, defining the set of target nodes V.

In particular, following notations for the graph properties are used:

- $N, |N| = n$ are the set of nodes in graph G and its cardinality, respectively.
- E is the set of edges in G.

- V denotes a subset of nodes of G, $V \subset N$, which corresponds to the SA list. $|V| = M$.
- T is the set of entry points (following the NSG formalism), $T \subset N$.

Next, a high-level procedure for TPG construction is given.

Threat path graph construction inputs. Main inputs for TPG construction are listed as follows.

- List of all SAs for the system protected.
- Detailed list of possible attacks against the system's SAs. Each threat should be presented as a detailed sequence of actions of the attacker. The description should be presented in the form of pairs “action performed - status obtained". For example, “enter the room - present in the room", “enter root password - logged in with root privilege".
- List of security controls under consideration.

Construction procedure. It is carried out as described in the list below.

- For each of the SAs, a node is created, forming the set of vertices V.
- Each threat scenario is validated against the set of all possible security controls. If security control is relevant for the threat scenario, described sequence of pairs $action - status$ should explicitly mention all actions and all statuses where the security control may be relevant. Threat scenarios should be augmented with additional pairs if needed.
- For each threat scenario, an “initial status" of the adversary is created. This status describes the initial pre-requisites of the adversary, his physical location, etc. This status should be obtained by the adversary by performing actions that are outside of the scope of the SRA. For example, if a computer network with a public IP address is protected, a potential status of the adversary may be “online with knowledge of the public IP".
- For all statuses, corresponding nodes are created. Different threats may share actions and statuses. In this case, a single node is created for the same set of shared statuses.
- For all actions that lead to a transition from one status to another, edges between corresponding nodes are created.

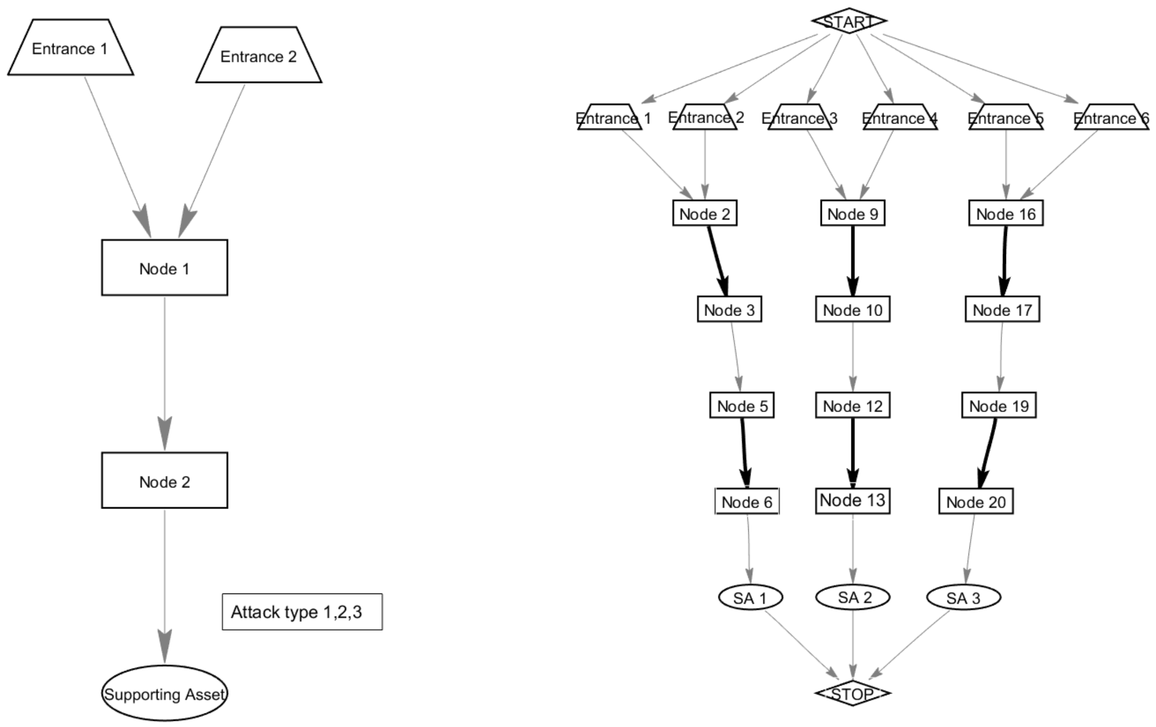


Figure 3.1: Graph transformation process. Image on the left-hand side represents the initial graph with three types of attack, two entry points, and a single SA. Image on the right-hand side shows the corresponding extended graph.

Security control allocation depends on the nature of the protective mechanisms. Some of the security controls may block certain actions of the adversary; others may prevent obtaining the status in principle and block all further actions. For example, assume we consider a node that corresponds to “root login to machine" condition. An authentication mechanism may prevent obtaining root access to the machine from a host with no ssh key. At the same time, an administrator login notification system (when the system administrator needs to authorise any logged users manually) may prevent any further actions even after logging in. From this perspective, the login system blocks a subset of incoming edges to the node, while the admin notification system blocks all possible further actions (or blocks the whole node).
For this reason, some of the security controls are assumed to block edges; some controls block nodes. Nodes or edges which where security controls can be enforced are referred to as “checkpoints". In order to simplify the mathematical formalisation, it is possible to transform the developed graph in a way that all blocks correspond to an “edge block", which is described further at the graph extension procedure.

The adversary's attack is formalised as a pair comprising the threat path and attack type, namely $(P(b,v,G),A_{vj})$, where $P(b,v,G)$ defines a simple path (with no loops) from $b \in T$ to $v \in V$. The adversary's strategy may be reduced to the standard NSG formulation by changing the structure of G. The new, extended graph G' is constructed as follows. First, the set of nodes related to the SAs is replicated L times, thereby meaning that $v_i \in V$ is

decomposed into $v_i^1, \ldots, v_i^L$. The same operation is performed for the nodes from T. The set of nodes and set of edges related to different attack types shall not intersect. Node v_i^j can be only connected to the nodes related to the attack type j. The same rule is relevant for the set T. Therefore, the selection of an entry node strictly determines the type of attack to be performed. Due to this approach, it is possible to alter the graph structure for different types of attacks, and significant flexibility is introduced into the model instantiation process.
For convenience, two additional modifications of the graph G are performed:

- All entry nodes from T are connected to an additional "start" node s. A "terminal" node t is connected to all nodes from V. This reduces the set of adversary strategies to a set of all paths in the extended graph $P(s,t,G')$.The selected path automatically determines the type of attack.

- For some mathematical problems, it is more convenient to work with edges rather than nodes. Examples of such well-established problems are least weighted paths, graph cuts. Further in the thesis, the least weighted path linear programming formulation will be used in the formulation of the game solver. For this reason, all the nodes from Q are split into pairs of nodes and a connecting edge. For consistency, the edge that corresponds to node $n \in N$ is denoted as (n^s, n^e).

Suggested reformulation allows speaking about the checkpoints only in terms of edges. $Q \subset E$ denotes a set of edges where security controls may be deployed. The computational complexity of this transformation is $O(mn)$ where m defines the number of attack types. The same order of expansion is required memory-wise. Such growth is not exponential, but may significantly increase the size of the graph in case of a large number of attack types. For the models considered in the thesis, the graphs were tens and hundreds of nodes for high-level analysis. However, it is important to stress that such a transformation does not significantly increase the number of edges, which may be estimated as $2m(|E|+n)$. This is important for the computational complexity of some of the algorithms, such as the least weighted path (used in chapter 4). It depends linearly on the number of edges and $O(n\log(n))$ on the number of nodes. This means that the complexity of the transformation decreases the performance of the algorithm almost linearly on m.
An illustration of the construction of graph G' from G is given in Figure 3.1. The graph on the left-hand side of the figure demonstrates a simplified version of an attack, which aims to steal, compromise, or delete mission-critical data from a computer. Thus, we define three types of attack and an SA. There are two entry points: one for access to the computer directly, from the exact physical location, and another via SSH. The first checkpoint represents login and password authentication, while the second node represents the data access process (e.g., location of the correct folder on the machine).

For notational convenience, the assumption is made in this thesis that the extended graph is initially used in the model structure. This means that the attack type index is omitted unless otherwise stated. Furthermore, for the remainder of this thesis, the extended graph is denoted as G.

It is obvious that the last two steps of SecRAM and ISO 27005 (i.e., risk evaluation and security control selection) can be mapped to the gain function definition and the selection process for an optimal defence strategy. In the proposed model, the set of implemented security controls mitigates the impact of the attacks, and is denoted by Ω. Controls are implemented at the subset of edges of the graph G (nodes of the initial graph). A list of possible controls should be inserted into the model. Let us denote that list by $L_d = \{C_1, \dots, C_r\}$. Then, the following assumption can be made:

$$\Omega = \{0,1\}^{r \times n} \tag{3.6}$$

This indicates the type of controls (among r proposed types) that are to be applied for each of the $|Q|$ edges that correspond to the nodes of the initial graph. This yields a control matrix representation of the model, with $\Omega_{ij} = 1$ if a control $C_j \in L_d$is implemented at checkpoint $(n_i^s, n_i^e) \in E$. The approach can be extended to procedural controls, which are not assigned to any edge $(n^s, n^e) \in E$. This formulation can be interpreted as indicating that such controls are assumed to be placed at the nodes from V, mitigating all attacks of a given type independent from the path.

As this thesis considers not only the problem of attack mitigation at the design stage but also the problem of attack detection, the concept of an “attack detector" is used in the chapter on the dynamic model. The formal assignment can be undertaken in a way that is similar to the security controls, where controls are assigned to nodes or edges.

3.2.3 General Recommendation for Model Development

The main purpose of the threat graph is to formalise the steps an adversary needs to perform in order to attack the targeted SA. The path in the threat graph defines the sequence of states that an adversary must pass through (which is analogous to the doors or rooms that an individual must pass through in the case of real-world movements), the machines that should be controlled in case of cyber-attacks, and others. During the experimental phase of this thesis, the following recommendations for the initial graph construction (i.e., not the extended graph construction) were developed:

- Split the nodes by their nature. If the node represents a position in the building, room, or geographic area, then it should be regarded as a “geographic(physic) node". This means that there is a geographic area associated with this point in which controls could

be placed. The edges between geographic nodes represent possible directions in which the attacker's position may have changed. The area that is “close" to the supporting asset, such that the attack type j has a link to the node in V, represents a given SA.
If the node represents control (or a level of control) over a certain machine, it is regarded as a “cyber node". Several nodes can represent different levels of control over the same machine. The links between nodes represent possible changes in the level of control, data manipulation, network communication sessions, or an obtaining of control over a different machine. It is likely that all the machines used for the definition of the “cyber nodes" are related to an SA. Therefore, if the given node represents an essential control over the machine, it is possible to perform certain types of attacks against the given machine.
Separation of different types of nodes is beneficial not only in terms of modelling convenience; at the same time, such a split is important in providing sufficient means for the detailed encoding of the actions of the adversary. Depending on the case, it is possible to provide semi-automatic rules for TPG validation (e.g., the physical attack cannot be performed from a cyber node, with specific exceptions, or it is impossible to upload malicious firmware from a physical node, given certain access requirements).

- If there is access from the node to some SA, the list of possible attacks from a given node against a corresponding SA should be specified. With no loss of generality, all types of attacks are possible against all SAs (though it is true that some attacks will have no impact, such as a distributed DoS attack against personnel).

- The system entry points are specified as nodes (geographical territories or machines), which should be monitored from the scope of SRA, and there is access to these nodes from external (that is, external to the system) territories or machines. For example, it could be a computer that has a direct connection to the external network or the Internet. In the geographic scope, the entry point could be an entrance to the building in which the system is located.

In addition to general recommendations, two specific cases are considered in the next subsections: firstly, action plan development; and secondly, attacks with preliminary conditions and sequential attacks.

Action Plan Development

The proposed model may be applied in a different manner, which is especially relevant for dynamic security games with multiple actions from the side of both the defender and attacker. Dynamic security games with action plans are established as a sequence of actions, and they may be represented using finite automata [6]:

$$\begin{cases} s_{t+1} = f(s_t, a_t, d_t) \\ D_{t+1} = D(s_{t+1}) \\ A_{t+1} = A(s_{t+1}, d_{t+1}) \end{cases} \tag{3.7}$$

where $s_t \in S$ denotes the system state; a_t, d_t represent the attacker's and defender's actions at time t, respectively; and A_t, D_t refer to the set of available actions for the attacker and defender, respectively, depending on the state and the other player's actions).

The intuitive understanding of equation system 3.7 is that the system moves to another state after the defender and attacker perform an action. The assumption is made that the attacker responds to the actions of the defender, and so this situation can be understood as consisting of a sequence of defender-attacker games. The set of terminal states S^* defines the end of the gain, after which the pay-off is calculated.

The action plan development game may be reduced to the TPG model as follows:

- Assume that the set of *SA* nodes is $S*$.
- The set of nodes of the graph is S.
- The set of edges is $\{(s_1, s_2) | \exists d, a : \quad s_2 = f(s_1, a, d)\}$.

Based on the above, if a security control d_{fixed} is selected for the node s_{fixed}, the action of the adversary is limited to the selection of a node $s(a) = f(s_{fixed}, a, d_{fixed}), a \in A(s_{fixed}, d_{fixed})$. The lack of security control at any node may be considered a so-called “null-action".

The possibility of employing decision-making graphs in the TPG expands the application scope when the TPG may be constructed in a mixed manner. For instance, if the adversary reaches some given node (which can be a physical position or a level of access), a counter-action can take place. In this case, the outgoing graph from the corresponding node relates to an action-planning type of graph. This can be repeated through the overall model several times.

The main disadvantage of the proposed methodology is the size of the graph. This is because it should take into account all possible combinations of actions (which is an exponential from the number of actions). Therefore, in general, the total number of nodes also grows exponentially. However, the expected connectivity of the graph might be low, which still may allow the application of this approach for limited-scale counter-actions.

Attacks with Preliminary Conditions and Sequential Attacks

An attack against a given SA is sometimes enabled by specific preliminary conditions, which is achieved by an attack against another SA. For example, a data spoofing attack against a

database is possible once the credentials for it have been obtained via brute force, or by executing a Heartbleed attack against a centralised password management server.
Formally, let us consider an attack of type t_2 against any of the SAs in an expanded graph $\{SA_{21},\ldots SA_{2n_2}\}$, which could be enabled after a successful attack of type t_1 against $\{SA_{11},\ldots SA_{1n_1}\}$. It is essential to remember that, within the expanded graph, the supporting asset node automatically determines the corresponding type of attack. Further discussion in this section is carried out in terms of the expanded graph.
Suppose that the graph G_1 describes the threat model for attack t_1, and that G_2 describes the attack t_2 under a condition where the attack t_1 has already happened. The resulting TPG should take into account the required sequence of actions and, accordingly, encode an attack against multiple SAs. To define the TPG construction procedure for this case, consider first the following notations:

- $G=(N,E)$ - TPG. Then let us denote:
  $N[\{SA_1,\ldots SA_n\}] = \{n\in N | \exists SA_i \in V : p(n,SA_i,G)\subset G\}$;
  $E[\{SA_1,\ldots SA_n\}] = \{(e_1,e_2)\in E | e_1,e_2 \in N[\{SA_1,\ldots SA_n\}]\}$;
  $G[\{SA_1,\ldots SA_n\}]$ is a graph defined by the pair $(N[\{SA_1,\ldots SA_n\}], E[\{SA_1,\ldots SA_n\}])$.

- We introduce an additional type of SA, which is defined as an ordered subset of the set V. The node $(SA_{i_1},\ldots,SA_{i_k})\in V^k$ indicates the set of "victim" SAs, as well as the order and the types of attacks (implicitly, after the expansion of the graph, as mentioned at the beginning of this subsection).

- $par(n)$ – A set of parents of a given node n. The definition is correct because the graph is directed.

The formal procedure for the TPG construction method from two linked sequential attacks is described in Table 3.1. The presented algorithm, the sequential attack TPG aggregation algorithm (SATAA), may be adjusted in terms of connections between the graphs. This is because, in the presented approach, all parent nodes for a given SA in graph G_1 are connected to all entry nodes of G_2. It is possible to supply additional inputs to SATAA, which could increase the precision of the TPG.
In a general case of L attacks, it is possible to create a set of pairs of sets of SA, denoting "enabling attack" – "subsequent attack and attack graph": $\{(SA^e,(SA^c,G^c))\}$. The subsequent attack could happen if any of the enabling attacks succeeds. In this case, the TPG described by graph G^c is *enabled* by attacks from SA^e. It is assumed that such a subset of SAs exists, which could have an "empty" enabler (i.e., can be executed without a "preparation" attack). For example, a DoS attack on a server could be executed with no preliminary attacks. The set of pairs "enabler-sequential" attacks is denoted by ES. The algorithm for the general case sequential attack construction is presented in Table 3.2. The multiple sequential attack TPG

```
Input:

  • Enabler attacks: Defined by a list of SAs: $SA_1 = \{SA_{11}, \ldots, SA_{1k}\}$ of the
    same attack type $t_1$.

  • Consecutive attacks: $SA_2 = \{SA_{21}, \ldots, SA_{2m}\}$ of same attack type $t_2$.

  • Graphs for single attack types: $G_1 = (N_1, E_1)$ for $t_1$ and $G_2$ for $t_2$ (under
    the assumption that $t_1$ has already happened).

Output: $G_{res}$ – TPG describing sequential attack.
  1. $G_{res} = (N, E)$, where $N := N_1, E := E_1$

  2. Construct a sub-graph $G_2^* = G_2[SA_2] = (N_2^*, E_2^*)$.

  3. For each $v \in SA_1$

      (a) $G_{2v} = copy(G_2^*)$. Redefine all SAs in $G_{2v}$ as follows:

              $SA_{2i} \to (v, SA_{2i})$

      (b) $N := N \cup N_{2v}, E := E \cup E_{2v}$
      (c) For each node $u \in parent(v)$ and $e_{2v}$ in the set of entry nodes $T_{2v}$:

              $E := E \cup \{(u, e_{2v})\}$
```

Table 3.1: Sequential attack TPG aggregation algorithm (SATAA)

```
Input:

  • Set $ES$.

Output: $G_{res}$ – TPG describing all sequential attacks.
  1. Select pair $(SA_1, (SA_2, G_2)) \in ES : SA_1 = \emptyset$ (exists by definition, as given
     above).

  2. $G_{res} := G_2$. $V_{res}$ denotes the current set of SAs in $G_{res}$.

  3. While $ES$ changes at each iteration, do:

      (a) Find $p = (SA^e, (SA^c, G^c)) \in ES : SA^e \cap V_{res} \neq \emptyset$. Remove $p$ from
          $ES$.
      (b) $V^e = SA^e \cap V_{res}$.

              $G_{res}, V_{res} := SATAA(V^e, SA^c, G_{res}, G^c)$

      (c) $SA^e := SA^e \setminus V^e$. If $SA^e \neq \emptyset$, return $p$ to $ES$.
```

Table 3.2: Multiple sequential attack TPG aggregation algorithm (M-SATAA)

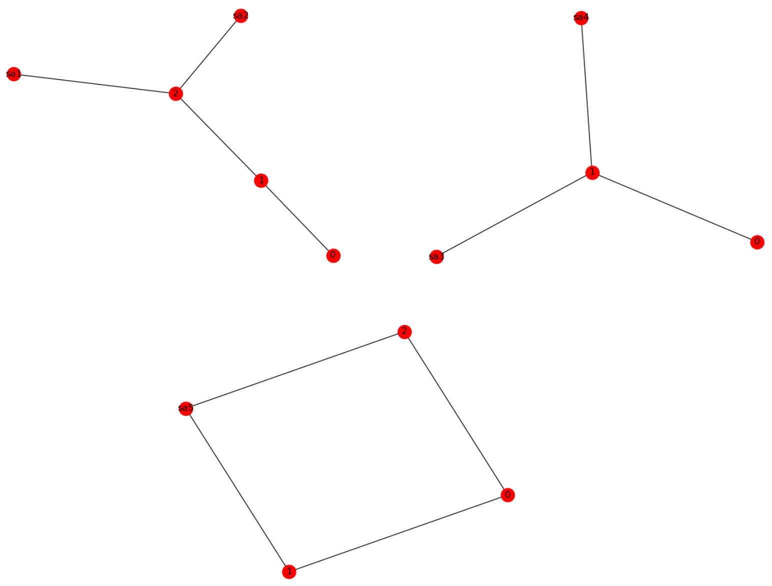


Figure 3.2: Sub-graphs used as an input for M-SATAA. 3 connected components: initial graph with $\emptyset$ enablers (upper-left), sub-graph enabled by an attack against $SA1$ or $SA2$ (upper-right), and sub-graph enabled by attacks against $SA2$ and $SA3$ (bottom).

aggregation algorithm, or M-SATAA, could be modified, allowing the definition of complex multi-stage attacks when the enabler is also a sequence of attacks. It requires a modification of the loop conditions (a) and (c). For the case of a multi-stage attack, the condition (a) would check for $p : \exists p_1^* \in p_1, v \in V_{res}, \{p_1^*\} \subset \{v\}$. Here the operator $\{v\}$ converts an ordered sequence into a set so that the order is not considered.

An example of graph construction is given in Figures 3.3 and 3.2. Three sub-graphs are supplied, as shown in Figure 3.2. One of the parts contains two SAs ($SA1$ and $SA2$), which could be attacked without specific preconditions, denoted by G_1. The second graph, containing another two SAs ($SA3$ and $SA4$), is “accessible" only after an attack against the SAs in the first graph, denoted by G_2. The third graph, G_3, contains a single SA ($SA5$), which can only be approached after $SA2$ and $SA3$ are attacked. Therefore, the input set for M-SATAA is as follows:

$$\begin{gathered} L = \{(\emptyset, (\{SA_1, SA_2\}, G_1), (\{SA_1, SA_2\}, (\{SA_3, SA_4\}, G_2), \\ (\{(SA_2, SA_3)\}, (\{SA_5\}, G_3))\} \end{gathered} \tag{3.8}$$

An example of an application of these algorithms on real-case scenarios for cybersecurity is given further in this thesis.

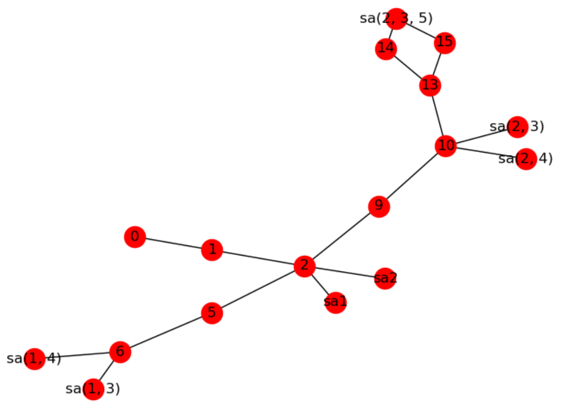


Figure 3.3: Example of M-SATAA application problem showing the resulting graph for sequential attacks constructed from the sub-graphs from Figure 3.2.

3.3 Comparison with Attack Trees

The proposed TPG method is inspired both by NSGs and attack trees. While the difference from NSGs is defined by the augmentations described in this chapter, the differences from the attack trees should be elaborated. The differences between attack trees and NSGs listed in the previous section remain in place: attack trees describe only one attack at a time, but they are more illustrative and have well-established development methodology. TPGs are bringing the NSGs closer to the SRA domain by introducing a mapping to several important entities of the process.

- TPGs contain an explicit link to the context establishment phase, i.e. supporting and primary assets are introduced.
- By construction, nodes and edges are linked to the information about the statuses and actions of the adversary, which is explicitly related to the nodes' conditions in attack trees. The main difference is that instead of the introduction of "and" node types, the graph is extended using SATAA algorithm. However, it is possible to introduce "and" nodes explicitly like in attack trees and analyse them at the stage of the optimisation problem solving.
- A high-level development procedure is provided for TPGs.

To summarise, TPGs may be considered as extended attack trees with explicit targets (assets), multiple attack scenarios captured in a single graph, and more complex topology. The development process for attack trees and TPGs shares a set of common features such as the definition of "entry nodes" (leaves), statuses (conditions and actions), attack goal (supporting assets). However, due to introduced complexity and "topological" introduction of "and" conditions using SATAA algorithm, TPGs could be less illustrative and not directly useful for manual analysis. However, some of the processes like threat quantification for the adversary can be performed in a similar way as for attack trees: cost assignment for each node and propagation from “entries" of the graph. TPGs share the same advantages of attack trees, like the formality of the description, possibility for threat logging and re-using, identification, and listing of main conditions that enable the attack.

3.4 Comparison with Cybersecurity Graph Models

In this section, the TPG is compared against the cybersecurity graphs used in the models presented in 2.6. The method described in [64] is omitted from the comparison here, as it relies on attack trees, which are discussed above. We will return to the method of [64] in section 4.5.3.

3.4.1 Infrastructure and Evolution Graphs

The SRA model described in [8] relies on a pair of graphs: infrastructure graph and evolution graph.

The method considers a system I, which is composed of a set of components C. Infrastructure graph $Ih(I)$ describes the system in terms of its main components and their dependencies. For each component, three security attributes from the CIA (confidentiality, integrity, availability) triad are defined. $Ih(I)$ includes one node $N(C)$ $\forall C \in I$, and one arc from $\{N(C_1),\ldots,N(C_k)\}$ to $N(C)$ for each dependency $\{C_1,\ldots,C_k\}$ of C. For each arc H, two attributes from the CIA triad are defined. A pair of attributes (a_1,a_2) for a link $(N(C_1),N(C_2))$ means that if the attribute a_1 is controlled for C_1, then the attribute a_2 is controlled for C_2.

The system I has a set of users. For each user u a set of rights is defined, where each right is presented as a pair $(C,A), C \in I, a \in CIA$. Each right defines an attribute in the corresponding component controlled by u. In [8] a special algorithm for the computation of the “extended" set of rights is presented, which analyses the structure of $Ih(I)$ and augments the rights of u based on subcomponent dependencies and arc labels (referred to as “transitive closure" of rights).

The evolution graph describes how a set of system users may acquire additional rights over the system by implementing a sequence of elementary attacks. Each elementary attack A has a set of precondition rights $pre(A)$, required to execute it, and a set of post-condition rights $post(A)$ that the user gains after executing the attack. An infrastructure security state (or just state) is defined as a set of pairs $\langle$user, user rights$\rangle$ for each user of the system. Each node of the evolution graph corresponds to a state of the system. An arc $\langle A,u\rangle$ connects two nodes n_1 and n_2 if a user u at state n_1 has all rights $pre(A)$, and execution of A by u would bring the

system to the state n_2. The graph may be constructed automatically, given the infrastructure graph, initial user rights, and elementary attack descriptions.
The initial state of the system is defined by the “extended" set of rights for each user. An attack is defined as a path in the evolution graph from the initial state to a set of “goal" states, where at least one user obtained a set of targeted rights.
The pair of infrastructure and evolution graphs have certain similarities with the asset and threat models proposed. This way, the asset model and infrastructure graph describe the main components of the system and their dependencies, while the evolution graph and TPG are describing the threat scenarios.
The infrastructure graph shows high-level dependencies between the components of the system and how those components interact with each other. The asset model shows a high-level dependency between system processes and their technical components, leaving apart the dependencies between the SAs. This brings the proposed method closer to the SecRAM and ISO27001 but brings a simplification of the system description.
The evolution graph is constructed automatically based on the elementary attack descriptions. TPG has a partial construction automatisation. Evolution graphs are able to represent a cooperative attack executed by multiple users, TPG does not have a representation mechanism for that. At the same time, TPG is much more explicit in terms of the targets, required sequence of actions, and attack types. TPG is designed to be applicable for both cyber and physical domains. The evolution graph paper does not explain how to adjust the model for the physical domain. TPG provides not only a sequence of elementary attacks but also a possibility to introduce a detailed description for each of them.
To conclude, a direct comparison of the proposed methods is not fully complete without taking into consideration the risk evaluation and mitigation mechanisms. However, by comparing the graph models, the following distinctive features may be listed.
Evolution and Infrastructure graphs:

- More detailed system component dependency description that may allow a deeper understanding of threat propagation (which parts of the system are vulnerable to which attacks).
- Automatic evolution graph construction from the inputs, which limits the possibility of human error.
- The evolution graph allows for the modelling of multiple attackers at a single moment of time.

Proposed method:

- The modelling is more explicit in terms of targets of the attacks (the exact set of supporting assets) and system services (primary assets) that are disrupted.

- TPG gives a more detailed description of each attack, including sub-steps. This allows for more complex defence modelling. Control might be deployed to prevent a specific action, which is a component of various attacks.

3.4.2 Compromise Graphs and Attack Paths

Compromise graphs were introduced in [45] as a model for quantitative cyber risk estimation methodology. The initial step of the proposed method is the identification of the main components and devices within the system. For each device, a corresponding list of vulnerabilities is composed. The paper suggests using open vulnerabilities identification libraries.
The next step of the analysis is the construction of the compromise graph. The graph has four types of nodes.

- Start - a single “entry" node of the graph. The attacker has no prior knowledge of the system.
- Launch - enough data has been collected by the attacker to develop an exploit. There is only one node of this type for each potential attack.
- User privilege - this state applies to a particular machine. Represents that the adversary has gained user privilege on that machine. Only one such node for each machine is available.
- Root privilege - similar to the previous type, represents root privilege on a particular machine.
- Target node - any condition where the attack has succeeded.

The edges in the compromise graph represent the change of the state of the adversary. Every change is assumed to be achieved by exploiting some vulnerability. Each arc is labelled by one of the following vulnerability categories.

- Reconnaissance
- Breach - represent edges starting from a “launch" node.
- Penetrate - represent edges starting from a user or root permission node and end on the same type of node.
- Escalation - escalation of the permissions on the same machine.
- Damage - represents a transition to a target node.

An attack is presented as a path from the “Start" node to some target node.
The asset definition phase in [45] is not strictly formalised. However, it suggests listing the main devices of the system and splitting them into primary targets and perimeter machines. For the identification of the security requirements of the primary targets, [45] suggests using the CIA triad. Such a representation of the attacker’s actions may be considered as a special case of a cybersecurity tailored TPG with additional annotation, where all nodes have limited types of categories and all arcs are explicitly labelled by a corresponding action. No other conceptual differences with the TPG are present. Explicit definition of the node types in compromise graphs brings stricter rules for the graph definition. However, this approach does not detail any methods for automatic graph construction for sequential attacks.

3.4.3 Network Security Risk Model State Graph

The network security risk model (NSRM) was introduced in [33] and was designed for the purpose of providing a measure of risk and risk management options. The system context is defined by multiple models, which include:

- Access level and barrier diagrams that provide a mechanism for capturing at a detailed level the requirements for obtaining a higher level of access to the system, describing impediments to achieving new privileges, and assessing the overall security of the system. The method may be considered as a set of fault trees with “and" and “or" junctions. Fault trees used in NSRM may be considered attack trees, where each leaf has an assigned probability of event occurrence.
- Hierarchical system decomposition, where the overall system is decomposed into sub-components. The decomposition reaches the lowest level of technological enablers. For the developed hierarchical decomposition, failure modes and main system processes are identified.
- Formalised hierarchical descriptions of attack scenarios, that include attacker’s objectives, attacker type and points of access.

The information above is used to decompose the overall network into a set of system components. Each component may be in two possible states.

- Secure (indexed as 0): the attacker has no access to the component’s resources.
- Compromised (indexed as 1): the attacker has full access to the component’s resources.

All components are mapped to the set of nodes of graph G that describes the possibilities to compromise different components of a system. If an arc connects node c_a with c_b, that means that c_b can be compromised from c_a.
Based on the system decomposition into n components, the tactical state-space S is defined as a set of all combinations states of all components and two additional terminal states of the attacker: “attack success" (state $n+1$) and “attack failure" (state $n+2$). Therefore, the tactical state may be described as $(n+2)$-dimensional Boolean vector. At each state, the attacker must choose one of the two types of actions.

- Attempt to compromise a secure network component. A component may be compromised if there exists a corresponding path in G.
- Attempt to disrupt one or more infrastructure processes. The set of processes that could be disrupted depends on the state.

In the case the system is in state s_i and the attacker performs an action u, the system may transit into a state $s_{i'}$ with the probabilities defined below.

- If the action is an attempt to compromise a set of components, the probability is evaluated as follows:
$$p_{i,i'}(u) = \prod_j p^j_{z,z'}(u)$$
, where $p^j_{z,z'}(u)$ is a probability the c_j will transition from state $z = s^j_i$ to $z' = s^j_{i'}$.
- If the action is a process disruption attempt in component k, the probability is denoted as $p_{i,n+1}(u) = p^k_m(u)$. Here m defines a disrupted process mode.
- The attacker is stopped and removed from the system with probability $p_{i,n+2}(u) = \prod_j p^j_{z,0}(u)$.

All these probabilities are evaluated using the fault trees developed at the context initialisation phase. The method provides a mathematically justified approach (given certain assumptions) for the calculation of state transition probabilities. Further, these probabilities are used to quantify the risk and identify the most vulnerable parts of the system.
The system and attack scenario description introduced in NSRM are completely different from TPG and asset model presented in the thesis. However, both of the formalisms are presenting the system as a set of inter-related processes and components, describe the attack scenario as a sequence of actions, and indicate exact processes disrupted (primary assets in the case of the model proposed). For this reason, it is hard to compare the distinctive features of each of the models. A more informative comparison of the models is presented for the risk evaluation phase of the NSRM and proposed model in section 4.5.

3.5 Summary

This chapter established a framework for generic attack modelling. As stated in the literature review part of the thesis, the approach is developed using graph models. The development of the approach was aligned with two main stages of SecRAM:

- Asset identification stage, which is transferred to the asset model and impact function definition.
- Threat scenario definition, which is transferred to the threat model, or TPG.

Due to strict limitations, as given in Section 2.1.4, model development is still quite flexible and requires significant expert involvement. This involvement may is especially relevant at the stages of the asset model definition, identification of the main functions and parameters. The level of detailisation of the description of the system may be different. Identification of the set of statuses and actions of the attacker at the TPG construction phase also relies on expert judgement and may vary depending on the analysis level.
However, the developed formalism allows the application of automatised decision-support methodologies, which are relevant both for the next stages of the SRA process and operation-time decision-making. These adaptations of the suggested approach are further detailed and validated on real-world scenarios in the next chapters of this thesis.
TPG development process is similar to the attack trees and may be used to analyse possible threats in a similar fashion. However, some complexity for manual analysis may be introduced by the size of the graph, which could be partially solved by automatised analysis. The descriptive capability of TPG was also compared with several cybersecurity graph models.

Chapter 4

Static Security Model: Description and Evaluation

4.1 Generic Problem Statement

In the previous chapter, an explanation of the threat graph preparation was given, linked with the main steps of SecRAM. However, the security risk assessment (SRA) process was covered only partially. This is because the graph represents only the domain and asset description. The domain and threat descriptions are linked to the threat scenarios evaluation and possible security control selection, especially at the stages of the expansion of the initial graph. However, the process of the exact allocation of security controls at checkpoints should be considered as an operation over the graph.
In this chapter, the problem of risk evaluation and security control selection is formalised in the generic formulation (following the requirements from Section 2.1.4). The aim is to take into account specific aspects of SRA, which includes consideration of all scenarios simultaneously and the consideration of multiple target functions. To summarise, this chapter deals with:

- Identification of a constructive way for impact function computation and definition of the game theory problem for risk treatment.
- Development of an algorithm for security controls allocation within threat path graph (TPG).
- Demonstrates the TPG and game theory model instantiation and validation over the remote tower case. Obtained results are compared with SecRAM and standard network security game-like (NSG) approach.

In SecRAM, risk evaluation process separates different types of impact or impact areas. Furthermore, the value of the impact is translated into a single number using the *max* operation, but this approach may differ for different SRA methods [36]. Therefore, for impact evaluation, we propose not a single “cost", but a vector of values. We do not define a common gain function, but evaluate each of the impact areas separately, defining a set of “optimal" solutions, which could be further considered by the experts or evaluated automatically using a specific cost function. This makes the analysis more flexible, as it would bring into consideration of the security analysts (users of the model) possible trade-off situations when one impact function is balancing against another.

The number of impact areas is denoted as m. Each of the types of impact is evaluated against given security controls Ω and attacks (paths in TPG G) $P : I_i(\Omega, P), i = 1, \dots, m$. Next, we formulate the underlying generic optimisation problem. As we consider each of the types of impact separately, we address the solution as a multi-objective optimisation problem (MOOP). Accordingly, we use Pareto-optimality (PO) as a condition for the security control selection. PO condition analyses each impact function separately, which allows for detection of the trade-off cases. In addition, the solution that maximises an arbitrary monotonic gain function is PO. For this reason, PO frontier identification allows the users of the system to select and evaluate different gain functions over the PO frontier identified.

Using the impact function definition from the previous chapter and [54, 59], the Pareto-optimisation problem for the zero-sum game is stated as follows:

$$\min_{\Omega} \left[\max_{P} I_1(\Omega, P), \dots, \max_{P} I_m(\Omega, P) \right] \tag{4.1}$$

The notation $\min_{\Omega}$ in 4.1 stays for the search of all Ω that bring PO “impact" vectors $\left[\max_{P} I_1(\Omega, P), \dots, \max_{P} I_m(\Omega, P) \right]$. It should be noted that we consider a $\min - \max$ problem, which implies that the attacker acts optimally, and with full information about the defender's security configuration. This is a worst-case scenario. This approach was selected because the obtained security configuration provides an upper bound for the impact. Indeed, suppose that the game is not a zero-sum game and an alternative gain function is defined for the attacker $U^a(\Omega, P)$. Therefore, the game is changed to:

$$\begin{gathered} S(\Omega) = \operatorname{argmax}_{P} U^a(\Omega, P) \\ \min_{\Omega} \left[\max_{P \in S(\Omega)} I_1(\Omega, P), \dots, \max_{P \in S(\Omega)} I_m(\Omega, P) \right] \end{gathered} \tag{4.2}$$

Here $S(\Omega)$ stays for the set of attacker's strategies that maximise the impact function U^a given security controls Ω. Considering the fact that $\forall i \in \left[m \right], \forall \Omega$

$$\max_{P} I_i(\Omega, P) \geq \max_{P \in S(\Omega)} I_i(\Omega, P) \tag{4.3}$$

This property is hold due to the fact that set $S(\Omega)$ is smaller than set of all paths in TPG G. It can easily be shown that $\forall\Omega_U$ that is a solution of equation 4.2 and $\forall\Omega$ that is a solution of equation 4.1 such that

$$\forall k \in \left[m\right] \quad \max_P I_k(\Omega,P) \geq \max_{P\in S(\Omega_U)} I_k(\Omega_U,P) \tag{4.4}$$

Indeed, based on the definition of PO, the solution Ω_U has the following property:

$$\forall\Omega \quad \forall k \max_{P\in S(\Omega_U)} I_k(\Omega_U,P) \leq \max_{P\in S(\Omega)} I_k(\Omega,P) \leq \max_P I_k(\Omega,P) \tag{4.5}$$

The first inequality is based on the definition of the Pareto-optimality, the second one follows from the inequality 4.3. Given that this property is derived for any Ω, it holds for the solution of 4.2.

The interpretation of this statement is that the obtained solution of equation 4.1 is “dominating" any other security game over the same model with different gain functions for the players.

An abstract definition of the impact function allows us to select an appropriate way of calculating the impact functions depending on the domain of the SRA. In the next section, we develop a solver for a specific case of an impact function definition, similar to [48].

4.2 Alignment with Security Risk Assessment Methodology

The TPG definition is presented so as to ensure that the proposed structure uses as many definitions and entities as is possible from the SecRAM procedure. This chapter suggests an approach for the alignment of the SecRAM procedures and the TPG construction process, and a consequent solution of the problem in equation (4.1). It is important to note that we do not focus on the methods for the solution of equation (4.1), which means that the alignment is generic and relevant for any way of evaluating the impact functions $I_i(\Omega,P)$. It is presented in the next table.

TPG + Static Model	SESAR SecRAM
1. Internal model construction • PA identification (possibly, in a form of functional constraints) • Objective function definition • Supporting asset definition • Type of attack definition Output: • SA set structure • Links between PA and SA • Links between PA and impact functions • Attack operators 2. Impact matrix calculation. Output: 3D tensor $\mathscr{A} \in \mathbb{R}^{m\times M\times L}$ – values of every Impact Function (of m) for every type of attack (of L) for every SA (of M))	1. Asset identification and valuation • PA identification • Impact identification • SA identification Output: • Links between PA and SA • Table of impacts for every PA for every impact area • Threat scenarios 2. Impact evaluation Output: table of impacts for every SA for every threat scenario
External model development • Threat graph definition • Definition of control lists • Constraints, price definition Output: • Threat graph structure • List of possible controls	Likelihood evaluation. Output: Table of likelihoods for every SA for every threat scenario
Pareto-optimization (4.1). Output: List of optimal security placements Ω	1. Risk level evaluation. Output: Risk level for every SA and threat scenario 2. Risk treatment. Output: Security configuration

It worth mentioning that such an alignment may be performed for ISO 27005 too, as it is a basis of SecRAM. So, the steps “Internal model construction" and “Impact matrix calculation" can be considered as a part of the context establishment phase of ISO 27005, where main functions (PA), assets, and impacts are defined. Parts related to the definition of the TPG, controls, and limitations, aligns with the risk identification phase. Pareto-optimisation of security controls placement is risk evaluation and risk treatment phases, as it is done for SecRAM. The proposed alignment is undertaken so as to show the most significant similarities and differences between the methodologies. Some observations of the alignment are presented in the following list.

- The “internal model construction" and “impact matrix calculation" steps, which are related to the TPG and Pareto-optimisation, can be replaced by the corresponding SecRAM steps repeated for every objective function. As objective functions are closely related to some of the impact areas, it is possible just to perform the impact evaluation for these impact areas separately. Therefore, the development of the internal model could be replaced by the procedures undertaken by security experts.
- The significant difference between the approaches is located in the “external model development - likelihood evaluation" alignment. The “relation" between the attacker possibilities and the system structure in SecRAM are expressed in likelihood terms without any explicit definition of the domain, while the mathematical approach solves this using a threat graph. However, SecRAM builds an attack representation as a sequence of actions to be performed, which may be considered as an implicit attack model construction.
- The “Pareto-optimisation" step can be regarded as a formalised version of the corresponding actions in SecRAM: risk evaluation and treatment.

4.2.1 Weak Pareto-Optimality of the Security Risk Assessment Methodology

Considering further the similarities between the SecRAM and TPG model with equation 4.1 optimisation, it can be shown that, under several assumptions, the result of the SecRAM analysis and security control selection can be a part of the resulting PO frontier from the solution of equation 4.1.

Assuming that the impact areas from SecRAM correspond to the impact functions in the main PO problem, the overall impact from the threat scenario P should be evaluated as $\max_{i\in[m]} I_i(\Omega, P)$. Assuming that the threat response strategy “mitigate" is selected, the set of security controls is optimised to minimise the maximal impact among all threat scenarios:

$$\min_{\Omega}\left[\max_{P}\max_{i\in[m]} I_i(\Omega,P)\right] = \min_{\Omega}\left[\max_{i\in[m]}\max_{P} I_i(\Omega,P)\right] \tag{4.6}$$

Consider the solution of the problem stated above, Ω_{secram}. While it does not necessarily belong to the set of PO solutions, it can be proven that it is Slater-optimal or weakly Pareto-optimal.

Definition: (for minimisation problem) Let $S \subset \mathbb{R}^k, k \geq 2, \forall x \in S, \quad \|x\| < \infty$, where $\|x\|$ is any norm defined in $\mathbb{R}^k$. Then $p \in S$ is a Salter-optimal (or weakly Pareto-optimal) vector if and only if $\nexists x \in S : \forall i \leq k \quad x_i < p_i$.

Let us prove that this solution belongs to the set of weak PO solutions of the problem in equation 4.1. Suppose the opposite, from which it follows that $\exists \quad \Omega^*$:

$$\forall i \in [m] \quad \max_{P} I_i(\Omega^*,P) < \max_{P} I_i(\Omega_{secram},P) \tag{4.7}$$

Assume that $i^* = \operatorname{argmax}_{i}\left[\max_{P} I_i(\Omega^*,P)\right]$. Therefore,

$$\max_{P} I_{i^*}(\Omega^*,P) < \max_{P} I_{i^*}(\Omega_{secram},P) \leq \max_{P}\left[\max_{j} I_j(\Omega_{secram},P)\right] \tag{4.8}$$

For this reason, Ω_{secram} cannot be an optimal solution, as the maximal impact function for Ω^* is smaller than the maximal impact function for Ω_{secram}. This fact contradicts the definition of Ω_{secram}, as it is a solution that minimises the maximal impact function. Therefore, Ω_{secram} produces a weakly PO solution.

In view of the above, it is clear that, under reasonable assumptions, the result of the analysis from SecRAM may be obtained as one of the solutions of the TPG multi-objective model (in case Pareto-optimality is replaced by weak Pareto-optimality), assuming that the security experts plan the security configuration such that the overall impact is minimized (i.e., by selecting the “mitigate" strategy). It is interesting to note that the internal $\max_{P}$ operator, which is taken by definition in SecRAM, corresponds to a zero-sum security game, which serves as an additional justification of the selected game type. This result means that none of the obtained PO solutions of the TPG multi-objective model can produce lower impact values for all impact types. It can be used as a high-level verification of both of the approaches in case they are used in parallel.

4.3 Static Model Instantiation

This section presents an instantiation of the proposed TPG-based model, providing a description of the solver for a specific case of problem (4.1).

To complete the model definition, one needs to define a constructive approach for the calculation of the impact function depending on the strategies of the game agents and the TPG structure. For the presented model, the assumption is made that, if a control is placed at a particular checkpoint, then the adversary that performs the corresponding type of attack is blocked with some probability. Let $D \in [0,1]^{r\times L}$, where L represents the number of possible attack types. In addition, $D_{ij} \in [0,1]$ represents the probability that control i blocks the attack type j. Thus, the block probability for checkpoint u and attack type j is calculated as given below:

$$w_u(\Omega) = 1 - \prod_{k=1}^{r}(1 - D_{kj})^{\Omega_{ku}} \tag{4.9}$$

Formula 4.9 states that the probability of attack block is equal to the 1 minus the probability that all controls fail to block the attack at checkpoint u (node or edge).
Further in this section, we will consider the extended graph by default . For this reason, the attack type index will be omitted, and the entries of matrix Ω and block probabilities $w_u(\Omega)$ are further indexed by the extended graph edges.
The number of SAs in the expanded graph is, by construction, equal to the product of the initial number of SAs and the number of attack types, ML. We define a matrix of pure impact values 3.3 as $\mathscr{A} \in \mathbb{R}^{ML\times m}$ with entries a_{vi}, where i refers to the pure impact type index caused by an attack against asset v.
Variables $D, \mathscr{A}$, and the (extended) graph structure G are considered as inputs to the proposed game. The impact functions are evaluated as a probabilistic expectation of the loss:

$$I_i(\Omega, P) = a_{v(P)i} \prod_{e\in P}(1 - w_e(\Omega)) \tag{4.10}$$

where $v(p)$ denotes the corresponding SA node in path P, $a_{v(P)i}$ is an entry of the matrix $\mathscr{A}$, and w_e represents the block probability for the controls placed at edge e (0 if no controls). Therefore, the attacker's strategy concerning I_i is reduced to searching the attack path that maximises the expected impact.
Also, it is important to introduce resource limitations for the defender that correspond to the real-life constraints caused by financial or business reasons. Otherwise, the optimal solution to the problem could be placing security controls at each checkpoint of the (extended) TPG.

4.3.1 Single-objective Optimisation

Consider the problem of searching for the optimal security configuration for a single impact type i:

$$\min_{\Omega}\left[\max_{P} I_i(\Omega, P)\right] \tag{4.11}$$

By taking a logarithm of the expected impact expression, it is possible to convert the product to a summation as follows:

$$P^* = \underset{P}{\operatorname{argmax}} I_i(\Omega, P) = \underset{P}{\operatorname{argmin}}(-\log(I_i(\Omega, P)) \tag{4.12}$$

Using statement (4.10) and property 4.12, the process of selection of least-weighted path (attacker's strategy) P^* may be formulated as follows:

$$P^* = \underset{P}{\operatorname{argmin}}\left[(-\log(a_{v(P)i})) - \sum_{e \in P} \log(1 - w_e(\Omega))\right] \tag{4.13}$$

It is clear that the values of $\log(1 - w_e(\Omega))$ are non-negative because $w_e \leq 1$. The value $-\log(a_{vi})$ is assumed to be positive without loss of generality, as this may be achieved by multiplying every value in $\mathscr{A}$ by a constant that is sufficiently small. Therefore, this optimisation problem can be represented as a least weighted path search with positive weights in G from s to t. For an edge $e \in A$ that is not connected to any SA the corresponding weight is $-\log(1 - w_e(\Omega))$. Edges that are connected to SA nodes have weights $-\log(a_{vi})$.
Combining 4.9 and 4.13, weights for a checkpoint (k,l) may be expressed as below:

$$\log(1 - w_e(\Omega)) = \sum_{u<r} \Omega_{(k,l)u} \log(1 - D_u) \tag{4.14}$$

Here $\Omega_{(k,l)u}$ defines the Boolean components of the matrix Ω that correspond to the checkpoint $(k,l) \in A$.
We also consider the optimisation problem of $-\log(I_i(\Omega, P))$, where the inner max operator of 4.11 is converted into min, and the outer min operator is substituted with max. To reduce the settle point search to a standard optimisation problem, the inner least weighted path problem is replaced by its dual mixed-integer linear programming (MILP) formulation, as shown in [48].
The dual translation of the MILP problem is not applicable in a general case. However, the inner minimisation problem here is a classical least weighted path problem. The linear programming (LP) formulation of the least weighted path problem has a property that all corner points are integer-valued. Therefore, the MILP may be replaced by a standard LP problem and the dual replacement can be correctly used.
The specific case of problem (4.11), which uses the impact function definition described

above, may be stated as follows:

$$\begin{cases} \max\limits_{\Omega, P(y)} y_t - y_s \quad w.r.t. \\ \forall (k,l) \in A \quad y_l - y_k \le \sum\limits_{u<r} \Omega_{(k,l)u} \log(1 - D_u) \\ \forall v \in V y_t - y_v \le -\log(a_{vi}) \\ f(\Omega) \le F \\ g(\Omega) = 0 \end{cases} \tag{4.15}$$

Problem 4.15 has a definition similar to the dual shortest path problem. For each node k of the graph G a "dual variable" y_k is defined. Second and third inequalities state that the difference between the dual variables that correspond to connected nodes is not larger than the weight of the connecting edge. These inequalities are of critical importance for further analysis. It is easy to show that if node t is reachable from node s (exists a path in G from s to t), the difference $y_t - y_s$ is not larger than the weight of any path that connects those nodes. If there exists a path $P = (t, p_1, \ldots, p_n, s)$ then

$$y_t - y_s = (y_t - y_{p_n}) + (y_{p_n} - y_{p_{n-1}}) + \cdots + (y_{p_1} - y_s) \le \sum_{e \in P} d_e \tag{4.16}$$

Here d_e denotes the weight of link e, which may have a different form depending on the presence of SA node in this link. Obviously, the minimal value is reached when $y_t - y_s$ equals to the weight of the minimal weighted path between t and s. The last inequality and equation in 4.15 (which are not initially presented in (4.11)) define the resource (or price) limitations for the defender. The optimisation problem may easily incorporate inequalities such as $f(\Omega) \le F$ or $g(\Omega) = 0$ as long the vector functions f and g are linear. They may present business-flow limitations over the security configuration Ω, system parameters, or feasibility (i.e., cost) constraints.

The following property holds for the optimisation problem (4.15):

Lemma 1. *For impact type i,*

$$\begin{gathered} \forall \quad \Omega : f(\Omega) \le F, g(\Omega) = 0, \quad \exists Y, P : \\ \max_Y (y_t - y_s) = \min_P \big[-\log(a_{v(P)i}) - \sum_{e \in P} \log(1 - w_e(\Omega)) \big] \end{gathered} \tag{4.17}$$

Proof. The lemma is proven using a constructive approach, variables Y, P are defined explicitly. Consider any fixed feasible Ω. In turn, define a set of variables $Y = Y(\hat{\Omega})$, where each $\hat{y}_l, l \in N$ is evaluated as follows:

$$\hat{y}_l = \min_{P(s,l,G)} \sum_{e \in P} -\log(1 - w_e(\Omega)) \tag{4.18}$$

The weights $w_e(\Omega)$ are fully determined by Ω and the graph structure G. The next task is to show that $\left(\Omega, Y(\hat{\Omega})\right)$ defines a feasible configuration of variables for equation (4.15). This trivially follows from the fact that, for each $e = (k,l) \in A$,

$$\hat{y}_l \leq \hat{y}_k + \log(1 - w_e(\Omega)) \tag{4.19}$$

Indeed, the weight of the optimal path to node l is not larger than the weight of the optimal path to node k plus the weight of the edge (k,l). It should be noted that for each feasible configuration Y,

$$\forall P(s,t,G) \quad y_t - y_s \leq \sum_{e \in P} \log(1 - w_e) \tag{4.20}$$

Therefore $Y(\hat{\Omega})$ provides a maximal value for the optimised function. The statement of the lemma is satisfied by the construction of $Y(\hat{\Omega})$. □

4.3.2 Multi-objective Optimisation

Consider the MOOP in equation (4.1). For the search of solutions that belong to the PO frontier, an iterative epsilon constraint (IEC) algorithm [63, 16] can be used. The IEC algorithm iteratively solves a sequence of optimisation problems using an abstract solver (also referred to as an “internal solver"). These optimisation problems are stated for one of the objective functions with different constraints over other functions at every iteration of IEC. In the scope of the MOOP in equation (4.1), the internal solver shall meet the requirements listed as follows.

1. The resulting solution of the internal solver must be a solution of the following problem:

$$\begin{cases} \min_{\Omega} \max_{P} I_1(\Omega, P) \quad w.r.t. \\ f(\Omega) \leq F \\ g(\Omega) = 0 \\ \max_{P} I_i(\Omega, P) \leq C_i, i \in 2, \dots m \end{cases} \tag{4.21}$$

   where C_i are input parameters.

2. The solver must produce solutions that belong to the PO frontier, given the same constraints as are present in the optimisation problem in equation 4.21.

The next task is to construct the procedure for the internal solver, suitable for IEC algorithm, which has to solve equation (4.1).

Internal solver as mixed-integer linear programming. Consider item (1) in the abovementioned list of requirements. First, we reduce the problem in equation (4.21) to a MILP. Hereafter we assume that functions f, g are linear. The initial optimisation of equation (4.15) must be modified to take into account the constraints over the various objective functions. This may be achieved by the following MILP, which may be considered as a constructive statement of the problem (4.21):

$$\begin{cases} Optimisation \quad problem \quad (4.15) \\ \forall v : y_v - y_s \geq \max\limits_{i \in 2,\ldots,m} (-\log(C_i) + \log(a_{vi})) \end{cases} \tag{4.22}$$

Theorem 1. *The value* Ω *is a solution of equation system* (4.21) *if and only if it is a solution of the problem* (4.22).

Proof. The proof consists of two parts, each part is proven constructively.
1. Denote the solution of (4.22) by Ω'. Let us prove that Ω' is a solution for (4.21). First, we will show that the constraints $\max\limits_P I_i(\Omega, P) \leq C_i$ are maintained for each value Ω from the feasible region defined by the limitations of (4.22). From the structure of the optimisation problem (property (4.16)), we have:

$$\begin{gathered} \forall P(s,t,G) : v \in P \\ -\left(\sum_{e \in P(s,v,G)} \log(1 - w_e) \right) - \log(a_{v(P)i}) \geq \\ y'_v - y'_s - \log(a_{v(P)i}) \end{gathered} \tag{4.23}$$

Therefore, the statement $\forall v \in V, i \in [m] \quad y'_v - y'_s \geq \log(C_i) + \log(a_{iv})$ implies the following:

$$\min_{P, v \in P} (-\sum_{e \in P} \log(1 - w_e)) \geq -\log(C_i) + \log(a_{vi}) \tag{4.24}$$

Alternatively, the statement can be reformulated as follows:

$$\min_{P, v \in P} (-\log(I_i(\Omega, P)) \geq -\log(C_i) \tag{4.25}$$

The right part of the inequality does not depend on v, therefore it may be re-stated:

$$\min_{P} (-\log(I_i(\Omega, P)) \geq -\log(C_i) \tag{4.26}$$

Therefore, the feasible region of (4.22) is a subset of feasible region of (4.21). Further, using

Lemma 1, one can obtain that $\forall \quad \Omega$

$$\begin{gathered}\min_P(-\log(I_1(\Omega,P))) = \hat{y}_t(\Omega) - \hat{y}_s(\Omega) \leq \\ \hat{y}_t(\Omega') - \hat{y}_s(\Omega') = \min_P(-\log(I_1(\Omega',P)))\end{gathered} \tag{4.27}$$

Therefore, Ω' is a solution of (4.21).

2. Let us prove that, if Ω^* is a solution of (4.21), then it is a solution of (4.22). First, we show the feasibility of Ω^*. Let us consider the configuration $(\Omega^*, \hat{Y}(\Omega^*))$ defined by Lemma 1. From (4.21), we have:

$$\min_{(P(s,t,G))}(-\log(I_i(\Omega,P))) \geq -\log(C_i) \tag{4.28}$$

Therefore, $\forall P(s,t,G) : v \in P$

$$-\left(\sum_{e\in P(s,v,G)} \log(1-w_e)\right) - \log(a_{iv}) \geq -\log(C_i) \tag{4.29}$$

which implies

$$\hat{y}_v - \hat{y}_s \geq -\log(C_i) + \log(a_{iv}) \tag{4.30}$$

The left part of the inequality does not depend on i, therefore:

$$\hat{y}_v - \hat{y}_s \geq \max_{i\in 2,\ldots,m}(-\log(C_i) + \log(a_{iv})) \tag{4.31}$$

Thus, $(\Omega^*, \hat{Y}(\Omega^*))$ lies in the feasible region of (4.22). From Lemma 1, we have that for every feasible Ω:

$$\begin{gathered}\hat{y}_t(\Omega) - \hat{y}_s(\Omega) = \min_P(-\log(I_1(\Omega,P))) \leq \\ \min_P(-\log(I_1(\Omega^*,P))) = \hat{y}_t(\Omega^*) - \hat{y}_s(\Omega^*)\end{gathered} \tag{4.32}$$

This fact implies that Ω^* is a solution of (4.22), and so the proof is complete. $\square$

Pareto-optimality of the internal solver. Let us denote the solver for the problem (4.22) by $\Phi(G,A,C)$, where variables G and A define the structure of the proposed NSG extension, and $C \in \mathbb{R}^{m-1}$ denotes the vector of constraints for $I_i(\Omega,P)$. The solver outputs an optimal security configuration Ω for the given constraints and structure. However, neither of the equation systems (4.22) and (4.21) guarantees that the resulting solution belongs to the set of PO solutions, which is stated by requirement (2) given above. To ensure that the resulting solution belongs to the PO frontier, we propose an iterative approach for the instantiation of the solver, required by the IEC algorithm. The methodology is detailed as follows.

For the optimisation problem defined by graph G, impacts $\mathscr{A}$, and constraints C, the following steps are to be performed:

(a) $\Omega' = \Phi(G, \mathscr{A}, C)$

(b) Ω^* is defined as a solution of the following optimisation problem:

$$\begin{cases} \max\limits_{\Omega, P(y)} \sum\limits_{i \in [m]} \left[y_{t_i} - y_s \right] \quad w.r.t. \\ \forall (k,l) \in A, l \notin V \quad y_l - y_k \leq \sum\limits_{u<r} \Omega_{(k,l)u} \log(1 - D_u) \\ \forall v \in V, i \in [m] y_{t_i} - y_v \leq -\log(a_{vi}) \\ f(\Omega) \leq F \\ g(\Omega) = 0 \\ y_{t_1} = y_t y_v - y_s \geq \max\limits_{i \in 2,\dots,m} (-\log(C_i) + \log(a_{vi})) \\ y_t - y_s = \hat{y}_t(\Omega') - \hat{y}_s(\Omega') \end{cases} \tag{4.33}$$

In (4.33) additional variables are introduced, y_{t_i}, which can be considered as “stop" nodes for each impact type from 2 to m. This is a convenient augmentation, as the problem optimises the values of the impacts $I_i, i \in [2, \dots, m]$. The physical meaning of the system (4.33) is to “strengthen" the solution of $\Phi(G, \mathscr{A}, C)$. While $\Phi(G, \mathscr{A}, C)$ guarantees that $I_1(\Omega', P)$ is minimal, there is no guarantee that the rest of the impact functions are minimal too, they just fit the threshold. For this reason, (4.33) defines an optimisation problem that holds the $I_1(\Omega, P)$ at the optimal value reached by Ω' for the given constraints (last equality). It builds a feasibility region for Ω that lies within the feasibility region of (4.22) because it includes the constraints of (4.22). And, it tries to optimise the values of the rest of the impact functions. Impact I_1 is taken into account in the optimised function “artificially", it is fixed. It is introduced for further denotation simplicity.
Let the solver defined by steps **(a), (b)** be denoted as $\Psi(G, \mathscr{A}, C)$. The next step is to show that the resulting solution, $\Omega^* = \Psi(G, \mathscr{A}, C)$, matches the requirement list for the internal solver of the IEC algorithm. The general motivation underlying this is that, due to the discrete nature of the optimisation problem $\Phi(G, \mathscr{A}, C)$, the set of optimal solutions cannot be a single point, but it must be some finite number of possible configurations. Significantly, this is required to select the PO configurations.

Theorem 2. *The solution produced by steps* **(a)** *and* **(b)** *is a PO solution, given the constraints in* $\Phi(G, \mathscr{A}, C)$

Proof. The proof consists of 2 parts: first, some additional properties of the problem 4.33 are studied, after that the statement is proven by contradiction.

First, let us consider the properties of the solution of the problem 4.33. Using the same reasoning used during the proof of the lemma, it is possible to show that, for each of the summands from the optimised function, the following statement holds for fixed Ω (if optimisation is carried out only over Y variables):

$$\left[y_{t_i} - y_s\right] = \min_{P(s,t_i,G)} \sum_{e \in P} \log(1 - w_e(\Omega)) + \log(a_{v(P)i}) \tag{4.34}$$

This statement can be proven using constructive approach, when each y_i is set to the weight of the optimal path from node s in the graph. The same property holds for $y_t - y_s$, as it is optimal by definition of $\Phi(G, \mathscr{A}, C)$.
Next, let us consider a solution defined by the following procedure:

$$\Omega^* = \Psi(G, \mathscr{A}, C) \tag{4.35}$$

Corresponding values of the auxiliary variables are denoted as Y^*. Suppose that it is not PO solution, which is to say that $\exists \hat{\Omega} : f(\hat{\Omega}) \leq F, g(\hat{\Omega}) = 0$
with the following property:

$$\forall i \in \left[m\right] \max_P I_i(\hat{\Omega}, P) \leq \max_P I_i(\Omega^*, P) \tag{4.36}$$

In addition, there exists a j for which the inequality is strict. Let us define the corresponding optimal paths for the solution $\hat{\Omega}$ as follows:

$$\hat{P}_i = \operatorname*{argmin}_P I_i(\hat{\Omega}, P) \tag{4.37}$$

A similar definition is given for P_i^*. By taking product over i, it is possible to conclude that:

$$\prod_i I_i(\hat{\Omega}, \hat{P}_i) \leq \prod_i I_i(\Omega^*, P_i^*) \tag{4.38}$$

By taking the logarithm and applying the result 4.34, we obtain the following inequality:

$$\min_{P(s,t_i,G)} \sum_{e \in P} \log(1 - w_e(\hat{\Omega})) + \log(a_{v(P)o}) \leq \left[y_{t_i}^* - y_s^*\right] \tag{4.39}$$

with a strict inequality for t_j. One can easily show that $\hat{\Omega}$ is a feasible point of the problem $\Psi(G, \mathscr{A}, C)$. Therefore, by computing the optimal configuration of $\hat{Y}$ for problem $\Psi(G, \mathscr{A}, C)$, and applying result 4.34 once again, we obtain:

$$\sum_{i \in \left[m\right]} \left[\hat{y}_{t_i} - \hat{y}_s\right] \leq \sum_{i \in \left[m\right]} \left[y_{t_i}^* - y_s^*\right] \tag{4.40}$$

which contradicts the definition of Y^* as a solution of $\Psi(G,\mathscr{A},C)$, thus completing the proof. $\square$

Thus, the set of PO solutions may be obtained by applying the IEC algorithm with $\Psi(G,\mathscr{A},C)$ as an internal solver.

4.4 Results and Discussion

This section presents results on the validation of the static TPG-based model, as well as an interpretation of the obtained results. The aim is to demonstrate that general real-world relationships between input characteristics are retained by the model, and to show some of the basic properties of the approach. To achieve this, an evaluation was performed using a simplified Remote Tower and an airport model.

4.4.1 Airport Model Description

In this section, an example of the proposed approach for security configuration selection is presented. The tests are performed on a Remote Tower (RMT) system, using inputs from standard SecRAM techniques.
The novel concept of RMT aims to enable the remote control of incoming or outgoing airport traffic. Traditionally, airflow control operations undertaken by air traffic control services are handled in a local control tower. Thus, the RMT concept is to equip an airport with a set of sensors, allowing operators in air traffic control to gain a complete overview of the runway and other mission-critical aspects from a remote control centre. Notably, remote control would provide a possibility for one RMT to perform air traffic control for several small airports simultaneously.
RMT equipment consists of cost-efficient optical camera sensors, and the video images that these systems capture reflect a broad range of the aerodrome area. This online video stream is displayed on a video panorama in order to provide the visual appearance from that facility without needing a direct, out-of-the-window view. Airflow management is then performed using sensor information.

Definition of Primary and Supporting Assets

To develop the internal model of the RMT, the initial results of SecRAM were used. It is essential to define all PAs and related SAs for the following development of the internal

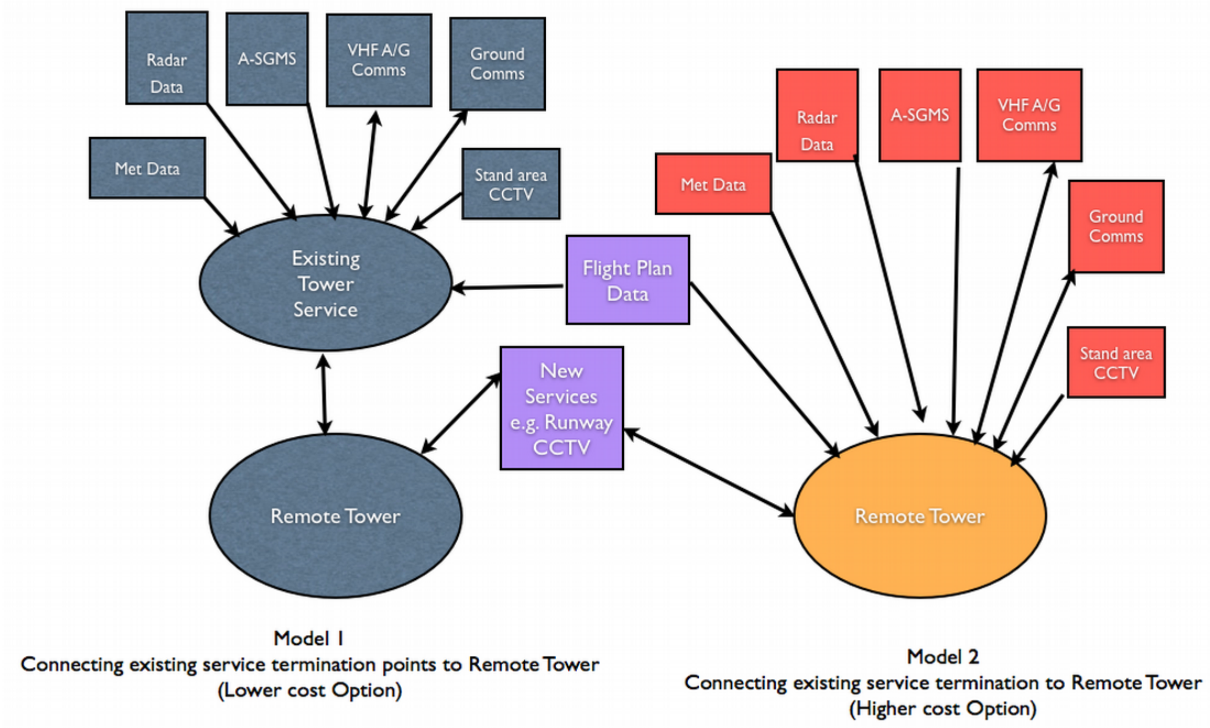


Figure 4.1: Scheme used for implementation of the airport threat model

model. Also, security impact functions should be defined. To follow the SecRAM approach, we use a subset of the impact areas defined in [1] as impact functions:

1. Safety (S) – Capability to provide all functions with minimal level of risk events.
2. Capacity (C) – Capability to provide all functions and services at the desired level.
3. Personnel (H) – Number of killed or injured people during functioning.
4. Economical (M) – Monetary loss in case of successful attack.

The list of SAs is presented schematically in Figure 4.1, consistent with the initial SecRAM document. In the next table, a correspondence between the SA identified in [1] and used in the current validation is established.

1. Remote Tower Facility: Corresponds to SA1 from the initial report and its sub-parts. This includes the air traffic controller working position, communication, information, control, flight data display, incident and distress alarms, visual tracking unit (VTU), and multi-display system.

2. Flight Information Service: Corresponds to SA2. Used to display and update met and operational flight information.
3. A/G Transmit/Receive Aerial Stations: Corresponds to SA3.
4. Aerodrome Staff: Corresponds to SA5.
5. Runway Visual Range Equipment: Corresponds to SA6. Consists of a set of cameras and optical sensors installed close to the runway.
6. Runway Approach Lights, Centre Line, Taxiway, and Stand Route Lighting: Corresponds to SA7 and all sub-parts.
7. Instrument Landing System: Corresponds to SA8 and SA8.1. Includes status monitoring for air traffic controller.
8. Aerodrome Visualisation System: Corresponds to SA9.1 and all sub-parts.
9. Binocular View: Corresponds to SA10. Includes a set of pan-tilt-zoom cameras for detailed observation of the landing and approaching area.
10. Transmission System: Corresponds to SA11 and all sub-parts. Consists of data between external sites (e.g. A/G coms sites and remote centre).
11. Radar Stations: Corresponds to SA12.
12. Audio Monitoring Stations: Corresponds to SA14. System consists of audio monitoring around the aerodrome and on tower.
13. Anemometer: Corresponds to SA15.
14. Processor of Station Identifier Tags for Visualisation Data: Corresponds to SA16.

To simplify the scenario, not all SAs from the original analysis are included in the modelling approach. SA17-SA24 and SA28 are related to the communication infrastructure between the tower and the airport. In the structure of the proposed TPG (as detailed in due course), these are observed on a higher level as, due to the selected attack types, no specification is needed.
SA25 and SA30 are considered to be a part of the overall hardware and software of the RMT, which is SA1 and SA10. SA26, consisting of the RMT and aerodrome building and facilities, is also considered to be out of the scope of the assessment as an attack that could damage the building is expected to be prevented before the airport is accessed. This is a decision similar to the SecRAM “transfer" option.
SA27 from the original report is considered to be a part of SA7 in this thesis. Additionally,

every PA is linked to the corresponding objective function, where every SA has a corresponding PA. This linking is presented in the table 4.1.

Primary Asset Description	Function	Supporting Assets, Names, and IDs	Impact Functions
Runway and Taxiway Visualisation Data	Enables air traffic control (ATCo) to exert effective control on the aerodrome surface	1, 4, 6, 11, 15	S, C, M
Air/Ground Communications	Link between ATCo and aircrew	1, 3, 11	S, C, M
Ground-to-Ground Voice Coms	Link between ATCo and aerodrome operators	1, 4, 11	S, C, M
Surveillance data	Provides approach and departure positions	1, 3, 7, 11, 12	S, C, M
Instrument landing systems (service itself or the system status data provided to the ATCo system)	Provides aircraft with precision approach aid	1, 3, 8, 11	S, C, M
Aircraft guidance systems for approach and taxi	Service assists the aircrew in moving the aircraft safely and safeguarding against incursions. Monitors provide ATCo with status data	1, 3, 7, 8, 11, 12	S, C, M
Runway visual range data	Key facility in determining aerodrome operating conditions	1, 4, 11, 6	S, M
Audio data	Provides ATCo with information on changing weather conditions or potential local environment issues	1, 4, 11, 13	C, M

Flight plan data, including airborne messages	Provides ATCo with planning data and rest of the system with flight activation messages	1, 2, 11	C, M
ASMGCS (if equipment in place)	Provides ATCo with a “picture" of aircraft position on the ground	1, 4, 9, 11, 15	S, C, M
Binocular view	Provides RMT ATCo with tools to inspect runway as part of issuing clearance to land or depart	1, 3, 10, 11, 15	S, C, M
Aerodrome meteorological conditions service	Service provides operating pilots with weather summary	1, 11, 14	S, C, M

Table 4.1: Links between supporting assets, primary assets and impact criteria, used in the model.

Attack Types and Impact Definition

All the PAs are related to information transfer from the RMT to the airport and backward. To define the set of required parameters, it is necessary to consider their relation to the CIA triad system-wide, as suggested by [2].
At the outset, the following definitions are introduced for the functional description of the PAs:

- $R^t(a)$ – Data rate (Mb/sec) of the information of type t incoming to ATCo, related to PA a.
- $L^t(a)$ – Delay (latency) of the information of type t transmitted from data sensors/sources of PA a, and received by ATCo.
- $R^t_{NC}(a)$ – Data rate (Mb/sec) of uncompromised information of type t incoming to ATCo, related to primary asset a. By “uncompromised information", we understand the information, which comes to a given communication node with no changes, as produced by the source node.

It is clear that $R^t_{NC}(a) \leq R^t(a)$.
Let us consider the CIA triad in the scope of the given definitions.

- Availability: To ensure system availability, several limitations should be satisfied. Namely, for any PA a, the data rate of given information type is higher than given limit: $R^t(a) \geq th^t(a)$. In addition, the total delay of information a is lower than the given limit: $L^t(a) < lim^t(a)$.
- Integrity: Provide accuracy and completeness of the assets. Fraction of the uncompromised data is not higher than the given limit: $(R^t_{NC}(a))/R^t(a) \geq th^t_{NC}(a)$.
- Confidentiality: Confidentiality of the information flow is not considered in the current RMT modelling.

Confidentiality is omitted from the analysis based on the considerations listed as follows.

- A significant part of the data is public and may be captured using simple radio equipment, such as ADS-B information, flight plan data, air/ground (voice) radio communications.
- Visualisation data, binocular view, instrument landing systems controls, and few other types of the data are not public. Obtaining that information may lead the attacker to the exact position of the aircraft and advice on the attack time. However, the current model operates under the assumption that the adversary has complete knowledge of the system attacked, so it would not add anything to the analysis.

Still, lack of assessment of confidentiality may lead to, for example, reputational losses under certain conditions. This may be represented in financial losses, too, if a specific type of attack is considered. Types of attacks considered in this thesis are targeting a direct impact on the performance of the airport/tower that leads to a decrease of safety and capacity of operations. For this reason, omitting the confidentiality of the data is acceptable.
Omitting the confidentiality for the RMT example still allows demonstrating the main properties and the behaviour of the static model. The confidentiality criteria for each PA is considered in an equivalent manner as integrity and availability. For this reason, adding confidentiality into the analysis could change the impact function values and, therefore, the set of PO solutions and optimal security control configurations. Additionally, if more attack scenarios are added, confidentiality consideration could change the topology of the underlying TPG. However, it does not change the modelling scheme conceptually.
A network topology is required to calculate the parameters $R^t(a)$, $L^t(a)$, and $R^t_{NC}(a)$ for each relevant information type. The data exchange network for the RMT was developed and is represented in Figure 4.2, where relevant acronyms are also introduced.

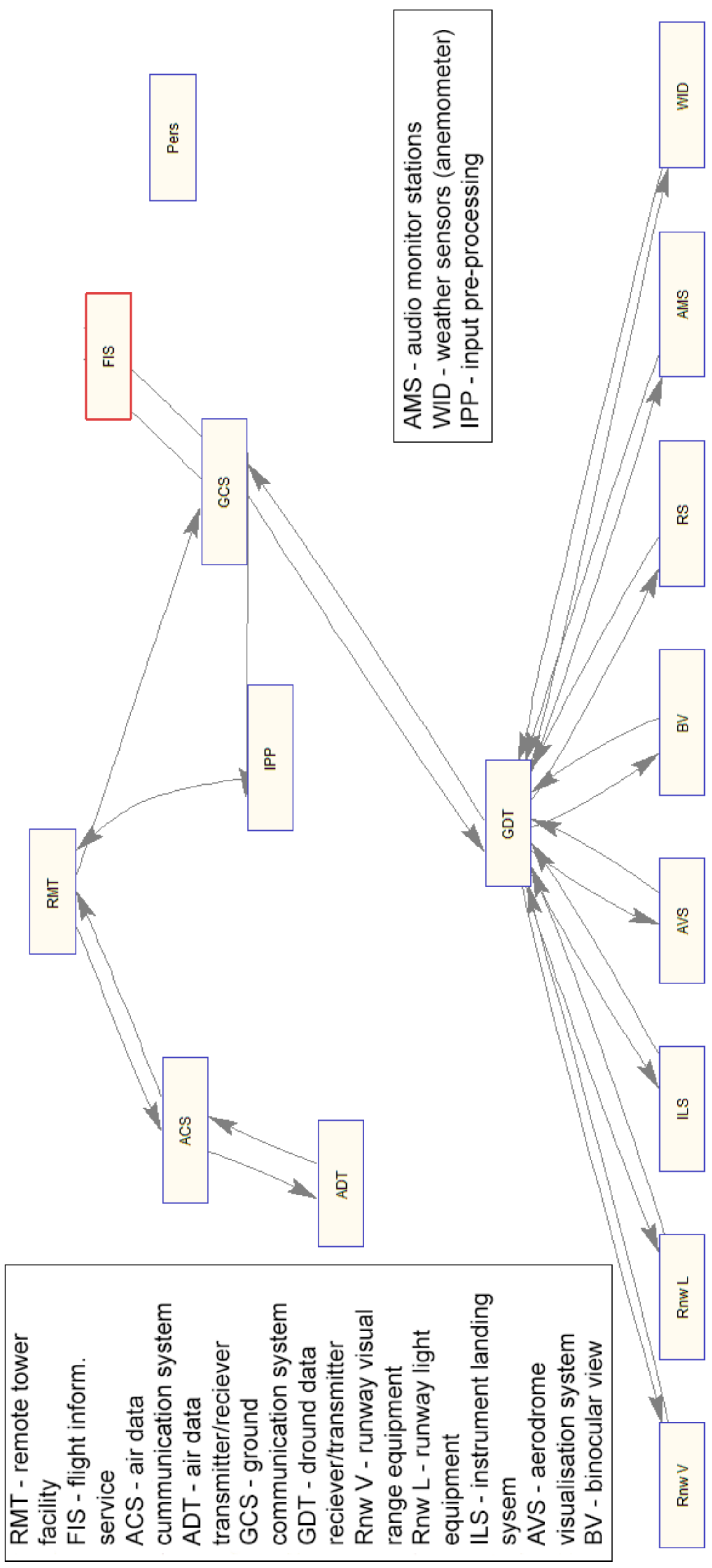

AMS - audio monitor stations
WID - weather sensors (anemometer)
IPP - input pre-processing
Pers
FIS
GCS
RMT
IPP
ACS
ADT
GDT
WID
AMS
RS
BV
AVS
ILS
Rnw L
Rnw V
RMT - remote tower facility
FIS - flight inform. service
ACS - air data cummunication system
ADT - air data transmitter/reciever
GCS - ground communication system
GDT - drround data reciever/transmitter
Rnw V - runway visual range equipment
Rnw L - runway light equipment
ILS - instrument landing sysem
AVS - aerodrome visualisation system
BV - binocular view

Figure 4.2: Network nodes are related to different SAs, and directed edges indicate information exchange

Figure 4.2 represent the topology of the information flows in terms of the supporting assets used to transmit the information. Ground data transfer (GDT) service of the airport collects the information from the airport sensors and sends it to the ground communication system of the remote tower. Ground communication system (GCS) accumulates the information from several possible airports and sends it to the input pre-processing service, which sends it to the RMT. In addition, GCS obtains the information about flight plans (planned routes of each flight) from the flight information service. Real-time aircraft communication from the RMT is done using air data transmitters/receivers. RMT controllers are able to control the airport sensors via the GCS and GTD.
The calculation of these values involves the parameters of the subsystems. As an example, a "binocular view" (BV) PA is considered. This PA corresponds to the video feed that air traffic controller receives from the pan-tilt-zoom cameras installed on the airport site. The air traffic controller should also be able to control the cameras remotely. This PA is required to provide the controller with real-time visual information of what is going on at the airport. Formulating in terms of information streams, this PA is related to the following services: firstly, information provision from the binocular to the ATCo; and secondly, control of the view parameters from ATCo.
The assumption is also made that the information is provided using the following "route":

- $BV \rightarrow ATCo$ ("video stream" (VS) information type): $BV \rightarrow GDT \rightarrow GCS \rightarrow IPP \rightarrow RMT$
- $ATCo \rightarrow BV$ ("control stream" (CS) information type): $RMT \rightarrow GDT \rightarrow GCS \rightarrow BV$

In this experiment, the set of limitations defining CIA for the BV PA is taken as follows [3]:

$$\begin{cases} R^{VS}(a) \geq 20Mbit/s \\ R^{CS}(a) \geq 10Kbit/s \\ (R^{VS}_{NC}(a))/(R^{VS}(a)) \geq 0.95 \\ (R^{CS}_{NC}(a))/(R^{CS}(a)) \geq 0.99 \\ L^{VS}(a) \leq 100ms \\ L^{CS}(a) \leq 100ms \end{cases} \tag{4.41}$$

The total $R^{VS}(a)$ parameter for the video stream is calculated as a minimum capacity of every link at the path ($BV \rightarrow ATCo$). The latency $L^{VS}(a)$ is calculated using "worst-case" estimation formulas, which involves the frame size parameter, the capacity of every link, and the latency at every node. The non-compromised data rate, $R^{VS}_{NC}(a)$, is estimated as a total data rate, $R^{VS}(a)$, multiplied by the "true data rate" of every node of the path. The true data rate depends on the error rate of the device or node, or on the character of the attack

performed.
Different attacks are intended to change the values of the internal parameter so that the limitations can be violated. For instance, the physical attack against GDT could increase the latency of the node, meaning that the total latency constraint of the BV PA would not hold.
Let us define the objective functions as a mapping.

- $C \in \{0,1\}$ – Represents system capacity. The system evaluation is performed in a Boolean manner. If the value is 1, then the system capacity is unacceptable, and if it is 0, then it is acceptable. The function is set to 1 if any of the CIA constraints of the PA related to this objective are violated.
- $S \in \{0,1\}$ – Represents system safety. The calculation approach is similar to the capacity.
- $H \in \{0,1,2,3,\dots\}$ – Represents personnel loss (i.e., the number of injured staff members). Occurs when a certain type of attack is performed against staff. The value depends on the path in the threat graph model.
- $M \in \mathbb{R}$ - Represents economic (i.e., monetary) loss. It depends on the type of attack and the attacked asset. In general, it can be calculated as an economic loss of the attack towards the SA summed with the prices of the affected PAs.

As pointed out above, the next step of the asset model instantiation is the definition of attack types against the system and the system's impact values. Types of attacks are established from the threat scenarios provided in [1]. Certain attack scenarios given in the report are not specific and may serve as an "additional bonus" to a specific attack being performed. Examples of such scenarios from [1] include "abuse of rights", "denial of actions", and "forging of rights". Taking these scenarios into account, and on the basis of security expert recommendations, the attack types are separated into two groups: internal and external. The group of the attack is essential during the development of the matrix D because some controls, despite being efficient against external attackers, are useless against internal attackers. Internal attacks are performed by an adversary that is a "part" of the system (e.g., a staff member or systems developer). By contrast, external attacks are performed by an adversary that is external to the system. A list of possible internal attacks, along with their impact on the attacked asset, is given below.

- Physical: It decreases capacity of the node (SA) to 0 Mbits/s, and increases latency to infinity. If the personnel SA are attacked, the number of human losses is equal to the number of people close to the SA. Economic loss is set at the price of the asset. This type of attack can be understood as a generalisation of the "blockade of facilities", "breach of personnel availability", and "theft/fraud and criminal damage" threats from the original SRA of RMT [1].

- Compromise of Information: This attack decreases the "true data rate" of the device to 0. There is no direct economic impact on personnel. In reference to [1], this can be understood as a generalisation of the "corruption of data", "data manipulation", "tampering of software", and "virus and trojan" attacks.

- Electronic Interference: This attack decreases the capacity of the node (SA) to 0. This type of attack may cause the destruction of the data transmission equipment, which could lead to the full replacement of the asset. During the modelling the economic impact is estimated to be half of the price of the asset that is attacked, which is a compromise value between "no damage" and "full replacement" cases. It relates to the following attack scenarios from [1]: "electromagnetic pulses or radiation", "denial of service", "viruses and trojans" (depending on the virus), "failure of telecommunications equipment", "failure of third-party service provision", and "hackers / social engineering".

The list of external attacks is the same, and the impact over the parameters is supposed to be the same. However, the separation between internal and external attacks is required because external attacks can be prevented by less "hard" controls, while certain procedural controls may only target internal attacks (e.g., staff monitoring).
The following attacks are not considered in the report:

- Indirect Disruptive Events: Attack scenario seems to be non-specific, which is the rationale for excluding it from consideration.

- Thermal Radiation: Omitted from consideration in [1] at risk assessment.

Threat Model

The report does not contain any detailed description of the steps taken by an attacker. Therefore, the TPG is developed based on expert suggestions from Eurocontrol.
The TPG threat model involves defining the structure of an attacker's actions. To develop the structure of the threat paths, it is necessary to investigate the system environment and structure. As described previously, two types of nodes were considered: physical nodes and cyber nodes, as proposed in 3.2.3.
Considering the physical nodes, the system can be separated into two parts: firstly, the part of the remote control operator interface; and secondly, the airsides of the controlled airports. In this thesis, a system with a single airside is considered. The airside of the airport is separated

into several sub-areas, which are related to different physical checkpoints.
Schematic representations of an airport and remote tower are presented in Figures 4.3 and 4.4. Bold borders of the rectangles indicate the airport's global sub-structures, including the airport building, the airside, and the staff entry. The thin border indicates the sub-areas related to the checkpoints in the threat graph. The graph's edges interconnect the “neighbour" sub-areas. The number of areas will be used further to indicate the checkpoint where a security control should be placed. If the asset is placed at the sub-area, then there is an access to the checkpoint with access to the asset. An additional checkpoint is placed “in front" of the asset so that the access to the asset is “individualised".

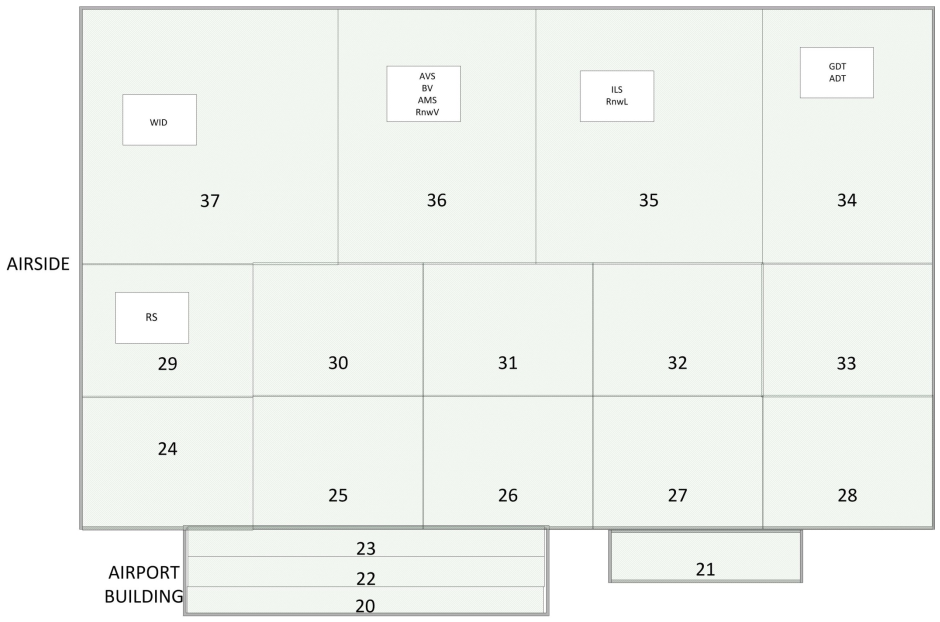


Figure 4.3: Scheme used for implementation of airport threat model

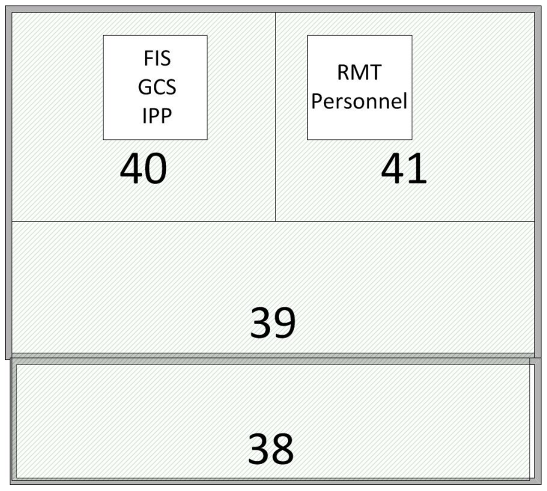


Figure 4.4: Scheme used for implementation of remote tower threat model

Nodes 20, 22, and 23 represent the separation of the airport building into nodes, and node 21 represents the staff entrance to the airside. Additionally, node 38 indicates the main entrance to the building, where the RMT operator interfaces are placed. Node 39 represents the “switch" between the server room and operator workplaces.
The cyber nodes were developed according to the network structure of the IT-kind supporting assets, as represented in Figure 4.2. The level of access was not specified because the proposed controls, which are discussed in due course, consider the structure of different access layers. If the asset related to the cyber checkpoint is placed at the given geographic position, access to the cyber-paths is possible from the given geographic checkpoint.
After applying the main steps of the graph transformation, the overall system threat model may be seen in Figure 4.5. In this figure, blue nodes represent cyber nodes and yellow nodes represent physical nodes. Each node number corresponds to a specific rectangle position in Figures 4.3 and 4.4.

As the next step, it is necessary to define the list of deployable and procedural checkpoints, as well as the block probability matrix D.
The presented list of controls is aligned with the security controls from [1]. As the original report operates at a much larger scale than the current model, only a part of the security controls were taken into account. Each of the security controls has constraints in terms of the installation node, the number of installations, and the different price.

- Authorisation Computer Procedure/Antivirus: Assumed to be effective (non-zero block probability) against compromise of information by an external attacker, labelled $C1$. Could be aligned with “anti-virus installation" and “data input credibility check" given in the original SecRAM RMT report.

- Checkpoint: Assumed to be effective against external physical and interference attacks, labelled $C2$. Could be aligned with “CCTV" and “automated access control" from RMT report.

- Full Security Scan: Assumed to be effective against external and internal physical and interference attacks, labelled $C3$. A combination of “guards", “barriers", and “automated access control" from the original report.

- Security Patrol: Assumed to be effective against external physical attack, labelled $C4$. It is related to “guards".

- Security Patrol (close to SA): Effective against compromise of information and internal or external physical interference, labelled $C5$. It is related to “guards".

In the list above, the effectiveness denotes a set of attacks with non-zero block probability. As the reader may be aware, certain internal attack types are not blocked by any controls. For this reason, two types of procedural controls are considered for enforcement. Procedural controls were added as an additional type of security control. These controls are assumed to “operate" between the nodes adjusted to the SAs and the SAs themselves. This formalisation was adopted because no cyber or physical location can be attributed to these controls. Nevertheless, they mitigate risk for a specific subset of SAs against certain attack types.

- P1, Staff Surveillance: Assumed to be effective (to a limited degree) against all types of internal attacks.
- P2, Enhanced Staff Surveillance: Assumed to be effective against all types of internal attacks, and more effective compared to the limited efficacy of P1.

Procedural controls are related to “personnel vetting" and “policy organisation & effective HR management" from the original report.

4.4.2 Experimental Results with Artificial Data

This experimental section of the thesis is separated into two different setups:

1. The first setup aims to complete SRA over the RMT system described in the previous sections. For this, a detailed description is given of the resulting Pareto-optimal frontier, and an analysis is undertaken of the solution of the system.
2. The second experimental setup aims to provide an analysis of the presented models, to compare these with the classical NSG, and to identify dependencies on the main parameters.

Remote Tower Threat Analysis

The results of the optimal security control assignment for the RMT model are described as follows. For modelling purposes, a number of different security controls are considered, and the corresponding results of the optimisation problems are listed. Two types of cost constraints are given: installation cost and maintenance cost. Thus, each security control is tagged by two types of costs. The costs for each security control are given in the table 4.2.

Security Control	Installation Cost	Maintenance Cost
$C1$	750	750

$C2$	1500	1500
$C3$	7500	2000
$C4$	750	750
$C5$	750	750
$P1$	5000	7000
$P2$	5000	14500

Table 4.2: Security controls' costs

As mentioned before, collected impact functions are highly correlated. Therefore, for all cases, the set of PO solutions contains just a single element. In the table 4.3, the results of the validation are presented, together with a discussion and analysis of the obtained results.

Constraints (Installation Cost, Maintenance Cost)	Optimal Solution	Discussion
(8400, 8500)	Resulting impact functions: • Security: 0.8571 • Safety: 0.8571 • Personnel: 5 • Economic: 0.4	Installation structure: • Deploy C4 at nodes 21, 20, and 37. • Deploy C1 at nodes 58, 60, 61, 62, 63, 65, 69, and 70. The current solution may be considered as the "worst-case" option in case of lack of resources. It is not possible to mitigate fully any of the attacks.

(16700, 16800)	Resulting impact functions: • Security: 0.4714 • Safety: 0.4714 • Personnel: 2.75 • Economic: 0.22	Installation structure: • Deploy C2 at nodes 21 and 20. • Deploy C4 at node 37. • Deploy C1 at nodes 58, 60, 61, 62, 63, 65, 69, and 70. Additionally, a procedural control of type P1 is enforced. The provided amount of resources is sufficient to mitigate all attacks, which reduces all impact types to a certain level. The main advantage is based on the enforcement of the procedural control P2, which reduces the internal attack probabilities.
(25000, 25300)	Resulting impact functions: • Security: 0.1714 • Safety: 0.1714 • Personnel: 0.5 • Economic: 0.04	Installation structure: • Deploy C2 at nodes 21 and 20. • Deploy C4 at node 21 and 20. • Deploy C1 at nodes 58, 60, 61, 62, 63, 65, 69, and 70. Additionally, a procedural control of type P2 is enforced. Due to the enlarged amount of resources, the mitigation level is increased even more. Mostly, it is achieved by using better procedural control and by adding more controls against physical attacks.

(50000, 50600)	Resulting impact functions: • Security: 0.1714 • Safety: 0.1714 • Personnel: 0.5 • Economic: 0.04	Installation structure: • Deploy C2 at nodes 22, 26, 27, 21, 19, and 20. • Deploy C4 at node 37, 28, 27, 26, 21, 20, and 19. • Deploy C1 at nodes 58, 60, 61, 62, 63, 65, 69,and 70. Additionally, a procedural control of type P2 is enforced. Despite the significantly larger amount of resources, no improvement is achieved in terms of impact function values. One way to account for this is by referencing the fact that procedural controls handle internal attackers, and the impact functions consider only the worst-case scenario. As the procedural control is the same for the previous case, no improvements are achieved. However, the average impact per asset should be lower due to the structure of the optimisation problem Ψ.

Table 4.3: Results of the static model validation

The experimental setup outlined in the previous table demonstrates that the security level improves with the growth of the amount of resources only up to a certain degree. After some threshold is reached, no improvement is possible. Adding more security controls and resources of any types can not improve the protection level due to the specifics of the formulation of the problem 4.2). It states a zero-sum game and only the attacks with the highest impact are considered. Some of the internal attacks are only mitigated with procedural controls. Therefore, as soon as the amount of resources is sufficient to deploy the best available procedural controls, it is impossible to minimise the impact functions by allocating more resources.

The level of improvement is dependent on the qualitative characteristics of the security

controls. The only improvement of the impact functions level may be achieved in case the security controls are enhanced or more security controls are added at different nodes.
Comparing the results of the model with the SecRAM results, a key similarity is that the “Guards" (related to $C4$ in the model), “Automated Access Control" ($C2$) and “Antivirus/Data Credibility Checks" ($C1$) are applied for all SAs in the scenarios with sufficient funds (security controls’ relation described in 4.4.1). However, security control $C3$ is not used anywhere, but the two reasons given below may account for this:

- “Barriers"($C3$) control is considered by the graph structure, assuming that the system is physically accessible only from a limited number of locations.
- “Guards" and “Automated Access Controls" are deployed wherever they are needed by the model (that is, in case of sufficient funds).

The model selects the procedural control, which mitigates all types of internal attacks. In the report, only “Personnel Vetting" control is enforced for all SAs, while the “Policies" control is implemented only for a subset of SAs (related to “Personnel"). The difference between the selected procedural controls comes from the simplification of the security controls used in the validation.
Compared to the standard manual SRA procedure, the proposed model provides several important advantages.

- It quantifies the protection level and resources required for the protection. Proposed model explicitly demonstrates that adding more resources into the security may not be effective if no new controls are considered because the impact functions are not reduced with the growth of available resources.
- The model automatises the process of risk treatment and risk re-evaluation. Security controls selection is done optimally with no manual analysis. In some cases, the model may be more precise by providing an explicit link to the optimal geographical position of the security control (which may be missing from the analysis).
- The model allows for a wide range of “what-if" simulations with no redevelopment of the most complex part: TPG. The validation demonstrates a range of possible thresholds with different risk levels. A similar type of analysis is demonstrated in terms of security controls efficiency in the following subsection. Such experiments are beneficial for cost optimisations or analysis of the security control’s efficiency.

Model Parameter Validation and Comparison

In this section, several experiments undertaken using simulated data are presented. The aim of the section is to illustrate the theoretical properties of the developed models.

The security configuration cost limitations give one of the key model parameters. Figure 4.6 shows the dependency between the cost limitation threshold and the minimal feasible impact, resulting as a solution of problem (4.11). The results are provided for two types of problems:

- For an approach with all attack mitigation probabilities taken from $[0,1]$ (i.e., “soft" model).
- For an approach with all attack mitigation probabilities taken from $\{0,1\}$ (i.e., “hard" model).

The charts in the figure demonstrate that, at low-cost thresholds, the impact value is comparable for both models. However, as the constraint is relaxed, the soft model assesses the impact in a significantly more conservative manner, decreasing as a partially constant function. Impact drops correspond to a new control added to the solution. At the same time, the hard model evaluates the system as fully-secured when there is at least one security control at every path in the graph. Therefore, the standard NSG neglects to consider the cumulative effect added by the security controls.

Figure 4.7 demonstrates the dependency of the impact function on security control efficiency, which corresponds to D_u parameters in 4.15. Efficiency parameters of the security controls grow linearly from 0 to 0.8 across 20 iterations, and at each iteration, an optimisation of a single impact function is performed. The resulting impact chart demonstrates a decreasing partially linear function with a variable slope. The linear form of the function is ensured by the form of equation (4.10). In a general case, the function would have a polynomial structure with an order equal to the number of controls at the “most vulnerable" path. In the proposed model, the order is one, which reflects that there is a single control that can block the most vulnerable directions (this is related to internal cyber-attacks). Slope change in the graph indicates that the adversary’s strategy (i.e., target selection) changes in line with the efficiency of security controls.

Evaluation of problem (4.1) is performed on the reduced number of impact functions. Impact function values pointed in the SecRAM document for the Remote Tower SRA are highly-correlated, which leads to a single PO solution. This is a positive result because it mitigates the problem of selecting of optimal solutions. For experimental purposes, the same RMT problem was considered with two impact functions. In particular, using a random number generator, the impact matrix was generated, thereby eliminating the correlation effect. Results for problem 4.2 are provided in the following table for fixed-cost thresholds.

Impact Function / Solution Number	Impact Function 1	Impact Function 2
Solution 1	0.1155	0.1585
Solution 2	0.1247	0.0950
Solution 3	0.1444	0.0841

The results above demonstrate the standard trade-off behaviour of the PO frontier.

Due to the structure of the extended graph construction approach, the problem size grows linearly depending on the number of attack types. Figure 4.8 presents the performance measurements for the airport model and demonstrates unstable growth in execution time growth with no visible increase in slope. This behaviour may be motivated by the tree structure of the graph. It leads to a linear increase in the number of edges (and, hence, equations) depending on the number of vertices.

4.4.3 Model Limitations

The proposed methods is intended to support the generic SRA decision-making process. However, several significant points were not considered due to the nature of the initial assumptions.

1. TPG is non-formal in a general case. The definition of nodes and arcs is a highly-heuristic procedure, and the influence of changing of the graph structure by adding nodes has not yet been investigated. However, in the case of the analysis of the detailed scheme of the system, the approach can be used with no such investigation, but this creates problems in terms of computational speed. A possible improvement or mitigation of this effect may be achieved in a way similar to attack trees, by creating a database of different typical sub-systems. Development of such a database may rely on already existing attack tree databases.

2. It is a complex procedure to estimate the performance of the attack operators. The dependence of the objective function on the parameters of the model is also an issue. However, the current state of the model allows working with the upper bound of the impact. Estimation of the impact may be enhanced and done in a more precise way by applying statistical means, which may be combined with the process of TPG database creation.

3. For complex systems, the reproduction of the impact functions requires complex modelling. Also, the influence of the security controls over the business processes is oversimplified and could be presented only in the form of a set of linear restrictions. However, some minor changes in the model are required to dealing with the last fact, and in significant number of cases, the business and cost limitations can be presented as linear constraints. Non-linear restrictions for Boolean matrix Ω may be re-introduced in the form of auxiliary variables. For example, disjunction function $\Omega_{ij} \vee \Omega_{kl}$ may be expressed in the form of the following linear in-equations over auxiliary Boolean

variable d_{ijkl}.

$$\begin{aligned} d_{ijkl} &\geq \frac{\Omega_{ij} + \Omega_{kl}}{2} \\ d_{ijkl} &\leq \Omega_{ij} + \Omega_{kl} \\ d_{ijkl} &\in \{0, 1\} \end{aligned} \tag{4.42}$$

“Not" function is linear. Therefore, the basis of all Boolean functions may be expressed in linear form. However, such an approach needs to be more effective, as it increases the MILP’s size.

4. It is assumed that only one attack against a single SA is performed. However, for some specific cases, the current model deals with multiple attacks by calculating only the maximal impact (a proof is provided in due course).

Let us consider the last item in the previous list, related to multiple attack types. Define the attacker’s strategy as a set of attacks performed simultaneously, namely $AT = \{P_1, \ldots, P_k\}$. Thus, the joint impact function can be defined as $I_k(\Omega, AT)$. This leads to the following MO optimisation problem:

$$\min_{\Omega} \left[\max_{AT} I_1(\Omega, AT), \ldots, \max_{AT} I_m(\Omega, AT) \right] \tag{4.43}$$

To limit the number of simultaneous attacks, the assumption can be made that $|AT| \leq K$. Therefore, the consideration approach depends on the definition of I_k and its dependency on AT. Two obvious definitions can be considered: firstly, maximal impact among AT; or secondly, a sum of impacts. For both definitions, the model defined by the proposed IEC algorithm may be applied directly with no changes. For this, with no lack of generality, it is assumed that $|AT| = K$. Detailed consideration of both cases and a proof of optimality of the IEC procedure is given as follows:

- Consider the “sum" joint impact function, defined as follows:

$$I_k(\Omega, AT) = \sum_{P_i \in AT} f_i\big(I_k(\Omega, P_i)\big) \tag{4.44}$$

Here f_i is any increasing real-value function. Thus, it follows that

$$\begin{aligned} &\max_{AT} I_k(\Omega, AT) = \max_{AT} \sum_{P_i \in AT} f_i(I_k(\Omega, P_i)) = \\ &\sum_i \max_{P_i} f_i(I_k(\Omega, P_i)) = \sum_i f_i(\max_{P_i} I_k(\Omega, P_i)) \end{aligned} \tag{4.45}$$

As the optimisation of a single-argument increasing function is equivalent to the optimisation of its argument, the IEC optimisation procedure is correct for the considered definition of the joint impact function.

- For the “maximum" joint impact function, it is obvious that

$$\max_{AT} I_k(\Omega, AT) = \max_{AT} \max_{P \in AT} I_k(\Omega, P) = \max_{P} I_k(\Omega, P) \tag{4.46}$$

When the joint impact function is structurally more complex, the proposed IEC procedure may not provide PO solutions. However, it does not limit the generality of the TPG and the definition given in (4.1).

4.5 Comparison with Cybersecurity Risk Assessment Models

In this section, the model described in this thesis is compared with several cybersecurity risk models discussed in 2.6. The instantiation phases of the models were discussed in 3.4, so the main attention is paid to the functionalities of the models, especially risk evaluation and mitigation mechanisms.

4.5.1 Hierarchical Risk Management using Evolution Graphs

The SRA methodology discussed in this section was introduced in [8] and the instantiation details were briefly introduced in 3.4.1. The risk mitigation mechanism of the model is discussed below.
Risk mitigation is performed within the scope of the evolution graph $Eg(I)$. This process is defined as “a step of the assessment that selects which elementary attacks should be stopped and, hence, which countermeasures should be implemented". Countermeasures $C(A)$ for elementary attack A may include changing the dependencies in the infrastructure graph $Ig(I)$ that result in the corresponding changes in $Eg(I)$; changing the initial access rights for the users; pruning the arcs labelled by A. The method mainly focuses on the pruning mitigation mechanism.
The overall risk mitigation criterion is defined as a selection of a minimal set of controls C that prevents the users from reaching the goal nodes. The set is considered minimal if it is impossible to select a smaller set of controls that stops all attacks in $Eg(I)$. The paper briefly mentions, but does not detail the algorithms for the selection of the cut. However, a formulation of a corresponding integer optimisation problem is not complex, as it has no significant differences from the standard graph-cut problem.

Comparing to the static model, the proposed method has a more flexible mechanism for the $Eg(I)$ adaptation as a security control, despite the fact that it is not detailed in the paper. The method introduces several measures for the efficiency of a set of security controls C, but those are focused only on the properties of C. The model does not define any user-related criteria on the price and business limitations for the deployed security controls apart from the minimality of the set. Security controls are assumed to have absolute efficiency, represented by a “Boolean" cut of the $Eg(I)$. No explicit impact values are used, as suggested in SRA standards.

At the same time, the proposed method allows for explicit modelling of multiple attackers, which is not directly possible in the static model. Some discussions on multiple attackers modelling are given in the previous section, but no explicit mechanism is introduced. In addition, the automatic generation of the graph $Eg(I)$ from $Ig(I)$ allows for semi-automatic evaluation of the security of the infrastructure at the design stage.

The main advantages of both of the models are summarised below.

- Evolutionary graph-based model advantages: explicit modelling of multiple attackers; possibility of semi-automatic $Eg(I)$ restructuring as a security control.
- Static model advantages: multiple explicit optimisation criteria, as suggested in SRA standards; cost-related constraints captured in the optimisation; no absolute-efficiency assumption of the security controls allows for “protection in-depth" modelling.

4.5.2 Quantitative Cyber Risk Estimation Methodology

The quantitative cyber risk estimation methodology (QCREM) [45] is described in this subsection and compared with the static model. It is based on the “compromise graph" description discussed in 3.4.2, which has a similar definition and construction procedure as TPG.

During the constructions of the compromised graph, for each edge, a non-negative weight vector is assigned. The weight vector defines the time required by the attackers with different skills to pass from one state to another. For example, one may define three skill levels: beginner, intermediate, and expert. This means that for each edge a three-dimensional time-vector is assigned. Some ways for assessing these values are provided in the paper. After that, a dominant attack path is identified for each skill level, which corresponds to the least-weighted path in the compromise graph from the “start" node to any “target" node.

QCREM allows for comparing two system configurations. In the paper it refers to a “baseline" and “improved" (with additional controls) system configurations. Graph construction and time estimation for the dominant paths is repeated for both of the system configurations. After that, a heuristic formula is used to assess the risk reduction level.

QCREM suggests a way to evaluate the risk reduction but provides no specific mechanism for security controls selection mechanism. Given the problem formulation, an approach similar to the static model may be used to do that (maximisation of the shortest path). However, the method still operates in terms of just one risk metric. A the same time, the static model defines the graph weights in a uniform probabilistic manner for all impact functions, which prevents a full replication of the skill-differentiation mechanism used by QCREM. The introduction of different weights for different impact functions is a technical task, but there are no guarantees that the results of the theorems will be valid.

- QCREM advantages: explicit attack skill introduction.
- Static model advantages: provides more than one measure of the risk; automatically suggests optimal security control configurations; cost and business limitations of the security controls taken into account; semi-automatic graph construction.

4.5.3 Cybersecurity Modelling Using Attack Trees and Vulnerability Indices

The cybersecurity framework analysed in this subsection was initially designed for power station protection [64] but can be generalised for supervisory control and data acquisition systems. The model focuses on password policies and port auditing.
The modelling scheme may be decomposed into the steps listed below.

- Identify attacker's objectives.
- Identify possible security vulnerabilities and formulate a set of and/or attack trees that describe the attack scenarios.
- For each leaf of the attack trees, determine cybersecurity conditions, compute the vulnerability index (detailed further).
- Compute the vulnerability index of each scenario and the overall system.
- Decision-making to improve system vulnerability.

The vulnerability index (VI) is a measure of how vulnerable a certain part of the system is. In order to evaluate that for each leaf, a cybersecurity condition ξ must be evaluated. It is computed using three conditions: lack of evidence of previous exploitations (1), advanced counter-measures are deployed (2), comprehensive password policies are enforced (3). ξ may take three different values.

- $\xi = 0.33$ if all conditions are satisfied.
- $\xi = 0.67$ if any two conditions are satisfied.
- $\xi = 1$ if less than two conditions are satisfied.

Next, a vulnerability index of each leaf G is defined as follows: $V(G) = \xi * \max(v_\alpha, v_\beta)$. Here $V(G)$ denotes the vulnerability index, v_α, v_β are measures of the strength of port auditing policies and password combination policies. The paper describing the model suggests some ways to evaluate those values given some parameters like password length and previous port audit results.
VI of a scenario is defined as a product of VIs of all leaves that are related to this scenario. VI of the whole system is identified as a maximal VI among all scenarios multiplied by the “mitigation" factor that is evaluated based on the deployed countermeasures enforced.
The method described allows to automatically evaluate the protection level for different configurations of predefined controls. Comparing the functionality of the static model and the method described above, it may be concluded that the overall functionality of the static model is wider but is achieved using different techniques. However, some of the mechanisms proposed in the paper may be re-used in the static model for the estimation of the effectiveness of the mentioned security controls. Both of the models are characterising the system's security using the worst-case scenario. ξ-value may be considered as a product of procedural controls in the static model, but with a higher degree of flexibility (different procedural controls may mitigate different attack scenarios).

- VI-based model advantages: semi-automatic evaluation of the effectiveness of two types of security controls.
- Static model advantages: wider range of SRA-related parameters are captured by the model; automatic selection of counter-measures based on multiple impact factors.

4.5.4 Network Security Risk Model

The network security risk model (NSRM) was introduced in [33] and its instantiation phase is detailed in 3.4.3. Here the risk evaluation and mitigation stages are covered.
Risk evaluation in NSRM is done by analysing the optimal behaviour of the adversary within the state transition graph. The expected return of the attacker from any transient state is denoted by J_i, where $i \in \{0, \ldots, n, n+1, n+2\}$ indicates the state number. Assuming an optimal behaviour of the attacker, J_i obeys the following rule:

$$J_i = \max_{u \in U^i} \left\{ G_i(u) + \sum_{j=1}^{n+2} p_{i,j}(u) J_i \right\}$$

U^i stays for the set of actions available at state i, $G_i(u)$ denotes the attacker's return from action u, $p_{i,j}(u)$ are state transition probabilities (briefly discussed in 3.4.3).
Defined rule for J_i may be rewritten in a vector form. The following denotation is introduced:

$$\pi(s_i) = \left\{ u \in U^i : u = \operatorname*{argmax}_{u \in U^i} \left[G_i(u) + \sum_{j=1}^{n+2} p_{i,j}(u) J_i \right] \right\}$$

Thus, the following vector equation is hold:

$$\mathbf{J} = \mathbf{G}(\pi) + \Phi(\pi)\mathbf{J}$$

where $\mathbf{J}, \mathbf{G}(\pi) \in \mathbb{R}^{n+2}$; the i-th component of $\mathbf{J}$ is J_i, of $\mathbf{G}(\pi)$ is $G_i(\pi(s_i))$. $\Phi(\pi) \in \mathbb{R}^{n+2 \times n+2}$, $\Phi(\pi)_{i,j} = p_{i,j}(\pi(s_i))$ for $i, j \leq n$. For states $n+1, n+2$ the only non-zero transition probability is for self-transition.
The method explains how the equilibrium equation can be solved and derives a formula for the gains of the attacker and defender.
The NSRM method allows for evaluating the security risks given certain parameters of the system. Most of the parameters rely on the estimations of the times required to “hack" certain components of the system, and times required by the detection mechanism to identify the attack and render the adversary harmless. Therefore, using given estimates it is possible to automatically quantify the cybersecurity risks of the system considered. In addition, the method allows for multiple impact types calculation. This was not detailed in the paper but it is obvious from the formulation of the equilibrium problem solution.
The static model also allows automatic evaluation of multiple impact functions. However, it does not provide any specific mechanisms for automatised selection of the efficiency parameters of the security controls, while NSRM relies on time estimates in order to calculate the probabilities of state transitions. Both of the methods can consider sequential attacks: NSRM by the means of model state combinations, the static model using M-SATAA algorithm. At the same time, the static model performs the automatic selection of security controls.

- NSRM advantages: automatic assessment of the efficiency of security controls.
- The static model advantages: automatic selection of the optimal security controls configuration.

4.6 Summary

In this chapter, a security configuration selection static model method was proposed. The method given in this chapter is based on the definition of the TPG offered in the previous chapter. It still operates with general entities in order to meet the requirements given in Section 2.1.4, which are necessary for an SRA support tool. Each step of the method is aligned with the corresponding SecRAM analysis phase.

The proposed modelling framework is based on the extension of the NSG formalism, and it deals with the MO optimisation technique. An algorithm for the solution was developed, and a proof was given of its correctness. Several of the model's theoretical properties were also considered.

The model was validated on the RMT project example, taking the original SecRAM evaluation of the security controls as a basis. The resulting configuration was compared against the security controls selected in [1], the RMT report. Additionally, simulations were used to demonstrate several of the model's theoretical properties.

The static model was compared against four different cyber SRA methods. For each of them functional advantages and shortages of the static model were identified.

The static model may be applied as a support tool for SRA and security risk mitigation. However, it does not consider the "in-operation" scenario for the system, namely the attack detection problem, consideration of which was a key motivation for this thesis.

The model was compared in terms of functionality with other competing high-level, generic cyber SRA methods. For each of the techniques, the static model has certain advantages and disadvantages. The static model performs an automatic selection of the security controls based on multiple impact types, which is not the case for any of the models considered. This brings the static model closer to the SRA standards described earlier. The static model provide means for capturing business limitations of the security controls and is not focussed on a specific types of controls.

At the same time, the model provides means for security control selection, but not for event detector selection, which should be aligned with the attack detection process. These problems constitute the focus of the next chapter.

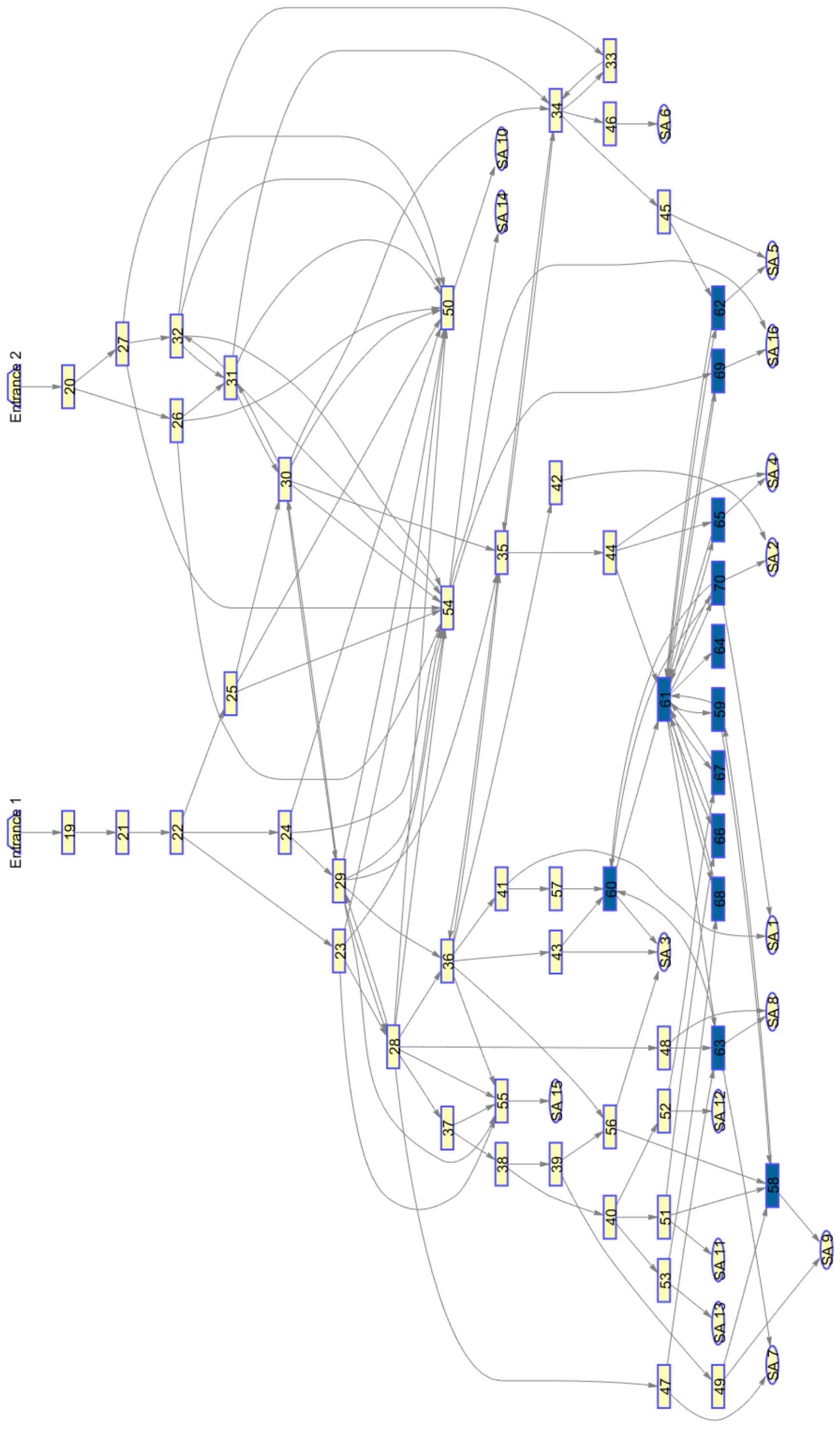


Figure 4.5: Threat graph model for remote tower

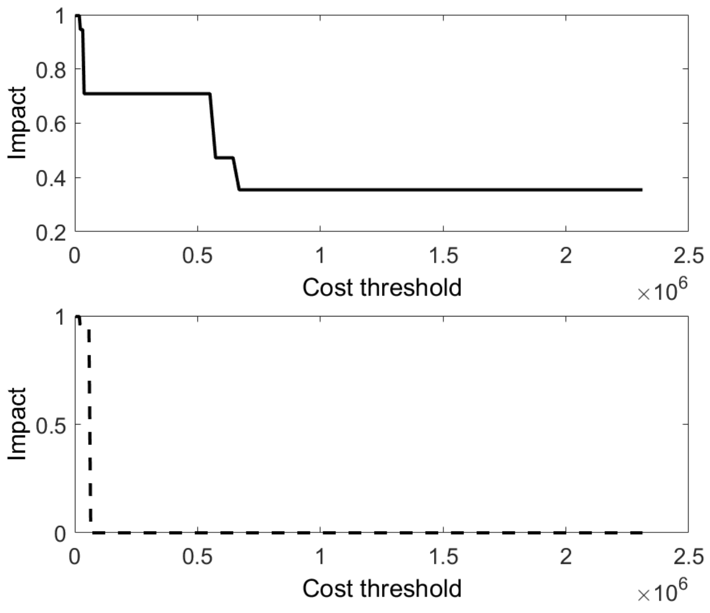


Figure 4.6: Minimal feasible impact depending on cost limitation. Top image (solid line) presents the dependency for the case of a “soft" model of security controls, while the bottom image (dashed line) presents the dependency for the “hard" model.

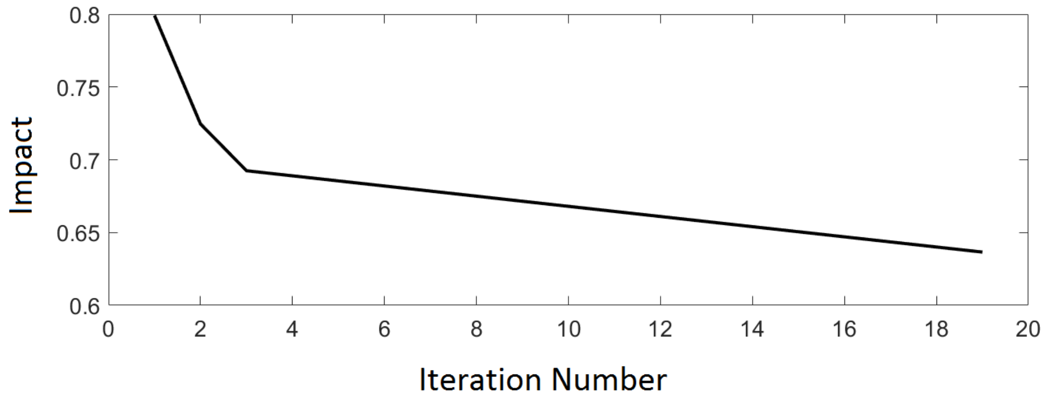


Figure 4.7: Dependence of impact function on efficiency of security controls

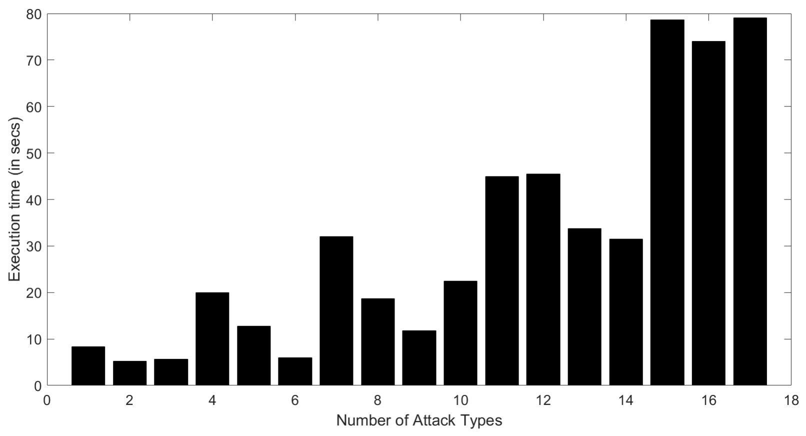


Figure 4.8: Execution time for problem (4.22) depending on the number of attack types

Chapter 5

Dynamic Security Model

The static model was introduced and described in the previous section. The aim of the static model is to perform the optimal assignment of security controls within the system. However, the static model neglects to consider ongoing “in-operation" development within the system and temporal dependency, which is critical for the continuous monitoring and situation awareness.

As the first step, it is necessary to provide a basis for this section’s discussion by defining the term “situation awareness". As defined in [28], the concept of situation awareness, as initially discussed during the First World War by Oswald Boelke, refers to the act of “gaining an awareness of the enemy before the enemy [gains] a similar awareness, and [devising] methods for accomplishing this." After some time, the definition of situation awareness was expanded at the system level. [71] pointed out that, to maintain an adequate awareness of system status, tracking the development of events as they gradually unfold is necessary. In the years since then, situation awareness has been defined differently in the literature, but almost all researchers define an agent environment, and refer to a perception of the goals and current status of the agents in time and space. The dynamic model introduced in the next paragraph addresses the problem of improving situation awareness within a system.

The dynamic model has the same grounding as the static model (i.e., the threat path graph, TPG). However, the main focus of the dynamic model is not threat prevention; rather, it is the detection of an ongoing attack, the estimation of the current “status" of the adversary, and the prediction of the set of possible targets. Therefore, the asset model part is not a significant element of the model’s definition because the impact estimation is not a priority (but still could be taken into account). The threat model is altered slightly to fit into the “event detection" domain. The system under protection is assumed to generate security-relevant information through a set of detectors. This information may be diverse, and in terms of its quality, it may also be heterogenous. Following the definition of situation awareness given above, this research’s operational definition of the main aim of the dynamic model is that it monitors relevant information and, moreover, addresses the following questions:

1. Is the system under attack?
2. How can event detectors be managed during security risk assessment (SRA), similar to security controls?

The second question can be considered an expansion of the SRA-support process, as discussed in the preceding chapter. It can be applied simultaneously with the static model optimisation process, but it requires a different apparatus for its formulation. Implementation of the dynamic model does not require any change in the approach used for TPG development. Therefore, it may be used in combination (or consequently) in order to accomplish the output of the static model.
The remainder of this chapter is organised as follows:

- The main additional entities and notations are introduced, and the questions stated above are formalised into specific probability estimation problems.
- BF-based dynamic model design and derivation is given for the estimation of the “state" of the adversary.
- BF-based model for “under attack" probability detection is derived, accompanied by theoretical analysis.
- Validation results over a cybersecurity intrusion detection dataset are provided, showing the performance of the model in relation to the two guiding questions for this chapter, as given above.

5.1 Threat Model Modification and Notations

To adjust the proposed TPG model for the purposes of attack detection and situation awareness, this section introduces several abstract entities. Consider a TPG, as defined in the previous section: $G = (N, A)$, with a set of nodes $N, |N| = n$, and a set of arcs A. The adjacency matrix of G is denoted as $\mathfrak{A}$. In addition, $E \subset N$ is the set of entry points, and $T \subset N$ is the set of targets. The letter t is used to denote time, and for modelling purposes, discrete time is used.
As the main aim of the model is attack detection, it is necessary to introduce the set of problem-relevant parameters which are to be estimated by the model. The adversary’s state space is given by $S = N \cup \{\emptyset\}$, where the empty set means that the attack has not started.
Let us formalise the information flow from the event detectors that may be received. Any data

source that produces any information that characterises the system state may be considered an event detector. For instance, consider two types of information: firstly, an alert from an intrusion detection system; and secondly, a “door opened" signal. Both signals can be used for incident investigation, and they complement each other. It is clear that the detection rate of “door-opened" signals is significantly higher than, and is not strictly related to, attack detection. However, if an intrusion does not happen without a door being opened, this signal may be useful in assessing false alerts from an intrusion system.
Formally speaking, the graph is equipped with a set of event detectors $\mathfrak{D}$. At each moment of time, the defender receives a set of detections

$$D_t = (d_1^t, \dots, d_{K_t}^t)$$

For each d_i, an associated vertex (or subset of vertices, given that the approach is easily extendable) exists, denoted as $v(d_i) = n \in N$. Each detection indicates the possible position of the adversary within the graph. Furthermore, each detector d is characterised by a set of parameters:

- True positive detection rate $p_{tp}(d)$: Probability of the detector sending information about an ongoing event in case an attack is happening.
- False positive detection rate $p_{fp}(d)$: Probability of the detector sending information about an ongoing event in case no attack is happening.

Returning to the “door-opened" example, in case the system has recorded an intrusion detection, there exist two combinations of events.

- Intrusion detection signal is followed by “door-open" signal.
- Intrusion detection signal is not followed by “door-open" signal.

The probability of the door being opened when no attack happens is quite high, so this type of detection alone does not give a lot of information. Assuming that the respective “door-open" detector is somewhat precise (which is a realistic assumption, given the simplicity of the event), the final false positive rate of the event is almost equal to the true positive rate. In case the intrusion detection signal is not followed by anything, the probability of this detection being correct is significantly reduced, as it implies a false negative condition for the “door-open" signal. At the same time, for the second combination, the probability of a true or false detection depends on the false positive rate of the “door-open" sensor.

5.2 Bayesian Filter Model

In this section, an exact probabilistic dynamic model formulation is described, along with the derivation of several important measures. It is assumed that the adversary is moving within the graph as a random Markov walk. At one step, the adversary may move to the neighbouring nodes of G. One may assume a uniform probability, but it is possible to embed a “motivation" parameter, which could parametrise the transition matrix.
The formalised problem is an estimation of $p_t(v_t, l|D_0^t)$, which is the probability distribution of the adversary’s position, given all detections obtained before and at moment t. To introduce an “ongoing attack" probability, an additional node is added to G, as discussed in due course. This family of models is referred to as Bayesian filters (BF), which is a baseline approach for object tracking problems [12]. Considering the proposed formalisation of the dynamic model, it may be considered as standard discrete-space tracking problem in the graph, hidden Markov model (HMM).

5.2.1 Model Derivation

Here, the case of a single adversary is detailed. The derivation is carried out using the belief propagation algorithm, which has an efficient implementation scheme for BF-type models: namely, incremental recursive estimation of the probability distributions using Bayes formula. The derivation relies on the probabilistic graphical model presented on 2.4. Hidden variables of the model correspond to the positions of the adversary within the graph (described by nodes of the graph), visible variables are the event detections. The goal is to derive a constructive method for the estimation of the position of the adversary at time moment t, given the observed detections at all times before t, detections obtained at time t, and probability distribution of the position of the attacker at time $t-1$: $p(v_{t-1}|D_0^{t-1})$.
The first step involves estimating a predictive distribution, which is then corrected using the data observed at the current moment of time.
Consider the joint distribution of v_t, v_{t-1}, D_t, given all detections before moment t, D_0^{t-1}:

$$p(v_t, v_{t-1}, D_t|D_0^{t-1}) = p(D_t|v_t)p(v_t|v_{t-1})p(v_{t-1}|D_0^{t-1}) \tag{5.1}$$

Therefore, “summing-out" the v_{t-1} variable, the probability distribution of v_t is defined as:

$$p(v_t|D_0^t) \propto p(D_t|v_t) \sum_{v_{t-1}} p(v_t|v_{t-1})p(v_{t-1}|D_0^{t-1}) \tag{5.2}$$

The term $p(v_t|v_{t-1})$ is assumed to be a pre-defined parameter of the model and is shortly discussed in the introduction of 5.2. In order to finish the derivation, let us define the detection likelihood function $p(D_t|v_t)$ as follows:

$$\begin{aligned} p(D_t|v_t) = \big(\prod_{d \in D_t : v(d) \neq v_t} p_{fp}(d)\big) \times \big(\prod_{d \notin D_t : v(d) \neq v_t} (1 - p_{fp}(d))\big) \times \\ \big(\prod_{d \in D_t : v(d) = v_t} p_{tp}(d)\big) \times \big(\prod_{d \notin D_t : v(d) = v_t} (1 - p_{tp}(d))\big) \end{aligned} \tag{5.3}$$

Equation 5.3 states that if the attacker is at node v_t, the probability of the observed detections D_t is defined as a product of following terms:

- Probabilities of false positive detection for all event detectors not at node v_t but present in D_t.
- Probabilities of true negative detection for all event detectors not at node v_t but present in D_t.
- Probabilities of true positive detection for all event detectors at node v_t and present in D_t.
- Probabilities of false negative detection for all event detectors at node v_t and not present in D_t.

Derived equation 5.2 and defined likelihood 5.3 allow to estimate the hidden variables of the model at each time step (referred sometimes as “iteration of the model").

5.2.2 Optimal Assignment of Event Detectors

This subsection considers the issue of optimal event detector placement. Optimality is defined in relation to the minimisation of the probability of an undetected attack. Two main sub-models are considered:

- Model with no false positives: For this model, only missed attack minimisation is performed.
- Model with false positives: For this model, we will consider an optimal event detector placement for best false alert filtering with constraints on real attack detection.

Initially, a theoretical analysis of the proposed framework is performed.

Theoretical Analysis of the Dynamic Model Framework

Let us consider the simplified model where no false or true positives are present. In addition, in this model, no security controls exist. In such a scenario, we iterate the system state, as presented before.

Let p_t be a vector of $p(v_t|D_0^t)$ for each node of the graph. The transition matrix $A(l)$ is defined by the structure of the graph such that each node of the graph defines a state in the transition matrix. For correctness, we define one additional state (or an additional node) v_{na}, which corresponds to the condition where the attack has not started. This node is “connected" to the entry nodes of the graph and has a self-transition probability a_{na}, which defines the probability that the attack will not start during the next time step. This node has no incoming links from the rest of the nodes in the graph. Furthermore, we refer to the augmented state vector $p_t \in \mathbb{R}^{n+1}$, where $p_t(v_{na})$, which is the probability of the adversary being at the additionally introduced node (i.e., no attack is happening and no attack has occurred). At that special node, no event detectors can be installed.

The formulas given in the previous section can be rewritten, for the simplified case considered here, using matrix notation:

$$p_t \propto \Gamma * A * p_{t-1} \tag{5.4}$$

Here, matrix Γ is a diagonal matrix with components equal to the false negative probabilities of detection for each node in the graph, which are defined as γ_{jj} for node $v_j \in N$.

Such an equation defines a recurrent definition of the state. We are interested in the static distribution, which defines the probability of the adversary remaining undetected until a given node in the graph, in case of an infinite observation period. One may define a “restricted" area in the graph such that the probability of the adversary reaching that area undetected is minimal. That subset may be related to some critical infrastructure area, where the system is most vulnerable (e.g., a firewall-protected sub-network).

Therefore, we are interested in the formulation $\lambda * p = \Gamma * A * p$, which leads to the eigenvector search for the matrix $\Gamma * A$. Here, it is easy to show that the lemma given below holds, as the proof below indicates.

Lemma 2. *If $p_{na} \neq 0$, then:*

1. *Matrix $\Gamma * A$ has an eigenvalue equal to a_{na}.*
2. *If p is a stationary distribution and $p(v_{na}) \neq 0$, then p is right-side eigenvector for the eigenvalue a_{na}.*

Proof. The lemma is proven directly, using the properties of the matrices A and Γ.

1. First, we prove the first statement that $\Gamma * A$ has an eigenvalue equal to a_{na}. Let us consider

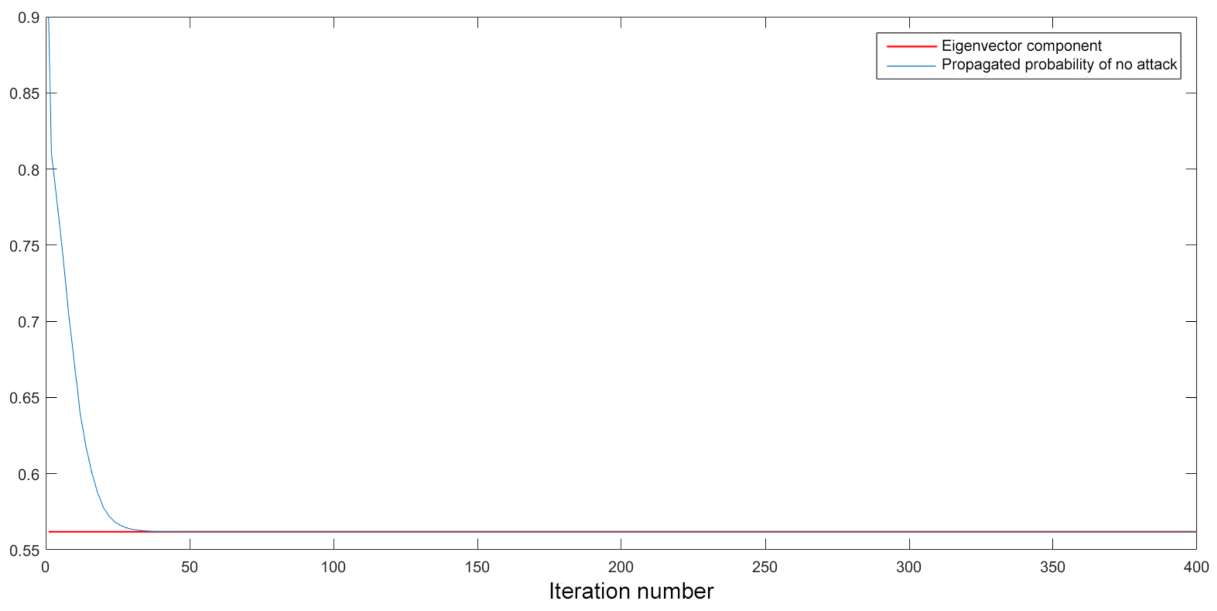


Figure 5.1: Propagation vs eigenvector component. It demonstrates the temporal convergence to the limit pointed in the lemma.

the structure of matrices A and Γ. As stated previously, the additional “no-attack" node has no incoming links and has a self-transition probability. This means that the last row of matrix A has one non-zero entry at the diagonal, which is equal to a_{na}. Γ is a diagonal matrix with entries corresponding to v_{na} equal to 1, and so the structure of the last row of $\Gamma * A$ is the same. Therefore, a_{na} is a solution to the characteristic polynomial

$$\det(\Gamma * A - \lambda * I) = 0$$

Therefore, a_{na} is an eigenvalue.
2. Next, we prove the second statement of the lemma. Consider the stationary distribution p, if it exists. Following the same idea as in the previous part of the proof, the component corresponding to v_{na} of the vector $(\Gamma * A * p)$ is equal to $a_{na} * p_t(v_{na}) = \lambda * p_t(v_{na})$. Therefore, if $p(v_{na}) \neq 0$, then it is correct that $a_{na} = \lambda$. □

Figure 5.1 outlines the evolution of $p_t(v_{na})$ through time with the corresponding component of $p(v_{na})$ with $p_{na} = 0.9$ for the TPG from the previous sections. However, the previous lemma only states the properties of the stationary distribution, which may not exist in a general case. The analysis of its existence is considered to exceed the scope of this thesis.

Impact of False Positive Events

We consider the model where event detectors may produce false positive events and no attack is ongoing. Figure 5.2 provides an example of the behaviour of $p_t(v_{na})$ for the false positive areas. One can see that the false events force $p_t(v_{na})$ to generate sudden drops, which rapidly disappear (i.e., in 2-6 time steps). However, the value $p_t(v_{na})$ represents the probability that the attack never started. Let us assume that the restricted area is defined as a 2-neighbourhood of the SA nodes. If we consider the same values for the restricted area and the SAs, we can state that:

- The baseline value for both of the areas is significantly higher than for v_{na}. Thus, the probability that the adversary is not at the restricted area is relatively high.
- The drops are significantly less wide and the probability of not being attacked is always higher for the restricted area rather than for the full graph.

Let us analyse the proposed model. For a moment of time t, the current distribution of the attacker's appearance p_t is defined by the sequence of matrices $\Gamma_0, \Gamma_1, ..., \Gamma_t$, where each matrix Γ_i is diagonal and represents a list of probabilities of true or false positive detections for each node, as derived earlier. Considering the expected value of p_t, given the “no attack" condition, the following recurrent formula may be derived:

$$\mathbb{E}p_t = \sum_{D_t} p(D_t|v_{na})\mathbb{E}\frac{\Gamma(D_t)Ap_{t-1}}{Z(D_t, p_{t-1})} \tag{5.5}$$

where the sum is taken over all possible configurations of values received from the event detections at time t. $\Gamma(D_t)$ represents the corresponding detection matrix for false or true positives. In addition, $Z(D,p)$ is the normalisation constant, which is computed as follows:

$$Z(D,p) = 1^T\Gamma(D)Ap \tag{5.6}$$

The expected distribution of the “no attack" case could be of benefit for attack detection problems, as shown in the experimental section of the thesis.
Assuming convergence in measure or in distribution $t \to \inf, p_t \to p$:

$$\mathbb{E}p = \sum_{D_t} p(D_t|v_{na})\mathbb{E}\frac{\Gamma(D_t)Ap}{Z(\Gamma(D_t), p)} \tag{5.7}$$

Let us analyse the meaning of $\mathbb{E}p$. It is clear that the component related to v_{na} describes the probability of no ongoing attacks in the case that observed detections are sampled from the distribution $p(D_t|v_{na})$. For the rest of the nodes of the graph the interpretation is similar. The component $\mathbb{E}p(v)$ describes the (average) probability that the adversary manages to reach node v by producing a set of event detections that fit the distribution of false positive reactions.

Due to the normalisation constant, the problem 5.7 cannot be converted into an eigenvalue problem, as was done for the case with no false positives. The expectation on the right-hand side of equation 5.7 might not be tractable, especially considering that the distribution over p is discrete. This follows from the fact that p represents a transformation of the multinomial binary distribution of the event detections. Therefore, an approximation of the random value parameter is suggested, which can be performed in the manner described in the following paragraphs.

Numerical Optimisation Techniques. To estimate $\mathbb{E}$, it is possible to use numerical methods. In this case, the following first-order approximation is considered:

$$\mathbb{E}\frac{\Gamma(D_t)Ap}{Z(\Gamma(D_t),p)} \approx \mathbb{E}\big(\frac{\Gamma(D_t)A\mathbb{E}p}{Z(\Gamma(D_t),\mathbb{E}p)} + \nabla\frac{\Gamma(D_t)A\mathbb{E}p}{Z(\Gamma(D_t),\mathbb{E}p)}(p-\mathbb{E}p)\big) = \frac{\Gamma(D_t)A\mathbb{E}p}{Z(\Gamma(D_t),\mathbb{E}p)} \tag{5.8}$$

Therefore, equation system 5.7 is transformed as follows:

$$\mathbb{E}p \approx \sum_{D_t} p(D_t|v_{na})\frac{\Gamma(D_t)A\mathbb{E}p}{Z(\Gamma(D_t),\mathbb{E}p)} \tag{5.9}$$

Equation system 5.9 defines a system of non-linear equations that may not have an analytical solution. For this reason, numerical solution methods can be applied.
One of the essential properties of the equation system 5.7 is that the number of equations is equal to the number of variables. Therefore, it is possible to apply Newton's method. Newton's method of equation system solution is equivalent to a minimisation of an auxiliary function using iterative second-order descent. Obviously, if $F(x): \mathbb{R}^n \to \mathbb{R}^n$, then

$$F(x) = 0 \iff ||F(x)||^2 = 0 \tag{5.10}$$

where $|| * ||$ denotes the L2-norm of a vector. The overall iteration of the method is based on the following first-order approximation:

$$||F(x+\delta)||^2 \approx ||F(x)+\nabla F\delta||^2 = ||F(x)||^2 + 2\langle\delta, \nabla F^T F(x)\rangle + \langle\nabla F\delta, \nabla F\delta\ \rangle \tag{5.11}$$

where $\nabla F \in \mathbb{R}^{n\times n}$. Assuming that the matrix is invertible, equating the derivative to zero gives the update rule:

$$x := x+\delta = x+(\nabla F^T\nabla F)^{-1}\nabla F^T F(x) = \nabla FF(x) \tag{5.12}$$

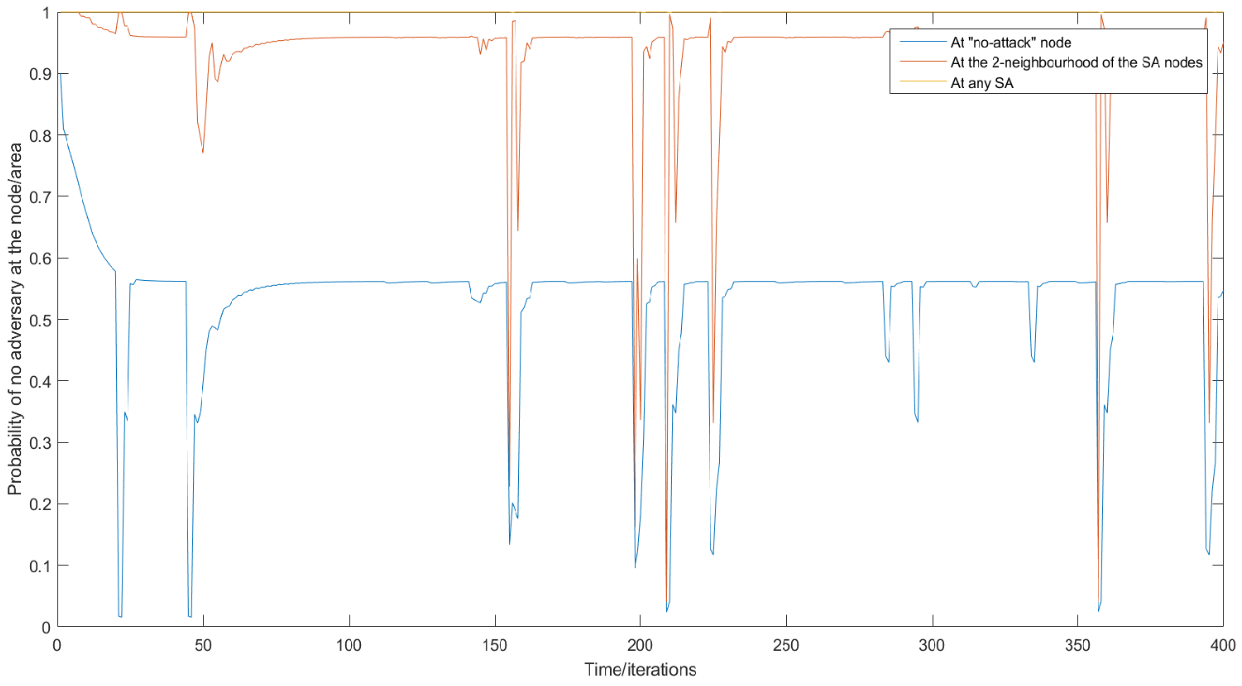


Figure 5.2: Propagation of “non-attack" probability with false positives

However, it is sometimes the case that the Jacobian matrix might not be invertible. In this case, the Levenberg-Marquardt method can be applied, in which the second-order term is regularised:

$$\delta = (\nabla F^T \nabla F + \lambda I)^{-1} \nabla F^T F(x) \tag{5.13}$$

where λ is a regularisation constant.

For a numerical solution of 5.9, additional limitations should be considered. One is the fact that the sum of the components of the solution is equal to 1. Another limitation is the non-negativity of the components of the solution p. These limitations define a set of solutions $\Theta = \{p \in \mathbb{R}^{n+1} | \langle p, 1 \rangle = 1, \forall i \in 1, \ldots, n+1, p_i \geq 0\}$.
Therefore, the update rule for the Levenberg-Marquardt algorithm can be adjusted using the gradient-projection method:

$$x := proj_{\Theta}\{x + \delta\} \tag{5.14}$$

Projection to the set Θ is a quadratic programming problem, which can be solved using standard tools.
The main drawback of the proposed approach is the need to compute the sum over all possible configurations at each iteration. If the number of nodes and event detectors is large, this action might be computationally expensive. Another drawback is that the derivation

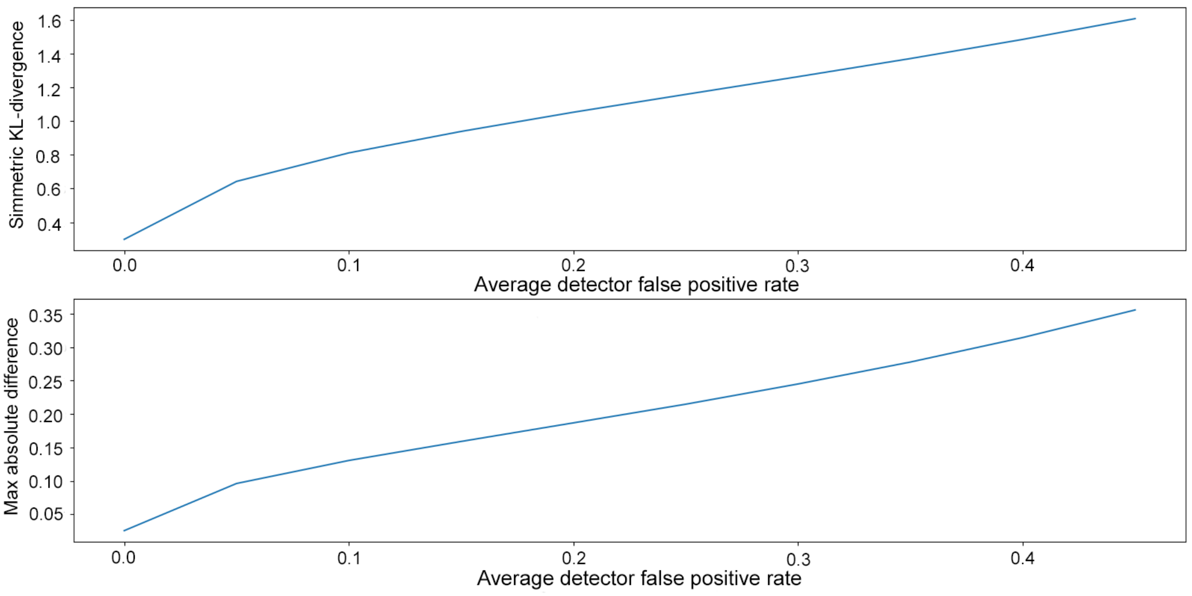


Figure 5.3: $\mathbb{E}p$ approximation quality, difference between sample estimation and Levenberg-Marquardt algorithm

of various statistics, which characterise the distribution of p, is complicated due to the requirement to present the time-propagation equation of type 5.7 and its approximations explicitly, as well as the gradients, and so on. The advantage of the method is its computational stability, which stems from the fact that it is not dependent on the detector parameters.

Sampling. Another approach used to estimate the expected value and the distribution of p (and other parameters) is the sampling method, in which the detections are generated according to their false positive rates. To estimate the expected value (or, for that matter, any other distribution parameter, such as variance or median), standard unbiased statistical assessments can be applied.

The sampling method has a significant advantage compared to numerical optimisation. This is because it does not require the summation over all possible configurations, which significantly decreases the computational complexity for large-scale systems. At the same time, variance of the estimator depends on the false positive probabilities. Assuming that the central limit theorem conditions hold, the size of the confidence interval depends linearly on the variance and decreases as a square root of the number of samples used for the estimation. Therefore, if the probability of false positives is large, the estimation may require a large number of samples. It may be possible to identify the exact number of samples using the confidence interval method for an unknown variance. For the approximation of the overall distribution of p, it is possible to apply density estimation methods such as kernel methods, Gaussian mixture models, and others.

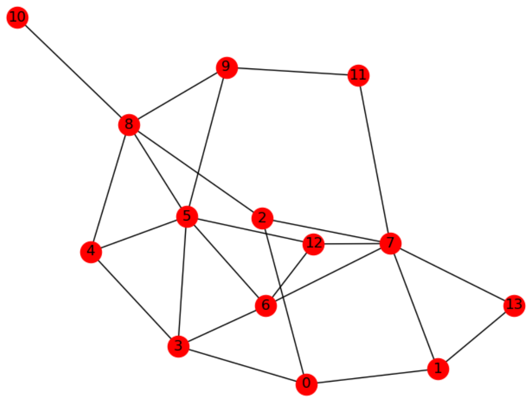


Figure 5.4: Graph used for dynamic model simulations, where the entry point is labelled with 0, and SA nodes range from 10 to 13

To evaluate the approaches for expected value approximation, numerical experiments were conducted in a simulated environment. For this purpose, a simplistic TPG-graph was constructed, consisting of 10 standard nodes and 4 SA nodes. A scheme for the graph is shown in Figure 5.4.

The experimental results indicated that the quality of the approximation depends almost linearly on the false positive rate of the sensors, as shown in Figure 5.3. Two metrics are given, which describe the dissimilarity between the expected value obtained using the Levenberg-Marquardt algorithm and the actual value obtained using sampling. One of the metrics shows the maximal absolute difference between the components of the distribution, while the second one represents the symmetric KL-divergence metric [12]. Monotonic growth with no signs of saturation may suggest that the approximation quality suffers significantly in case of large false positive rates of the sensors within the system. At the same, for all tested false positive values, the maximal absolute difference criteria remained within one sigma value (estimate).

Figure 5.5 demonstrates the projected samples of p over two SA nodes, as well as two projected expected value approximations: firstly, the sampling method; and secondly, the numerical method.

5.2.3 Attack Detection Method Assumptions

Proposed Bayesian filter model estimated the probability distribution of the attacker's position given the detections up to the current moment of time. Thus, this probability distribution p_t is a function of random values D_0^t, so it is a random value itself. Using the approximations of 5.7 it is possible to obtain some of the main statistics of the random variable of p_t for

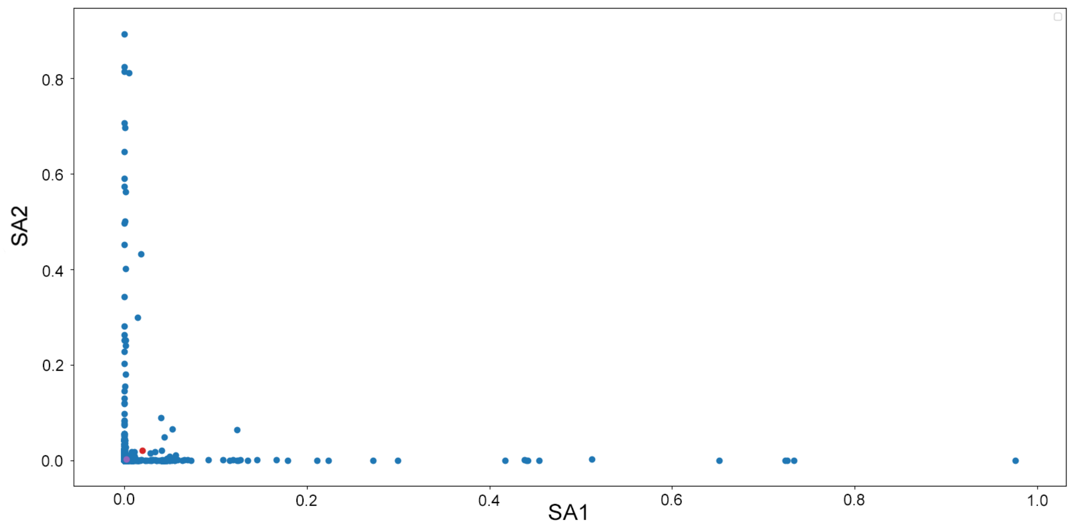


Figure 5.5: Sample projection (blue dots) and $\mathbb{E}p$ approximations for two methods: Sampling (violet dot) and Levenberg-Marquardt (red dot)

the case of non-attacks happening, such as expected value and the variance. Estimation of the probability distribution of p_t for the case of an ongoing attack should define different functions for different steps of the adversary. It can be performed in iterative manner. Suppose an estimation is done for an attack described by a path in the graph $V = (v_0, v_1, \ldots, v_M)$ that starts at time moment t_s. For time $t_s + k$ the attacker moves to the node v_k, and the corresponding probability of observing detections D_{t_s+k} at different nodes of the graph is defined by $p(D_{t_s+k}|v_k)$, as stated at 5.3.
The expected value of the distribution of p_{t_s+k} may be evaluated as follows:

$$\mathbb{E}p_{t_s+k} = \sum_{D_{t_s+k}} p(D_{t_s+k}|v_k)\mathbb{E}\frac{\Gamma(D_t)Ap_{t_s+k-1}}{Z(\Gamma(D_{t_s+k}), p_{t_s+k-1})} \tag{5.15}$$

A similar rule may be used to derive other relevant statistics. The formula is similar to the static distribution of the system for the case of no attacks. However, it has a few important key differences:

- The distribution is not static, and it evolves over time. Therefore, the inference should be repeated for each time step separately.
- Probability distribution $p(D_{t_s+k}|v_k)$ is different for each step k.

Therefore, the structure of the distributions for the cases of ongoing attacks is more complex than for the case of no attack ongoing.
A theoretically justified approach for attack detection would be to select such a path V that

maximises the probability of previously observed data. However, such a solution would require an estimation of distributions of p_t (or relative statistics) for all possible attack paths in the graph and comparing the observed values against those distributions at each moment of time. This procedure is complex and theoretically expensive. For these reasons, within the thesis, a simplified version of the attack detection mechanism is applied that analyses the components of p_t estimates related to the nodes of the graph, which are close to the targets. The metric used to detect an ongoing attack at time moment t is a probability that the adversary is present in one of the *critical nodes* of the graph, expressed by term $\langle w, p_t \rangle$. Here w is a vector of zeros and ones, $w(v) = 1$ if node v belongs to the critical section of the graph. The critical nodes are defined by the user. Possibly, it can be the whole graph without v_{na} node. The key assumption of the model is that components of p_t (except for node v_{na}) are significantly higher than $\mathbb{E}p$ in 5.7 in case of an ongoing attack. This assumption may be informally justified in the case of high true positive and low false positive rates of the detectors, which may lead to a significant difference between the distributions $p(D_{t+s}|v_k), v_k \in V$ and $p(D_{t_s+k}|v_{na})$. This fact may be heuristically justified by the following optimisation problems.

5.2.4 Optimal Event Detector Placement

In this section, the event detector placement optimisation problem is considered. It addresses question 2 from the initial model formulation given at the beginning of this chapter. Let us consider the vector $\mathbb{E}p$ from 5.7 and it's physical meaning. Intuitively, event detection allocation problem should minimise the probability of the attacker reaching critical nodes of the graph and producing a set of event detections indistinguishable from false positive reactions.

The main assumption of the modelling mentioned in the previous section may be formally translated to the optimisation process. If the probability of reaching node v with event detections indistinguishable from false positives is minimised, an ongoing attack would produce a completely different set of event detections that could be recorded by the system. This is not direct proof for the proposed approach, but an informal justification. Attack detection optimisation process is computationally complex and is not considered within the current thesis for the reasons mentioned in the previous section.

Therefore, optimisation problem can be considered with respect to the two main objectives:

- Minimisation of the probability of a successful undetected attack (for static distribution).
- Minimisation of the impact of false events in terms of probability drops (i.e., variance minimisation problem).

Minimisation of Probability of Successful Non-detected Attack. Let us consider the problem of minimal non-detection probability. The problem takes into account the static distribution of the system with no false positive events, and it minimises the probability of a successful attacker intruding into the restricted subset of nodes (similar to the critical nodes, but may be a different subset). Such a subset of nodes within the graph is denoted as $w \in \{0,1\}^{|N|+1}$.

Denote the event detection placement within the graph by $\mathfrak{E}$. Thus, using the results of lemma 2, the problem may be formulated in the form of mixed-integer linear programming as follows:

$$\begin{cases} \min\limits_{\mathfrak{E}} \langle w, p \rangle \\ \Gamma(\mathfrak{E}) A p = a_{na} p \\ \langle 1, p \rangle = 1 \\ p \geq 0 \\ F(\mathfrak{E}) \leq G \end{cases} \tag{5.16}$$

where $F(\mathfrak{E}) \leq G$ represents any arbitrary linear constraints over $\mathfrak{E}$ (e.g., cost or deployment limitations, maintenance costs, or business limitations).

By considering the case of a system with false positives, it is possible to conclude that the evaluation of the expected value of p and the respective non-detection probability is an iterative algorithm. This makes the evaluation of the loss-function non-trivial, and it complicates the application of the standard gradient or corner-point methods. However, it is possible to apply heuristic optimisation techniques, including evolutionary methods and the simulated annealing method.

Minimisation of Impact of False Events in Terms of Probability Drops. This quality function can be applied effectively in combination with the sampling probability estimation method. This is because it requires both the information about the expected value and the variance of the static distribution p. The suggested loss function can be formulated as follows:

$$\begin{cases} \min\limits_{\mathfrak{E}} \langle w_a, \mathbb{E}p \rangle + \langle w_v, \sqrt{\mathbb{D}p} \rangle \\ \mathbb{E}p = \sum_{D_t} p(D_t | v_{na}) \mathbb{E} \frac{\Gamma(D_t) A p}{Z(\Gamma(D_t), p)} \\ \langle 1, p \rangle = 1 \\ p \geq 0 \\ F(\mathfrak{E}) \leq G \end{cases} \tag{5.17}$$

where the parameters w_a and w_v represent the weights of the expected value and the standard deviation, respectively. To introduce a “physical meaning" into the optimisation function,

Node	False Positive	True Positive
1	0.1	0.8
2	0.1	0.8
3	0.1	0.8
4	0.2	0.9
7	0.2	0.9
8	0.5	0.9
9	0.2	0.9

Table 5.1: Initial attack detectors allocation and associated parameters

the following rules are suggested to be used for weight selection:

- The weight corresponding to a_{na} should be non-positive, while the rest of the nodes, especially the SA, should be non-negative.
- The weights of the variances should be set according to the following rule: $\forall i \in [N+1] \quad w_{vi} = n_{sig} w_{ai}$.

In this way, a lower bound of $\mathbb{E}p_{na} + n_{sig}\mathbb{D}p_{na}$ is maximised and the upper bound for the remaining nodes $\mathbb{E}p_i + n_{sig}\mathbb{D}p_i$ is minimised.
The stated optimisation problem was tested on the same simulated TPG presented in Figure 5.4. The following function was selected as the minimisation target:

$$(1-p_0) + \sum_{i\in[4,\ldots,9]} (\mathbb{E}p + 3\sqrt{\mathbb{D}p}) \tag{5.18}$$

The first term of the suggested optimisation function minimises an undetected attack probability and is computed directly via the previously described eigenvalue approach. The second term corresponds to the upper bound of the probabilities of the nodes, and it is computed using the model with false positive detections. Given that the sampling approach provides a means for variance estimation, it was applied to compute this part. Nodes $[4,\ldots,9]$ in the target function sum correspond to the set of nodes, which are directly connected to the SA nodes of TPG.
Initial placement of attack detectors in the model was set as presented in Table 5.1. The optimisation of the selected target function was performed over all possible replacement of the existing event detectors within the graph nodes $[1,\ldots,9]$. Simulated annealing was selected as an optimisation technique due to its simplicity.
For each iteration, Figure 5.6 demonstrates the evolution of the target function values throughout the optimisation process. Due to the stochastic nature of the algorithm, the value descent is not monotonic. However, simple replacement of the event detectors produced a reduction of the upper bound (and, therefore, the false-positive impact) by more than 15%.

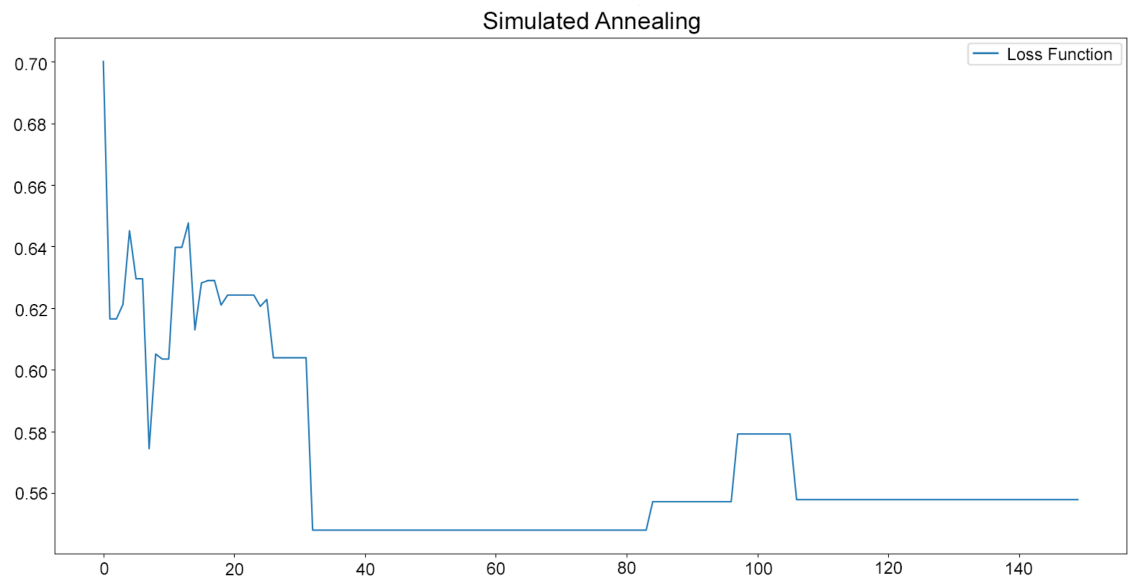


Figure 5.6: Simulated annealing optimisation process of the target function

Node	False Positive	True Positive
1	0.2	0.9
2	0.2	0.9
3	0.1	0.8
4	0.1	0.8
6	0.5	0.9
8	0.1	0.8
9	0.2	0.9

Table 5.2: Allocation of attack detectors and their parameters after optimisation

The produced event detector allocation is presented in Table 5.2 Figure 5.7 demonstrates the change of the detection variance $\mathbb{D}p$ for each node after applying computed allocation.

5.2.5 Attack Detection Algorithm

In this section, an algorithm for attack detection based on the derived Bayesian filter technique is described. This algorithm is a heuristic that relies on the assumptions described in 5.2.3 and is validated over a cybersecurity dataset later in the next section.
The key idea of the proposed attack detection algorithm is that some “attacker presence" probabilities p_t in the graph nodes are significantly higher than $\mathbb{E}p$ in case of an attack happening.
The model assumes that a system with a set of event detectors is observed. The information from the event detectors is streamed in real-time and the model analyses them and performs an online two-class “attack/no attack" classification. The key steps of the algorithm

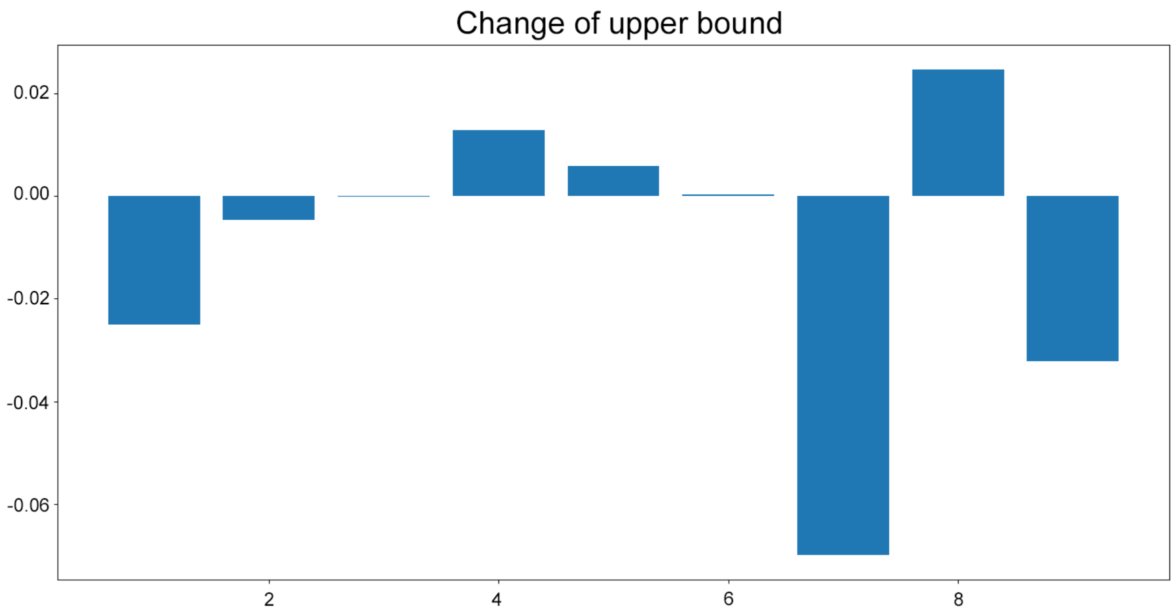


Figure 5.7: Change of variance $\mathbb{D}p$ after optimisation

initialisation are described in the following list.

- Build the TPG graph G for the system using the techniques described in 3. Add the auxiliary node p_{na} as explained at the beginning of this chapter.
- For each event detector within the system, assign it to a specific node within the graph. Each event detector should be characterised by its false positive and true positive rates.
- Optional step: optimise event detection placement using the approach 5.2.4.
- Select a set of “critical nodes". A node is considered critical if the presence of an attacker at a given node is unacceptable from the point of view of the security policy of the organisation. This may be established because the TPG nodes have clear references to specific physical/logical states of the system and adversary. For example, a specific node may represent a state of “root access to a server", which may be considered critical. This set is denoted by C.
- Estimate $\mathbb{E}p$ for G and event detector assignment. These statistics could be estimated using the techniques described in the previous sections.
- Assign the initial node probability state to its initial value p_0. In the case when no attack is expected to happen at the first iteration, the initial state could be selected as $p_{0,na} = 1$ and $p_{0,v} = 0$ for the rest of nodes v in G.

After the initialisation of the algorithm is complete, online classification is performed in two steps for each moment of time t.

1. Obtain event detections D_t. Evaluate p_t as given by 5.2.

2. Perform under-attack classification using the following rule:

$$cls_t = \begin{cases} noattack \quad if \quad \left(\sum_{v \in C} p_{t,v} - \mathbb{E}p_v\right)_+ < th \\ attack \quad otherwise \end{cases}$$

Here $(*)_+$ stays for a function that returns the input value if the argument is positive and 0 if the argument is negative. th is an input parameter for the model, which could be selected based on cross-validation, as shown in the next section. It may be selected based on the receiver operating characteristic (ROC) curve, which could be evaluated using attack simulations.
The intuitive meaning of the rule 2 is that it measures the increase of the probability from the expected one. If the total probability of the attacker's presence within the critical area is significantly higher than the expected assessments of the model, an attack flag is raised.

5.3 Model evaluation

Model evaluation was performed over the Intrusion Detection Evaluation Dataset 2017 (CICIDS2017) [58]. It is an open dataset available for download after filling the registration form. This dataset contains some of the most up-to-date and commonly seen attacks, and so it resembles real-world data (PCAPs). Additionally, the dataset includes the results of the network traffic analysis using CICFlowMeter with data exchange flows formed as typical machine learning feature vectors. Each recording (vector) includes the time stamp, source and destination IPs, source and destination ports, and protocol and attack types (label).
For the CICIDS2017 dataset, the abstract behaviour of 25 users was simulated based on the HTTP, HTTPS, FTP, SSH, and email protocols used for data exchange. The simulated attack types included brute force FTP, brute force SSH, DoS, Heartbleed, Infiltration, Botnet, and DDoS.
The following experiments were carried out over the selected dataset:

- Optimal event detector selection over limited resources.

- “Under attack" condition detection, based on the estimated variances of the attack flag.

At the moment of model development, there was a number of alternative datasets available on Kaggle [65] or Zendoo [50]. However, CICIDS2017 had several important advantages.

- It had a proper description of the system with multiple hosts, where SRA is more applicable compared to a single host network driver analysis like for Zendoo.

- CICIDS2017 includes a machine learning dataset in the form of comma-separated value files, which are convenient for detector training.
- It had raw network capture data that allowed for using additional non-ML event detectors, as presented in section 5.3.3. This is not presented in most ML-driven cybersecurity datasets like Kaggle.

In order to perform the experiments listed above, a TPG should be developed and additional event detector types should be specified.

5.3.1 Dataset Description

The CICIDS2017 dataset consists of a recording of network activity logs for five sequential working days (i.e., from Monday to Friday). The network topology includes a modem, firewall, switches, and routers. Network nodes have a variety of operating systems, including Windows, Ubuntu, and Mac OS X. During the recorded period of time, several attacks were undertaken against the network. Information about the network state was logged for each moment of time. In the experiments, the network consisted of an internal part (with two servers), which was bridged to the external network.
The attacks came from two agents residing in the external network with known IP addresses (for labelling simplicity). The dataset includes the following attack types:

- Brute force FTP attack using Patador Python script for brute force attacks. This type of attack potentially grants access to the data on the FTP server, or some portion of it.
- Brute force SSH attack using Patador. This attack may grant control over the victim machine, which – as before – is potentially partial depending on the installed policies.
- DoS attack using Slowloris and Slowhttptest tools. These tools implement “low and slow" attack types that are not easily detectable and use low bandwidth. Hulk script was also applied, which is easier to identify. Finally, the GoldenEye tool for security testing was used to attack the network. The aim of this type of attack was to render a specific service unavailable.
- Heartbleed attack (OpenSSL vulnerability) on port 444. The vulnerability is linked to a bug in the OpenSSL implementation, which allows internal information to be obtained from the server using so-called “heartbeat" messages.
- Various web attacks, including cross-site scripting (XSS), SQL injection, and brute force attacks. These attacks types may grant partial access to the data on the machine and partial control over the machine.

- Infiltration attack. This type of attack exploits vulnerabilities in any applications installed on the target machine. In the CICIDS2017 dataset, these applications are Dropbox, Meta Exploit, Windows Vista, and Cool Disk for Mac. Depending on the specific attack type, the attacker may achieve partial control over the machine and partial data access.
- Botnet attacks using ARES, targeting IoT devices. These attacks operate using scans for open debug bridges (usually Android devices, possibly virtual), as well as by uploading malicious software to the corresponding devices. This software can subsequently be used to recruit a device to the botnet, thereby gaining control over the machine's data or computational resources. Also, the botnet can be used to implement a DoS attack against the given network, but this is not the case for the current dataset.
- Port scan. An attack that sends client requests to a range of server port addresses on a host, trying to find an active port and, in turn, attempting to exploit a known vulnerability associated with the service.
- Distributed DoS attack using LOIC application. This attack has the same approach and same aim as DoS attack, but the flooding requests are sent from different hosts, which complicates the counter-action that a defender must take.

The dataset contains the timing information for each attack type, as well as the data about the IP addresses of the victims and attackers. The dataset is available on the following links:

- Dataset description form, where the dataset is explained and a download link is given.
- The dataset itself. Split into two parts: raw packet capture (pcap) files and pre-processed machine learning dataset. The raw data contains the information about all messages sent across the network. Machine learning dataset presents the information about the network data exchange in an aggregated form of feature vectors.

5.3.2 Event Detectors

The application of machine learning (ML) models is now widespread as a strategy for addressing cybersecurity and intrusion detection problems. For the proposed dataset, this methodology is well-suited, as the data is provided in the format of feature vectors, which is a classical input for ML model training.
The feature vectors are formed based on the available "*.pcap" files, which record network packet traffic. This detailed information is aggregated into vector form using the CICFlowMeter tool. Each vector corresponds to a communication session between two

given network nodes, representing a set of characteristics (e.g., session duration, total packets sent/lost). In total, 78 characteristics are measured. Each vector is labelled by a corresponding attack type or by a normal behaviour label (“benight").
Considering data representation in CICDS2017, intrusion (event) detection can be formalised as a classification problem. A classifier model forms a function from the input set X, which is usually a $\mathbb{R}^n$, to a *finite* set of output labels L. The training procedure can be formalised as a function mapping of the training dataset D (usually represented by a set of feature vectors, matrix) to a set of all classifiers [66]. This can be expressed as follows:

$$T : \mathbb{R}^{n\times M} \to (\mathbb{R}^n \to L) \tag{5.19}$$

Therefore, the recorded CICFlowMeter vector-form information can be used for ML-based event detector input. Due to the localisation of the scope of the vector (as a characteristic that describes a communication session only between two machines in the given period of time), the resulting event detectors could be applied on a per-machine basis.

Classification Problem

To select suitable event detectors, several state-of-the-art classification models were considered, including the following:

- Bagged decision trees or random forests
- Linear/kernel classifiers, support vector machines
- Neural networks
- Deep generative models

Detailed information about the quality of some of the most successful approaches is given at section 5.3.3. It provides information about neural networks (NNs) and deep generative models (DGMs) that achieve the best quality. Non-linear SVMs had incoherent convergence and a large number of kernels in the decision function, for these reasons they were ruled out. Bagged decision trees achieved significantly lower quality in both false positive and true positive rates compared to deep learning approaches. This result is aligned with the current state-of-the-art in ML, as in the last decade, NNs have grown significantly in popularity, showing top results in different ML contests in various domains. DGMs represent a “mixture" of NN models and probability distribution approximations by Bayesian inference. Therefore, for the validation of the dynamic model, NN-type classifiers were considered in this thesis.

One-class Classification

Another approach, similar to classification, which is applicable for the domain, is known as anomaly detection (often denoted “one-class" classification). Anomaly detection is widely applied to security and safety problems, including the field of cybersecurity. This class of models aims to distinguish “normal" system behaviour from abnormal behaviour, which displays outliers. For the CICDS2017 dataset, a possible application of one-class classifiers is to assume that the vectors with the “benight" label constitute a normal class, while all the vectors labelled by any attack type are considered outliers. The training of one-class classifiers does not require training over anomaly examples, which stems from the fact that the methodology creates a model only for the normal data and measures the deviation from that. This limitation decreases the precision of the modelled distribution of normal data, but it can capture anomalies (attack types for CICDS2017) that are not represented in the dataset. This leads to a potentially higher capacity to generalise [10].
In the scope of the selection of ML-driven event detectors, several of the most common types of one-class classifiers were tested: one-class SVMs, compression NNs, independent or principal component analysis, and variational auto-encoders. The tests demonstrated that the quality of the one-class models was significantly lower compared to regular classifiers. The best result was achieved with a variational auto-encoder, which had an attack detection rate of 45% and a true negative rate of 95%. However, these results are significantly outperformed by the classical classification model. Therefore, despite the potentially high generalisation capability, further consideration of one-class classification models was ruled out in this thesis.

Classification Neural Networks

NN models represent a construction concept for parametric models, where a superposition of linear (weights or convolution) and non-linear (activation) functions are alternated. This pair is often named as a *layer*. Formally, a neural network can be represented as follows:

$$F = f_n \circ W_n \circ f_{n-1} \circ W_{n-1} \circ \dots f_1 \circ W_1 \tag{5.20}$$

where n denotes the number of layers, and $\circ$ denotes a superposition operator.
The term “model training" refers to an optimisation of the loss function over the training dataset by the model parameters. Usually, the loss function is designed to be an upper bound of the error of the model over the dataset. In the case of NN models, optimisation is usually performed over the parameters W_i, while the rest of the parameters are considered to be fixed, or they are selected during cross-validation (i.e., hyper-parameters). NN training is usually based on different adaptations of stochastic gradient descent [29], when the partial

derivatives of the model are computed using the back-propagation algorithm.
Classification NNs are usually trained using cross-entropy criteria (CEC). CEC is a non-symmetric difference measure between two probability distributions, defined over the same domain. For two distributions $q(x)$ and $p(x)$, $x \in X$ the CEC is defined as follows:

$$CEC(p,q) = -\int_X p(x)\log(q(x))dx \tag{5.21}$$

In the case of the discrete probability distribution domain, as is the case for classification problems, the integral is replaced by a sum. For each classification sample $x \in \mathbb{R}^n$, the classification NN output defines a probability distribution $q(y|x) = F(y,x)$, where $y \in L$ represents a label of the output class. For the training samples $(x_i, y_i) \in D$, the target distribution may be defined as a one-hit vector $oh(y_i) \in \{0,1\}^{|L|}$, which represents the corresponding ground-truth class. The loss function for the NN is usually defined in the following way:

$$E(D,F) = \sum_{(x_i,y_i)\in D} CEC(oh(y_i), F(\cdot, x_i)) \tag{5.22}$$

This type of loss function accumulates (or averages) the cross-entropy loss over the whole dataset D.

Deep Generative Models

Generative models represent a family of methods that can be used to learn a data distribution using unsupervised learning. Generative models are widely applied in transfer learning, data sampling, and anomaly detection [29]. All types of generative models aim to learn the data distribution of the training set, and the typical application of the trained model is the possibility to generate samples. The main difference between the standard approaches for data distribution approximation is an application of NNs, which ensures the capacity to handle complex interdependencies in the data.
Two of the most commonly used and efficient approaches are variational autoencoders (VAEs) and generative adversarial networks (GANs). VAE training typically aims to maximise the lower bound of the data log-likelihood [38], whereas GANs aim to achieve equilibrium between the generator and discriminator [30]. However, as mentioned before, the one-class classification approaches did not yield high-quality results in processing CICDS2017. For this reason, it is necessary to consider a model that can be extended to a standard classification task. Such an extension is further demonstrated for VAEs.
Standard VAEs are trained so as to project high-dimensional data X to a latent space consisting of a smaller number of dimensions T. The concept is based on the maximisation of the likelihood of the training set $P(D)$ of independent data samples and simultaneous training

of the projection. This is performed in a way that is similar to the expectation-maximisation algorithm derivation (i.e., using log-likelihood decomposition), as shown below.

$$\begin{aligned}\log(P(D)) = \int q(T|D)\log(P(D))dT = \int q(T|D)\log\big(\frac{P(D)q(T|D)}{q(T|D)}\big)dT = \\ \int q(T|D)\log\big(\frac{P(D|T)p(T)q(T|D)}{p(T|D)q(T|D)}\big)dT = \\ \mathbb{E}_{q(T|D)}\log\big(\frac{p(D,T)}{q(T|D)}\big) + KL(q(T|D)||p(T|D))\end{aligned} \tag{5.23}$$

The term KL denotes the Kullback-Leibler (KL) divergence, which is a positive dissimilarity measure between two probability distributions. In certain special cases, it is equivalent (but not equal) to the CEC.

Therefore, considering the non-negativity of the KL divergence, the following lower bound can be derived from 5.23:

$$\log(P(D)) \geq \mathbb{E}_{q(T|D)}\log\big(\frac{p(D,T)}{q(T|D)}\big) = \mathbb{E}_{q(T|D)}\log\big(p(D|T)\big) - KL(q(T|D)||p(T)) \tag{5.24}$$

The inequality becomes an equality if $q(T|D) = p(T|D)$. Furthermore, q is named as an approximate latent variable distribution.

In VAE, two models are trained simultaneously, as shown below:

- The approximate distribution $q(T|D)$ is defined as an NN with parameters θ: $q_\theta(T|D)$.
- The distribution $p(D|T)$ is modelled using an NN with parameters ϕ, $p_\phi(D|T)$.
- The assumption is made that the dataset is independent such that, if $D = \{x_1,\dots,x_M\}$, then $p(D,T) = \prod_{i=1}^{M} p(x_i,t_i)$. The same assumption applies for $q(T|D)$ and $p(T)$.
- The distribution $p(T)$ is fixed, in continuous cases usually zero-mean Gaussian.
- The expectation taken over T at 5.24 is usually approximated using Monte-Carlo (MC) methods, by sampling from the corresponding distribution $q(T|D)$. Often, the so-called “one-shot" strategy is used for gradient calculation, when just a single sampled item is used for the approximation at each iteration. Thus, for gradient calculation purposes, the lower bound for a sample x_i is transformed into the following:

$$E(x_i,\theta,\phi) = \sum_{t_i^k \sim q(t|x_i)} \big(\log(p(x_i|t_i^k)) - \log(q(t_i^k|x_i)) + \log(p(t_i^k))\big) \tag{5.25}$$

Following the points above, the typical process for VAE training involves selecting a batch from the dataset D, selecting sample latent variables for each data instance from the batch, computing the loss and gradients of E with respect to θ, ϕ using equation 5.25, and performing the optimisation step (i.e., any type of gradient descent).

The described method is suited for distribution approximation problems, but not for classification. In classification tasks, the training samples are represented by pairs (x_i, y_i). To enable the classification using VAE, the following assumptions are made:

- The variables x_i and y_i are conditionally independent given the corresponding latent variable t_i, i.e. $p(x_i, y_i|t_i) = p(x_i|t_i)p(y_i|t_i)$
- The approximate distribution $q(t_i|x_i, y_i)$ depends only on x_i, and not on y_i.

Therefore, the sampled batch loss function 5.25 is transformed as follows:

$$\begin{aligned} E(x_i, y_i, \theta, \phi) = \sum_{t_i^k \sim q(t|x_i)} \left(\log(p(x_i, y_i|t_i^k)) - \log(q(t_i^k|x_i, y_i)) + \log(p(t_i^k))\right) = \\ \sum_{t_i^k \sim q(t|x_i)} \left(\log(p(x_i|t_i^k)) + \log(p(y_i|t_i^k)) - \log(q(t_i^k|x_i)) + \log(p(t_i^k))\right) \end{aligned} \tag{5.26}$$

The term E obtained at 5.26 defines the loss function for the classification VAE and can be used during training. All the distributions are defined in the same way as given in the explanations before 5.25. The term $p(y_i|t_i^k)$ is modelled using classification NN.

Due to the stochastic nature of the model, which stems from the use of MC sampling for each training batch, the convergence is slower than for standard NN models. To overcome this problem, a novel Nesterov-type gradient descent algorithm was applied, which is known as LAMB [73]. This algorithm stabilises the training batch variance and speeds-up the training process.

5.3.3 Results and Discussion

In this subsection, the experimental results over the CICDS2017 dataset are presented. At the outset, the quality of two selected event detectors, classification NN and VAE, is detailed. After that, the results of the application of the dynamic model over these event detectors are presented.

Quality Measurement

In this work, standard event detection quality measures were used (i.e., false positive rate, true positive rate, precision, and area under the ROC-curve, or AUC).

Quality measurement was performed via cross-validation with 10 iterations. At each iteration, the overall dataset was split into two parts: a training set (70% of the data) and a test set (30% of the data). The partition was performed randomly, but the following restrictions were considered:

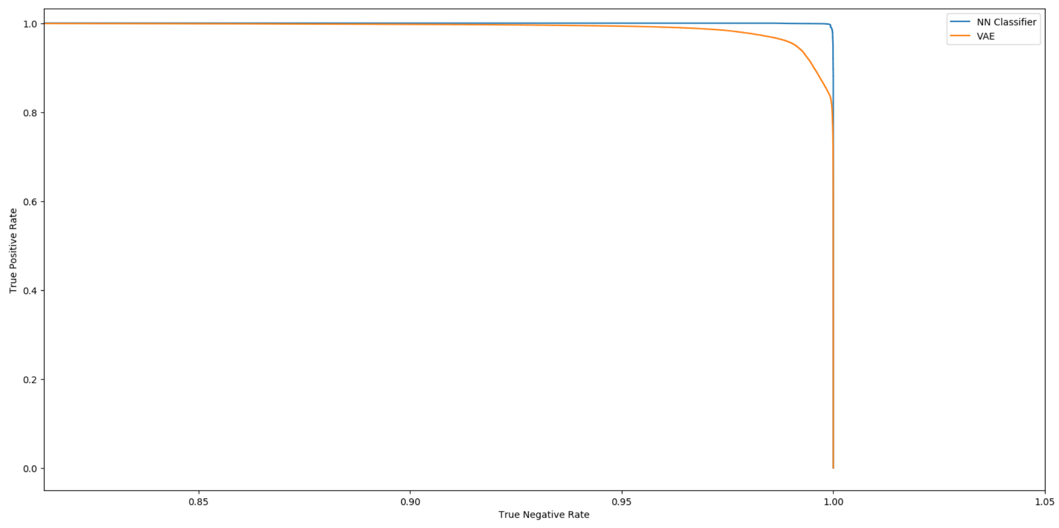


Figure 5.8: ROC curve for attack detection of classification neural network and variational autoencoder

- To balance the training and testing for different classes, the partition was performed individually for each attack type (and “benight" data for each date independently). Therefore, for each attack type, 70% of the samples were assigned to the training set, and the remainder were assigned to the test set.

- The vector type data in the dataset are presented in chronological order. Therefore, temporal dependency cam be considered for processing, which is a feature of the dynamic model. Also, the vectors representing close moments of time are situated proximately in the feature space. Training and testing over close data samples may bias the error assessment, leading to lower error rates on the training set due to data homogeneity. Therefore, the training and testing partition was established by selecting random sequences of data from different moments in time.

Event Detector Performance

Figure 5.8 presents detailed performance results for the selected models in terms of event detection (i.e., only “attack" and “no attack" classification). The figure also represents the ROC curves for both models, as built over the test samples.

As indicated in the graph, the average detection quality was high for both models, achieving more than 97% detection rate with less than 1% false positives. Precision and recall values for the fixed false positive rate of 3% are presented in Table 5.3.

The results demonstrated that VAE performance was less favourable compared to NN in terms of the average detection rate. However, this representation misses the per-label

Model	AUC	Recall	True Positive Rate	Precision
Variational Autoencoder	0.991859	0.969999	0.938469	0.984691
Classification Neural Network	0.998750	0.969999	0.996006	0.998992

Table 5.3: General precision metrics for selected models

Attack Type	VAE True Positive Rate	VAE AUC	NN True Positive Rate	NN AUC
Web Attack Brute Force	0.864333	0.981523	1.0	0.999646
Web Attack XSS	0.9492386	0.990792	1.0	0.999397
Web Attack SQL Injection	1.0	0.995415	0.363636	0.9205171
DDoS	0.998646	0.999595	0.9998958	0.9999207
Port Scan	0.9985727	0.99971	0.9999790	0.9999817
DoS Slowloris	0.912119	0.987601	0.9954049	0.998552
DoS Slowhttptest	0.9981862	0.998693	0.9915356	0.9948188
DoS Hulk	0.97460855	0.99608	1.0	0.9999713
DoS GoldenEye	0.9854274	0.997484	1.0	0.9999833
Heartbleed	0.83333334	0.881489	0.6666667	0.8798783
Infiltration	0.72727273	0.959772	0.18181818	0.8998308
Port Scan	0.997964067	0.999182	0.99995802	0.9998049
BotNet	0.255499	0.899293	0.983079	0.992576
SSH-Patador	0.457110	0.930525	0.987020	0.995782
FTP-Patador	0.998323	0.998343	0.903983	0.989259

Table 5.4: Per-attack precision metrics for selected models

classification quality, which is given in Table 5.4.

As shown in the table, VAE outperformed NN for several attack types. It is essential to mention that these attack types were represented with a small number of samples in the training dataset. Comparison of tables 5.4 and 5.3 serves as empirical evidence that the NN classifier requires more balanced data, and might not be resilient to imbalanced datasets. NN biases its results to classes that are better presented in the dataset, which results in a higher average precision. At the same time, VAEs still can be used for the detection of attacks that have smaller representation in the data.

Dynamic Model Preparation

In this sub-section, the preliminary steps required for optimal event detector allocation and computation of the “under attack" probability are presented. TPG is also developed based on the topology of the network used for dataset generation.

Target System Description. TPG development depends on network topology to ensure effective security and protection. In the CICDS2017 dataset, there are two main components in the network: “trusted" or “victim" internal network with an IP range of “192.168.10.*"; and secondly, an external network with the attacker’s machines (referred to as the "attacker network") with an IP range of “205.174.165.*". In this thesis, the attackers are referred to as Kali, who was operating from “205.174.165.73", and Win, who was operating from “205.174.165.69", “70", and “71".
The victim network has two machines with public IPs:

- Web Server 16 Public: 192.168.10.50 and 205.174.165.68
- Ubuntu Server 12 Public: 192.168.10.51 and 205.174.165.66

The victim network consisted of 12 machines in total, each running a different operating systems: Windows Vista, 7, 8.1, 10; Ubuntu server, 14.4, 16.4; and Mac OS X. Thus, different vulnerabilities could be exploited against different machines.
The network firewall operates with the IP address of 205.174.165.80. The other machines in the victim network are introduced while describing attack scenarios. Most attacks target public servers (e.g., web DoS attacks), but the infiltration attack in the recorded data targets all machines inside the victim network by operating in two steps: firstly, a vulnerability on the target machine is exploited, granting control over some machine functions; and secondly, by using the victim’s machine, an attack on the rest of the hosts in the network can be performed using a trusted IP.
TPG development was accomplished by following the methodology presented in the previous chapter. The main difference is that, in this case, there was no impact minimisation task, meaning that the impact matrix definition was omitted.

Definition of PA, SA , as well as the links between them. As the network for the dataset is defined generically, with no explanation of the primary purpose of the activity, the assumption was made that each machine on the network had its own specific role and served a dedicated function. Therefore, the high-level PA could be defined as “Data on Host" (DoH) and “Operations on Host" (OoH). DoH denotes all the sensitive or operation-related non-public information stored on the corresponding machine. OoH is related to the possibility that the specific function of the machine will become operational. In this context, the SAs were also defined in a high-level way, and this occurred independently for each host in the victim network. The SAs for each host were DoH (repeating the corresponding PA), Command and Control (CCo), and Computational Capacity (CCa). CCo was considered independently from CCa, as it may serve as an enabler for a sequential attack using the controlled host (e.g., infiltration attack), while the second one may serve just for the degradation of the OoH.

Threat Scenario Definition. In the proposed TPG model, all the CICDS2017 attack types were presented in the threat scenarios. Each attack type was considered independently as follows:

- DoS and DDoS attacks in the dataset were presented only against the public IPs despite the fact that, in a general case, these attacks can be performed against the host machines within the victim network. However, this can be performed against the internal network only after gathering access to the network via NAT procedures associated with the firewall or any other method. By contrast, public machines can be attacked without preliminary conditions. The TPG is constructed in a way that take the indicated sequence of actions into account. The targeted SA for each host is CCa. Attacks of these types are denoted further as $DoS_{GoldenEye}$, DoS_{Hulk}, DoS_{sht} (for Slowhttptest), DoS_{sl} (for slowres), $DDoS$. In general, this family of attacks is denoted as DoS.

- The representation of web attacks in the TPG followed the same approach as was the case with DoS. This implies an assumption that each host is running an instance of a specific type of web server. The information about running services is not given explicitly in the dataset, and therefore it is assumed that each host could be a victim. All SAs related to a host form a set of potential targets. Attacks of these types are denoted as WA_{XSS}, WA_{BF}, WA_{SI} (for Sql Injection), or WA in general.

- Infiltration attacks. Targets all hosts, but could be performed after additional information is gathered via preliminary attack (e.g., via web attack or Heartbleed attack). It may target all SAs associated with a given host. This type of attack is denoted as Inf.

- Brute force SSH attack and FTP attacks are possible against any host. For simplicity, it is assumed that an FTP server is running only on the machine with IP 192.168.10.50. The SSH attack targets all the SAs of a host, while the FTP targets only DoH. These two attacks are defined by BF_{ssh} and BF_{ftp}, BF for the family.

- Heartbleed attack may target any machine with a service using OpenSSL library, and which is accepting incoming connections. Therefore, all hosts are supposed to be vulnerable to this attack type. The attack can be performed under the same conditions as the DoS attack. It targets the DoH, and it is denoted as HB.

- Port scan targets all hosts after a firewall NAT process. As the dataset does not detail an exact vulnerability used by port scan, a raw “*.pcap" data was used to identify the nature of the attack. According to “*.pcap" files from the dataset, the attacker machine was sending an “nmap" command to the victim machines, which identifies open ports and (possibly) the name of the corresponding running service. Therefore, this type of attack targets the *DoH* SA type but mainly serves as an enabler for other attack types. It is denoted as *PS*.

- Botnet attacks are possible against all hosts running a specific type of debug bridge. In the dataset, the attacks are not limited just to one host, but to all Windows machines: Windows 10 192.168.10.15, Windows 7 192.168.10.9, Windows 10 192.168.10.14, Windows 8 192.168.10.5, and Windows Vista 192.168.10.8. Therefore, the assumption was drawn that these machines are vulnerable to these types of attack. This attack targets all SAs, and it is denoted by *BN*.

The SAs for the non-expanded graph are denoted by type of asset (DoH, CCa, CCo) and the IP address (e.g., $DoH(192.168.10.51)$). An additional type of attack, which is related to the NAT process on the firewall, is denoted as $NATP$.

TPG Construction. By considering the set of proposed threats, it is important to analyse the prerequisites for each attack against each node. Attacks that exploit certain vulnerabilities may require knowledge of specific credentials. For instance, an infiltration attack using Dropbox requires a an upload of CCo software to the victim machine via synchronisation with cloud storage. These credentials can be obtained from a centralised password management tool using another attack (e.g., the Heartbleed or web attacks).
For the reasons stated above, the TPG was constructed using the sequential attack methodology described in the previous chapter, namely the application of SATAA and M-SATAA. As noted in the description of M-SATAA, a set of attack pairs and graphs should be defined. Since the sequential attack algorithms are applied over the extended graph, a denotation over the set of extended SAs should be introduced. The naming follows the following structure $\{A\} - \{SAT\}(IP)$
, where A represents the attack type acronym, SAT denotes the SA type acronym, and IP specifies the targeted host. The set of internal hosts is defined by IP_{INT}, while public hosts are denoted as IP_{PUB}.
The sequences of attacks are described above in the threat scenario definition. It is assumed that the NAT process is not detectable and does not require any pre-requisites. This assumption was made because the process is not outlined or found in the labelled vector dataset, and it is not obviously identifiable in the “pcap" data. Therefore, it was excluded from the scope of the assessment. The formalisation of the attack sequences is presented in the following list:

- As a first step, it is necessary to identify the attacks with no pre-conditions. Let us denote $P_{EXT} = \{A - t(A)(ip) | A \in DoS \cup WA \cup BF \cup PS \cup HB, ip \in IP_{PUB}, t \in T(A)\}$.

Here, $T(A)$ represents all SA types targeted by the corresponding attack type A. The first item in the list is defined as $\{(\emptyset, (NATP \cup P_{EXT}, G_1)\}$. It was impossible to retrieve any signs of communication between the firewall and the attacker's IPs in the "*.pcap" file. Therefore, the part of the graph related to $NATP$ is trivial. The rest of the attacks were performed against the public IPs.

- Let us define $P_{INT} = \{A - t(A)(ip) | A \in DoS \cup WA \cup BF \cup PS \cup HB, ip \in IP_{INT}, t \in T(A)\}$. Then the second item in the list is $\{(NATP, (P_{INT}, G_2)\}$. Attacks against internal IPs are possible after the firewall NAT procedure is performed.

- Additional sequential attacks against public hosts are introduced as follows:

$$(\{PS - DoH(ip) | ip \in IP_{PUB}\}, (\{BN - t(ip) | ip \in IP_{PUB}, t \in T(BN)\}, G_{ebn}) \tag{5.27}$$

  Botnet attack after a port scan:

$$\begin{gathered}(\{A - DoH(ip) | ip \in IP_{PUB}, A \in WA \cup BF\},\\ (\{Inf - t(ip) | ip \in IP_{PUB}, t \in T(Inf)\}, G_{einf})\end{gathered} \tag{5.28}$$

  An infiltration attack follows after WA, HB, or BF attacks.

- Sequential attacks against internal network hosts, constructed in the same way as for the public network part:

$$(\{PS - DoH(ip) | ip \in IP_{INT}\}, (\{BN - t(ip) | ip \in IP_{INT}, t \in T(BN)\}, G_{ibn}) \tag{5.29}$$

  For a botnet attack after PS. Sequential attack structure for infiltration is as follows:

$$\begin{gathered}(\{A - DoH(ip) | ip \in IP_{PUB}, A \in WA \cup BF\},\\ (\{Inf - t(ip) | ip \in IP_{INT}, t \in T(Inf)\}, G_{iinf})\end{gathered} \tag{5.30}$$

  Here, it is assumed that the data stored on the public machines might be used for an attack against the internal network machines (which is an obvious vulnerability), but this explains the sequence of attacks given in the dataset and gives an opportunity to demonstrate the sequential attack scenario application using TPG.

To finalise the definition of TPG, it is necessary to define the graphs from the attack sequences described above: G_1 and G_2 for the public and internal attacks, and $G_{ebn}, G_{einf}, G_{ibn}, andG_{iinf}$ for different types of sequential attacks.
The general approach for constructing the graph assumes that the set of steps needed to perform a given attack against each host is similar. This assumption is postulated partly due to a lack of information about the services running on different machines. The data

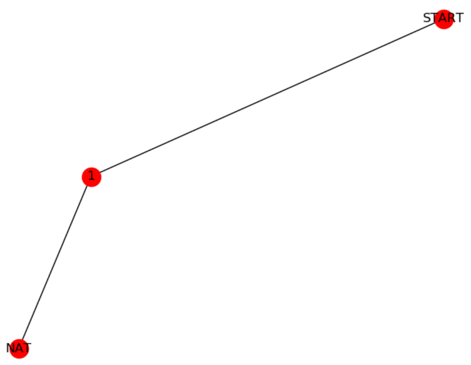


Figure 5.9: TPG for NAT Process. Here, an additional SA is introduced, representing a consistency of the firewall rules.

about network topology and the nature of attacks provides the information required for attack sequence definition, but no additional details are given. The rest of the data is organised in the form of vectors, describing the communication sessions between two hosts.
Each attack type is described separately in the next paragraphs. In turn, the overall graph is constructed.

NAT Process on Firewall. As mentioned previously, the dataset does not contain any records about the NAT processes on the firewall presented in vector or in “pcap" format. For this reason, the graph is kept trivial. It is presented as an entry node, the sequential node representing the execution of the procedure, and the SA. As no information is given, it is assumed that no event detectors are deployed in this case. This makes the condition of the experiment more complicated as, in reality, it is possible to analyse the logs and events at the service. The corresponding graph is presented in Figure 5.9.

Brute Force Attacks: SSH and FTP . The target of this attack is to gain access to the host machine via sequential login attempts using semi-random credential generation. The SSH attack grants access to the machine on the level of the corresponding user. In this experiment, all the users are assumed to be “sudoers" (which is to say, they have admin rights).
Therefore, the first action after the start of an attack is a login request to the system using the Patador script. SSH access is usually protected by a password, which can be considered a security control. The next step involves gaining control over the system. In turn, for each type of asset, an additional abstract action is added.
The graph for an FTP attack has the same structure, given the limited set of SAs under attack.

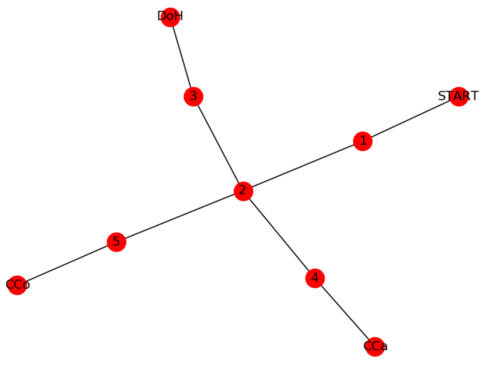


Figure 5.10: TPG for brute force SSH attack against predefined host

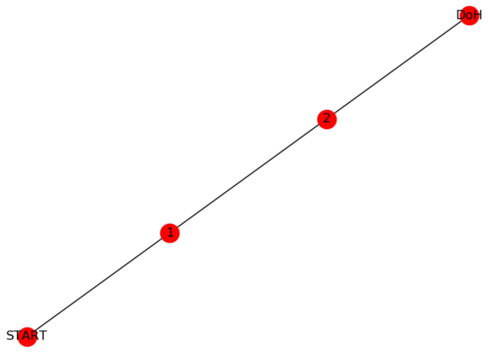


Figure 5.11: TPG for brute force FTP attack against predefined host. The distinguishing feature compared to the SSH TPG is that there exists only one SA, *DoH*.

DoS Attacks. This set of attacks targets the CCa of the victim host. The attack is organised in the form of launching a specific script and performing an attack. Therefore, the graph has three different simple paths from the entry point to the corresponding SA of the victim host. Each path has two main steps: script launch and pre-SA node. Figure 5.12 represents the graph for DoS attack.
It should be noted that, for the case of a DDoS attack that is executed after a botnet attack, the graph might be different. However, this case is omitted from consideration as it is not presented in the experiment's dataset.

Heartbleed Attack. A Heartbleed attack occurs when requests are sent to services on victim machines, thereby leading to the acquisition of pseudo-random information stored in the machine's memory. To execute this attack, the "heartbeat" OpenSSL message is sent to the services of the machine, usually running on port 443. The graph targeting a specific host is considered to be trivial, and it is represented as in 5.13, which is as follows:

- Start node
- Node, representing a communication "heartbeat" exchange between the attacker and

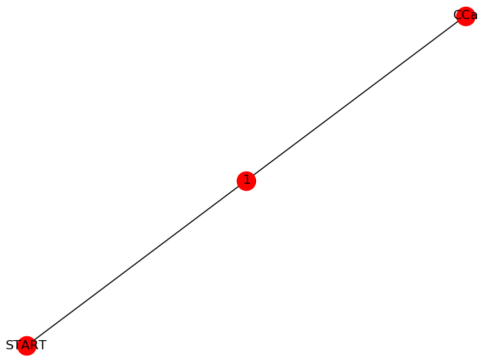


Figure 5.12: TPG for DoS attack against predefined host. Targeted SA: *CCa*

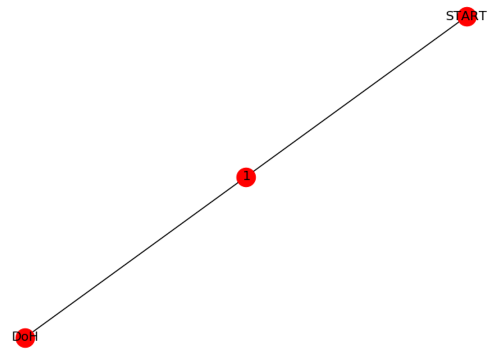


Figure 5.13: TPG for Heartbleed attack against predefined host. Targeted SA: *DoH*

the victim machines

- Node, representing the acquisition of relevant information (pre-SA node)
- DoH SA node

Web Attacks. For each type of web attack, it is necessary to detail the graph structure independently given that the approach differs. Additionally, the sequence of actions involved in each attack depends on the targeted service, which influences the TPG construction process. To obtain more details about the victim services, “pcap" data was used as a reference in the present experiment. For all cases, communication was undertaken using HTTP with no SSL encryption, and so it was possible to obtain information.

- Brute-force web attack is the simplest example. It targets to perform authentication on target web-server. Construction of the graph is based on the recorded data, as it depends on the specifics of the targeted web server. Brute force-related “pcap" files, as well as the vector representation of the data, indicate that the attack was recorded as a communication session between the victim machines and the firewall (external IP). The communication session targeted port 80 and contained the scripts “login.php" and

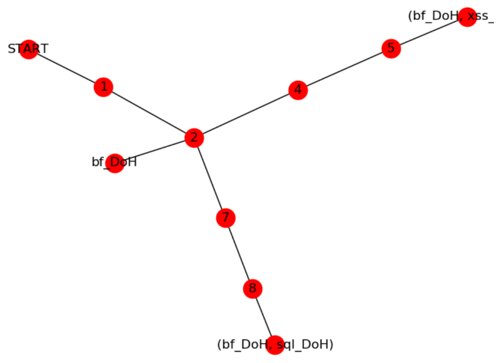


Figure 5.14: Joint TPG for 3 different types of *WA* after applying M-SATAA. SA naming preserves the formalism introduced in the algorithm description.

the image “login.png", along with POST requests with content “username=admin", and so on. Therefore, it is assumed that the attacker made a login attempt to the protected webpage. As no additional information is given, it is assumed that the targeted SA is DoH. The implementation of the graph is the same as it is for FTP *BF* attacks.

- SQL injection attacks target data-driven applications by inserting malicious SQL code for execution. In the “pcap" data, attacks against the server are represented by the concatenation of SQL expression to the parameters of the GET requests (requesting information about usernames and passwords in the database). Therefore, it is assumed that the attack is performed against a web application, targeting internal host data. The attack is represented as a path from 3 nodes: start, communication session, DoH.

- XSS attacks usually target user web applications by sending malicious scripts in requests. Again, the nature of the attack depends on the targeted service or infrastructure. In the “pcap" data, the communication session for the attack is represented in the form of GET HTTP requests, which contain the script in the request’s parameters. Corresponding replies contained a list of sites. Therefore, it is assumed that the XSS attack is targeting the web application data stored on the host. The attack is represented as a path from 3 nodes: start, communication session, and DoH.

By considering the exact sequence of attacks performed in the dataset, it is assumed that the SQL injection and XSS attacks are only possible after the brute force attack against the server. Therefore, the joint graph for a sequential attack (after the application of M-SATAA) is prepared for the whole *WA* family, as shown in Figure 5.14.

Infiltration. This attack has a preparatory stage, which aims to obtain information about potentially vulnerable services. In the dataset, the Inf attack is performed using Dropbox

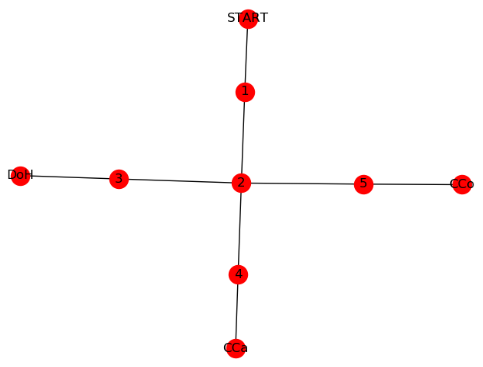


Figure 5.15: TPG for infiltration attack against predefined host. Targets all types of SAs.

vulnerability, that allows uploading CC software on the synchronised machines. To achieve this, the corresponding Dropbox credentials are needed. After obtaining the credentials, the attack is undertaken in several sequential steps:

- Upload CC software to the user account and wait for it to synchronise on the victim machine.
- CC auto-launches on the victim machines and allows external communication.
- Obtain control over the machine (all SAs under possible attack).

All these actions are represented in the corresponding graph given in Figure 5.15.

Port Scan. A portscan attack shall be performed against the machine, which considers the attacker machine to be from the trusted network. The attacker actions are consecutive. Therefore, the corresponding TPG is constructed as a simple path, consisting of start node, “nmap" node, pre-SA node, and *DoH* SA node.

Botnet. This attack is performed after the *PS* attack. As this attack type is not presented in the validation scenario, TPG construction is trivial. Otherwise, the construction process depends on the exact type of vulnerability exploited for each service. In this scenario, the *BN* attack has a trivial representation with three linearly connected nodes: start node, attack execution, pre-SA node, and SAs. The TPG structure is the same as in the case of the infiltration attack, as illustrated in Figure 5.15.

Overall Graph Construction Process. The sub-graphs required for the construction of the joint attacks include the following:

- The graph G_1 is constructed as a union of three graphs: NAT process graph, and graphs from P_{EXT} attacks against both public IPs.

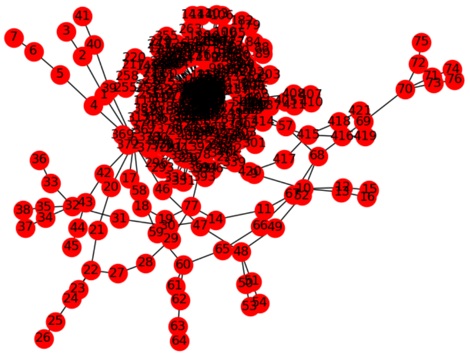


Figure 5.16: High-level view of TPG for CICDS2017dataset

- The graph G_2 is constructed in the same way as G_1, but with no $NATP$ attack. It uses the attack graphs against all internal hosts defined by P_{INT}.
- The remaining graphs are directly mapped to the corresponding attack types for each vulnerable host.

The final extended TPG was constructed using the previously defined sequential attack lists, the sub-graphs for each attack type, and M-SATAA. The resulting graph was used to validate the dynamic model. It is important to stress that semi-automatic graph construction procedures, such as M-SATAA, are crucial for TPG development. This is because the resulting model tends to be large (but sparse). The TPG for the CICDS2017 dataset consists of 422 nodes in total.

The resulting TPG structure is presented in Figure 5.16, giving a high-level illustration of the model. Two main sub-models are obviously visible: the sparse structure at the lower-left part of the graph represents the G_1, while the dense circle part shows that G_2. G_2 is centred around the NAT attack, which gives access to all attacks against the internal network.

Dynamic Model Application

Two event detectors detailed in the previous section were used: VAE and NN. They were deployed at different nodes of the TPG. As the TPG structure defines explicitly the corresponding type of attack for each node, the parameters of the event detection placements (false positive and false negative detection rates) were defined as shown in Table 5.4. In this experiment, there were no additional event detectors apart from the listed ones (for example, SSH authorisation event could serve as such). This can be attributed to the lack of information in the dataset “pcap" data. For this reason, each host had only one possible event detection placement node, which was related to the network communication action of the attack.

For the validation of the dynamic model, three possible event detection setups were compared:

- Random event detector allocation among the graphs.
- Optimised event detector allocation using the simulated annealing algorithm.
- General network analyser with no specific allocation. It is equivalent to the application of the data mining model directly to the feature vector type of data.

To carry out the comparison, it was necessary to derive the optimised allocation.

Optimal Event Detector Placement. Optimisation of event detector allocation was undertaken using simulated annealing allocation. The computation of the static distribution was achieved using the sampling approach, where the rationale for this stemmed from the fact that the numerical optimisation was too slow on the available computational resources. The target function was defined in a way, similar to 5.17 in order to balance the probability of an undetected attack with the false positive detections. Initial allocation of event detectors for the simulated annealing algorithm was selected randomly.
The “adaptation" part of the simulated annealing algorithm was modified to fit the size of the problem. At each iteration, two types of modification were applied several times in order to derive from the current “optimal" allocation, as detailed below:

1. Move a random existing event detector from the current node to a different “free" node.
2. Select a random node with a detector and change the detector’s type.

The “adaptation" part of the algorithm demonstrated that the number of event detectors remained the same, while only the placement changed. The number of applications of the listed modifications decreased at each iteration as $O(\frac{1}{\sqrt{n}})$, where n denotes the iteration number. At step 0, modifications were applied 150 times. Overall, the optimisation procedure had 1,500 iterations. Figure 5.17 outlines the typical behaviour of the optimisation algorithm when a fast function decreased at the beginning and slowed down as the number of iterations increased. There is no standard approach for the selection of the number of iterations, but several heuristics exist. For this current experiment, the number of iteration was fixed to 1,500, which is a compromise in terms of the time and the achieved result. As the experiment sought to show the added value of the proposed optimisation procedure compared to “general network analyser" and random allocation, even a sub-optimal solution was considered to be acceptable for the purpose.
By considering different components of the target function, the following behaviour was observed:

- The undetected attack probability did not fall, and it increased slightly at the end of the optimisation procedure, as shown in Figure 5.19.

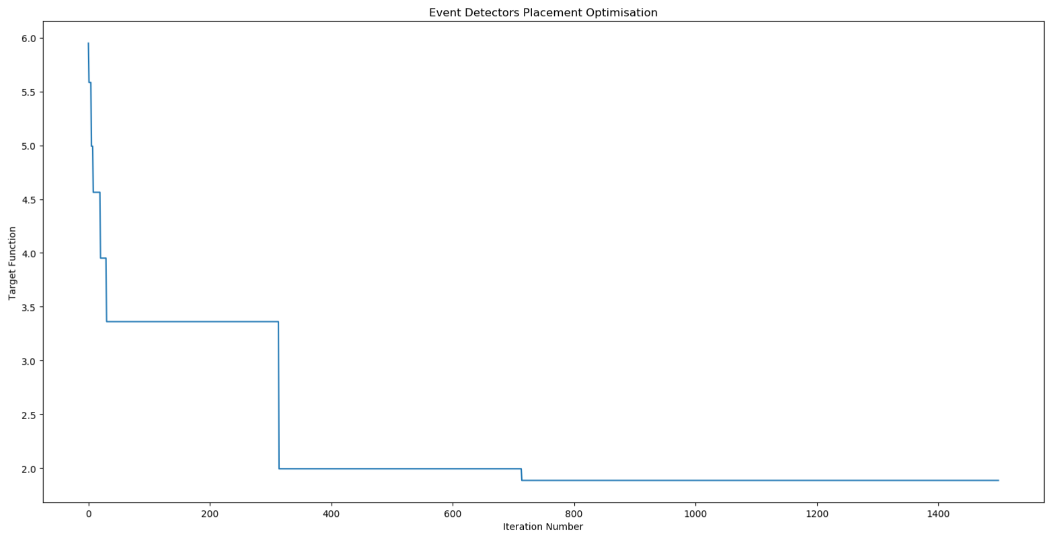


Figure 5.17: Event detector placement optimisation process for CICDS2017

- The variance upper bound probability dropped significantly at the initial stages of the optimisation and, in general, resembled the fall of the target function.

The fact that the undetected probability did not fall for the optimised allocation may be considered a form of trade-off behaviour when more sensitive event detectors (with lower false negative rate and higher false positive rate) were replaced with detectors with fewer false positives and true positives. This replacement resulted in a lower variance of the static distribution and led to a better attack detection capability of the model, as shown in the next paragraph. Lower false positive rates allow using lower probability confidence thresholds for overall attack detection, which neutralises the effect of the increased undetected attack probability.

Under Attack Probability Evaluation. This was undertaken by analysing the behaviour of the event detections coming from the NN and VAE model, allocated in the developed TPG graph. An algorithm for attack detection described in subsection 5.2.5 is applied.
Model instantiation phases related to graph construction and event detector set definition are described in subsections 5.3.3 and 5.3.2 correspondingly.
The metric used for attack detection was the $1 - p_{na}$ probability of the dynamic model for two allocations: random and optimised. Such a selection of a metric is equivalent to defining the critical set of nodes C as all nodes of the graph except v_{na}. This way, the function 2 is fully-defined except from the threshold th. In order to avoid static selection of threshold and observe various combination of true and false positive rates, the ROC curve is analysed. A comparison with fixed event detectors (VAE and NN) was given, as if these models would

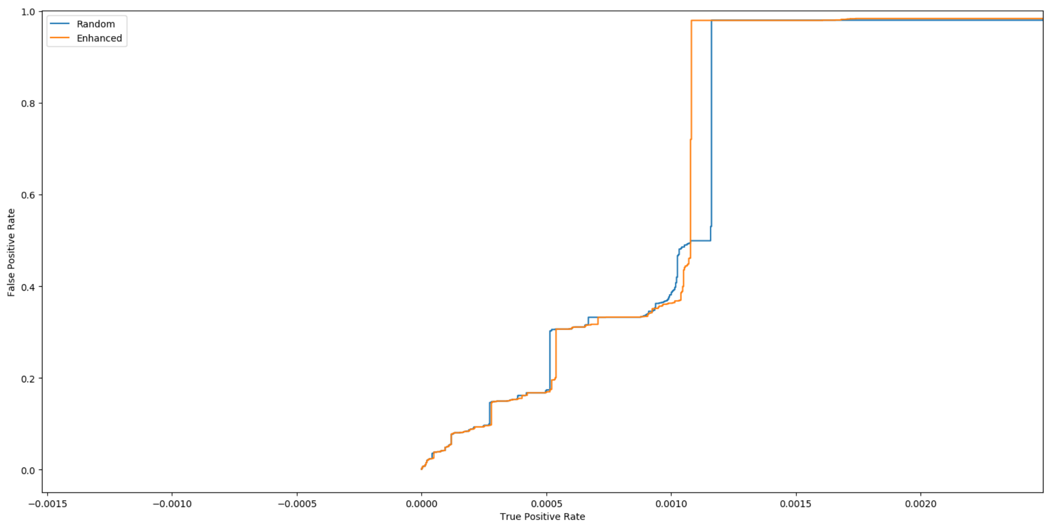


Figure 5.18: “Under attack" detection ROC curves CICDS2017

be applied without TPG at all, similar to standard ML algorithms over vector data.
In the experimental data, the attacker performed two main types of attack against the system: WA, as described in the section above, and INF attack sequentially. For each data vector from the machine learning dataset, it was possible to evaluate the response of each of the detectors (if deployed on the given host). In this way, the attack probability for each node of the TPG could be calculated. The VAE and NN models were configured such that the true positive and false positive detection probabilities were the same as those given in Table 5.4.
The quality of attack detection was compared using ROC curves, as shown in figure 5.18.
The false positive reaction probability fell dramatically compared to the application of general network analysis. The false positive probability for each event detector was fixed at 3%, while the aggregated rate of false reactions was only a fraction of a percent. This can be explained by the fact that the significant part of the false events was recorded for attacks that succeeded a different type of attack, which was not recorded beforehand. Therefore, the detection did not fit into the “temporal context" of all recorded events.
At the same time, the true positive detection probability for each attack type was slightly lower compared to the application with no TPG (by 0.5% for each type of attack). This difference can be attributed to the fact that the dynamic model does not react to each detection, but analyses the probability that the detection fits the context. The false negatives usually occurred in the first vector describing the attack. However, usually, there were several sequential vectors related to the same attack.
The difference in terms of the maximal achievable true detection rate between the optimised and random allocation was negligible, amounting to less than 0.2%. At the same time, a relatively significant difference existed in terms of the number of false detections, amounting

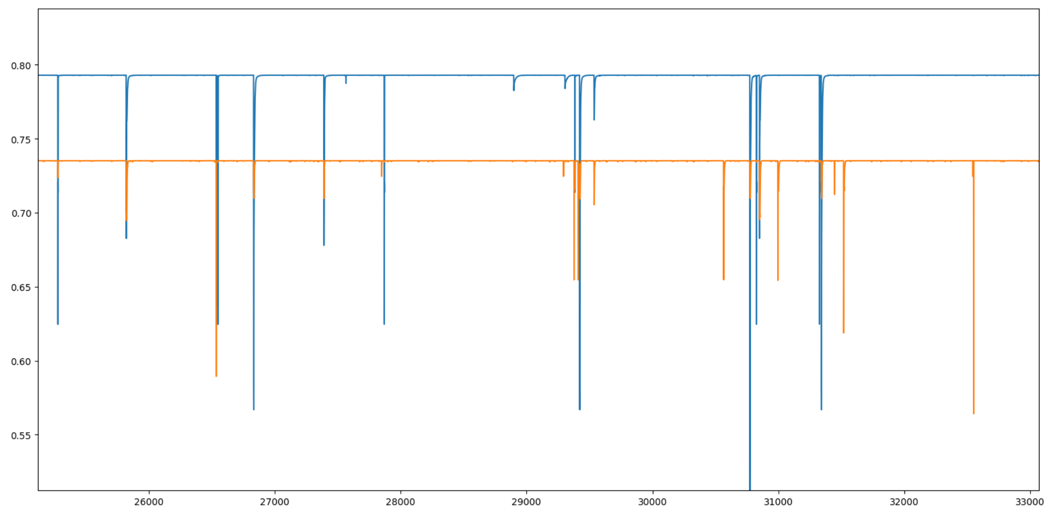


Figure 5.19: Behaviour of p_{na} probability under no attacks for the optimised and random allocation for CICDS2017 dataset. Orange line corresponds to optimised allocation and blue line corresponds to random allocation.

to approximately 0.01% of the total number of registered events. Given that the total false positive rate was 0.11%, such a reduction resulted in a 10% improvement.

The difference between the optimised and random allocation is illustrated in Figures 5.19 and 5.20. On average, the false-positive-related probability drops of the optimised allocation were low in terms of their amplitude, as demonstrated in Figure 5.19. After the occurrence of the false positive, the sequential increase of the respective p_{na} value was faster, and it reached the “baseline" in several time steps, as shown in Figure 5.20. The demonstrated behaviour supports the result shown in Figure 5.18, where a sufficient reduction was recorded in the false positive rate for optimised allocation.

Considering the respective baseline values for both allocations, it is reasonable to state that the optimised allocation yielded counter-intuitive results with a lower p_{na} value. One way to account for this is by referencing the fact that the event detectors were placed at the part of the graph related to “no preceding" attack conditions, and they had lower false true positive rate. Therefore, their output was taken by the dynamic model with lower confidence, as one can conclude from Section 5.2. For the same reason, the respective maximal true positive detection range of the optimised model was slightly lower.

5.3.4 Model Limitations

The temporal model described in this chapter is based on the TPG approach. Therefore, it suffers from the same TPG-grounded problems, including the absence of a fully-formalised

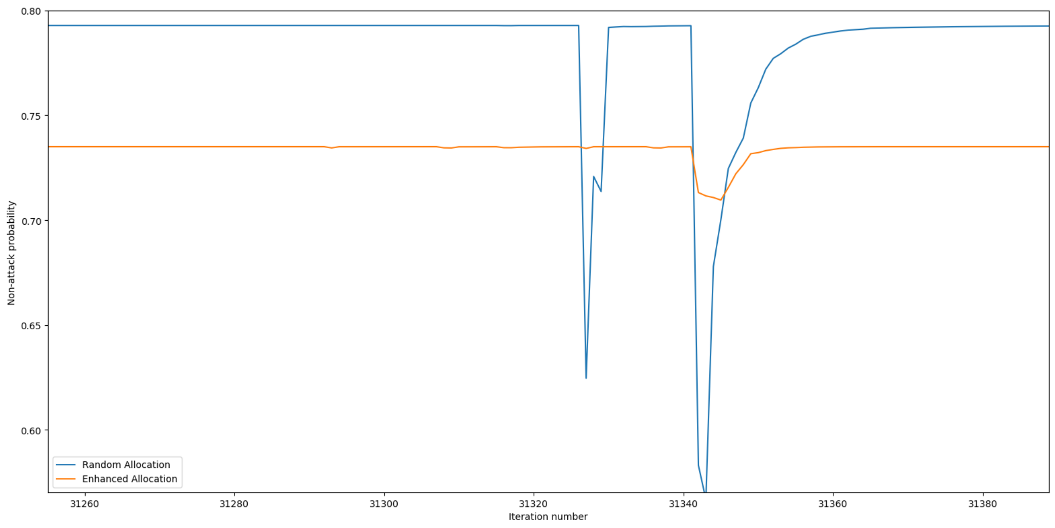


Figure 5.20: Behaviour of p_{na} probability under no attacks for the optimised and random allocation for the CICDS2017 dataset, with detailed representation of a specific false event. Orange line corresponds to optimised allocation and blue line corresponds to random allocation.

methodology for graph definition and consideration of a single attack at one moment of time. The second problem becomes more significant because the analysis is undertaken in an online manner, and it is impossible to converge the analysis of multiple simultaneous attacks to a consideration of a worst-case scenario, as it was done in the static model description. In future research, extending the model for multiple-attack cases using latent-variable modelling is a promising idea, when the number of ongoing attacks is a hidden variable, which needs to be derived from the incoming data.
The presented dynamic model has the following drawbacks, related to its formulation:

- The conditional probabilities of the transition matrix, as defined by the graph, do not have a specific rule for their definition. In this chapter's experiments, a uniform probability distribution was used, which is a case of “no-preferences". It is possible to improve the model by making the conditional probability dependent on the ongoing attack and estimating the attack as a hidden variable in Bayesian inference.

- A temporal component was trivially introduced into the model (i.e., the time spent by the attacker at each node is supposed to be the same). To mitigate this problem, it is possible to define the time spent explicitly, as a probability distribution over “time to pass a given edge". In this way, it is possible to define a joint probability over the pairs (n,t), where n is a node and t is time. Alternatively, it is possible to introduce loops into the graph, as an absence of loops is not a requirement for the theoretical properties of the dynamic model (notably, loops are not possible in the static model).

5.4 Summary

In this chapter, an extension of the TPG was presented, which sought to capture the problem of attack detection. The dynamic model was built on the same basis as the static model. This fact simplified the process of applying both methods to the system because the model instantiation prerequisites could be partially reused.
The dynamic model deals with the problem of allocating event detectors within the TPG nodes, and it estimates the probability that the attacker is currently at a given node (or set of nodes). An analysis of the behaviour of such a probability is given under different conditions, and an asymptotic behaviour model is provided, as well as a method for the identification of the “under attack" condition depending on the given metric. A detection scheme was proposed, given specific assumptions of the quality of the event detectors.
A problem of “optimal event detection placement" was also considered, which targeted the allocation of the event detectors in the graph in a way that minimised (or maximised) a given metric. The metric could be related to early attack detection, minimisation of false detection impact, and others.
Both problems were described and validated on the CICDS2017 cybersecurity dataset. This chapter’s experiment also demonstrated how to produce a TPG model for a cybersecurity domain, filling some of the gaps in the example given in the static model remote tower. The results demonstrate a drastic reduction in the false positive detection rate compared to the regular application of ML-based network flow analysers.

Part III

Conclusion

Chapter 6

Contributions overview and future work

The aim of this thesis was to address the problem of mathematical modelling and to support security risk assessment (SRA) activities. As indicated in the first part of this thesis, the main requirements for the models that the researcher sought to develop were generality in terms of applications and a common language with existing SRA techniques. After reviewing relevant literature, the graph-based approach was selected for the instantiation of the model, and game theory (in particular, network security games, or NSGs) and probabilistic graphical models were used as the basis for the decision-making process.
The key contributions of this thesis reside in the following areas:

- Definition of the threat path graph (TPG) in chapter 3.
- Development of a game theoretic model for general SRA support in chapter 4.
- Proposal of a security awareness model for “in-operation" streaming data processing in chapter 5.

The TPG development process was described with an explicit alignment to the SecRAM method, providing general recommendations for the definition of the structure of the graph. The graph definition was conducted so as to handle multiple attack types. Special attention was paid to the case of sequential attack definition when one attack enables another and multiple assets are targeted. For this purpose, two algorithms for semi-automatic graph construction were presented.
An NSG-based game theoretic model, referred to throughout this thesis as the static model, was augmented with several algorithmic extensions, including complex security controls structure definition, procedural controls formalisation, and multi-objective optimisation

problem statement. The generic game definition converged on a sequence of issues in multi-integer linear programming (MILPs), and this convergence was theoretically justified. To support the statement of generality, model instantiation was aligned with SecRAM procedures, demonstrating the correspondence with the main steps of this classical methodology. The considered approach, despite involving a large number of parameters, strictly limits the scope of the analysis defined by the graph structure, the properties of the controls, and the impact functions. The primary benefit of the approach is the capability to support and verify expert judgements, and to offer “what-if" considerations in terms of control selection and changes to the goal statements. The combination of TPG and the static model was applied to the remote tower example. The remote tower model development process was compared with the respective steps of the SecRAM analysis report for the same system. The results obtained from the static model were compared with the security control allocation from the SecRAM report.

The problem of “in-operation" event detection was addressed by the probabilistic model over the TPG in the form of the dynamic model. This model dealt with the enhancement of event detector allocation within the system and offered a definition of attack detection criteria. Several of the proposed model’s theoretical properties were derived and proven. Based on these properties, an event detection criterion was introduced, and the event detector’s allocation problem was stated. A family of methods for the solution of the allocation problem was proposed. Developed algorithms were applied to a cybersecurity intrusion detection open dataset. This experiment demonstrated the process of graph development for a cybersecurity problem (by using the semi-automatic algorithms from the previous chapter), event detector allocation, and attack detection. The results were compared with the “plain" results with no graphical model, and a significant reduction in the false positive rate was identified.

It is important to note that both of the models share the same basis. Furthermore, they are applicable to sequential problems: system security design and situational awareness in operation.

6.1 Modelling Process Outline

This section summarises the proposed modelling framework for SRA support. It consists of three components: TPG model for system description, game theory-based model for security controls allocation, and “dynamic" model for on-line data processing and attack detection. TPG development is split into two main phases listed below.

- Asset model development.

- Threat model development.

Asset model instantiation is close to the context establishment phase of ISO 27005. It identifies the main functions of the system (primary assets), main assets that enable these functions (supporting assets), and corresponding vulnerabilities. For each possible vulnerability (or attack type) against every asset, a set of impacts is evaluated. Values of these impacts are determined depending on the level of “interruption" of the primary assets.
The threat model may be considered as a combination of NSGs and attack trees. It corresponds to the risk treatment phase of the SRA process. The threat model describes a set of steps and statuses that the attacker needs to go through in order to reach the supporting asset. TPGs provide means for a description of complex, sequential attacks. Some of the arcs of the TPG may be used for security controls or event detectors deployment.
Security controls may be allocated on the TPG by using the “static model" described in the thesis, which determines a set of possible Pareto-optimal event allocations. “Static model" relies on game theory and may consider different types of limitations over the set of security controls. These limitations may correspond to business or price constraints.
Apart from security controls, it is possible to allocate event detectors on the graph. The event detection allocation mechanism proposed in this thesis relies on the minimisation of undetected attack probability (given some assumptions). The allocation may be done using a combination of Monte-Carlo and heuristic optimisation algorithms.
Parts related to security controls and event detector allocation may be aligned with risk treatment and risk acceptance parts of the SRA.
Apart from that, this thesis proposes methods for attack detection based on aggregated analysis of the registrations coming from all event detectors. Event detector allocation and analysis is based on Bayesian inference techniques and probabilistic graphical models (namely, Bayesian filter).
The implementations of both the static and dynamic models are available at the following gitlab link, which contains:

- Data for the airport model.
- Matlab scripts for the airport static model.
- Python scripts for event detector training and dynamic model application over CICDS2017.

For any questions regarding the implementation please contact Denis Kolev.

6.2 Model Limitations

The limitations of the static and dynamic models were described in Chapters 4 and 5, respectively. While certain drawbacks arise from the specifics of the mathematical formalisation (e.g., simplification of attack mitigation for the static model), others are grounded in the TPG implementation scheme. The most critical drawbacks of both models include the following:

- Each model considers a single action at one point in time. This problem is “encoded" into the TPG, and it leads to complexities when introducing multiple attacks into both the static and dynamic models. Additionally, it leads to an overwhelming growth of the graph when a set of unordered conditions must be met. This problem could be overcome by using the SATAA and M-SATAA algorithms, but still, the graph size will be large, and it could result in computational problems for the models. The SATAA and M-SATAA algorithms for a set of n unordered enabling conditions would produce $O(n!)$ additional nodes. It is likely that the graph size issue could be partially fixed by the introduction of additional labels to the graph links and nodes. However, this would be likely to “spoil" the least weighted path formulation of the problem, meaning that the static and dynamic models should be partially redeveloped.

- Certain attack types are characterised by a waiting pattern (e.g., stealing credentials using malicious software). This type of attack has no convenient means for representation in the dynamic model, as the sniffer waits until the credentials are used by the victim. This time span is not pre-determined and can hardly be assessed in terms of average waiting time, which confounds an accurate definition of the transition matrix. This issue is less important for the static model in the form in which this thesis presents it, though it could introduce an additional problem in a more complex risk mitigation scheme. One of the possible solutions is to make the transition matrix time-dependent such that some of the links have non-zero probabilities only if specific conditions are satisfied.

- There is also an implicit dependency between the set of detectors and the TPG. Although the TPG could be formally developed independent from the specific list of event detectors and security controls, the resulting graph might not be adaptable to the considered set of security controls. For example, the TPG for the case of “SSH login" in case of public key protection and with no public key protection could be different. The possibility of a public key mechanism creates a full separate branch of the graph, related to “pubkey" stealing or login from an authorised IP. For the cases in which the list of security controls and detectors is preliminarily fixed, this fact does not create a problem, but it could undermine the method’s scalability.

6.3 Future Work

Directions for future work for each of the models are defined by the drawbacks listed in the corresponding chapters, as well as suggested possibilities for improvement.
Apart from limitation-related points, an attempt to create a stronger link between the models should be made when event detectors and security controls are considered simultaneously. This addition could be critical for the cases when a security control serves as an event detector. It could be partially implemented by merging the optimisation problems of static and dynamic models. To correctly define these types of problems, an explicit dependency between “attack mitigation" and “attack detection" should be introduced.
Another significant large-scale modification would be counter-action planning by deploying additional controls over the graph depending on the observed detections. This could be possibly formalised as a sequential application of the static model within dynamic model iterations.
The last important direction for further research involves the creation of a library of graphs for well-known attacks, such as the ones mentioned in dynamic model validation. Such a library would simplify graph creation up to an application of the M-SATAA algorithm over a set of existing, semi-finished sub-graphs.

Bibliography

[1] 06.03.01 Remote and Virtual Tower Security Risk Assessment . Technical report, 2013.

[2] ATM Security Coordination and Support, Final Report. `https://www.sesarju.eu/sites/default/files/160602_Final_Project_Report.pdf`, 04 2016.

[3] Whitepaper: Introduction to remote virtual tower. `https://www.frequentis.com/sites/default/files/support/2018-02/RVT_whitepaper.pdf`, 02 2018.

[4] Ali Abdollahi, Krishna R. Pattipati, Anuradha Kodali, Satnam Singh, Shigang Zhang, and Peter B. Luh. *Probabilistic Graphical Models for Fault Diagnosis in Complex Systems*, pages 109–139. Springer International Publishing, Cham, 2016.

[5] David L. Alderson, Gerald G. Brown, W. Matthew Carlyle, and R. Kevin Wood. Solving Defender-Attacker-Defender Models for Infrastructure Defense. In R. Kevin Wood and Robert F. Dell, editors, *Proceedings of the 12th INFORMS Computing Society Conference: Research, Computing, and Homeland Defense*, pages 28–49. INFORMS, 2011.

[6] Sally Almanasra, Khaled Suwais, and Muhammad Mohd Arshad. *The Applications of Automata in Game Theory*, pages 204 – 217. IGI Global, 05 2013.

[7] Paul Ammann, Duminda Wijesekera, and Saket Kaushik. Scalable, graph-based network vulnerability analysis. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, CCS '02, page 217–224, New York, NY, USA, 2002. Association for Computing Machinery.

[8] F. Baiardi, C. Telmon, and D. Sgandurra. Hierarchical, model-based risk management of critical infrastructures. *Reliability Engineering & System Safety*, 94(9):1403–1415, 2009. ESREL 2007, the 18th European Safety and Reliability Conference.

[9] Michael Bell, U Kanturska, Jan-Dirk Schmöcker, and Achille Fonzone. Attacker-defender models and road network vulnerability. *Philosophical Transactions of the*

Royal Society A-Mathematical physical and Engineering Sciences, 366:1893–1906, 07 2008.

[10] C. Bellinger, S. Sharma, and N. Japkowicz. One-class versus binary classification: Which and when? In *2012 11th International Conference on Machine Learning and Applications*, volume 2, pages 102–106, 2012.

[11] Sheela V. Belur and Jonathan Gloster. Mathematical model for security effectiveness figure of merit and its optimization. In Belur V. Dasarathy, editor, *Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security 2008, Orlando, FL, USA, March 17-18, 2008*, volume 6973 of *SPIE Proceedings*, page 697305. SPIE, 2008.

[12] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[13] Christian Borgelt and Rudolf Kruse. *Graphical Models: Methods for Data Analysis and Mining*. John Wiley and Sons, Inc., USA, 01 2002.

[14] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[15] Matthew Brown, Bo An, Christopher Kiekintveld, Fernando Ordonez, and Milind Tambe. Multi-objective optimization for security games. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '12, page 863–870, Richland, SC, 2012. International Foundation for Autonomous Agents and Multiagent Systems.

[16] Matthew Brown, Bo An, Christopher Kiekintveld, Fernando Ordonez, and Milind Tambe. An extended study on multi-objective security games. *Autonomous Agents and Multi-Agent Systems*, 28(1):31–71, January 2014.

[17] Matthew Brown, Sandhya Saisubramanian, Pradeep Varakantham, and Milind Tambe. STREETS: game-theoretic traffic patrolling with exploration and exploitation. In Carla E. Brodley and Peter Stone, editors, *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*, pages 2966–2971. AAAI Press, 2014.

[18] Yuan Cao, Haibo He, Hong Man, and Xiaoping Shen. Integration of Self-Organizing Map (SOM) and Kernel Density Estimation (KDE) for network intrusion detection. In Edward M. Carapezza, editor, *Unmanned/Unattended Sensors and Sensor Networks VI*, volume 7480, pages 146 – 157. International Society for Optics and Photonics, SPIE, 2009.

[19] A. Chakrabarti and G. Manimaran. Internet infrastructure security: A taxonomy. *Netwrk. Mag. of Global Internetwkg.*, 16(6):13–21, November 2002.

[20] Yulia Cherdantseva, Pete Burnap, Andrew Blyth, Peter Eden, Kevin Jones, Hugh Soulsby, and Kristan Stoddart. A review of cyber security risk assessment methods for scada systems. *Computers & Security*, 56:1–27, 2016.

[21] Sabarathinam Chockalingam, Wolter Pieters, André Teixeira, and Pieter van Gelder. Bayesian network models in cyber security: A systematic review. In Helger Lipmaa, Aikaterini Mitrokotsa, and Raimundas Matulevičius, editors, *Secure IT Systems*, pages 105–122, Cham, 2017. Springer International Publishing.

[22] José R. Correa, Tobias Harks, Vincent Kreuzen, and Jannik Matuschke. Fare evasion in transit networks. *Operations Research*, 65(1):165–183, 2017. No data used.

[23] Do Cuong, Nguyen Tran, Choong Seon Hong, Charles Kamhoua, Kevin Kwiat, Erik Blasch, Shaolei Ren, Niki Pissinou, and Sundararaj Iyengar. Game theory for cyber security and privacy. *ACM Computing Surveys*, 50:1–37, 05 2017.

[24] Alex Dali and Christopher Lajtha. Iso 31000 risk management: “the gold standard”. *EDPACS*, 45(5):1–8, May 2012.

[25] Karel Durkota, Viliam Lisy, Christofer Kiekintveld, and Branislav Bosansky. Game-theoretic algorithms for optimal network security hardening using attack graphs. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS ’15, pages 1773–1774, Richland, SC, 2015. International Foundation for Autonomous Agents and Multiagent Systems.

[26] Russell C. Eberhart and Yuhui Shi. Comparison between genetic algorithms and particle swarm optimization. In *Proceedings of the 7th International Conference on Evolutionary Programming VII*, EP ’98, page 611–616, Berlin, Heidelberg, 1998. Springer-Verlag.

[27] Marcel Frigault, Lingyu Wang, Anoop Singhal, and Sushil Jajodia. Measuring network security using dynamic Bayesian network. In *Proceedings of the 4th ACM Workshop on Quality of Protection*, QoP ’08, page 23–30, New York, NY, USA, 2008. Association for Computing Machinery.

[28] Richard D. Gilson. Special issue preface. *Human Factors*, 37(1):3–4, 1995.

[29] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[30] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[31] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Commun. ACM*, 63(11):139–144, October 2020.

[32] Samuel N. Hamilton, Wendy L. Miller, Allen Ott, and O. Sami Saydjari. Challenges in applying game theory to the domain of information warfare. In *In 4th Information survivability workshop (ISW-2001/2002*, 01 2002.

[33] Matthew H. Henry and Yacov Y. Haimes. A comprehensive network security risk model for process control networks. *Risk Analysis*, 29(2):223–248, 2009.

[34] J. Hird, M. Hawley, and C. Machin. Air traffic management security research in SESAR. In *2016 11th International Conference on Availability, Reliability and Security (ARES)*, pages 486–492, 2016.

[35] Hannes Holm, Matus Korman, and Mathias Ekstedt. A Bayesian network model for likelihood estimations of acquirement of critical software vulnerabilities and exploits. *Information and Software Technology*, 58:304 – 318, 2015.

[36] ISO/IEC 2018. Technical report.

[37] Manish Jain, Vincent Conitzer, and Milind Tambe. Security scheduling for real-world networks. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, pages 215–222, Richland, SC, 2013. International Foundation for Autonomous Agents and Multiagent Systems.

[38] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes, 2013.

[39] R. Koelle. Security situation management: Towards developing a time-critical decision making capability for sesar. In *Availability, Reliability and Security (ARES), 2014 Ninth International Conference on*, pages 544–551, Sept 2014.

[40] R. Koelle and M. Hawley. Sesar security 2020: How to embed and assure security in system-of-systems engineering? In *Integrated Communications, Navigation and Surveillance Conference (ICNS), 2012*, pages E8–1–E8–11, April 2012.

[41] Aron Laszka, Xenofon Koutsoukos, and Yevgeniy Vorobeychik. *Towards High-Resolution Multi-Stage Security Games*, pages 139–161. Springer International Publishing, Cham, 2019.

[42] Antonio Marotta, Gabriella Carrozza, Luigi Battaglia, Patrizia Montefusco, and Vittorio Manetti. Applying the SecRAM methodology in a cloud-based ATM environment. In *Proceedings of the 2013 International Conference on Availability, Reliability and Security*, ARES '13, page 807–813, USA, 2013. IEEE Computer Society.

[43] Karol Gotz Martin Hawley. Universty of Twente, SecRAM tutorial slides. `https://telluur.com/utwente/master/CRM%20-%20Cyber%20Risk%20Management/Weeks%205-10%20%28Assignment%29/SESAR%20%28literature-slides%29/Gotz%20and%20Hawley%20-%202016%20-%20SESAR%20SecRAM%20Tutorial.pdf`, 04 2016.

[44] Sjouke Mauw and Martijn Oostdijk. Foundations of attack trees. In Dong Ho Won and Seungjoo Kim, editors, *Information Security and Cryptology - ICISC 2005*, pages 186–198, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[45] M.A. McQueen, Wayne Boyer, M.A. Flynn, and G.A. Beitel. Quantitative cyber risk reduction estimation methodology for a small scada control system. pages 226– 226, 02 2006.

[46] Jelena Mirkovic and Peter Reiher. A taxonomy of DDoS attack and DDoS defense mechanisms. *SIGCOMM Comput. Commun. Rev.*, 34(2):39–53, April 2004.

[47] T. Morita, S. Yogo, M. Koike, T. Hamaguchi, S. Jung, I. Koshijima, and Y. Hashimoto. Detection of cyber-attacks with zone dividing and PCA. *Procedia Computer Science*, 22:727 – 736, 2013. 17th International Conference in Knowledge Based and Intelligent Information and Engineering Systems - KES2013.

[48] D. P. Morton, F. Pan, and K. J. Saeger. Models for Nuclear Smuggling Interdiction. *IIE Transactions*, 38(1):3–14, 2007.

[49] M. Narkus-Kramer, R. Stroup, and J. Merkle. Achieving NextGen's air traffic management operational improvements. *Aerospace and Electronic Systems Magazine, IEEE*, 24(7):10–13, 2009.

[50] Matthew Nunes. Dynamic Malware Analysis kernel and user-level calls. `https://zenodo.org/record/1203289#.YFhi-1OeQ5k`, 2018.

[51] Frans A. Oliehoek, Rahul Savani, Jose Gallego-Posada, Elise van der Pol, Edwin D. de Jong, and Roderich Gross. GANGs: Generative adversarial network games, 2017.

[52] F. Pasqualetti, F. Dörfler, and F. Bullo. Attack detection and identification in cyber-physical systems. *IEEE Transactions on Automatic Control*, 58(11):2715–2729, Nov 2013.

[53] J. B. Pearson and E. B. Stear. Kalman filter applications in airborne radar tracking. *IEEE Transactions on Aerospace and Electronic Systems*, AES-10(3):319–329, 1974.

[54] James Pita, Manish Jain, Janusz Marecki, Fernando Ordóñez, Christopher Portway, Milind Tambe, Craig Western, Praveen Paruchuri, and Sarit Kraus. Deployed ARMOR protection: The application of a game theoretic model for security at the Los Angeles International Airport. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems: Industrial Track*, AAMAS '08, pages 125–132, Richland, SC, 2008. International Foundation for Autonomous Agents and Multiagent Systems.

[55] James Pita, Milind Tambe, Chris Kiekintveld, Shane Cullen, and Erin Steigerwald. GUARDS: Game theoretic security allocation on a national scale. In *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '11, page 37–44, Richland, SC, 2011. International Foundation for Autonomous Agents and Multiagent Systems.

[56] Hemant Rathore, Swati Agarwal, Sanjay K. Sahay, and Mohit Sewak. Malware detection using machine learning and deep learning. *Lecture Notes in Computer Science*, page 402–411, 2018.

[57] Bruce Schneier. Attack Trees - Modeling security threats. *Dr. Dobb's Journal*, December 1999.

[58] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy - Volume 1: ICISSP,*, pages 108–116. INSTICC, SciTePress, 2018.

[59] Eric Shieh, Bo An, Rong Yang, Milind Tambe, Craig Baldwin, Joseph DiRenzo, Ben Maule, and Garrett Meyer. PROTECT: A deployed game theoretic system to protect the ports of the United States. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '12, page 13–20, Richland, SC, 2012. International Foundation for Autonomous Agents and Multiagent Systems.

[60] Shailendra Singh and Sanjay Silakari. An ensemble approach for cyber attack detection system: A generic framework. In *Proceedings of the 2013 14th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, SNPD '13, page 79–84, USA, 2013. IEEE Computer Society.

[61] Safeeullah Soomro, Mohammad Riyaz Belgaum, Ruchin Jain, and Zainab Alansari. Review and open issues of cryptographic algorithms in cyber security. In *2019 International Conference on Computing, Electronics Communications Engineering (iCCECE)*, 08 2019.

[62] G. Sreenu and M A Durai. Intelligent video surveillance: a review through deep learning techniques for crowd analysis. *Journal of Big Data*, 6:48, 06 2019.

[63] Milind Tambe. *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned.* Cambridge University Press, USA, 1st edition, 2011.

[64] Chee-Wooi Ten, Manimaran Govindarasu, and Chen-Ching Liu. Cybersecurity for critical infrastructures: Attack and defense modeling. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 40:853–865, 07 2010.

[65] Christian Urcuqui. Malicious and Benign Websites. `https://www.kaggle.com/xwolf12/malicious-and-benign-websites/home`, 2017.

[66] Vorontsov K. V. Moscow state university, machine learning introduction. `http://www.machinelearning.ru/wiki/images/f/fc/Voron-ML-Intro-slides.pdf`, 2018.

[67] Vasin and Morozov. Moscow State University, Introduction to game theory. `http://techlibrary.ru/b/2j1a1s1j1o_2h.2h.,_2u1p1r1p1i1p1c_2j.2j._2j1c1f1e1f1o1j1f_1c_1t1f1p1r1j2f_1j1d1r_1s_1q1r1j1m1p1h1f1o1j2g1n1j_1l_2e1l1p1o1p1n1j1l1f._2003.pdf`, 2003.

[68] Y. Wang, J. Wong, and Andrew Miner. Anomaly intrusion detection using one class SVM. In *Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, 2004.*, pages 358 – 364, 07 2004.

[69] Linda Weiland and G Wei. Evaluating the impact of NextGen's air traffic system on aviation security. *MATEC Web of Conferences*, 189:10030, 01 2018.

[70] Shiuh-Ku Weng, Chung-Ming Kuo, and Shu-Kang Tu. Video object tracking using adaptive Kalman filter. *J. Vis. Comun. Image Represent.*, 17(6):1190–1208, December 2006.

[71] David D. Woods. *Tasks, Errors, and Mental Models*, chapter Coping with Complexity: The Psychology of Human Behaviour in Complex Systems, pages 128–148. Taylor & Francis, Inc., Bristol, PA, USA, 1988.

[72] Peng Xie, Jason Li, Xinming Ou, Peng Liu, and Renato Levy. Using Bayesian networks for cyber security analysis. In *2010 IEEE/IFIP International Conference on Dependable Systems Networks (DSN)*, pages 211–220, 09 2010.

[73] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training BERT in 76 minutes, 2019.