Évaluation dynamique de risque et calcul de réponses basés sur des modèles d'attaques bayésiens

François-Xavier Aguessy

THALES

THÈSE DE DOCTORAT CONJOINT TELECOM SUDPARIS et L'UNIVERSITE PIERRE ET MARIE CURIE

Spécialité

École doctorale : Informatique

Présentée par

François-Xavier AGUESSY

Pour obtenir le grade de
DOCTEUR DE TELECOM SUDPARIS

Évaluation Dynamique de Risque et Calcul de Réponses Basés sur des Modèles d'Attaques Bayésiens

---

Soutenue le 22 09 2016
devant le jury composé de :

Eric TOTEL
Professeur, Centrale-Supélec / *rapporteur*

Radu STATE
Professeur, Université du Luxembourg / *rapporteur*

Bruno DEFUDE
Professeur, Telecom SudParis / *examinateur*

Guillaume DOYEN
Maitre de conférences, Université de Technologie de Troyes / *examinateur*

Grégory BLANC
Maitre de conférences, Télécom SudParis / *examinateur*

Olivier BETTAN
Responsable du laboratoire de cybersécurité, Thales Services / *encadrant de thèse*

Hervé DEBAR
Professeur, Télécom SudParis / *directeur de thèse*

Vania CONAN
HDR, Thales Communications & Security / *co-directeur de thèse*

Thèse No : 2016TELE0016

Acknowledgements

First, I would like to warmly thank my thesis director, Hervé, for his trust since the beginning of my PhD, and for his kind support during these three years. He has advised my scientific research with great expertise and involvement.

Many thanks to Olivier that has supervised my work at Thales. In addition to being a friendly manager, he has always found the right balance between proposing new projects to get new ideas and preserving time to make the PhD work progress.

Thanks to Vania, my thesis co-director in Thales. His advice has always been precious to be capable of getting the bigger picture, even when day-to-day work was absorbing.

Thanks to each one of my Thales's colleagues from both Gennevilliers and Palaiseau. They have been a real motivation to continue this work with perseverance. Thanks for the rich discussions we had to imagine how to make the world a better place.

Thanks to my research team of Telecom SudParis. Thank you to Gregory for his numerous detailed proofreading and continual enhancement proposal. Many thanks to the other PhD student for their precious advice.

I could not end these acknowledgements without thanking my family, and in particular my beloved wife Sarah. She has been the most important daily support I could imagine. Thanks to Capucine, that came into the world at the end of this work and which illuminates our lives. She immediately slept through the night, which made much easier this thesis writing.

Abstract

Information systems concentrate invaluable resources, generally composed of the computers, and servers that process the data of an organisation. They constitute an increasingly attractive target for attackers. Given the number and complexity of attacks, security teams need to focus their actions on the most important attacks, in order to select the most efficient security controls. Because of the threat posed by advanced multi-step attacks, it is difficult for security operators to fully defend against all vulnerabilities when deploying countermeasures. Deploying intrusion detection sensors to monitor attacks exploiting residual vulnerabilities is not sufficient and new tools are needed to assess the risk associated with the security events produced by these sensors. In this PhD thesis, we build a complete framework for static and dynamic risk assessment, leveraging prior knowledge on the information system (*e.g.*, network topology, vulnerabilities, *etc.*) and dynamic events (*e.g.*, intrusion alerts, attack detection, *etc.*), to propose responses to prevent future attacks.

First, we study how to remediate the potential attacks that can happen in a system, using logical attack graphs. We build a remediation methodology to remove the most relevant attack paths extracted from a logical attack graph. In order to help an operator to choose between several remediation candidates, we rank them according to a cost of remediation combining operational and impact costs. We implement this method using MulVAL attack graphs and several publicly available sets of data.

Then, we study the dynamic attacks that can occur in a system. Although attack graphs were proposed to represent known multi-step attacks that may occur, they are not directly suited for dynamic risk assessment. Several extensions of static risk assessment models have been proposed in the literature to accommodate dynamic risk assessment, but they suffer from common limitations, such as existing cycles. We present how static risk assessment models can be generalised in a Generic Attack Model. Then, we present how to extend Generic Attack Models to build two new dynamic risk assessment models to evaluate the attacks that are the most likely. First, the Bayesian Attack Model (BAM), a Bayesian network-based extension to the Generic Attack Model, built to handle cycles and to suit various information system configurations. Then, we extend the Bayesian Attack Model as the Hybrid Risk Assessment Model (HRAM). This hybrid model is subdivided in two complementary models : (1) Dynamic Risk Correlation Models, correlating a chain of alerts with the knowledge on the system to analyse ongoing attacks and provide the hosts' compromise probabilities, and (2) Future Risk Assessment Models, taking into account existing vulnerabilities and current attack status to assess the most likely future attacks. We study the sensitivity of their probabilistic parameters and of the parameters of the input Generic Attack Model. Finally, we validate the accuracy and usage of both these dynamic risk assessment models in the domain of cybersecurity, by building them from topological attack graphs.

Résumé

Les systèmes d'information sont d'une valeur inestimable, car ils rassemblent l'intégralité des systèmes utilisés pour le stockage et le traitement des données d'une organisation. Ils représentent donc une cible de plus en plus attractive pour les attaquants. Les opérateurs de sécurité doivent se concentrer sur les attaques les plus importantes pour sélectionner les mesures de sécurité les plus efficaces. Cependant, ils n'arrivent pas à se protéger de toutes les vulnérabilités, notamment à cause de la menace des attaques multi étapes. Il n'est pas suffisant de déployer des sondes de détection pour surveiller l'exploitation de ces vulnérabilités résiduelles et les opérateurs de sécurité ont besoin de nouveaux outils pour évaluer le risque associé aux événements remontés par ces sondes. Dans cette thèse de doctorat, nous construisons une méthodologie complète d'analyse statique et dynamique de risque prenant en compte la connaissance à priori d'un système avec les événements dynamiques, afin de proposer des réponses permettant d'empêcher les attaques futures.

Tout d'abord, nous étudions comment corriger les attaques potentielles qui peuvent arriver dans un système, en s'appuyant sur les graphes d'attaque logique. Nous proposons une méthodologie de remédiation qui consiste à corriger les chemins d'attaque les plus significatifs extraits du graphe d'attaque logique. Les remédiations candidates sont classées en fonction de leur coût opérationnel et leur impact sur le système. Cette méthodologie est mise en oeuvre à partir des graphes d'attaque générés par MulVAL et en s'appuyant sur des sources ouvertes de données.

Bien que les graphes d'attaques permettent de représenter précisément les attaques multi étapes connues qui peuvent se produire dans un système d'information, ils ne peuvent pas être directement utilisés pour supporter l'évaluation dynamique de risque. Plusieurs extensions de ces modèles statiques ont été proposées dans l'état de l'art, mais celles-ci possèdent des limitations, par exemple la gestion des cycles. Nous présentons donc, tout d'abord, comment les modèles statiques d'analyse de risque peuvent être généralisés dans un modèle d'attaque générique. Nous pouvons alors étendre ce modèle pour construire deux modèles d'analyse dynamique de risque. Le premier modèle, le modèle d'attaque bayésien, est une extension du modèle d'attaque générique basé sur des réseaux bayésiens, mais capable de gérer les cycles et donc applicable à n'importe quel système. Le deuxième modèle, le modèle hybride d'évaluation de risque en est une extension. Celui-ci est divisé en deux modèles complémentaires : (1) les modèles de corrélation de risque, qui corrèlent un ensemble d'alertes avec la connaissance sur le système pour analyser les attaques en cours et fournir les probabilités de compromission des états du système. (2) les modèles d'évaluation du risque futur, qui prennent en compte les vulnérabilités existantes et l'état courant des attaques, pour évaluer les attaques futures les plus probables. La sensibilité des paramètres probabilistes et des paramètres du modèle d'attaque générique pris en entrée est évaluée sur les deux modèles. Enfin, nous validons l'exactitude des résultats et l'utilisation de ces deux modèles d'évaluation dynamique de risque en les construisant à partir de graphes d'attaque topologiques.

Contents

Chapter 1

Introduction

1.1 Context and objective

According to the National Institute of Standards and Technology (NIST), an information system is a "discrete set of information resources organised for the collection, processing, maintenance, use, sharing, dissemination, or disposition of information" [Kis13]. Thus, an information system concentrates invaluable information resources, generally composed of the computers and servers that process the data of an organisation. Information systems are increasingly complex: they contain heterogeneous equipment having specific vulnerabilities, protected by complex security policies implemented on distributed policy enforcement points (*e.g.*, firewalls, intrusion prevention systems). Given the number and complexity of attacks, security teams need to focus their actions on the most important attacks, in order to select the most efficient security controls. In critical environments, security operators generally know most of the vulnerabilities of their information system thanks to the regular use of a vulnerability scanner, a software assessing locally or remotely the vulnerabilities of equipment or software. Unfortunately, many vulnerabilities are not patched, either because patching may disrupt critical services, or because these vulnerabilities are not a high priority for system administrators, regarding the general security policy and the limited resources (time and money) available. As a result, security operators need to regularly assess the level of security of their information system, taking into account the newly detected vulnerabilities, to prevent the most critical or likely attacks.

According to Mandiant [Man16], in 2015, only 47% of breaches were detected internally. This is due to the fact that the most impacting attacks are advanced attacks composed of several successive steps. Each step in itself may be illegitimate (*e.g.*, the exploitation of a vulnerability in software) or legitimate (*e.g.*, a user with administrators privilege accessing sensitive data). For example, an attacker first subverts a client computer using a spear-phishing email exploiting a software vulnerability, then attacks the Active Directory using another software vulnerability to obtain administrator privileges, and, thanks to these privileges, accesses with legitimate credentials a database server that contains sensitive data. Even if a few single attack steps of such attacks are detected, they are not perceived as critical on their own. In order to defend against the complete sequence of steps and assess the associated risks, we need to model multi-step attacks showing that the sequence of steps results in a greater impact than the sum of its individual vulnerabilities. However, risk assessment, and in particular dynamic risk assessment (*i.e.*, regular update of risk assessment at operational time, according to the occurring attacks) is difficult. In order to formalise such multi-step attacks, several models have been proposed, mainly tree- or graph-based models. An attack graph, for example, is a risk analysis model containing all the

paths an attacker may follow in an information system. Attack graphs have been studied for more than 15 years. So, they are reaching to be industrialised and several tools generate attack graphs. Their use is attractive because they use already available information (vulnerability scans and network topology). However, they scale with difficulty and thus are applicable mainly to small and medium networks.

As a first line of defence, security operators deploy protection measures (*e.g.*, software patches and perimeter access control with firewalls) to prevent attacks. Unfortunately, there are always possible bypasses. So, as a second line of defence, security operators deploy sensors (*e.g.*, Host or Network Intrusion Detection Systems) generating alerts when an attacker attempts to exploit a vulnerability. When these sensors produce security events, operators need to evaluate the risk brought by ongoing attacks, to respond appropriately: this process is called dynamic risk assessment. Operators currently lack automated tools to correlate this dynamic data with the prior-knowledge they have about their information system. This correlation is done manually, with the only support of security dashboards to visually concentrate such information. The impact analysis being left to the operators skills and knowledge about the system. It is thus limited and cannot apply to the complexity of large-scale information systems.

According to the National Information Assurance Glossary [oNSS10], a risk is "*a measure of the extent to which an entity is threatened by a potential circumstance or event, and typically a function of 1) the adverse impacts that would arise if the circumstance or event occurs; and 2) the likelihood of occurrence*". As a result, the risk is generally considered in Information Security Management Systems (ISMS) as the combination of the likelihood of the exploitation of vulnerabilities and their impact on the system. According to NIST in [Nat12], the determination of the likelihood of occurrence of the attacks takes as input the potential threat sources and the attack predisposing conditions (*e.g.*, the vulnerabilities). Once the likelihood of attacks has been assessed, the next step is to determine their magnitude of impact. Finally, from likelihood and impact, we can compute the risk. In order to make risk assessment dynamic, the process is maintained over time and its results have to be communicated regularly to security management operators.

Once either static or dynamic risk assessment has been performed by security operators, they have to find means to decrease the risk, by deploying responses. The deployment of responses has to take into account from one side the potential attacks that have to be remediated and their impact. On the other side, it has to take into account the response itself and the impacts it might have on the whole system (whether good or bad). This is also done manually, since there is no comprehensive tool to dynamically assess the risk of multi-step attacks and evaluate responses to decrease this risk.

This PhD thesis is in the field of information system security. It aims at providing a complete framework for static and dynamic risk assessment including prior knowledge on the information system (*e.g.*, network topology, vulnerabilities, *etc.*) and dynamic events (*e.g.*, intrusion alerts, attack detection, *etc.*), and proposing responses. In this framework, we focus on the likelihood component of the risk, for both attacks and responses. Indeed, methodologies to estimate likelihood do not depend on the system in which they are implemented, contrary to the impact assessment which may require adaptation for the target organisation. This framework applies to information systems that are (1) targeted by complex attacks, (2) where system administrators have a relatively good knowledge of their information system: they know the system cartography, most of the vulnerabilities thanks to vulnerability scanners and (3) in which they have deployed detection mechanisms. This framework may apply to different types of environments, in which a high level of security is needed, *e.g.*, critical information systems, cyberphysical-systems or even industrial systems.

1.2 Challenges in current security approaches

Building a framework for static and dynamic risk assessment brings four main scientific challenges. The first challenge is the methodology to build a dynamic risk assessment model merging different information sources: the prior-knowledge of network configurations and vulnerabilities, and the dynamic security events. The second challenge is the scalability of the model generation and processing. The third challenge is the expressiveness of the model, to take advantage of the interesting properties of each possible representation of the model (topological, logical, probabilistic), without suffering from the associated drawbacks with respect to the previous scalability objectives. The last main challenge is the usability of the model, in order to make it useful for security operators, in particular by enabling the computation of responses to the attacks.

1.2.1 Challenge 1 : methodology to build a dynamic risk assessment model

The first main scientific challenge of this PhD thesis is the definition of a methodology to build a dynamic risk assessment model merging different information sources. In order to enable dynamic risk assessment, the model has to take into account, from one side, the prior-knowledge that security operators have one their system (*e.g.*, the network topology, the vulnerabilities and their exploitability metrics). On the other side, it has to take into account the dynamic security events happening in the system (*e.g.*, correlated alerts, human reports). The initial models of the state of the art (*e.g.*, attack trees [Sch99, Ing09] or attack graphs [PS98, OGA05a, JNO05, Art02]) have been designed for static risk assessment, so they do not contain information about attack detections, nor do they represent attack status (see sections from 2.2.1 to 2.2.3 for more details). These models have been extended to dynamic risk assessment models (*e.g.*, attack nets [McD00] or Bayesian attack graphs [LM05, FW08, FWSJ08], see sections 2.2.4 and 2.2.5 for more details). However, these extensions have limitations, such as the ability to handle topological cycles, and thus are not applicable for information systems of arbitrary topologies and size. Thus, the state of the art misses a dynamic risk assessment model that can apply to any information systems, to correlate the prior-knowledge with dynamic events.

1.2.2 Challenge 2 : scalability

The second challenge we face when using attack graphs-based models for risk assessment is scalability. This challenge already occurs for static risk assessment. It is much more important and is a barrier for dynamic risk assessment adoption. An attack graph is a model defined globally for an information system, containing all attacks that can be carried out. Thus, it can be huge for a real system. When we transform this model for dynamic risk assessment, new nodes are added (*e.g.*, places and transitions of attack nets [McD00, DMCR06] or attack action nodes and local observation nodes of Bayesian Networks for Cyber Security [XLO$^+$10], see sections 2.2.4 and 2.2.5 for more details). Moreover, in probabilistic models, the graph structure must be acyclic to compute the probabilities of attacks, which is generally not the case of attack graphs (*e.g.*, [LM05] and [PDR12] mention the challenge of cycles in Bayesian attack graphs). One solution to the cycle challenge is to explode the cycles (*e.g.*, this solution is mentioned in [OBM06] and the explosion of an attack graph in several attack paths is described in [KCBC$^+$09]), but it strongly increases the size of the model, thus the scalability challenges. Thus, the state of the art misses a dynamic risk assessment model that is scalable, even if the input attack model contains cycles.

1.2.3 Challenge 3 : expressiveness

Several approaches exist to represent the attacks that can happen in an information system. First, the *logical models* represent an attack as a logical predicate requiring successful preconditions for the attack to be possible. This is the case, for example, of attack trees and its extensions [Sch99, BP03, RP05, CY07], logical attack graphs [PS98, SHJ+02, OGA05a] and attack nets and its extensions [McD00, DMCR06, PML09] (see sections from 2.2.1 to 2.2.4 for more details). This type of model accurately represents the process by which humans estimate whether or not an attack is possible. However, they are very verbose and even for information systems of a few machines, they can be huge and grow rapidly with the number of components of the information system. As a result, they become quickly inappropriate for humans or even machine processing. *Topological models* provide a higher-level view of the attacks possible in an information system, by representing an attack as a way to gain access from a machine to a new one. This is the case, for example, of topological attack graphs [JNO05, JNK+11, Art02] (see subsection 2.2.3 for more details). However, topological models do not explain with details how attacks are done and thus are less accurate. Finally, *probabilistic models* assign probabilities to hosts or attack steps in the information system. This is the case, for example, of Bayesian attack graph and its extensions [QL04, DLK04, LM05, FW08, XLO+10] (see subsection 2.2.5 for more details). However, these models generally have strong requirements in their structure (*e.g.*, they must not have any cycle). So, there exist at least three main representation of attacks in the state of the art. Each one has advantages and drawbacks. Thus, the state of the art misses a dynamic risk assessment model that can take advantage of the interesting properties of each type of attack representation, without suffering the associated drawbacks.

1.2.4 Challenge 4 : usability

Once we have succeeded in building a usable dynamic risk assessment model, the last significant challenge is the usability of this model, in order to make it useful for security operators. In the state of the art, the attack models are generally used to support the computation of responses. For example, we can use them to improve the detection [NJ08, GTHM14] or to select the best hardening configurations to deploy [NJOJ03], to measure the risk reduction after applying a response [MBFB06] or even to compute all hardening options in a network [WNJ06] (see section 3.5 for more details). Selecting the response to deploy is not an easy task, as it must take into account, both the attack to be remediated, and the impact the response may have on the system. So, the last challenge is to exploit the new opportunities brought by the dynamic risk assessment model, to improve the selection of responses.

1.3 Thesis statement

In order to solve the challenges described above, we propose in this thesis a risk assessment and response framework that brings together three contributions: dynamic risk assessment model, scalability improvement and support to response computation. The first one is the building of a dynamic risk assessment model. We define an explicit model and process for handling cycles. This process is supported by a clear definition of a set of model parameters. The sensitivity of the model toward these parameters is studied in the validation. Second, we provide a significant performance improvement in terms of number of nodes and vulnerabilities that an explicit model can include, over the existing state of the art. While classic Bayesian attack graph models are usually demonstrated over a few nodes and vulnerabilities, we show that our model can be realistically computed at the scale of an enterprise information system. Third, we detail how the use of the

risk assessment model can support the computation of responses. From a logical model, we build a methodology to select the best set of remediations that should be deployed to prevent an attack. We also show how a probabilistic risk assessment model enriches the granularity of probability of potential or occurring attacks, thus improving the selection of efficient responses.

1.3.1 Contribution 1: dynamic risk assessment model

The first contribution of this PhD is the definition of a risk assessment model, correlating the alerts and a priori-knowledge on the system, to assess the current risk status of an information system and predict possible futures. This contribution aims at partially resolving the first and third challenge of sections 1.2.1 and 1.2.3 and is detailed in chapter 5 and chapter 6. This contribution has been highlighted in the conference paper *Hybrid Risk Assessment Model based on Bayesian Networks* presented in the 11th International Workshop on Security [ABBCD16] and was awarded best student paper.

The model we build is capable of representing the attacks that may occur and the ones that are ongoing. It outputs probabilities that attacks have succeeded and that assets may have been compromised. These probabilities provide security operators with the capability to manage priorities according to the criticality of ongoing attacks. This model can either be used for static risk assessment, to predict the attacks that are the most likely to happen, or for dynamic risk assessment, by taking into account the correlated alerts that are received with the known attack steps and vulnerabilities. In order to take advantage of the multiple representations of attack models (logical, topological, and probabilistic). The model we define forms a topological representation, but is based on a probabilistic model, to represent both the logical conditions necessary to carry out the attacks, and the probabilities of occurrence of the attacks according to the context.

1.3.2 Contribution 2: scalability improvement

The second main contribution is the scalability improvements of the risk assessment model. This contribution aims at resolving the second challenge of subsection 1.2.2 and is detailed in chapter 5 and chapter 6. This contribution has been highlighted in the conference paper *Hybrid Risk Assessment Model based on Bayesian Networks* presented in the 11th International Workshop on Security [ABBCD16].

In this PhD thesis, we present models supporting dynamic risk assessment. They contain all the attack paths that could be followed by an attacker in a system. Due to the cycle breaking process, redundancy is introduced in the models, according to possible sources or destinations of attacks. As a result, there are a huge number of paths in the models from which most are very unlikely to have been followed by attackers. So, in our models, we add pruning functions allowing to select for further exploration the paths that are the most likely to have been exploited or likely to happen and thus the analysis is efficient even for medium-sized information systems. Moreover, with the gathering of hosts or servers that are similar for the attack model (same vulnerabilities and authorised accesses), the risk assessment model can even apply to bigger information systems.

1.3.3 Contribution 3: support to response computation

The last main contribution of this PhD is the ability to use risk assessment models to support the computation of responses. This contribution aims at partially resolving the fourth challenge of subsection 1.2.4 and is detailed in chapter 4 and at the end of chapter 7. This contribution has been highlighted in the conference paper *Remediating Logical Attack Paths Using Information System*

Simulated Topologies presented in the Computer & Electronics Security Applications Rendez-vous 2014 [AGBC14] (*i.e.*, computation of remediations for logical attack paths) and in the conference paper *Adjustable Fusion to support Cyber Security Operators* presented in the Human Aspects of Information Security, Privacy, and Trust: Third International Conference, Held as Part of HCI International 2015 [AODLLF15] (*i.e.*, usability for cybersecurity operators of the analyses based on attack graphs).

The risk assessment model gives the attacks that are the most likely to happen and the possible futures for those that are happening. Then the responses are evaluated regarding their ability to prevent those likely attacks. Finally, the costs and impacts of the responses are taken into account to choose the best response candidates that will be proposed to the operators. We develop in chapter 4 a methodology to compute remediations relying on a static attack model: logical attack graphs. Then, at the end of chapter 7, we present how we can extend this methodology to compute responses to occurring attacks, by leveraging the advantages of dynamic risk assessment models.

This thesis is organised as follows: chapter 2 presents the state of the art of attack models. Chapter 3 presents the state of the art of the computation of responses using attack models. Then, chapter 4 introduces a methodology to compute remediations to potential attacks using logical attack graphs. Chapters 5 and 6 presents two dynamic risk assessment models: the Bayesian Attack Model and the Hybrid Risk Assessment Model. Chapter 7 validates these models in the domain of cybersecurity, by building them from topological attack graphs. Chapter 8 concludes this work and presents future work.

Chapter 2

State of the art of attack models

At an organisational level, several methods help to analyse the risk of information systems and keep those systems secure. For example, ISO/IEC 27000 [ISO14] is the Information Security Management Systems (ISMS) Family of Standards providing recommendations on security, risk and control in an information system. In particular, ISO/IEC 27005 [ISO11a] describes a methodology to manage the risks while implementing an ISMS. Another well-known method for the analysis of risks in information systems is EBIOS (Expression of Needs and Identification of Security Objectives) [Sec04]. These standards present global methodologies to manage risks in organisations. They generally combine (1) technical tools (*e.g.*, vulnerability scanner) to assess, for example, the vulnerabilities of the target system and the likelihood of attacks and (2) organisational methodologies (*e.g.*, stakeholder interviews) to identify the critical assets and consequences of successful attacks. The technical tools may rely on a model of the system to represent all the legitimate or attack actions that are possible in the modelled system. In this PhD thesis, we focus on those technical approaches. Thus, in this first chapter, we survey the state of the art of attack models in the literature. This study aims at evaluating the limitations of the usage of attack models for dynamic risk assessment.

From this state of the art, we build a Generic Attack Model, an attack model generalising the most interesting properties of the attack models of the state of the art and that can apply to any domain. This will be the main input from which we build the Bayesian Attack Model described in chapter 5 and the Hybrid Risk Assessment Model described in chapter 6.

2.1 Graph-based models

As a many attack models are based on graphical models (*e.g.*, in the last 20 years, more than 30 papers present a new graph-based attack model), such as trees, graphs, Bayesian networks or Petri nets, we will first present in this section the main definitions and the notations regarding the graph-based models. These concepts are directly used in section 2.2, to describe the attack models of the state of the art.

2.1.1 Graphs

2.1.1.1 Graphs, directed graphs and directed acyclic graphs

According to the Encyclopedia of Mathematics [Haz89],:

Definition 1 *A* **graph** *is a set V of* vertices *and a set E of unordered and ordered pairs of vertices; denoted by $G(V, E)$. An unordered pair of vertices is said to be an* edge*, while an ordered pair is said to be an* arc*. A graph containing edges alone is said to be non-oriented or* undirected*; a graph containing arcs alone is said to be oriented or* directed*.*

Starting from this definition of graph we define additional notations that will be useful all along this thesis:

- A *loop* l is an edge which begins and ends at the same vertex:
  $l = (v_e, v_e) \in E$, with $v_e \in V$.
  For example, in the graph of Figure 2.1a, there is a loop on the vertex v_6.
- The *degree* $deg(v)$ of a vertex $v \in V$ is the number of edges and arcs incident to the vertex (*i.e.*, v is one of the two vertices of the edge):
  $deg(v) = Card(\{e = (v, v_i) \in E, \forall v_i \in V\} \bigcup \{e = (v_i, v) \in E, \forall v_i \in V\})$
  For example, in the graph of Figure 2.1a, $deg(v_0) = 2$, $deg(v_1) = 3$ and $deg(v_2) = 4$.
- A *path* from $v_1 \in V$ to $v_n \in V$ is a sequence of edges in which all vertices v_i are different except the starting and ending vertices:
  $\{e_i = (v_i, v_{i+1}) \in E, i \in \{1..n-1\}, \forall i, j \in \{2..n-2\}, v_i \neq v_j\}$.
  The definition is identical for *directed paths* in directed graphs.
  For example, in the graph of Figure 2.1a, a path from v_5 to v_3 is displayed in red. It is composed of the three edges $(v_5, v_2), (v_2, v_4), (v_4, v_3)$.
- A graph is *connected* if any pair of distinct vertices is connected by at least one path:
  $\forall (v_1, v_n) \in V \times V, v_1 \neq v_n \implies \exists p = \{e_i = (v_i, v_{i+1}) \in E, i \in \{1..n-1\}, \forall i, j \in \{2..n-2\}, v_i \neq v_j\}$.
  Figure 2.1a shows an example of a connected graph of 7 vertices $(v_i, i \in \{1..7\})$.
- A graph is *fully-connected* or *complete* if any pair of distinct vertices is connected by at least one edge: $\forall (v_1, v_2) \in V \times V, v_1 \neq v_2 \implies \exists e = (v_1, v_2) \in E$.
  In Figure 2.1a, the subgraph constituted of nodes v_0, v_1 and v_2 and the green edges is fully connected.
- A (simple) *cycle* is a non-empty closed path (*i.e.*, a path from $v_n \in V$ to v_n):
  $\{e_i = (v_i, v_{i+1}) \in E, i \in \{1..n-1\}, v_1 = v_n, \forall i, j \in \{2..n-2\}, v_i \neq v_j\}$ The definition is identical for *directed cycles* in directed graphs.
  For example, in the graph of Figure 2.1a, a cycle from and to v_2 is displayed in green. It is composed of the three edges $(v_2, v_0), (v_0, v_1), (v_1, v_2)$.

In a *directed graph* $G(V, A)$,

- The *parent* or *source* of an arc $(v_1, v_2) \in A, v_1 \in V, v_2 \in V$, is v_1.
  In Figure 2.1b, the vertex v_1 is the parent of the arc (v_1, v_2)
- The *child* or *destination* of an arc $(v_1, v_2) \in A, v_1 \in V, v_2 \in V$, is v_2.
  In Figure 2.1b, the vertex v_2 is the child of the arc (v_1, v_2)
- The *incoming arcs* of a node v are all the arcs for which v is the child:
  $\forall a = (v_1, v) \in A$, with $v_1 \in V$.
  Figure 2.1c shows an example of a graph in which the node v has 3 incoming arcs: a_1, a_2 and a_3.

- The *outgoing arcs* of a node v are all the arcs for which v is the parent:
  $\forall a = (v, v_2) \in A$, with $v_2 \in V$.
  Figure 2.1d shows an example of a graph in which the node v has 3 outgoing arcs: a_1, a_2 and a_3.
- the *indegree* $deg^-(v)$ of a vertex $v \in V$ is the number of arcs in A whose destination is the vertex v:
  $deg^-(v) = Card(\{v_i, \forall v_i \in V, (v_i, v) \in A\})$
  In Figure 2.1c, $deg^-(v) = 3$.
- the *outdegree* $deg^+(v)$ of a vertex $v \in V$ is the number of arcs in A whose source is the vertex v:
  $deg^+(v) = Card(\{v_i, \forall v_i \in V, (v, v_i) \in A\})$
  In Figure 2.1d, $deg^+(v) = 3$
- a *root* is a vertex $v \in V$ for which $deg^-(v) = 0$ (no incoming arc).
  In the example of Figure 2.1e, there are two roots: v_0 and v_1.
- a *sink* is a vertex $v \in V$ for which $deg^+(v) = 0$ (no outgoing arc).
  In the example of Figure 2.1e, there are three sinks: v_3, v_4 and v_5.

A *directed acyclic graph* is a directed graph in which there are no cycles.

The example of Figure 2.1e is a directed acyclic graph.

2.1.1.2 `AND-OR` graphs

An *`AND-OR` graph* is a directed graph in which each vertex is either an `OR` or an `AND`. It is the graphical representation of the reduction of objectives into conjunction and disjunction of sub-objectives. A vertex represents a sub-objective and according to its type (`AND` or `OR`), it requires either the conjunction or disjunction of its children, to be fulfilled. A root node of an `AND-OR` graph can be called *precondition* as it does not require any other node to be fulfilled.

Figure 2.2 shows an example of an `AND-OR` graph with 6 nodes (v_0 to v_5). Nodes v_0, v_1 and v_2 are preconditions, they do not require any previous node to be fulfilled. Nodes v_3 and v_4 are `AND` nodes, they require the verification of all their parents to be fulfilled. For example, the node v_3 requires the verification of both v_1 and v_2 to be fulfilled. Nodes v_5 is an `OR` node, it requires the verification of at least one of its parents to be fulfilled. For example, the node v_5 requires the verification of v_3 or v_4 to be fulfilled.

2.1.1.3 Trees, directed trees and polytrees

According to the Encyclopedia of Mathematics [Haz89],

Definition 2 *A* **tree** *is a connected non-oriented graph which does not contain cycles.*

Thus, a tree $T(E, V)$ is a graph in which any two vertices are connected by exactly one path. In a tree, a *leaf* l is a vertex with one incident edge: $l \in E, deg(l) = 1$. Figure 2.3 shows an example of a tree with 5 vertices ($v1$ to $v5$). It contains 3 leaves, the red vertices: $v1$, $v4$ and $v5$.

A *directed tree* is a directed graph which would be a tree if the directions of the edges were ignored.

A *polytree* is a directed acyclic graph in which the related undirected graph is a tree. In other words, it is a directed acyclic graph for which there are no undirected cycles either.

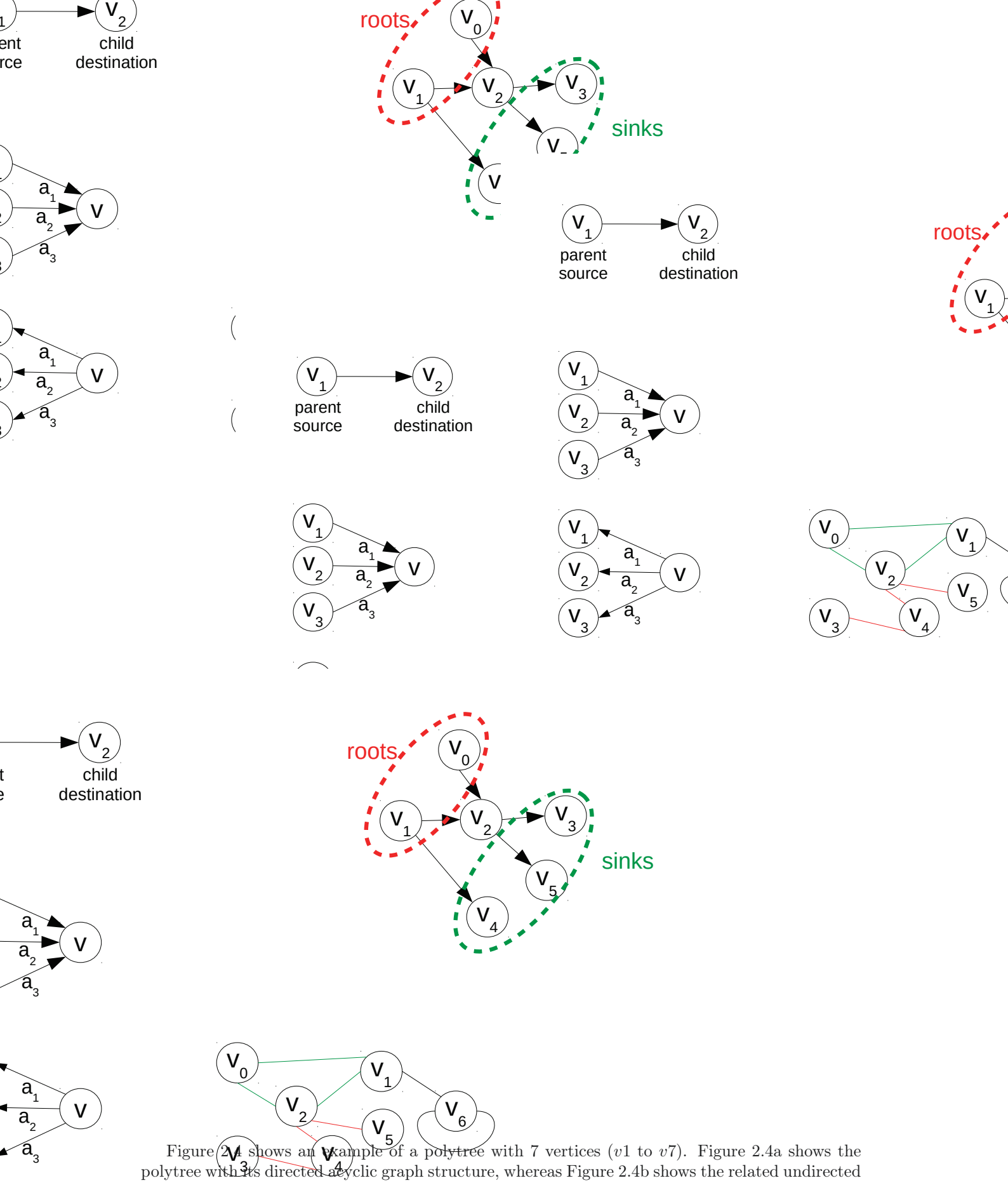


Figure 2.4 shows an example of a polytree with 7 vertices ($v1$ to $v7$). Figure 2.4a shows the polytree with its directed acyclic graph structure, whereas Figure 2.4b shows the related undirected tree. Each new arc in Figure 2.4b would create an undirected cycle and prevent the undirected graph to be a tree. Thus, any new directed arc in the directed acyclic graph of Figure 2.4a would prevent this graph of being a polytree.

A *AND-OR tree* is an `AND-OR` graph which is a directed tree.

FIGURE 2.2 – Example `AND-OR` graph

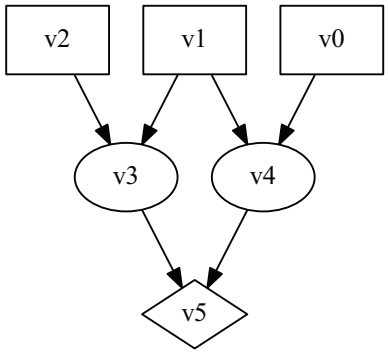
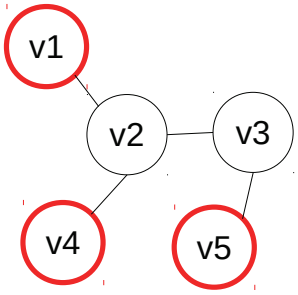


FIGURE 2.3

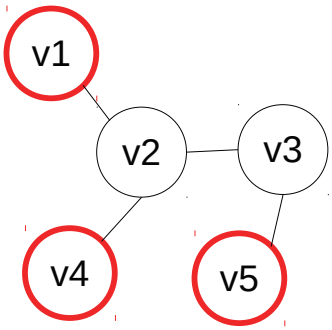


2.1.2 Bayesian networks

A *Bayesian network*, also called *belief* or *causal network*, is a probabilistic graphical model introduced by Judea Pearl [Pea86]. It can be modelled as a Directed Acyclic Graph, where nodes represent random variables that can be in mutually exclusive states, and edges represent causal dependencies among variables. For discrete random variables, these dependencies can be specified using a conditional probability table associated with each child node, enumerating its probability, according to the states of its parents.

Definition 3 *A **conditional probability table** of a discrete random variable child, toward a set of discrete random variables parents is a table containing the probabilities of each state of the child, according to all possible values of all parents.*

Bayesian networks are specially interesting for bidirectional *inference*, *i.e.* computing the probability of each state of all nodes of the network, given evidence, *i.e.* nodes that have been set to a specific state. In the general case, exact inference is a NP-hard problem and can be done efficiently only on small networks, using the algorithm of Lauritzen and Spiegelhalter [LS88]. However, if the structure of the graph is a polytree, it can be done in quasi-linear time, using Pearl's Belief Propagation Algorithm [Pea88]. It is also possible to do approximate inference using methods based on Monte-Carlo sampling [Pea87].



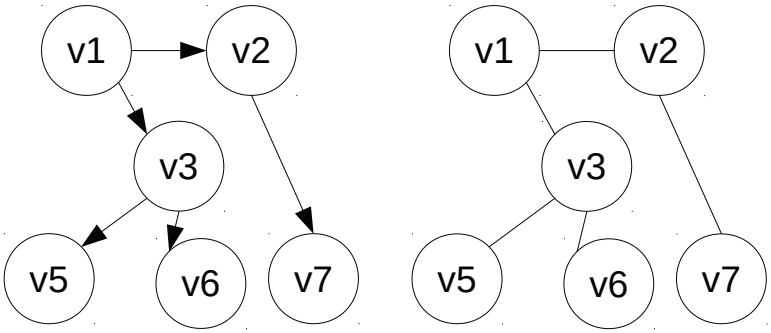
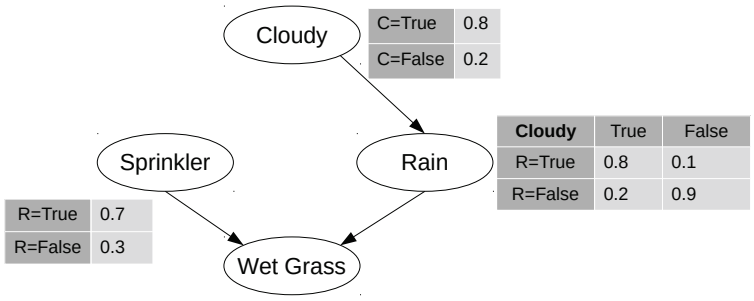


Figure 2.5 shows a famous simple example of Bayesian network. It describes the causal relationship of wet grass in a garden with the weather and the running of a sprinkler.



This network contains 4 nodes representing random variables:

Cloudy: The random variable representing whether or not the weather is cloudy.

Sprinkler: The random variable representing whether or not the sprinkler was on during the night.

Rain: The random variable representing whether or not it has rained during the night.

Wet Grass: The random variable representing whether or not the grass is wet this morning.

Each node is associated with its conditional probability table describing its relation toward its parent(s). Each node may be observed and the observations (evidence) can be used to compute the probabilities of the other random variables. For example, if we observe that the grass is wet, and the weather is not cloudy, we fix the states of the *Wet Grass* node to *true* and *Cloudy* node to *false*. Then the Bayesian inference gives:

$$P(Sprinkler = True | WetGrass = True \cap Cloudy = False) = 0.959 \tag{2.1}$$

$$P(Rain = True | WetGrass = True \cap Cloudy = False) = 0.145 \tag{2.2}$$

Thus, in this case, it is much more likely that the sprinkler was on during the night, rather than it has rained.

A Bayesian network is *a direct representation of the world, not a reasoning process* [PR98]. Contrary to, for example, `AND/OR` graphs in which an arc represents a logical deduction and the flow of information goes in the direction of the arc, in a Bayesian network, an arc represents a causal dependency and, during inference, the reasoning process can propagate in any direction (either prediction or abduction)

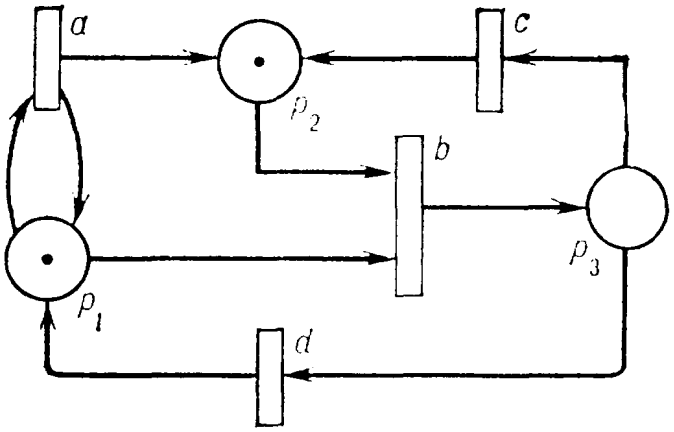
2.1.3 Petri nets

According to the Encyclopedia of Mathematics [Haz89],

Definition 4 *A* **Petri net** *is a mathematical model of discrete dynamical systems [...] introduced by C. Petri in the 1960-s. A Petri net is a set* $N = (T, P, F, M_0)$*, where* T *is a finite set of symbols called* transitions*,* P *is a finite set of symbols called* places*,* $P \cap T = \emptyset$*,* $F : T \times P \cup P \times T \to \{0, 1\}$*, is an* incidence function *and* $M_0 : P \to \{0, 1, ...\}$ *is an* initial marking*.*

A Petri net is generally represented with an oriented graph where places are represented by circle vertices, transitions by rectangle vertices and token by dots located on places, as shown in Figure 2.6. This example Petri net contains 3 places (p_1, p_2 and p_3) and 4 transitions (a, b, c and d). One token is on place p_1 and another on place p_2.

FIGURE 2.6 – Example of a Petri net [Haz89]



The places represent conditions and the transitions the events that may occur. The incidence function, representing the arcs between places and transitions describe which places are post- or pre-conditions of a transition. From an initial state of the conditions (M_0), the net operates by passing markings from place to place, when transitions are *fired*, in discrete time. The number of marks of a place is generally called *token*. A transition may be fired, if it is *enabled*, *i.e.* when they are sufficient tokens on its inputs places.

Petri nets are generally used for simulations, with or without an execution policy determining the behaviour of the simulation. They are non-deterministic as any enabled transition may fire.

Many extensions of the elementary Petri Nets (as presented above) exists: coloured Petri nets, deterministic timed transitions Petri nets, Stochastic Petri nets, *etc.* Each extension enriches the Petri formalism to answer new issues.

For example, one of the most used extensions of Petri Nets are *coloured Petri Nets* [Jen87]. It adds the support of values contained in each token. With memory, tokens can contain information about past states and this information (colour) can be used to validate transitions. Moreover, coloured Petri Nets generalise most important properties of Petri Nets.

A *disjunction Petri net* is very close to a base Petri net. The only exception is that only one of all the incoming arcs is necessary to enable a transition.

A *deterministic timed transition Petri net* (DTTPN), is a coloured Petri net where transitions represent a time delay to go from one place to another.

An *interval timed coloured Petri net* is an extension of coloured Petri nets where timestamps are associated with tokens and transitions with firing delay.

2.2 Attack models

Having presented the main definitions and concepts of graph-based models, we now survey the attack models of the state of the art. Note that in this state of the art, we will not enter into the details of attack-response models, as these models rely on the usual attack models (*i.e.*, the ones that will be presented here) and only add new nodes, or new features to include responses into the model. As a result, they do not bring new capabilities to represent potential nor occurring attacks.

The first models for multi-step attack modelling of the state of the art are tree-based models such as attack trees. They have been quickly extended by graph-based models such as attack graphs. However such models are not directly suited for dynamic risk modelling. As a result, Petri net extensions such as attack nets and Bayesian network extensions such as Bayesian attack graphs include ongoing attack probabilities and add new nodes for dynamic risk modelling.

2.2.1 Attack trees and models based on trees

Schneier presents in [Sch99] the *attack trees*: `AND-OR` trees in which nodes represent security relevant attack actions that can be refined recursively until they become atomic actions. Attack trees have been formalised by Mauw and Oostdijk in [MO05]. In [Ing09], Ingoldsby details how attack trees can be used for Threat Risk Analysis. Paul proposes in [Pau14] a systematic approach to automate the construction and maintenance of big attack trees and applies this methodology to industrial use cases, such as the Galileo risk assessment program. In an attack tree, there is only one attacker's main goal: the root of the tree. It is a static model, as it cannot model dependencies between nodes: an arc leads to a more accurate description of an attack element.

As attack trees have been inspired by research in safety and reliability, Brooke and Paige

extends in [BP03] *fault trees* applicability to security. It is an increment of attack trees. This model includes well-known concepts from safety analysis: additional connectors (*e.g.*, priority `AND`, exclusive `OR`), specific nodes type (*e.g.*, basic events, conditioning events, undeveloped events, *etc.*) and transfer symbols used to break up large trees into several smaller ones. However it has the same limitations as attack trees.

Ray and Poolsapassit extend attack trees as *augmented attack tree* [RP05], to model how far an attacker has progressed in an information system. The main difference with a standard attack tree is a couple of integers associated with each node of the augmented attack tree. The first element of this couple represents the current attack status of a node. It can vary in time when the attacks happen. The second element represents the least effort to compromise the related node. The ratio between the two integers give the current compromise probability. This model is thus dynamic and can model the progression of an attacker, but applies to only one main attack.

Ordered Weighted averaging (OWA) trees are used by Yager in [Yag06] to represent attacks in a way very similar to attack trees. OWA operators are more expressive than the `AND-OR` operators of attack trees. In OWA trees, operators can represent quantifiers such as *most*, *some*, *half of*, *etc.* which model better uncertainty. They have the same limitations as attack trees.

Enhanced Attack Trees are a model presented by Camtepe and Yener in [CY07] which adds the support of temporal dependencies between components and expiration of attack elements. In order to do that, a new type of node is introduced: `Ordered-AND`. Several attributes are added to each node: *Time-To-Live* which defines a lifetime for actions, *Attack Level* and *Attack Probability* which define the ratio of accomplished actions and the probability of attacks completed of a subgoal. This model comes with the *Enhanced Tree Automaton* whose goal is to take as input a stream of aggregated alerts messages and to output the corresponding enhanced attack tree. This automaton can be extended into an *Enhanced Parallel Automaton*, in order to handle complex attacks which are combinations of several enhanced attack trees. Arnold *et al.* also extends attack trees with a sequential operator in [AHPS14]. When each leaf of such a *Time-dependant Attack Tree* is annotated with a probability distribution of the time needed for this step, it can be propagated in the entire tree to obtain the time distribution for the whole attack. The sequential operator is formally defined by conjunctive nodes with a notion of progress of time. Jhawar *et al.* give in [JKM+15] the first formal description of an equivalent of *Enhanced Attack Trees* without explicit notion of time, but a more abstract notion of causality. This model is called *SAND attack trees* as `Sequential-AND` is another name for `Ordered-AND`. These models enhance attack trees with more accurate operators, but are still static and allow to represent only one main attack.

In a nutshell, the tree structure is the first to model accurately successive attack elements. It is thus the basis of nearly all the other attack models. The tree structure is very simple, it is thus generally too simple to model exactly complex succession of elements. Original attack trees are not very convenient to model progress of attackers in an information system for at least 3 reasons: (1) they do not provide the ability to model the position of an attacker (or of several attackers) in the model, (2) they do not describe the dependencies between attack elements , (3) they do not allow to model several attacks. Thus they have been extended in several models, to represent dependencies between attack elements (Fault Trees for Security), or to model the progression of an attacker (Augmented attack trees). However, without additional models such as the automaton and the list of trees in the enhanced attack trees, the models based on trees are not rich enough to represent occurring attacks.

2.2.2 Models based on direct acyclic graphs

A *cryptographic direct acyclic graph* presented by Meadows in [Mea98] is a graph-based model defined for visual description of cryptographic protocol attacks. The nodes represent the attack

stages and the arcs model dependencies between the stages toward an attacker's objective. Visual methods are used to represent the difficulty of each stage. Cryptographic DAGs are an informal model, in which nodes contain an unstandardized text describing the attack stages. However it is one of the first to model dependencies between attack actions. But, as in attack trees and their extensions, a cryptographic DAG does not provide the ability to model the position of an attacker and was not designed to model several attacks.

In a nutshell, the direct acyclic graph structure is a logical evolution of the tree structure, in order to model more complex elements. It is more powerful than trees, but does not allow to model all attacks, particularly because it does not contain cycles.

2.2.3 Attack graphs and models based on directed graphs

The main limitation of attack trees is that they only describe one main attack, they cannot represent several different attacks in the same model. On the contrary, an attack graph is an attack model containing several different attacks using a graph rather than a tree.

An attack graph is a model regrouping all the paths an attacker may follow in an information system. It has been first introduced by Phillips and Swiler in [PS98]. This widely used formalism covers many heterogeneous models. There are two major types of attack graphs: logical attack graphs and topological attack graphs. Logical attack graphs are closer to attack trees. They are based on `AND-OR` logical directed graphs. The nodes are logical facts describing the actions that can be carried out by an attacker, or the pre-requisites to carry them out, and the edges represent the dependency relations between the nodes. Topological attack graphs are based on directed graphs. The nodes represent topological assets (host, IP address, *etc.*) and the edges represent possible attacks between such nodes. A summary of the state of the art on the early papers about attack graphs (from 2002 to 2005) has been presented by Lippmann and Ingols [LI05], a more recent one by Kordy *et al.* [KPCS13]. Shandilya *et al.* also discuss in [SSS14] the state of the art of attack graph generation and their use in security systems

In [SPEC01], Swiler *et al.* present how to generate an attack graph from security attributes and vulnerabilities in computer networks. Sheyner *et al.* present in [SHJ+02] an automated technique to build an attack graph using symbolic model checking. Thus, contrary to the work of Swiler *et al.* that concentrates on malicious events, it can also take into account benign system events that can be part of an attack. Moreover, this symbolic model checking technique works backward (starting from the initial state) and thus do not explore unreachable paths or vulnerabilities. One difficulty to build an attack graph is to have an accurate description of the vulnerabilities as input. In [SMC09], Schuppenies *et al.* present a data structure to specify vulnerability information used for attack graph generation. They also detail how they fill this data structure with information from the National Vulnerability Database[1] using the Common Vulnerabilities and Exposures[2], the Common Vulnerability Scoring System [FIR15], and the Open Vulnerability and Assessment Language [3].

The tools that have been developed to automatically generate attack graphs from network topology, service description and vulnerability scans are called attack graph engines. There are three engines of note: MulVal, TVA and NetSPA. MulVAL, The Multi-host, Multi-stage Vulnerability Analysis Language tool is an open-source logical attack graph engine created by Ou *et al.* [OGA05a, OBM06]. It uses Datalog, a logic programming language, in order to generate an attack graph (XML file) in which nodes are related to each other with logical relations (`OR` or `AND`). As for

[1] NIST National Vulnerability Database, `https://nvd.nist.gov/`
[2] MITRE Common Vulnerabilities and Exposures, `https://cve.mitre.org/`
[3] NIST Open Vulnerability and Assessment Language `http://oval.mitre.org/`

topological attack graphs, the two main engines are Topological Vulnerability Analysis tool (TVA) presented by Jajodia *et al.* [JNO05, JNK+11] (later commercialised under the name *Cauldron*) and Artz's NetSPA [Art02].

McQueen *et al.* present in [MBFB06] the *compromise graphs*, based on directed graphs. They use this model to assess the efficiency of technical security measures (*e.g.*, system hardening or addition of firewalls) on risk reduction and apply it on SCADA Control Systems. The nodes of the compromise graphs represent the stages of an attack, detailing how a given target can be compromised. The edges are weighted according to the estimated time required to complete the compromise. This time is modelled thanks to a random process and is used to compare several paths in the graph. This model does not take into account the dependencies between identical vulnerabilities on different machines.

An example of the implementation of attack graphs containing information about attackers has been presented by Wu *et al.* in [WFM+07, Wu09]: the *Incident Graph* (I-Graph). This model is a type of attack graph in which each intrusion goal is represented by a node in an `AND-OR/Quorum` graph. The I-Graph is generated semi-automatically using two types of data: vulnerability descriptions and system services description. The system services are described within a direct graph where nodes represent services and edges represents an intrusion-centric channel, *i.e.*, if the origin node of an edge is compromised, the destination can spread to the destination node. Then, when an attack occurs, attack sub-graph instances are created and updated at runtime, to model happening concurrent or overlapping attacks. The I-Graph is then used in the 3 versions of ADEPTS. ADEPTS I provides automated response to multi-stage security attacks (treats only survivability challenges). ADEPTS II proposes a means to compute efficiently optimal responses to these attacks. Then, ADEPTS III adds the consideration in this model of 0-day attacks.

In a nutshell, attack trees and attack graphs are very similar models enabling the representation of multi-step attacks with respectively one or several attack goals. They contain very accurate description of attacks (with logical attack graphs/trees) or more high-level vision of attacks (with topological attack graphs/trees). However, these models alone are not adapted to directly represent ongoing attacks, to contain multiple attackers or to be triggered by alerts and detections. To model ongoing attacks, attack graphs models have been completed with incident graphs, built during runtime when attack elements occur.

2.2.4 Attack nets and models based on Petri nets

An attack net is a multi-step attack model which was presented by McDermott *et al.* in [McD00]. It is based on a disjunctive Petri net in which (1) places represent security states of entities in the system, (2) transitions represent events, commands or data that causes the change of state of one entity, (3) tokens indicate the progress of an attack. The place of the token represents the fact that the attacker has gained control of the corresponding entity, in the state represented by the place. To change place, an attacker must be in the previous place(s) and validate the transition(s).

In [DMCR06], Dalton *et al.* present a method to build a Generalised Stochastic Petri net from an attack tree. The basic operations on attack trees (conjunction and disjunction) can indeed be very easily modelled using Petri nets. Places model a goal or subgoal of an attack, and transitions model a means used to achieve a goal, associated with its difficulty and estimated duration. It is also possible to add in the model the probability of success or failure of attack elements, and then, with the simulation of the Petri net, to estimate which attacks are more likely to happen.

The model presented by Dahl and Wolthusen in [DW06] is based on interval timed coloured Petri nets. It is an extension McDermott's attack nets in which transitions are associated with an interval of time. It is applicable to multi-agent and multi-stage attacks.

In [PML09], Pudar *et al.* propose a novel model based on attack nets to extend attack trees: PENET. The models used in PENET are Deterministic Timed Transitions Petri Nets, a model close to the one used in Attack nets, but a bit simpler, because the time of a transition is deterministic, rather than a random variable. Places represent attacker places of interest, subgoals and main goal, and transitions represent the time delay needed to compromise the next goal. A specific colour (attack precondition) may be required to pass a transition. If a transition is fired, its time delay is the time for an attacker to compromise next state. Two main computations can be done from this model: (1) the build of the coverability tree, to find the most likely attack and (2) time-domain analyses (time to reach the root place, number of possible intrusion in a time period, *etc.*).

The Key-Challenge Petri Net model described in [KVK09] by Kiviharju *et al.*, is based on coloured and stochastic Petri nets in which places represent the protected information security items with one or several of the following properties: Confidentiality, Integrity and Availability. Tokens represent players (attackers or defenders) and transitions represent a challenge an attacker must answer before entering a new place (*i.e.*, an attack).

The topological model presented by Henry *et al.* in [HLZ10] is based on simple Petri nets. Places model privileges acquired by an attacker. Transitions model escalation of privileges succeeded thanks to an attack. It defines new metrics to quantify risk.

In a nutshell, the main advantage of attack models based on Petri nets compared to graphs or trees is that they allow to model concurrency and progress of several attacks with labelled tokens in the same model. However, the models have the following limitations: (1) it cannot be used to model several privileges for an attacker (a token can only be at one place), (2) it is not easy to adapt the model to take into account the addition of tokens during runtime, according to a detection or alert, (3) the Petri net formalism cannot be changed or adapted. It has been created to know if states are accessible, by simulation, not to represent complex attack behaviour.

2.2.5 Bayesian attack graphs and models based on Bayesian networks

Qin *et al.* present in [QL04] an extension of attack trees using a Bayesian network. They describe this model as especially adapted to evaluate the likelihood of attack goals or to predict future attacks. What is quite interesting in their approach is that they use class/type of attacks rather than exact attack description, which can decrease significantly the size of the model. This model can describe accurately dependencies between attack elements, but cannot model the position of attackers and attack elements are too general to describe a real attack. Moreover, this formalism can describe only one attack in the same model.

Halfway between attack graphs and Bayesian networks, the behaviour based Bayesian network introduced by Dantu *et al.* in [DLK04] is a model that takes into account the estimated behaviour of an attacker, according to its attack profile, to improve the vulnerability analysis. It uses an attack graph to represent relations between attack steps, and a Bayesian network for risk inference, using probabilities taking into account the attacker behaviour. This model is not really adapted for modelling ongoing attacks, but is rather used to analyse a risk level.

A Bayesian attack graph, introduced by Liu and Man in [LM05], is built of nodes which are random variables representing a host in a specific system state (a true state means that the host is compromised) and edges representing possible exploits that can be instantiated from a source host to a target node. The major concern of building such a Bayesian network from an attack graph is due to the structure of a Bayesian network: it must be acyclic, while attack graphs often contain cycles. To avoid cycles, this paper assumes that an attacker will never backtrack once reaching a compromised state, but does not detail how such assumption is used to build the model. This

model is well adapted to compute the posterior probability (how likely an attacker may carry out an attack) and giving an observation to deduce the most probable attack paths that may have been followed by the attacker. However, it is not easy to represent the path of several attackers in the same model.

The knowledge-based Bayesian Model of Althebyan and Panda is constructed thanks to two complementary models: a knowledge graph and a dependency graph [AP08]. It has been designed to describe only insider attacks (an insider being defined as *someone who has access to and has some knowledge about an organisation information system*). A knowledge graph is a graph that represents knowledge units of an insider. The Dependency Graph is a hierarchical graph that shows all dependencies among objects in the system. The knowledge Bayesian attack graph is a kind of attack graph which is built with data from the two previous models. For each node, which represents an object in the system, is assigned a probability value. The knowledge Bayesian attack graph models relations between knowledge elements of the information system. It is thus not really practical to describe precisely where an attacker is located and which privilege he may have acquired. But the information contained in this model are complementary and may help to describe how a privilege could be acquired. Moreover, the Bayesian model allows to model precisely the propagation of probabilities in the graph, since the probabilities are not only attached to a node (object disclosure likelihood) but can also be attached to arcs (probability of one or more nodes given the knowledge of one or more nodes).

Frigault and Wang present in [FW08] how Bayesian Attack Graphs are used to calculate security metrics in an information system. They later extend this model in [FWSJ08] to continuously measure security level in a dynamic environment. This model adds temporal factors such as the availability of exploits or patches, to represent the evolution of the severity of a vulnerability. This model also represents dependencies among exploits, to remove the hypothesis that the exploitation of different vulnerabilities are independent. The base model used is a Dynamic Bayesian network, a sequence of Bayesian networks representing a time slice, allowing to monitor and update the system as time proceeds and even predict further behaviour of the system. This model adds the possibility to take into account the evolution of the exploitability of a vulnerability according to the time.

The Bayesian network model presented by Xie *et al.* in [XLO+10] models the uncertainty of occurring attacks. This model improves the usual process to build a Bayesian network from a logical attack graph with three properties: (1) it identifies several types of uncertainty, (2) most of its parameters can be computed automatically, and (3) it is not sensitive to perturbations in the parameters choice. This model adds two new types of nodes dedicated to dynamic security modelling. The addition of the *attack action node*, parent of attack nodes, models whether or not an action of the attacker has been done. This node is completed with a *local observation node* modelling inaccurate observations (IDS alerts, logs, *etc.*). This model also takes into account the possible existence of 0-day vulnerabilities via residual probabilities.

Cole [Col13] uses a Credal network (a Bayesian network with imprecise probabilities) to represent parameters uncertainty and detect attack paths. He demonstrates that the uncertainty is too high for single-step attacks. However, for multi-step attacks, it is possible to achieve high confidence in the detections. The key limitation is scalability. The computational cost of inferences in a Credal network is prohibitive for real network topologies. He thus applies this methodology only on linear topologies (*i.e.*, a linear sequence of exploits that can be followed to accomplish a specific attack goal).

In a nutshell, the Bayesian networks add to the advantages of direct acyclic graphs powerful tools to compute and propagate probabilities between nodes of the graph. Moreover, the dependencies between nodes are not anymore `AND` or `OR` relations, but are probabilities of occurrence with a set of predecessors, which is much more expressive. Finally, when a

state is set as compromised for an attacker (high probability), the attacker can leave again from this node even after several other actions. It is thus a very interesting model as it can model complex dependencies and provide tools to make efficient calculations on this model. However, two important challenges arise when we want to use Bayesian network for modelling ongoing multi-step attacks. First, performance challenges can occur, as the conditional probability table size is exponential in the number of parents for a node. Secondly, a Bayesian network must be based on an acyclic graph which is generally not the case of attack graphs (an attacker can come back to a machine he has already exploited). Heuristics allow to suppress cycles, but they suppress paths that could be followed by an attacker.

2.2.6 Other models

This section presents other attack models that are not directly based on attack trees, attack graphs, attack nets or Bayesian attack graphs, but that are close enough to those models to be interesting in this taxonomy of attack models.

In [PCB10], Piètre-Cambacédès *et al.* present how the boolean logic driven Markov processes (BDMP) model, that was rather created for reliability and availability, can be adapted to security modelling. This model is similar to an attack tree with two more elements: a new kind of link (triggers) and leaves representing attack steps or security events, which are associated with Markov processes. This allows for sequences and simple dependencies modelling, while enabling efficient quantification. This model allows to compute for example the probability for the attacker to reach his objective in a given time or the probability of a sequence of elements. The major advantage of BDMP is that they are dynamic, contrary to attack trees which are static. So, they can model accurately sequence of attack events and can be used for time-dependent analysis. BDMP is also presented as less complicated (and thus more efficient) than Petri nets. However, BDMP does not offer the possibility to model the progress of different attackers in the model and as it is based on a tree is aimed for only one attack goal.

Wang *et al.* present in [WZK13] a framework exploring an attack graph with a Hidden Markov Model (HMM) to represent the probabilistic interactions between system observations and states. The Hidden Markov Model estimates the attack states (consequence of attack on the system), according to uncertain observations. Observations represent the observable subjects that can be used to characterise attack states. Hidden Markov Model presents the same advantages as BDMP, over attack trees. But as this model is built from an attack graph rather than a tree, it can in addition apply to several attacks of a system.

2.3 Comparison of attack models

Table 2.1 summarises the attack models that have been presented in section 2.2. Horizontal lines in the table represent a different family of base-model. The first group on entries contains tree-based attack models, the second one graph-based attack models, the third one Petri net-based models, the fourth one Bayesian network-based models and the last one the Markov processes-based models.

For each model identified by its reference, authors, date and name, 7 elements are presented and compared: (1) the base model used, (2) the model operators, (3) whether the model is logical or topological (column “Topo / Logical”), (4) the maximum number of scenarios that can be contained in a model (column “Max nb scen.”), (5) how the temporal dependency between steps is modelled (column “Temp depend.”), (6) the maximum number of attackers that can be represented in the model (column “Position of attacker”), (7) the probability propagation in the model (column “Prob. propag.”).

2.3.1 Basic model

The basic model represents the graph-based model used as the basis of the attack model.

The possible values are:

Tree: The attack model is based on a tree.

DAG: The attack model is based on a Directed Acyclic Graph.

DG: The attack model is based on a Directed Graph (that may contain cycles).

PN: The attack model is based on a Petri net or one of its extensions (disjunctive Petri net, generalised stochastic Petri net, interval timed coloured Petri net, deterministic timed transitions Petri net, coloured and stochastic Petri net).

BN: The attack model is based on a Bayesian network or one of its extensions (Dynamic Bayesian network).

CN: The attack model is based on a credal network, an extension of Bayesian networks with imprecise probabilities.

MP: The attack model is based on Markov processes.

HMM: The attack model is based on a Hidden Markov Model.

2.3.2 Model operators

The next column compares the operators that can be expressed within the model. Operators can be grouped in 3 categories: simple node dependencies, logical operators, probabilistic operators.

2.3.2.1 Simple dependencies

A simple dependency in a graph-based model is represented by an arc, a directed edge.

Node dependency: The *node dependency* is the simplest dependency modelled by an arc that represents that a node requires its parent to be achieved.

Weighted dependency: The *weighted dependency* is an extension of the *node dependency* in which each arc is also associated with a weight representing, for example, the difficulty or duration of an attack from the parent node to the child node.

2.3.2.2 Logical operators

Logical operators are generally used in logical graph-based models. In such models, each node contains a logical fact and represents a logical operator. This operator describes how the fact is deduced or can be reached from the facts of its children, according to the model. The leaves, the nodes with no children, are generally the starting facts or preconditions from which other facts are deduced.

`AND`**:** The `AND` operator describes the requirement of the achievement of *all* the facts of its children for the logical fact of a node to be achieved.

`OR`**:** The `OR` operator describes the requirement of the achievement of *at least one* fact of its children for the logical fact of a node to be achieved.

`OAND`**:** The `ORDEREDAND` or `PRIORITYAND` operator describes the requirement of the achievement of *all* the facts of its children *in a specified order* for the logical fact of a node to be achieved.

`XOR`**:** The `XOR` operator describes the requirement of the achievement of *exactly one* fact of its children for the logical fact of a node to be achieved.

`Quorum`**:** The `Quorum` operator describes the requirement of the achievement of *at least* N facts of its children for the logical fact of a node to be achieved. The minimum required quorum N being attached to each Quorum node.

OWA Operators: The *Ordered Weighted Averaging (OWA)* operators are a generalisation of the `AND-OR` operators providing a class of operators between `AND` and `OR` (*e.g.*, defining formally *most*, *at least half*, *some*, *etc.*).

2.3.2.3 Probabilistic dependencies

Bayesian networks specify the probabilistic dependencies between nodes using conditional probability tables. Each node is associated with a conditional probability table giving the probability of a node according to the states of its parents.

2.3.3 Topological or logical model

This column compares whether a model has a topological or logical representation of the attacks in an information system. A logical representation of an attack is made of logical facts that are linked together by operators (*e.g.*, as it is done in `AND/OR` trees or graphs). These models generally detail with great accuracy the proceedings of an attack. However, they can be very big and cannot easily be processed by machines nor humans. On the contrary, topological models of attacks follow the topology of the system modelled and represent an attack as a way to exploit a topological asset from another one. Thus, they are less detailed, but, for real systems, are much easier to process or understand by humans.

2.3.4 Maximum number of scenarios in a model

This column describes the maximum number of scenarios that can be represented in the model. Some models (*e.g.*, models based on attack trees) can only represent *one* main attack scenario. Other models (*e.g.*, models based on attack graphs) can represent *several* attack scenarios.

2.3.5 Temporal dependency

This column describes how temporal dependencies between nodes are represented in the model. The temporal dependency can describe either the potential attacks (*i.e.*, the duration of the potential fact of a node or the transition between two nodes) or the attacks that have occurred (*i.e.*, the exact time when attack events happened).

The possible values are:

None: There is no temporal dependency between the nodes of the model. Generally, child nodes contain a more accurate description or a way to realise the fact contained in their parent nodes.

Steps: The temporal dependency is modelled by the succession of nodes in models. For example, a directed arc represents that the parent node happened before the child node, or, the children of a `OAND` node specify the order in which the events should happen.

Time: The model contains temporal information allowing to describe the duration or the time when the fact of a node happened. Thanks to that the temporal dependency can be described accurately using time.

2.3.6 Position of attacker

This column describes the maximum number of attackers that can be represented in the model.

The possible values are:

None: No information about attackers or occurring attacks is contained in the model.

One attacker: The model represents all the attacks that are made by one attacker, or it represents all the attacks that are happening in a system, but without distinguishing attacks of separate attackers.

Several attackers: The model can represent the attacks of several distinct attackers concurrently.

2.3.7 Probability propagation

This column describes how the attack probabilities can be propagated in the model. According to the model, probabilities represent either the probabilities of potential attacks, or the compromise status of assets of a system, according to the occurring attacks.

The possible values are:

None: No probabilities are contained in the model.

Simple: Probabilities may be associated with nodes of the model and simple propagation of probabilities between nodes are possible.

Complex: Each node of the model is associated with both a probability and a table of probability or distribution function describing accurately how the probabilities propagate on this node. Complex mathematical algorithms are also proposed to compute and propagate, both forward and backward, the probabilities of each node of the model.

TABLE 2.1 – Summary chart comparing attack models

Reference	**Authors**	**Year**	**Model Name**	**Basic Model**	**Model Operators**	**Topo / Logical**	**Max nb scen.**	**Temp de-pend.**	**Position of attacker**	**Prob. Propag.**
[Sch99]	Schneier	1999	Attack trees	Tree	`AND` / `OR`	Logical	One	None	None	None
[BP03]	Brooke and Paige	2003	Fault Trees for Security	Tree	`AND` / `OR` / `PAND` / `XOR`	Logical	One	None	None	Simple
[RP05]	Ray and Poolsapassit	2005	Augmented Attack trees	Tree	`AND` / `OR`	Logical	One	Steps	One attacker	Simple
[Yag06]	Yager	2006	OWA Trees	Tree	OWA Operators	Logical	One	None	None	None
[CY07]	Camtepe and Yener	2007	Enhanced Attack Trees	Tree	`AND` / `OR` / `PAND`	Logical	Several	Steps	Several attackers	Simple
[AHPS14]	Arnold *et al.*	2014	Time-dependant Attack Tree	Tree	`AND` / `OR` / `PAND`	Logical	One	Time	None	None
[JKM+15]	Jhawar *et al.*	2015	SAND attack trees	Tree	`AND` / `OR` / `PAND`	Logical	One	Steps	None	None
[Mea98]	Meadows	1998	Cryptographic DAG	DAG	Node dependency	Logical	Several	Steps	None	None
[OGA05a]	Ou *et al.*	2005	MulVAL	DG	`AND` / `OR`	Logical	Several	Steps	None	Simple
[JNO05]	Jajodia *et al.*	2005	TVA	DG	Node dependency	Topo	Several	Steps	None	None
[Art02]	Artz	2002	NetSPA	DG	Node dependency	Topo	Several	Steps	None	None
[MBFB06]	McQueen *et al.*	2006	Compromise Graphs	DG	Weighted dependency	Topo	Several	Time	None	Simple
[WFM+07]	Wu *et al.*	2007	Incident Graph	DG	`AND` / `OR` / Quorum	Logical	Several	Time	Several attackers	Simple
[McD00]	McDermott *et al.*	2000	Attack nets	PN	`AND` / `OR`	Logical	One	Steps	Several attackers	None
[DMCR06]	Dalton *et al.*	2006	Generalised Stochastic Petri Nets	PN	`AND` / `OR`	Logical	One	Time	Several attackers	Simple
[DW06]	Dahl and Wolthusen	2006	Interval Timed Colored Petri Nets	PN	`AND` / `OR`	Logical	One	Time	Several attackers	None
[PML09]	Pudar *et al.*	2009	PENET	PN	`AND` / `OR` / `PAND` / `XOR`	Logical	One	Time	Several attackers	None
[KVK09]	Kiviharju *et al.*	2009	Key-Challenge Petri Net	PN	`AND` / `OR`	Logical	Several	Time	Several attackers	None

TABLE 2.1 – Summary chart comparing attack models

Reference	**Authors**	**Year**	**Model Name**	**Basic Model**	**Model Operators**	**Topo / Logical**	**Max nb scen.**	**Temp de-pend.**	**Position of attacker**	**Prob. Propag.**
[HLZ10]	Henry *et al.*	2010	Coupled Petri Nets	PN	Node dependency	Topo	Several	Steps	Several attackers	Simple
[QL04]	Qin *et al.*	2004	Bayesian Networks for security	BN	Probabilistic dependencies	Logical	One	None	None	Complex
[DLK04]	Dantu *et al.*	2004	Behavior based Bayesian networks	BN	Probabilistic dependencies	Logical	Several	Steps	One attacker	Complex
[LM05]	Liu and Man	2005	Bayesian attack graph	BN	Probabilistic dependencies	Topo	Several	Steps	One attacker	Complex
[AP08]	Althebyan and Panda	2008	Knowledge Bayesian Attack Graph	BN	Probabilistic dependencies	Logical	Several	None	One attacker	Complex
[XLO+10]	Xie *et al.*	2010	Bayesian Networks for Cyber Security	BN	Probabilistic dependencies	Logical	Several	Steps	One attacker	Complex
[FWSJ08]	Frigault and Wang	2008	Dynamic Bayesian Network	BN	Probabilistic dependencies	Logical	Several	Steps	One attacker	Complex
[PCB10]	Piètre-Cambacédès *et al.*	2010	Boolean logic Driven Markov Processes	MDP	`AND` / `OR`	Logical	One	Time	One attacker	Simple
[WZK13]	Wang *et al.*	2013	AG-HMM	HMM	`AND` / `OR`	Logical	Several	Steps	One attacker	Complex

2.4 Analysis of the state of the art of attack models

2.4.1 Analysis of the existing attack models of the state of the art

We have seen in this chapter that many graph-based attack models have been proposed in the state of the art. These models are mainly based on trees, graphs, Petri nets or Bayesian networks. First, attack trees and other models based on trees have been proposed in the literature, issuing from the research in safety and reliability *e.g.*, [Sch99, BP03]. However, these models contain only one main attack refined in several steps or actions. Thus, they are too simple to represent all the possible attacks in a system.

Attack graphs and models based on directed graphs extend attack trees. Their goal is to represent all the attacks that can happen in an information system. They give either a very accurate description of attacks (with logical attack graphs [OGA05a]) or more high-level vision of attacks (with topological attack graphs [JNK$^+$11]). However, these models alone are not adapted to directly represent ongoing attacks, to represent multiple attackers or to be triggered by alerts and detections.

Attack nets [McD00] and models based on Petri nets add to those models the ability to model the concurrency and progress of several attacks in the same model. However they cannot model several privileges for an attacker and cannot be used in runtime with real alerts and detections.

Finally, models based on Bayesian attack graphs [LM05] and Bayesian networks add powerful tools to compute and propagate compromise probabilities in the graph. They model complex dependencies and can be used in runtime to represent the evolution of attacks according to real alerts or detections. Thus it seems to be the most interesting base model of the state of the art for supporting dynamic risk assessment using prior knowledge.

2.4.2 Challenges and limitations of the models based on Bayesian networks

Two important challenges arise when we want to use the models based on Bayesian networks for modelling ongoing multi-step attacks: (1) a possible low performance, and (2) a Bayesian network must be based on an acyclic graph, which is generally not the case of attack models.

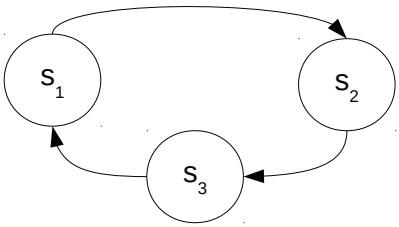
2.4.2.1 Low performances

The first main challenge is due to at least two reasons. The first cause of the performance challenge is the size of the conditional probability tables (CPT), associated with each node of the network. This size is s^p, where s is the number of states of the node (generally $s = 2$ for the states compromised/not-compromised) and p is the number of parents of the node, which can be big in an attack graph, as it usually represents the number of hosts that can attack the node. Thus the size of each conditional probability table can be huge. The second cause of the performance challenge is that the exact inference of the whole network in the general case is NP-hard. In such a network, as soon as there are more than hundreds of nodes, the inference is infeasible.

2.4.2.2 Cycles challenge

Attack models are defined globally for a system, containing all potential attacks that can happen. It thus almost always contains cycles, especially inside local networks in which any host can attack

any other one. For example, a host h_1 may be able to attack another host h_2 that can also attack h_1 (directly or in s
shows an attack gı
s_3 can attack s_1, t



A common assumption to break cycles in attack graphs is that an attacker will not backtrack, *i.e.*, come back on a node he has already successfully exploited. This is reasonable because backtracking does not bring new information about attack paths. It has been properly justified by Ammann *et al.* in [AWK02] and by Liu and Man in [LM05]. However, the solutions of the state of the art for Bayesian modelling of an attack graph such as the ones of Liu and Man [LM05] and Poolsappasit *et al.* [PDR12] use this assumption to delete arbitrary possible attack steps. In reality, it is thus impossible to know a priori which path the attacker can choose. Deleting paths in the Bayesian model thus suppresses possible attacker actions. Ou *et al.* also end at this conclusion in [OBM06] for logical attack graphs and explain that it also applies to other attack graph models as well, even if this challenge has not yet been properly addressed.

2.4.3 Key features of the models of the state of the art

From the comparison of the models of the state of the art of Table 2.1, we gather the seven features of the models of the state of the art that may be interesting to apply to Bayesian networks. First, for the model operators, all logical operators (Node dependencies, `AND`, `OR`, `XOR`, `Quorum`, a subset of OWA operators) can be represented with the probabilistic dependencies of Bayesian networks. However, the ordered dependencies (`PAND`, `PAND`) cannot be represented directly in a Bayesian network node. So, these dependencies should be represented by the succession of nodes representing the order between the attack steps.

The logical representation of attacks is interesting because it details the logical preconditions and how they are used by an attacker to carry out an attack. However this representation is very verbose and it causes the model to be huge for real systems. The topological representation is much more concise and easy to understand for an operator and to process by a computer. However, it does not represent the details of the attacks. The ideal would be to have a hybrid view of attacks, with the conciseness of topological representation of attacks, and the details of the use of preconditions of logical representations.

It is much more interesting to have a model that can contain all possible attack scenarios, because it mutualises the attack steps that can happen in different scenarios. Moreover, when attack steps are detected, they cannot always be attached to a specific attack scenario. Thus, in models that include several attack scenarios, the detected attack steps will highlight iteratively the scenario that is used by the attacker.

The temporal dependency modelled using successive nodes is the one used in Bayesian attack graphs. It represents simply the succession of attack steps. Time dependencies, using either deterministic or random functions is an interesting addition. However, time parameters and func-

tions are particularly difficult to predict (*e.g.*, how long would it take to an attacker to exploit a vulnerability ?).

The ability to represent the position of attackers is the fundamental requirement of a dynamic risk assessment model. It should at least be able to represent a compromise status of several nodes of an information system. It is even better if the model allows to differentiate the propagation of several attackers, each one having different privileges on several hosts of the system, and being able to attack the system, starting from each of these hosts.

Finally, the ability to propagate complex compromise or attack probabilities in the model is a big bonus that is brought directly by the models based on Bayesian networks. Even if this propagation has important computational costs, it is a very interesting feature for security operators to be able to know the most likely attacks that have happened or the steps that may happen next.

2.5 Introducing a Generic Attack Model

Sections 2.2 to 2.4 show that there is a rich variety of attack models in the state of the art. These models have different levels of detail, allow to represent more or less accurately attack elements with several kinds of operators (logical, probabilistic, *etc.*). There is thus a real need to define a *Generic Attack Model*, an attack model that generalises most of the attack models of the state of the art and applies to any domain. Having this Generic Attack Model as entry point will allow us to be able to have a single type of input while being able to apply our process to any type of attack model (attack tree, logical or topological attack graph, *etc.*) and apply to any application domain, with the level of detail wanted by the security operator. The structure of the Generic Attack Model is based on a graph. The building of the Generic Attack Model has three stages.

2.5.1 First stage: states and transitions

The first stage of a Generic Attack Model is the addition of nodes and edges, the basis of all risk models of the state of the art that are based on a graph. Thus, the Generic Attack Model can be represented as a directed graph with nodes and edges. In the Generic Attack Model, we call the nodes *states*, and the edges *transitions*.

Definition 5 *A **state** $s \in S$ of the Generic Attack Model, with S the set of all states, is a node representing a state of an attacker.*

A state is, for example, a privilege on a host, the possession of specific data, a physical or logical location of the attacker, *etc.*

Definition 6 *A **transition** $t(s_{src}, s_{dst}) \in T$, with $s_{src} \in S$, $s_{dst} \in S$ and T the set of all transitions is an edge from a **source** state to a **destination** state, representing that a transition allows an attacker to move from the source state to the destination state.*

- *Each transition has a required **type** of transition, describing how or why the attacker can move between states.*

A transition may be done deliberately by the attacker (*e.g.*, exploitation of a vulnerability, data or credential thief, physical movement, *etc.*) or the attacker may have no part in it (*e.g.*, change in the context, *etc.*).

The first stage of the building of the Generic Attack Model is the minimum necessary to represent the attacks that an attacker can do in a system within a graph (nodes and edges).

Definition 7 *A **Generic Attack Model*** GAM *is a directed graph* GAM(S,T) *where:*

- S *is a set of **states** and,*
- T *is a set of **transitions**.*

2.5.2 Second stage: conditions and requirements

The second stage to enrich the Generic Attack Model is the addition of conditions on the states and of conditional probability tables.

Definition 8 *A **condition** c is a fact that, if verified, impacts the enabling of a transition. It is associated with a probability of success* P(c).

Property 1 ***A transition can be associated with a set of conditions**. In such case, a conditional probability table (cf., Def. 3) specifies the requirements of the transition towards the conditions.*

For the most common and simplest cases of attack models, the conditional probability table of a transition toward its conditions can be specified as a special simple conditional probability table, such as an `OR`, an `AND`, a `XOR`, or a Quorum table.

Property 2 *A **state can be associated with a conditional probability table** specifying the requirements of the state towards its incoming transitions.*

In the same way, for the most common and simplest cases of attack models, the conditional probability table of a state toward its incoming transitions can be specified as a special simple conditional probability table, such as an `OR`, an `AND`, a `XOR`, or a Quorum table.

With the addition of these elements, the Generic Attack Model is able to represent most of the attack models of the state of the art:

- attack trees and attack graphs and most of their extensions, only with `AND`-`OR` conditional probability tables
- Bayesian networks and Petri net-based models, using conditional probability tables.

Conditions are not required as a transition may not need another fact to be possible. If the conditional probability tables are not filled in, we consider by default in the Generic Attack Model that a transition represents an `AND` (it requires all its conditions to be possible) and that a state represents an `OR` (it requires at least one transition to be possible).

2.5.3 Third stage: detection sensors

The addition of detection sensors is the last stage to enrich the Generic Attack Model and to enable its use for dynamic risk assessment.

Definition 9 *A **sensor** of a state, transition, or condition is an oracle issuing an alert when the related element (state, transition or condition) has been detected. It is associated with a false negative and a false positive rate.*

A sensor represents, for example, an Intrusion Detection System, a System Event Management, or a human report.

Even if, in the literature, the models generally associate sensors with either transitions or states, the Generic Attack Model can associate a sensor with any other element of the model (*e.g.*, a condition).

2.6 Conclusion

In this chapter, we have surveyed the state of the art of attack models. These models are based on trees, graphs, Petri nets or Bayesian networks. The first models that were proposed were static and, thus, only fit for static risk assessment (*e.g.*, attack trees, attack graphs). The ones that followed were able to represent the propagation of attackers in the system and were, thus, suitable for dynamic risk assessment.

To the best of our knowledge at the time of this survey, models based on Bayesian networks are the most promising for dynamic risk assessment, since they provide powerful tools to compute and propagate probabilities in the graph. They allow the modelling of complex dependencies, which make them appropriate to represent the evolution of attacks according to real alerts or detections at runtime. However, they raise two challenges when we want to use these models for modelling ongoing multi-step attacks: (1) a possible low performance, and (2) a Bayesian network must be based on an acyclic graph, which is generally not the case of attack models.

From this state of the art, we also gather six features of attack models that are relevant to dynamic risk assessment: (1) the probabilistic dependencies and succession of nodes allowing to represent any other operators, (2) a hybrid representation of attacks combining logical and topological representation of attacks, (3) an attack model containing all possible attack scenarios, (4) a temporal dependency represented either by succession of nodes or by time parameters, (5) the ability to represent the position of several attackers, and (6) the ability to propagate complex attack probabilities in the model.

Finally, as there are many different types of attack models in the state of the art which can be used to represent different kinds of attacks with different granularity, we present the Generic Attack Model, a generalisation of most of the other attack models. This model can be created from the other existing models (*e.g.*, attack tree, logical or topological attack graph) and defines a single format that can be used as starting point to build a dynamic risk assessment model, regardless of the granularity and domain of application of the model. We will present in chapter 7 how we can build a Generic Attack Model from an attack graph.

We will present in the next chapter how papers of the state of the art leverage attack models, to support the computation of responses.

Chapter 3

Response computation using attack models

An attack model describes the prior-knowledge about a system and the potential attacks that may occur inside. Moreover, some of them take into account dynamic information such as alerts or logs, to contain the status of ongoing attacks. Thus, the attack models are particularly interesting to compute responses that are either remediations (security measures allowing to reduce a potential risk) or counter-measures (security measures allowing to reduce an occurring attack).

Remediations and countermeasures to an attack can be grouped in three categories: corrective, active and passive. The only way to really correct a vulnerability is to apply a patch, a piece of software fixing the vulnerability. If such correction to a vulnerability cannot be applied, security operators can choose to apply automatic real-time responses changing the likelihood or the consequences of the exploitation of the vulnerability. Real-time response to intrusion is a major research topic, that is why, since 1996, there are a lot of papers on this subject and different approaches exist. According to the type of responses provided by a system, different names can be found in the literature [SSEJJ12]: (1) *Intrusion Detection System* (IDS), for a system that only provides passive responses (alerts, reports, logs...), (2) *Intrusion Prevention System* (IPS), for a system that also provides a capability to filter infected flows, (3) *Intrusion Response System* (IRS), more generally, to a system that provides other types of responses to an intrusion.

In this chapter we present the state of the art of the response computation based on attack models. In order to do that, we first present the three main types of responses: corrective, passive or active responses. Then, we survey the techniques to compute possible responses to prevent attacks relying on attack models. Finally, we study the criteria and metrics that can be used to choose a response among several candidates. We use this state of the art of response computation using attack models in chapter 4.

3.1 Definition of response, remediation and countermeasures

We use the term *response* to represent an action or group of actions aiming at what is called "risk treatment" in the standard ISO 27000 [ISO14]:

> **risk treatment**: process to modify risk. Risk treatment can involve:
>
> 1. avoiding the risk by deciding not to start or continue with the activity that gives

rise to the risk;

2. taking or increasing risk in order to pursue an opportunity;
3. removing the risk source;
4. changing the likelihood;
5. changing the consequences;
6. sharing the risk with another party or parties (including contracts and risk financing); and
7. retaining the risk by informed choice.

Risk treatments that deal with negative consequences are sometimes referred to as "risk mitigation", "risk elimination", "risk prevention" and "risk reduction". Risk treatment can either create new risks or modify existing risks.

Note that in our context, as we detail in the next sections, the responses rather focus on 1. avoiding the risk, 3. removing the risk source, 4. changing the likelihood, 5. changing the consequences or 7. retaining the risk by informed choice.

A response can be called in different ways, according to the context in which it is deployed. A *remediation* is a response used to treat a risk that is latent, a vulnerability is present in a system, but has not yet been exploited. On the contrary, a *counter-measure* is a response used to treat an on-going risk, toward an occurring attack, exploiting existing vulnerabilities.

3.2 Corrective remediations

The first type of response that we will present is the correction of the exploitable vulnerability. It is a remediation, rather than a countermeasure, as this kind of response is always deployed to treat a latent risk that has been identified, when operators have the possibility (time-wise, patch availability, *etc.*) to deploy it.

The correction of vulnerabilities is generally implemented by patch management software, helping system administrators to deploy patches. Many commercial products already exists to help automate the deployment of patches. Moreover, it has been standardised by several national or international institutes, such as the National Institute of Standards and Technology (NIST) in [Nat05] or the International Organization for Standardization (ISO) in [ISO11b]. So, this technology can be considered mature.

NIST defines in [Nat05] the principles and methodologies organisations can use to manage exposure to vulnerabilities through the deployment of patches. It also provides guidance about how to prioritise, obtain, test, and apply patches. To implement a patch and vulnerability management program, NIST advise creating a group of people called the Patch and Vulnerability Group (PVG). The duties of the PVG are: (1) create a system inventory, (2) monitor for vulnerabilities remediations and threats, (3) prioritise vulnerability remediation, (4) create an organisation-specific remediation database, (5) conduct generic testing of remediations, (6) deploy vulnerability remediations, (7) distribute vulnerability and remediation information to local administrators, (8) perform automated deployment of patches, (9) configure automatic update of applications whenever possible and appropriate, (10) verify vulnerability remediation through network and host vulnerability scanning, and (11) train administrators on how to apply vulnerability remediation. This standard also defines security metrics for patch and vulnerability management, to measure the effectiveness of the patch and vulnerability management program and apply corrective action as necessary. There are three types of patch and vulnerability metrics:

The susceptibility to an attack: For example, the number of patches, of vulnerabilities or of network services.

Mitigation response time: For example, the response time for vulnerability and patch identification, for patch application or for emergency configuration changes.

Cost: For example, the cost of the Patch and Vulnerability Group, of system administrators, of enterprise patch and vulnerability management tools, or cost of program failures.

According to White [Whi06], existing vulnerability and patch management products can be grouped in 4 main categories:

Patch management software: A patch management software (*e.g.*, IBM Tivoli Endpoint Management for Patch Management, Hercules AVR Security Target) is used to manage patches on a complete information system, even for heterogeneous systems.

Configuration managers: A configuration manager (*e.g.*, Microsoft Systems Management Systems or Puppet) is used to centrally manage a park of machines. Those managers can generally deploy patches on the whole park of machines, from a central point.

Package managers: Generally used on UNIX-based operating systems, package managers (*e.g.*, APT, RPM) are used to install and update software. They are totally adapted to install security patches, especially because they have efficient tools to resolve dependencies.

Vulnerability scanners: A vulnerability scanner (*e.g.*, Nessus[1] or OVAL[2]) is a software that automatically detects, either remotely or locally the vulnerabilities of other software. In order to that, it first looks for the installed software and try to find their exact version. Then it compares the list of software to a knowledge base of vulnerabilities to deduce the probable vulnerabilities. In addition to listing the vulnerabilities of software, vulnerability scanners can also list the patches, when they are available, that can correct those vulnerabilities.

In addition to existing products, research papers have also proposed to enhance patch management. In [CTT05], Chang *et al.* describe an architecture of an automated patch management system for company with a large scale heterogeneous information system. Patch deployment is done in 5 steps: (1) receive information about vulnerabilities and patches, (2) find impacted equipment, (3) plan deployment and tests, (4) test patches on copies of production machines on which they will be applied (*e.g.*, on virtual machines) (5) automatically deploy patches on all machines.

In [NGGT10], Nunez *et al.* present a distributed system to deploy patches. This solves one main challenge of current patch management systems, centralisation. A centralised system is a single point of failure (SPOF) and the deployment of patches with a centralised system can be very slow. The distributed patch management security system is constituted of client nodes receiving updates from several distributed server nodes. The distribution of a patch takes place in two steps. The first one is to divide the set of hosts in the system in groups of around 10 hosts. The patch is sent to one machine of each group, which will be the patch server relay. The second step is the transfer of the patch from each server relay to other machines of the group. Then, the patch can be installed by each machine.

The patch management technology is quite mature but it suffers from limitations detailed by Cavusoglu *et al.* in [CCZ08]:

Patches side effects: The patched application or another application of the information system may not work correctly after the application of a patch. Many tests have to be performed

[1] Tenable Nessus, `http://www.tenable.com/products/nessus`
[2] NIST Open Vulnerability and Assessment Language, `http://oval.mitre.org/`

(often manually) on all platforms to prevent conflicts or functionality changes before applying a patch.

Service restart after patching: Often, the restart of the service (or even the host) is not possible for production services, while being required after applying a patch.

Difficulty to apply patches: Sometimes, the application of a patch needs several manual actions and rather seems like a migration. So, it has to be done carefully, to prevent data loss.

Time lag between vulnerability disclosure and patch availability: The duration between a vulnerability disclosure and the availability of the patch fixing it can be long and depends on the software vendor. So there is a long period or time during which the system stays vulnerable.

Unmaintained software: When software is no longer maintained, because it is too old or its vendor does not exist any more, it has no longer patches or updates, even for known vulnerabilities.

Dependencies on older versions of software: There exists sometimes a dependency on an older version of software (*e.g.*, a web browser) that is required to use a specific business application.

Lack of standard: There is currently no standard for applying patches for different vendors on different platforms. Each software vendor and operating system has its own policies to propose and install patches.

Lack of roll-back: The roll-back mechanism after the installation of a patch is not available or possible for numerous patches. It prevents easy cancellation of the installation of a patch.

The main consequence of these limitations is that patch deployment still requires human intervention. So, there are too many vulnerabilities to patch and system administrators have to select which patches should be deployed in priority.

3.3 Passive responses

As the correction of vulnerabilities using patch management is not always possible, for example on machines hosting critical services that cannot be restarted, security operators may choose to use other types of responses to be protected against exploitation of vulnerabilities. These responses are generally done during exploitation by Intrusion Detection, Prevention or Response systems when they detect that a vulnerability is exploited. The first component necessary to build a system that can detect or respond to intrusion is the installation and configuration of an intrusion detection system. The main goal of such a system is to provide the capability to detect what looks suspicious in network flows, on applications or on a system. Bhuyan *et al.* summarise in [BBK14] the detection techniques that have been proposed in the literature. There are two main mechanisms to detect attacks:

Misuse detection Misuse detection [Axe00] is a knowledge-based detection that generally uses rules or signatures (*i.e.*, a pattern describing precisely what should be detected). Misuse detection has many advantages: it is fast and accurate on known attacks, it describes precisely the intrusion detected and it has very few false positive errors. However, signatures must be very precise, to limit false positive, but they also must not be too numerous, for performance

reasons. As a result, signatures often may not detect all the attack variants that exploit a given vulnerability. The second issue is that there are many vulnerabilities that are not known, and for such vulnerabilities, there are no signatures, so no detection. The last issue is that writing an appropriate signature for an attack or a vulnerability is long and sometimes not possible. So, attacks may not have any signature and they will not be detected by such a system.

Anomaly detection The dual detection method is anomaly detection [BBK14]. It is a behaviour-based detection that uses a model of the normal system behaviour and detects deviations, when a behavioural anomaly happens. Anomaly detection requires periodic measurements, compared to a model. Contrary to misuse detection, anomaly detection can detect unknown attacks and does not necessarily need signatures, but sometimes a training period for the system to learn the habits of the normal behaviour. However, anomaly detection also has several drawbacks. The first one is that it is very difficult to model precisely the normal behaviour of a system. Even with models that include a training period, the training period may contain attacks (which should be abnormal activities) or not contain a rare legitimate activity. And if the model does not represent perfectly the normal behaviour, the detection system will trigger false negatives and false positive alerts. The other drawback of anomaly detection is that it obviously detects only anomalies, but does not know whether or not there is an attack and which attack it is. So the detection is much less informative than a misuse detection.

In conclusion, misuse-based detection must be accurate enough to avoid false positives but must also detect all the variations of attacks related to a vulnerability which is not trivial and sometimes not possible. Moreover, some known attacks and all unknown attacks (0-days) cannot be detected by a misuse-based IDS. Conversely, behaviour-based detection is difficult to configure and produces a lot of false positives. Hybrid techniques use both misuse and anomaly-based techniques to attempt to detect known as well as unknown attacks.

Once an attack has been detected by a misuse-based or behaviour-based Intrusion Detection System, different kinds of responses may be taken. The passive responses regroup the simplest actions that can be done after a detection, aiming at alerting an operator or better detecting an attack. These responses do not aim at stopping the attack. This is generally the first kind of responses deployed by operators, as it can help security operators to know what happened in their information system. A system that only provides passive responses (alerts, reports, logs, *etc.*) is generally called Intrusion Detection System (IDS). Tucker *et al.* present in [TFGB07] a taxonomy of Intrusion Detection Systems.

Using taxonomies of Carver [Car00], Stakhanova *et al.* [SBW07b] and Tanachaiwiwat *et al.* [THC02], here is a list of possible passive responses to an intrusion:

Generate an alert: It is the first obvious response of an automatic system, it informs the security operator that an attack is happening. Alerts could be emails, notifications on a dedicated platform, sending a text message, *etc.*.

Generate a detailed report: This report can contain much information allowing an operator to know what really happened during the intrusion (*e.g.*, the time of the intrusion, the attack severity, the exploit used, the IP address or the account used, *etc.*).

Enable extra logging: When the behaviour of a user is unclear, an appropriate response is to enable extra logging, to collect more information about his activity. This can lead to two advantages. If the attack is proven during the intrusion, the attacker may be stopped. If the attack is proven after the intrusion, the security team may know exactly, thanks to the logs, what has been done.

Enable additional intrusion detection systems: If the intrusion is not sure, it's possible to enable additional intrusion detection systems that are not always enabled because they may be too heavy or slow. These systems may allow the system to ensure if this is really being attacked.

Saving logs on an external storage: During several attacks, the logs could be modified or deleted by the attacker. It may be a good idea to save the logs on a read-only support, to know precisely what the attacker did, after the attack.

3.4 Active responses

The active responses regroup the actions preventing the exploitation of a vulnerability that still exists, after the deployment of the response. They generally use techniques validating the interactions between the attacker and the system. This is for example the case of simple filtering by a firewall or an Intrusion Prevention System. An IPS is a component that adds to the detection functions of an IDS the ability to block packets or flows. An IPS blocks an illegitimate traffic flow that has been flagged as abnormal thanks to a signature or due to its statistical behaviour [SBW07c]. Two kinds of IPS exist: Host Intrusion Prevention Systems (HIPS) and Network Intrusion Prevention Systems (NIPS) that monitor and block respectively the process, files and network flows on a host, or the packets on the network.

More generally, an Intrusion Response System (IRS) is a system that provides other types of responses to a detection. This is a currently active research topic and many recent papers treat this subject, as summarised by Shameli-Sendi *et al.* in [SSEJJ12].

Using taxonomies of Carver [Car00], Stakhanova *et al.* [SBW07b] and Tanachaiwiwat *et al.* [THC02], here is a list of possible active responses to an intrusion:

- *System-based responses*

  Deactivate ports or services: If a compromised network service is the relay of an attack, stopping it or blocking its port may stop the attack.

  Delete a process of a user: If an illegal process is launched on a host, killing the process will prevent its execution.

  Limit the capabilities of users: When a user is suspicious, it is possible to forbid him the use of commands that are potentially damageable to the system.

  Freeze the account of a user: If a user account may have been compromised, an appropriate answer is to block the user account to make sure that he may not be the origin of another attack.

  Force an additional authentication factor: On infected machines, forcing the users to use an additional authentication factor is a good was to stop an attack, without interrupting services.

  Change passwords: When a password has been compromised, changing it may prevent the attacker to get into the information system.

  Shutdown an infected machine: Sometimes, it is the only means to stop an attack. However, this response is often not the best to have for several reasons. The first one is that stopping a machine could stop very not-compromised and important services. Moreover, shutting down the machine delete the memory and loose very important information for post-mortem analysis. Finally, when rebooting the exploited vulnerability will be still there and the attacker could start again the same attack.

Create backups: Even if it is often too late when the attack has begun, doing backup may be useful to know what has been modified by the attacker and to restore the files changed by the attacker

Reinstall the OS: To restart on a clean configuration, it is possible after an infection to reinstall an operating system automatically. This is obviously easier with virtual machines when it is possible to launch a clean template where software is already installed and configured.

Use *Temporary Shadow Files*: These are copies of original files which are encrypted. During the attack, system files which are modified could then be restored. It is a means to assure the integrity of the attacked system.

Give access to false files: By using technologies like *honeypot*, it is possible to trap an attacker by giving him false data.

Warn the attacker: Sometimes, the only fact on warning the attacker that he has been detected will stop him.

- *Network-based responses*:

  Terminate the TCP connexion: To immediately stop an attacker connexion, it is possible to send a TCP packet with RST flag to the attacker and/or victim.

  Block an IP address: If the IP address of the attacker can be identified, blocking this address may be enough to stop the attack.

  Trace back the attacker connection: If possible, it is useful to know who and where the attacker is, in order to prevent him from starting again an attack.

So, many different responses are possible to stop and attack or prevent its expansion. Those responses are either system-based or network-based and may more or less disrupt legitimate services. First an intrusion response system has to compute the responses that can prevent an attack. Then, in the set of those candidate responses it has to select the best response(s) to stop the attack.

3.5 Computing and ranking responses using attack models

The selection of a response by an IPS or IRS is generally static (the same response is always assigned to a detection) but is sometimes adaptive (the response can change according to the context or experience). When the selection of the response is static, it has been manually set by a security operator. We will thus discuss in this section the possible means used by adaptive IRS to compute responses and to rank them, in order to select the most appropriate one.

3.5.1 Computing responses using attack models

Several papers of the state-of-the-art propose techniques to compute or select responses using attack models. Those techniques rely on an attack model to configure the detection mechanisms for passive responses or to select and prioritise corrective remediations or active responses.

3.5.1.1 Configuration of detection for passive responses

Attack graphs are used by Noel and Jajodia [NJ08] to compute the optimal locations to deploy IDSs in an information system: they allow to minimise the cost of sensors while keeping a complete coverage of potential attack paths. First, they use TVA attack graphs to predict all the possible ways to reach critical assets in a network. Then, they aggregate the graph according to the network

accesses, to decrease the complexity. Finally, they find the minimum number of positions of IDS to cover the attack graph with an efficient greedy algorithm.

In [GTHM14], Godefroy *et al.* present how they compute correlation rules automatically from extended attack trees, to detect complex attack scenarios. In the transformation process from an attack tree to the corresponding correlation rule, the only manual process is the first step, the identification of the elementary attack actions that can be attached to the leaves of the attack tree. This technique also relies on a knowledge base containing the system cartography, the available sensors and their configuration *etc.*. Thus, it separates the expert knowledge needed to create the correlation rules in three low-coupled domains: attack scenario specification, system cartography and sensors modelling.

3.5.1.2 Selection of corrective remediations or active responses

In [NJOJ03], Noel *et al.* present a methodology to compute hardening options from an exploit dependency graph, a type of attack graph. They start from the attack goals and do a backward traversal of the graph to compute the minimal sets of necessary initial conditions. Then, they minimise hardening costs of responses by choosing the minimal hardening measures.

In [MBFB06], McQueen *et al.* use a compromise graph to measure the risk reduction of a security measure. They generate a compromise graph on the currently deployed system and on a enhanced system, the same system with a security measure deployed. Once the compromise graphs have been generated, they estimate in each graph the dominant attack path, the path with lowest time to compromise. Then, they compare the dominant attack path in the current system with the one on the enhanced system to measure the improvement of the enhanced system.

In [CAB+06], Cuppens *et al.* describe how the LAMBDA language (LAnguage to Model a dataBase for Detection of Attacks), can be used to model attacks. An attack in LAMBDA is composed by (1) a precondition, describing the state of the system required to do the attack, (2) a post-condition, describing the state of the system after the attack, (3) a detection, describing the expected alerts when the attack occurs, and (4) a verification, describing a condition to check that the attack actually occurred. The LAMBDA language can also model countermeasures. A countermeasure in LAMBDA is composed by (1) a precondition, describing the state of the system required for the countermeasure, (2) a post-condition, describing the state of the system after applying the countermeasure, (3) an action, describing the actions to apply the counter-measure, and (4) a verification, describing a condition to check that the counter-measure has actually been applied. Then, using the concept of anti-correlation, the LAMBDA language allows to compute the appropriate countermeasures for an attack.

Kijsanayothin and Hewett present in [KH10] a methodology based on a static analysis of attack graphs using logical expressions in order to select cost-effective countermeasures. This methodology relies on a conditional preference network, a graphical model representing preference relationships among decision variables. This network allows to know among two decisions the one that best fit the preferences. Kijsanayothin and Hewett use it to choose the most preferred countermeasure among the set of possible countermeasures computed by logical expressions

In [WNJ06], Wang *et al.* base their analysis on the initial conditions of a logical attack graph to compute all hardening options for a network. The initial conditions represent the logical conditions that are not a consequence of another exploit in the attack graph. They can thus be disabled by responses independently of the other exploits. In order to compute possible responses, they start from the goal condition, describing the state they want to protect, and travel backward along the graph, until finding sufficient and necessarily set of initial conditions to remediate. This approach has been improved by Albanese *et al.* in [AJN12] with a near-optimal algorithm more efficient and cost-sensitive. This algorithm is a near-optimal approximation algorithm, which thus scales

almost linearly with the size of the attack graph. They remove the usual assumption that hardening actions are independent. In addition to the attack graph, they also add a cost model which takes into account the impact of interdependent actions.

Dewri *et al.* present in [DPRW07] a methodology to compute the optimal security hardening responses from an attack tree, using a multi-objective optimisation algorithm. It aims at addressing the administrators' dilemma: how to select the appropriate hardening measures with a minimum cost and being under a fixed budget constraint.

In [NJ09], Noel and Jajodia describe several methods to prioritise the deployment of patches depending on a TVA topological attack graph. They present three different kinds of recommendations to harden a network: (1) at the attack source, to prevent other attacks from this source, (2) at the attack goal, to protect a specific attack goal, (3) a minimum-cost hardening, to deploy the minimum of patches in the network while protecting an attack goal from an attack source. They also use TVA attack graphs to correlate intrusion alarms based on attack causality.

In [HLZ10] Henry *et al.* use Petri nets to identify high-value risk mitigation opportunities to use an informed search over the coverability set. The coverability set gathers the places and transitions representing the industrial process failure and attacker actions that cause these failures. First, they sort the process failure modes of the coverability set by decreasing severity. Then, they identify the first-order, second-order, third-order and fourth-order transitions, *i.e.* the transitions of coverability set allowing to enter a process failure in respectively one, two, three or four steps. Finally, they identify the risk mitigation opportunities as the host resources or global resources control failure models that are preconditions of first-, second-, third or fourth order transitions.

Wang *et al.* uses in [WZK13] a Hidden Markov Model to infer the optimal security hardening measures. In order to do that, they apply a heuristic algorithm on the HMM to compute the optimal measures.

In [MRT15], Miehling *et al.* use a Bayesian attack graph to compute optimal defences policies to protect a critical subset of resources of a computer network. They assume that the input Bayesian attack graph has already been computed (by the techniques of Frigault and Wang [FW08] or Xie *et al.* [XLO+10]). The countermeasures are modelled by a set of binary actions that can prevent attacker actions Then, they use a partially observable Markov decision process to formulate and resolve the defender challenge that can partially observe the attacker's actions.

3.5.1.3 Conclusion

We have seen in this subsection that we can use the attack models to support the computation of responses. First, attack models allow to optimise the configuration and deployment of passive responses, for example by using attack graphs to find the optimal locations to deploy IDSs. Or we can also use these models (*e.g.*, attack trees) to compute the correlation rules that will be used to configure sensors to detect more complex attack scenarios. We can also use attack models to compute corrective remediations or active responses. For example, the attack graphs can support the computation of the minimal set of responses to deploy in order to protect attack goals, to compute all hardening options for a network, or even to prioritise the deployment of patches in a system. We can also used Petri nets, Markov Model and Bayesian attack graphs to identify the best possible risk mitigations to protect a subset of critical resources.

3.5.2 Topological and functional models of an information system

Many response selection methodologies require a topological or functional model of the information system, in order to select the right response to deploy and assess its impact. Recent developments

show that topological models of an information system can be created automatically from network scans.

In [MMDD09], Morin *et al.* present M4D4, a data model formalising the topology and security systems of a network. It stores contextual information (*e.g.*, topology and cartography), attack and vulnerabilities, analysers (*e.g.*, detection probes, firewall, vulnerability scanners) and events and alerts. It can also interact with external relational databases to retrieve other types of facts. Then, it can be queried with simple queries to retrieve topological information or more advanced ones to perform alert correlation.

In [JN10], Jajodia and Noel detail how to build a network model of the information system (all authorised connections throughout the network) from vulnerability scans targeting the entire network, performed from each of the subnets. It can also be created by importing the configuration of network devices as it is done in commercial solutions such as in Skybox Security [3]. The model should be as accurate as possible to be exploitable. Nevertheless, its completeness is not guaranteed by the currently available technology.

The functional model of an information system is complementary to the topological one. It contains the dependencies between components of the information system and can be used to improve the automation of the deployment of remediations. In [TK02], Toth and Kruegel present a dependency model allowing to determine the impact of remediations on the whole system. In [KCBCD10], Kheir *et al.* present a dependency graph modelling dependencies between system resources. The graph is built manually or preferably automatically and contains all system resources (networks, machines, services, user groups) and their dependencies. In this graph, each resource has a level of confidentiality (C), integrity (I) and availability (A). Then, changes (either attack or response effects) are simulated by spreading C-I-A values into the graph, according to dependencies, to see their impact on system resources.

Functional models also help the impact analysis of cyber-defence missions. In this domain, missions are decomposed in tasks, relying on information system assets. In [SSL15], Sun *et al.* build a System Object Dependency Graph (SODG) and a Mission-Task-Asset (MTA) map, associating the missions and tasks with their related assets (system objects, processes, files, *etc.*). Both models are used to build a Bayesian network inferring the probabilities of tasks and missions of being impacted by intrusions collected by evidences (logs, security sensors, *etc.*). Thanks to that, they can evaluate the impact of attacks on tasks and missions.

3.6 Response selection criteria

According to Carver [Car00] and Mu *et al.* [MSL10], the selection of an appropriate response to respond to an intrusion detection can take into account several criteria.

- *Criteria related to the attack*:

  Type of attack: The response depends on the type of attack that the defender has to deal with (protocol and applications used, vulnerability exploited, *etc.*).

  Timing: According to the point in which the attack is (before an attack, during an attack or after an attack), some responses will be more appropriate.

  Type of attacker: If the type of attacker can be known, the response has to be chosen according to the type of attacker and his skills. For example, the response to a bot script will be different to the response to a targeted attack by a high-level attacker.

  Attack severity: According to the importance of the damages that can be caused by an

[3] Skybox security, inc., `http://www.skyboxsecurity.com/`

attack, the response will be different. For example, the same attack on machines that are not as much critical can have different responses.

Goals of the attack: Even if it is often very difficult to know it a priori, it is much easier to protect an information system when only few machines need to be protected.

Confidence in the alert: Sensors are not entirely reliable and can have false positives. It is thus good to take into account this uncertainty in the received alerts and take the responses accordingly.

Number of alerts: The number of alerts received in a short period of time or that are part of the same attack scenario is often an indicator of the power of the attack, the confidence in the alerts, or may allow to identify the type of attacker.

- *Criteria related to the response*:

  Effectiveness of the response: A response will be chosen to stop an attack. Its ability to effectively block the attack has to be considered in the selection process.

  Probability of failure of the response: Strongly related to the effectiveness of the response, the probability that the response fails and does not stop the attacker has to be taken into account to compare responses.

  Intensity of the response: A stronger answer is more likely to stop an attacker, but it might also have bigger side effects.

  Negative impacts of the response: All responses have negative impacts on the information system. Those impact must be taken into account, because they are sometimes more harmful than the attack itself.

  Cost of the response: The operational costs of deploying a response or the duration to implement it can be strong limitations to select a response.

  Durability of the response: This criterion is less important for a short-term response, but it can be balanced with the operational costs, as a long-term response, event more expensive to deploy can be a better choice.

- *Criteria related to the target*:

  Vulnerability exposure: If the asset targeted by an attack is not vulnerable to the used vulnerability, it is irrelevant to deploy a response.

  Confidentiality, Integrity and Availability constraints: If the target of the attack or of the host has strong constraints of Confidentiality, Integrity or Availability, the response selection has to take into account those constraints.

  Environmental constraints: There may also be many constraints external to the information system itself, which may need to be taken into account to select an appropriate response to an attack (*e.g.* legal or ethical constraints).

3.7 Response selection methodologies

Several methodologies have been proposed in the state of the art to select the responses to deploy.

The Analytic Hierarchy Process (AHP) has been used by Wu *et al.* in [WXX+08] to select the responses to deploy. It is a structured technique to take complex decisions to take into account several criteria. The 4 criteria taken into account to select the responses are: (1) the effectiveness of the response, (2) the influence of the response on other services, (3) the time needed to deploy

the response, and (4) the resources necessary to deploy it. The main advantage of the AHP is that the administrator only has to compare two responses on each criterion to quantify the importance of each criterion for a response selection, rather than setting arbitrary values. Then, the priorities to give to the criteria are deducted from the eigenvector of the matrix containing these values. In [KLI08], Kim *et al.* also propose a method based on the AHP to find a higher ROSI (see section 3.8.4 for more details about ROSI).

It is also possible to select a remediation using a dependency graph implementing three main steps: (1) Build the model. This step consists to put in the graph all system resources (networks, machines, services, user groups) and their dependencies. This can be built manually or preferably automatically. (2) Update the model with current system state. Information is gathered from monitoring and surveillance systems to specify the confidentiality, integrity or availability value of the monitored resources. Then we use the dependency graph to propagate these values to all resources of the model. (3) Compare response to an attack. The last step begins with the simulation in the dependency graph of the effects of each response (after simulating or detecting an attack). Then, thanks to a comparison operator of a model state, we can compare possible responses and know which is the better to deploy. In [KDCB+09a, KCBCD10], Kheir *et al.* propose a framework modelling dependencies which handles both confidentiality, integrity and availability. This model represents a network with a graph where nodes are resources. They are linked together with arcs modelling dependencies relations between these resources. Each node can have values representing the level of confidentiality (C), integrity (I) and availability (A) of the resource. These values are computed by taking into account the intrinsic state of the resource then by propagating the values of the other nodes, according to the dependencies, by using an algorithm described in the article. The possible responses (eventually "do nothing") are finally applied on the current state of the system and are compared using a function that matches a value representing the quality of user experience (QoE) to the current state.

Another approach to choose a response based on game theory is presented by Zhu *et al.* in [ZLYS10]. The attacker and defender are modelled as two players with opposite objectives interacting with the same system. This model allows to take into account both the damage of the attacker with the collateral damages of the responses of the defender and can also consider risk factors such as sensors errors. The approach of the Response and Recovery Engine (RRE) introduced by Zonouz *et al.* in [ZKSY09, AZ12] is also based on game theory. Using a attack-response tree, RRE takes into account the uncertainty of the IDS alerts to estimate the system security level. It can also predict the attacker next step, in order to propose the most appropriate response.

The MITRE proposes in its Systems Engineering Guide [MIT14] the Cyber Risk Remediation Analysis (RRA), a method to choose among responses, in order to reduce the compromise probability of an asset. It integrates in the global process of *Mission Assurance Engineering* whose goals are: (1) to define the most important assets of an information system, with a *Crown Jewels Analysis* (CJA), (2) to find their risks with a *Cyber Threat Susceptibility Assessment* (TSA), (3) and decrease these risks with the RRA. The RRA's goal is to establish a list of responses mitigating the attacks, with minimal costs. The starting point of RRA is the output of TSA, a Threat Susceptibility Matrix ranking potential attacks on an asset on a risk scale from 1 to 5. The first step of the RRA is to identify in the matrix which attacks to mitigate on which assets. Then, for each asset, a table mapping each attack with possible responses associated with their effects and effectiveness on the attack and their cost is used to choose the responses allowing to decrease all selected attacks, with a minimal cost.

3.8 Metrics to balance damage and response cost

Besides the methods used to select a response, many metrics can be used to compare responses with each other. We will now list the main metrics that can be used to compare responses.

3.8.1 Intrusion response cost assessment

[SSBW09] describes a method to compute a response cost. Three criteria are used:

the operational cost of the response (OC): The cost related to deploy and maintain the response

the response goodness (RG): Represents the prevention or attenuation of damage that can do the response

the response impact on the system (RSI): Quantifies the negative effect that response may have on the system

So, the cost of the response is $RC = OC + RSI - RG$. Seven steps are necessary to estimate this total cost. The method presented in [SBW07a] is similar, a bit less accurate, but has fewer steps.

3.8.2 Return on investment

The Return of Investment (ROI) is a simple index that represents the efficiency of an investment. Generally, to calculate the ROI, the benefit of an investment is divided by the cost of the investment. A ROI for attack mitigation is often defined by the following formula [CM05]:

$$\frac{ALE_{beforeS} - ALE_{afterS}}{\text{cost of security measure}} \tag{3.1}$$

where ALE is the Annual Loss Expectancy, the expected impact cost due to the attack.

3.8.3 Return on attack

In [CM05], Cremonini *et al.* present a new index to improve the simple return on investment: the Return-on-Attack (ROA) index. The goal of the ROA index is to reflect the average and supposed impact of a security solution on attackers' behaviours. The ROA is defined as

$$\frac{\text{Gain from successful attack}}{\text{cost before measure} - \text{loss caused by measure}} \tag{3.2}$$

The attacker wants to maximise this index, whereas the defender wants to minimise it. The ROA adds to the ROI the representation of the impacts that solutions have on attackers' behaviours.

3.8.4 Return on security investment

The Return of Investment for a security investment (ROSI) is defined in [SAS06] as

$$\frac{(\text{Risk Exposure} \times \text{RM}) - \text{Cost of measure}}{\text{Cost of measure}} \tag{3.3}$$

where Risk Exposure = $ALE = SLE \times ARO$ (ALE stands for the Annual Loss Exposure, SLE for Single Loss Exposure and ARO for the Annual Rate of Occurrence). The ALE has already been defined in 3.8.2. The SLE represents the estimated cost due to one exploitation of a security flaw during one year. The ARO represents the estimated number of times this security flaw will be exploited. RM stands for Risk Mitigation, it is a level ($0 \leqslant \text{RM} \leqslant 1$) that represents the effectiveness of the measure to protect against the exposure.

3.8.5 Return on response investment

RORI (Return-On-Response-Investment) is an index defined in [KCBCD10, Khe10] as:

$$RORI = \frac{RG - (CD + OC)}{CD + OC} \tag{3.4}$$

where RG is the Response Goodness, it measures the ability of the response to reduce the cost of the attack. CD is the response collateral damages, it represents the cost added by the response to the other services and OC is the response operational costs, the costs to deploy the response.

This index is one of the first to consider not only the response collateral damages (modelled by the CD) but also the response effects on intrusion (modelled by RG).

An improved RORI has been proposed by Granadillo *et al.* in [GDJ+12]. This index handles the choice of applying no countermeasure to compare with the results when applying a security measure. This index is also relative to the size of the infrastructure, it is thus easier comparable in different environments. Here is the formula to compute the improved RORI :

$$RORI = \frac{(ALE \times RM) - ARC}{ARC + AIV} \times 100 \tag{3.5}$$

where ALE the Annual Loss of Exposure and RM the Risk Mitigation are defined as in 3.8.4, ARC, the Annual Response Cost represents OC+CD in standard RORI, the cost incurred by implementing the countermeasure. AIV, the Annual Infrastructure Value is the global cost expected to have on the system in a year.

This improved RORI can be extended, as described in the same article, to apply to combined countermeasure selection for a single attack or multiple attacks.

3.9 Analysis and conclusion

The state of the art of computation of responses to protect a system is very large. Responses aim at handling risk. There are three types of responses. The remediations are corrective responses that correct the vulnerability in order to prevent its further exploitation. They are generally implemented by patch management software and this technology is mature, but suffers from several limitations that cause that patch deployment still requires human intervention and cannot be fully automated. As a result, security administrators generally have to prioritise the deployment of patches. When corrective remediations are not possible, the second type of responses that is generally used by administrators is active response. An active response groups the actions preventing the exploitation of a vulnerability, but without correcting it. This is generally implemented by filtering, at the network layer, the packets bearing signs of an attempted or successful vulnerability exploitation (*e.g.*, by an Intrusion Prevention System). But it can also be many other types of actions that are grouped under the name Intrusion Response Systems. Finally, the passive remediations group the detection of the exploitation and its report. This is a last resort but is widely

used, as it can assist security operators in getting insights on what happened in their information system. A system that only provides passive responses (*e.g.*, alerts, reports) is generally called Intrusion Detection System.

In order to select the responses to deploy for risk handling, it is possible to rely on attack models. An attack model can support the configuration or deployment of passive responses (*e.g.*, using an attack graph to find the optimal locations to deploy IDSs or to compute the correlation rules used to configure sensors to detect complex attack scenarios). We can also use an attack model to compute corrective remediations or active responses. For example, attack graphs can support the computation of all hardening options for a network, or even to prioritise the deployment of patches in a system. We can also use Petri nets, Markov Models and Bayesian attack graphs to identify the best possible risk mitigations to protect a subset of critical resources. On the contrary, topological and functional models are generally used to analyse the impacts of the responses to deploy. There are several either topological or functional models that have been proposed in the state of the art and their usage is generally complementary to attack models. Rather than describing all the attacks that are possible in a network, they describe the legitimate services or missions of the system and describe how the services depends on each other.

Finally, to select the responses to deploy, several criteria can be taken into account, either related to the attack, either related to the response, or related to the target of the attack. Then, methodologies to select the responses generally rely on metrics allowing to balance damage and responses cost. These metrics generally take into account both the positive (*i.e.*, against the potential attack, for example with the help of an attack model) and negative effects of the responses (*i.e.*, its negative impact on the system, for example with the help of a functional model), in addition to the cost of the response.

To conclude, several criteria can be taken into account when computing/selecting a response and this decision can rely on the use of several models, either attack or topological models. As a result, in order to select the attacks to respond to, the first step is to conduct a proper risk assessment. Either a static risk assessment for the computation of remediations for potential attacks, either a dynamic risk assessment to compute active responses for occurring attacks. However, these risk assessment methods lack a comprehensive methodology to compute remediations that could apply to any type of attack, with any kind of remediations that are proposed by the security operators.

Chapter 4

Remediating the logical attack paths of an attack graph

Chapter 3 showed that one main usage of risk assessment models is to support the computation of responses in order to prevent or mitigate the risk of attacks. It also shows that the risk assessment methods lack a comprehensive methodology to compute remediations that could apply to any type of attack, with any kind of remediations.

In this chapter, we present the use of an existing static attack model, the logical attack graphs, to compute remediations to potential attacks. This is the first necessary step to understand how a multi-step attack model is valuable to compute responses. It will also allow to identify the elements required in a dynamic risk assessment model to support the computation of responses in operational time. Operational time in our context means the time needed for a human to properly understand a situation and take a decision (*i.e.*, a few minutes).

We design in this chapter a generic remediation process correcting the most relevant attack paths extracted from a logical attack graph. The method to remediate these paths consists in (1) selecting the set of preconditions necessary to reach the attack path target, (2) building the set of remediation candidates to correct the set of necessary preconditions, if possible, and (3) rank the remediation candidates according to a cost function considering both operational and impact costs. It relies on a generic remediation database matching vulnerabilities with their remediations. This process corrects only the most relevant attack paths rather than the whole attack graph which is generally too complex to analyse.

This chapter is part of the contribution described in subsection 1.3.3, the use of risk assessment models to support the computation of responses, and tackles the challenge described in subsection 1.2.4: the usability of attack models.

4.1 Attack paths and preconditions

The main model on which we build this remediation process is a logical attack graph from which we extract attack paths. First, we introduce the formal representation of these models.

4.1.1 Attack path representation

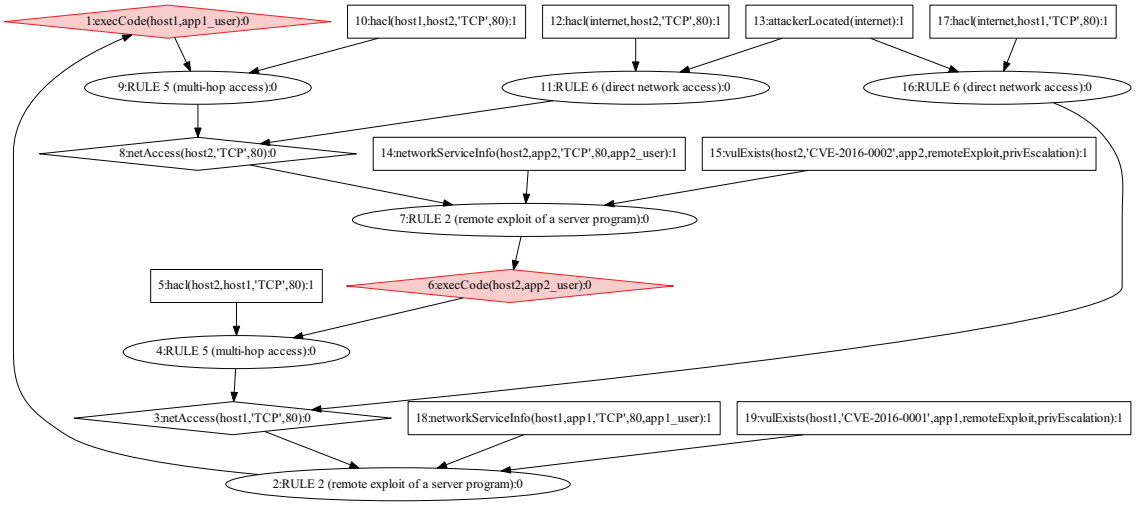
Definition 10 *A **logical attack graph** G is a directed `AND-OR` graph represented by $G(V, A)$ where:*

- *V is a set of vertices that describe logical facts. Each vertex could be an `AND` (respectively an `OR`) vertex, meaning that this vertex needs the* conjunction *(resp. the* disjunction*) of its incoming arcs to be true.*
- *A is a set of directed arcs that represent a logical dependency from the child vertex to the parent one.*

Definition 11 *A **precondition** in a logical attack graph is a vertex whose indegree is 0,* $\deg^-(v) = 0$ *(no incoming arcs).*

Figure 4.1 shows an example of a simple logical attack graph generated with the attack graph engine MulVAL [OGA05a, OBM06]. Ellipse vertices represent `AND` and diamond vertices represent `OR`. The rectangle vertices are the preconditions. This graph shows how an attacker on the Internet can exploit the vulnerability *CVE-2016-0001* on the application *app1* on host *host1* to get the privilege *app1_user* on this machine. From the Internet, or from the already exploited *host1*, the attacker can exploit the vulnerability *CVE-2016-0002* on the application *app2* on host *host2* to get the privilege *app2_user* on this machine.

FIGURE 4.1 – Example of logical attack graph



In an attack graph as defined above, security operators can choose attack targets. These are the vertices describing important final steps for an attacker. For example, in Figure 4.1, the targets are the nodes in red. They represent the acquisition of a new privilege on a host, represented by the *execCode* primitive.

Based on these targets we build attack paths using a bottom-up approach from the target to the upper preconditions of the attack graph. They can be ranked according to their impact and probability of occurrence, but we will not enter into much detail as this subject has been already discussed many times in the state of the art. For example, in [MBZ+06], Mehta *et al.* propose two algorithms to rank the states of an attack graph, in order to concentrate only on relevant subgraphs. In [SO08], Sawilla and Ou generalise Google's Page Rank algorithm as AssetRank and

apply it to MulVAL's attack graph to extract from the attack graph the most vertices. Birkholz *et al.* present in [BEJS10] how to extract from a vulnerability graph (a kind of attack graph) what they call attack trees (which are equivalent to our definition of attack path here). In order to do that, they apply a modified version of the Dijkstra algorithm to detect the highly vulnerable attack paths. So, from a logical attack graph, we can extract ranked attack paths, as defined below.

Definition 12 *An **attack path** is an acyclic and* logically valid *subgraph of an attack graph with one* target *and several* preconditions.

Definition 13 *The **target** of an attack path is the vertex whose outdegree is 0,* $\deg^+(v) = 0$ *(no outgoing arcs).*

Definition 14 *A subgraph S of an attack graph G is **logically valid** if S contains at least one vertex and for each vertex $v \in S$, $v \in G$ and*
if $\deg^-_G(v) > 0$ *(more than one incoming arc in G):*

- *if v is an* `AND`*, all the parents and incoming arcs of v in G are in S,*
- *if v is an* `OR`*, at least one parent of v and its incoming arc in G is in S.*

An attack path may have several intermediate goals but has only one main goal: the target of the attack path. It contains one, several or all the possible paths in the attack graph allowing to compromise this target.

Figure 4.2 shows two attack paths extracted from the two targets of the logical attack graph of Figure 4.1. The attack path of Figure 4.2a is built from the target node $execCode(host1, app1\_user)$ and contains 8 nodes of the original graph. The attack path of Figure 4.2b is built from the target node $execCode(host2, app2\_user)$ and contains 17 nodes of the original graph. The attack path of Figure 4.2a is included in the one of Figure 4.2b. Indeed, the target node of the first one is an intermediate target of the second one.

4.1.2 Remediations and sufficient preconditions

Proposing remediations to an attack path is searching the means to prevent the attacks and protect its target. An attack path is a logical graph: a fact is true if and only if the conjunction or disjunction (according to the type of node) of its parents is also true. As a precondition does not have any parent, it is not deducted from any other vertex. As a result, we build our remediation using this axiom:

Axiom 1 *Preconditions are the first conditions from which all other vertices are deducted and thus the only vertices for which remediations can be applied.*

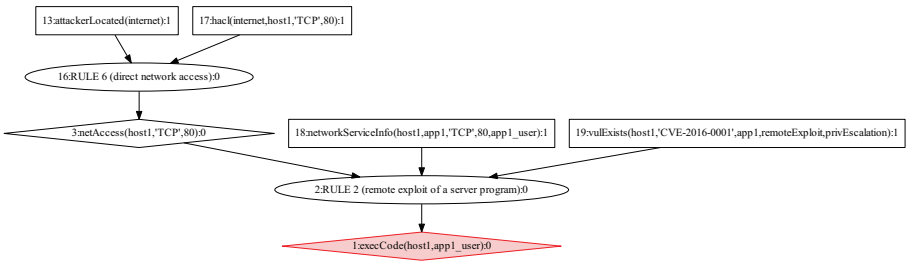
The attack path contains one target that remediations has to protect. As the attack path is an `AND-OR` graph, it is possible to compute all conjunctions of to-be-remediated preconditions, sufficient to protect the target. This logical expression SP can be represented with a set of disjunctions containing conjunctions as following:

$$SP = \bigvee_i SP_i = \bigvee_i \bigwedge_j SP_{i,j} \tag{4.1}$$

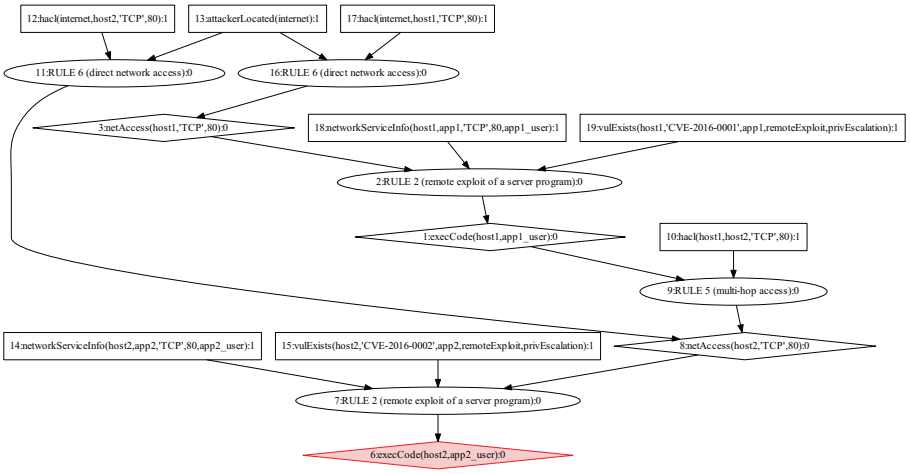
where $\bigvee$ is logical `OR`, $\bigwedge$ is logical `AND`, SP_i is a conjunction of preconditions sufficient to protect the target (i indexing the conjunction of preconditions) and $SP_{i,j}$ is a precondition to remediate (j indexing the preconditions).

FIGURE 4.2 – Examples of attack paths

(A) Attack path 1



(B) Attack path 2



In fact, computing SP is identical to find all the conjunctions of preconditions sufficient to delete the target vertex according to the `AND`/`OR` formalism.

Definition 15 *A conjunction of precondition SP_i is **sufficient** to delete the target t of an attack path AP if the deletion of each precondition $SP_{i,j} \in SP_i$ and its propagation in AP imply that henceforth $t \notin AP$.*

Figure 4.3 shows the recursive algorithm computing SP. We call this algorithm on the target of the attack path and it follows recursively along the arcs. All conjunctions of preconditions computed with this algorithm allow to prevent the access to the target, if remediated. Of course all preconditions cannot be remediated. If this is the case in a conjunction, the entire set of preconditions will not be usable to successfully protect the target of the attack path.

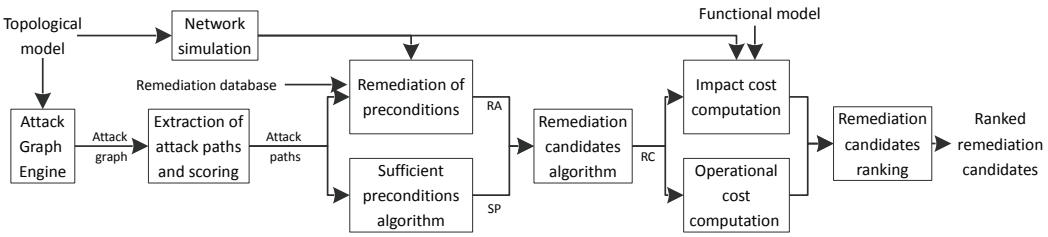
4.2 Remediation of an attack path

Figure 4.4 shows a diagram summarising the remediation process. The remediation process starts with an attack graph that generates the logical attack graph from a topological model of an information system. Then, attack paths are extracted from the attack graph and scored. For each scored attack path, we compute remediations by, (1) remediating the preconditions individually (potentially by using a remediation database and a network simulation tool), and (2) identifying

FIGURE 4.3 – Recursive algorithm computing the conjunctions of sufficient preconditions

```
1: function computeSP(Vertex v)                          ▷ Returns the list SP to delete the vertex v
2:     if v is a precondition then                       ▷ v has no parent
3:         return [[v]]                                  ▷ it is the only precondition
4:     else if v is an AND then
5:         res ←[[]]
6:         for each parent p of v do
7:             res ← res;computeSP(p)
8:         end for
9:         return res                                    ▷ ⋁_{p∈{parents of v}} computeSP(p)
10:    else if v is an OR then
11:        res ←computeSP(first parent of v)
12:        for each other parents p of v do
13:            res ←conjunctionOfSets(res,computeSP(p))
14:        end for
15:        return res                                    ▷ ⋀_{p∈{parents of v}} computeSP(p)
16:    end if
17: end function
18: function conjunctionOfSets(A, B)                     ▷ Makes the conjunction of sets A and B
19:    result ← [[]]                                     ▷ A and B are Or/And sets: A = ⋁_i A_i, A_i = ⋀_k A_{i,k}
20:    for i = 1 to size(B) do
21:        buildingResult ← A
22:        for j = 1 to size(buildingResult) do
23:            buildingResult[j] ← buildingResult[j]; B[i]
24:        end for
25:        result ← result; buildingResult
26:    end for
27:    return result                                     ▷ (⋁_i A_i) ∧ (⋁_j B_j) = ⋁_{i,j} (A_i ∧ B_j)
28: end function
```

the sets of preconditions that are sufficient to protect the attack path target. From both these results, the remediation candidates algorithm build sets of remediation candidates. The ranking of these remediation candidates is done by two cost functions: (1) the impact cost function that assess the cost of the impact of this remediation on the system, according to a functional model and



4.2.1 Remediate a precondition

A precondition contains a logical fact describing what can be used by an attacker. We detail real preconditions and their remediations in subsubsection 4.4.2.3, but will first describe two general methodologies that can be applied to preconditions.

4.2.1.1 Simple remediations to preconditions

The first case appearing when remediating a precondition is a simple remediation that can be applied to negate this precondition. This usually corresponds to the corrective responses presented in the state of the art. It requires a database that makes the link between the precondition (*e.g.*, the vulnerability) and how we can remediate it (*e.g.*, a patch).

4.2.1.2 Remediations using the network topology

Some remediations require more advanced techniques. This is the case of those which try to prevent the *exploitation* of a vulnerability. It corresponds to the active responses of the state of the art. Computing such remediations requires an accurate knowledge of the flows exchanged on the network and thus require to simulate the network topology.

4.2.2 Remediation candidates for an attack path

Each remediation potentially contains several elementary actions. More formally, for each attack path, we have succeeded in computing:

- A disjunction of conjunctions of **sufficient preconditions** to remediate, in order to protect the target of the attack path: SP
- A disjunction of conjunctions of **remediation actions** sufficient to prevent a precondition p:

$$RA(p) = \bigvee_i RA_i(p) = \bigvee_i \bigwedge_j RA_{i,j}(p) \tag{4.2}$$

  where $RA_i(p)$ is a conjunction of remediation actions allowing to prevent the precondition p (i indexing each conjunction) and $RA_{i,j}(p)$ is a remediation action (j indexing each action of the conjunction) each remediation action $RA_{i,j}$ is constituted of the tuple (action to apply, machine to deploy it).

We need to combine SP and RA to compute a disjunction of **remediation candidates** containing the actions that allow to protect the target:

$$RC = \bigvee_i RC_i = \bigvee_i \bigwedge_j RC_{i,j} \tag{4.3}$$

where RC_i is a remediation candidate (i indexing the number of candidates) and $RC_{i,j}$ is a remediation action (j indexing the number of actions in the candidate).

Figure 4.5 shows an algorithm computing such remediation candidates.

FIGURE 4.5 – Algorithm computing the remediation candidates

```
1: function computeCandidates(SP, RA) ▷ Returns all remediation candidates protecting an attack path.
2:     res ←[[]]
3:     for SP_i in SP do
4:         res ← res ; computeRemedsToPreconds(SP_i, 1, RA)
5:     end for
6:     return res
7: end function
8: function computeRemedsToPreconds(SP_i, j, RA) ▷ Returns all conjunctions of actions allowing to remediate the preconditions of SP_i starting from j.
9:     SP_{i,j} ← SP_i[j] ▷ j^th precondition to remediate
10:    RA_j ← RA[SP_{i,j}] ▷ Remediations of j^th precond
11:    if empty(RA_j) then ▷ j^th precondition cannot be remediated
12:        return [[]]
13:    else
14:        if j=size(SP_i) then ▷ Terminaison of recursion
15:            return RA_j
16:        else
17:            RA_{j+1..n} ←computeRemedsToPreconds(SP_i, j + 1, RA)
18:            return conjunctionOfSets(RA_j, RA_{j+1..n})
19:        end if
20:    end if
21: end function
```

4.3 Costs of remediations

The last essential point for our remediation method is to estimate the cost of a candidate. This helps an operator to choose between several candidates remediating the same attack path. We have identified two principal sources of cost that must be considered: the operational and the impact costs.

4.3.1 Operational cost

The first important cost for an operator deploying a remediation is the operational cost (OC). It represents the difficulty to implement the remediation on the assets and to maintain it. For each remediation action $RC_{i,j}$ that should be applied, we identified four categories in which this cost can be split.

1. **Remediation cost (RC)**: This is the cost of the input required to apply the remediation, for example, the price of a patch or of a signature.
2. **Deployment costs (DC)**: This is the cost representing the workload to apply the remediation on the concerned machine.
3. **Test costs (TC)**: This is the cost to test that all important features of the information system are still working as expected, after the deployment.
4. **Maintenance costs (MC)**: This is the cost per year of the maintenance induced by the remediation. It reflects for example the increase of CPU, memory, storage and treatment of logs that will be induced.

These elements can be expressed in a monetary unit and we detail in Subsection 4.4.2.5 how to compute them. The operational cost of a remediation action is the sum of all these elements as shown in Equation 4.4.

$$OC(RC_{i,j}) = RC + DC + TC + MC \tag{4.4}$$

To simplify the estimation of the operational costs of a candidate containing several remediation actions, we made the assumption that these actions are independent. This assumption has been introduced and justified by Gonzalez-Granadillo *et al.* in [GJDC12] as Axiom 1. This is, moreover, the most common case, as the remediation actions are usually deployed on different machines or are of different types. With this assumption, the operational cost of a candidate is the sum of the operational costs of its actions, as shown in Equation 4.5.

$$OC(RC_i) = \sum_j OC(RC_{i,j}) \tag{4.5}$$

4.3.2 Impact cost

As this subject has been frequently treated in the state of the art (for example, see in subsection 3.5.2), and we prefer to focus in this work on the methodology which is independent of the system studied, we propose here a basic impact cost (IC) function that measures the loss due to the unavailability of services after the deployment of a remediation. This function uses a list of (1) dependencies of business applications toward services, (2) services toward network accesses and (3) interdependencies between services. In addition to this are added cost values related to the temporary and permanent unavailability (UC) of business applications.

Thanks to those parameters, we can compute the cost of unavailability of all business applications before (on the real system) and after (on a simulated system) deploying a candidate RC_i, by checking recursively that the service dependencies is verified as shown in Equation 4.6.

$$IC(RC_i) = \sum_{ba \in businessApp} isImpactedBy(RC_i, ba) * UC(ba) \tag{4.6}$$

The impact cost is certainly the most important part to take into account when deploying a remediation but it is perhaps also the most difficult to quantify, as it is hard to estimate the cost if a business application is unavailable and to know which applications will be disrupted by a remediation candidate. It is therefore very important to provide security operators with indications about such a cost, to help them choose at best the remediation to deploy.

4.3.3 Ranking remediation candidates

These costs allow us to attach a global cost C to a candidate as shown below:

$$C(RC_i) = OC(RC_i) + IC(RC_i) \tag{4.7}$$

The candidate cost function can be considered as a ranking function taking as input an unsorted set of remediation candidates and that outputs a set of the same candidates sorted according to their cost. This allows a security operator to select one of the candidates that has the lowest cost.

As the remediation candidates cost function is only used to compare candidates with each other, even if the cost parameters are not assigned exactly, it does not change significantly the order between them. Thus, the details of the cost model parameters are not required, but only need to represent a tendency, in order to conserve the ranking between candidates. This assumption has also already been justified by Gonzalez-Granadillo *et al.* in [GJDC12].

4.4 Validation

4.4.1 Complexity

What must be well understood before talking about the complexity of our algorithms is that in this chapter, we propose remediations to attack *paths* and not to a whole attack *graph*. We thus have smaller complexity issues. Indeed, an attack graph is usually a large graph whereas, an attack path is smaller, because it focuses only on the most impacting or the most likely ways to access a target.

The complexity of the algorithm computing the candidates is not a significant issue, because (1) it is linear in the number of conjunctions of preconditions and (2) the number of remediations for one precondition is generally low. The algorithm computing SP depends highly on the structure of the input attack path, especially on the number of parents of each vertex. In the best case (each vertex has only one parent) this algorithm is linear in the number of vertices. In the worst case (each vertex has several parents), the complexity is exponential in the number of parents of `OR` vertices. This is the factor that most influences the complexity. We made simulations on non-realistic graphs with different varying parameters (number of parents, `OR` nodes, `AND` nodes, preconditions...) to validate these results. Nevertheless, in practice on several real use cases, we found that the number of parents for `OR` vertices in attack paths is generally low: an average of 1.7 per `OR` vertex in attack paths produced by MulVAL. This can be simply explained knowing that, in an attack path, as explained above, we only have few different possibilities to compromise a target, alternatives creating disjunctions in the attack path. It implies that this methodology generally scales well, if the attack paths are properly generated.

4.4.2 Experimental validation

4.4.2.1 Simulation of the network topology

In order to compute remediations to attack paths, we need a simulated network topology representing the target IS. Thus, we created a network simulator that accurately reproduces simple network behaviours of hosts: we are able to simulate exchanges between hosts, calculate routes, test if a packet can pass firewall rules, *etc.* We designed a pivot file in which we put the topological information needed by the simulator. We also created connectors to automatically build such a file. The first connector we built was a python server that gathered the topological information collected by agents deployed on Linux machines into the pivot format. The second connector was built for the European Research Project PoSecCo[1], where we had an ontology containing the network topology We thus implemented a connector that was querying in the ontology for the information needed. It has been extended for FI-WARE[2] as CyberCAPTOR and is available on Github[3].

[1] PoSecCo Research Project, `http://www.posecco.eu`
[2] FI-WARE Research Project, `https://www.fiware.org/`
[3] CyberCAPTOR, `https://github.com/fiware-cybercaptor/cybercaptor-server`

TABLE 4.1 – Main MulVAL preconditions and their remediations

Preconditions	Description	Possible remediations
hacl(src, dst, port, protocol)	The host *src* has access to *dst* on *port* using *protocol*	Deploy a firewall rule
vulExists(host, vulID, program)	*program* on *host* has a vulnerability *vulID*	Apply a patch or deploy a Snort rule
networkServiceInfo(host, program, protocol, port, user)	*program* on *host* launched as *user* open *port* using *protocol*	Stop this network service
hasAccount(user, host, account)	*user* has *account* on *host*	Disable this account

4.4.2.2 Generation of attack paths

We use the open source attack graph engine MulVAL [OGA05b]. It requires three types of inputs: topological, filtering and vulnerability information. We combine our simulated topology with a vulnerability scan done with Nessus[4] by merging the information about services and their vulnerabilities extracted from the scanner report into our topology. MulVAL inputs are stored in a file using the Datalog language. It outputs an XML file containing the logical attack graph computed thanks to its engine from which the attack paths are extracted.

4.4.2.3 Preconditions in MulVAL and their remediations

The main preconditions proposed by MulVAL to model attacks and their remediations is shown in Table 4.1. This table describes, for 4 types of MulVAL preconditions ($networkServiceInfo$, $hacl$, $vulExists$ and $hasAccount$), their description and possible remediations. We will focus here only on three relevant types of remediation for an enterprise: applying a patch, deploying a rule on an IPS (preventing *vulExists()*) and deploying a firewall rule (preventing *hacl()*), because the two other preconditions (*networkServiceInfo()* and *hasAccount()*) have no realistic remediations (*e.g.*, deleting a user account or uninstalling an application is generally not a viable solution).

4.4.2.3.1 Application of a patch In order to propose the right patch to a vulnerability, we use the parameter in the fact of the precondition *vulExists* containing the identifier of a vulnerability, generally a CVE (Common Vulnerabilities and Exposures) [5]. We use this identifier to look for known patches in the remediation database we describe in Subsection 4.4.2.4.

4.4.2.3.2 Deployment of a firewall rule To compute the firewall rule that should be deployed, we use all the parameters of the fact *hacl(src, dst, port, protocol)*. This precondition explains the network access the attacker needs for his attack. So, it should be negated by the rule to deploy, which should have the following form:

```
DROP FROM src TO dst:port USING protocol
```

It can be generated according to the type of firewall aimed. For example, we propose an automatic generation of iptables [6] firewall rules.

The last challenge we need to deal with for the firewall rules proposal is where it should be deployed. We use here the topology simulation presented in subsubsection 4.2.1.2 to determine

[4]Tenable Nessus, `http://www.tenable.com/products/nessus`
[5]MITRE Common Vulnerabilities and Exposures, `https://cve.mitre.org/`
[6]Netfilter iptables, http://www.netfilter.org/projects/iptables/index.html

the route followed by packets between *src* and *dest:port*. We then deduce on which machine the firewall rule can be deployed.

4.4.2.3.3 Deployment of an IPS rule The last type or remediation we will detail is the deployment of IPS rules for Snort [R+99] which prevent the exploitation of a vulnerability. For each *vulExists* related to a *hacl*, we know (1) The Snort rules that may exist to prevent the exploitation of the vulnerability by searching its identifier in the remediation database presented in Subsection 4.4.2.4, and (2) the network routes that may be used by the attacker to exploit this vulnerability, by using the simulation of the network and a deduction process similar to the calculation of the firewall rules. On each route, we must have a Network IPS where we can deploy the rule. Otherwise, the remediation is not possible. The rules we propose here must be used with Snort in inline mode and they begin with the *drop* keyword, meaning that we use it as an IPS.

4.4.2.4 Filling the remediation database

One challenge of the proposition of remediation is the ability to build a remediation database automatically. We will describe here how we solve it.

Database model We use a relational model stored in a SQLite file. We choose to use a model similar to the one used in the National Vulnerability Database[7] to represent vulnerabilities. Then, we added two tables corresponding to the types of remediations. In order to have a N-to-N association with the vulnerabilities, we also add a join table for each kind of remediation.

Patches We used the NVD[8] to find the links toward patches that correct vulnerabilities. Among the attributes related to a CVE, a *reference* can point to a website describing how to patch the vulnerability. So, we parse the dumps of the NVD, extract the links toward patches and store it in the database. Around 20% of the CVE have a "PATCH" reference attached.

Snort rules In the standard format of a Snort rule, there is an option "reference" which often contains a CVE. In the freely available database of rules provided by Sourcefire[9], nearly 50% of the rules are related to a CVE.

4.4.2.5 Providing the cost parameters

Operational costs Operational costs depend largely on the company and on the remediation. So, we choose in our prototype to assign parameters per types of remediation. Generally, the difference of operational costs between remediations of the same type is low, but it may also be possible to add the cost parameters into the remediation database, in order to be able to attach to each remediation specific operational costs parameters.

Impact costs The description of dependencies used for impact costs is also totally dependent on the information system and has to be provided by the security operator. To describe these dependencies, we use an XML file in which the dependencies relations are described according to a dependency graph.

[7] NIST National Vulnerability Database, `https://nvd.nist.gov/`
[8] NIST National Vulnerability Database, `https://nvd.nist.gov/`
[9] Sourcefire, `https://www.snort.org/products`

4.4.2.6 Simple experiment scenario

In order to validate the whole remediation method described in this chapter, we applied it on several test topologies. We will present the remediations on a simple scenario implementing the main concepts.

4.4.2.6.1 Network topology and attack scenario The first scenario we implement rely on a topology that we deployed on virtual machines. It contains 5 Linux-based hosts: a web server, a database server, an administration machine inside a LAN, a firewall that protects the LAN and the servers from the Internet, and the attacker's machine that is on the Internet. We configure the firewall in such a way that the web server is the only service that is accessible from the Internet and the LAN. The web server needs the database server to work properly and has a full access to it. The enterprise has two business applications using the IT: an Extranet which is rarely used and an Intranet which is used for all employees and is thus much more critical. The database server also contains some confidential information that the company wants to protect.

When the web server is exploited, the attacker can access the database server and with another exploit can try to gain access to all the data it contains. This is the attack path that will be described in the rest of this scenario.

4.4.2.6.2 Generation of the attack path and proposition of remediations To collect the topological information, we use the python agents described in Subsection 4.4.2.1. We generate MulVAL inputs and launch the attack graph engine, then extract the attack paths and select the one presented above. It chains the exploitation of two vulnerabilities: the first one CVE-2004-1315 is on the web server, the second one, CVE-2012-3951, is on the database server.

We use our prototype to visualise the attack path to correct and the remediation candidates. We present here the four most relevant candidates, ranked by their cost. We also explain for each candidate why its cost is low, medium or high.

1. The first candidate is the deployment on the firewall of the Snort rule sid:12610 that allows to block the exploitation of the first vulnerability. This remediation has a lot of advantages, because it doesn't interrupt any legitimate service, is not too expensive (deployment can be nearly fully automated), and blocks successfully the attack. This candidate has no impact costs, a low operational cost and thus a very low global cost, that is why this is the first one to be proposed.

2. The second one is the proposal of a patch to the first vulnerability. As the first one, it doesn't have any impact on normal service, but has much more operational costs, because deploying a patch need human intervention. So, this candidate has a low global cost and thus is the second one to be proposed.

3. The third one is a firewall rule that blocks all the traffic from the Internet to the web server on http port. It has a low operational cost, because it can be automatised, but has a medium impact because it cuts the access from the Internet to the web server, even if it keeps all the accesses from the LAN. So this candidate has a medium global costs and thus is the third to be proposed.

4. The last one is a firewall rule that blocks all the traffic to the web server on http port. It also has a low operational cost, but has a huge impact because it also cuts all the accesses for the employees of the LAN to the web server. So this candidate has a high global cost and thus is the last one to be proposed.

4.4.2.7 Results on PoSecCo's testbed

We will now present the results of this method applied on the testbed of the FP7 European Research Project PoSecCo[10].

The main use case in which the PoSecCo prototypes have been tested has two main business services: a broadcaster Internet distribution and a corporate streaming service. These services have several security requirements and run on a testbed that has been deployed during the project, on which prototypes have been tested. It contains around twenty machines (some are representing server farms) and eight routers. All the topological information needed for our prototype is collected from an ontology and the attack paths extracted are ranked according to their impact on security requirements.

On the twenty machines and eight routers, there are more than a thousand vulnerabilities in total. It was chosen for this project that there will be one attack path per target, gathering the relevant ways to compromise it. Thus, after establishing a list of five hosts to protect in priority, the operator has five attack paths to assess and correct. These attack paths contain between thirty and hundreds of nodes, the possible remediations are computed in a few seconds. Due to project limitations, only two types of remediations are proposed: patches and firewall rules. For each attack path, many candidates are proposed (up to a hundred), and are ranked according to their operational and impact cost. The first candidates (lower cost) offer the best compromise between efficiency and cost and should be the best option for a security operator.

During the project, the prototype implementing this approach was presented to end-users that compared their risk analysis and its remediation, in anticipation of a change in the testbed topology, with and without the prototype. The end-users, independent of the project, concluded that the scenario using this methodology was much more efficient: it reduces the analysis from four hours to twenty-two minutes and could reduce the number of people needed for this task from between three and six to only one. The result of this evaluation can be found in PoSecCo's Deliverable 1.7, in scenario SP06 [DMM+13].

4.5 Related work

The papers which describe the closest approach to our work are [WNJ06] and [AJN12]. In [WNJ06], Wang *et al.* base their analysis on the preconditions of an attack graph to compute ways to prevent attacks. But even when they evoke the cost to choose one remediation solution rather than another, they do not present a cost function to sort candidates as we did in this chapter. In [AJN12], Albanese *et al.* extend [WNJ06] mainly by adding a cost model, similar to the one we present here, and by improving the complexity of the algorithm to compute candidates. But what distinguishes our approach from both these is that we do not compute remediations to an attack *graph* but to attack *paths*, meaning that our algorithms are working with smaller inputs. We are convinced that it is much more sound and efficient to correct only the paths that are significant rather than reasoning on the global attack graph. This was assessed on realistic use-cases.

What is also original in our approach is that our remediation computation is generic. In [NJ09] and [NJ08], Noel and Jajodia propose various types of remediations to predefined types of attacks modelled by attack graphs. The method we present here is more generic. Indeed, the expressiveness of logical attack graphs allows the modelling of every attack described with `AND`/`OR` conditions and our method applies to all of them, without the limitations identified in the related work. Dealing with new attacks only implies to define remediation for potential new kinds of preconditions. These remediations can be simple or may require network topology simulation, as the ones presented in

[10] PoSecCo Research Project, `http://www.posecco.eu`

this chapter.

Furthermore, several databases that contain remediations exist. However, each database is dedicated to a type of remediation. For example, the NVD[11] contains information about patches and Snort rules databases contain only Snort rules[12]. In this chapter, we design a remediation database and fill it using several available online sets of data. This database contains different kinds of remediations and can be extended to provide new types of remediations.

4.6 Conclusion

We present in this chapter a method describing how to compute remediations for ranked attack paths extracted from an attack graph. Attack graphs have been widely used for assessing proactively the security level of an information system, we chose to use them in order to propose solutions to enhance this security level, by computing remediations preventing the most important attack paths. Using scored attack paths extracted from an attack graph allows us to remediate only the very likely or impacting paths that lead to main assets which is much more efficient than remediating the whole attack graph.

We have stated that the only vertices on which we compute remediations, within a logical attack path, are the preconditions. We have implemented algorithms to cluster these nodes into conjunctions of sufficient preconditions to be remediated, in order to protect the target of an attack path. Then, after explaining how to compute remediation actions to prevent a precondition, we detailed their combination with the sufficient conjunctions of preconditions to determine the candidates. As the operator has to choose one remediation among several candidates providing the same remediation objective, we assign to each remediation a global cost combining operational and impact costs. To calculate topological remediations to certain preconditions and to assess the effects of remediations on the system, we have designed a simulated network topology.

The logical model used for modelling attack graphs is deterministic and not dynamic. Thus, it has to be extended into a quantitative model to represent dynamic attacks and to model them more accurately, in order to generalise this remediation computation process into a response computation process. However, this logical model has the advantage to be efficiently generated and processed and is well suited to model potential attacks.

As a result, we have developed in this chapter a methodology to compute remediations that applies to static models. It has appeared that risk is a key indicator to select the attack paths to correct, by combining the likelihood of the path with the impact it may have on the system. So, in order to generalise this remediation computation methodology into a response computation methodology for occurring attacks, the first step is to build a reliable dynamic risk assessment model giving the riskiest assets that need to be corrected.

[11] NIST National Vulnerability Database, `https://nvd.nist.gov/`
[12] Sourcefire, `https://www.snort.org/products`

Chapter 5

Bayesian Attack Model

In chapter 4, we have taken a static model from the state of the art as introduced in chapter 2 and have developed a process to compute remediations preventing attacks that are the most likely to occur in a system. We came to the conclusion that the first component necessary to build a response process to occurring attacks is to build a reliable dynamic risk assessment model. There are two main metrics necessary for risk assessment: the likelihood and the impact of the likely futures. The impact component of the risk strongly depends on the system to study and of the organisation. Its computation can rely on a functional model of the system, but it cannot be fully automated. On the contrary, methodologies to estimate attacks' likelihood do not depend on the system in which they are implemented. In this chapter, we thus focus on the likelihood computation. Although many attack models have been proposed for static risk assessment, such as attack graphs (*cf.* chapter 2), no model is fit for dynamic risk assessment, as they usually lack sufficient granularity. Thus, in this chapter, we build a model that aims at assessing the risk brought by the residual attacks that remain accessible in a system.

Given the advantages brought by Bayesian attack graphs (*cf.* chapter 2), we consider that they provide a strong foundation for dynamic security modelling. Our objective is thus to propose solutions for the two issues highlighted in the state of the art: cycles and performance. Our proposal enables the use of Bayesian networks for dynamic risk assessment applied to real-scale information systems. The Bayesian Attack Model we propose is built from a Generic Attack Model. In this chapter, we first describe the Bayesian Attack Model architecture, its structure and the nodes probability tables. We validate this model with simulations and study its sensitivity toward its parameters. Finally, we compare the Bayesian Attack Model with the related work.

This chapter is part of the contribution described in subsection 1.3.1, the definition of a dynamic risk assessment model, and tackles the challenges described in section 1.2.1 and 1.2.3: the methodology to build an dynamic risk assessment model merging different information sources and taking advantage of the different types of attack representation.

5.1 Bayesian Attack Model architecture

The Bayesian Attack Model (BAM) described all along this section is built from a Generic Attack Model (GAM) and a set of alerts, related to Generic Attack Model's sensors. Figure 5.1 summarises the global architecture of the Bayesian Attack Model. In this example, it is built from a Generic Attack Model containing 3 states. The Bayesian Attack Model is composed of submodels called Bayesian Attack Trees (Bayesian Attack Tree). Bayesian Attack Trees and Bayesian Attack

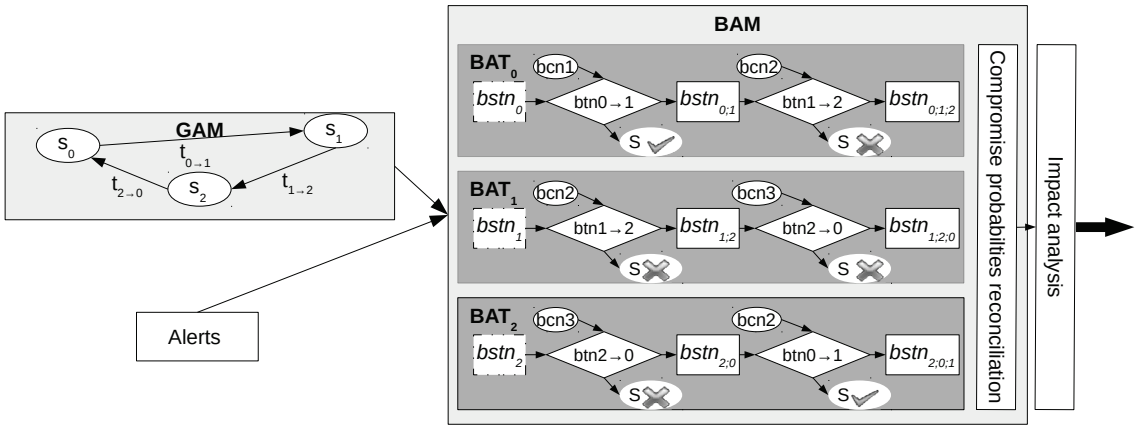


FIGURE 5.1 – Bayesian Attack Model Architecture

5.1.1 Breaking the cycles of a Generic Attack Model

As mentioned in subsubsection 2.4.2.2, attack graphs almost always contain cycles. As a Generic Attack Model is a generalisation of attack graphs and other attack models, it can also be cyclic. Solutions of the state of the art for Bayesian modelling either do not mention the cycle challenge, or they delete possible attacker actions.

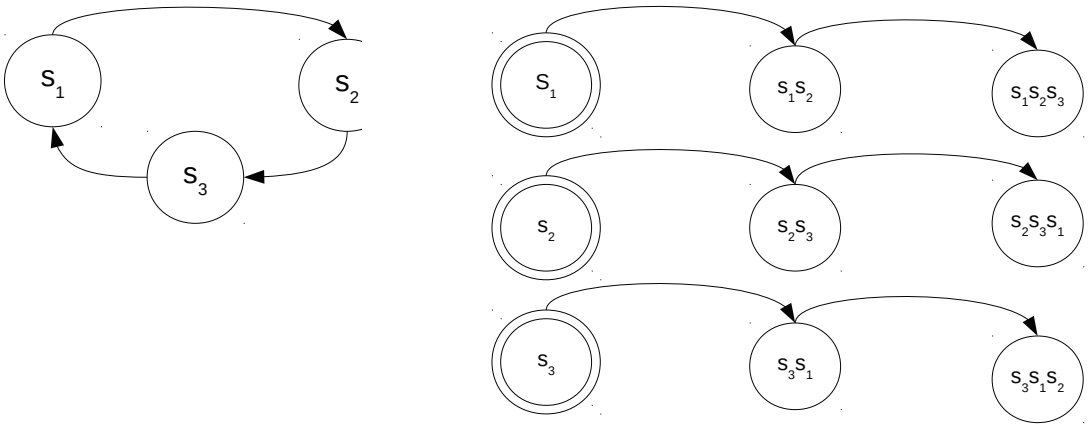
A solution to break cycles, while keeping all possible paths, is to enumerate all paths, starting from every possible attack source, keeping in the nodes a memory of the path of the attacker. So, using this memory, we can build an acyclic attack model by ensuring that the paths do not backtrack on already exploited nodes.

Figure 5.2 shows an example of the breaking of cycles in a Generic Attack Model. Figure 5.2a shows a Generic Attack Model of three states, containing a cycle (s_1 as a transition to s_2, s_2 has a transition to s_3 and s_3 has a transition to s_1). Figure 5.2b shows an equivalent model with three acyclic models, starting from all possible states (s_1, s_2 and s_3). Notice that the semantics inside the newly built nodes are a bit different: a node $s_1s_2s_3$ means that the attacker controls the state s_3, having first compromised s_1, then s_2, finally s_3.

Unfortunately, this cycle breaking process causes a combinatorial explosion in the number of states of the newly built model. We discuss in subsection 5.2.5 how we mitigate such limitation.

5.1.2 Grouping transitions

In a Generic Attack Model, there may exist many transitions between two states. Transitions can be of different types, depending on the attack (*cf.* Def. 6). Generally, in large Generic Attack Models, there are very few different types of transitions, as detailed in subsection 7.1.1. In order to reduce the size of the model, while preserving information, we group all transitions of the same



type between two states into a single vertex with (1) a new condition: a multivariable boolean function (usually, an `OR`) of all conditions applying to the grouped transitions, (2) an attached sensor node activated only when the boolean function of grouped sensors is true.

When several conditions c_i of a transition t are grouped in one condition c, we define the probability of successful exploitation associated with this new condition. For example, when grouping several conditions c_i "a vulnerability is exploited on the destination state" into one new condition c "at least one vulnerability of the list is exploited on the destination state", we assume that the exploitation of each vulnerability is independent, to compute its probability of exploitation $P(c)$. This is an acceptable approximation since we consider all the existing vulnerabilities between two states. Thus, the probability of exploitation $P(c)$ becomes:

$$P(c) = P(\bigvee_{i \in \{\text{vulnerabilities of } t\}} c_i) = 1 - \prod_{i \in \{\text{vulnerabilities of } t\}} (1 - P(c_i)) \tag{5.1}$$

5.2 Complete Bayesian Attack Model

5.2.1 Representation of a Generic Attack Model transition in the Bayesian Attack Model

A transition in the Generic Attack Model is an edge that can be associated with conditions and can be related to a sensor. In the Bayesian Attack Model, we detail the transitions, the conditions, and sensors as nodes, in order to model the probabilistic interactions between such elements, using the nodes detailed below. Each node represents a boolean random variable with two mutually exclusive states.

Definition 16 *A **Bayesian state node*** bam-stn$(\mathrm{s}_1, \cdots, \mathrm{s}_n)$*, with* $\forall i, \mathrm{s}_i \in \mathrm{S}$ *(cf. Def. 7), is a node of the Bayesian Attack Model representing the random variable describing the state of compromise of* s_n *using the path of the Generic Attack Model* $\mathrm{s}_1 \to \cdots \to \mathrm{s}_n$*. This node has two mutually exclusive states:* `compromised` *and* `not-compromised`*.*

Definition 17 *A **Bayesian transition node*** bam-tn(as)*, with* t ∈ T *(cf. Def. 7), is a node of the Bayesian Attack Model representing the random variable describing the success of* t*. This node has two mutually exclusive states:* `succeeded` *and* `failed`*.*

Definition 18 *A **Bayesian condition node*** bam-cn(c)*, with* c *a condition (cf. Def. 8), is a node of the Bayesian Attack Model representing the random variable describing that the condition* c *is fulfilled. This node has two mutually exclusive states:* `verified` *and* `not-verified`*.*

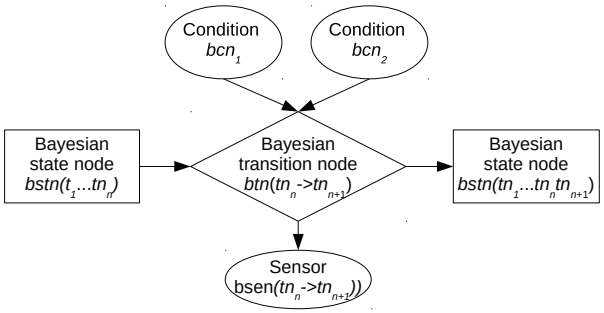
Definition 19 *A **Bayesian sensor node*** bam-sen(s)*, with* s *a sensor (cf. Def. 9), is a node of the Bayesian Attack Model representing the random variable describing the state of the sensor* s*. This node has two mutually exclusive states:* `alert` *and* `no-alert`*.*

These nodes are linked with edges, indicating that the child node has a conditional dependency to the state of its parents. For example, a Bayesian sensor node is the child of its related monitored element, and a Bayesian state node is the child of a Bayesian transition node.

Definition 20 *A **Bayesian edge*** bam-e*, is a link from a parent node to a child node that represents a conditional dependency of the child toward its parent.*

Figure 5.3 shows the details of the representation of a transition from s_n (source) to s_{n+1} (target). It is composed of a Bayesian transition node that binds a Bayesian state node to another one. This transition has two conditions ($bam - cn_1$ and $bam - cn_2$) and a sensor ($bam - sen$).

FIGURE 5.3 Bayesian Attack Model Transition



5.2.2 Bayesian Attack Tree and Bayesian Attack Model

The complete Bayesian Attack Model is composed of a family of Bayesian Attack Trees (BAT), as defined below, each one issued from one attack source.

Definition 21 *A **Bayesian Attack Tree** is a Bayesian network* BAT(AS, DAG,P) *where:*

- *AS is a special Bayesian state node, the attack source of this Bayesian Attack Tree.*
- DAG(BN,E) *is a polytree structure, constituted of*
  - *BN, the Bayesian nodes* BN=[bam-stn],[bam-tn],[bam-cn],[bam-sen] *(cf. Defs. 16-19)*
  - *E, the set of edges* $E = \{bam - e\}$ *representing a conditional dependency between the nodes (cf. Def. 20).*

- P *is a set of local probability distributions, associated with each node of* DAG. *As all nodes are discrete random variables, the local probability distributions can be specified within a conditional probability table.*

To build the whole structure of one Bayesian Attack Tree of the Bayesian Attack Model, we start from each state of the Generic Attack Model, as a potential attack source of a Bayesian Attack Tree. Then, we recursively add the transitions contained in the Generic Attack Model with the nodes described in subsection 5.2.1. To avoid cycles, each transition is added, as soon as its destination state has not been already compromised during the currently followed path, as described in subsection 5.1.1. This can be achieved thanks to the memory of past states in Bayesian state nodes. Figure 5.4 presents the algorithm used to build a Bayesian Attack Tree. Note that this algorithm introduces the `nbSteps` parameter that is presented in detail in subsection 5.2.5. This building process also ensures that the graph structure of each Bayesian Attack Tree is a polytree: a Directed Acyclic Graph for which there are no undirected cycles either. This allows to use very efficient exact inference algorithms in the Bayesian network such as Pearl's algorithm [Pea88].

FIGURE 5.4 – Algorithm to build a Bayesian Attack Tree

```
1: function buildBayesianTree(attackSource)        ▷ build the Bayesian Attack Tree tree from attackSource state
2:     bayesianTree ← createEmptyBayesianTree()
3:     bayesianAttackSource ← bayesianTree.addState(attackSource)        ▷ create a new Bayesian state node and add it to the tree
4:     for each transition in attackSource.getOutTransitions() do        ▷ get possible next transitions
5:         addTransition(0, bayesianAttackSource, transition, bayesianTree)
6:     end for
7: end function
8: function addTransition(currentNbStep, fromBayesianState, transition, bayesianTree)        ▷ add transition to the Bayesian Attack Tree bayesianTree
9:     if currentNbStep < nbSteps then        ▷ did not reach nbSteps yet
10:        if not fromBayesianState.history().contains(transition.destination) then        ▷ the destination state has not been already compromised
11:                ▷ then, we add all the Bayesian nodes related to the transition to the bayesianTree
12:            bayesianTree.addTransitionNode(transition)
13:            bayesianTree.addConditionsNodes(transition.conditions)
14:            bayesianTree.addSensorNode(transition)
15:            nextState ← bayesianTree.addState(transition.destination)
16:            bayesianTree.addSensorState(transition.destination)
17:            for each nextTransition in transition.destination.getOutTransitions() do
18:                addTransition(currentNbStep+1, nextState, nextTransition, bayesianTree)        ▷ add recursively the next possible transitions
19:            end for
20:        end if
21:    end if
22: end function
```

The complete Bayesian Attack Model is constituted of the set of all Bayesian Attack Trees. As we build a Bayesian Attack Tree from each state of the Generic Attack Model, the Bayesian Attack Model contains exactly $|S|$ Bayesian Attack Trees.

Definition 22 *The* ***Bayesian Attack Model*** Generic Attack Model $(\{BAT_i\})$*, is a family of* $|S|$ *Bayesian networks where, for all* i *in* $\{1..N\}, \mathrm{BAT}_i$ *is a Bayesian Attack Tree, whose attack source is the state* i *in the Generic Attack Model.*

5.2.3 Reconciliation of probabilities

As each Bayesian state node contains the history of states that can lead to this node, many Bayesian state nodes can represent the same state, in several Bayesian Attack Trees, when the attacker used a different path to reach it (*e.g.*, the node $bam - stn(s_1 \rightarrow s_2 \rightarrow s_4)$ is different from $bam - stn(s_2 \rightarrow s_3 \rightarrow s_4)$, even if the attacker has the control of the same state s_4 at the end.).

In the complete Bayesian Attack Model, we thus have many Bayesian state nodes representing the same state. However, what most interests a security operator is the attacks that are the most likely to compromise his states. Thus, as output of the reconciliation of probabilities, we assign to a state a probability of compromise that is the maximum of the probabilities of Bayesian state nodes targeting the same state.

$$P(s_k) = \max_{i \in \{1..N\}} P_{BAT_i}(s_k)) = \max_{i \in \{1..N\}} (\max_{\{\forall s_1..s_{k-1}\} \in P_{BAT_i}} bam - stn(s_1, ..., s_{k-1}, s_k)) \quad (5.2)$$

5.2.4 Bayesian Attack Model usage

We build our Bayesian Attack Model from the knowledge that the security operators have about the potential attacks in their system: the Generic Attack Model. Then, we change the state of the Bayesian sensor nodes according to the alerts received from the sensors.

`no-alert` If the sensor exists and is deployed in the system, as long as it has not issued any alert, all related sensor nodes of the Bayesian Attack Model (that may appear in several Bayesian Attack Trees) are set to the `no-alert` state.

`alert` When the sensor raises an alert corresponding to this element, the Bayesian sensor nodes are set to the `alert` state. If the sensor also gives an alert confidence probability, it is possible to set the state `alert` to this probability.

`no-info` The Bayesian nodes for which there is no compromise information (no deployed sensor, Bayesian condition nodes, *etc.*) are not set in any state and their probability are updated by the Bayesian inference.

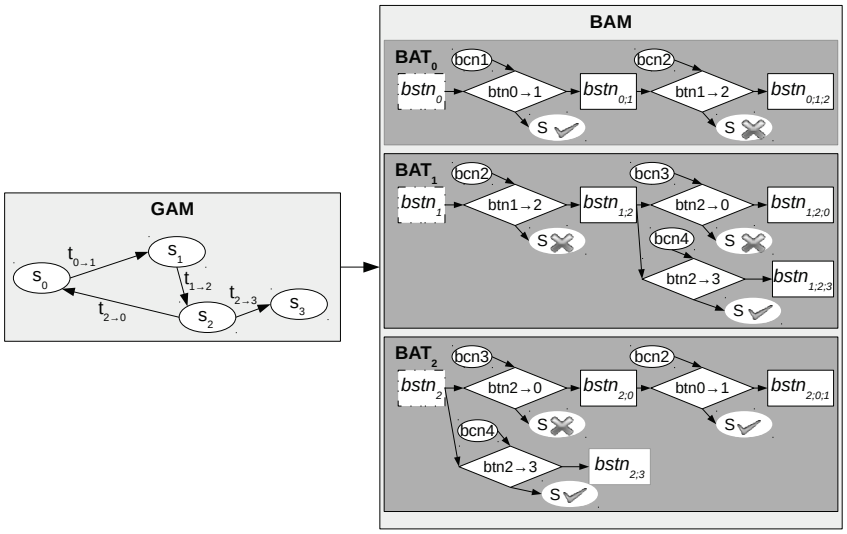
Each time the a node of a Bayesian Attack Tree changes state (when we receive an alert and fix a Bayesian node in a different state), we use a Bayesian network belief propagation algorithm (Lauritzen or Pearl's inference algorithm) to update the probabilities of each state at all the nodes. Then, for each state of the Generic Attack Model, the maximum probability of the state `compromised` of all related Bayesian state nodes, provides security operators with the probability of the states being compromised, as described in subsection 5.2.3.

5.2.5 Model size limitation

Use of a `nbSteps` parameter to prevent performance issues: The main limitation when implementing this model is the combinatorial explosion of the number of nodes, due to the redundancy introduced by the cycle breaking process. In order to improve the performance and prevent this combinatorial explosion, we limit the number of successive transitions added to each Bayesian Attack Tree, according to a `nbSteps` parameter. Thus, we can contain the number of nodes to process in the Bayesian Attack Model, as detailed in section 5.5

Figure 5.5 shows an example of the generation of a Bayesian Attack Model from a Generic Attack Model with 4 states, with `nbSteps` = 2. This Bayesian Attack Model is constituded of 3

Bayesian Attack Trees: BAT_0 starting from s_0, BAT_2 starting from s_1 and BAT_2 starting from s_2. Even if there is a path $s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3$ (3 steps) in the Generic Attack Model, as `nbSteps` = 2, this
$s_1 \rightarrow s_2$



Impact of the `nbSteps` parameter on the outputs of the Bayesian Attack Model: Thanks to the redundancy of the model, and as each state node is an attack source of a Bayesian Attack Tree, if a transition is discarded in a Bayesian Attack Tree, it will be in another Bayesian Attack Tree, closer to the Bayesian Attack Tree attack source. For example, in Figure 5.5, the transition $btn_{2\rightarrow3}$ is in BAT_1 and BAT_2. The probabilities of Bayesian state nodes in a Bayesian Attack Tree represent the probability of the attacker exploiting this node *starting from the attack source*. As long as no attack has been detected on a path, the probability of a state compromise decreases rapidly as a function of the length of the path between the attack source and the node. During initial probability computation, the probabilities of nodes far from the attack sources are very `low`. These probabilities are below the maximum used during the probability reconciliation detailed in subsection 5.2.3 and do not have any effect on final state compromise probabilities. In that case, the `nbSteps` parameter has no impact on the final results. For example, in Figure 5.5, as the transition $btn_{1\rightarrow2}$ has not been detected, $btn_{2\rightarrow3}$ will have a higher probability in BAT_2 than in BAT_1 (assuming that s_1 and s_2 have the same attack source probability, *i.e.*, $bstn_1$ and $bstn_2$ have the same probability).

The key limitation this parameter introduces is when attacks start being detected and introduced in a path. More precisely, the limitation arises when more than two alerts are injected in the model. For example, to compute the combined impact of two alerts relative to each other, they need to appear in the same Bayesian Attack Tree. The maximum compromise probability of the state related to the first alert will be in the Bayesian Attack Tree in which it is the attack source. If the second alert is attached to a state that is more than `nbSteps` away (*i.e.*, separated with more than `nbSteps` −2 missed alerts), it will not be in the same Bayesian Attack Tree and these two attacks will be taken into account separately. This will prevent the increase of probabilities of the nodes between the two alerts. For example, in Figure 5.5, as `nbSteps`= 2, there is no possible missed alerts. In this example, if $btn_{0\rightarrow1}$ and $btn_{2\rightarrow3}$ are part of the same attack scenario, as $btn_{1\rightarrow2}$ has not been detected, there is no Bayesian Attack Tree in which two successive transitions have `alert` sensors. Alerts may be separated by nodes without alerts for two reasons: if there are

not enough sensors or if there are false negatives, both undesired cases.

As a summary, the only case when the impact of the limitation of the Bayesian Attack Tree depth to `nbSteps` is significant is when there are more missed alerts than `nbSteps`−2 between two successive alerts for the same attack. These assumptions are validated by the experimental validation of section 5.7.

5.3 Conditional probability tables

We now specify the local probability distribution associated with each node, describing the probability dependencies of a node toward his parents. As the nodes are discrete random variables, we describe the probability dependencies using conditional probability tables.

Bayesian state nodes The conditional probability table of states highly depends on the type of transition and of the attack modelled. Thus, we fill the conditional probability table of a Bayesian state node with the one of the related state of the Generic Attack Model.

Bayesian transition nodes The conditional probability table of transitions highly depends on the type of transition and of the conditions. Thus, we fill the conditional probability table of a Bayesian transition node with the one of the related transition of the Generic Attack Model.

Sensor nodes A Sensor node has only one parent, the element related to the sensor. Its conditional probability table thus contains only two values and their complementaries representing the `false-positive` and `false-negative` rates attached to the sensor. For example, the conditional probability table of a Bayesian sensor node of a transition is described in Table 5.1.

TABLE 5.1 – Conditional probability table of a Bayesian sensor node related to a transition

bam-tn	`succeeded`	`failed`
bam-sen		
`alert`	1−`false-negative`	`false-positive`
`no-alert`	`false-negative`	1−`false-positive`

Attack sources The attack source of a Bayesian Attack Tree is a Bayesian state node without parents. As such, it does not have a complete conditional probability table, but only a prior probability value and its complementary. This *attackSourceProbability* parameter represents the *a priori* probability of having an attack issued from this node. It thus has to be set by the operators, knowing the risk that an attack starts from a state. It can be deduced from a risk evaluation methodology (*e.g.*, ISO 27005 [ISO11a]). In a typical information system, for example, a high probability can be set to the Internet (*e.g.*, 0.7), a medium one to servers in a demilitarised zone (internal subnetwork protected by a firewall exposing external-facing services on the Internet) (*e.g.*, 0.4), and a small one for production database servers (*e.g.*, 0.1).

Attack conditions The Attack conditions also do not have any parents. Their probability is the probability of occurrence $P(c)$ associated with the condition. It highly depends on the type of condition modelled by this node. For example, for a condition describing the presence of a list of vulnerabilities on a state. The estimation of the probability of occurrence of this group of conditions follows the process detailed in subsection 5.1.2, with values for each vulnerability, coming from the Exploitability Metrics of the CVSS, as explained in subsubsection 7.2.2.3.

5.4 Impact analysis

The last component necessary to build our Bayesian Attack Model is the impact analysis function. The goal of this function is to take the states of the Generic Attack Model with their compromise likelihood computed by the Bayesian Attack Model and an impact score associated with each state to give a risk score to states. We apply the usual equation to compute the risk R of a state s, with the probability of compromise P of the state computed by the Bayesian Attack Model, and the impact I of its compromise:

$$R(s) = P(s) \times I(s) \tag{5.3}$$

In this work we focus on the likelihood computation ($P(s)$), whose methodology is independent of the system studied, whereas the impact analysis ($I(s)$) strongly depends on the system and organisation. Thus, we only associated with each state a fixed impact score. Moreover, in order to validate the probabilistic results of the Bayesian Attack Model, we assign to each state the same impact value ($I(s) = 1, \forall s$) for the validation.

5.5 Bayesian Attack Model complexity evaluation

The main computation done on each Bayesian Attack Tree of the Bayesian Attack Model is the execution of the belief propagation algorithm (probability inference), computing the probability of all nodes, according to evidences, nodes set to a specific state. The complexity of the inference in a Bayesian network is directly linked to the number of nodes and structure of the network. We estimate the number of nodes M of a Bayesian Attack Tree, depending on $|S|$, the number of state nodes in the Generic Attack Model, and k the maximum number of *consolidated* transitions between two states in the Generic Attack Model (*i.e.*, the maximum number of different types of transitions). M is also strongly depending on the existence of transitions between the states. This depends strongly on the type of attacks represented in the Generic Attack Model. Thus for this complexity evaluation, we consider the worst case: there are k transitions between each pair of states. For each transition, we add ≈ 4 nodes to the Bayesian Attack Model (sometimes few more, according to the number of conditions). Thus, in the worst case, for each Bayesian Attack Tree, starting from an attack source, the number of nodes to add is

$$M \sim 4 \times k \times (|S| \times \cdots \times (|S| - \texttt{nbSteps} - 1)) = 4 \times k \times \frac{|S|!}{(|S| - \texttt{nbSteps})!} = \mathcal{O}(|S|^{\texttt{nbSteps}}) \tag{5.4}$$

Even if the number of nodes in each Bayesian Attack Tree is high, the Bayesian inference can be done efficiently. Indeed, as the structure is a *polytree*, efficient inference algorithms can be used. For example, Pearl's belief propagation algorithm is linear in the number of nodes [Pea88].

Thus, for each Bayesian Attack Tree, in the worst case, the complexity of the construction and probability inference $\mathcal{C}(BAT)$ is $\mathcal{C}(BAT) = \mathcal{O}(|S|^{\texttt{nbSteps}})$. Finally, for the whole Bayesian Attack Model, as there are at most $|S|$ attack sources, in the worst case, the complexity of the inference in the whole model $\mathcal{C}(BAM)$ is

$$\mathcal{C}(BAM) = |S|.\mathcal{C}(BAT) = \mathcal{O}(|S|^{\texttt{nbSteps}+1}) \tag{5.5}$$

The calculations on each Bayesian Attack Tree are independent. So, they may be easily done in parallel, which gives in practice, $\mathcal{C}(BAM) = \mathcal{O}(|S|^{\texttt{nbSteps}})$ with $|S|$ processors.

5.6 Bayesian Attack Model performance evaluation

In order to dynamically assess the risk of a system, the Bayesian Attack Model has to be evaluated each time a correlated alert, or a set of correlated alerts is received: the sensor nodes are set in their new states, then the probabilities are updated. The duration of such a process needs to be quite fast for the operator to properly understand the risk in operational time. We simulate random Generic Attack Models, with different parameters (number of states, number of transitions, topology) to evaluate the performance of the Bayesian Attack Model. Then, we generate random attack scenarios with four successive transitions. Finally, we evaluate the Bayesian Attack Model on the different scenarios.

We generate random realistic Generic Attack Models, as shown in Figure 5.6, containing from 1 to 50 states, regrouped in M clusters. In each cluster, there is a transition between each pair of states of the cluster (*i.e.*, states in a cluster are fully-connected "in mesh"). All the states of a cluster have a transition to all the states of the next cluster in cascades. By varying the number of clusters in the Generic Attack Model, we change the topology of the Generic Attack Model and its number of transitions. For example, in a topology of $M = 50$ clusters, each state of the Generic Attack Model has only one transition to a next state. In a topology of $M = 10$ clusters, for the maximum number of states ($|S| = 50$), each state of the Generic Attack Model has a transition to the

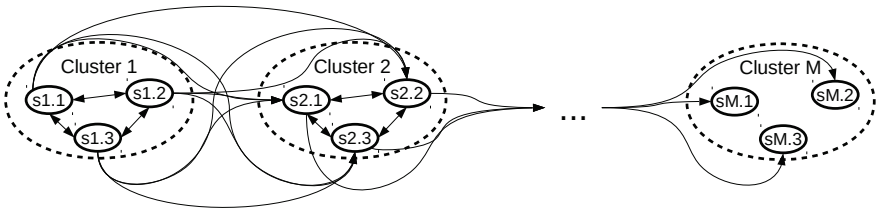


FIGURE 5.6 – Generic Attack Model topology for simulations

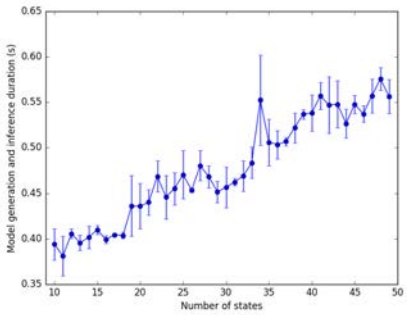
The results of the duration in seconds of the Bayesian Attack Model generation and the inference after the evaluation of one scenario of four successive transitions, on these generated Generic Attack Model, is displayed in Figure 5.7. The parameters of the Bayesian Attack Model are in the default values detailed in section 5.7. In Figure 5.7a, with 50 clusters (*i.e.*, 50 states in sequence, linked by only one transition) the processing time is very fast (< 0.60 seconds). In Figure 5.7b, with 30 clusters, the states are no longer in sequence and the processing time is a bit slower (< 1 seconds). With 20 and 10 clusters, in Figure 5.7c and 5.7d, the density of transitions between states is higher, so the duration is also higher, but still acceptable (< 2 minutes). Thus, these simulations show that for medium-sized topologies (up to 50 states), even when they are quite strongly connected (up to 10 clusters), the duration of the Bayesian Attack Model generation and of the inference remains acceptable (< 2 minutes).

5.7 Parameter sensitivity analysis

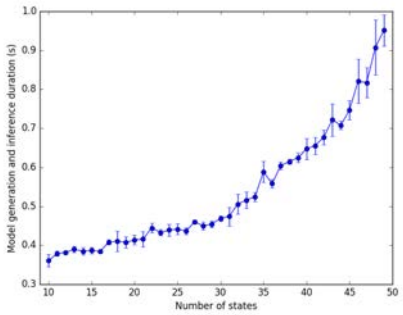
The Generic Attack Model has three parameters that can be customised, according to the system it represents and the possible attacks. Moreover, the Bayesian Attack Model adds three more parameters that can be customised by security operators using the tool. Thus it is interesting to evaluate how sensitive the Bayesian Attack Model is toward those parameters. This allows to evaluate if an uncertainty in the choice of the parameter has a significant impact on the final results of the Bayesian Attack Model.

FIGURE 5.7 – Duration in seconds of Bayesian Attack Model execution, according to the number of states and clusters in the Generic Attack Model

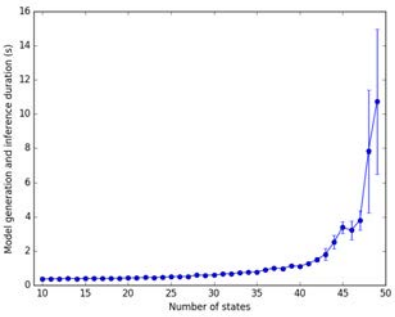
(A) 50 clusters



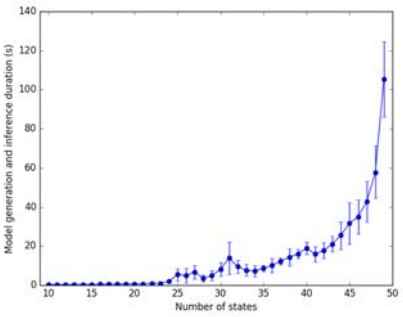
(B) 30 clusters



(C) 20 clusters



(D) 10 clusters



Note that the Generic Attack Model also allows to customise the whole conditional probability tables associated with the states or transitions, but it is difficult to represent the impact of changes in the conditional probability tables, as any logically valid probability value is possible. However, in practice the conditional probability tables are generally `AND` or `OR` tables, representing the logic to reach the places or carry out the transitions. In such a case, they do not have any impact on the probabilities, only on the required parents. For our experiments, we use `OR` conditional probability tables, because they were the most common in the real use cases.

5.7.1 Generic Attack Model parameters summary

Table 5.2 summarises the four parameters of the Generic Attack Model: `false-positive`, `false-negative`, `probability-attack-source` and `probability-unknown-attack`. Each parameter is associated with its description and the default value used in the experimentations.

5.7.2 Bayesian Attack Model parameters summary

Table 5.3 summarises the two parameters introduced by the Bayesian Attack Model: `nbSteps`, and `probability-new-transition`. Each parameter is associated with its description and the default value used in the experimentations.

TABLE 5.2 – Proposed values of the Generic Attack Model parameters

Parameter name	Meanings	Proposed value	Proposed value explanation
`false-positive`	False positive rate of each sensor. Sensors may raise an alert, even if the transition has not been completed, or the state was not reached by the attacker.	0.02	Even if operators try to minimise false positives, their occurrence is still inevitable.
`false-negative`	False negative rate of each sensor. Sensors may not raise an alert, even if the transition has been completed, or the state was reached by the attacker.	0.005	This value is generally smaller than the false positive rates as security operators try to prevent as much as possible the occurrence of false negatives for transitions for which they have deployed a sensor.
`probability-attack-source`	A priori probability of an attack issued from the state. An internal state may be the source of an attack. Probability of all other nodes than the actual attack source	0.1 for all nodes, 0.7 for the attack source of the scenarios	This value highly depends on the type of state that is represented, but except for well-known potential attack sources, the probability for other states can be relatively small as these states have fewer chances of being source of attacks.
`probability-unknown-attack`	Probability that an unknown attack allows to reach a place, without requiring the previous transition. This also allows to represent the security level needed in the system.	0.001	Very small probability of having a 0-day, an unknown vulnerability.

TABLE 5.3 – Proposed values of the Bayesian Attack Model parameters

Parameter name	Meanings	Proposed value	Proposed value explanation
`nbSteps`	Number of successive transitions to keep in the Bayesian Attack Model.	2	Allows to recognise multi-step attacks with at most 1 missing alert (*i.e.*, 2 steps means 3 transitions). *cf.* subsection 5.2.5 for full explanation.
`probability-new-transition`	Probability that the attack propagates through a new transition.	0.3	70% of chance that the attacker does not continue his attack. He may have already found on this state what he was looking for.

5.7.3 Parameter sensitivity analysis simulation scenarios

In order to study the impact of these parameters on the results of the Bayesian Attack Model, we simulated a random Generic Attack Model of 35 states as presented in section 5.6, on which we apply six detection scenarios, corresponding to an attack of three transitions, starting from the main attack source (*i.e.*, its probability of being the attack source is 0.7), with and without detection anomalies, as summarised in Table 5.4.

Then, we compute the compromise probabilities, results of the Bayesian Attack Model, according to the change of the parameter. Finally, we plot the variation interval of these probabilities, on the whole parameter variation interval, for all Generic Attack Model states. We explain in Figure 5.8 how to read the figures result of this sensitivity analysis. Figure 5.8a explains the bar graph figures. This kind of figure represents the whole variation of the states compromise probability, according to the variation of the parameter. The ordinate of the bars represent the compromise probability computed by the Bayesian Attack Model of the states of the Generic Attack Model. The height of each bar is the variation of compromise probabilities of related state, all along the variation of the parameter. On the left are presented the states known as `compromised` in the attack scenarios. Figure 5.8b explains the plot figures. This kind of figure gives an intuition on

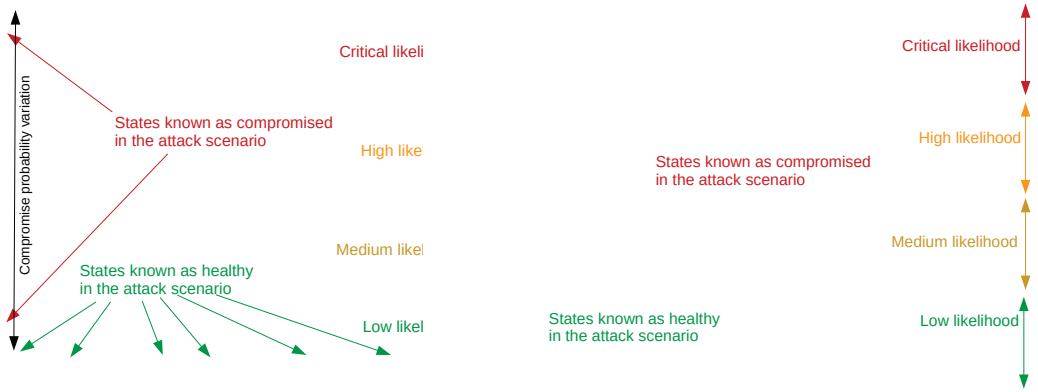
TABLE 5.4 – Parameter sensitivity analysis simulation scenarios

Scenario	$S_1 \rightarrow S_2$	$S_2 \rightarrow S_3$	$S_X \rightarrow S_{X+1}$	$S_3 \rightarrow S_4$	Comment
1	✓	×	×	×	First `alert`
2	✓	✓	×	×	Second `alert`
3	✓	✓	×	✓	Third `alert`
4	✓	×	×	✓	`no-alert` for the second transition
5	✓	×	×	✓	`no-info` for the second transition
6	✓	✓	✓	✓	Three `alert` and a false positive `alert` on another transition

Caption:
$\{S_i, i \in \{1,4\}\}$: States of the attack scenario; $\{S_X, S_{X+1}\}$: States of the false positive alert; $S_i \rightarrow S_{i+1}$: Detection on the transition from state S_i to state S_{i+1};
O: `no-info` sensor; ✓: Sensor set to `alert` ; ×: Sensor set to `no-alert`.

how the states compromise probability varies, according to the variation of the parameter. In both figures, the states that are known as `compromised` (source and/or destination of attack scenario transitions) are shown in *red* (bar or plot), whereas states known as healthy are shown in *green*. The coloured background represents different levels of compromise probability according to the value of the compromise probabilities: above 0.75 and `critical` probability, above 0.5 it is a `high` probability, above 0.25 it is a `medium` probability and above 0, it is a `low` probability.

FIGURE 5.8 – Explanation of the figures on variation of parameters



5.7.4 Bayesian Attack Model initial results

The results of the Bayesian Attack Model for this attack scenario with the parameters in their default values are shown in Figure 5.9.

Scenario 1 Figure 5.9a shows the status of the compromise probabilities after the first alert of the attack scenario. The source state of the transition that has been detected is the most likely source of attack so its compromise probability is `critical` (*i.e.*, $p > 0.75$), the transition destination state has a `high` compromise probability(*i.e.*, $0.75 > p > 0.5$): it is not yet sure that an attack is happening, the alert might be a false positive. The other states have very `low` compromise probabilities (*i.e.*, $p < 0.25$), since they have not been detected.

Scenario 2 After the second alert, the compromise probabilities shown in Figure 5.9b confirms that an attack is actually happening: the first two `compromised` states have `critical` probabilities(*i.e.*, $p > 0.75$), and the third `compromised` state has a `high` compromise probability(*i.e.*, $0.75 > p > 0.5$).

Scenario 3 After the third alert, the compromise probability of the fourth attacked state also increases, as shown in Figure 5.9c. However, this probability only slightly increases (until a `medium` probability). This is surely because of the `nbSteps` parameter, that makes this state be in another Bayesian Attack Tree than the three others `compromised` states, Bayesian Attack Tree whose attack source probability is much lower than the actual attack source.

Scenario 4 The fourth scenario, represented in Figure 5.9d, contains a false negative (the second transition). As a result, the compromise probabilities of the states are lower than for the third scenario, but still allow to distinguish the `compromised` states, from the `not-compromised` ones. In practice, for such a case, a security operator would need to investigate if the scenario had happened, or if the third alert was a false positive.

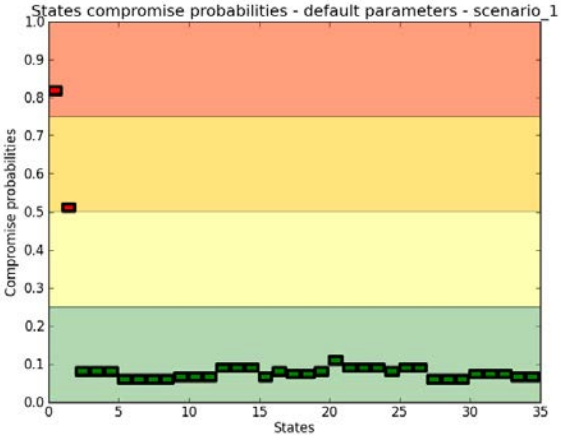
Scenario 5 The fifth scenario, represented in Figure 5.9e, contains a transition for which there is no information. Thus, the probabilities are between the scenario with a false negative, and the scenario with all alerts (in particular for the second `compromised` states) .

Scenario 6 The last scenario contains all alerts, but also includes a false positive alert (not related to the previous ones). We can see this alert with the `not-compromised` states whose probability have increased in Figure 5.9f, compared to the results in Figure 5.9c. However, as this alert is not part of the global attack scenario, the probabilities of the "false positive states" are lower than the actually `compromised` states. Thus, the Bayesian Attack Model allows to distinguish the false positives from the actually happening attack scenarios.

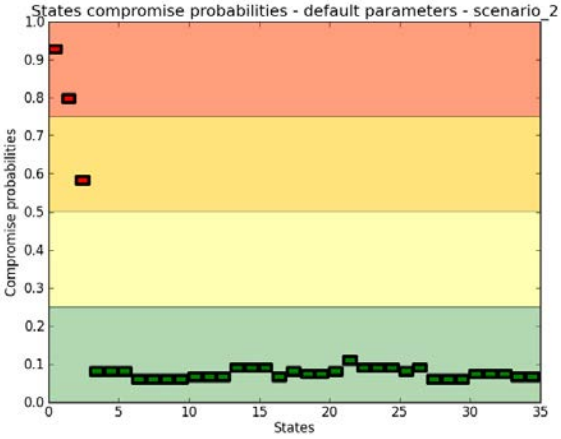
So, the compromise probabilities computed thanks to the Bayesian Attack Model shows the evolution of an attack scenario, according to the received alerts. The scenarios 1 to 3 shows the normal evolution of an attack where all attack transitions are detected. In such case, each new alert confirms that this multi-step attack is actually happening. The only limitation of the Bayesian Attack Model that can be noticed, thanks to this experimentation, is due to the `nbSteps` parameter preventing a large increase of the states far from the actual attack source.

FIGURE 5.9 – Bayesian Attack Model results with parameter default values

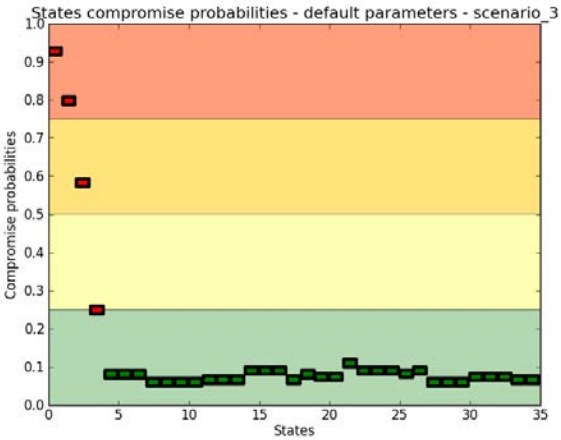
(A) 1st `alert`



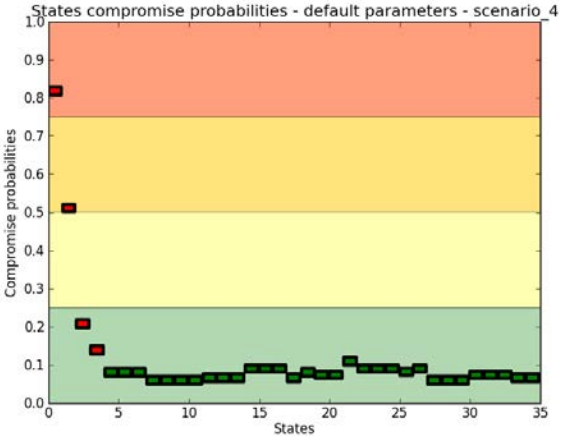
(B) 1st and 2nd `alert`



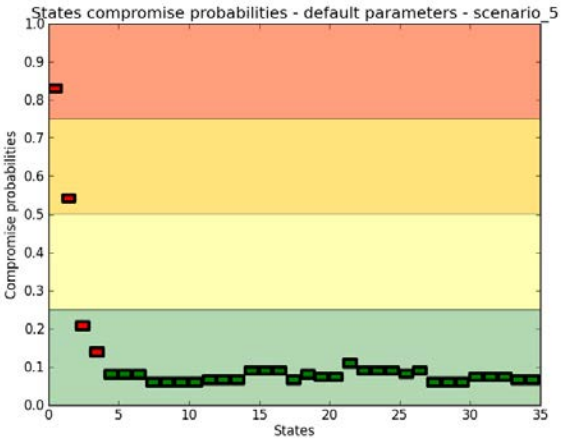
(C) 1st, 2nd and 3rd `alert`



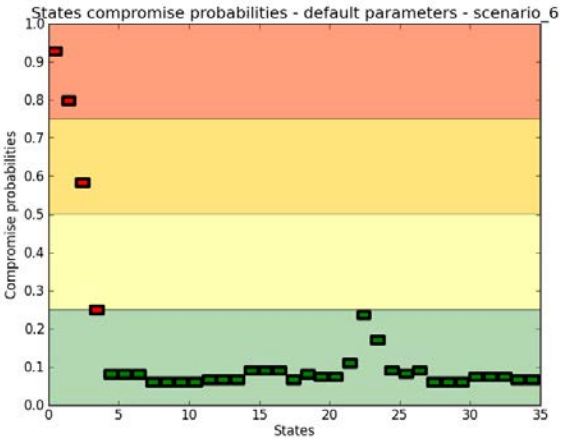
(D) 1st `alert`, 2nd `no-alert`, 3rd `alert`



(E) 1 `alert`, 2nd `no-info`, 1st `alert`



(F) 1 `alert`, 1 false positive `alert`, 3rd and 4th `alert`



5.7.5 Generic Attack Model parameter: `false-positive`

The `false-positive` parameter is associated with each sensor of the Generic Attack Model. It represents the probability of an alert raised by the related sensor to be a false positive alert, *i.e.* an alert raised without successive attack of the related state or transition.

Generally, to be usable in practice and limit irrelevant investigations, sensors are configured to strongly limit the probability of false positives. Thus the `false-positive` parameter should remain low (*e.g.*, from 0 to 5%). However, to be sure to take into account sensors with higher `false-positive` values (for example, an anomaly detection sensor with many false positives), we analyse the sensitivity of this parameter on its whole variation interval $]0, 1[$. Figure 5.10 shows the results of this analysis.

From 0 to 0.2, the `false-positive` parameter is very sensitive, but only for `compromised` states. The compromise probabilities decrease with the increase of this parameter. The more alerts there are in the attack scenario, the more sensitive this parameter is, for the alerts involving more detected transitions. We can see that, for example, with the evolution from Figure 5.10a to 5.10c. This can be explained by the fact that if there are more chances that the alerts can be false positives, this chance is larger for the states far from the most potential attack source, that need several alert (potentially false positives) to be attacked. In Figure 5.10d and 5.10e, with a `no-alert` or `no-info` sensor, the impact of this parameter is even bigger, as the certainty of this attack is only for very low values of the `false-positive` parameter. Note also that, for the Scenario 6 of Figure 5.10f, the `not-compromised` states concerned by the false positive alert have `high` compromise probability, for very accurate sensors (very low values of the `false-positive` parameter), which seems legitimate. The `false-positive` parameter is insensitive with `not-compromised` states, except for the scenario 6, containing a false positive alert.

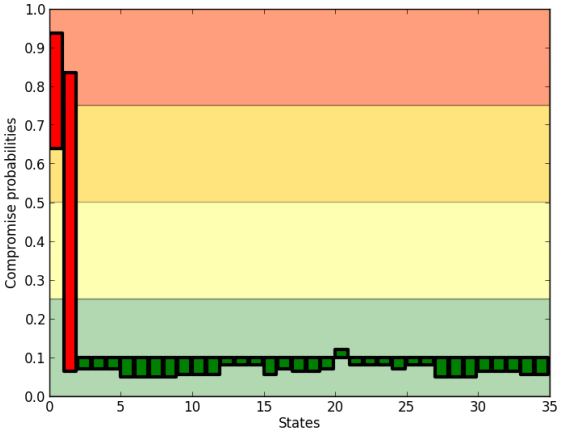
From 0.2 to 0.95, the `false-positive` parameter is a little sensitive for the `compromised` states that are significantly impacted by this parameter from 0 to 0.2. It is insensitive for `not-compromised` states.

From 0.95 to 1, for all scenarios, there is a little increase of compromise probabilities for both `compromised` and `not-compromised` states, which all converge to their initial attack source probability (`probability-attack-source` for all sources, except the actual attack source that has a 0.7 probability in these simulations).

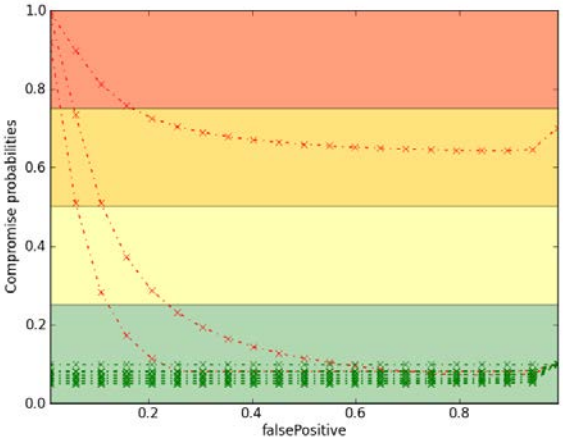
This sensitivity analysis shows that, especially from 0 to 0.2, the `false-positive` parameter has an important impact on the compromise probabilities of the `compromised` states of the Bayesian Attack Model. With the increase of the Generic Attack Model `false-positive` parameter, the compromise probabilities of the `compromised` states decrease quickly. This variation interval is the most frequent interval for this parameter. Thus the `false-positive` parameter will have to be calibrated carefully, in order to have accurate compromise probabilities for `compromised` states. This parameter has a low impact on the compromise probabilities of `not-compromised` states. However, in most cases, this parameter does not impact the ranking between the `compromised` states.

FIGURE 5.10 – Sensitivity of the `false-positive` parameter from 0 to 1

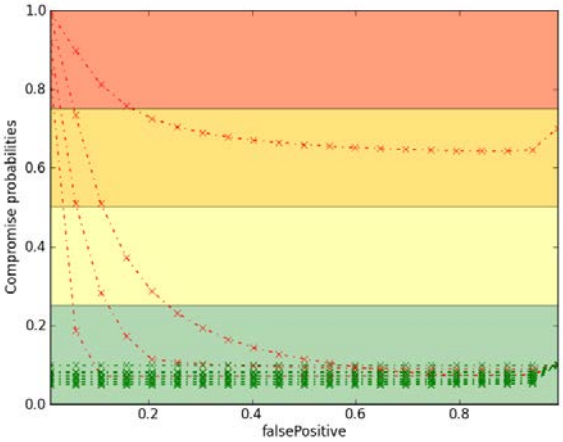
(A) Scenario 1



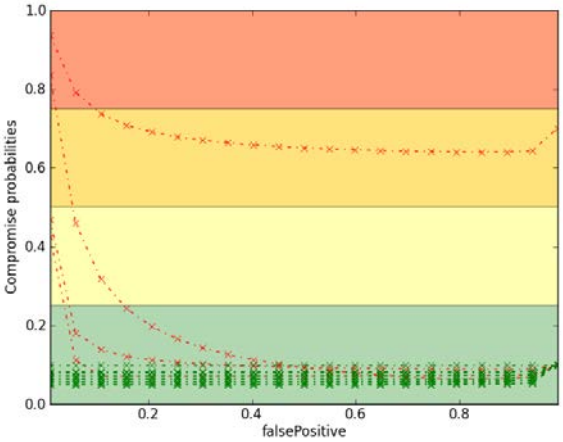
(B) Scenario 2



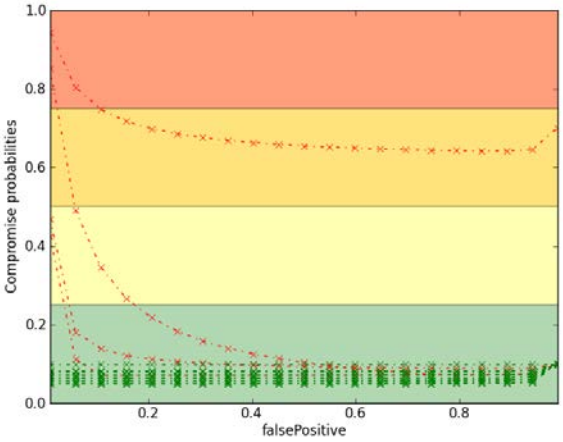
(C) Scenario 3



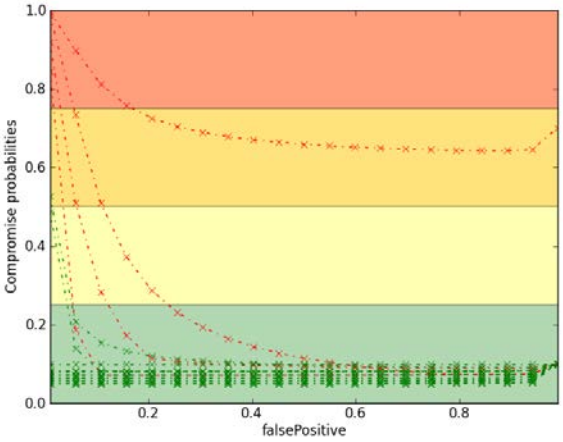
(D) Scenario 4



(E) Scenario 5



(F) Scenario 6



5.7.6 Generic Attack Model parameter: `false-negative`

The `false-negative` parameter is associated with each sensor of the Generic Attack Model. It represents the probability of false negative alerts for a sensor, *i.e.* an alert that is not raised by the sensor, while the attack actually succeeded.

Generally, to limit a wrong security feeling (*i.e.*, feeling safe, while there is no real protections), sensors are configured to limit at most the false negatives for known attacks, for which a detection method is enabled. Thus the `false-negative` parameter should stay low (*e.g.*, from 0 to 5%). Beyond, the detection measures are generally not deployed. However, to be sure to take into account sensors with higher `false-negative` values (for example, a detector of a metamorphic malware), we analyse the sensitivity of this parameter on its whole variation interval $]0, 1[$. Figure 5.11 shows the results of this analysis.

From 0 to 0.3, the `false-negative` parameter is a little sensitive for `compromised` states. The compromise probabilities slowly decrease with the increase of this parameter. The impact is approximately the same, whatever the number of alerts there is in the scenario. The most important impact is in Scenario 6, in Figure 5.11f for which the probability of the last `compromised` state decreases faster than the states of the false positive alert. Thus after 0.05, there the `not-compromised` state, source of the false positive transition has a probability higher than one of the `compromised` state. The `false-negative` parameter is insensitive with `not-compromised` states, except for the scenario 6, containing a false positive alert.

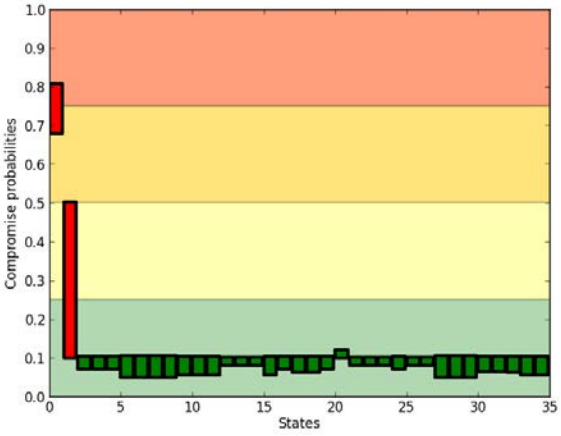
From 0.3 to 0.9, the `false-negative` parameter is moderately sensitive for the `compromised` states. The compromise probabilities decrease with the increase of this parameter, but their order does not change. The impact is similar whatever the number of `compromised` states there is in the scenario (see for example Figure 5.11a to 5.11c). For Scenario 4 and Scenario 5, with either one `no-alert` or one `no-info` sensor, the decrease curve goes slower, as it is more likely that there has been a false negative or that the third alert is a false positive. On all scenarios, the probability of `not-compromised` states slowly converge.

From 0.95 to 1, for all scenarios, both `compromised` and `not-compromised` states have converged to their initial attack source probability (`probability-attack-source` for all sources, except the actual attack source that has a 0.7 probability in these simulations). Thus the variation of probability is very light.

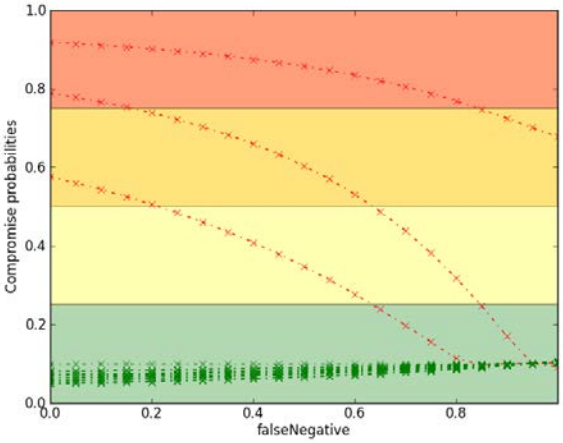
This sensitivity analysis shows that, on its whole variation interval, the `false-negative` parameter has a medium impact on the compromise probabilities of the Bayesian Attack Model. With the increase of the Generic Attack Model `false-negative` parameter, the compromise probabilities of the `compromised` states decrease slowly. Moreover, on its usual variation interval (from 0 to 0.1), the `false-negative` parameter has a low impact. It has almost no significant impact on the probability of `not-compromised` states. This parameter almost does not impact the ranking between all `compromised` nor `not-compromised` states, except for the scenario 6 with a false positive, in which, for low value of the parameter (0.05), the false positives get higher values than one actually `compromised` state.

FIGURE 5.11 – Sensitivity of the `false-negative` parameter from 0 to 1

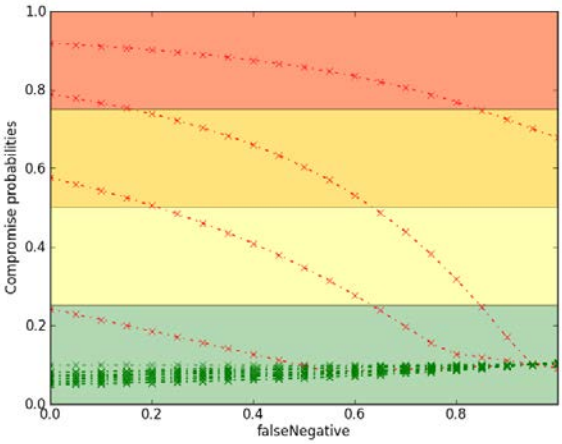
(A) Scenario 1



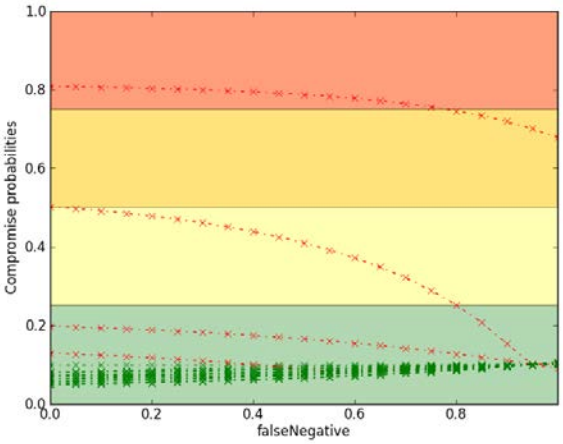
(B) Scenario 2



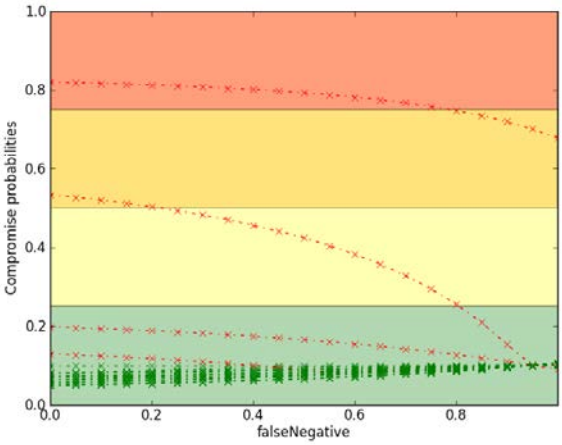
(C) Scenario 3



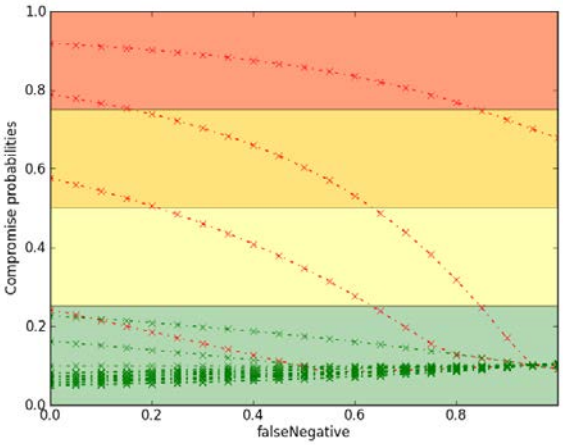
(D) Scenario 4



(E) Scenario 5



(F) Scenario 6



5.7.7 Generic Attack Model parameter: `probability-attack-source`

The `probability-attack-source` parameter is the probability associated with each state of the Generic Attack Model, representing the probability of this state to be a source of attack. The value of this parameter is set by the security operators according to the risk analysis of the system.

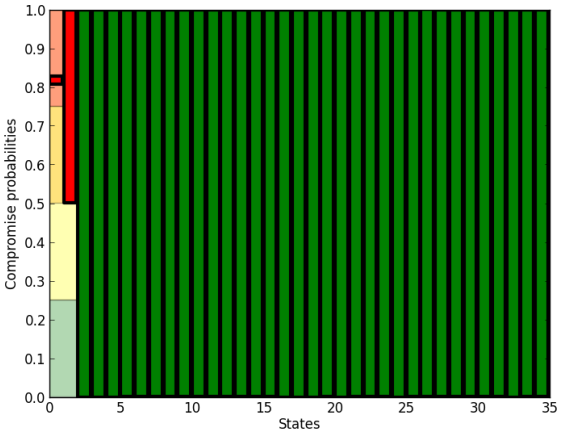
According to the state, this parameter can take any value possible from 0 to 1. We thus simulate the variation of this parameter on $[0, 1]$ and analyse its impact on the states compromise probabilities. This parameter can generally be deduced from a risk evaluation methodology (*e.g.*, ISO 27005 [ISO11a] or EBIOS [Sec04]). Figure 5.12 shows the results of this analysis.

From 0 to 1, the impact of the `probability-attack-source` parameter is relatively homogeneous for all scenarios, and on the whole variation interval. The impact of this parameter is important, for all states. The `probability-attack-source` parameter determines the probability of all states being an attack source, except the actual attack source, the only state whose probability does not change in Figures 5.12a to 5.12f, on the whole parameter variation interval. Thus, the value of the `probability-attack-source` parameter is the minimum that each state, either `compromised` or `not-compromised` will have in the Bayesian Attack Model. Moreover, the `compromised` states generally increase quicker than the `not-compromised` ones, as their transition source is more and more likely with the increase of the parameter. With high parameter values, the chaining of attack transitions in the scenario is less important.

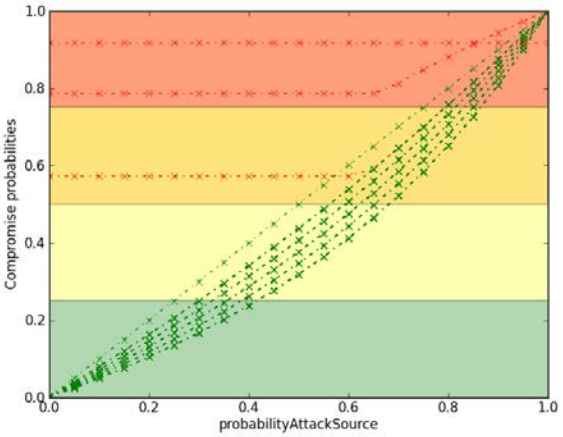
This sensitivity analysis shows that, on its whole variation interval, the `probability-attack-source` parameter has a very significant impact on the compromise probabilities of the `not-compromised` states of the Bayesian Attack Model. With the increase of this parameter, the compromise probabilities of the `not-compromised` states increase regularly. The compromise probability of `compromised` states is also increasing with this parameter, but it is less important than `not-compromised` states. With significant variation of the `probability-attack-source` parameter, the ranking between either `compromised` or `not-compromised` states changes.

FIGURE 5.12 – Sensitivity of the `probability-attack-source` parameter from 0 to 1

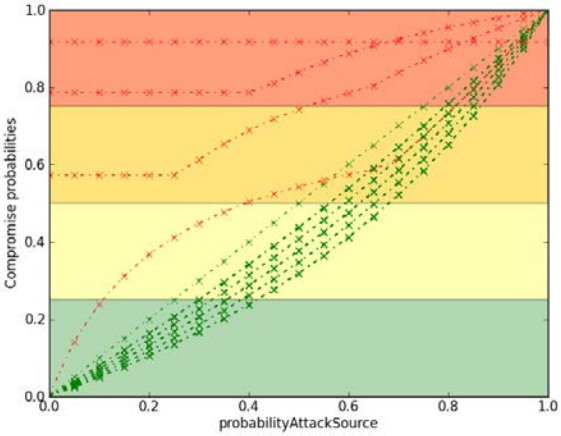
(A) Scenario 1



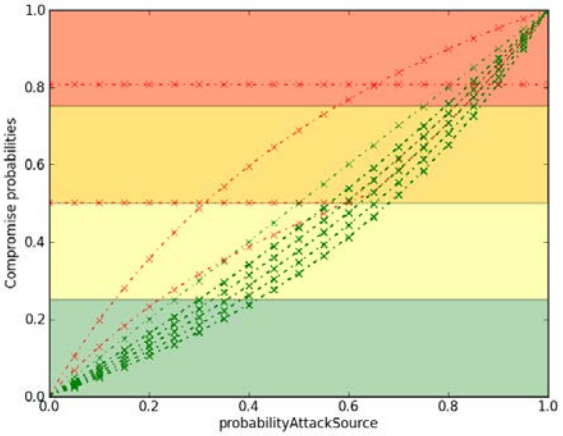
(B) Scenario 2



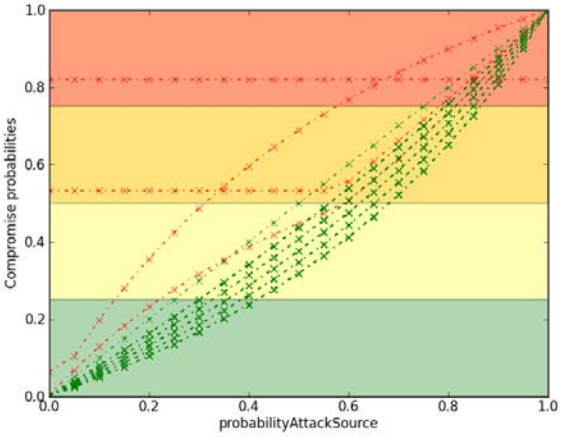
(C) Scenario 3



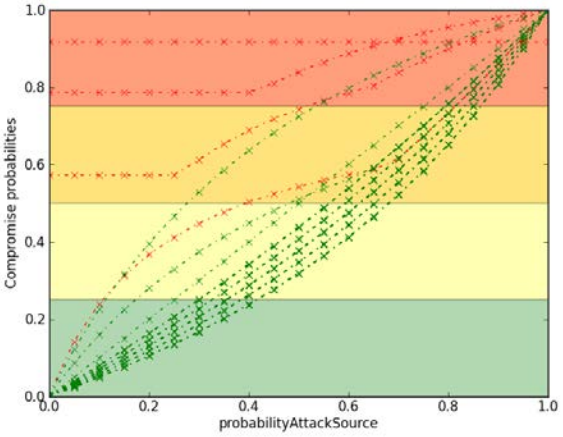
(D) Scenario 4



(E) Scenario 5



(F) Scenario 6



5.7.8 Generic Attack Model parameter: `probability-unknown-attack`

The `probability-unknown-attack` parameter is the probability that an unknown attack allows the attacker to reach a place, without requiring the previous transition. This is a kind of wild card representing all the attacks unknown for the defender, but known from the attacker.

This value should usually stay very low ($< 1\%$), otherwise this means that there is an issue in the modelling of the attack scenarios that can happen in the system (*e.g.*, not good knowledge of the system by the defender, partial knowledge of the vulnerabilities that are in the system, *etc.*). However, there might be good reasons to set higher values to this parameter. For example, in systems in which a very high level of security is needed, we can increase the value of the `probability-unknown-attack` parameter, to represent very motivated attackers that may have access to unknown attack transitions (*e.g.*, unknown 0-days vulnerabilities). We thus simulate the variation of this parameter on $[0, 0.5]$ and analyse its impact on the states compromise probabilities. Figure 5.13 shows the results of this analysis.

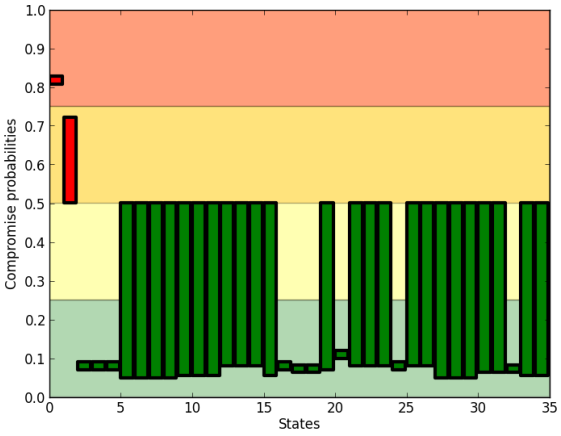
From 0 to 0.05, the `probability-unknown-attack` parameter is a little sensitive for `compromised` states. The compromise probabilities slowly increase with the increase of this parameter. The more alerts there are in the attack scenario, the more important increase there is of the `compromised` states. For example, in Scenario 2 of Figure 5.13b, the compromise probabilities of `compromised` states increase slowly on this variation interval. In Scenario 3 of Figure 5.13c, the compromise probability of the newly attacked state increase faster. The `probability-unknown-attack` parameter is insensitive with `not-compromised` states.

From 0.05 to 0.5, the `probability-unknown-attack` parameter is moderately sensitive for both the `compromised` and `not-compromised` states. The compromise probabilities increase with the increase of this parameter, but the order of `compromised` states and between `compromised` and `not-compromised` states does not change, except for very high values of the parameter (> 0.35). In all scenarios, more than half of `not-compromised` states increase slowly with the increase of the parameter. The probability of the other states do not increase. This must be due to the fact that these states cannot be attacked directly (*i.e.*, with only one transition) from the `compromised` states. For such states, the increase of the `probability-unknown-attack` parameter has nearly no impact, as the attacker will need to exploit two or more unknown transitions to reach them. For the scenario 4 of Figure 5.13d, with a `no-alert` sensor, and the scenario 5 of Figure 5.13e, with a `no-info` sensor, the impact of the `probability-unknown-attack` is more important for the states after the `no-alert` sensor, as these states are more likely to be reached after an unknown attack (that obviously is not detected). This effect is similar for two `not-compromised` states of the scenario 6 of Figure 5.13f, with a false positive alert. Indeed, the states of the false positive alert are more likely to have been attacked with an unknown attack, for a higher value of the `probability-unknown-attack` parameter.

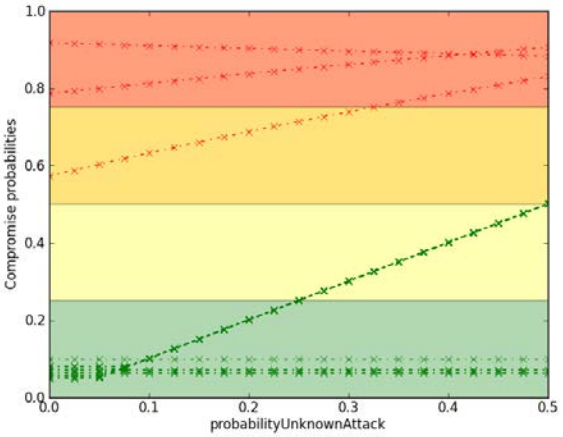
This sensitivity analysis shows that, on its common variation interval (0 to 1%), the `probability-unknown-attack` has no real impact on the compromise probabilities of both `compromised` and `not-compromised` states in the Bayesian Attack Model. Moreover, even with higher values, there is an impact on compromise probabilities, but it does not change the order of `compromised` states and between `compromised` and `not-compromised` states.

FIGURE 5.13 – Sensitivity of the `probability-unknown-attack` parameter from 0 to 0.5

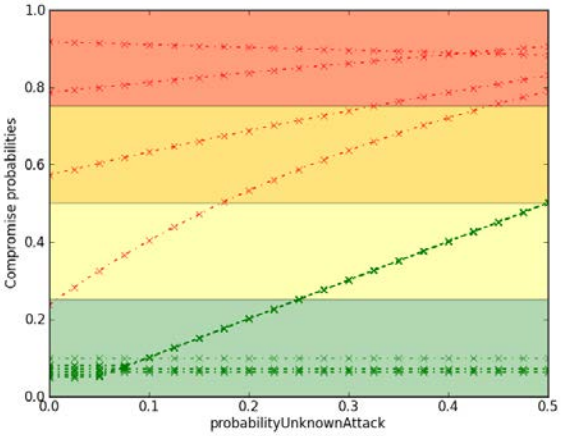
(A) Scenario 1



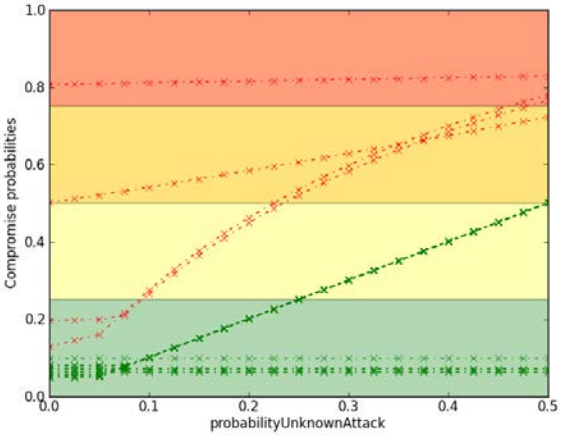
(B) Scenario 2



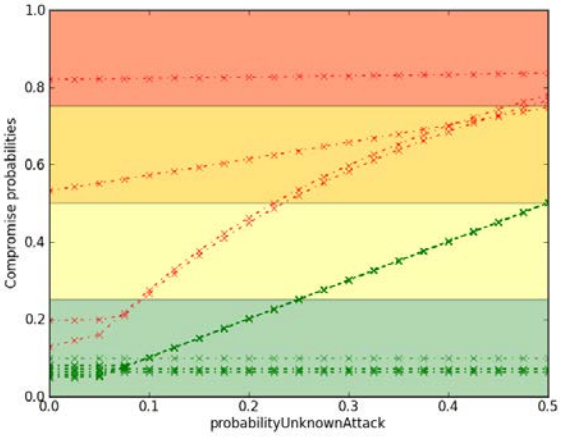
(C) Scenario 3



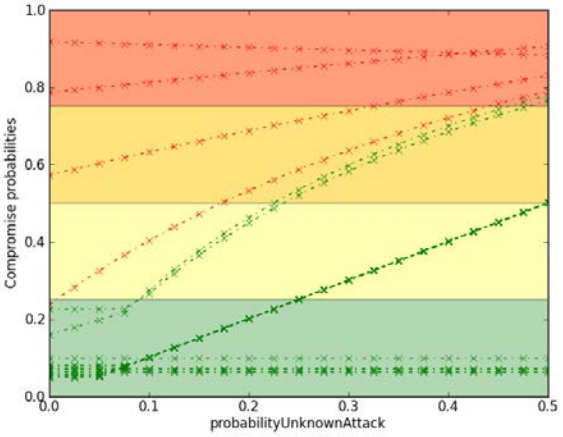
(D) Scenario 4



(E) Scenario 5



(F) Scenario 6



5.7.9 Bayesian Attack Model parameter: `nbSteps`

The `nbSteps` parameter represents the number of successive transitions to keep in the Bayesian Attack Model. This parameter prevents the combinatorial explosion of the number of nodes in the Bayesian Attack Model. The detailed explanations about this parameter are in subsection 5.2.5.

On the Generic Attack Model with 35 states, we cannot increase this parameter to high values to analyse the impact of the `nbSteps` parameter with high values. Thus, we analyse the impact of the `nbSteps` parameter on the Generic Attack Model of 35 states used for other parameters, as well as on a smaller topology of 12 states. Figure 5.14 shows the results of this analysis.

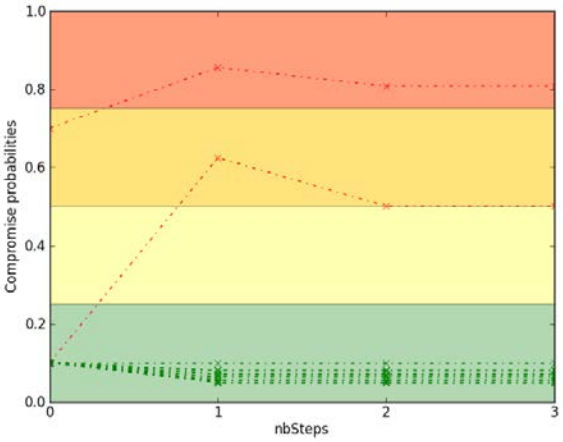
On Generic Attack Models of 35 states, the `nbSteps` parameter is very sensitive for `compromised` states, with low values. With the increase of the `nbSteps` parameter, the compromise probabilities of `compromised` states vary until they converge to their final value. By comparing Scenario 1 of Figure 5.14a and Scenario 2 of Figure 5.14b, we can notice that the final value seems to be reached when the `nbSteps` parameter is equal to the number of the actually detected transitions + 1. This parameter is moderately sensitive for `not-compromised` states. In Scenario 1 of Figure 5.14a and Scenario 4 of Figure 5.14c, there is a decrease in the compromise probabilities of `compromised` states, from `nbSteps` = 1 to `nbSteps` = 2. Indeed, as there is only one transition that has been detected, when there are two transitions in the model (*i.e.*, `nbSteps` = 2) the second transition that has not been detected make decrease the value of the probabilities of `compromised` states. In the other case, for 3 of Figure 5.14b and 6 of Figure 5.14d, the increase of the `nbSteps` parameter makes the probabilities of `compromised` states increasing. Indeed, when there are a sufficient number of steps in the Bayesian Attack Model, all attack transitions of the scenario are in the same model and contribute to the increase of compromise probabilities.

On Generic Attack Models of 12 states, the analysis of the sensitivity of the `nbSteps` parameter confirms the results we have for the biggest model. With the increase of the `nbSteps` parameter, the compromise probabilities of `compromised` states vary until they converge to their final value. This final value is reached when the number of detected transitions in the model is `nbSteps` −1. This parameter is a little sensitive for `not-compromised` states.

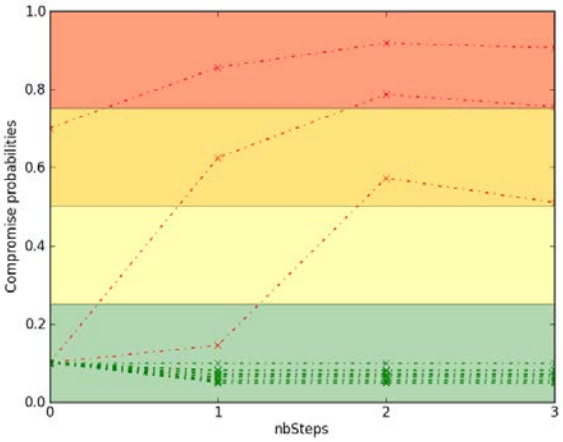
This sensitivity analysis shows that the `nbSteps` parameter impacts particularly the compromise probabilities of the `compromised` states of the Bayesian Attack Model. With the increase of this parameter, the compromise probabilities of the `compromised` states vary until they converge to their final value. The `not-compromised` states are moderately impacted by the variation of this parameter.

FIGURE 5.14 – Sensitivity of the `nbSteps` parameter

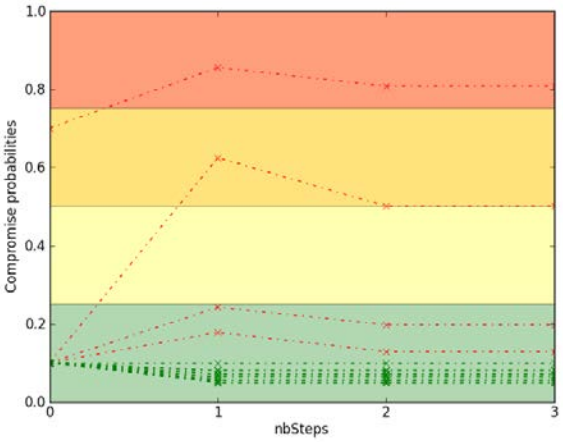
(A) Scenario 1



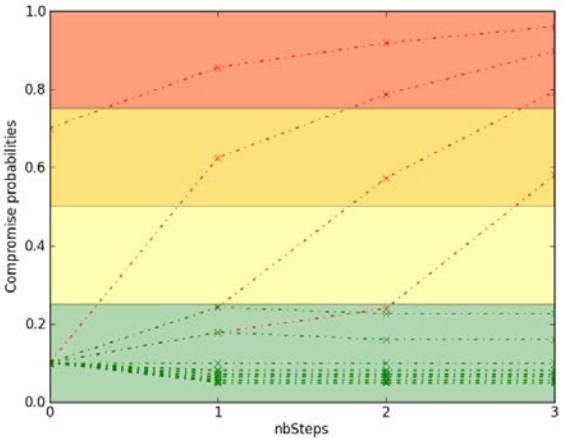
(B) Scenario 2



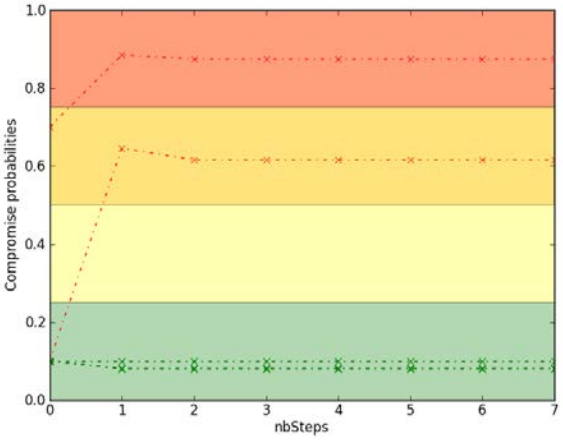
(C) Scenario 4



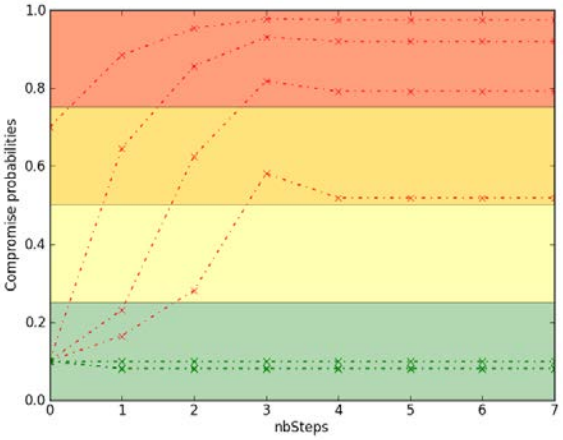
(D) Scenario 6



(E) Scenario 1 (Generic Attack Model of 12 states)



(F) Scenario 3 (Generic Attack Model of 12 states)



5.7.10 Bayesian Attack Model parameter: `probability-new-transition`

The `probability-new-transition` parameter is the probability that the attacker propagates through a new transition. This parameter represents the fact the even if an attack is possible, the attacker may or may not do it. For example, the attacker may have already found what he was looking for.

The value of this parameter is difficult to estimate, so the `probability-new-transition` parameter has to be evaluated on its whole possible variation interval. We thus simulate the variation of this parameter on $[0, 1]$ and analyse its impact on the states compromise probabilities. Figure 5.15 shows the results of this analysis.

From 0 to 0.15, the `probability-new-transition` parameter is very sensitive for `compromised` states. Their compromise probabilities quickly increase with the increase of this parameter. This parameter is little sensitive for `not-compromised` states. Around the 0 value of this parameter, for all scenarios, both `compromised` and `not-compromised` states converge to their initial attack source probability (`probability-attack-source` for all sources, except the actual attack source of the scenario that has a 0.7 probability in these simulations). The compromise probabilities of all `compromised` states increase until they reach a maximum, whose value depends on the certainty of this state to be `compromised` in the related scenario. We can explain this behaviour by the fact that for very low values of the `probability-new-transition` parameter, the attacker has a very little chance to do an attack and pass a transition. The rank between all `compromised` and `not-compromised` states is kept, except for the scenario 6 of Figure 5.15f, for which the two `not-compromised` states of the false positive transition have higher probabilities than the `compromised` state with the smallest probability, for very low value of the `probability-new-transition` parameter.

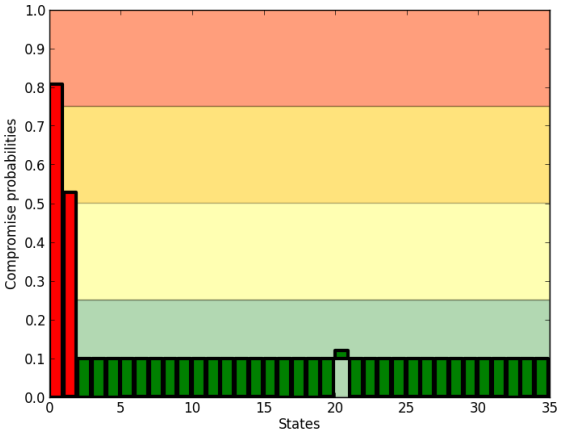
From 0.15 to 0.3, the `probability-unknown-attack` parameter is a little sensitive for both the `compromised` and `not-compromised` states, for nearly all scenarios. Once the maximum of compromised probabilities has been reached, this parameter is near insensitive.

From 0.3 to 0.8, the `probability-new-transition` parameter is sensitive again for `compromised` states. Their compromise probabilities decrease with the increase of this parameter. The `probability-new-transition` parameter is nearly insensitive with `not-compromised` states. As they have reached their maximum, the compromise probabilities of all `compromised` states decrease until they reach their final value around 0. We can explain this behaviour by the fact that for higher values of the `probability-new-transition` parameter, the attacker has many chances to do an attack and pass a transition. Thus, all the transitions after the `compromised` states should also be followed by the attacker. But as their sensors are not activated, the attacker did not do these attacks. Thus the `compromised` states are in fact more probability `not-compromised` and the `alert` sensors of the scenarios are most likely false positives. There is one state whose compromise probability is 0.1 (*i.e.*, the `probability-attack-source` value), for all scenarios, whatever the value of the `probability-new-transition` parameter is. This is a `not-compromised` state that is not the source and destination of any transition. Thus it cannot be used to reach any other state, nor be attacked from a state. As a result, its probability stays at `probability-attack-source`. The rank between all `compromised` and `not-compromised` states is globally kept, except for the `not-compromised` state which probability is constant.

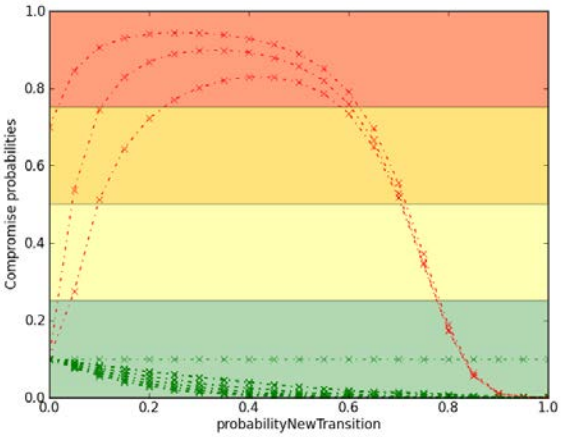
From 0.8 to 1, as the compromise probabilities of all states have reached their minimum, the `probability-new-transition` parameter is insensitive, for both `compromised` and `not-compromised` states.

FIGURE 5.15 – Sensitivity of the `probability-new-transition` parameter from 0 to 1

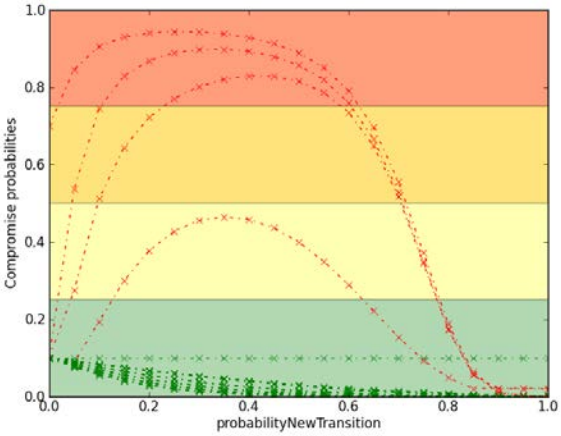
(A) Scenario 1



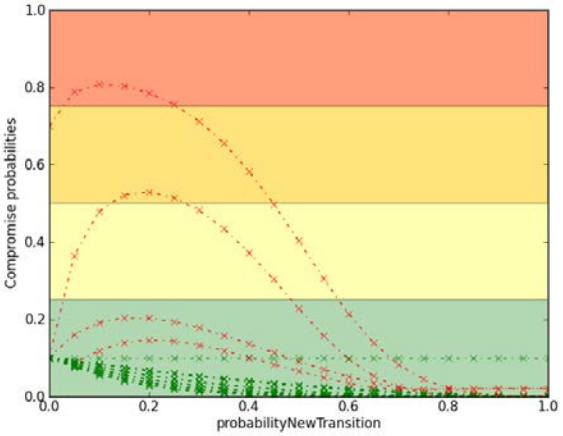
(B) Scenario 2



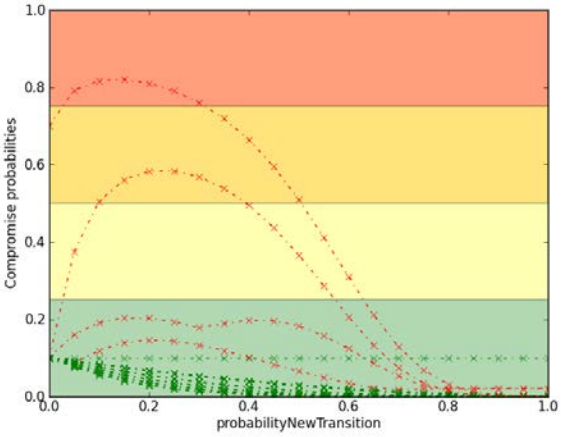
(C) Scenario 3



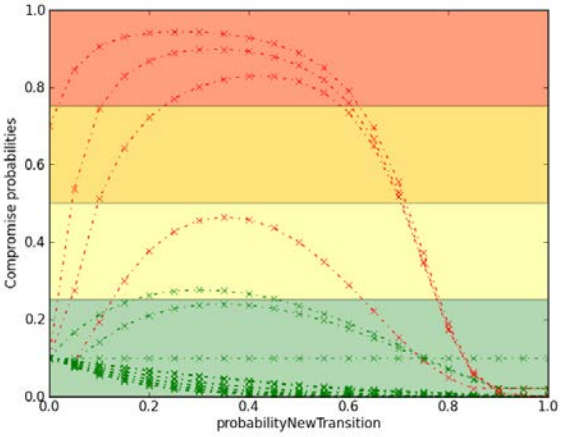
(D) Scenario 4



(E) Scenario 5



(F) Scenario 6



This sensitivity analysis shows that, on its whole possible variation interval, the `probability-new-transition` parameter has an important impact on the compromise probabilities of the `compromised` states of the Bayesian Attack Model. With the increase of this parameter, the compromise probabilities of the `compromised` states increase until they reach a maximum (reached around 0.15), then they decrease. The rank of the states is globally kept.

5.8 Related work

Few people proposed enhancements to improve attack graphs with Bayesian networks, to use them for dynamic risk assessment [QL04, LM05, XLO+10]. However, they do not describe how they manage cycles that are inherent to attack graphs. In [XLO+10], Xie *et al.* present an extension of MulVAL attack graphs using Bayesian networks, but they do not mention how to manage the cycle challenge, while MulVAL attack graphs frequently contain cycles. In the same way, in [FW08], Frigault and Wang do not mention how they deal with the cycle challenge constructing Bayesian attack graphs. In [LM05], Liu and Man assert that to delete cycles, they assume that an attacker will never backtrack. The same assumption is used by Poolsappasit *et al.* in [PDR12]. However, they both do not present how they deal with this assumption to keep all possible paths in the graph, while deleting cycles. We propose here a novel model that explodes cycles in the building process, keeping all possible paths while deleting the cycles, to compute the Bayesian inference.

The Bayesian model presented by Xie *et al.* in [XLO+10] is based on logical attack graphs. It is thus very verbose and can be huge for real information systems. In [LM05], Liu and Man's model is a topological graph, in which are added violation states. It is thus quite compact, but does not detail the attacks, their conditions and, mainly, the sensors that can change state. Thus, the only observations that can be set on this model are observations on topological nodes. The model we present is applicable to a Generic Attack Model that we define formally. So, it is a hybrid vision of logical and topological models which is much more compact than those based on logical attack graphs. It contains the logical conditions necessary to carry out the attacks, in order to keep all information important to model attacks, and add sensor nodes that can be activated with alerts. Moreover, we also add several improvements (attack nodes gathering, polytree structure of Bayesian Attack Tree, *etc.*) that either reduce the size of the graph structure or improve the performance of the inference. We thus constrain the size of the graph in which we do Bayesian inference, while conserving all paths by linearising cycles.

The experimental validation we did on the Bayesian Attack Model is on a simulated models that are far bigger than the state of the art. For example, Xie *et al.* assess their model on a topology of 3 states and 3 vulnerabilities [XLO+10], Liu and Man on a topology of 4 states and 8 vulnerabilities [LM05]. The real world examples used by Frigault and Wang in [FW08] contain at most 8 vulnerabilities on 4 states. The test network used by Poolsappasit *et al.* in [PDR12] contains 8 states in 2 subnets, but with only 13 vulnerabilities. Thanks to our polytree model, we successfully run our Bayesian Attack Model efficiently on simulated topologies with up to 70 states.

5.9 Summary and conclusion

We presented in this chapter a Bayesian Attack Model, representing all the possible attacks in an information system. This model enables dynamic risk assessment. It is built from a Generic Attack Model. Sensor nodes can be activated by dynamic security events to update the compromise probabilities of states, which rank the risk level of ongoing attacks. This model handles the cycles that are inherent to the Generic Attack Model and thus is applicable to any input attack model, with multiple potential attack sources. The cycle breaking process significantly increases the

number of nodes in the model, but thanks to the polytree structure of the Bayesian networks we build, the inference remains efficient, for medium-sized systems.

We also studied precisely the sensitivity of the results of the Bayesian Attack Model, toward the parameters of the Generic Attack Model or the parameters introduced by the Bayesian Attack Model. We summarise in Table 5.5 the results of the sensitivity analysis of all parameters. We also give in this table the range of variation on which we conduct the sensitivity analysis for the given parameters. We present first the Generic Attack Model parameters then, after the double line, the Bayesian Attack Model parameters.

TABLE 5.5 – Sensitivity analysis of the parameters of the Bayesian Attack Model

Name	**Variation range**	**Ranking influence**	**Probability influence**
`false-positive`	$[0-1]$	Low impact (between `compromised` states with `low` probabilities and `not-compromised` states).	Important impact (fast decrease) on `compromised` states on $[0-0.2]$, then almost no impact except for extreme values (> 0.9). No impact for `not-compromised` states except for extreme values (> 0.9).
`false-negative`	$[0-1]$	Low impact (between `compromised` states with `low` probabilities and `not-compromised` states). Almost no impact on its usual variation interval $[0-0.1]$.	Medium impact (decrease) on `compromised` states. Low impact on `not-compromised` states. On its usual variation interval $[0-0.1]$, very low impact.
`probability-attack-source`	$[0-1]$	Impacts ranking when other states have higher source probability than actual attack source.	Significant impact on the probabilities of the `not-compromised` states (increase).
`probability-unknown-attack`	$[0-0.5]$	Impact on the ranking of `compromised` states, but only for high values of the parameter (> 0.35).	Medium impact on both `compromised` and `not-compromised` states. No real impact on its common variation interval ($[0, 0.01]$)
`nbSteps`	$[[0-7]]$	Almost no impact.	Medium impact on the probabilities of the `compromised` states until they converge to their final values. Low impact on the `not-compromised` states.
`probability-new-transition`	$[0.0-1.0]$	Impact on extreme values (close to 0 and 1).	Important impact on the probability of `compromised` states. Low impact on the probability of `not-compromised` states (for both, increase, then decrease).

The most interesting result of this analysis, in order to know where the response priorities are, is the *ranking influence*, which allows the prioritisation of responses. It describes the impact of the variation of each parameter on the rank of state's probabilities (on the whole parameter variation range, for all alert scenarios). This rank will determine the priorities of security operators in their system. The *probability influence* describes the effect of the variation of the parameters on the absolute value of the state's compromise probability. Only three parameters have a significant impact on the rank of the states probabilities: `probability-attack-source`, `probability-unknown-attack`, and `probability-new-transition`. The `probability-attack-source` parameter can be estimated quite accurately with a risk analysis methodology, which gives the security risk of each state, according to its position in the system. The `probability-unknown-attack` parameter has no impact, if it is low (< 0.35), which is a reasonable assumption for a normally protected system. The `probability-new-transition` parameter does not change the ranking on most of its

variation interval, for all scenarios. Moreover, the maximum dispersion of compromise states is reached before this value, around 0.2. Around this value, imprecision will not change the ranking of the compromised states. This is a comforting result, as the compromised state's ranking is not too sensitive to up to medium changes on the inputs parameters, on their usual values. Three parameters have a high impact on the absolute value of the compromise probabilities of states on (at least) a portion of their variation interval: `false-positive`, `probability-attack-source`, and `probability-new-transition`. With a medium uncertainty on such parameters (*e.g.*, 0.01), the variation of the absolute value of the probabilities is medium (*e.g.*, up to 0.1). The other parameters, `false-negative`, `probability-unknown-attack`, and `nbSteps` have a medium impact on absolute values of probabilities. With a medium uncertainty on such parameter (*e.g.*, 0.1), the variation of the absolute value of the probabilities is low (*e.g.*, up to 0.1). So, in the Bayesian Attack Model, absolute value of compromise probabilities may be significantly impacted by uncertainty on parameters, but rank is not impacted by the variation of the parameters, if the uncertainty on the parameters is not too high.

As a result, we have built a dynamic risk assessment model that is suitable for any attack model (either cyclic or acyclic) that can be specified within the Generic Attack Model. It is efficient for up to medium-sized models. Its results are relatively accurate for the rank of the compromise probabilities of states, even with uncertainty on the parameters. However, in order to have exact absolute values on the compromise probabilities, the values of the parameters have to be accurately chosen. One other drawback of this model is that it mixes the increase of compromise probabilities due to already occurred attacks and likely futures. As a result, this model is not fully suited to support the computation of responses.

Chapter 6

Hybrid Risk Assessment Model

We presented in the previous chapter a Bayesian Attack Model that can be used for dynamic risk assessment. The Bayesian Attack Model is built from a Generic Attack Model and contains all the attacks that can happen in a system. Its sensor nodes can be activated by dynamic security events to update the compromise probabilities of states. These probabilities allow to rank the risk of ongoing attacks. However, the Bayesian Attack Model suffers from three main limitations.

- It can be used only for small and medium systems.
- It does not take into account the order in which the alerts are received. In the Bayesian Attack Model, all the sensors that have raised alerts are activated in the model, but without taking into account the order in which they have been received.
- For each state, the output of the Bayesian Attack Model is a unique value cumulating its probability of being already compromised and being compromised in a near future, according to the received alerts. There is no easy way in the results of the Bayesian Attack Model to make a distinction between these two reasons of being compromised.

The model we propose in this chapter is a new hybrid model combining attack graphs and Bayesian networks for dynamic risk assessment. It is an extension of the Bayesian Attack Model. This model is subdivided into two complementary models: (1) The Dynamic Risk Correlation Models correlate a chain of alerts with the knowledge on the system to analyse ongoing attacks and provide the probabilities of hosts being compromised, (2) The Future Risk Assessment Models take into account existing vulnerabilities and the current attack status to assess which potential attacks are most likely to occur in the next future. Dynamic Risk Correlation Models aim at threat likelihood assessment, identifying where the attack comes from. It outputs probabilities that attacks are completed and that assets of the information system are compromised. These probabilities provide security operators with the capability to manage priorities according to the likelihood of ongoing attacks. Future Risk Assessment Models aim at threat mitigation, identifying the most likely and impacting next steps for the attacker. This problem separation provides a significant performance improvement in terms of the number of nodes, enabling scalability.

This chapter is part of the contribution described in subsection 1.3.1, the definition of a dynamic risk assessment model, and the contribution described in subsection 1.3.2, the scalability of the risk assessment model. It tackles the challenges described in section 1.2.1, 1.2.2 and 1.2.3: the methodology to build a dynamic risk assessment model merging different information sources and taking advantage of the different types of attack representation and which is scalable.

6.1 Hybrid Risk Assessment Model architecture

The Hybrid Risk Assessment Model (HRAM) takes the same inputs as the Bayesian Attack Model presented in chapter 5: a Generic Attack Model and a set of alerts.

Determining the likelihood of occurrence of occurring attacks can be separated into two distinct problems: (1) Threat impact assessment, identifying where the attack comes from and (2) Threat mitigation, identifying the most likely next steps for the attacker. Each problem has its dedicated models, Dynamic Risk Correlation Models (DRCMs) and Future Risk Assessment Models (FRAMs), combined to provide a complete Hybrid Risk Assessment Model. The architecture of the Hybrid Risk Assessment Model is presented in Figure 6.1. As for the Bayesian Attack Model, we take as input a Generic Attack Model. First, we build Dynamic Risk Correlation Models from

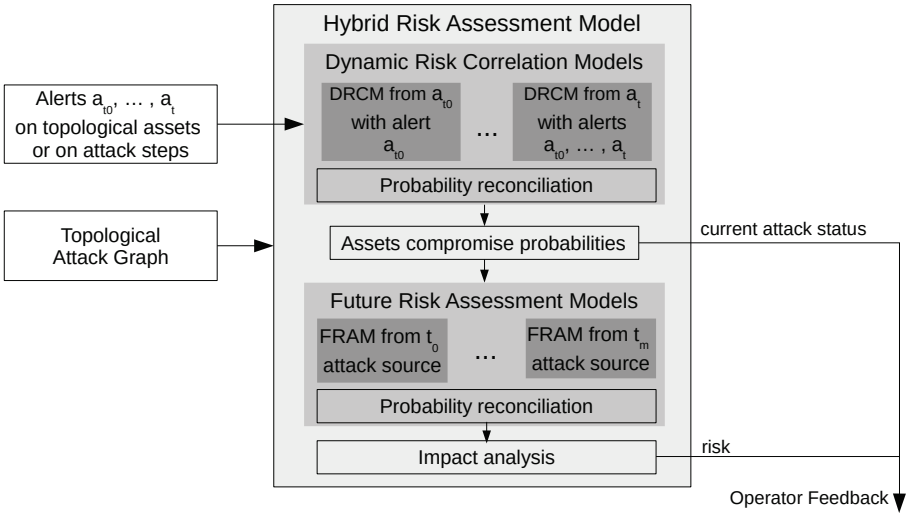


Figure 6.1 – Hybrid Risk Assessment Model Architecture

6.2 Dynamic Risk Correlation Model

We will first present in this section the details of Dynamic Risk Correlation Models: their building process, usage, structure, and probability tables.

6.2.1 Building process and usage

The goal of the Dynamic Risk Correlation Model is to provide explanations for correlated alerts that have been raised by sensors in a system. By *explanation*, we mean the identification of the likely source nodes that have been compromised and that have enabled the attacker to launch the detected attack. A Dynamic Risk Correlation Model is built from the most recent alert received,

the *target*, and explains why this alert has been generated, taking into account past alerts. As soon as a new alert is received, a new Dynamic Risk Correlation Model is built. Older Dynamic Risk Correlation Models are kept in parallel with the newly generated DRCM, to manage scenarios with several distinct simultaneous attacks (a new alert is not related to older ones) as detailed at the end of subsubsection 6.2.5.2. Probabilities of all kept Dynamic Risk Correlation Models are consolidated as detailed in subsection 6.2.4.

We build the structure of the Dynamic Risk Correlation Models according to the Generic Attack Model and the received alerts. Then, we set the states of the DRCM Sensor Nodes according to the previous security alerts received by the sensors:

- If the sensor of a transition, state or condition exists and is deployed in the network, if it has not issued any alert, all related DRCM Sensor Nodes are set to the `no-alert` state.
- If the sensor has raised an alert corresponding to this transition, state or condition, the related DRCM Sensor Nodes are set to the `alert` state.
- If the transition, state or condition has no deployed sensor, there is `no-info` about this sensor. So, the related DRCM Sensor Nodes cannot be set in any state and their probabilities are updated by the Bayesian inference.

Note that if sensors give an alert confidence probability, it is possible to set the state `alert` to this probability.

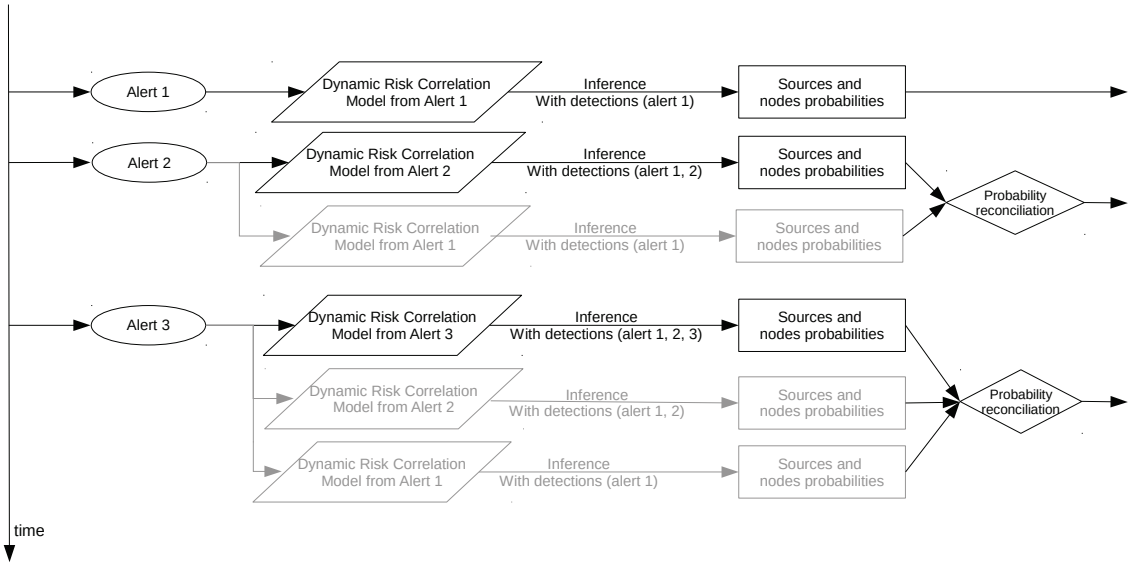
Finally, we use a Bayesian network belief propagation algorithm (Lauritzen or Pearl's) to update the probabilities of each state at all the nodes. The reconciliation of the probability the Dynamic Risk Correlation Models gives the compromise probabilities of the states of the system, according to the received alerts.

The building process of the Dynamic Risk Correlation Model according to 3 successive alerts received by the system is shown in Figure 6.2. When the first alert is received, a Dynamic Risk Correlation Model is built to detail the possible causes of this alert. The sensor(s) corresponding to this alert in the Dynamic Risk Correlation Model are set to the `alert` state and Bayesian inference is done. It gives the compromise probabilities of the states of the system, after the first alert. Then, the alert 2 is received. A new Dynamic Risk Correlation Model is built to detail the possible causes of the second alert. The sensor(s) in the new Dynamic Risk Correlation Model corresponding to the alerts 1 and 2 are set to the `alert` state. If alert 1 is not in this new DRCM, the two alerts are not yet corresponding to the same attack. Thus, the Dynamic Risk Correlation Model is kept in parallel to the new one, to take into account both unrelated attacks. The reconciliation of the probability of both Dynamic Risk Correlation Model gives the compromise probabilities of the states of the system, after the two alerts. Finally, when the system receives alert 3, we build a third Dynamic Risk Correlation Model in a similar way and set the sensor(s) corresponding to the alerts 1, 2 and 3 to the `alert` state. If the past alert(s) are not related to alert 3, we keep the corresponding DRCM(s) in parallel to the newly built one. The reconciliation of the probability the Dynamic Risk Correlation Models gives the compromise probabilities of the states of the system, after the three alerts.

6.2.2 Representation of a transition

A transition in the Generic Attack Model is an edge from a state to another, associated with several conditions and can be related to a sensor. In the Dynamic Risk Correlation Model, similarly to the Bayesian Attack Model, we detail the states, transitions, conditions, and their sensors as nodes, in order to model the probabilistic interactions between such elements, using the nodes detailed below. Each node represents a boolean random variable with two mutually exclusive states.

FIGURE 6.2 – Building of the Dynamic Risk Correlation Model according to the alerts received



Definition 23 *The **DRCM Target Node*** drcm-tan(s) *or* drcm-tan(t)*, with* $s \in S$ *or* $t \in T$ *(cf. Def. 7), is the node of the Dynamic Risk Correlation Model describing the state of compromise of* s *or* t*, the target of the DRCM. This node has two mutually exclusive states:* `compromised` *and* `not-compromised`*.*

The target of the Dynamic Risk Correlation Model corresponds to the element, either state or transition, concerned by the lastly received alert in the system.

Definition 24 *A **DRCM State Node*** drcm-stn($s_n, \cdots, s_1$)*, with* $\forall i, s_i \in S$ *(cf. Def. 7), is a node of the Dynamic Risk Correlation Model representing the random variable describing the state of compromise of state* s_1*, to compromise the target of the DRCM* s_n *using the path of the Generic Attack Model* $s_n \leftarrow \cdots \leftarrow s_1$*. This node has two mutually exclusive states:* `compromised` *and* `not-compromised`*.*

Definition 25 *A **DRCM Attack Source Node*** drcm-ason(s)*, with* $s \in S$ *(cf. Def. 7), is a node of the Dynamic Risk Correlation Model representing the random variable describing that the source of attack is* s*. This node has two mutually exclusive states:* `source` *and* `not-source-of-attack`*.*

Definition 26 *A **DRCM Transition Node*** drcm-tn(t)*, with* $t \in T$ *(cf. Def. 7), is a node of the Dynamic Risk Correlation Model representing the random variable describing the attack success of transition* t*. This node has two mutually exclusive states:* `succeeded` *and* `failed`*.*

Definition 27 *A **DRCM Condition Node*** drcm-cn(c)*, with* c *a condition (cf. Def. 8), is a node of the Dynamic Risk Correlation Model representing the random variable describing that the condition* c *is fulfilled. This node is a boolean variable with two mutually exclusive states:* `verified` *and* `not-verified`*.*

Definition 28 *A **DRCM Sensor Node*** drcm-sn(s)*, with* s *a sensor (cf. Def. 9), is a node of the DRCM representing the random variable describing the state of the sensor* s*. This node is a boolean variable with two mutually exclusive states:* `alert` *and* `no-alert`*.*

These nodes are linked with edges, indicating that the child node has a conditional dependency to the state of its parents.

Definition 29 *A **Dynamic Risk Correlation Model edge*** drcm-e*, is a link from a parent node to a child node in the DRCM, representing a conditional dependency of the child toward its parent.*

6.2.3 Complete model

Definition 30 *A **Dynamic Risk Correlation Model** is a Bayesian network represented by* DRCM(drcm-tan, DAG,P) *where:*

- *drcm − tan(cf. Def. 23) is the DRCM Target Node, representing the element impacted by the lastly generated alert.*
- DAG(DRCMN,E) *is a polytree structure, constituted of*
  - *DRCMN, the DRCM nodes* DRCMN=[drcm-stn],[drcm-ason],[drcm-tn],[drcm-cn], [drcm-sn] *(cf. Defs. 24-28)*
  - *E, the set of edges* $E = \{$drcm-e$\}$ *representing a conditional dependency between the nodes (cf. Def. 29).*
- P *is a set of local probability distributions, associated with each node of* DAG*. As all nodes are discrete random variables, the local probability distributions can be specified within a conditional probability table.*

Following the process described in subsection 5.1.1, we construct each Dynamic Risk Correlation Model in such a way as not to have any cycles, but to keep all possible attack paths "directed to" the target. The Dynamic Risk Correlation Model is built from the lastly received alert. Then, we recursively add the transitions and states allowing to compromise the target. We store in each DRCM State Node the path, from this state to the target of the Dynamic Risk Correlation Model. This allows to ensure that the building process never comes back on a previously exploited node and thus the Dynamic Risk Correlation Model does not have cycles, but contains all possible causes of the latest received alert.

Moreover, we design this building process in order to generate a graph structure of the Dynamic Risk Correlation Model which is a polytree (*i.e.*, directed graph with no directed nor undirected cycles). This implies, for example, to duplicate the condition and sensor nodes (*i.e.*, new conditions and sensors for each added attack step). The Dynamic Risk Correlation Model being a polytree satisfies the requirements of Pearl's inference algorithm [Pea88], which is quasilinear in the number of nodes. Thus, the inference in such a Dynamic Risk Correlation Model with a polytree structure containing duplicated nodes is more efficient and consume less memory, in comparison with a directed acyclic graph structure with fewer nodes (no duplicates), for identical results.

Figure 6.3 shows an example of a Dynamic Risk Correlation Model built from an alert on state s_1 (the node in dotted lines on the left) from a Generic Attack Model of 3 states (s_1, s_2 and s_3). DRCM State Nodes are represented by a rectangle shape, DRCM Attack Source Nodes by a five-sided shape, DRCM Transition Nodes by a diamond shape, and DRCM Condition Nodes by an oval shape.

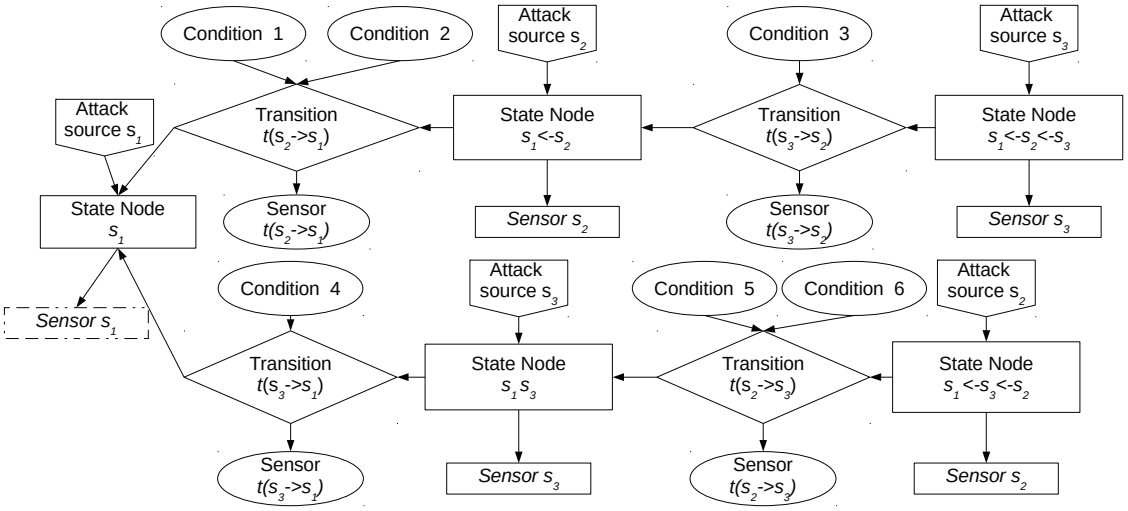


FIGURE 6.3 – Dynamic Risk Correlation Model

6.2.4 Reconciliation of probabilities

The outputs of the Dynamic Risk Correlation Models are of two types: (1) the probabilities of attack sources, describing how likely an asset is to be the source of the attack impacting the target of the Dynamic Risk Correlation Model, and (2) the compromise probabilities, describing how likely it is that an asset has been compromised along the path of the attacker.

As a DRCM State Node contains the attack path from its related state to the target, many DRCM State Nodes represent the same Generic Attack Model's state. Indeed, the attacker can potentially use several different paths to reach the target: for example, $s_1 \leftarrow s_2 \leftarrow s_4$ is different from $s_1 \leftarrow s_3 \leftarrow s_4$, but in both cases, the attacker starts from the same state s_4 to attack the target s_1. In a Dynamic Risk Correlation Model $DRCM_i$, we thus have many DRCM State Nodes and DRCM Source Nodes representing the same state s. We chose to give the operator the worst case for compromise probability of states. Thus, as output of a Dynamic Risk Correlation Model, we assign to a state:

- a probability of compromise Pc that is the maximum of the probabilities of DRCM State Nodes related to this asset:

$$Pc_{DRCM_i}(s) = max_{node \in \{\text{DRCM States}(s)\}} Pc_{DRCM_i}(node) \tag{6.1}$$

- an attack source probability Ps that is the maximum of the probabilities of DRCM Source Nodes related to this asset:

$$Ps_{DRCM_i}(s) = max_{node \in \{\text{DRCM Attack Source}(s)\}} Ps_{DRCM_i}(node) \tag{6.2}$$

On the other hand, as described in subsection 6.2.1, several older Dynamic Risk Correlation Models are kept in parallel with the new ones generated each time a new alert is received. They can be related to different simultaneous attacks. These Dynamic Risk Correlation Models $DRCM_i$ give different sources and compromise probabilities for the same state s. Thus, the second level of probability reconciliation is done between all kept Dynamic Risk Correlation Models. Similarly to the single Dynamic Risk Correlation Model case, we want to present the operator with a view of the worst case, and assign to a state s a probability of compromise $Pc(s)$ and an attack

source probability $Ps(s)$ that is the maximum of the related probabilities in all the Dynamic Risk Correlation Models

$$Pc(s) = max_{DRCM_i \in \{\text{kept DRCMs}\}} Pc_{DRCM_i}(s) \tag{6.3}$$

$$Ps(s) = max_{DRCM_i \in \{\text{kept DRCMs}\}} Ps_{DRCM_i}(s) \tag{6.4}$$

So, in order to reconcile the different probabilities that we have for a state, with a Dynamic Risk Correlation Model or between Dynamic Risk Correlation Models we chose to keep the maximum of all probabilities relative to this state.

6.2.5 Pruning

The main limitation when implementing the Dynamic Risk Correlation Model is the combinatorial explosion of the number of nodes, due to the cycle breaking process. This process introduces a lot of redundancy which increases significantly the size of the model. In order to improve the performance and prevent this combinatorial explosion, we implement two types of pruning: the pruning within a Dynamic Risk Correlation Model and the pruning between Dynamic Risk Correlation Models.

6.2.5.1 Pruning within a Dynamic Risk Correlation Model

Within a Dynamic Risk Correlation Model, we provide a practical way to cut marginally relevant attack paths (paths with extremely low probabilities) while preserving the other paths.

The probabilities of DRCM State Nodes represent the probability of the attacker having exploited the DRCM Target, *by exploiting this state*. As long as no alert has been sent by the sensors located along this path, the probability of a node being compromised decreases rapidly as a function of the length of the path between the DRCM State Node and the DRCM Target. Moreover, thanks to the multiple Dynamic Risk Correlation Models that are kept, if a detected transition is discarded in a Dynamic Risk Correlation Model, it will be in another older Dynamic Risk Correlation Model, closer to its detected nodes, thanks to the redundancy of the model.

According to the state of the sensors along a path, we have different pruning policies, summarised in Figure 6.4. The rules applied when building a Dynamic Risk Correlation Model are the following:

- We keep exploring and memorising the path from the target asset, as long as we find `alert` sensors.
- For `no-alert` sensors, when there are more than `max-number-no-alert-to-explore`, we discard the path and keep only `max-number-no-alert-to-keep` nodes.
- For `no-info` sensors, when there are more than `max-number-no-info-to-explore`, we discard the path and keep only `max-number-no-info-to-keep` nodes, but with values for the parameters bigger than for `no-alert` sensors.
- As soon as an `alert` sensor is found on an explored path, the counters of `no-alert` and `no-info` are reset to 0.

6.2.5.2 Selection of the Dynamic Risk Correlation Models to keep

The last important feature about this model is the selection of the Dynamic Risk Correlation Models to keep. Indeed, as one new Dynamic Risk Correlation Model is generated each time an

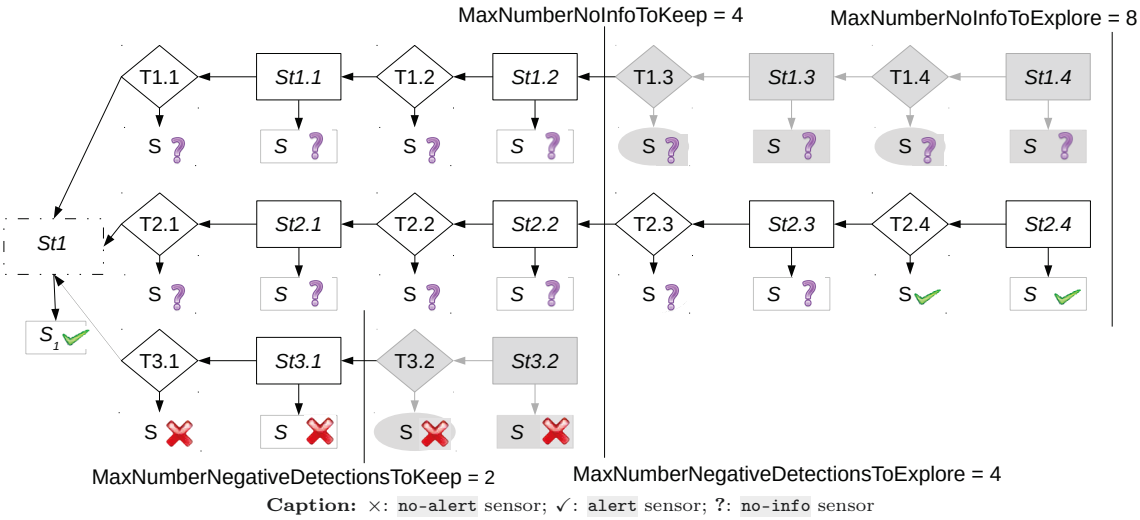


FIGURE 6.4 – Pruning policies in Dynamic Risk Correlation Model

alert is received, the number of models to keep can increase quickly. However, when alerts are part of the same attack, they will be part of the Dynamic Risk Correlation Model whose target is the sensor that raised the latest alert. Moreover, there will be more `alert` sensors in this Dynamic Risk Correlation Model (increasing probabilities of all nodes of the DRCM), and at most the same number of `no-alert` sensors (decreasing probabilities) than in all previous Dynamic Risk Correlation Models related to the same attack. Thus, all the previous Dynamic Risk Correlation Models related to the same attack are irrelevant, because they are included in the lastly generated Dynamic Risk Correlation Model. Their probabilities will be lower so they do not change the maximum of state compromise probabilities.

The only different Dynamic Risk Correlation Models that are useful to keep are those not related to the same attacks because they bring new information about the currently happening attacks. They can be identified by having at least one DRCM State Node with a higher probability than all the ones of the latest Dynamic Risk Correlation Model, for the same state. These attacks could be part of a more global attack scenario, starting from different sources, that has not yet converged or it might be distinct attacks that are happening simultaneously. That is why we may need to keep several Dynamic Risk Correlation Models in parallel.

6.2.6 Conditional probability tables

We now specify using conditional probability tables the local probability distribution associated with each node, describing the probability dependencies of a node toward his parents. As most nodes of the Dynamic Risk Correlation Model are very similar to the ones of the Bayesian Attack Model and thus have the same conditional probability table, we will specify here only the nodes which have different probability tables, compared to the Bayesian Attack Model.

For example, the DRCM sensor nodes have exactly the same conditional probability table as Bayesian sensor nodes, as presented in section 5.3. The DRCM condition nodes have the same probability value as Bayesian condition nodes, as presented in Table 5.3. The DRCM attack sources have the same probability value as Bayesian attack sources, as presented in Table 5.3.

6.3 Future Risk Assessment Model

We will now present the second family of models of the Hybrid Risk Assessment Model: Future Risk Assessment Models (FRAMs).

6.3.1 Building process and usage

The goal of a Future Risk Assessment Model is to evaluate among all possible futures, the ones that are the most likely to happen. As indicated by Figure 6.1, a Future Risk Assessment Model is built from each attack source, according to the Dynamic Risk Correlation Models' reconciled compromise probabilities. Then, we use a belief propagation algorithm to update the probabilities of all the nodes. When there is a completed attack transition or a compromised state, a Future Risk Assessment Model taking this node as starting point is built or updated and the branches from this attack step are deleted in all other Future Risk Assessment Models. Indeed, this attack is no longer a possible future, as it has happened and will be investigated in its own Future Risk Assessment Model. Even if the structure of a Future Risk Assessment Model does not change with alerts, its probabilities of conditions and attack sources can be updated. For example, the a-priori condition probability of a vulnerability that has already been exploited is set to "1".

We build recursively the structure of each Future Risk Assessment Model according to the Generic Attack Model, starting from the attack source. Then, we use a Bayesian network belief propagation algorithm to update the probabilities of each state at all the nodes. There is no need to set evidences in Future Risk Assessment Models because it represents hypotheses in the future, which, by definition, may only be observed later.

6.3.2 Representation of a transition

The representation of a transition in the Future Risk Assessment Model is very similar to the one of the Bayesian Attack Model. But, as this model is only used to predict possible futures, there are no sensor nodes. In the same way, each node represents a boolean random variable with two mutually exclusive states.

Definition 31 *The **FRAM Attack Source Node*** fram-ason(s)*, with* $\mathrm{s} \in \mathrm{S}$ *(cf. Def. 7), is a node of the Future Risk Assessment Model representing a potential source of attack* s *whose possible futures need to be evaluated. This node has two mutually exclusive states:* `compromised` *and* `not-compromised`*.*

Definition 32 *A **FRAM State node*** fram-stn$(\mathrm{s}_1, \cdots, \mathrm{s}_n)$*, with* $\forall i, \mathrm{s}_i \in \mathrm{S}$ *(cf. Def. 7), is a node of the Future Risk Assessment Model representing the random variable describing the state of compromise of state* s_n *using the path of the Generic Attack Model* $\mathrm{s}_1 \rightarrow \cdots \rightarrow \mathrm{s}_n$*. This node has two mutually exclusive states:* `compromised` *and* `not-compromised`*.*

Definition 33 *A **FRAM Transition node*** fram-tn(t)*, with* $\mathrm{t} \in \mathrm{T}$ *(cf. Def. 7), is a node of the Future Risk Assessment Model representing the random variable describing the attack success of transition* t*. This node has two mutually exclusive states:* `succeeded` *and* `failed`*.*

Definition 34 *A **FRAM condition node*** fram-cn(c)*, with* c *a condition (cf. Def. 38), is a node of the Future Risk Assessment Model representing the random variable describing that the condition* c *is fulfilled. This node has two mutually exclusive states:* `succeeded` *and* `failed`*.*

These nodes are linked with edges, indicating that the child node has a conditional dependency to the state of its parents.

Definition 35 *A **FRAM edge*** fram-e*, is a link from a parent node to a child node in the Future Risk Assessment Model, representing a conditional dependency of the child toward its parent.*

6.3.3 Complete model

Definition 36 *A **Future Risk Assessment Model** is a Bayesian network represented by* FRAM(fram-ason, DAG,P) *where:*

- $fram-ason$ *is the FRAM Source Node, a potential source of attack whose possible futures need to be evaluated.*
- DAG(FRAMN,E) *is a polytree structure, constituted of*
  - $FRAMN$*, the FRAM nodes* FRAMN=[fram-ason],[fram-stn],[fram-tn],[fram-cn] *(cf. Defs. 31-34)*
  - E*, the set of edges* E = {fram-e} *representing a conditional dependency between the nodes (cf. Def. 35).*
- P *is a set of local probability distributions, associated with each node of* DAG.

We build the Future Risk Assessment Model similarly to the Bayesian Attack Model, in such a way as not to have any cycle. Moreover, we design the graph structure of the Future Risk Assessment Model in order to be a polytree. This allows to use very efficient exact inference algorithms such as Pearl's algorithm [Pea88].

Figure 6.5 shows an example of a Future Risk Assessment Model issuing from state s_1 from a Generic Attack Model with 3 states. The FRAM Attack Source is represented by a five-sided

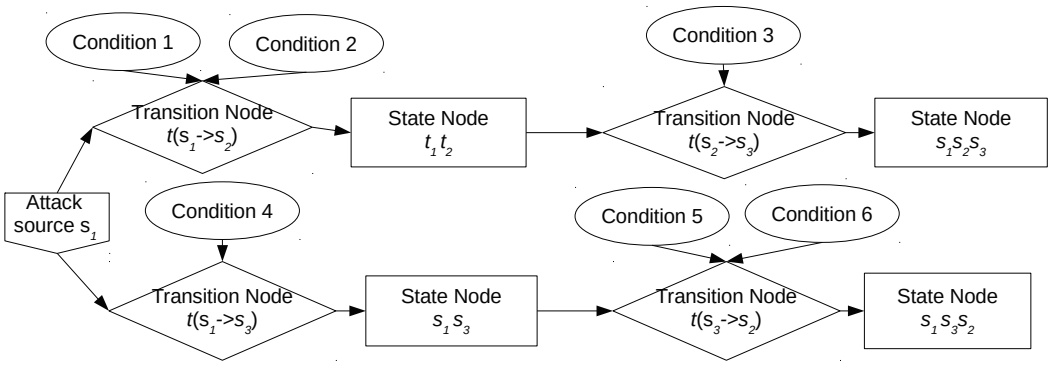


FIGURE 6.5 – Future Risk Assessment Model

6.3.4 Reconciliation of probabilities

The outputs of a Future Risk Assessment Model are the compromise probabilities, describing how likely it is that a state will be compromised in the near future, by an attacker, according to the status of compromise of states in the system.

However, as a FRAM State Node contains the attack path from the FRAM Source Node to this node, similarly to the Dynamic Risk Correlation Model, many FRAM State Nodes can represent the same state, when the attacker used a different path. In a Future Risk Assessment Model, we have many FRAM State Nodes representing the same state of the information system. As output of the reconciliation of probabilities, we chose to give to the operator the worst case, for the probability of the states being compromised in the near future. Thus, as output of a Future Risk Assessment Model, we assign to a state a probability of compromise in the future that is the maximum of the probabilities of FRAM State Nodes targeting the same state.

6.3.5 Pruning in a Future Risk Assessment Model

As a Future Risk Assessment Model does not include any evidence (*i.e.*, it does not contain sensor nodes which are set in a specific state), the Bayesian inference is much easier to compute. Moreover, it has fewer nodes than a Dynamic Risk Correlation Model, thus its combinatorial explosion of nodes is not as important. However, the combinatorial explosion of the number of nodes, due to the cycle breaking process is still present. If they are not built carefully, there is a lot of redundancy in all Future Risk Assessment Models, which increases significantly the size of the model. A major part of this redundancy can be deleted, at the building of a new Future Risk Assessment Model, by deleting all the paths issued from all sources of other Future Risk Assessment Models. All these subtrees can be deleted safely, as their probabilities will be lower than the probabilities of the Future Risk Assessment Model issued from the attack source. Moreover, as we only want to predict the near future, we can limit to a small number of steps, `nbSteps-possible-futures`, (*e.g.*, 3) the next steps to compute.

It would also be possible to add a sounder pruning by keeping the nodes (and the rest of the path) only if their compromise probability is above a certain probability (*e.g.*, probability above 0.01). Indeed, as the Future Risk Assessment Models do not have evidence and have a polytree structure, the propagation of probabilities is a forward propagation and the probabilities are decreasing with the depth in the polytree. It is thus possible to know when computing the probabilities the maximum compromise probabilities of the children of the current node. However, this would require to customise the inference algorithm.

6.3.6 Conditional probability tables

We now specify using conditional probability tables the local probability distribution associated with each node, describing the probability dependencies of a node toward his parents. As the nodes of the Future Risk Assessment Model are very similar to the ones of the Bayesian Attack Model, they have the same conditional probability tables.

For example, the FRAM Source Node has the same probability value as Bayesian Source Nodes, as presented in Table 5.3. The FRAM Condition Nodes have the same probability value as Bayesian Condition Nodes, as presented in Table 5.3.

6.4 Impact analysis

We use exactly the same simple impact analysis component in the Hybrid Risk Assessment Model to the one of the Bayesian Attack Model, that has been already presented in section 5.4

6.5 Hybrid Risk Assessment Model complexity evaluation

The complexity of the Hybrid Risk Assessment Model depends on the complexity of its underlying models: Dynamic Risk Correlation Models and Future Risk Assessment Models. Thus, we study successively the complexity of both types of models.

6.5.1 Dynamic Risk Correlation Model

As in the Bayesian Attack Model, the main computation done on each Dynamic Risk Correlation Model is the execution of the belief propagation algorithm (probability inference), computing the probability of all nodes, according to evidences, nodes set to a specific state. The complexity of the inference in a Bayesian network is directly linked to the number of nodes and structure of the network. We estimate the number of nodes M of a Dynamic Risk Correlation Model, depending on $|S|$, the number of state nodes in the Generic Attack Model, k the maximum number of *consolidated* transitions between two states in the Generic Attack Model (*i.e.*, the maximum number of different types of transitions), o the maximum number of alerts received until the generation of the Dynamic Risk Correlation Model, and the pruning parameters presented in Figure 6.4. For this complexity evaluation, we consider the worst case: there are k transitions between each pair of states.

Let $maxPruning = max($`max-number-no-alert-to-keep`, `max-number-no-info-to-keep`$)$. In each branch, the sensors of transitions and states will be either in the `alert` state (maximum o times), or in the `no-alert` or `no-info` state (maximum $maxPruning$ times). Thus, each branch is at maximum $maxLength = maxPruning + o$ long. As in Bayesian Attack Model, for each transition, we add ≈ 5 nodes to the Dynamic Risk Correlation Model (sometimes a few more, according to the number of conditions). Thus, in the worst case, for each Dynamic Risk Correlation Model, targeting a transition or state, the number of nodes to add M is

$$\begin{aligned} M \sim 5 \times k \times (|S| \times \cdots \times (|S| - maxLength - 1)) &= 4 \times k \times \frac{|S|!}{(|S| - maxLength)!} \\ &= \mathcal{O}(|S|^{maxLength}) = \mathcal{O}(|S|^{maxPruning+o}) \end{aligned}$$

However, even if the number of nodes in each $DRCM$ is high, the Bayesian inference can be done efficiently. Indeed, as the structure is a *polytree*, efficient inference algorithms such Pearl's belief propagation algorithm can be used. The complexity of the inference in a Dynamic Risk Correlation Model $\mathcal{C}(DRCM)$ is thus:

$$\mathcal{C}(DRCM) = |S|.\mathcal{O}(|S|^{maxPruning+o}) = \mathcal{O}(|S|^{maxPruning+o+1}) \tag{6.5}$$

We keep several Dynamic Risk Correlation Models in parallel, to take into account when there are alerts that are part of different attacks. As a result, there are at most o Dynamic Risk Correlation Models in parallel. We build a new model each time the system receives a new correlated alert. Older models may be kept in memory, but they are not processed anymore and no inference is done. We only compare the compromise probabilities of the $|S|$ states with the compromise probabilities of the newly generated DRCM, to assess whether or not we keep older Dynamic Risk Correlation Models.

So, for each new alert received, the total of computations $\mathcal{C}$ done for all Dynamic Risk Correlation Models is:

$$\mathcal{C} = \mathcal{C}(DRCM) + |S| \times o = \mathcal{O}(|S|^{maxPruning+o+1}) \tag{6.6}$$

Even if this algorithmic complexity seems prohibitive, it is a worst-case analysis and, in practice, it is quite efficient, even for big systems, as detailed in section 6.6.

6.5.2 Future Risk Assessment Model

Computations on Future Risk Assessment Models are much simpler than those on Dynamic Risk Correlation Models. Indeed, these models have a limited number of nodes, and the depth of next futures to predict can be small (*e.g.*, 3 is a good value).

In the same way, we estimate the number of nodes M of a DRAM, depending on $|S|$, the number of state nodes in the Generic Attack Model, and k the maximum number of *consolidated* transitions between two states in the Generic Attack Model. For this complexity evaluation, we consider the worst case: there are k transitions between each pair of states. For each transition, we add ≈ 3 nodes to the Future Risk Assessment Model (sometimes few more, according to the number of conditions). Thus, in the worst case, for each Future Risk Assessment Model, starting from a potential attack source, the number of nodes to add M is

$$M \sim 3 \times k \times (|S| \times \cdots \times (|S| - nbFutureSteps - 1)) \\ = 4 \times k \times \frac{|S|!}{(|S| - nbFutureSteps)!} = \mathcal{O}(|S|^{nbFutureSteps})$$

Thus, for each Future Risk Assessment Model, in the worst case, the complexity of the construction and probability inference $\mathcal{C}(FRAM)$ is $\mathcal{C}(FRAM) = \mathcal{O}(|S|^{nbFutureSteps})$. Finally, for the whole Future Risk Assessment Models, as there are at most $|S|$ attack sources, in the worst case, the complexity of the inference in the whole model $\mathcal{C}(FRAMs)$ is

$$\mathcal{C}(FRAMs) = |S|.\mathcal{C}(FRAM) = \mathcal{O}(|S|^{nbFutureSteps+1}) \tag{6.7}$$

The calculations on each $FRAM$ are independent. So, they may be easily done in parallel, which gives in practice, $\mathcal{C}(FRAMs) = \mathcal{O}(|S|^{nbFutureSteps})$ with $|S|$ processors.

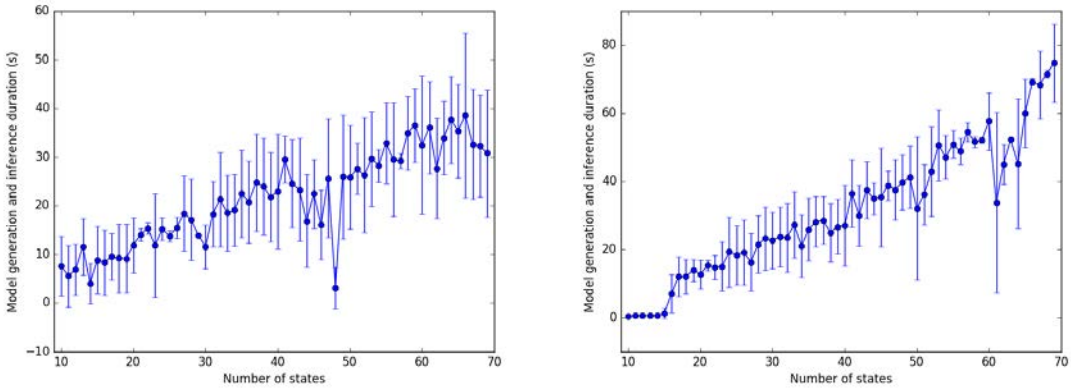
6.6 Hybrid Risk Assessment Model performance evaluation

Similarly to the Bayesian Attack Model, the Hybrid Risk Assessment Model has to be evaluated each time a correlated alert, or a set of correlated alerts is received: new sensor nodes are set in their new states, then the probabilities are updated. Such a process must be quite fast (less than 5 minutes is good), for the operator to properly understand the risk in operational time. We evaluate the Hybrid Risk Assessment Model on the simulated random Generic Attack Models presented in section 5.6.

The results of the duration in seconds of the Hybrid Risk Assessment Model generation and the inference after the evaluation of one scenario of four successive transitions, on the generated Generic Attack Model, is displayed in Figure 6.6. The parameters of the Hybrid Risk Assessment Model are in the default values detailed in section 6.7. These simulations show that for medium-sized topologies (up to 70 states), even quite strongly connected (up to 10 clusters), the duration of the Hybrid Risk Assessment Model generation and of the inference remains acceptable ($<$ 1 minutes 30 seconds).

FIGURE 6.6 – Duration in seconds of Hybrid Risk Assessment Model execution, according to the number of states and clusters in the Generic Attack Model

(A) 70 clusters (B) 10 clusters



6.7 Parameter sensitivity analysis

As the Bayesian Attack Model, we build the Hybrid Risk Assessment Model from an Generic Attack Model. The Hybrid Risk Assessment Model also introduces new parameters. Thus, in this section, we study the sensitivity of the Hybrid Risk Assessment Model toward the Generic Attack Model and the Hybrid Risk Assessment Model's parameters.

6.7.1 Generic Attack Model and Hybrid Risk Assessment Model parameters summary

In chapter 5, Table 5.2 summarises the four parameters of the Generic Attack Model: `false-positive`, `false-negative`, `probability-attack-source` and `probability-unknown-attack`.

Table 6.1 summarises the four parameters introduced by the Dynamic Risk Correlation Model of the Hybrid Risk Assessment Model: `max-number-no-info-to-explore`, `max-number-no-alert-to-explore`, `max-number-no-info-to-keep` and `max-number-no-alert-to-keep`. Each parameter is associated with its description and the default value used in the experimentations.

TABLE 6.1 – Default values of the Dynamic Risk Correlation Model parameters

Parameter name	**Meanings**	**Default value**	**Default value explanation**
`max-number-no-info-to-explore`	Number of successive `no-info` nodes to explore in the Dynamic Risk Correlation Model.	6	Maximum number of successive transitions without sensors allowed in the system.
`max-number-no-info-to-keep`	Number of successive `no-info` nodes to keep in the Dynamic Risk Correlation Model.	2	We need to keep at least the `no-info` node corresponding to the state.
`max-number-no-alert-to-explore`	Number of successive `no-alert` nodes to explore in the Dynamic Risk Correlation Model.	4	Maximum number of successive false negatives allowed in the system.
`max-number-no-alert-to-keep`	Number of successive `no-alert` nodes to keep in the Dynamic Risk Correlation Model.	0	A `no-alert` sensor does not bring any relevant compromise information.

Table 6.2 summarises the two parameters introduced by the Future Risk Assessment Model of the Hybrid Risk Assessment Model: `nbSteps-possible-futures` and `probability-new-transition`. Each parameter is associated with its description and the default value used in the experimentations.

TABLE 6.2 – Default values of the Future Risk Assessment Model parameters

Parameter name	**Meanings**	**Default value**	**Default value explanation**
`nbSteps-possible-futures`	Number of successive transitions to keep in the Future Risk Assessment Model.	2	Allows to assess possible futures with up to 2 following transitions. *cf.* subsection 6.3.5 for full explanation.
`probability-new-transition`	Probability that the attack propagates through a new transition.	0.5	70% of chance that the attacker does not continue his attack. He may have already found on this state what he was looking for.

6.7.2 Hybrid Risk Assessment Model initial results

In the same way to what we did with the Bayesian Attack Model, in order to study the impact of all the parameters on the results of the Hybrid Risk Assessment Model, we simulated a random Generic Attack Model of 35 states as presented in section 5.6, on which we apply six detection scenarios, corresponding to an attack of three transitions, with and without detection anomalies, as summarised in Table 5.4. Then, we compute the compromise probabilities, results of the Future Risk Assessment Model, according to the change on the parameters. These probabilities give the actual compromise status in addition to the probabilities of possible futures (in order to have results that can be compared with the Bayesian Attack Model). Finally, we plot the variation interval of the compromise probabilities of states, on the whole parameter variation interval, for all Generic Attack Model states.

The results of the Hybrid Risk Assessment Model for these attack scenarios with the parameters in their default values are shown in Figure 6.7.

Scenario 1 Figure 6.7a shows the compromise probabilities output of the Hybrid Risk Assessment Model of each state of the Generic Attack Model, after the first alert of the attack scenario. As the transition that has been detected issues from the main source of attacks, it is very likely and both the source and destination states of this transition have `critical` compromised probabilities (*i.e.*, $p > 0.75$). The other states have very `low` compromise probabilities (*i.e.*, $p < 0.25$), since they have not been detected.

Scenario 2 After the second alert, the compromise probabilities shown in Figure 6.7b confirms that an attack is actually happening and henceforth, the three `compromised` states (sources and destinations of attack transitions) have `critical` probabilities(*i.e.*, $p > 0.75$).

Scenario 3 Similarly, after the third alert, the four `compromised` states have `critical` probabilities(*i.e.*, $p > 0.75$), as shown in Figure 6.7c. Note also that the compromise probability of two `not-compromised` states has a little increased. These states are probably likely possible attack futures or other states that are in a possible attack path, compatible with the alerts.

Scenario 4 The fourth scenario, represented in Figure 6.7d, contains a false negative (the second transition). As a result, the compromise probabilities of the states are lower than for the third scenario, but still allow to distinguish surely the `compromised` states, from the `not-compromised` ones. The `compromised` states still have `critical` probabilities, whereas the `not-compromised` ones have `low` probabilities.

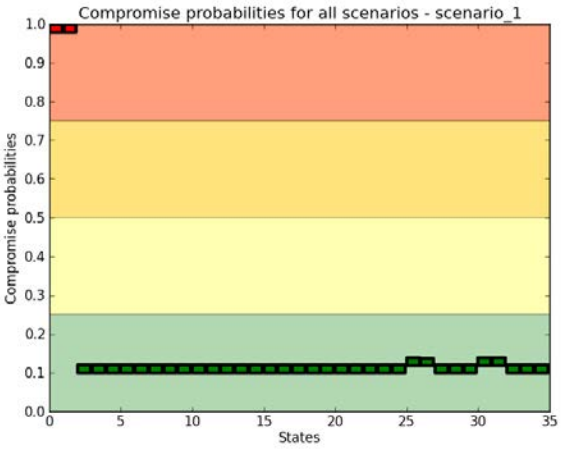
Scenario 5 The fifth scenario, represented in Figure 6.7e, contains a transition for which there is a `no-info` sensor. As this transition cannot be detected, the results of the model are nearly the same as the one of the third scenario.

Scenario 6 The last scenario contains all alerts, but also includes a false positive alert (not related to the previous ones). As this alert is not related to the last alert of the scenario, it has not been taken into account into the Hybrid Risk Assessment Model and both `compromised` and `not-compromised` states have the same probability as in scenarios 3 and 5.

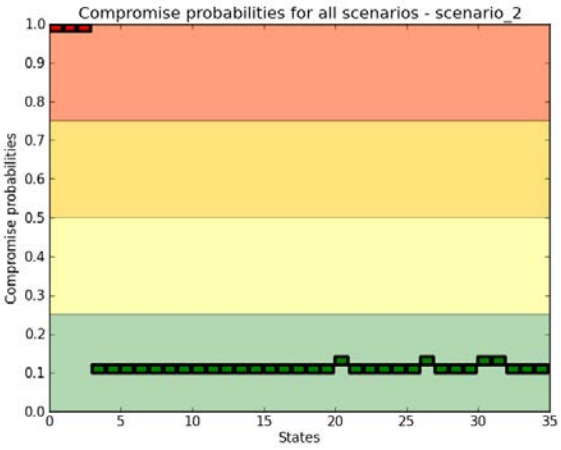
So, the compromise probabilities computed thanks to the Hybrid Risk Assessment Model shows the evolution of an attack scenario, according to the received alerts. The scenarios 1 to 3 shows the normal evolution of an attack where all attack transitions are detected. In such case, each new alert confirms that this multi-step attack is actually happening. Even with detection anomalies (`no-info` alert, false negative or false positive alert) in the scenario, the Hybrid Risk Assessment Model recognises the whole scenario that is happening, with a significant gap between the probabilities of `compromised` states and the ones of `not-compromised` states.

FIGURE 6.7 – Hybrid Risk Assessment Model results with parameter default values

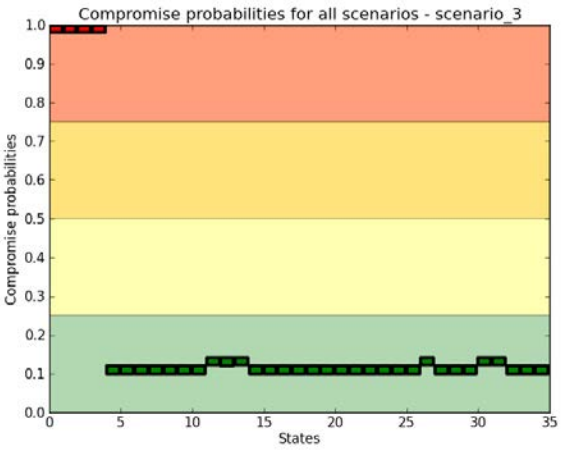
(A) 1st `alert`



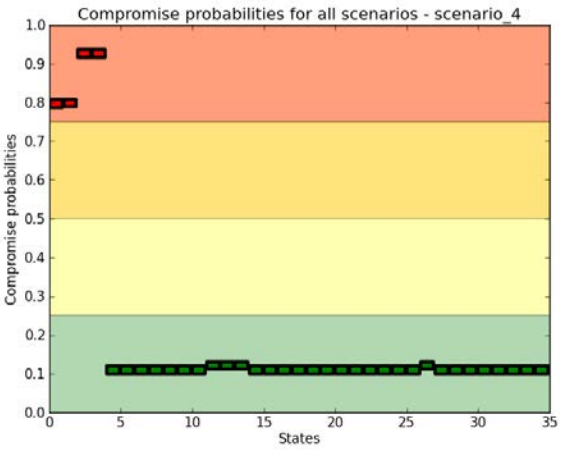
(B) 1st and 2nd `alert`



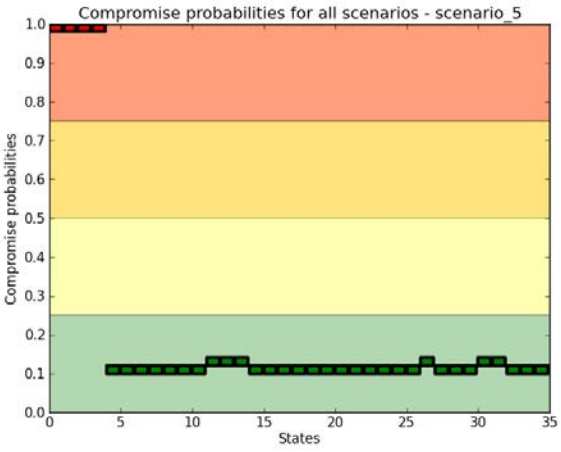
(C) 1st, 2nd and 3rd `alert`



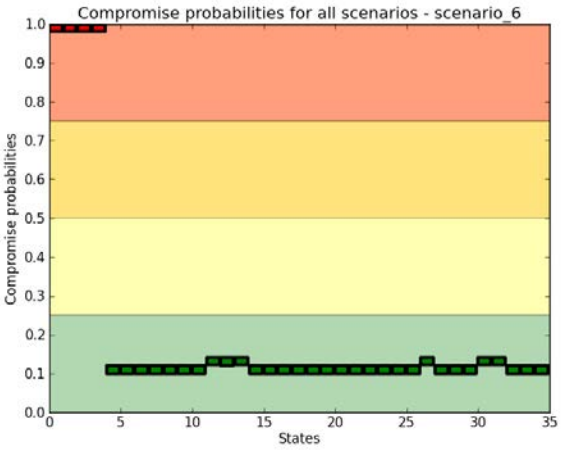
(D) 1st `alert`, 2nd `no-alert`, 3rd `alert`



(E) 1 `alert`, 2nd `no-info`, 1st `alert`



(F) 1 `alert`, 1 false positive `alert`, 3rd and 4th `alert`



6.7.3 Generic Attack Model parameter: `false-positive`

The `false-positive` parameter is associated with each sensor of the Generic Attack Model. It represents the probability of an alert raised by the related sensor to be a false positive alert, *i.e.* an alert raised without successive attack of the related state or transition.

Generally, to be usable in practice and limit irrelevant investigations, sensors are configured to limit at most the false positives. Thus the `false-positive` parameter should stay low (*e.g.*, from 0 to 5%). However, to be sure to take into account sensors with higher `false-positive` values (for example, an anomaly detection sensor with many false positives), we analyse the sensitivity of this parameter on its whole variation interval ($]0, 1[$). Figure 6.8 shows the results of this analysis for the Hybrid Risk Assessment Model.

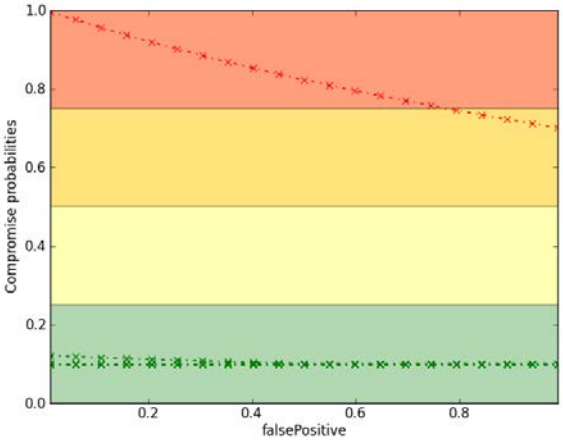
From 0 to 0.8, the `false-positive` parameter is moderately sensitive for `compromised` states. Their compromise probabilities decrease with the increase of this parameter. The `false-positive` parameter is nearly insensitive with `not-compromised` states, for all scenarios. When there are more alerts, this parameter is slightly less sensitive. This can be seen, for example, by comparing the compromise probabilities of scenarios 2 and 3 of Figure 6.8b and 6.8c with the ones of Scenario 1 of Figure 6.8a. This behaviour can be explained by the fact that, with more alerts, it is more likely that an attack is actually occurring. Thus, having a false positive attack scenario is less probable. In Scenario 4 of Figure 6.8d, with a `no-alert` sensor, the impact of this parameter is bigger. The probabilities of the 4 `compromised` states of the attack scenario decrease significantly, except for the main attack source that keeps its initial `probability-attack-source` of 0.7. We can explain this faster decrease of compromise probabilities by the fact that the 3 steps the attack scenario are less likely with potentially false positive alerts surrounding a `no-alert` sensor. In scenarios 5 and 6 of Figure 6.8e and 6.8f, the impact of the `no-info` alert or a false positive alert has no impact on the impact of the `false-positive` parameter.

From 0.8 to 1, there is a fast increase of the compromise probabilities of `compromised` states and a few `not-compromised` states until the 1 probability. As this range of value for the `false-positive` parameter is extreme, it has edge effects.

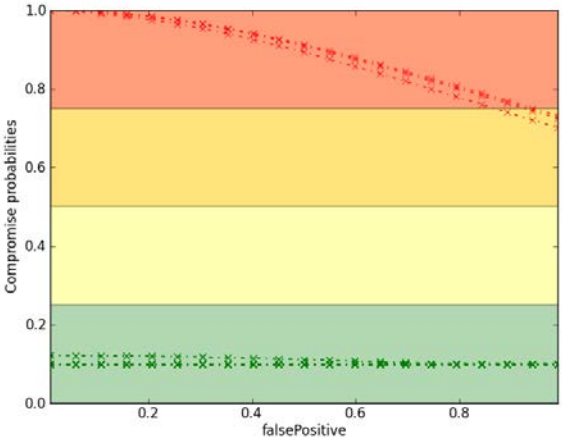
This sensitivity analysis shows that, on most of its variation interval (and by far the most frequently used in practice), the `false-positive` parameter has a very low impact on the compromise probabilities of the `compromised` states of the Hybrid Risk Assessment Model. This parameter has a low impact on the compromise probabilities of `not-compromised` states. For extreme values of the `false-positive` parameter (> 0.8), the compromise probabilities of `compromised` and few `not-compromised` states tend to 1. Thus, this parameter should not be used on the interval $[0.8, 1]$, which is anyway an interval that is not realistic in practice.

FIGURE 6.8 – Sensitivity of the `false-positive` parameter from 0 to 1

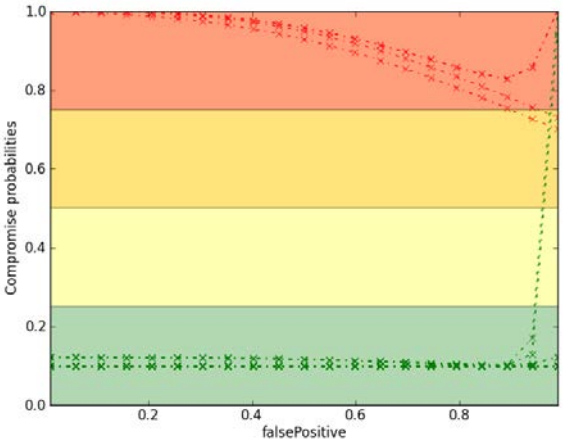
(A) Scenario 1



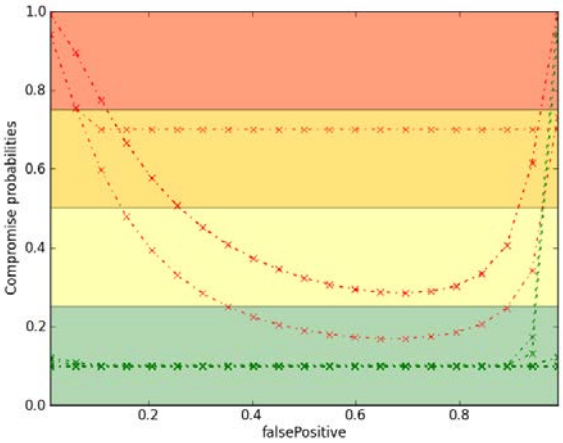
(B) Scenario 2



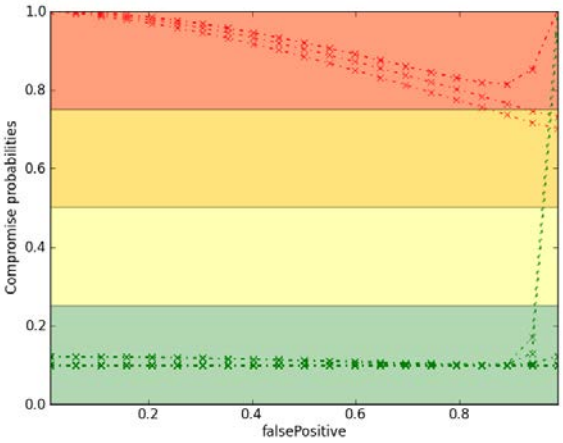
(C) Scenario 3



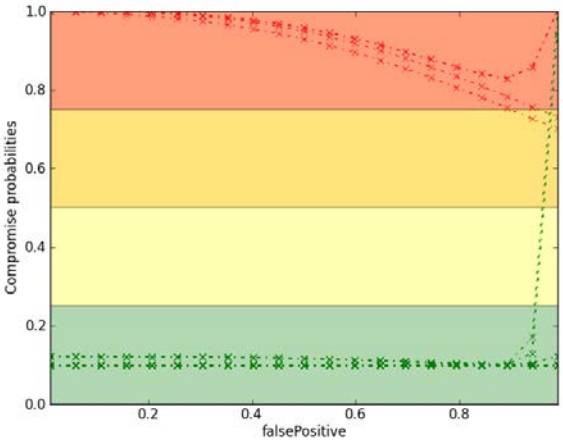
(D) Scenario 4



(E) Scenario 5



(F) Scenario 6



6.7.4 Generic Attack Model parameter: `false-negative`

The `false-negative` parameter is associated with each sensor of the Generic Attack Model. It represents the probability of false negative alerts for a sensor, *i.e.* an alert that is not raised by the sensor, while the attack actually succeeded.

Generally, to limit a wrong security feeling (*i.e.*, feeling safe, while there is no real protections), sensors are configured to limit at most the false negatives for known attacks, for which a detection method is enabled. Thus the `false-negative` parameter should stay low (*e.g.*, from 0 to 5%). Beyond, the detection measures are generally not deployed. However, to be sure to take into account sensors with higher `false-negative` values (for example, a detector of a metamorphic malware), we analyse the sensitivity of this parameter on its whole variation interval $]0, 1[$. Figure 6.9 shows the results of this analysis for the Hybrid Risk Assessment Model.

From 0 to 0.2, the `false-negative` is a little sensitive, for `compromised` states. On scenarios 3 to 6 of Figures 6.9c to 6.9f, three `not-compromised` states are very sensitive to variation in the `false-negative` parameter. These states are part of other potential transitions that may cause the detected third attack transition. When the probability of having false negative increases, the probability of these likely transitions having occurred, without effective alert, increases. Thus the compromise probabilities of the states impacted by these transitions increases. On the scenario 4 of Figure 6.9d, in which there is a `no-alert` false negative alert, the probabilities of `compromised` states, are smaller for very low values of the `false-negative` parameter. Indeed, if it is very unlikely that there is a false negative, the global scenario of 3 alerts (around a `no-alert`) is also unlikely.

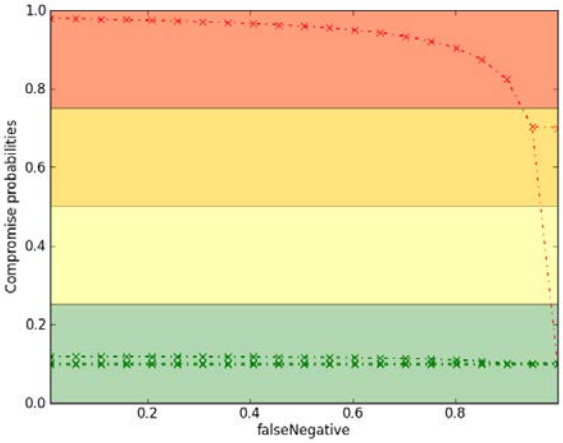
From 0.2 to 0.9, the parameter `false-negative` is nearly insensitive for both `compromised` and `not-compromised` states. There is only a tiny decrease of the probability of `compromised` states, for all scenarios.

From 0.9 to 1, for all states and scenarios, there is a very significant decrease of compromise probabilities for both `compromised` and `not-compromised` states, which all converge to their initial attack source probability (`probability-attack-source` for all sources, except the actual attack source that has a 0.7 probability in these simulations).

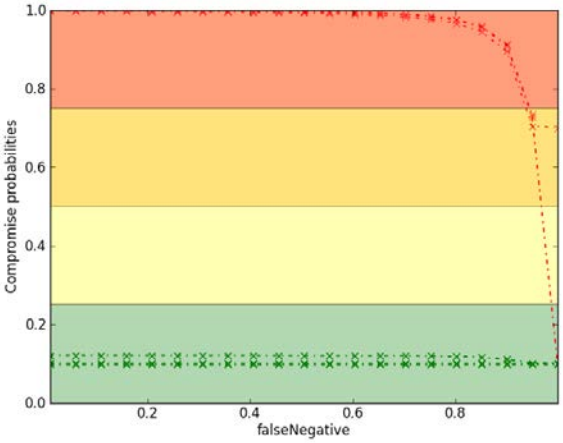
This sensitivity analysis shows that, on most of its variation interval (and by far the most frequently used in practice), the `false-negative` parameter has a little impact on the compromise probabilities of the `compromised` states of the Hybrid Risk Assessment Model. However, when its value is above 5%, it impacts significantly the `not-compromised` states that may be impacted by likely transitions close to the detected ones.

FIGURE 6.9 – Sensitivity of the `false-negative` parameter from 0 to 1

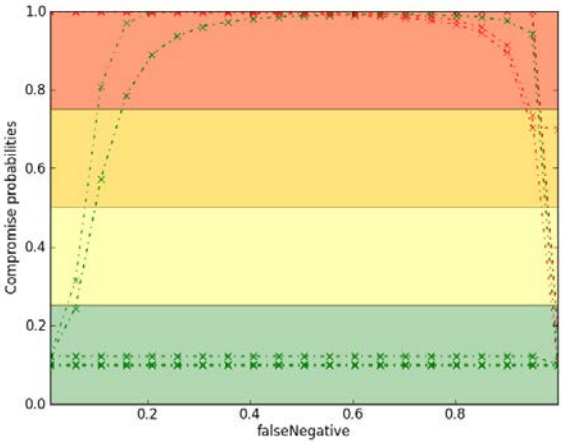
(A) Scenario 1



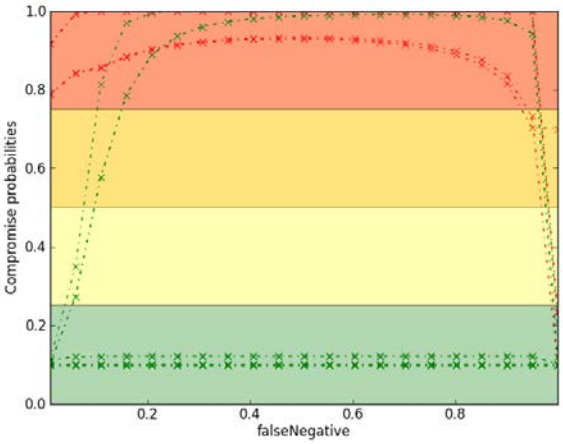
(B) Scenario 2



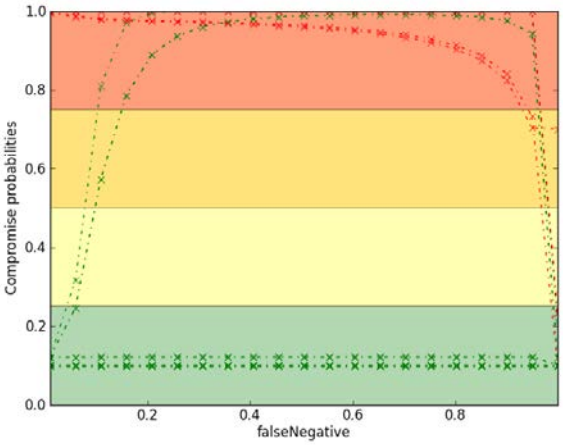
(C) Scenario 3



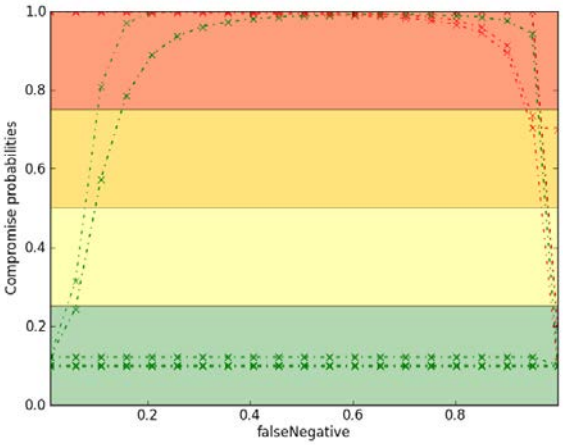
(D) Scenario 4



(E) Scenario 5



(F) Scenario 6



6.7.5 Generic Attack Model parameter: `probability-attack-source`

The `probability-attack-source` parameter is the probability associated with each state of the Generic Attack Model, representing the probability of this state to be a source of attack. The value of this parameter is set by the security operators according to the risk analysis of the system.

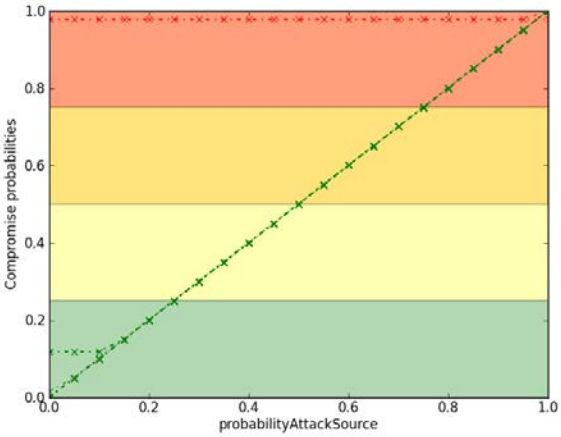
According to the state, this parameter can take any value possible from 0 to 1. We thus simulate the variation of this parameter on $[0, 1]$ and analyse its impact on the states compromise probabilities. This parameter can generally be deduced from a risk evaluation methodology (*e.g.*, ISO 27005 [ISO11a] or EBIOS [Sec04]). Figure 6.10 shows the results of this analysis.

From 0 to 1, the impact of the `probability-attack-source` parameter is relatively homogeneous for all scenarios of Figures 6.10a to 6.10f, and on the whole variation interval. It has nearly no impact on the compromise probabilities of `compromised` states, except for the Scenario 4, with a false negative. The impact of this parameter is medium, for all `not-compromised` states. The `probability-attack-source` parameter determines the probability of all states being an attack source, except the actual attack source, on the whole parameter variation interval. As this parameter is used especially as the probability of each attack source of a Future Risk Assessment Model, the value of the `probability-attack-source` parameter is the minimum that each state, either `compromised` or `not-compromised` will have in the Future Risk Assessment Model. This is why there is almost a straight line from (0,0) to (1,1) for `not-compromised` states for all scenarios. In scenario 4 of Figure 6.10d, with a false negative `no-alert` sensor, the probability of `compromised` states slowly decrease then increase with the increase of the `probability-attack-source` parameter. For very low values of the parameter, the probabilities of the first states are `high`, as the main source of attack is more likely to be the source of this attack scenario, than the other attack sources. Then, the parameter increases and as the other sources of attacks are more likely to be the source of the detected attack transition, the actual attack source is less likely. Finally, the compromise probabilities of those states increase again when the probability of all attack sources is higher.

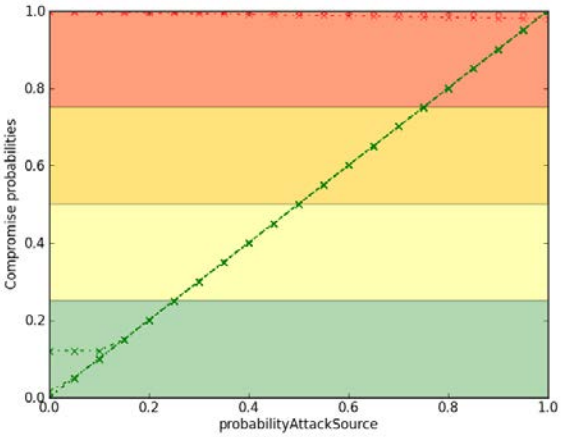
This sensitivity analysis shows that, on its whole variation interval, the `probability-attack-source` parameter has a significant impact on the compromise probabilities of the `not-compromised` states of the Hybrid Risk Assessment Model. It has nearly no impact on probabilities of `compromised` states. On its whole variation interval, with the increase of this parameter, the compromise probabilities of the `not-compromised` states increase slowly. Indeed, this parameter represents the a priori probability of the attacker to start an attack from the states other than the actual attack source of this attack scenario.

FIGURE 6.10 – Sensitivity of the `probability-attack-source` parameter from 0 to 1

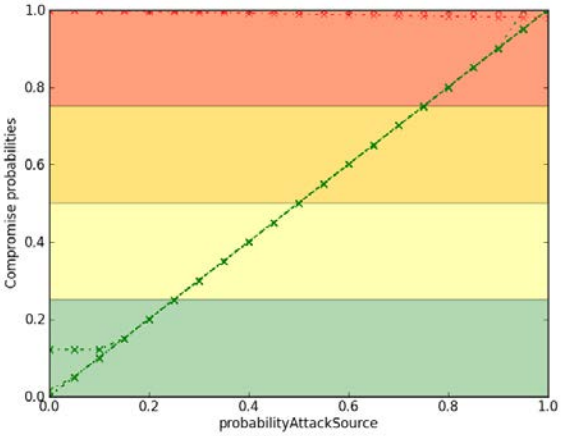
(A) Scenario 1



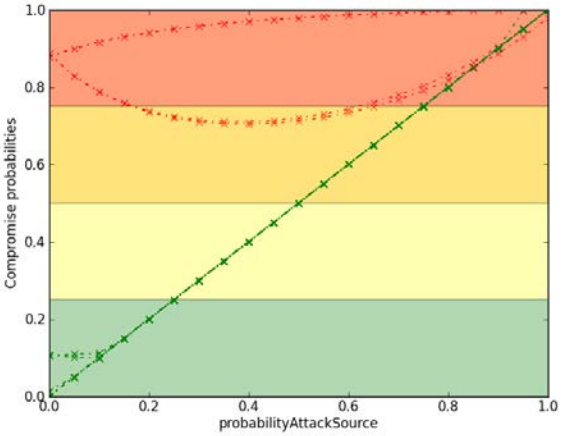
(B) Scenario 2



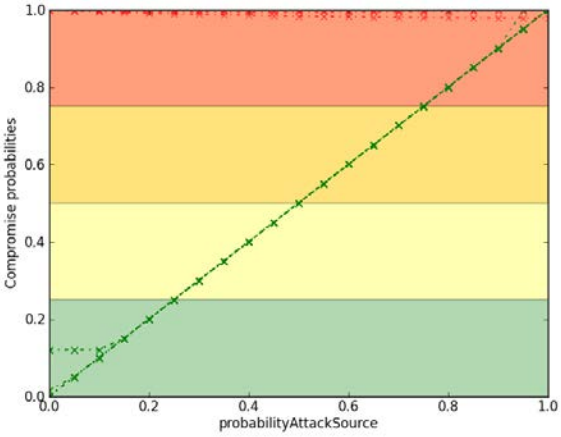
(C) Scenario 3



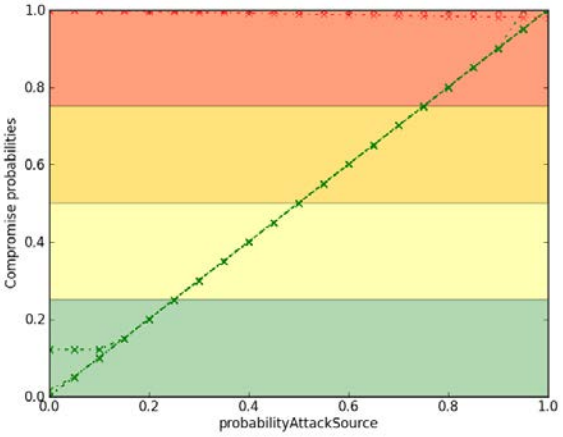
(D) Scenario 4



(E) Scenario 5



(F) Scenario 6



6.7.6 Generic Attack Model parameter: `probability-unknown-attack`

The `probability-unknown-attack` parameter is the probability that an unknown attack allows the attacker to reach a place, without requiring the previous transition. This is a kind of wild card representing all the attacks unknown for the defender, but known from the attacker.

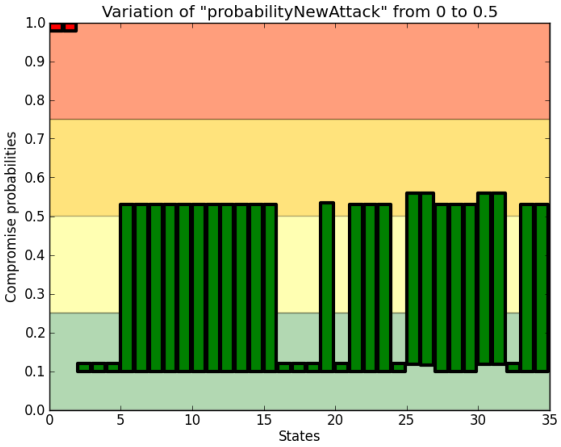
This value should usually stay very low ($< 1\%$), otherwise this means that there is an issue in the modelling of the attack scenarios that can happen in the system (*e.g.*, not good knowledge of the system by the defender, partial knowledge of the vulnerabilities that are in the system, *etc.*). However, there might be good reasons to set higher values to this parameter. For example, in systems in which a very high level of security is needed, we can increase the value of the `probability-unknown-attack` parameter, to represent very motivated attackers that may have access to unknown attack transitions (*e.g.*, unknown 0-days vulnerabilities). We thus simulate the variation of this parameter on $[0, 0.5]$ and analyse its impact on the states compromise probabilities. Figure 6.11 shows the results of this analysis.

From 0 to 0.5, the impact of the `probability-unknown-attack` parameter is relatively homogeneous for all scenarios of Figures 6.11a to 6.11f, and on the whole variation interval. It has nearly no impact on the compromise probabilities of `compromised` states, except for the Scenario 4, with a false negative. The impact of this parameter is medium, for all `not-compromised` states. In all scenarios, more than half of `not-compromised` states increase slowly with the increase of the parameter. The probability of the other states do not increase. This must be due to the fact that these states cannot be attacked directly (*i.e.*, with only one transition) from the `compromised` states. For such states, the increase of the `probability-unknown-attack` parameter has nearly no impact, as the attacker will need to exploit two or more unknown transitions to reach them. For the scenario 4 of Figure 6.11d, with a `no-alert` sensor, the probabilities of `compromised` states increase slowly as it is more and more likely that the missing alert has not been detected, because it was an unknown attack.

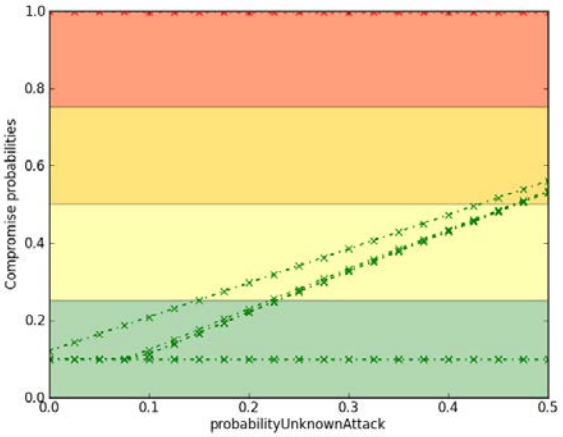
This sensitivity analysis shows that, on its whole variation interval the `probability-unknown-attack` parameter has a little impact on the compromise probabilities of the `not-compromised` states of the Hybrid Risk Assessment Model. With the increase of this parameter, the compromise probabilities of the `not-compromised` states slightly increase. It has no impact on probabilities of `compromised` states.

FIGURE 6.11 – Sensitivity of the `probability-unknown-attack` parameter from 0 to 0.5

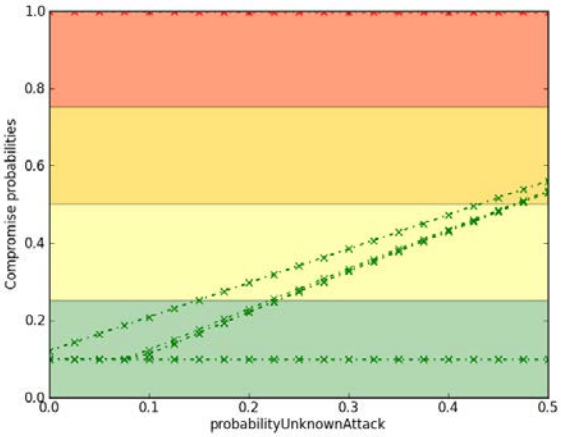
(A) Scenario 1



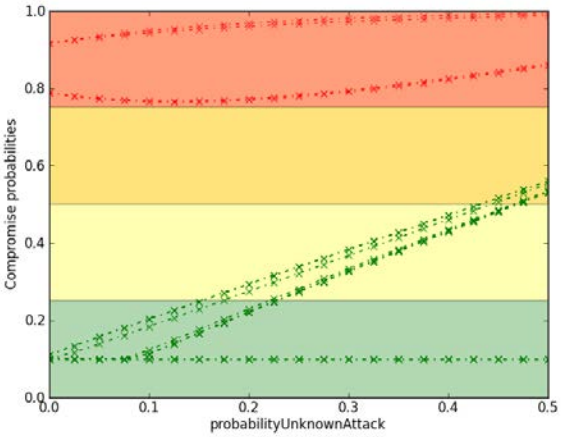
(B) Scenario 2



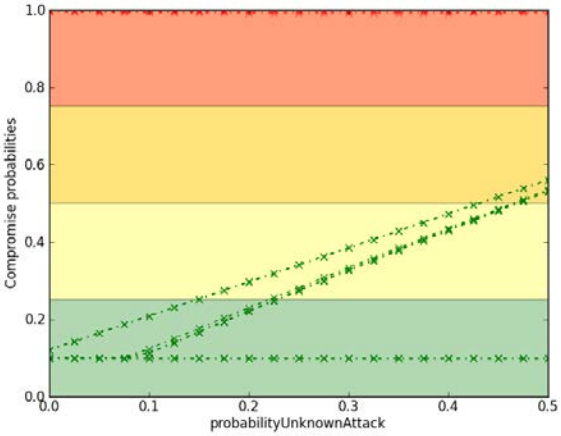
(C) Scenario 3



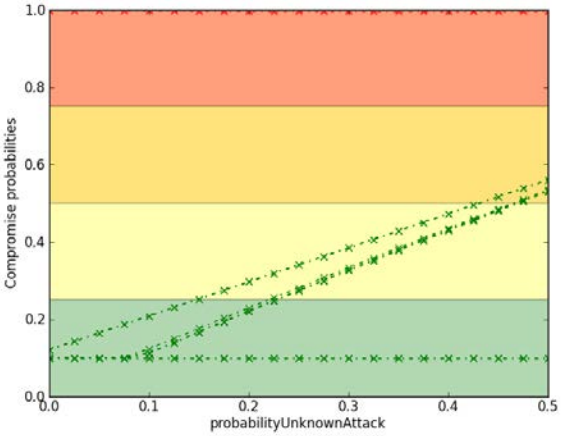
(D) Scenario 4



(E) Scenario 5



(F) Scenario 6



6.7.7 Dynamic Risk Correlation Model parameter: `max-number-no-info-to-explore`

The `max-number-no-info-to-explore` parameter represents the number of `no-info` sensors to investigate within a Dynamic Risk Correlation Model after one `alert` sensor, and before seeing another `alert` sensor. This parameter represents in fact the maximum of `no-info` sensors that can be between two `alert` sensors in an attack scenario.

In order to measure the impact of this pruning parameter, we simulated the variation of this parameter on $[[0 - 10]]$ Figure 6.12 shows the results of this analysis.

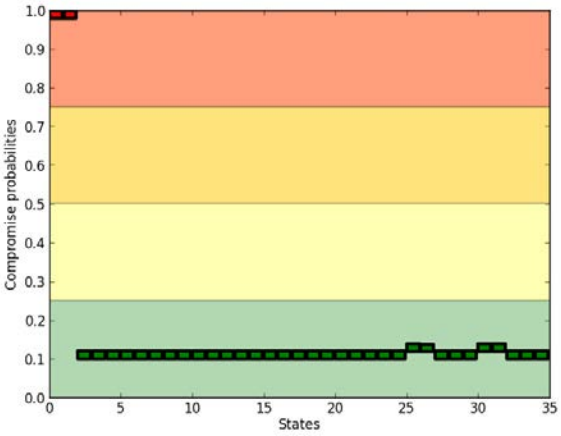
From 0 to 3, the `max-number-no-info-to-explore` impacts principally the `compromised` states, depending on the scenario. For all scenarios, this parameter is nearly insensitive for `not-compromised` states. In Scenario 1, 2, 3 and 6 of Figures 6.12a, 6.12b, 6.12c and 6.12f, as soon as the `max-number-no-info-to-explore` parameter is > 1, all states have their final compromise probabilities. The value 1 of the `max-number-no-info-to-explore` corresponds to the Bayesian State nodes (that, by default, have a `no-info` sensor) that are parents of the `alert` Bayesian Transition nodes. When the value of the parameter is 0, the Bayesian State nodes are not explored and their compromise probabilities are lower. In scenarios 4 and 5, there are detection anomalies (one false negative or one `no-info`). Each detection anomaly requires to increment the `max-number-no-info-to-explore` parameter (twice, one for a Bayesian State node and one for a Bayesian Transition node, if it is related to a `no-info` sensor), for all states to have their final compromise probabilities.

From 3 to 10, the `max-number-no-info-to-explore` is insensitive for both `compromised` and `not-compromised` states.

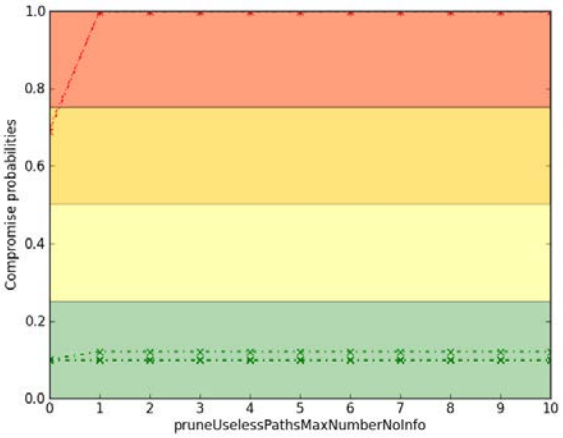
This sensitivity analysis shows that the `max-number-no-info-to-explore` parameter has an impact on the compromise probabilities of the `compromised` states, but for very low values (< 3), for all scenarios. From 3, this parameter does not have any impact on compromise probabilities.

FIGURE 6.12 – Sensitivity of the `max-number-no-info-to-explore` parameter from 0 to 10

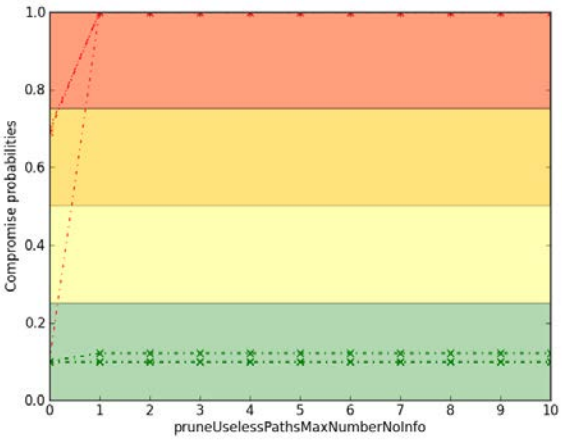
(A) Scenario 1



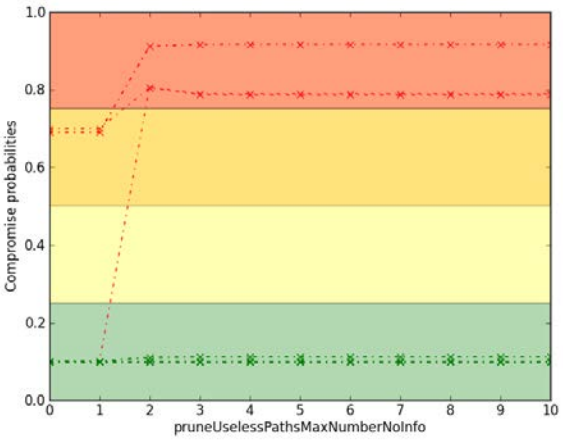
(B) Scenario 2



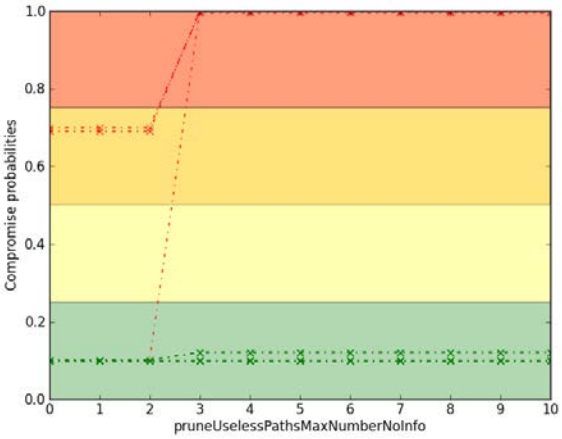
(C) Scenario 3



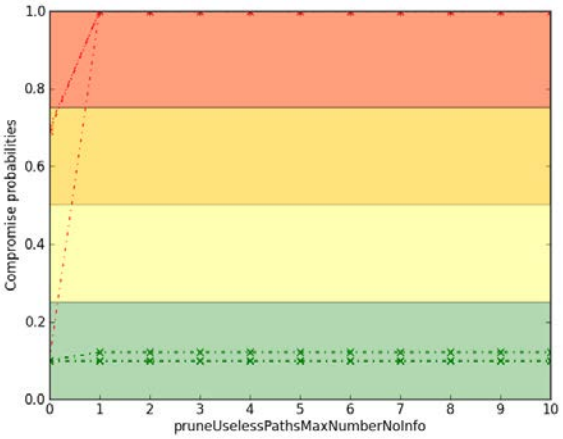
(D) Scenario 4



(E) Scenario 5



(F) Scenario 6



6.7.8 Dynamic Risk Correlation Model parameter: `max-number-no-info-to-keep`

The `max-number-no-info-to-keep` parameter represents the number of `no-info` sensors to keep when there are too many `no-info` sensors after an `alert`. This parameter represents in fact the maximum of successive `no-info` sensors that will follow an `alert` sensor in an attack scenario.

In order to measure the impact of this pruning parameter, we simulated the variation of this parameter on $[[0 - 10]]$ Figure 6.13 shows the results of this analysis.

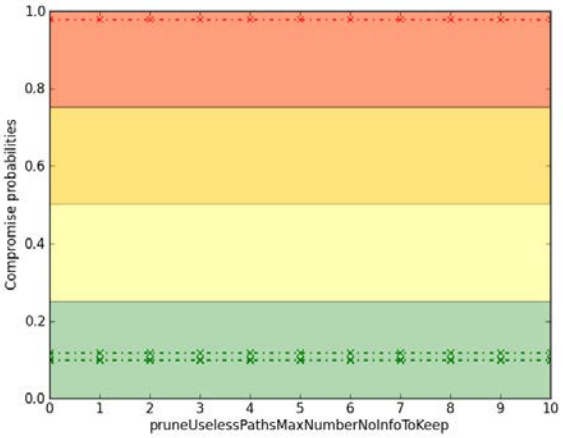
From 0 to 1, the `max-number-no-info-to-keep` impacts principally the `compromised` states, depending on the scenario. For all scenarios, this parameter is nearly insensitive for `not-compromised` states. For all scenarios of Figures 6.12a to 6.12f, as soon as the `max-number-no-info-to-explore` parameter is > 1, all states have their final compromise probabilities. The value 1 of the `max-number-no-info-to-keep` corresponds to the Bayesian State nodes (that, by default, have a `no-info` sensor) that are parents of the `alert` Bayesian Transition nodes. When the value of the parameter is 0, the Bayesian State nodes are pruned and their compromise probabilities are lower.

From 1 to 10, the `max-number-no-info-to-keep` is insensitive for both `compromised` and `not-compromised` states.

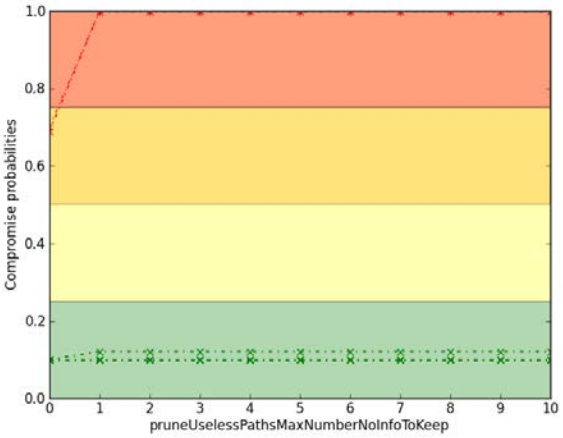
This sensitivity analysis shows that the `max-number-no-info-to-keep` parameter has an impact on the compromise probabilities of the `compromised` states, but only for the 0 value, for all scenarios. From 1, this parameter does not have any impact on compromise probabilities.

FIGURE 6.13 – Sensitivity of the `max-number-no-info-to-keep` parameter from 0 to 10

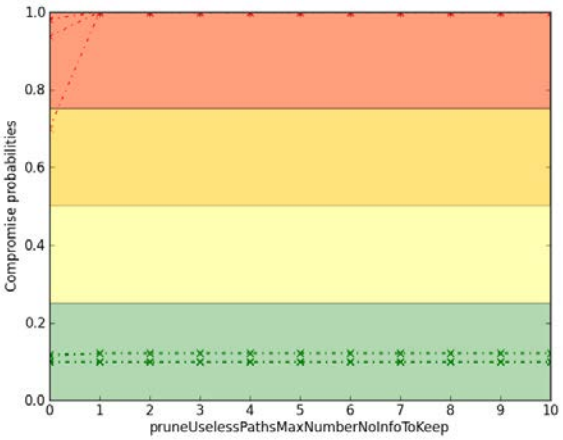
(A) Scenario 1



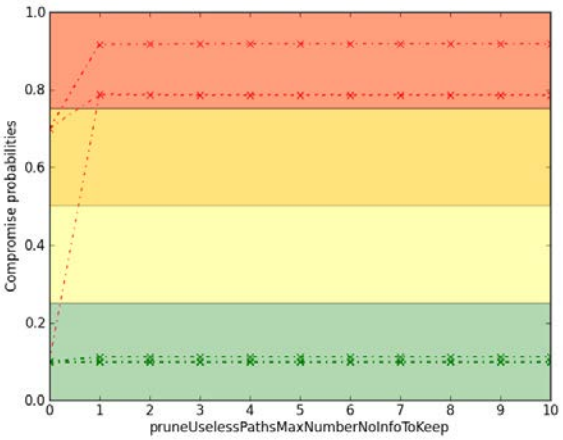
(B) Scenario 2



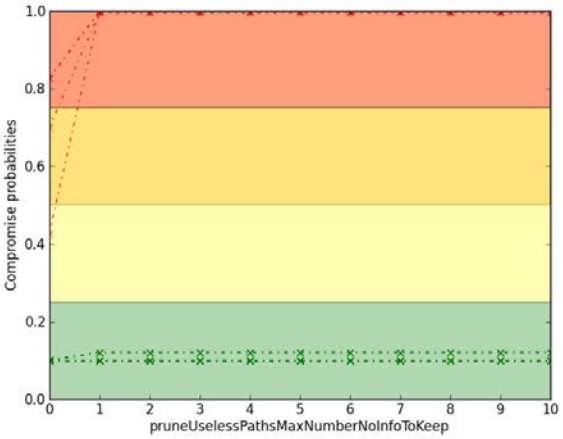
(C) Scenario 3



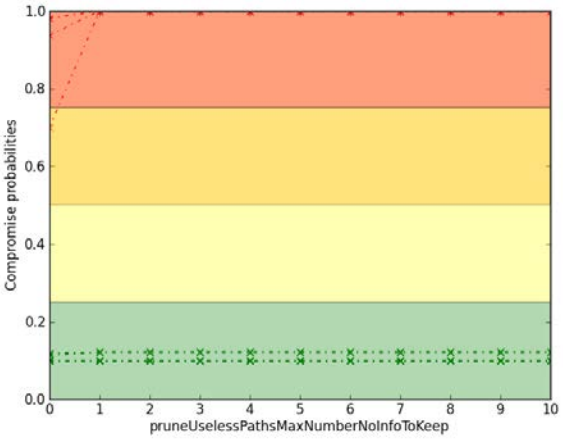
(D) Scenario 4



(E) Scenario 5



(F) Scenario 6



6.7.9 Dynamic Risk Correlation Model parameter: `max-number-no-alert-to-explore`

The `max-number-no-alert-to-explore` parameter represents the number of `no-alert` sensors to investigate within a Dynamic Risk Correlation Model after one `alert` sensor, and before seeing another `alert` sensor. This parameter represents in fact the maximum of `no-alert` sensors that can be between two `alert` sensors in an attack scenario.

In order to measure the impact of this pruning parameter, we simulated the variation of this parameter on $[[0 - 10]]$ Figure 6.14 shows the results of this analysis.

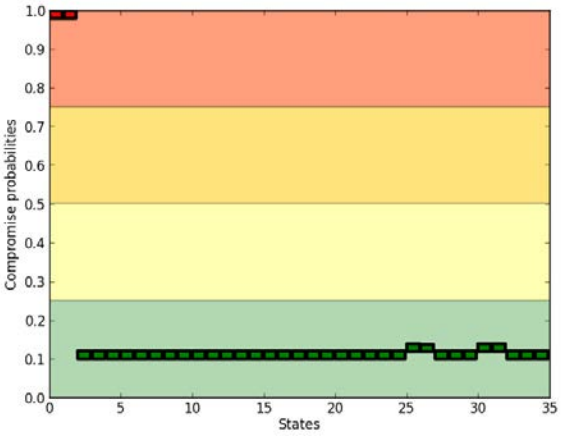
From 0 to 2, the `max-number-no-alert-to-explore` impacts only the `compromised` states of the scenario 4. For the other scenarios, this parameter has no influence on the probability of `compromised` states. For all scenarios, this parameter is nearly insensitive for `not-compromised` states. In Scenario 4 of Figure 6.14d, there is a false negative. This false negative requires to increment the `max-number-no-alert-to-explore` parameter to explore enough `no-alert` nodes to reach the next `alert` node. Thus, from `max-number-no-alert-to-explore` $= 2$, all states to have their final compromise probabilities.

From 2 to 10, the `max-number-no-alert-to-explore` is insensitive for both `compromised` and `not-compromised` states.

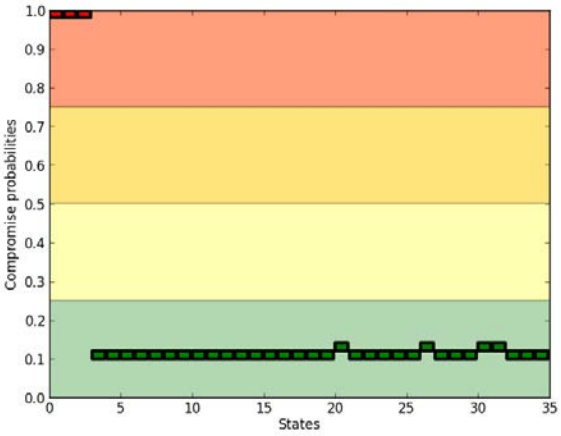
This sensitivity analysis shows that the `max-number-no-alert-to-explore` parameter has an impact on the compromise probabilities of the `compromised` states, but for very low values (< 2), and only when there is a false negative. From 2, this parameter does not have any impact on compromise probabilities.

FIGURE 6.14 – Sensitivity of the `max-number-no-alert-to-explore` parameter from 0 to 10

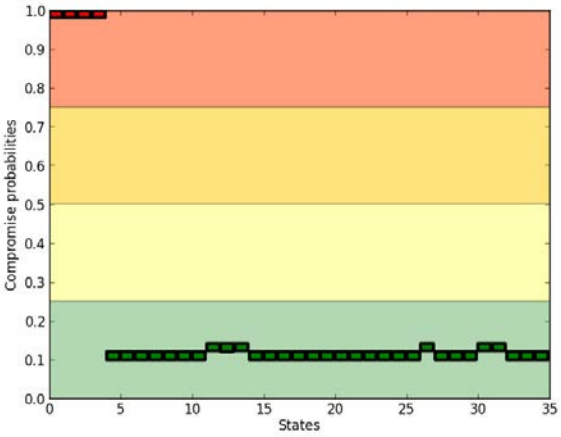
(A) Scenario 1



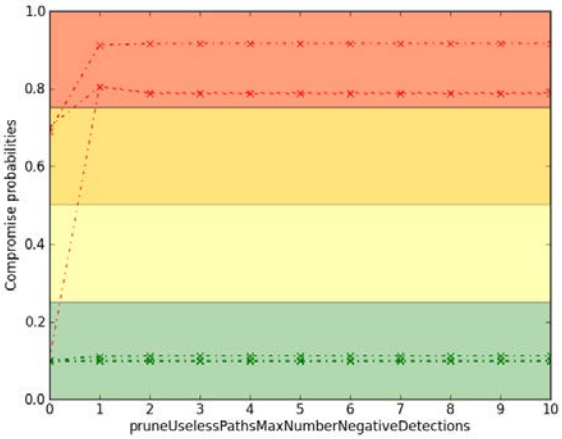
(B) Scenario 2



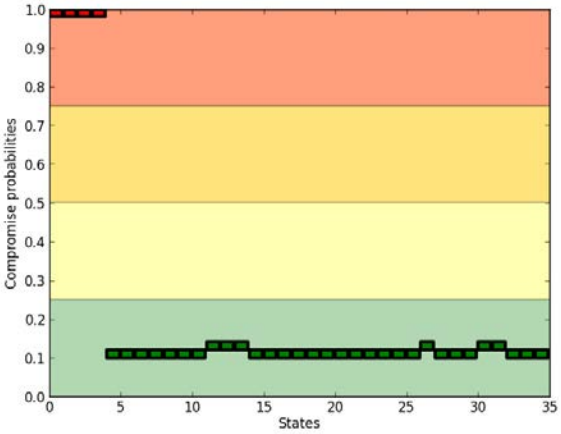
(C) Scenario 3



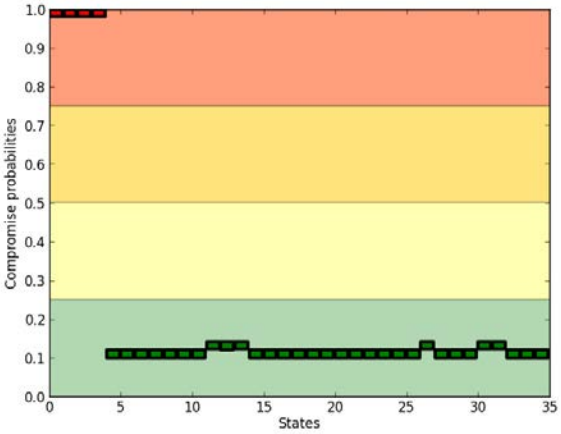
(D) Scenario 4



(E) Scenario 5



(F) Scenario 6



6.7.10 Dynamic Risk Correlation Model parameter: `max-number-no-alert-to-keep`

The `max-number-no-alert-to-keep` parameter represents the number of `no-info` sensors to keep when there are too many `no-alert` sensors after an `alert`. This parameter represents in fact the maximum of successive `no-alert` sensors that will follow an `alert` sensor in an attack scenario.

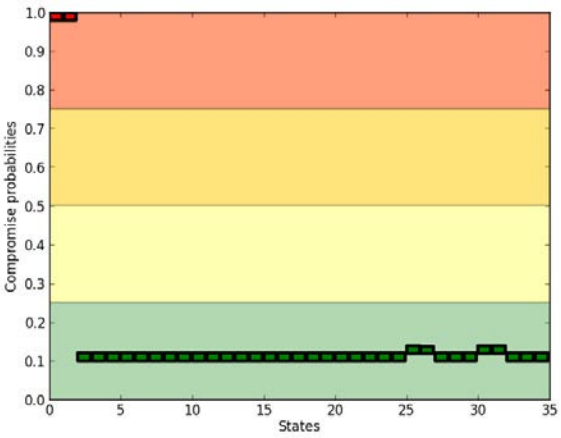
In order to measure the impact of this pruning parameter, we simulated the variation of this parameter on $[[0 - 10]]$ Figure 6.15 shows the results of this analysis.

From 0 to 10, the `max-number-no-alert-to-keep` is insensitive for both `compromised` and `not-compromised` states.

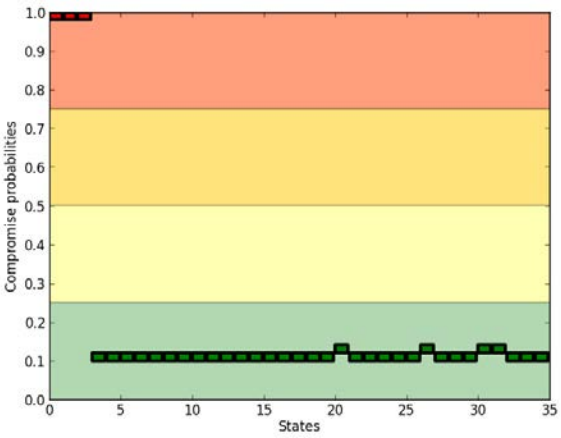
This sensitivity analysis shows that the `max-number-no-alert-to-keep` parameter does not have any impact on compromise probabilities.

FIGURE 6.15 – Sensitivity of the `max-number-no-alert-to-keep` parameter from 0 to 10

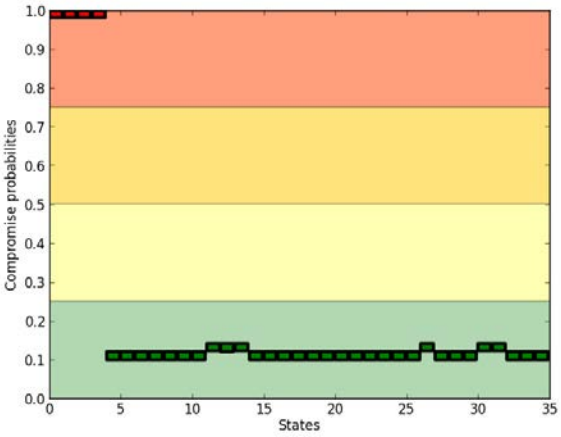
(A) Scenario 1



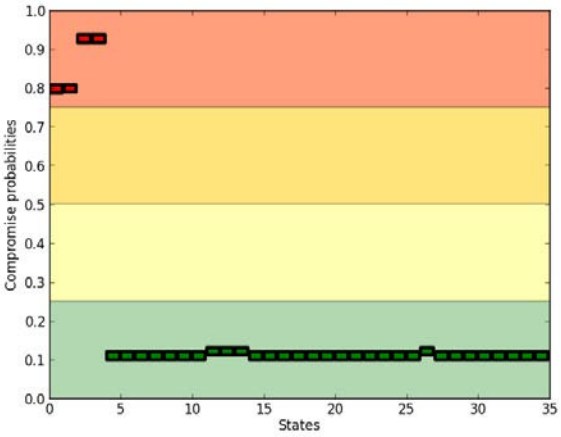
(B) Scenario 2



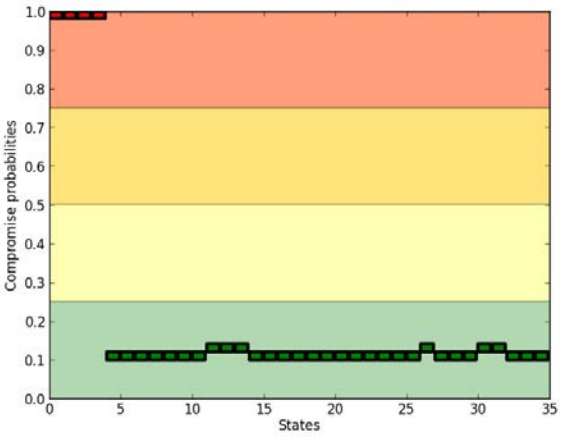
(C) Scenario 3



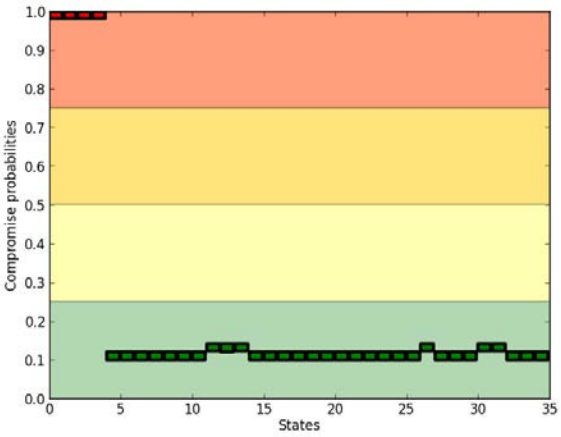
(D) Scenario 4



(E) Scenario 5



(F) Scenario 6



6.7.11 Future Risk Assessment Model parameter: `nbSteps-possible-futures`

The `nbSteps-possible-futures` parameter represents the number of successive transitions to keep in each Future Risk Assessment Model. This parameter prevents the combinatorial explosion of the number of nodes in the Future Risk Assessment Model. Detailed explanations about this parameter are in subsection 6.3.5.

It is unrealistic to seek to "predict the future" too many steps in advance. Even predicting 4 steps of what will do an attacker is very optimistic. Thus, we study the sensitivity of this parameter on $[[0-4]]$ Figure 6.16 shows the results of this analysis.

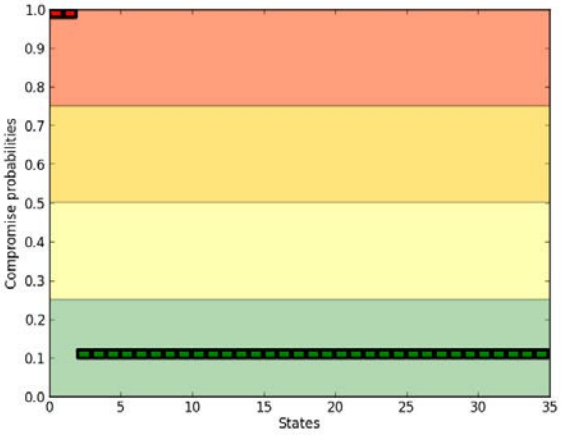
From 0 to 2, for all scenarios, the `nbSteps-possible-futures` is a little sensitive for `not-compromised` states. When the number of `nbSteps-possible-futures` increases, the probability of the `not-compromised` states that are possible futures increase slightly. This parameter has no impact on the `compromised` states, because these states have already been attacked, and they are not any more likely futures.

From 2 to 4, the `nbSteps-possible-futures` is insensitive for both `compromised` and `not-compromised` states.

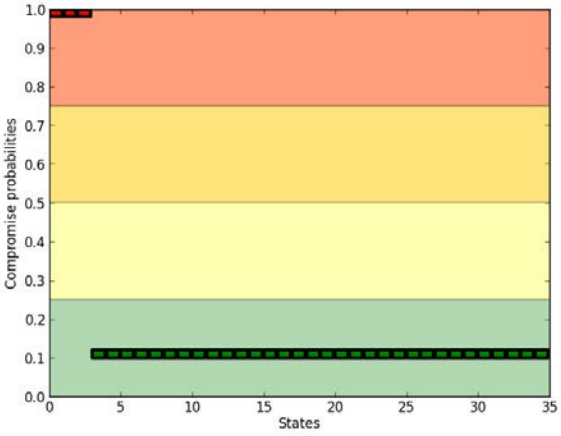
This sensitivity analysis shows that the `nbSteps-possible-futures` parameter has a very little impact for its low values (< 3) on the compromise probabilities of the `not-compromised` states of the Future Risk Assessment Model. It does not impact the probabilities of the `compromised` states.

FIGURE 6.16 – Sensitivity of the `nbSteps-possible-futures` parameter

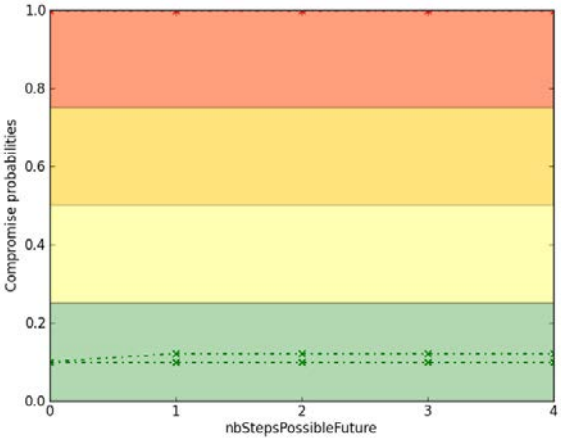
(A) Scenario 1



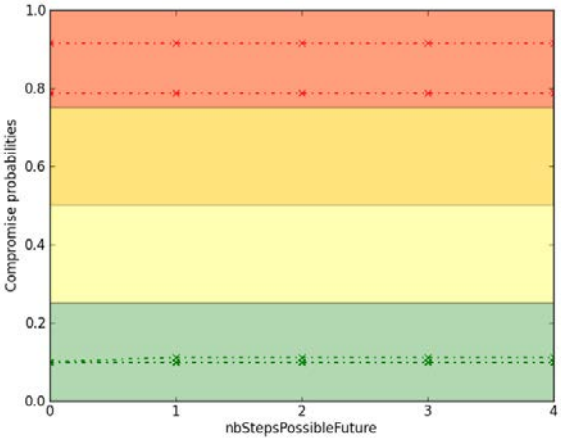
(B) Scenario 2



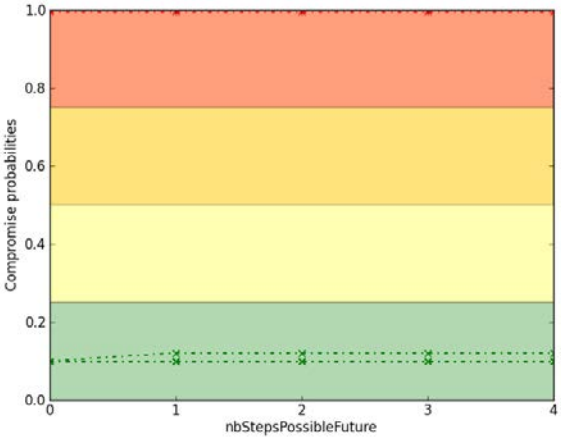
(C) Scenario 3



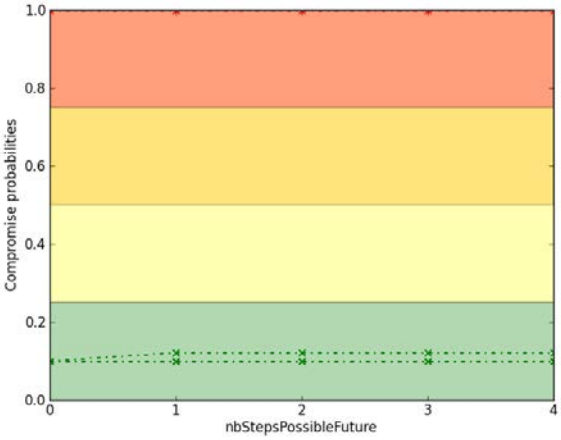
(D) Scenario 4



(E) Scenario 5



(F) Scenario 6



6.7.12 Future Risk Assessment Model parameter: `probability-new-transition`

The `probability-new-transition` parameter is the probability that the attacker propagates through a new transition in the future. This parameter represents the fact the even if an attack is possible, the attacker may or may not do it. For example, the attacker may have already found what he was looking for.

The value of this parameter is difficult to estimate, so the `probability-new-transition` parameter has to be evaluated on its whole possible variation interval. We thus simulate the variation of this parameter on $[0, 1]$ and analyse its impact on the states compromise probabilities. Figure 6.17 shows the results of this analysis.

From 0 to 0.1, the `probability-new-transition` is insensitive for both `compromised` and `not-compromised` states.

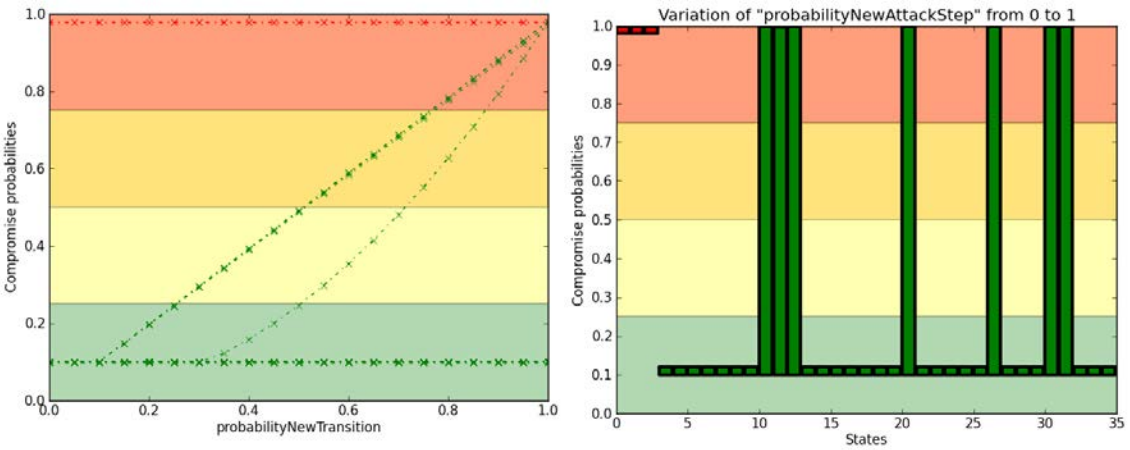
From 0.1 to 1, the `probability-new-transition` is insensitive for both `compromised` states. However, for `not-compromised` states, it has a significant impact on the compromise probabilities. When the `probability-new-transition` parameter increases, the probability of `not-compromised` states increases. According to the scenarios, the growing curve of the `not-compromised` states may be different, but the tendency is the same for all of them. Only a few nodes have their probability changing according to the value of this parameter: 5 in Scenario 1, 7 in Scenario 2, 9 in Scenario 3. These are the likely next attack states that may be attacked. The different curves of `not-compromised` states represent the number of attack transitions from the `compromised` states (*e.g.*, we can recognise the curves $y = x$ and $y = x^2$ in Figures 6.17a, 6.17c, 6.17d, 6.17e and 6.17f). In Scenario 4 of Figure 6.17d, some of the `compromised` states have probabilities lower than 1. Thus, the curve of the possible futures of those states is multiplied by the source compromised probabilities.

This sensitivity analysis shows that, on its whole possible variation interval, the `probability-new-transition` parameter has an important impact on the compromise probabilities of the `not-compromised` states of the Future Risk Assessment Model. With the increase of this parameter, the compromise probabilities of some of the `not-compromised` states increase with this parameter.

FIGURE 6.17 – Sensitivity of the `probability-new-transition` parameter from 0 to 1

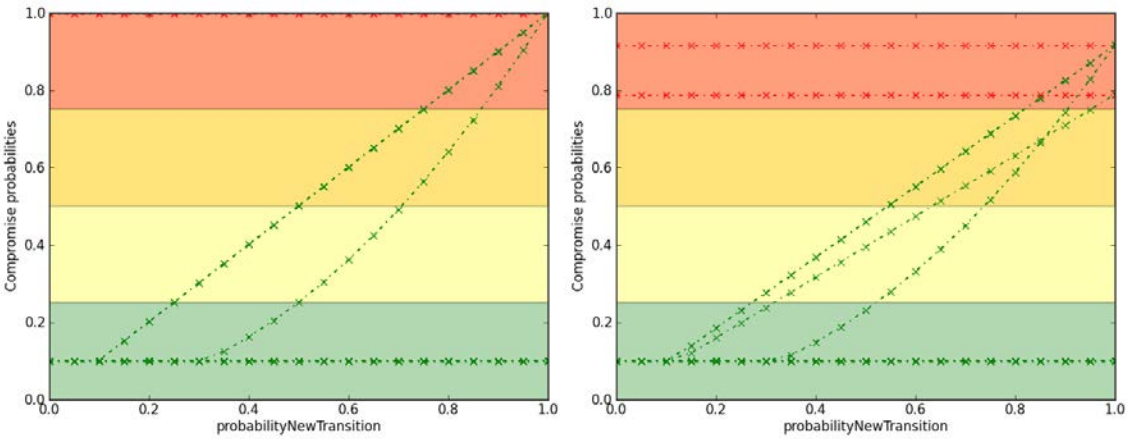
(A) Scenario 1

(B) Scenario 2



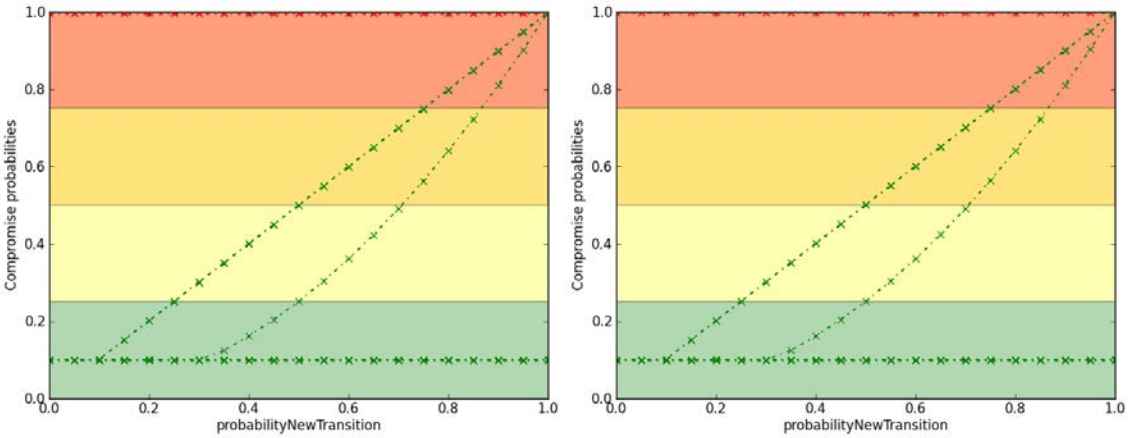
(C) Scenario 3

(D) Scenario 4



(E) Scenario 5

(F) Scenario 6



6.8 Related work

As described in section 5.8, few people propose enhancements to improve attack graphs or trees with Bayesian networks, in order to use them for dynamic risk assessment [QL04, LM05, XLO+10]. However, they do not describe how they manage cycles that are inherent to attack graphs. In chapter 5, we have presented a new risk assessment model that explodes cycles in the building process, keeping all possible paths while deleting the cycles, to compute the Bayesian inference. In this chapter, we extend this model as an Hybrid Risk Assessment Model. The Hybrid Risk Assessment Model also manages cycles in the input model.

In this work, we focus on the likelihood component of risk assessment. Thus, we use a simple impact function as output of the FRAMs, matching each compromised state with a fix impact value. Other works of the state of the art rather focus on the impact component. For example, Kheir *et al.* in [KDCB+09b] details how to use a dependency graph to compute the impact of attacks on Confidentiality, Integrity and Availability. This work is complementary to ours as we could add this kind of impact function after the FRAMs to compute a more accurate attack impact.

Models of the state of the art such as [FWSJ08] use Dynamic Bayesian Networks to monitor and predict future status of the system. However, such a model uses a sequence of Bayesian networks, which can be huge to process. The model we propose here separates the two objectives of dynamic risk assessment, and thus keeps only the past information necessary to explain all alerts (pertinent Dynamic Risk Correlation Models) and to update the models to evaluate potential futures (Future Risk Assessment Models). Moreover, the building process and exploitation of Dynamic Risk Correlation Models takes into account the temporality of raised alerts to determine attacks. Finally, contrary to other models based on Bayesian attack graphs, our model can distinguish several distinct simultaneous attacks in the alerts raised in a system, by analysing the appropriate Dynamic Risk Correlation Models.

Our experimental validation on the Hybrid Risk Assessment Model uses realistic simulated topologies that are far bigger than the state of the art. For example, Xie *et al.* assess their model on a topology of 3 hosts and 3 vulnerabilities [XLO+10], Liu and Man on a topology of 4 hosts and 8 vulnerabilities [LM05]. The real world examples used by Frigault and Wang in [FW08] contain at most 8 vulnerabilities on 4 hosts. The test network used by Poolsappasit *et al.* in [PDR12] contains 8 hosts in 2 subnets, but with only 13 vulnerabilities. Thanks to our polytree models, we successfully run our Hybrid Risk Assessment Model efficiently on simulated topologies with up to 70 states.

The Hybrid Risk Assessment Model is an extension of the Bayesian Attack Model that brings many enhancements. First, it separates the impact on the compromise probabilities of the past events and the possible futures. This allows an operator to know directly if an asset as a `high` compromise probability because it may have already been compromised, or because it is an easy next step of attack. Second, the Hybrid Risk Assessment Model is much less sensitive to the parameters (both of the Generic Attack Model and intrinsic) than the Bayesian Attack Model. No parameter change the ranking of the compromise probabilities of the states and a few parameters impacts significantly the absolute values of those probabilities. Even those sensitive parameters are much less impacting that those of the Bayesian Attack Model. Finally, the Hybrid Risk Assessment Model is more scalable than the Bayesian Attack Model.

6.9 Summary and conclusion

We present in this chapter a Hybrid Risk Assessment Model, combining the dynamic risk correlation and the future risk assessment analysis. This model is an extension of the Bayesian Attack Model

which aims at dynamic risk assessment. Hybrid Risk Assessment Model is subdivided into two complementary models: Dynamic Risk Correlation Models and Future Risk Assessment Models. Dynamic Risk Correlation Models are built according to dynamic security events, to update the compromise probabilities of states. We use these probabilities to build Future Risk Assessment Models, to compute the most likely futures. This combination of two complementary models separates the compromise status of assets between past attacks and likely futures.

Like we did for the Bayesian Attack Model, we studied precisely the sensitivity of the results of the Hybrid Risk Assessment Model, toward the parameters of the Generic Attack Model and the parameters introduced by the Hybrid Risk Assessment Model. We summarise in Table 6.3 the results of the sensitivity analysis of all parameters. We also give in this table the range of variation on which we conduct the sensitivity analysis for the given parameters. We present first the Generic Attack Model parameters then, after the double line, the Dynamic Risk Correlation Model parameters, finally, of the Future Risk Assessment Model parameters.

TABLE 6.3 – Sensitivity analysis of the parameters of the Hybrid Risk Assessment Model

Name	**Variation range**	**Ranking influence**	**Probability influence**
`false-positive`	$[0-1]$	No impact, except for the scenario containing a false negative in which it impacts `compromised` states.	Very low impact on its most frequently used interval ($[0-0.25]$). Then, low impact for both `compromised` and `not-compromised` states. Except for the attack scenario including a false negative for which the impact is more important.
`false-negative`	$[0-1]$	No impact on low values. Important impact above 0.05	Important impact for `not-compromised` states (strong increase). On very high values of the parameter, strong impact for all states.
`probability-attack-source`	$[0-1]$	Almost no impact.	Medium impact on probability of `not-compromised` states. Low impact on `compromised` states for the scenario with a false negative alert.
`probability-unknown-attack`	$[0-0.5]$	No impact.	Medium impact on the probabilities of `not-compromised` states. Low impact on the most frequently used interval ($[0-0.05]$). No impact for `compromised` states.
`max-number-no-info-to-explore`	$[[0-10]]$	No impact.	Important impact on the probabilities of `compromised` states when parameter < 3. Above 3, no impact. No impact for `not-compromised` states.
`max-number-no-info-to-keep`	$[[0-10]]$	No impact.	Important impact on the probabilities of `compromised` states when parameter < 2. Above 2, no impact. No impact for `not-compromised` states.
`max-number-no-alert-to-explore`	$[[0-10]]$	No impact.	Important impact on the probabilities of `compromised` states when parameter < 2. Above 2, no impact. No impact for `not-compromised` states.
`max-number-no-alert-to-keep`	$[[0-10]]$	No impact.	No impact
`nbSteps-possible-futures`	$[[0-4]]$	No impact.	Very low impact on the probabilities of `not-compromised` states, no impact on `compromised` states.
`probability-new-transition`	$[0-1]$	No impact.	High impact on the next possible futures `not-compromised` states.

The *ranking influence* describes the impact of the variation of a parameter on the rank of state's probabilities (on the whole parameter variation range, for all alert scenarios). This rank will determine the priorities of security operators in their system. The *probability influence* describes

the effect of the variation of the parameters on the absolute value of the state's probability. All the Hybrid Risk Assessment Model parameters have almost no to no impact on the ranking of the compromised states on their most frequently used interval, which is a comforting result. Note, however, that the `false-positive` and `false-negative` parameters may impact the rank of the `compromised` states, if their variation is too important, or if there are too many false negatives.

The `false-negative` parameter has an important impact on the absolute values of the compromise probabilities of `not-compromised` states for low values of the parameter. With a little uncertainty on such parameters (*e.g.*, 0.01), the variation of the absolute value of the probabilities is medium (*e.g.*, up to 0.1). Three parameters have a medium impact on the absolute values of the compromise probabilities of states on their variation interval: `probability-attack-source`, `probability-unknown-attack` and `probability-new-transition`. With a medium uncertainty on such parameters (*e.g.*, 0.1), the variation of the absolute value of the probabilities is also medium (*e.g.*, up to 0.1). Note that the `probability-attack-source` impacts the Dynamic Risk Correlation Model and the Future Risk Assessment Model, whereas the `probability-new-transition` parameter impacts only the possible futures. The `probability-attack-source` parameter can be estimated quite accurately with a risk analysis methodology, which gives the security risk of each state, according to its position in the system. The `probability-new-transition` must be set by the operator, according to the level of security needed in his system, and to the motivation of the attackers he his facing. For the pruning parameters of the Dynamic Risk Correlation Model (`max-number-no-info-to-explore`, `max-number-no-info-to-keep`, `max-number-no-alert-to-explore` and `max-number-no-alert-to-keep`) when they are above quite low values (2 or 3 depending of the parameter) they have no impact. So, in the Hybrid Risk Assessment Model, the ranking of states is not impacted by the variation of the parameters, and the absolute value of compromise probabilities is slightly impacted by a up to medium uncertainty on most parameters, except the `false-negative` parameter that is quite sensitive and might add false positives (*i.e.*, give strong compromise probabilities to `not-compromised` states), if its value is not little enough (*i.e.*, if the administrator says that the sensors might not raise an alert even if an attack has occurred, the model might give `high` probabilities to actually `not-compromised` states).

As a result, similar to the Bayesian Attack Model, we have built a dynamic risk assessment model that is suitable for any attack model (either cyclic or acyclic) that can be specified within the Generic Attack Model. Its results are accurate for the rank of the compromise probabilities of states, even with uncertainty on the parameters. In order to have exact absolute values on the compromise probabilities, the value of the `false-negative` parameter has to be accurately chosen. In addition to the Bayesian Attack Model, the Hybrid Risk Assessment Model brings many enhancements. First, it separates the impact on the compromise probabilities of the past events and the possible futures. Second, the Hybrid Risk Assessment Model is less sensitive to the parameters (both of the Generic Attack Model and intrinsic) than the Bayesian Attack Model. Finally, the Hybrid Risk Assessment Model is more scalable than the Bayesian Attack Model. Thus, this dynamic risk assessment model seems to be much more suitable for the computation of responses to the occurring attacks. In the next chapter, we will apply the Bayesian Attack Model and the Hybrid Risk Assessment Model to the cyber security domain, by building them from a topological attack graph.

Chapter 7

Application to cybersecurity: topological attack graphs

In chapter 5 and chapter 6 we develop two models, the Bayesian Attack Model and the Hybrid Risk Assessment Model to enable dynamic risk assessment and allow the computation of responses for occurring attacks. As these models are built from a Generic Attack Model, they can be applied to different domains. The only requirement is that the attacks are modelled using a graph-based model fitting into the Generic Attack Model. Thanks to its advantages over the Bayesian Attack Model, the Hybrid Risk Assessment Model seems more promising for the computation of responses to the occurring attacks. Nevertheless, the results of both these models need to be evaluated and compared on real use-cases.

In the cybersecurity domain, the potential attacks are generally specified with an attack graph, either logical, or topological. Logical attack graphs are more detailed than topological attack graphs, but they are also much bigger, which prevent them from being used for large information systems. As a result, the input Generic Attack Model we use to build and evaluate the dynamic risk assessment models is built from a topological attack graph. Thus, in this chapter, we first introduces topological attack graph and present how we can build them, either using existing attack graph engines, either by building a new engine, more scalable than the existing ones. Then, we apply the dynamic risk assessment models and assess their results on a use-case. Finally, we come back to the original challenge, the computation of responses for occurring attacks, and see how the dynamic risk assessment models we have built can integrate into the remediation methodology we defined in chapter 4 to compute responses to current attacks.

This chapter is part of the contribution described in subsection 1.3.3, the ability to use risk assessment models to support the computation of responses. It tackles the challenge described in section 1.2.4: how to use the results of the dynamic risk assessment models in order, for example, to compute responses to occurring attacks.

7.1 Topological attack graph generation

A topological attack graph is the base attack model from which we build our dynamic risk assessment models for cybersecurity. We first formally define this model, then present how it can be generated.

7.1.1 Definitions

Definition 37 *A **topological attack graph*** TAG *is a directed graph* TAG(TN,AS) *where:*

- $TN = \{\text{TN}\}$ *is a set of **topological nodes**: the assets of an information system,*
- AS *is a set of **attack steps**, the edges representing that an attack allows an attacker to move from the parent topological node to the child topological node.*
  - *Each attack step has a **type** of attack, describing how the attacker can move between nodes (exploitation of a vulnerability, credential thief, etc.).*
  - *Depending on the type of attack, each attack step is associated with a set of **conditions*** [c].
  - *Some attack steps are associated with a **sensor** that may raise an alert indicating that this attack has been detected.*

A topological attack graph can be generated with an attack graph engine, for example MulVAL [OGA05a] or TVA [JNK+11], as detailed in subsection 7.1.2. Topological nodes represent, for example, a host, an IP address or a computer cluster. In usual attack graphs, there are a few (*e.g.*, 2 or 3) different types of attacks (*e.g.*, the exploitation of vulnerability on a remote host, the local exploitation of a vulnerability).

Definition 38 *A **condition*** c *is a fact that needs to be verified, for an attack step to be possible. It is associated with a probability of success* P(c).

The condition fact is, for example, "a vulnerability is exploited on the destination host".

Definition 39 *A **sensor*** s *of an attack step or of a topological node is an oracle issuing an alert when the attack step has been detected or the topological has been detected as* `compromised`*. It is associated with a false negative and a false positive rate.*

A sensor represents, for example, a Network or Host Intrusion Detection System, or a System Event Management.

7.1.2 Topological attack graph generation from existing attack graph engines

In order to be able to use any type of attack graph engine to build the Bayesian Attack Model, we chose a generic definition of a topological attack graph: a graph of topological assets as nodes, and attack steps as edges. This graph can thus be generated either by topological attack graph engines, or by logical attack graph engines, as described below. We implemented the generation of our Bayesian Attack Model starting from both types of attack graph.

Using topological attack graph engines Starting from a topological attack graph engine, such as TVA [JNO05], we simply transcribe the graph in the new format, with IP addresses as topological nodes, and exploits as attack steps (with the exploitation of a vulnerability as a condition).

Using logical attack graph engines Starting from a logical attack graph engine, such as MulVAL [OGA05a], we first need to define topological nodes in the graph (for example, we chose *execCode(host, privilege)*: the acquisition of a privilege on a host). Then, we process the graph to find how these topological nodes are bound together, also extracting the preconditions needed by the subgraph between the topological nodes. This subgraph will be *summarised* in the topological attack graph, as an Attack Step.

7.1.3 Hierarchical attack graph engine architecture

Several attack graph engines exist. They have been developed either by research laboratory or by companies. However, in practice, this software suffers from several limitations:

- The first limit that can be noticed is that all the engines have either a logical or a topological view of the attack. None of them is a hybrid view combining the advantages of the topological view (much more concise view of the attack graph) and the advantages of the logical view (full knowledge of the conditions that are necessary for an attack and how they are combined). The topological attack graph engines do not keep all the logical conditions necessary to generate the attack steps. The logical attack graph engines need further computations to extract from the logical graph its topological structure.
- The existing attack graph engines are not scalable. They cannot be distributed and thus are limited in the number of hosts or vulnerabilities they can handle.
- The commercial products does not allow to change the rules describing the attacks that can be modelled in the attack graph.
- The only available Open Source Software, MulVAL, uses an obscure mix of several languages (mainly Datalog, Java, C++, Bison *etc.*), has strange behaviours (*e.g.*, the results of the Datalog engine are written in an output file which can be huge and that is then parsed) and has a syntax not easy to understand.

As the currently existing attack graph engines are not satisfying we implement a new attack graph engine, which is topological but keeps logical conditions. It can be distributed on a cluster of servers and thus is scalable. Moreover, it relies on topological hierarchy to improve performances.

7.1.3.1 Architecture

The high-level architecture of SAGE, the Scalable Attack Graph Engine is shown in Figure 7.1. This engine has two types of inputs:

Static engine configuration These inputs allow to configure the attack graph engine, according to the preferences of the user. They specify the attacks taken into account (in the attack rules description file) and the execution parameters.

Dynamic topological inputs These inputs describe the assets of the information system (network topology) and their connexion (flow matrix). This topology can be updated according to changes in the information system, in order to update the attack graph.

The applications obtain the computed topological attack graph using a REST API. The topological attack graph is constituted of a set of attack steps, each attack step being associated with its required conditions.

7.1.3.2 Technologies involved

The Scalable topological attack graph Engine relies on different topologies, to store, process and compute the topological attack graph. Figure 7.2 summarises these technologies.

- The topological data (network topology, flow matrix, vulnerabilities) are stored in a graph database (OrientDB) which can be distributed. A Python script parses topological data from vulnerability scanners and CSV topological files and add it to the graph database. The graph database also receives the results of the topological attack graph computation.

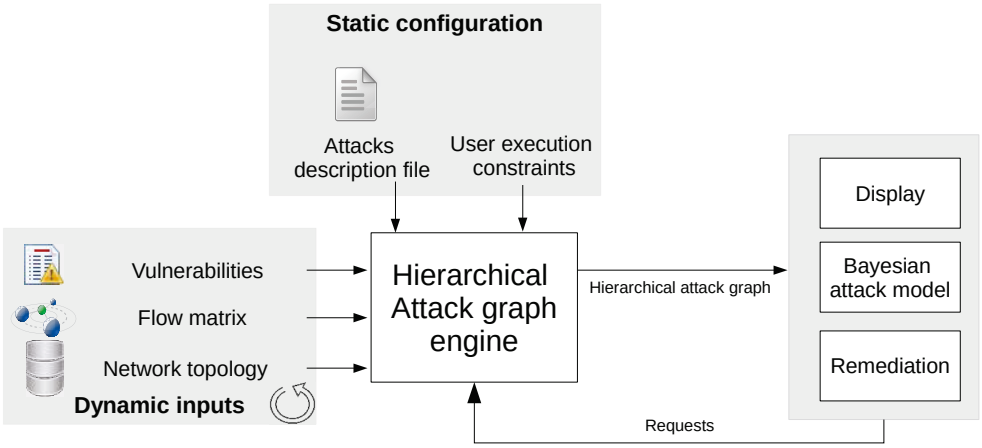


FIGURE 7.1 – Scalable Attack Graph Engine architecture

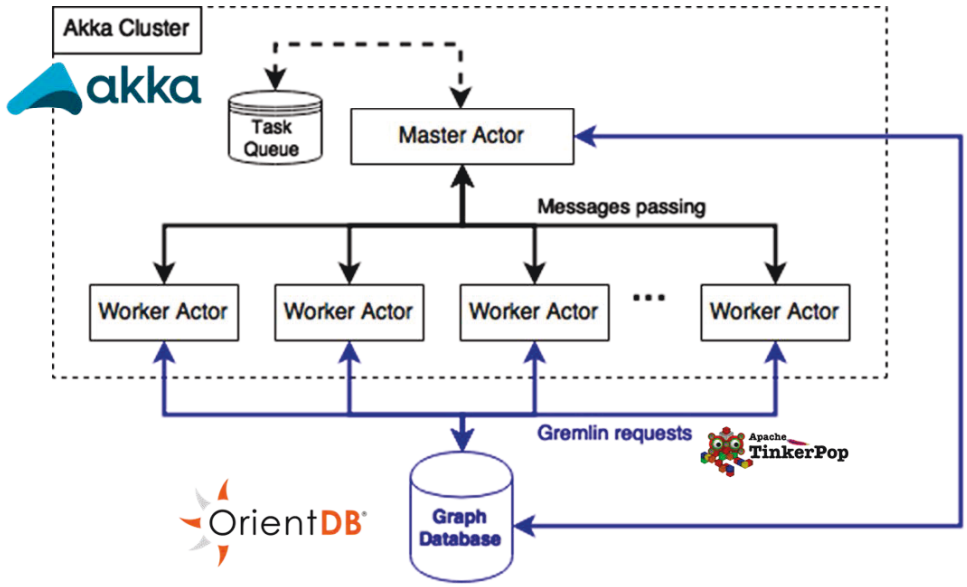


FIGURE 7.2 – Scalable Attack Graph Engine technologies

- The graph database is queried through the language Gremlin of the framework TinkerPop. TinkerPop is a framework for Graph Databases that brings tools (*e.g.*, graph query language, proxy server *etc.*) and aims at becoming a standard for all graph database.
- The attack graph engine core is developed in Scala, using the Akka framework for distributed computations. This framework relies on the concept of actors. *Master actors* orchestrate the *tasks* that have to be done. *Worker actors* do the task that master send to them with *messages*.
- Scala also allows to use a Domain Specific Language (DSL) to write the attack rules.

7.2 Building a Generic Attack Model from a topological attack graph

We have several attack graph engines that we can use as input for dynamic risk assessment: MulVAL, Cauldron (TVA) and SAGE. As described in subsection 7.1.2, we can generate a topological attack graph (*cf.* Definition 37) using MulVAL and Cauldron. SAGE has been designed in order to generate a topological attack graph. But, in order to use the dynamic risk assessment models of chapter 5 and chapter 6, we need to fit the topological attack graph model into the Generic Attack Model.

7.2.1 Generic Attack Model nodes

The nodes of the Generic Attack Model are built from the nodes of the topological attack graph in that way:

State: The topological nodes of the topological attack graph are the states of the Generic Attack Model.

Transition: The attack steps of the topological attack graph are the transitions of the GAM

Condition: The condition of the topological attack graph is the condition of the GAM. They are attached to each transition.

Sensor: The sensors of the topological attack graph are those of the GAM. They are associated with either the states (topological node), or the transitions (attack step).

7.2.2 Generic Attack Model probability tables

In addition to the nodes, the Generic Attack Model can be customised with conditional probability tables associated with states and transitions.

7.2.2.1 Conditional probability tables of states

A GAM state node (representing a topological node) has one parent for each type of attack that can be used to compromise it. Its probability table represents a *noisy*-`OR`. At least one `succeeded` attack step is needed to make this node `compromised`. If no known attack step has `succeeded`, there is still a little probability that an attack of this topological node may be an unknown one (*e.g.*, a 0-day). We denote it by *pua* (*probabilityUnknownAttack*). Such a conditional probability table is described in Table 7.1. The first two lines represent all possible states of the parents (GAM transitions nodes). The last two lines contain the probabilities of each state of the GAM state node according to the states of its parents.

Table 7.1 – Conditional probability table of a Bayesian state node

sn 1	`succeeded`	`failed`	`succeeded`	`failed`
asn 2	`succeeded`		`failed`	
tn				
`compromised`	1	1	1	$1 - pua$
`not-compromised`	0	0	0	pua

with $pua = probabilityUnknownAttack$.

7.2.2.2 Conditional probability tables of transitions

A GAM transition node (representing an attack step) has two types of parents: (1) one state node, the source of the attack, which is required to perform the attack, (2) one or more condition nodes: the conditions of the attack step. Depending on the type of attack modelled, the condition nodes may not exist for the attack node.

The *probabilityNewAttackStep* parameter represents the fact that an attacker may have reached his objective. Even if he has `compromised` the topological node and conditions are `verified`, it is not certain that he will attempt to propagate through the execution of a new exploit. We describe in Table 7.2 the conditional probability table of a GAM transition nodes, for the exploitation of a vulnerability.

TABLE 7.2 – Conditional probability table of a GAM transition node "exploitation of a vulnerability"

tn	`compromised`	`not-compromised`	`compromised`	`not-compromised`
cn	`verified`		`not-verified`	
asn				
`succeeded`	pnt	0	0	0
`failed`	$1 - pnt$	1	1	1

Caption: pnt: `probability-new-transition`.

7.2.2.3 Probability of success of conditions

There is only one main type of conditions in usual topological attack graphs: the exploitation of a vulnerability of the target topological node. For such conditions, in our experiments, we use an approximation of the probability of successful exploitation using information coming from the Exploitability Metrics of the Common Vulnerability Scoring System (CVSS) [FIR15]. It is deduced from (1) the Attack Complexity (AC), (2) Privileges Required (PR), (3) and User Interaction (UI) values, as well as the Attack Vector (AV), which is taken into account when constructing the topological attack graph.

7.2.3 Generic Attack Model parameters default values

We will now present here the value of the Generic Attack Model parameters that we use for the cybersecurity experimental validation of the dynamic risk assessment models and explain their default values. These parameters are used to fill the conditional probability tables of the Generic Attack Model.

The use cases in which we want to use the dynamic risk assessment models represents typical critical information systems. Such a system is managed by a security operator who often uses a vulnerability scanner. Thus, most vulnerabilities are known, but there is still a little chance (*e.g.*, 0.1%) that a very motivated attacker knows a 0-day, a non-public vulnerability. As the system contains known unpatched vulnerabilities, sensors are deployed to raise an alert as soon as one of the vulnerabilities is exploited. These sensors have a medium chance (*e.g.*, 2%) to raise false positives, when an attack do not succeed while being detected. However, for the known vulnerabilities for which a sensor is deployed, the probability of having a false negative is lower (*e.g.*, 0.5%). The security operator knows that most attacks may come from the Internet (*e.g.*, probability of the Internet being a source of attack of 70%), even if internal hosts may also be a new source of attacks (undetected phishing, malicious employee, *etc.*) with a lower probability (*e.g.*, 10%)).

Table 7.3 summarises the default values of the parameters used to build the Generic Attack Model and explanation for the cybersecurity use-case: `probability-unknown-attack`, `false-positive`, `false-negative`, `probabilityInternet`, and `probability-attack-source`. Each parameter is associated with its description and the default value that was chosen for the use-cases.

TABLE 7.3 – Default values of the parameters for the cybersecurity use case

Parameter name	Default value	Meanings	Default value explanation
`probability-unknown-attack`	0.001	Probability that an unknown attack occurs.	Very small probability of having a 0-day, a unknown vulnerability.
`false-positive`	0.02	False positive rate of each sensor.	Sensors may raise an alert, even if the attack has not succeeded.
`false-negative`	0.005	False negative rate of each sensor.	This value is smaller as it only concerns vulnerabilities for which a sensor has been deployed.
`probability-Internet`	0.7	A priori probability of an attack coming from the Internet.	The internet is the main source of attacks. Thus, 70% of chances of being a source of attack, 30% not to be a source.
`probability-attack-source`	0.1	A priori probability of an attack issued from an internal host.	An internal host may issue an attack. Thus, 10% of chances of being a source of attack, 90% not to be a source.

7.3 Experimental validation of the dynamic risk assessment models for cybersecurity

Once we have built the Generic Attack Model for the cybersecurity use case, we have the main input of the dynamic risk assessment models. So, we build those models, with their parameters in the default values presented in Table 5.3 for the Bayesian Attack Model, and in Table 6.1 and Table 6.2 for the Hybrid Risk Assessment Model. Then, we evaluate the accuracy of the results of the models for this cybersecurity use case. The Bayesian Attack Model and Hybrid Risk Assessment Model are implemented in Java, using the SMILE Bayesian Network library [Dru99].

7.3.1 Accuracy evaluation

7.3.1.1 Methodology

To evaluate the accuracy of the results of the dynamic risk assessment models (*i.e.*, how close the compromise probabilities are to the truth), we simulate attack scenarios of 5 successive steps on random information systems topologies and compare the theoretical results with the outputs of the dynamic risk assessment models. We generate the random realistic topologies, as shown in Figure 7.3, containing various numbers of hosts, in 7 subnets. These topologies are representative of a real network in which defense in depth is implemented: all the hosts of a subnet have access to the hosts of a

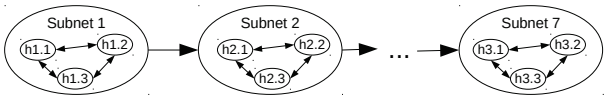


FIGURE 7.3 – Information system network topology for simulations

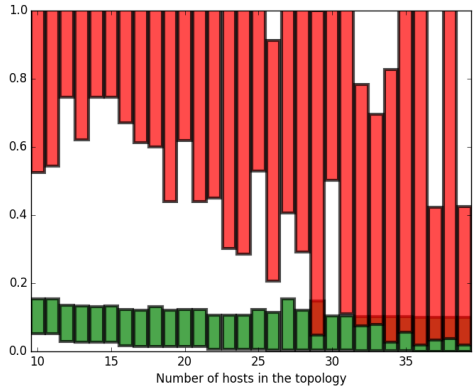
We compare the theoretical results known in the attack scenarios with the results of the compromise probabilities of the dynamic risk assessment models (Bayesian Attack Model and Dynamic Risk Correlation Model). In each scenario, we know the nodes that are compromised and healthy, *i.e.*, nodes with a theoretical probability of respectively 1 and 0. We can then assess if the dynamic risk assessment models probabilities of `compromised` nodes are close to 1, and if the dynamic risk assessment models probabilities of `not-compromised` nodes are close to 0.

7.3.1.2 Bayesian Attack Model

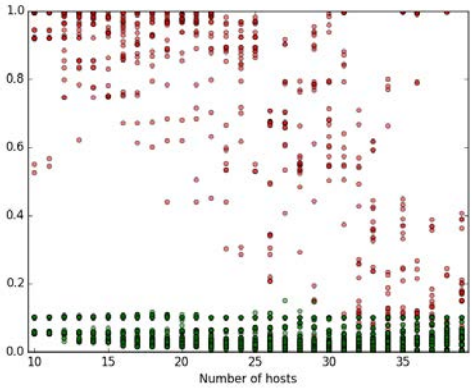
The results of this study for the Bayesian Attack Model are shown in Figure 7.4. These figures present the results of the accuracy evaluation of the Bayesian Attack Model on random simulated topologies. The green bars or dots represent the hosts not as `not-compromised` in the attack scenario. The red bars or dots represent the hosts not as `compromised` in the attack scenario. In other words, this graph shows the errors (in terms of distance to the theoretical values 1 and 0) of `compromised` and `not-compromised` nodes. Figure 7.4a shows a bar from the minimum to the maximum of those probabilities values. Figure 7.6b shows a dot for the probability of each host of a simulation scenario. When these figures show a large free space between the probability of `compromised` hosts and the probability of `not-compromised` hosts, this means that it allows to distinguish exactly `not-compromised` and `compromised` hosts, for example with a boundary at the probability of 0.5.

FIGURE 7.4 – Accuracy of the results of the Bayesian Attack Model according to the number of hosts

(A) Minimum and maximum probability of `compromised` and `not-compromised` hosts.



(B) All probability values of `compromised` and `not-compromised` hosts.



These figures show that the Bayesian Attack Model is quite accurate for a low number of hosts (until 25 hosts). Then, we can see a gradual deterioration of the results with the increase of the number of hosts. From 30 hosts, this deterioration causes a mix between the probabilities of the hosts that are actually `compromised` and the ones that are `not-compromised`. As a result, in such cases, the Bayesian Attack Model introduces false negatives or false positives.

7.3.1.3 Hybrid Risk Assessment Model

We perform the same simulations for the Hybrid Risk Assessment Model. Contrary to the Bayesian Attack Model, this model distinguishes the hosts that may have been `compromised`, thanks to the Dynamic Risk Correlation Models, and the ones that may be compromised in the near future, thanks to the Future Risk Assessment Models. We thus perform the accuracy evaluation for both types of results.

Figure 7.5 shows the results of the accuracy evaluation of the Dynamic Risk Correlation Models. Contrary to the Bayesian Attack Model, the results for the Dynamic Risk Correlation Model are rather identical, independently of the number of hosts there are in the topology. Moreover, these figures show a large separation (whitespace) between the probability of `compromised` hosts and the probability of `not-compromised` hosts, for all values. This means that it allows to distinguish exactly `not-compromised` and `compromised` hosts. This means that there are no false negatives nor false positives introduced by the Dynamic Risk Correlation Models.

FIGURE 7.5 – Accuracy of the results of the Dynamic Risk Correlation Model according to the number of hosts

(A) Minimum and maximum probability of `compromised` and `not-compromised` hosts.

(B) All probability values of `compromised` and `not-compromised` hosts.

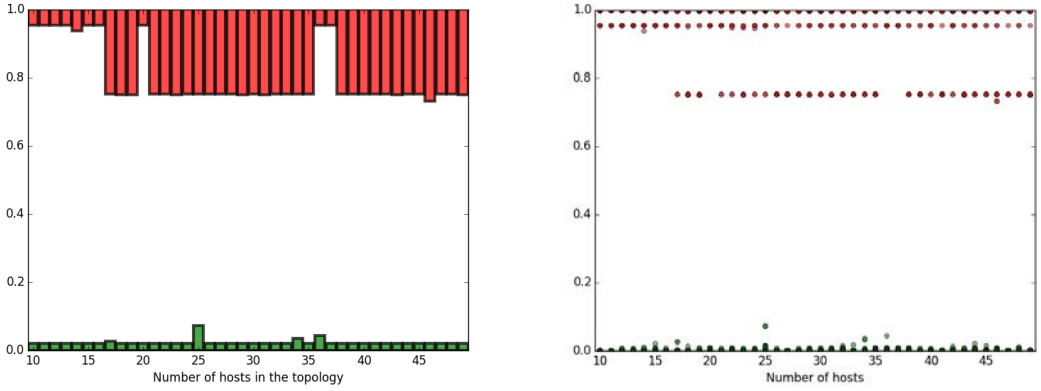
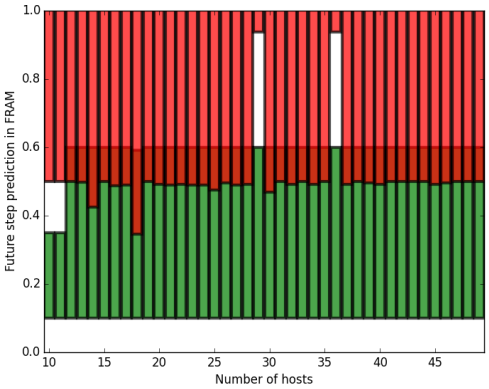


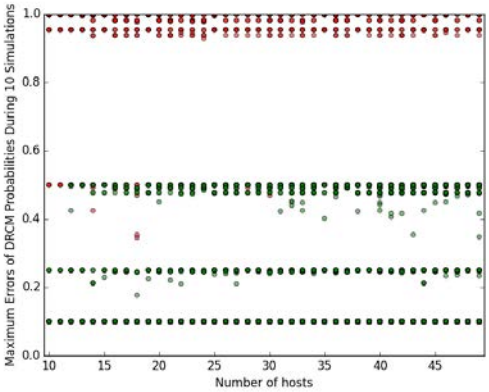
Figure 7.6 shows the results of the accuracy evaluation of the Future Risk Assessment Models. As for the Dynamic Risk Correlation Models, the results for the Future Risk Assessment Models are rather identical, whatever the number of hosts there are in the topology. These figures show that obviously it is much more complicated to predict the potential futures that identify the past attacks. There is no longer a large free space between the probability of `compromised` hosts and the probability of `not-compromised` hosts. Indeed, even if the future next step that the attacker chooses in our simulated scenarios is one of the most likely (easiest CVE vulnerability to exploit), it is not the only one likely future that can be done. Thus, there is a superimposition of the probably next `compromised` and `not-compromised` hosts on the probabilities from 0.5 to 0.6. However, apart from the hosts that have a probability from 0.5 to 0.6, for which the operator has to investigate deeper, the other hosts (which constitute the majority) can be clearly distinguished. The ones with a probability lower to 0.5 are `not-compromised`, the ones with a probability above 0.6 are `compromised`.

FIGURE 7.6 – Accuracy of the results of the Future Risk Assessment Model according to the number of hosts

(A) Minimum and maximum probability of `compromised` and `not-compromised` hosts.



(B) All probability values of `compromised` and `not-compromised` hosts.



7.3.1.4 Results comparison

The Bayesian Attack Model provides satisfying results when the number of hosts is not too high (<25 hosts): it allows to recognise properly the hosts that are compromised. However, for a higher number of hosts it does not allow to distinguish for sure `compromised` from `not-compromised` hosts.

On the contrary, the Hybrid Risk Assessment Model has results that are similar independently of the number of hosts. For past compromised hosts, the Dynamic Risk Correlation Models enables to distinguish `compromised` hosts from `not-compromised` hosts with a very high success rate, as there is a large difference between the `compromised` hosts (probability > 0.7) and the `not-compromised` hosts (probability < 0.1). Moreover, the Future Risk Assessment Models allow to predict the potential futures (if the attacker choose one of the easiest next attack steps) with a relatively high success rate. Indeed, the majority of hosts have a probability that is below 0.5 (*i.e.*, surely a `not-compromised` host) or above 0.6 (*i.e.*, surely a `compromised` host). Few hosts have their probability from 0.5 to 0.6. For those hosts, the model cannot distinguish whether or not they are the likely future that will be chosen by the attacker. Thus the security operator should protect the likely next step targeting its valuable assets, to prevent the attacks he wants to prevent.

7.3.2 Use-case validation

7.3.2.1 Use-case presentation

In order to validate the accuracy of the results, while keeping the scenarios simple for explanations, we implemented a real infrastructure of 11 virtual machines, for a total of a hundred vulnerabilities. A host (that will be called host A, thereafter) can be attacked from the Internet, and can attack the other hosts G to J of its subnetwork. The latter hosts can attack hosts A, C and D. This network topology is representative of a real information system, where an ingress firewall (host K) protects the LAN (E to J), and where publicly accessible servers are put in a demilitarised zone (A to D). The topological attack graph used to populate the Bayesian Attack Model has been

generated from a report of the vulnerability scanner Nessus on this infrastructure.

From this network topology we apply 5 scenarios summarised in Table 7.4. The attack is carried out through three attack steps. In scenarios 1 to 3, attack steps are detected and alerts are generated. Scenarios 4 and 5 represent detection anomalies.

TABLE 7.4 – Simulation scenarios

S	$I \rightarrow A$	$A \rightarrow G$	$G \rightarrow D$	**Comment**
1	✓	×	×	First alert
2	✓	✓	×	Second alert
3	✓	✓	✓	Third alert
4	✓	*O*	✓	`no-info` for the second step (=no sensor)
5	✓	×	✓	`no-alert` for the second step (possible false negative for the second step or false positive for the third step)

Caption S: Scenario number; $I \rightarrow A$: Attack from the Internet to host A; $A \rightarrow G$: Attack from host A to host G; $G \rightarrow D$: Attack from host G to host D; O: No values set (=`no-info`); ✓: Sensor nodes set to `alert` (an alert has been issued for this attack step); ×: Sensor node has been set to `no-alert`.

These scenarios represent the dynamic evolution of a system with different possible situations:

- Scenarios 1, 2, then 3: Normal evolution of an attack during the time.
- Scenarios 1, then 4: Evolution of an attack in which an attack step cannot be detected (no sensor for this step).
- Scenarios 1, then 5: Evolution of an attack in which an attack step has not been detected while there was a sensor for this step.

We assume that the alerts given by the sensors are binary (`alert`, `no-alert`), *i.e.*, we do not have alert confidence.

7.3.2.2 Results and analysis for the Bayesian Attack Model

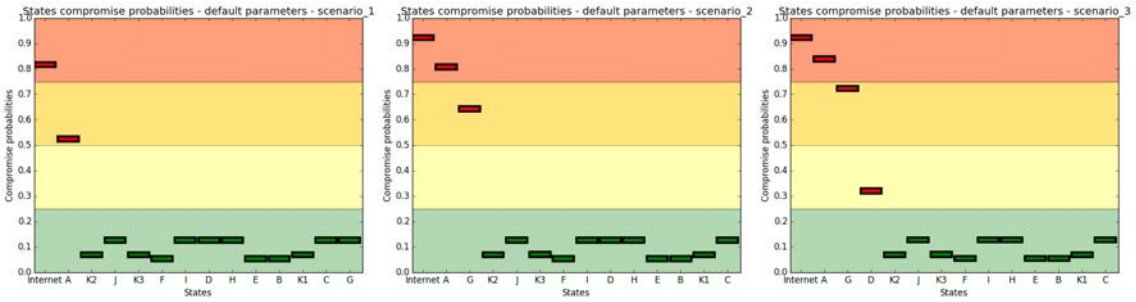
The results of the compromise probabilities of each topological node calculated by the Bayesian Attack Model, for each scenario are shown in Figure 7.7.

These figures show the five scenarios of the use case presented in subsubsection 7.3.2.1. The bars represent the hosts of the topology. Their ordinate is the compromise probability of the abscissa host. The background colours give an idea of the threshold that could be taken to define the compromise risk level of the hosts. For example, the hosts with a compromise probability under the lowest line ($probability \leq 0.25$) have a `low` risk of being compromised, above the lowest line ($0.25 < probability \leq 0.50$) have a `medium` risk, above the second line ($0.50 < probability \leq 0.75$) have a `high` risk, and above the upper line ($0.75 < probability$) have a `critical` risk of being compromised.

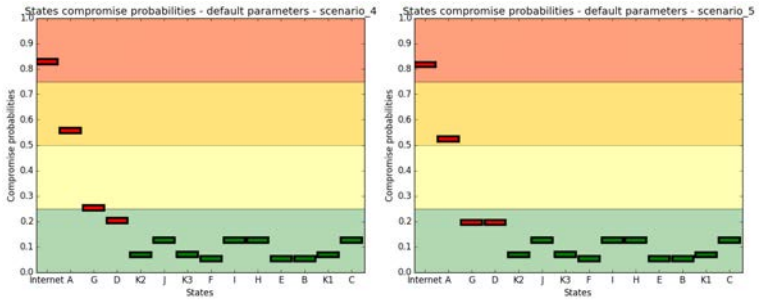
In the scenarios 1, 2 and 3, the sensors corresponding to the 3 steps attack are set progressively. Each new sensor set as `alert` confirms the attack that is currently happening and increases the compromise probability of the previous and future states. For example, in scenario 3, the Internet, and the 3 victim hosts are in the `high` or `critical` risk zone. In scenario 4, and scenario 5, when there is a missing alert or a false negative / false positive, the probabilities of a currently happening attack are lower, but higher than the residual risk and than the probabilities of scenario 1, that should precede this state. So, a security operator may investigate the appropriate machines to confirm or disprove the attack.

FIGURE 7.7 – Results of the Bayesian Attack Model for use-case scenarios

(A) Scenario 1 (B) Scenario 2 (C) Scenario 3



(D) Scenario 4 (E) Scenario 5



7.3.2.3 Results and analysis for the Hybrid Risk Assessment Model

The results of the compromise probabilities of each topological node calculated by the Hybrid Risk Assessment Model, for each scenario are shown in Figure 7.8.

These figures show the five scenarios of the use case presented in subsubsection 7.3.2.1. Like for the Bayesian Attack Model, the bars represent the hosts of the topology.

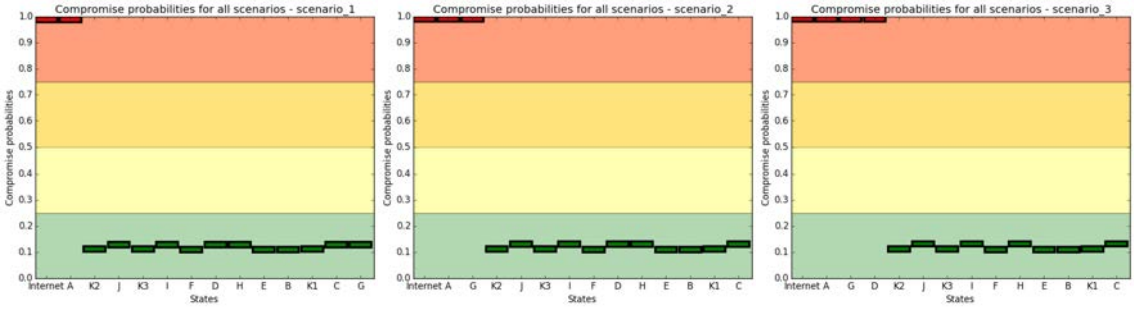
In the scenarios 1, 2 and 3, the sensors corresponding to the 3 steps attack are set progressively. Each new sensor set as `alert` confirms the attack that is currently happening and increases the compromise probability of the previous and future states. For example, in scenario 3, the Internet, and the 3 victim hosts are in the `critical` risk zone. In scenario 4, there is a missing alert, the probabilities of a currently happening attack are nearly identical to the scenario 3 as there is `no-info` for an attack step, which is surrounded by two attack steps. In scenario 5, there is a false negative / false positive, the probabilities of a currently happening attack are a bit lower, but stays `high` for the four hosts that may be compromised. So, a security operator needs to investigate the appropriate machines to confirm or disprove the attack.

7.4 Computation of cybersecurity responses using dynamic risk assessment models

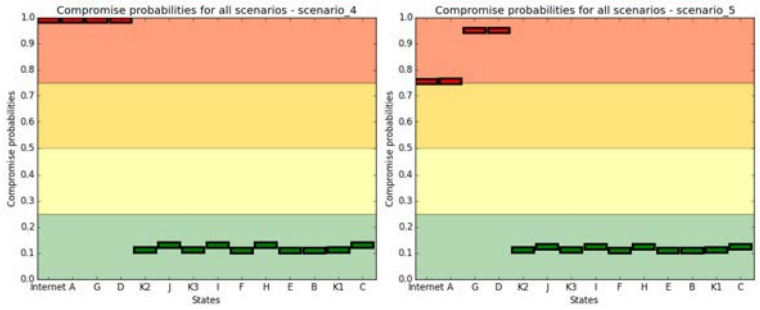
We have studied in detail the probabilistic behaviour of the two dynamic risk assessment models presented in chapter 5 and chapter 6. We have validated its usage for cybersecurity by building them from a topological attack graph in section 7.2 and its results on a use-case in section 7.3. We

FIGURE 7.8 – Results of the Hybrid Risk Assessment Model for use-case scenarios

(A) Scenario 1 (B) Scenario 2 (C) Scenario 3



(D) Scenario 4 (E) Scenario 5



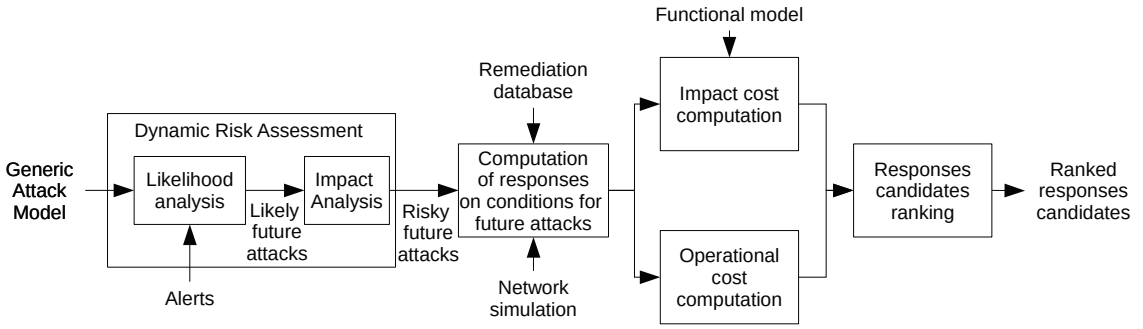
will now come back to the original problem (the risk analysis and computation of responses for occurring attacks) and study how we can use our models for this.

7.4.1 Response computation process

In section 4.2, we present a remediation process to prevent potential attacks. This process can be generalised to response computation for potential and occurring attacks, as shown in Figure 7.9. From a Generic Attack Model, we conduct the dynamic risk assessment, composed of the likelihood analysis and the impact analysis. Then, from the risky possible future attacks, we compute the responses from the conditions of future attacks. These responses candidates are then ranked by taking into account the response impact cost, and the response operational costs.

The main differences of this process with the one for remediation computation, presented in Figure 4.4 are:

Extraction of attack paths and scoring: The extraction of attack paths and scoring for the remediation process is replaced, in the dynamic response computation process, by the dynamic risk assessment, that can be done, either by the Bayesian Attack Model or by the Hybrid Risk Assessment Model. Indeed, the dynamic risk assessment combines (1) the likelihood analysis, computing the future attacks that are the most likely to happen, according to the occurring attacks and the threat context, and (2) the impact analysis, computing from those likely attacks the ones that can cause the most damages. The outputs of this analysis are the risky future attacks, that are an equivalent for dynamic risk analysis of the attack paths with a high score.



Remediation of preconditions and sufficient preconditions algorithm: These two components allowing to compute the remediation candidates are replaced, in the dynamic response computation process, by the computation of responses on conditions for future attacks. This new component takes as input a possible future attack (a path in the Bayesian network) and computes responses allowing to prevent or decrease the likelihood of this attack. An example of such a component is presented in subsection 7.4.2. As for the remediation computation, this component may rely on a remediation database, and on a network simulation tool, to compute the network-based remediations.

Apart the two components presented above, the end of the response computation process is identical to the remediation process detailed in section 4.2. Moreover, we have also already presented the first component (dynamic risk assessment) in chapter 5 and chapter 6. Thus, the only component that we still need to detail is the second component: *Computation of responses on conditions for future attacks*, which we will call hereafter the response computation component.

7.4.2 The response computation component

7.4.2.1 Inputs and outputs

The inputs of the response computation components are the outputs of the dynamic risk assessment. So, it is a set of risky future attacks for which we know the likelihood, which we want to decrease with the responses. The future attacks in the Bayesian Attack Model are only described by the compromise probabilities of the states. In the Hybrid Risk Assessment Model, as there are separate models for the analysis of possible futures, the description of future attacks is much more detailed. Future attacks are described by a set of Bayesian models issuing from likely sources of attack and targeting an asset that has high risk. As this description of likely future attacks is much richer, we will build the rest of our response computation component on the outputs of the Hybrid Risk Assessment Model, rather than on those of the Bayesian Attack Model. Similarly to the remediation computation component of section 4.2, the response computation component can rely on a remediation database and a network simulation tool, to compute the appropriate responses to prevent an attack.

The output of the response computation component is a list of response candidates, composed of a disjunction of conjunction of response actions. Each candidate allows to prevent (entirely or partly, according to the response computation component) a risky future attack. The format of these responses is identical to the one of the remediation presented in subsection 4.2.2.

7.4.2.2 Example of response computation component

We will present here only one simple possible solution that could be used to compute the responses, taking inspiration of what we presented for remediations and leveraging the advantages brought by the Bayesian networks of Future Risk Assessment Model.

The solution we propose to compute responses to attacks is the following. The goal of this process is to protect a risky asset from the likely next steps of attacks. As presented in subsection 6.3.4, we potentially have several Future Risk Assessment Models targeting the risky asset. Moreover, in those Future Risk Assessment Models, there may be several FRAM State nodes related to this risky asset. We thus have several places (corresponding to several paths) that need to be checked, in order to protect the risky asset. However, for each of these FRAM State nodes (potentially in different Future Risk Assessment Models) we have a probability of future compromise representing how likely this path to the risky asset is. As the future compromise likelihood we present to the operator is the maximum of all FRAM State nodes related to an asset, in order to decrease this reconciled probability, we need to first decrease the probability of the FRAM State node with maximum probability or completely delete its attack path.

We present in Figure 7.10 an algorithm computing responses to protect a risky asset. It is called on one risky asset that we want to protect, and for which an acceptable risk has been set. As we suppose the impact constant during the computation of the response candidates, the only parameter of the risk that can vary is the likelihood. As a result, reducing the risk of an asset with responses is identical to reducing its future compromise probability. Thus, while the risk of this asset in Future Risk Assessment Models is not low enough, we try to correct the FRAM State Node with the maximum probability. The correction of a FRAM State Node is done by computing the possible responses that prevent the most important condition (*i.e.*, the one that is the *most required* for an attacker on the path) that can be corrected on the path from the FRAM Attack Source Node to the FRAM State Node. We sort the conditions from the most important to the less important, by multiplying, in all FRAM Transition Nodes, the conditions initial probabilities with the value of the `succeeded` state in conditional probability tables with the condition `failed` and all other parents as `succeeded`, and sorting them in ascending order. For example, a value of 0 for the `succeeded` state of a FRAM Transition Node with a `failed` condition means that the `failed` condition is necessary to reach the state. When there are several conditions on the path that have the same importance, we select the one attached to the transition that is the closest to the FRAM Attack Source. Indeed, this will allow to stop the attacker as early as possible, while protecting at best the risky asset.

Note that this algorithm follows a greedy approach and is thus rather efficient (*i.e.*, the $computeReponseToProtectFramNode$ algorithm is linear in the number of conditions on the path from the FRAM Attack Source Node to the FRAM State Node of the risky asset and the $computeResponseCandidates$ algorithm is linear in the number of FRAM State Nodes related to the risky asset), but it does not consider all possible cases. As a result, the computed response candidates may not include the global optimal response. However, this algorithm allows to have an insight of the possibilities that can give the Hybrid Risk Assessment Model for response computation. The Bayesian network based models allow to compute very accurately the impact of the responses actions on the attack likelihood.

7.5 Conclusion

In this chapter, we have applied the dynamic risk assessment models we developed in chapter 5 and chapter 6 to the cybersecurity domain. In order to do that, we first needed to populate a Generic Attack Model with cybersecurity attacks. This was done using a topological attack graph engine.

FIGURE 7.10 – Algorithm computing the response candidates

```
1: function computeResponseCandidates(riskyAsset, acceptableRisk, frams)  ▷ Compute response candidates protecting a risky asset.
2:     responseCandidates ← [[]]
3:     currentRisk ← frams.getRisk(riskyAsset)
4:     while currentRisk > acceptableRisk do  ▷ The risk is too high for riskyAsset
5:         nodeToProtect ← frams.getRelatedFramNodeWithMaxProba(riskyAsset)  ▷ Get the FRAM State Node related to riskyAsset, with the maximum probability
6:         reponses ← computeReponseToProtectFramNode(nodeToProtect,frams)  ▷ Compute responses to protect nodeToProtect, cf. below
7:         if empty(reponses) or otherStopCondition then  ▷ Add here stop conditions that prevent the protection of riskyAsset until acceptableRisk
8:             return responseCandidates
9:         else
10:            responseCandidates ← conjunctionOfSets(responseCandidates,reponses)
11:            currentRisk ← frams.getRisk(riskyAsset)
12:        end if
13:    end while
14:    return responseCandidates
15: end function
16: function computeReponseToProtectFramNode(nodeToProtect, frams)  ▷ compute responses to protect a FRAM node by preventing the most effective condition, and apply the response to frams.
17:    fram ← nodeToProtect.getRelatedFram()
18:    sortedConditions ← fram.getSortedConditionsOnPathFromSourceTo(nodeToProtect)  ▷ get the conditions ranked by impact on nodeToProtect (from the most necessary to the least necessary to attack this node).
19:    for condition in sortedConditions do
20:        if condition.canBeCorrected() then
21:            frams.correctConditionAndUpdateProbabilities(condition)
22:            return condition.getPossibleResponses()  ▷ returns the list of possible response to correct this condition.
23:        end if
24:    end for
25:    return [[]]
26: end function
```

We defined a topological attack graph and presented how we can generate a topological attack graph from an existing attack graph engine. But as the existing engines are not efficient enough, we implemented a new attack graph architecture, SAGE (the Scalable Attack Graph Engine) that can be distributed on several computation nodes. As we can build a Generic Attack Model from a topological attack graph, we experimentally validated the results of the Bayesian Attack Model and of the Hybrid Risk Assessment Model for cybersecurity.

On the simulation-based accuracy evaluation of the results, the results of the Hybrid Risk Assessment Model are far better than the ones of the Bayesian Attack Model. The Hybrid Risk Assessment Model allows to distinguish accurately the `compromised` from the `not-compromised` states for the analysis of past alerts, for any number of hosts, whereas the Bayesian Attack Model is less precise. For the analysis of possible futures (not possible in the Bayesian Attack Model), the results of the Hybrid Risk Assessment Model are obviously worse, as it is nearly impossible to predict the future attacks for sure, but it allows to distinguish the attacks that are the most likely

to happen.

For the use-case base validation, the results are the same, the ones of the Hybrid Risk Assessment Model are better than the ones of the Bayesian Attack Model in which few `compromised` hosts have compromise probabilities very close to `not-compromised` hosts. On the contrary, the Hybrid Risk Assessment Model perfectly distinguishes the `compromised` from the `not-compromised` hosts, even if there is a false negative or a sensor with no information.

As a result we retained the Hybrid Risk Assessment Model to build a response computation architecture, inspired by the one of chapter 4. Most of the principles are identical in the response computation process based on a dynamic risk assessment model in comparison with the remediation computation process based on logical attack paths. The analysis of the logical attack graph to extract the riskiest attack paths is replaced by the dynamic risk assessment that computes the riskiest future attacks. The ranking of response candidates can be identical to the ranking of remediation candidates. The only component that is different is the response computation component that can leverage the advantages of the probabilistic dynamic risk assessment model to compute the best thing to deploy, by assessing their impact on all possible futures.

To conclude, we validated that our dynamic risk assessment models, and especially the Hybrid Risk Assessment Model can apply to the cybersecurity domain, being built from a topological attack graph. Moreover, we showed that the methodology we built for remediation computation may be extended to the computation of dynamic responses relying on the dynamic risk assessment models to correct the riskiest current attacks.

Chapter 8

Conclusion and perspectives

8.1 Contributions

In this PhD thesis, we proposed a risk assessment and response framework. This framework brings three main contributions. The first contribution is the definition of a dynamic risk assessment model. The second contribution is the scalability of this model. The last contribution is the methodology to use risk assessment models to support the computation of responses.

8.1.1 Contribution 1: dynamic risk assessment model

The first contribution of this PhD thesis is the definition of a risk assessment model, correlating the alerts and a priori knowledge on the system, to assess the current risk and predict possible futures. This contribution is detailed in chapter 5, which presents the Bayesian Attack Model, and chapter 6, which presents the Hybrid Risk Assessment Model. The Bayesian Attack Model is a necessary first step to transform a Generic Attack Model into a model enabling dynamic risk assessment. It takes into account dynamic security events to update the compromise probabilities of assets. It manages input models containing cycles, which is common for attack models. The Hybrid Risk Assessment Model is an incremental improvement of to the Bayesian Attack Model which enables scalability and better accuracy in the results, by separating the compromise status of assets between successful compromises and likely future attempts by the attacker. This model takes advantage of the multiple representations of attack models (logical, topological, and probabilistic): it forms a topological representation, but is based on a probabilistic model, to represent both the logical conditions necessary to carry out the attacks, and the probabilities of occurrence of the attacks according to the context. The Hybrid Risk Assessment Model relies on a set of parameters depending on the input Generic Attack Model (sensors false positive or false negative rates, probabilities of attack sources) or are inherent to the Hybrid Risk Assessment Model, either for the Dynamic Risk Correlation Model (parameters for the pruning of paths with too much `no-info` or `no-alert` sensors), or for the Future Risk Assessment Model (number of successive attack steps for possible futures, probability of a new transition). We analyse the sensitivity of the model toward these parameters. It shows that, in the Hybrid Risk Assessment Model, the ranking of states is generally not impacted by the variation of the parameters, and the absolute value of compromise probabilities is slightly impacted by an up to medium uncertainty on most parameters, except the

`false-negative` parameter that is quite sensitive and might add false positives, if its value is not little enough.

The dynamic risk assessment model we present in this thesis brings at least three significative improvements compared to the state of the art. The first one is the ability to build the model from an attack model containing cycles, for example, the vast majority of attack graphs. While all Bayesian network-based models of the state of the art do not manage cycles, the model we present can handle input models with cycles. On the other hand, the Hybrid Risk Assessment Model separates in appropriate models the analysis of the past (within Dynamic Risk Correlation Models) and of the future (within Future Risk Assessment Models). Thus, contrary to the other dynamic risk assessment models of the state of the art, the probabilities of compromise do not confuse the probabilities of being already compromised with those of being compromised in a near future. Finally, contrary to the other models based on Bayesian attack graphs, our model can distinguish several distinct simultaneous attacks in the alerts raised in a system, by analysing the appropriate Dynamic Risk Correlation Models.

The methodology to build a dynamic risk assessment model merging different information sources has been validated for cybersecurity, using topological attack graphs. Indeed, we showed that we can easily transform a topological attack graph into a Generic Attack Model, the main input of our models. Then, we apply the Bayesian Attack Model and the Hybrid Risk Assessment Model to a real use-case of eleven virtual machines, for a total of a hundred vulnerabilities, on which we run five attack scenarios. The results of the models are satisfying and correspond to the real attacks that were carried out.

The sensitivity analysis of the results of the Hybrid Risk Assessment Model toward its parameters showed that most parameters are not very sensitive. However, the exact value of a few of these parameters may be very difficult to estimate. For example, `probability-new-transition` represents the will of an attacker to make a new attack transition. There is an important uncertainty on the value of this parameter. On the other hand, the `false-negative` associated with each sensor is very sensitive for low values and may introduce false positives in the results, if its value is not low enough. A small variation on such parameter can have a significant impact on the final compromise probability of assets. However, the order between the compromised assets is generally kept, even with an important variation. As a result, the Hybrid Risk Assessment Model gives accurately to security operators the order in which they have to investigate or respond to occurring attacks. But the parameters need to be calibrated accurately, according to the security requirements of the system, in order to give accurate absolute values of the compromise probabilities of the assets. We detail this perspective of our work in subsection 8.2.1.

The dynamic risk assessment models we present are built from a Generic Attack Model. We show in chapter 7 that this Generic Attack Model can be built from logical and topological attack graphs, enabling the use of Bayesian Attack Model and the Hybrid Risk Assessment Model for cybersecurity. However, we can also generate the Generic Attack Model for other types of attack models enabling to use the dynamic risk assessment models in other domains. We detail this perspective of our work in subsection 8.2.4.

8.1.2 Contribution 2: scalable risk assessment model

The second main contribution of this PhD thesis is the scalability of the risk assessment models. This contribution is detailed in chapter 5, for the Bayesian Attack Model, and go one step further in chapter 6 with the Hybrid Risk Assessment Model. Scalability is an important challenge of static risk assessment model, but it is much more important for dynamic risk assessment models which need to be evaluated frequently. Moreover, in order to use probabilistic models such as the Bayesian networks, the cycles have to be exploded which increases significantly the size of the

models to process. Thus the Bayesian Attack Model and the Hybrid Risk Assessment Model have pruning policies, in order to prevent the explosion of nodes due to the cycle breaking process. For the Bayesian Attack Model, we limit the exploration depth to a fix number of steps, from every possible attack source. For the Hybrid Risk Assessment Model, the model has been defined in a way that it only explores the paths that are likely to happen and thus it is more performing, while keeping longer likely paths.

Both Bayesian Attack Model and Hybrid Risk Assessment Model have been successfully implemented on simulated topologies of size far beyond the state of the art. For example, the models of the state of the art were evaluated on topology with from 3 to 8 hosts. Thanks to our polytree model, we successfully run our dynamic risk assessment models efficiently on simulated topologies with up to 70 states.

The scalability of the Bayesian Attack Model and of the Hybrid Risk Assessment Model was validated on simulated topologies of arbitrary size (from 0 to 70 hosts) but also arbitrary connectivity. The hosts are gathered in a varying number of fully-connected subnetworks, with each subnetwork able to attack any host of an other subnetwork, in cascade.

However, even if the scalability of the risk assessment models presented in this thesis are more performing than the state of the art, it works only for medium information systems (up to 70 machines). We detail this perspective of our work in subsection 8.2.2.

8.1.3 Contribution 3: risk assessment model to support computation of responses

The last main contribution of this PhD is the ability to use risk assessment models to support the computation of responses. This contribution is detailed in chapter 4 and in section 7.4. The risk assessment models give the attacks that are the most likely to happen and the possible futures for those that are happening. We show in this PhD thesis that we can use both logical attack paths (*i.e.*, static attack models) and the Hybrid Risk Assessment Model (dynamic risk assessment models) to compute possible responses. Then, the responses can be evaluated to know whether or not they can prevent the likely attacks. Finally, the costs and impacts of the responses are taken into account to choose the best response candidates that will be proposed to the operators.

For the computation of remediation, what distinguishes our approach from most of the similar approaches of the state of the art is that we do not compute remediations to an attack *graph* but to attack *paths*, meaning that our algorithms are working with smaller inputs, correcting the most likely attacks, and not all possible attacks. What is also original in our approach is that our remediation computation is generic. Dealing with new attacks only implies to define remediation for potential new kinds of preconditions by filling the generic remediation database we present. Moreover, this remediation computation methodology can be adapted to compute responses preventing current attacks.

The remediation methodology was validated on realistic use-cases. The first use-case is a simple experiment scenario of five hosts in which we compute the remediation candidates for the most serious attack path. The second use-case is in the context of the FP7 European Research Project PoSecCo with a network topology of twenty machines for more than a thousand vulnerabilities and real end-users. The end-users concluded that remediating attack scenarios using this methodology was much more efficient than doing this analysis manually.

Our work currently computes the responses only for the most likely potential attacks, but we showed that thanks to dynamic risk assessment models, they could also be computed to prevent occurring attacks. However, this has not been extensively studied in our work and stays a perspective of our work, as detailed in subsection 8.2.3.

8.2 Perspectives and future work

8.2.1 Calibration of the models parameters

One of the most difficult challenges of probabilistic models is the estimation of the models parameters. As we showed in this work, the uncertainty on the parameters of dynamic risk assessment models can have a significant impact on the final values of the compromise probabilities. Thus, security operators using a dynamic risk assessment model have to be careful when setting these parameters. In this PhD thesis we study precisely the sensitivity of the results of the models toward their parameters. We thus have identified the parameters for which we particularly need accuracy, to have accurate output compromise probabilities. However, future work is still needed on a methodology to estimate accurately such parameters.

A first solution to this challenge is to find an already existing and accurate source of data for as much parameter as possible. For example, we proposed in this thesis to use the CVSS as main input of vulnerability exploitation probability. But this score is not sufficient enough to be really discriminating for vulnerabilities, as most vulnerabilities have the same CVSS score (a very high score). That is why we have introduced a `probability-new-transition` parameter to prevent a too fast increase of the compromise probabilities. A more discriminating source of data for the conditions of transitions would make this parameter irrelevant and allow to get more accurate results. For other parameters, new sources of data have to be found.

Another interesting solution would be to take advantage of the ability of Bayesian networks to learn the parameters from data. Indeed, Bayesian networks are especially interesting for structure or parameters learning from data. In our context, it would be very interesting to learn and update the parameters of the Bayesian Attack Model or the Hybrid Risk Assessment Model, according to the confirmation or negation by the security operators of the results of the models, after manual checks. This will allow to progressively update the parameters and, for example, give priority to the most accurate sensors.

8.2.2 Scalability of the risk assessment models for larger information systems

In the state of the art, dynamic risk assessment models based on Bayesian networks were evaluated on small topologies of up to 10 nodes. Due to the cycle breaking process, the Bayesian Attack Model and Hybrid Risk Assessment Model are limited to up to medium Generic Attack Models (up to 70 nodes). In order to be able to use those dynamic risk assessment models for bigger Generic Attack Models, we need new heuristics to go one step further for scalability.

A solution that could be used for this challenge is to cluster the states that behave in the same manner in the Generic Attack Model, *i.e.*, to determine the cluster of states of the Generic Attack Model that are targeted by the same transitions. States with identical input and output transitions are very common in Generic Attack Models. For example, in a cluster of servers, many hosts are using the same templates and have identical network accesses. Thus they can be targeted by the same attack sources, exploiting the same vulnerabilities. For each of those clusters of states we can create a new node in a higher lever hierarchical Generic Attack Model that thus contains a much smaller number of states. So, future work has to study how we can use this hierarchy in Generic Attack Models to model bigger information systems, while keeping details of the actual states that have been compromised and the exact attack transition that has been taken by the attacker.

A complementary solution to this challenge is the distribution of the computations of Bayesian networks. Indeed, both Bayesian Attack Model in Hybrid Risk Assessment Model uses many

Bayesian networks in which computations are done separately and then are reconciliated. Thus, the Bayesian networks build and inference computations, which are the most expensive in the dynamic risk assessment models, can be easily distributed on several nodes, allowing to decrease significantly the time needed to evaluate the model. Moreover, several approximate Bayesian network inference algorithms rely on many independent sampling. As a result, with the use of such algorithms, an inference in a big network could also be done much faster on several nodes.

8.2.3 Computation of responses using dynamic risk assessment models

In this PhD thesis, we only investigated in detail the remediation computation using static risk assessment models such as logical attack graphs. However, the dynamic risk assessment models we built have been designed to support the computation of responses for occurring attacks too. Future work will investigate how the Bayesian Attack Model and the Hybrid Risk Assessment Model can be used to compute the best responses to prevent an occurring attack.

One advantage of Bayesian networks is that they allow a quantitative probabilistic analysis. A possible solution to assess the impact of remediations is to add new "response" nodes in the Bayesian models, attached to the transition they prevent. Then we can compute the impact of this response on the entire system with a Bayesian inference on this new Bayesian network. This will allow to quantify the security improvement brought by the response and the residual risk.

8.2.4 Application of the Bayesian Attack Model and Hybrid Risk Assessment Model to other domains

In this thesis, we build our dynamic risk assessment models from a Generic Attack Model. In the domain of cybersecurity, we build the Generic Attack Model from an attack graph. But this Generic Attack Model has been designed to be generic and to apply also to other domains. An interesting future work would be to investigate in which other domains there are graph-based attack models which can be specified as a Generic Attack Model, to be able to build the dynamic risk assessment models.

For example, we can build a Generic Attack Model from a list of possible attack scenarios, rather than technical attack graphs, in order to apply the dynamic risk assessment models for a higher-level view of cybersecurity attacks, or cyberphysical security. In such attack scenarios, states represent the privilege of the attacker, either in physical or in a logical world. For example, the position of an attacker in a server room. Transitions represent attack events or change of state of an attacker. For example, the plug of a malicious USB key on a server. Sensors can also be transposed to the physical world, for example, the detection of a person in a room by a surveillance system.

Another field, closer to attack graphs on usual information systems is the virtualised environment. Thanks to the attack graph engine we developed during this PhD, we can compute the attack graphs of such volatile infrastructure. Indeed, this engine is efficient and distributed. So, it can manage the frequent changes that can happen in Cloud environments. Moreover, its attack rules can be refined to include the specificity of such environments. Then, we can apply the dynamic risk assessment models to the generated attack graph.

List of Figures

List of Tables

Glossary of Acronyms

BAM	Bayesian Attack Model
BAT	Bayesian Attack Tree
CPT	Conditional Probability Tables
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
DRCM	Dynamic Risk Correlation Model
EBIOS	Expression of Needs and Identification of Security Objectives
FRAM	Future Risk Assessment Model
GAM	Generic Attack Model
HRAM	Hybrid Risk Assessment Model
HIPS	Host Intrusion Prevention System
IS	Information System
IDS	Intrusion Detection System
IPS	Intrusion Prevention System
IRS	Intrusion Response System
ISMS	Information Security Management System
NIPS	Network Intrusion Prevention System
NIST	National Institute of Standards and Technology
PVG	Patch and Vulnerability Group
TAG	Topological Attack Graph

Bibliography

[AHPS14] Florian Arnold, Holger Hermanns, Reza Pulungan, and Mariëlle Stoelinga. Time-dependent analysis of attacks. In *Principles of Security and Trust: Third International Conference, POST 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014, Proceedings*, pages 285–305. Springer, 2014.

[AJN12] Massimiliano Albanese, Sushil Jajodia, and Steven Noel. Time-efficient and cost-effective network hardening using attack graphs. In *42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 1–12. IEEE, 2012.

[AP08] Qutaibah Althebyan and Brajendra Panda. A knowledge-based bayesian model for analyzing a system after an insider attack. In *IFIP International Information Security Conference*, pages 557–571. Springer, 2008.

[Art02] Michael Lyle Artz. *NetSPA: A Network Security Planning Architecture*. PhD thesis, Massachusetts Institute of Technology, 2002.

[AWK02] Paul Ammann, Duminda Wijesekera, and Saket Kaushik. Scalable, graph-based network vulnerability analysis. In *9th ACM Conference on Computer and Communications Security*, 2002.

[Axe00] Stefan Axelsson. Intrusion detection systems: A survey and taxonomy. Technical report, Chalmers University of Technology, Goteborg, Sweden, 2000.

[AZ12] Saman Aliari Zonouz. *Game-theoretic intrusion response and recovery*. PhD thesis, University of Illinois at Urbana-Champaign, 2012.

[BBK14] Monowar H Bhuyan, Dhruba Kumar Bhattacharyya, and Jugal Kumar Kalita. Network anomaly detection: methods, systems and tools. *Communications Surveys & Tutorials, IEEE*, 16(1):303–336, 2014.

[BEJS10] Henk Birkholz, Stefan Edelkamp, Florian Junge, and Karsten Sohr. Efficient automated generation of attack trees from vulnerability databases. In *Working Notes for the 2010 AAAI Workshop on Intelligent Security (SecArt)*, pages 47–55, 2010.

[BP03] P J Brooke and R F Paige. Fault trees for security system design and analysis. *Computers & Security*, 2003.

[CAB+06] Frédéric Cuppens, Fabien Autrel, Yacine Bouzida, Joaquin Garcia, Sylvain Gombault, and Thierry Sans. Anti-correlation as a criterion to select appropriate countermeasures in an intrusion detection framework. In *Annales des télécommunications*, volume 61, pages 197–217. Springer, 2006.

[Car00] C A Carver. Intrusion Response Systems: A Survey. *Department of Computer Science, Texas A & M University, College Station, TX*, 2000.

[CCZ08] H Cavusoglu, H Cavusoglu, and J Zhang. Security patch management: Share the burden or share the damage? *Management Science*, 54(4):657–670, April 2008.

[CM05] Marco Cremonini and Patrizia Martini. Evaluating information security investments from attackers perspective: the return-on-attack (ROA). In *Fourth Workshop on the Economics of Information Security*, pages 800–830. Citeseer, 2005.

[Col13] Robert Cole. *Multi-step Attack Detection via Bayesian Modeling Under Model Parameter Uncertainty*. PhD thesis, The Pennsylvania State University, 2013.

[CTT05] C.W. Chang, D.R. Tsai, and J.M. Tsai. A cross-site patch management model and architecture design for large scale heterogeneous environment. In *Security Technology, 2005. CCST'05. 39th Annual 2005 International Carnahan Conference on*, pages 41–46. IEEE, 2005.

[CY07] Seyit Ahmet Camtepe and Bulent Yener. Modeling and detection of complex attacks. In *2007 3rd International Conference on Security and Privacy in Communications Networks and the Workshops - SecureComm 2007*, pages 234–243. IEEE, 2007.

[DLK04] Ram Dantu, Kall Loper, and Prakash Kolan. Risk management using behavior based attack graphs. In *Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004. International Conference on*, volume 1, pages 445–449. IEEE, 2004.

[DMCR06] G C Dalton, R F Mills, J M Colombi, and R A Raines. Analyzing Attack Trees using Generalized Stochastic Petri Nets. *Information Assurance Workshop, 2006 IEEE*, 2006.

[DMM+13] Lukas Demetz, Ronald Maier, Markus Manhart, Henrik Plate, and Matthias Fitz. D1.7 - final project evaluation. Technical report, PoSecCo European Project from the 7th Framework (project no. 257129), 2013.

[DPRW07] Rinku Dewri, Nayot Poolsappasit, Indrajit Ray, and Darrell Whitley. Optimal security hardening using multi-objective optimization on attack tree models of networks. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 204–213. ACM, 2007.

[Dru99] Marek J Druzdzel. Smile: Structural modeling, inference, and learning engine and genie: a development environment for graphical decision-theoretic models. In *Aaai/Iaai*, pages 902–903, 1999.

[DW06] Ole Martin Dahl and Stephen D Wolthusen. Modeling and execution of complex attack scenarios using interval timed colored petri nets. *Fourth IEEE International Workshop on Information Assurance*, pages 12 pp.–168, 2006.

[FIR15] FIRST-Forum of Incident Response and Security Teams. Common vulnerability scoring system v3.0: Specification document. Technical report, 2015.

[FW08] M Frigault and Lingyu Wang. Measuring network security using bayesian network-based attack graphs. *Computer Software and Applications, 2008. COMPSAC '08. 32nd Annual IEEE International*, pages 698–703, 2008.

[FWSJ08] Marcel Frigault, Lingyu Wang, Anoop Singhal, and Sushil Jajodia. Measuring network security using dynamic bayesian network. In *the 4th ACM workshop*, pages 23–30, New York, New York, USA, 2008. ACM, ACM Press.

[GDJ+12] Gustavo Gonzalez Granadillo, Hervé Débar, Grégoire Jacob, Chrystel Gaber, and Mohammed Achemlal. Individual countermeasure selection based on the return on response investment index. In *Computer Network Security*, pages 156–170. Springer, 2012.

[GJDC12] Gustavo Gonzalez Granadillo, Greagoire Jacob, Hervé Debar, and Luigi Coppolino. Combination approach to select optimal countermeasures based on the rori index. In *Second International Conference on Innovative Computing Technology*, pages 38–45. IEEE, 2012.

[GTHM14] Erwan Godefroy, Eric Totel, Michel Hurfin, and Frédéric Majorczyk. Automatic generation of correlation rules to detect complex attack scenarios. In *Information Assurance and Security (IAS), 2014 10th International Conference on*, pages 23–28. IEEE, 2014.

[Haz89] M. Hazewinkel, editor. *Encyclopaedia of Mathematics*. Springer Netherlands, 1989.

[HLZ10] Matthew H Henry, Ryan M Layer, and David R Zaret. Coupled petri nets for computer network risk analysis. *International Journal of Critical Infrastructure Protection*, 3(2):67–75, 2010.

[Ing09] Terrance R Ingoldsby. Attack tree-based threat risk analysis. *Amenaza Technologies Ltd. Copyright*, 2010, 2009.

[ISO11a] ISO/IEC. ISO 27005: 2011. *Information technology–Security techniques–Information security risk management. ISO*, 2011.

[ISO11b] ISO/IEC 27035:2011. Information technology – Security techniques – Information security incident management. Technical report, International Organization for Standardization, 2011.

[ISO14] ISO/IEC 27000:2014. Information technology – Security techniques – Information security management systems – Overview and vocabulary. Technical report, International Organization for Standardization, 2014.

[Jen87] Kurt Jensen. Coloured petri nets. *Petri nets: central models and their properties*, pages 248–299, 1987.

[JKM+15] Ravi Jhawar, Barbara Kordy, Sjouke Mauw, Saša Radomirović, and Rolando Trujillo-Rasua. Attack trees with sequential conjunction. In *ICT Systems Security and Privacy Protection*, pages 339–353. Springer, 2015.

[JN10] S Jajodia and S Noel. Advanced cyber attack modeling analysis and visualization. Technical report, DTIC Document, 2010.

[JNK+11] S Jajodia, S Noel, P Kalapa, M Albanese, and J Williams. Cauldron mission-centric cyber situational awareness with defense in depth. In *Military Communications Conference*, pages 1339–1344, 2011.

[JNO05] S Jajodia, S Noel, and B O'Berry. Topological analysis of network attack vulnerability. *Managing Cyber Threats*, pages 247–266, 2005.

[KCBC+09] Wael Kanoun, Nora Cuppens-Boulahia, Frédéric Cuppens, Samuel Dubus, and Antony Martin. Success likelihood of ongoing attacks for intrusion detection and response systems. In *Computational Science and Engineering, 2009. CSE'09. International Conference on*, volume 3, pages 83–91. IEEE, 2009.

[KCBCD10] Nizar Kheir, Nora Cuppens-Boulahia, Frédéric Cuppens, and Hervé Debar. A Service Dependency Model for Cost-Sensitive Intrusion Response. In Dimitris Gritzalis, Bart Preneel, and Marianthi Theoharidou, editors, *Computer Security – ESORICS 2010*, pages 626–642. Springer Berlin Heidelberg, 2010.

[KDCB+09a] N Kheir, H Debar, N Cuppens-Boulahia, F Cuppens, and J Viinikka. Cost evaluation for intrusion response using dependency graphs. In *Network and Service Security*, pages 1–6. IEEE, 2009.

[KDCB+09b] Nizar Kheir, Hervé Debar, Nora Cuppens-Boulahia, Frédéric Cuppens, and Jouni Viinikka. Cost evaluation for intrusion response using dependency graphs. In *Network and Service Security, 2009. N2S'09. International Conference on*, pages 1–6. IEEE, 2009.

[KH10] Phongphun Kijsanayothin and Rattikorn Hewett. Analytical approach to attack graph analysis for network security. In *Availability, Reliability, and Security, 2010. ARES'10 International Conference on*, pages 25–32. IEEE, 2010.

[Khe10] N Kheir. *Response Policies and Counter-measure: Management of Service Dependencies and Intrusion and Reaction Impacts*. PhD thesis, Ecole nationale supérieure des télécommunications de Bretagne, 2010.

[Kis13] Richard Kissel. Glossary of key information security terms. *NIST Interagency Reports NIST IR*, 7298:3, 2013.

[KLI08] Do Hoon Kim, Taek Lee, and H P In. Effective security safeguard selection process for return on security investment. In *Asia-Pacific Services Computing Conference, 2008. APSCC '08. IEEE*, pages 668–673, December 2008.

[KPCS13] Barbara Kordy, Ludovic Piètre-Cambacédès, and Patrick Schweitzer. Dag-based attack and defense modeling: Don't miss the forest for the attack trees. *CoRR*, March 2013.

[KVK09] Mikko Kiviharju, Teijo Venäläinen, and Suna Kinnunen. Towards modelling information security with key-challenge petri nets. *14th Nordic Conference on Secure IT Systems*, pages 190–206, 2009.

[LI05] R P Lippmann and K W Ingols. An annotated review of past papers on attack graphs. Technical report, DTIC Document, 2005.

[LM05] Y Liu and H Man. Network vulnerability assessment using Bayesian networks. *Defense and Security*, 2005.

[LS88] Steffen L Lauritzen and David J Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society*, 1988.

[Man16] Mandiant Consulting. M-Trends 2016 Special Report. Technical report, FireEye, Inc., 2016.

[MBFB06] M A McQueen, W F Boyer, M A Flynn, and G A Beitel. Quantitative Cyber Risk Reduction Estimation Methodology for a Small SCADA Control System. In *System Sciences, 2006. HICSS '06. Proceedings of the 39th Annual Hawaii International Conference on*, page 226, 2006.

[MBZ+06] Vaibhav Mehta, Constantinos Bartzis, Haifeng Zhu, Edmund Clarke, and Jeannette Wing. Ranking attack graphs. In *Recent advances in intrusion detection*, pages 127–144. Springer, 2006.

[McD00] J P McDermott. Attack net penetration testing. In *the 2000 workshop*, pages 15–21, New York, New York, USA, 2000. ACM Press.

[Mea98] C Meadows. A representation of protocol attacks for risk assessment. In *Proceedings of the DIMACS Workshop on Network ...*, 1998.

[MIT14] MITRE corporation. The MITRE Systems Engineering Guide. Technical report, 2014.

[MMDD09] Benjamin Morin, Ludovic Mé, Hervé Debar, and Mireille Ducasséc. M4D4: a Logical Framework to Support Alert Correlation in Intrusion Detection. 2009.

[MO05] Sjouke Mauw and Martijn Oostdijk. Foundations of attack trees. In *ICISC'05: Proceedings of the 8th international conference on Information Security and Cryptology*. Springer-Verlag, December 2005.

[MRT15] Erik Miehling, Mohammad Rasouli, and Demosthenis Teneketzis. Optimal defense policies for partially observable spreading processes on bayesian attack graphs. In *Proceedings of the Second ACM Workshop on Moving Target Defense*, pages 67–76. ACM, 2015.

[MSL10] C. Mu, B. Shuai, and H. Liu. Analysis of response factors in intrusion response decision-making. In *Computational Science and Optimization (CSO), 2010 Third International Joint Conference on*, volume 2, pages 395–399. IEEE, 2010.

[Nat05] National Institute of Standards and Technology. Creating a Patch and Vulnerability Management Program. Technical report, 2005.

[Nat12] National Institute of Standards and Technology. SP 800-30 Rev. 1: Guide for Conducting Risk Assessments. Technical report, 2012.

[NGGT10] Y. Nunez, F. Gustavson, F. Grossman, and C. Tappert. Designing a distributed patch management security system. In *Information Society (i-Society), 2010 International Conference on*, pages 162–167. IEEE, 2010.

[NJ08] S Noel and S Jajodia. Optimal ids sensor placement and alert prioritization using attack graphs. *Journal of Network and Systems Management*, 2008.

[NJ09] S Noel and S Jajodia. Proactive intrusion prevention and response via attack graphs. Technical report, Addison-Wesley Professional, 2009.

[NJOJ03] Steven Noel, Sushil Jajodia, Brian O'Berry, and Michael Jacobs. Efficient minimum-cost network hardening via exploit dependency graphs. In *Computer security applications conference, 2003. proceedings. 19th annual*, pages 86–95. IEEE, 2003.

[OBM06] Xinming Ou, Wayne F Boyer, and Miles A McQueen. A scalable approach to attack graph generation. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 336–345. ACM, 2006.

[OGA05a] X Ou, S Govindavajhala, and A W Appel. Mulval: A logic-based network security analyzer. In *Proceedings of the 14th conference on USENIX Security Symposium-Volume 14*, pages 8–8. USENIX Association, 2005.

[OGA05b] X Ou, S Govindavajhala, and A W Appel. Mulval: A logic-based network security analyzer. In *USENIX Security Symposium*, 2005.

[oNSS10] Committee on National Security Systems. CNSS N 4009: National Information Assurance Glossary. Technical report, 2010.

[Pau14] Stéphane Paul. Towards automating the construction & maintenance of attack trees: a feasibility study. *Proceedings GraMSec 2014*, 2014.

[PCB10] Ludovic Pièvre-Cambacédès and Marc Bouissou. Beyond attack trees: dynamic security modeling with Boolean logic Driven Markov Processes (BDMP). In *European Dependable Computing Conference*, pages 199–208, 2010.

[PDR12] Nayot Poolsappasit, Rinku Dewri, and Indrajit Ray. Dynamic security risk management using bayesian attack graphs. *Dependable and Secure Computing*, 2012.

[Pea86] Judea Pearl. Fusion, propagation, and structuring in belief networks. *AI*, 1986.

[Pea87] Judea Pearl. Evidential reasoning using stochastic simulation of causal models. *AI*, 1987.

[Pea88] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

[PML09] Srdjan Pudar, G Manimaran, and Chen-Ching Liu. PENET: A practical method and tool for integrated modeling of security attacks and countermeasures. *Computers & Security*, 28(8):754–771, November 2009.

[PR98] Judea Pearl and Stuart Russell. *Bayesian networks*. Computer Science Department, University of California, 1998.

[PS98] Cynthia Phillips and Laura Painton Swiler. A graph-based system for network-vulnerability analysis. In *the 1998 workshop*, pages 71–79, New York, New York, USA, 1998. ACM Press.

[QL04] Xinzhou Qin and Wenke Lee. Attack plan recognition and prediction using causal networks. In *Computer Security Applications Conference, 2004. 20th Annual*, pages 370–379, 2004.

[R+99] Martin Roesch et al. Snort: Lightweight intrusion detection for networks. In *LISA*, volume 99, pages 229–238, 1999.

[RP05] Indrajit Ray and Nayot Poolsapassit. Using attack trees to identify malicious attacks from authorized insiders. In *ESORICS'05: Proceedings of the 10th European conference on Research in Computer Security*. Springer-Verlag, September 2005.

[SAS06] Wes Sonnenreich, Jason Albanese, and Bruce Stout. Return on security investment (ROSI)-A practical quantitative model. *Journal of Research and Practice in Information Technology*, 38(1):45–56, 2006.

[SBW07a] N Stakhanova, S Basu, and J Wong. A cost-sensitive model for preemptive intrusion response systems. In *Proceedings of the 21st International Conference on Advanced Networking and Applications*, pages 428–435. IEEE Computer Society, 2007.

[SBW07b] N Stakhanova, S Basu, and J Wong. A taxonomy of intrusion response systems. *International Journal of Information and Computer Security*, 1(1):169–184, 2007.

[SBW07c] N Stakhanova, S Basu, and J Wong. A taxonomy of intrusion response systems. *International Journal of Information and Computer Security*, 1(1):169–184, 2007.

[Sch99] Bruce Schneier. Attack Trees. *Dr. Dobb's Journal*, December 1999.

[Sec04] Secrétariat Général de la Défense Nationale. EBIOS - Expression of Needs and Identification of Security Objectives. Technical report, 2004.

[SHJ+02] Oleg Sheyner, Joshua Haines, Somesh Jha, Richard Lippmann, and Jeannette M Wing. Automated generation and analysis of attack graphs. In *Security and privacy, 2002. Proceedings. 2002 IEEE Symposium on*, pages 273–284. IEEE, 2002.

[SMC09] Robert Schuppenies, Christoph Meinel, and Feng Cheng. Automatic extraction of vulnerability information for attack graphs. *Hasso-Plattner-Institute for IT Systems Engineering, University of Potsdam*, 2009.

[SO08] R E Sawilla and X Ou. *Identifying critical attack assets in dependency attack graphs*. Springer, 2008.

[SPEC01] Laura P Swiler, Cynthia Phillips, David Ellis, and Stefan Chakerian. Computer-attack graph generation tool. In *DARPA Information Survivability Conference & Exposition II, 2001. DISCEX'01. Proceedings*, volume 2, pages 307–321. IEEE, 2001.

[SSBW09] C Strasburg, N Stakhanova, S Basu, and J S Wong. Intrusion response cost assessment methodology. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, pages 388–391. ACM, 2009.

[SSEJJ12] A Shameli-Sendi, N Ezzati-Jivan, and M Jabbarifar. Intrusion response systems: survey and taxonomy. *SIGMOD*, 2012.

[SSL15] Xiaoyan Sun, Anoop Singhal, and Peng Liu. Who touched my mission: Towards probabilistic mission impact assessment. In *Proceedings of the 2015 Workshop on Automated Decision Making for Active Cyber Defense*, pages 21–26. ACM, 2015.

[SSS14] Vivek Shandilya, Chris B Simmons, and Sajjan Shiva. Use of attack graphs in security systems. *Journal of Computer Networks and Communications*, 2014, 2014.

[TFGB07] C J Tucker, S M Furnell, Ghita, BV, and P J Brooke. A new taxonomy for comparing intrusion detection systems. *Internet Research*, 17(1):88–98, 2007.

[THC02] S Tanachaiwiwat, K Hwang, and Y Chen. Adaptive intrusion response to minimize risk over multiple network attacks. *ACM Trans on Information and System Security*, 2002.

[TK02] T Toth and C Kruegel. Evaluating the impact of automated intrusion response mechanisms. In *Computer Security Applications Conference*, pages 301–310. IEEE, 2002.

[WFM+07] Yu-Sung Wu, Bingrui Foo, Yu-Chun Mao, Saurabh Bagchi, and Eugene H Spafford. Automated adaptive intrusion containment in systems of interacting services. *Computer networks*, 51(5):1334–1360, 2007.

[Whi06] Dominic Stjohn Dolin White. *Limiting Vulnerability Exposure through effective Patch Management: threat mitigation through vulnerability remediation*. PhD thesis, Rhodes University, 2006.

[WNJ06] Lingyu Wang, Steven Noel, and Sushil Jajodia. Minimum-cost network hardening using attack graphs. *Computer Communications*, 29(18):3812–3824, 2006.

[Wu09] Yu-Sung Wu. *Achieving high survivability in distributed systems through automated response*. PhD thesis, Purdue University, 2009.

[WXX+08] Z. Wu, D. Xiao, H. Xu, X. Peng, and X. Zhuang. Automated intrusion response decision based on the analytic hierarchy process. In *Knowledge Acquisition and Modeling Workshop, 2008. KAM Workshop 2008. IEEE International Symposium on*, pages 574–577. IEEE, 2008.

[WZK13] Shuzhen Wang, Zonghua Zhang, and Youki Kadobayashi. Exploring attack graph for cost-benefit security hardening: A probabilistic approach. *Computers & security*, 32:158–169, 2013.

[XLO+10] Peng Xie, Jason H Li, Xinming Ou, Peng Liu, and R Levy. Using Bayesian networks for cyber security analysis. In *Dependable Systems and Networks (DSN), 2010 IEEE/IFIP International Conference on*, pages 211–220, 2010.

[Yag06] Yager. OWA trees and their role in security modeling using attack trees. *Information Sciences*, 176(20):27–27, October 2006.

[ZKSY09] S.A. Zonouz, H. Khurana, W.H. Sanders, and T.M. Yardley. Rre: A game-theoretic intrusion response and recovery engine. In *Dependable Systems & Networks, 2009. DSN'09. IEEE/IFIP International Conference on*, pages 439–448. IEEE, 2009.

[ZLYS10] J. Zhu, Y. Liu, X. Yang, and X. SUN. Dynamic game based intrusion response model. *Journal of Computational Information Systems*, pages 2199–2211, 2010.

Author's publications

Publications in peer-reviewed conferences

[AGBC14] F.-X. Aguessy, L. Gaspard, O. Bettan and V. Conan. Remediating Logical Attack Paths Using Information System Simulated Topologies. In *Computer & Electronics Security Applications Rendez-vous 2014*, pages 187–203, 2014.

[AODLLF15] F.-X. Aguessy, O. Bettan, R. Dobigny, C. Laudy, G. Lortal and D. Faure. Adjustable Fusion to support Cyber Security Operators. In *Human Aspects of Information Security, Privacy, and Trust: Third International Conference, HAS 2015, Held as Part of HCI International 2015, Los Angeles, CA, USA, August 2-7, 2015. Proceedings*, pages 143–153, 2015.

[MDMSBACLTM15] B. Mathieu, G. Doyen, W. Mallouli, T. Silverston, O. Bettan, F.-X. Aguessy, T. Cholez, A. Lahmadi, P. Truong and E .Montes de Oca. Monitoring and Securing New Functions Deployed in a Virtualized Networking Environment. In *the First International Workshop on Security Testing And Monitoring (STAM), in conjunction with 10th International Conference on Availability, Reliability and Security (ARES), 2015*, pages 741–748, 2015.

[RABBC16] F. Reynaud, F.-X. Aguessy, O. Bettan, M. Bouet, V. Conan. Attacks against Network Functions Virtualization and Software-Defined Networking: State-of-the-art. In *Workshop on Security in Virtualized Networks (Sec-Virtnet 2016), workshop of 2nd IEEE Conference on Network Softwarization (NetSoft 2016)*, 2016.

[ABBCD16] F.-X. Aguessy, O. Bettan, G. Blanc, V. Conan, H. Debar. Hybrid Risk Assessment Model based on Bayesian Networks. In *11th International Workshop on Security, IWSEC 2016, Tokyo, Japan, September 12-14, 2016, Proceedings*, 2016.

Contributions to collaborative projects

[Doc1511] P. Truong, B. Mathieu, F.-X. Aguessy, O. Bettan, E. Montes de Oca, A. Ortiz, W. Mallouli, T. Nguyen, A. Ploix, M. EL Aoun, G. Doyen, A. Lahmadi, T. Cholez. D1.1 - Virtualization Techniques: Analysis and Selection. DOCTOR project, <ANR-14- CE28-000>, started in 01/12/2014 and supported by the French Systematic cluster, 2015.

[Doc1512] P. Truong, B. Mathieu, F.-X. Aguessy, O. Bettan, E. Montes de Oca, A. Ortiz, W. Mallouli, H .Long Mai, A. Ploix, G. Doyen, T. Cholez, X. Marchal and C. Enclos. D1.2 - Architecture of the DOCTOR Virtualized Node. DOCTOR project, <ANR-14- CE28-000>, started in 01/12/2014 and supported by the French Systematic cluster, 2015.

[Doc1621] P. Truong, B. Mathieu, F.-X. Aguessy, F. Reynaud, T. Combe, W. Mallouli, G. Doyen, T. Nguyen, R. Cogranne, T. Cholez and X. Marchal. D2.1 - Security analysis of the virtualized NDN architecture. DOCTOR project, <ANR-14- CE28-000>, started in 01/12/2014 and supported by the French Systematic cluster, (To be published).

[Vir1531] VIRTUALIS Consortium. D3.1 - Spécifications techniques du module d'analyses de risques. VIRTUALIS project, FUI 15, 2015.

[Vir1632] VIRTUALIS Consortium. D3.2 - Briques logicielles d'analyse de risques adaptées à la sécurité physique/logique. VIRTUALIS project, FUI 15, (To be published).

[FIC151711] The FI-Core Consortium. D.17.1.1: Contribution to FIWARE Reference Architecture (Security). Future Internet - Core, European Project from the 7th Framework Program, Project reference: 632893, 2015.

[FIC151721] The FI-Core Consortium. D.17.2.1: FI-WARE GE Open Specifications. Future Internet - Core, European Project from the 7th Framework Program, Project reference: 632893, 2015.

[FIC151731] The FI-Core Consortium. D.17.3.1: SW Release (version of components delivered for integration in testbed). Future Internet - Core, European Project from the 7th Framework Program, Project reference: 632893, 2015.

[FIC151741] The FI-Core Consortium. D.17.4.1: Installation and Administration Guides. Future Internet - Core, European Project from the 7th Framework Program, as part of the Future Internet Public-Private Partnership Programme (FI-PPP), Project reference: 632893, 2015.

[FIC151751] The FI-Core Consortium. D.17.5.1: User and Programmers Guide. Future Internet - Core, European Project from the 7th Framework Program, as part of the Future Internet Public-Private Partnership Programme (FI-PPP), Project reference: 632893, 2015.

[FIC151771] The FI-Core Consortium. D.17.7.1: Technical Roadmap. Future Internet - Core, European Project from the 7th Framework Program, as part of the Future Internet Public-Private Partnership Programme (FI-PPP), Project reference: 632893, 2015.

[FIW14813] The FI-WARE Consortium. D.8.1.3: GE Open Specifications. FI-WARE: Future Internet Core Platform, European Project from the 7th Framework Program, as part of the Future Internet Public-Private Partnership Programme (FI-PPP), Project reference: 285248, 2014.

[FIW14823] The FI-WARE Consortium. D.8.2.3: SW Release (version of components delivered for integration in testbed). FI-WARE: Future Internet Core Platform, European Project from the 7th Framework Program, as part of the Future Internet Public-Private Partnership Programme (FI-PPP), Project reference: 285248, 2014.

[FIW14833] The FI-WARE Consortium. D.8.3.3: Installation and Administration Guide. FI-WARE: Future Internet Core Platform, European Project from the 7th Framework Program, as part of the Future Internet Public-Private Partnership Programme (FI-PPP), Project reference: 285248, 2014.

[FIW14843] The FI-WARE Consortium. D.8.4.3: User and Programmers Guide. FI-WARE: Future Internet Core Platform, European Project from the 7th Framework Program, as part of the Future Internet Public-Private Partnership Programme (FI-PPP), Project reference: 285248, 2014.

[FIW14853] The FI-WARE Consortium. D.8.5.3: Unit Testing Plan and Report. FI-WARE: Future Internet Core Platform, European Project from the 7th Framework Program, as part of the Future Internet Public-Private Partnership Programme (FI-PPP), Project reference: 285248, 2014.

[FIW15CAB] P. CAO, F.-X. Aguessy, P. Bisson Poster - FI-WARE Security Monitoring Generic Enabler Demonstrator. Trust in the Digital World (TDW), 2015.

Appendix A

French abstract - Résumé français

A.1 Introduction

Les systèmes d'information sont d'une valeur inestimable, car ils rassemblent l'intégralité des systèmes utilisés pour le stockage et le traitement des données d'une organisation. Ils représentent donc une cible de plus en plus attractive pour les attaquants. Cependant, ils sont aussi de plus en plus complexes, car ils sont composés d'équipements matériels ou logiciels divers, provenant de nombreux constructeurs ou fournisseurs. Ils reposent sur des logiciels qui sont souvent vulnérables, et qui ne sont pas corrigés, soit, car c'est impossible, soit par manque de temps. Ainsi, la sécurité de tels systèmes est très difficile à maitriser, alors qu'elle est souvent nécessaire pour la survie de l'organisation.

Les opérateurs de sécurité qui doivent protéger les systèmes d'information critiques connaissent généralement la plupart des vulnérabilités de leur système, grâce à des scanneurs de vulnérabilités. Ils doivent faire face à des attaques toujours plus nombreuses et complexes. Les attaques qui ont le plus d'impact sont les attaques multi étapes. Par exemple, l'attaquant commence par exploiter une machine accessible depuis internet. Puis, à partir de celle-ci, il compromet la machine d'un administrateur. Il peut alors accéder légitimement au contrôleur du domaine et prendre le contrôle d'un serveur hébergeant des données critiques. Il existe plusieurs formalismes permettant de représenter ce type d'attaque. Par exemple, un graphe d'attaque est un modèle d'analyse de risque qui contient toutes les attaques connues qui peuvent être effectuées par un attaquant dans un système.

En première ligne de défense, les opérateurs déploient des mesures de protection (par exemple, des correctifs logiciels ou du contrôle d'accès, à l'aide de pare-feu) pour se prémunir des attaques. Cependant, il y a toujours des contournements possibles. En deuxième recours, ils déploient donc des sondes de détection qui les alertent dès qu'un attaquant exploite une vulnérabilité connue. Quand ils reçoivent des alertes, ils doivent réévaluer le niveau de risque de leur système pour choisir les réponses appropriées. C'est ce qui s'appelle l'évaluation dynamique de risque.

Cette thèse de doctorat est dans le domaine de la sécurité des systèmes d'information. Elle vise à construire un système d'analyse statique et dynamique de risque prenant en compte le savoir à priori sur un système d'information (topologie réseau, vulnérabilités, *etc.*) et les informations dynamiques reçues dans ce système (alertes d'intrusion, *etc.*). Il s'applique à des systèmes qui sont visés par des attaques complexes, où les administrateurs ont une bonne connaissance de leur

système et dans lesquels sont déployées des sondes de détection.

Quatre problèmes principaux se posent dans les approches actuelles. Le premier est la difficulté de la construction d'un modèle permettant de concilier les informations de sécurité connues à priori et celles qui arrivent dynamiquement dans le système. Le deuxième problème est le passage à l'échelle d'un tel modèle qui peut être très gros pour un système avec de nombreuses machines vulnérables. Le troisième est la difficulté de profiter des avantages des différentes représentations possibles des attaques qui existent (topologiques, logiques ou probabilistes). Le dernier problème concerne l'exploitation qui peut être faite après avoir construit le modèle d'attaque: comment s'appuyer sur ce modèle pour pouvoir calculer les réponses appropriées pour protéger un système d'information ?

Afin de résoudre ces problèmes, nous proposons dans ce manuscrit de thèse trois contributions permettant de construire un système d'analyse statique et dynamique de risque et de calcul de réponses. La première contribution est la construction d'un modèle probabiliste d'évaluation de risque pouvant s'appliquer à n'importe quel type de système, qu'il possède des cycles ou non. Nous étudions précisément la sensibilité des résultats du modèle en fonction de ses paramètres. Ensuite, nous permettons d'améliorer significativement le nombre de noeuds et de vulnérabilités qui peuvent être pris en compte par rapport aux modèles existants de l'état de l'art. Enfin, nous décrivons comment les modèles d'attaque peuvent être utilisés pour servir de support au calcul de réponses permettant de se protéger des attaques.

A.2 État de l'art

L'analyse de l'état de l'art présentée ici s'organise en deux parties. D'une part, nous étudions les modèles existants dans l'état de l'art permettant de représenter les attaques multi étapes. D'autre part, nous analysons comment ces modèles ont été utilisés pour calculer les réponses empêchant les attaques potentielles ou en cours.

A.2.1 Modèles d'attaque

Un grand nombre de modèles d'attaque de l'état de l'art s'appuient sur des modèles basés sur les graphes (arbre, graphes, réseaux de Petri, réseaux bayésiens, processus de Markov, *etc.*).

Les premiers modèles à avoir été proposés dans la littérature pour représenter les attaques multi étapes sont les *arbres d'attaque* [Sch99, Ing09, Pau14]. Ceux-ci reposent sur des arbres ET/OU dans lesquels les noeuds représentent les actions possibles pour un attaquant qui sont raffinées récursivement jusqu'à devenir des actions atomiques. Les arbres d'attaque ont été étendus dans de nombreux travaux, par exemple pour ajouter d'autres opérateurs: *OU-Exclusif*, *ET-Ordonné* [BP03], *quelques*, *la plupart* [Yag06], ou pour ajouter la position d'un attaquant [RP05] ou encore des informations temporelles [CY07, AHPS14]. La structure d'arbre a été la première permettant de modéliser précisément la succession d'éléments d'attaque dans un système. C'est donc la base de la plupart des modèles suivants. Cependant, elle ne permet pas de représenter facilement la progression d'un attaquant dans un système, car elle ne contient pas de notion de position de l'attaquant et décrit une seule attaque principale.

Pour répondre à ce dernier problème, les *graphes d'attaque* ont été proposés pour s'appuyer sur un graphe, plutôt qu'un arbre, pour décrire les attaques [PS98]. Il existe deux catégories principales de graphes d'attaque [LI05, KPCS13]. Dans la première, les *graphes d'attaque logiques*, les noeuds représentent des faits logiques qui ont besoin soit de la conjonction, soit de la disjonction de leurs parents pour être possibles. Les arcs représentent donc une relation de dépendance entre ces faits logiques. Dans la deuxième catégorie, les *graphes d'attaque topologiques*, les noeuds représentent

des actifs d'un système d'information (machines, interfaces réseau, *etc.*). Chaque arc représente une attaque possible à partir d'un actif, qui permet de prendre le contrôle d'un autre actif du système. Les graphes d'attaque sont générés par des moteurs de graphes d'attaque tels que MulVAL [OGA05a, OBM06], TVA [JNK+11, JNO05] ou NetSPA [Art02], à partir d'un inventaire de la connectivité réseau du système et de ses vulnérabilités. Les graphes d'attaque sont très proches des arbres d'attaque, mais permettent de représenter plusieurs attaques multi étapes dans le même modèle. Ils permettent de décrire précisément ce type d'attaque, mais ne sont pas suffisants pour décrire les attaques en cours ou contenir plusieurs attaquants.

D'autres modèles d'attaque sont basés sur les *réseaux de Petri*. Un réseau de Petri est constitué de places (représentant des états), de transitions (représentant des actions pour passer d'un état à un autre) et de jetons qui sont situés sur les places (chaque jeton représentant un objet dans un état). Les réseaux de Petri sont généralement utilisés pour faire des simulations d'accessibilité des places. Un *réseau d'attaque* [McD00] est basé sur un réseau de Petri dans lequel les places représentent les états de compromission des entités du système, les transitions représentent des événements d'attaque et les jetons représentent la progression d'un attaquant qui possède les privilèges associés à la place où il est. Les réseaux d'attaque ont été étendus par divers formalismes pour, par exemple, prendre en compte la difficulté ou la durée des étapes d'attaque [DMCR06, DW06] ou ajouter des prérequis à certaines étapes d'attaque [PML09]. L'avantage principal des réseaux de Petri par rapport aux graphes ou aux arbres est qu'ils permettent de modéliser, dans un seul modèle, la progression concurrente de plusieurs attaquants dans un système, à l'aide de différents jetons. Cependant, un réseau de Petri ne peut pas être utilisé pour représenter l'ensemble des privilèges acquis par un attaquant (un jeton ne peut être qu'à une seule place en même temps). De plus, il n'est pas facile d'adapter ce type de modèles pour prendre en compte l'ajout de détections en cours d'exécution. Ce formalisme est plutôt fait pour savoir si certains états peuvent être atteints, et quelle est leur probabilité de l'être ou la durée pour y arriver.

Enfin, la dernière famille principale de modèles d'attaque de l'état de l'art regroupe les modèles basés sur les *réseaux bayésiens*. Un réseau bayésien est un graphe orienté acyclique dans lequel chaque noeud possède plusieurs états distincts. Il est associé à une table de probabilité décrivant la probabilité d'être dans un état de ce noeud, étant donné chaque état possible de ses parents. Une fois ce graphe spécifié, le formalisme des réseaux bayésiens fournit des algorithmes efficaces permettant de calculer la probabilité de chaque état du réseau, étant donné des preuves, des noeuds qui ont été fixés dans un état particulier. Les *graphes d'attaque bayésiens* [LM05, FW08] sont constitués de noeuds représentant une machine dans un état spécifique (généralement *compromis* ou *non compromis*) et d'arcs représentant une attaque possible. Ils ont été étendus pour ajouter des informations temporelles, par exemple sur l'évolution de la disponibilité de correctifs ou d'exploits [FWSJ08], ou pour ajouter des informations sur l'incertitude des détections [XLO+10]. Les réseaux bayésiens ajoutent aux modèles basés sur des graphes des outils puissants permettant de calculer les probabilités des différents états des noeuds du graphe. De plus, les dépendances entre les noeuds ne sont plus des ET ou des OU, mais des probabilités d'occurrence compte tenu de l'état des parents, ce qui est beaucoup plus expressif. Enfin, l'attaquant peut quitter chacun des états qui ont été précédemment compromis. Cependant, les modèles d'attaque bayésiens de l'état de l'art ne permettent pas de représenter l'évolution concurrente de plusieurs attaques simultanées. Il se pose aussi deux problèmes lorsque l'on veut utiliser ces modèles en pratique. Le premier est un problème de performance, car l'inférence bayésienne peut être très couteuse à calculer pour de gros réseaux. Le deuxième problème concerne la présence de cycles, très fréquente dans les graphes d'attaque, et qui n'est pas possible dans les réseaux bayésiens. La plupart des travaux de l'état de l'art soit n'évoquent pas le problème des cycles, soit fournissent des solutions non satisfaisantes pour les corriger.

Pour conclure, il y a un grand nombre de modèles d'attaque qui ont été proposés dans la littérature. Certains sont limités uniquement à l'analyse statique de risque, d'autres permettent

aussi de représenter dynamiquement les attaques qui sont en train de se produire. Les modèles qui paraissent les plus intéressants sont ceux basés sur les réseaux bayésiens, mais ils possèdent des limitations pour être vraiment utilisables.

Les modèles existants ont différents niveaux de détail et ils permettent de représenter plus ou moins précisément les étapes d'attaque, en utilisant différents opérateurs logiques ou probabilistes. Or, il est possible qu'une organisation ait déjà des modèles d'attaque existants, ou bien un opérateur peut vouloir faire une analyse avec un niveau de granularité précis. Ainsi, il est utile de définir un modèle d'attaque générique *GAM*, qui généralise la plupart des modèles d'attaque de l'état de l'art, et qui permet d'avoir un seul type de point d'entrée, tout en étant compatible avec la plupart des modèles d'attaque de l'état de l'art basés sur des graphes.

La première étape permettant de construire ce modèle générique *GAM* est d'ajouter les noeuds et arcs: les *états* et les *transitions*. Un état représente l'état d'un attaquant dans le système (par exemple, un privilège sur une machine, la possession d'une donnée, la position d'un attaquant). Une transition est un arc qui représente la possibilité pour un attaquant d'aller d'un état source a un état destination. Chaque transition à un type qui décrit pourquoi et comment l'attaquant peut passer d'un état à un autre (par exemple, l'exploitation d'une vulnérabilité, le vol d'identifiants). En plus de ces deux éléments, il est possible d'ajouter au GAM des conditions, des faits qui doivent être vérifiés pour qu'une transition soit possible, et des sondes de détection, qui permettent de s'assurer que d'autres éléments (un état, une transition ou une condition) sont vérifiés. Enfin, les états, et transitions peuvent être associés à des tables de probabilités qui décrivent la relation qui existe entre une transition et ses conditions, ou bien entre un état et ses transitions entrantes. Les tables peuvent représenter des opérateurs logiques simples (OU, ET, OU-Exclusif, *etc.*) ou bien des relations probabilistes plus complexes. Ainsi, nous pouvons construire un modèle d'attaque générique *GAM* qui peut contenir les informations de la plupart des modèles d'attaque graphiques existants et être un unique point d'entrée, applicable à n'importe quel domaine et avec différents niveaux de granularité.

A.2.2 Calcul de réponses s'appuyant sur les modèles d'attaque

Les réponses regroupent les actions ayant pour but de traiter un risque. Elles peuvent être de deux types: des *remédiations*, c'est à dire des actions ayant pour but de traiter un risque latent, de corriger une vulnérabilité existante, ou des *contre-mesures*, pour corriger le risque d'une attaque en train de se produire. Les réponses peuvent être regroupées en trois catégories: les réponses correctives, les réponses actives et les réponses passives. Les réponses *correctives*, généralement des remédiations ont pour but de résoudre un problème en corrigeant la vulnérabilité, par l'application d'un correctif logiciel. Cette technologie est assez mûre, mais possède certaines limitations [CCZ08] qui font que l'application de correctifs nécessite la plupart du temps une intervention humaine (par exemple, pour des tests de non-régressions) et que parfois ils ne peuvent pas être appliqués (par exemple, à cause de la nécessité de redémarrer un service critique ou de l'indisponibilité des correctifs). Les réponses *actives* ont pour but d'empêcher l'exploitation d'une vulnérabilité, mais sans la corriger. Elles sont généralement mises en oeuvre par des systèmes de prévention d'intrusion [SBW07c] (uniquement des fonctions de filtrage) ou des systèmes de réponses aux intrusions [Car00, SBW07b, THC02] qui peuvent appliquer d'autres types de réponses (arrêt de services ou machines, changement de la politique de contrôle d'accès, *etc.*). Les réponses *passives* [TFGB07] sont un dernier recours, pour être alerté si une vulnérabilité est exploitée. Elles sont généralement implémentées par des systèmes de détection d'intrusion situés sur le réseau ou sur les machines.

Plusieurs articles de l'état de l'art s'appuient sur les modèles d'attaque pour sélectionner des réponses aux attaques. Généralement, ceux-ci sont utilisés pour voir l'impact sur le système entier de l'application d'une réponse et mesurer la réduction du risque [MBFB06, NJ09]. Sinon, ils peu-

vent aussi être le support utilisé pour le calcul des réponses [CAB+06, WNJ06, AJN12]. Ainsi, les modèles d'attaque peuvent permettre d'optimiser la configuration et le déploiement de réponses passives (par exemple, un graphe d'attaque peut permettre de trouver les points de déploiement optimaux de sondes de détection) ou de calculer des remédiations ou d'autres types de réponses permettant de protéger les cibles potentielles d'attaque. Au contraire, les modèles topologiques et fonctionnels du système d'information [MMDD09, JN10, TK02, KCBCD10, SSL15] sont généralement utilisés pour évaluer l'impact des réponses à déployer. Au lieu d'écrire toutes les attaques possibles dans un système, ils présentent les services légitimes ou les missions du système et leurs interdépendances. Leur utilisation est complémentaire aux modèles d'attaque.

Pour sélectionner les réponses à déployer, plusieurs critères peuvent être pris en compte, soit reliés à l'attaque (par exemple, le type d'attaque, sa durée), soit reliés à la réponse (par exemple, l'efficacité de la réponse, ses effets négatifs), soit reliés à la cible (par exemple, son exposition à la menace, ses contraintes en confidentialité, intégrité ou disponibilité) [Car00, MSL10]. Ensuite, plusieurs méthodologies [WXX+08, KLI08, KCBCD10, ZLYS10, AZ12] et métriques [CM05, SAS06, SBW07a, SSBW09, KCBCD10, GDJ+12] permettent de sélectionner les réponses qui seront effectivement déployées sur le système en essayant de trouver un équilibre entre les effets positifs de la réponse (par exemple, contre les attaques potentielles, en s'appuyant sur un modèle d'attaque), ses effets de bord (par exemple, contre les effets négatifs sur le système, en s'appuyant sur un modèle fonctionnel) et son coût.

Pour conclure, un grand nombre de critères peuvent être pris en compte pour calculer et sélectionner une réponse à une attaque. La première étape nécessaire pour pouvoir prendre une décision est d'effectuer une évaluation de risque, soit statique, pour prévenir des attaques potentielles, en phase de conception du système, soit dynamique, pour prévenir des attaques en cours dans un système en cours d'exploitation. Ce modèle peut permettre à la fois d'identifier les attaques auxquelles il faut répondre, mais aussi de savoir l'impact des réponses sur ces attaques potentielles.

A.3 Calcul de remédiations aux chemins d'attaque logiques

Les modèles d'attaque sont un support intéressant permettant de calculer des remédiations. Nous allons présenter ici une méthodologie permettant de remédier les chemins d'attaque les plus significatifs extraits d'un graphe d'attaque logique. Un graphe d'attaque logique décrit précisément toutes les actions qui peuvent être faites par un attaquant dans un système afin de compromettre toutes les cibles d'attaques possibles. Au contraire, un chemin d'attaque est un extrait du graphe complet qui ne contient que les chemins probables et/ou les plus impactants qui visent à compromettre *une* cible d'attaque. Corriger un chemin d'attaque permet donc de protéger une cible critique d'un système pour être sûr qu'elle ne soit pas compromise par des attaques connues.

La figure A.1 représente le processus de remédiation de chemins d'attaque. La première étape est la génération du graphe d'attaque logique, à l'aide d'un moteur de graphe d'attaque, à partir d'un modèle topologique du système. À partir de ce modèle sont extraits les chemins d'attaque. Chaque chemin d'attaque est associé à un score dépendant de sa criticité et de sa probabilité d'occurrence. Celui-ci permet de classer les chemins d'attaque entre eux. C'est alors que commence le processus de remédiation pour chaque chemin d'attaque qui doit être corrigé.

Un chemin d'attaque est un graphe logique `ET-OU`. Le fait logique contenu dans chaque noeud du graphe est vérifié uniquement si la conjonction ou la disjonction (suivant le type de noeuds) de ses parents est vérifiée. Comme les préconditions (les noeuds sans parents) n'ont pas de parents, elles ne sont pas déduites depuis d'autres noeuds. Ce sont donc les premières conditions à partir desquelles l'ensemble des autres noeuds est déduit. Ainsi, ce sont les seuls noeuds où l'on peut appliquer des remédiations. Un algorithme permet de calculer les ensembles de préconditions suffisantes,

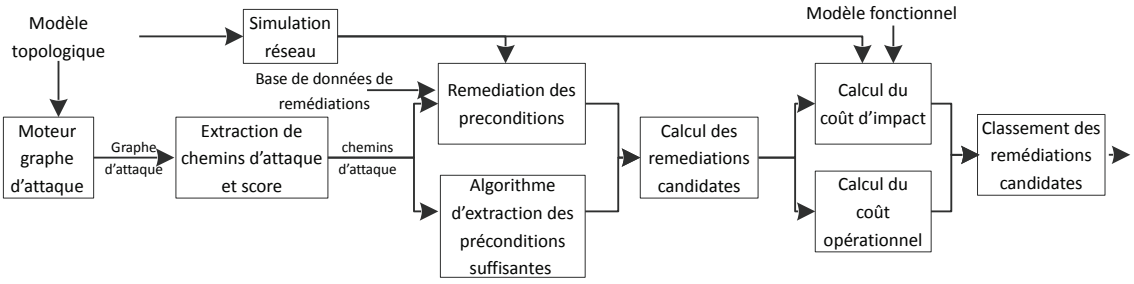


FIGURE A.1 – Processus de remédiation

si corrigées, pour protéger la cible du chemin d'attaque. Chaque ensemble possible contient une conjonction de remédiations qui doivent être toutes corrigées pour protéger la cible. Chaque remédiation décrit une condition qui permet à l'attaquant de faire une attaque. Le processus de remédiation s'appuie sur une base de données qui contient les remédiations connues associées à des conditions d'attaque. À partir des ensembles de préconditions suffisantes et des remédiations possibles de chaque précondition, on calcule les remédiations candidates, permettant de corriger le chemin d'attaque.

Pour comparer plusieurs remédiations permettant de corriger un même chemin d'attaque, le processus de remédiation comporte un composant qui évalue le coût de la remédiation. Le coût d'une remédiation est divisé en deux composants qui peuvent être exprimés en unités monétaires: le coût opérationnel et le coût d'impact. Le coût opérationnel est constitué (1) du coût direct de la remédiation (par exemple, le coût d'un patch ou d'une signature d'attaque), (2) des coûts de déploiement (par exemple, la charge de travail nécessaire à déployer la remédiation sur la machine concernée), (3) des coûts de test (par exemple, vérifier après un correctif que les applicatifs fonctionnent correctement et que la vulnérabilité est corrigée), (4) des coûts de maintenance (par exemple, l'augmentation, suite à l'ajout d'une nouvelle signature, de la charge CPU et du temps de traitement des alertes générées). Le deuxième composant du coût, le coût d'impact, peut être calculé à l'aide d'un modèle fonctionnel du système. Il détaille le coût que pourrait avoir sur le système le déploiement de cette nouvelle remédiation. Pour cela, la remédiation candidate potentielle est déployée sur un réseau simulé sur lequel on analyse son impact sur les services légitimes, grâce au modèle fonctionnel. La combinaison du coût opérationnel et d'impact permet d'assigner à chaque remédiation candidate un coût global et donc de les classer entre elles.

Nous venons donc de présenter un processus permettant de calculer les remédiations à des chemins d'attaque extraits depuis un graphe d'attaque. Les graphes d'attaque ont été très utilisés pour évaluer de façon proactive la sécurité des systèmes d'information. Nous avons choisi de les utiliser ici plutôt comme support permettant de proposer des remédiations pour empêcher des attaques possibles. Le fait de choisir seulement certains chemins d'attaque au lieu de corriger tout le graphe permet d'avoir beaucoup moins de problèmes de performance pour calculer les remédiations possibles.

Le formalisme utilisé par les graphes d'attaque logiques est déterministe et statique. Ainsi, il est nécessaire de l'étendre en un modèle quantitatif et dynamique, pour pouvoir représenter des attaques en cours et permettre de généraliser ce processus de calcul de remédiation. L'un des indicateurs clés permettant de choisir les chemins d'attaque à corriger est leur risque, qui combine la probabilité d'occurrence avec l'impact qu'ils peuvent avoir sur le système. Ainsi, afin de pouvoir généraliser ce processus de calcul de remédiation en un processus de calcul de réponse, la première étape est de construire un modèle d'évaluation dynamique de risque permettant de savoir les attaques les plus risquées en cours.

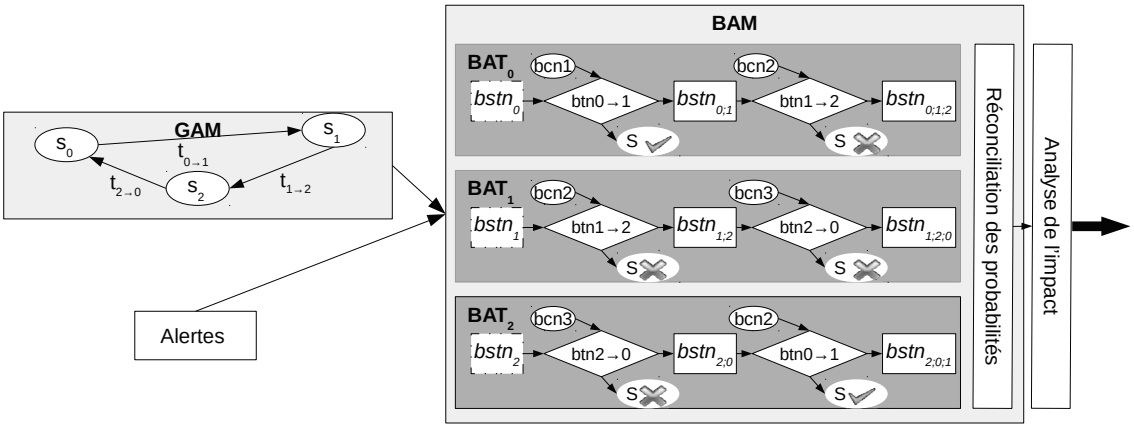


FIGURE A.2 – Architecture du modèle d'attaque bayésien *BAM*

A.4 BAM, le modèle d'attaque bayésien

Il est apparu, avec l'étude de l'état de l'art et la construction de la méthodologie de calcul de remédiation, que la première étape nécessaire pour calculer des réponses est de faire une analyse de risque technique. Or, il n'y a pas de modèle d'analyse de risque satisfaisant dans l'état de l'art. Le risque est constitué de deux composants. Le premier est la probabilité d'occurrence des attaques. Le deuxième est l'impact des futures attaques probables. L'impact est un composant qui dépend beaucoup du système étudié et de l'organisation auquel il appartient. Son calcul peut reposer sur un modèle fonctionnel du système, mais ne peut pas être complètement automatisé. Au contraire, les méthodologies permettant d'évaluer la probabilité des attaques ne dépendent pas du système. Comme vu dans l'état de l'art, il existe plusieurs modèles permettant de faire de l'analyse statique de risque, par exemple les graphes d'attaque. Par contre, il n'y a pas de modèle permettant de faire de l'analyse dynamique de risque avec une granularité suffisante et qui peut s'appliquer à n'importe quel système. Le modèle d'attaque bayésien *BAM* que nous présentons ici vise à évaluer le risque apporté par les attaques possibles et en cours dans un système, en pouvant s'appliquer à n'importe quel système. Il est construit à partir du modèle d'attaque générique *GAM* décrit précédemment, et peut ainsi s'appliquer à différents domaines et avec une description des attaques plus ou moins haut niveau.

L'architecture du modèle d'attaque bayésien *BAM* est présentée dans la Figure A.2. Ce modèle est construit à partir d'un *GAM*, et d'un ensemble d'alertes. Il est constitué d'un ensemble d'arbres bayésiens appelés *BAT* dont chacun part d'une source d'attaque potentielle. Le fait de construire plusieurs modèles chacun partant d'une source d'attaque permet de casser les cycles possibles du modèle d'attaque pris en entrée, en énumérant les chemins possibles à partir de la source d'attaque. Dans le *BAM*, on représente une transition par un ensemble de noeuds permettant de modéliser: l'étape source, l'étape destination, la transition elle-même, ses conditions et les sondes de détection, si elles existent. La figure A.3 montre la représentation dans le *BAM* d'une transition entre les états tn_n et tn_{n+1}. Celle-ci comporte deux conditions et une sonde de détection.

Le modèle bayésien complet est construit en ajoutant récursivement les transitions d'attaque à partir de chaque source d'attaque, dans le *BAT* correspondant. Ensuite, les probabilités sont réconciliées par une fonction qui garde le maximum des probabilités correspondant à un état du *GAM*, ce qui correspond au pire cas possible de compromission de cet état. C'est cette probabilité qui pourra être utilisée pour calculer le risque associé à cet état.

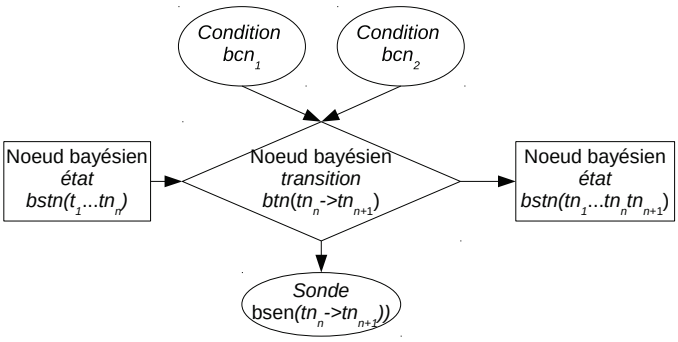


FIGURE A.3 – Représentation d'une transition du modèle d'attaque bayésien

Pour construire le modèle d'attaque bayésien *BAM*, 6 paramètres entrent en jeu. Ceux-ci peuvent être, soit des paramètres probabilistes du modèle *GAM* pris en entrée, soit des paramètres intrinsèques du modèle bayésien. Nous étudions la sensibilité des résultats du modèle *BAM* (les probabilités de compromission des états) en fonction des variations de ces paramètres. Le tableau A.1 présente les paramètres impliqués et résume les résultats de cette étude de sensibilité. Le résultat le plus intéressant de cette analyse, pour savoir où déployer des réponses, est l'*influence du paramètre sur le classement* des états compromis. Cette colonne décrit l'impact qu'a la variation de chaque paramètre sur le classement des probabilités de compromission des états, sur tout l'intervalle de variation. Ce classement, pondéré par le risque associé aux états, va donner l'ordre dans lequel les attaques en cours seront analysées par les opérateurs de sécurité. La colonne *influence sur la probabilité* décrit l'impact de la variation du paramètre, sur les valeurs absolues des probabilités de compromission des états.

Trois paramètres ont un impact significatif sur le classement des probabilités de compromission des états. Or, soit ceux-ci peuvent être estimés assez précisément par les résultats d'une analyse de risque antérieure (par exemple, `probability-attack-source`), soit ils n'ont pas d'impact sur leur intervalle usuel d'utilisation (par exemple, `probability-unknown-attack` ou `probability-new-transition`). Ce résultat est rassurant quant aux résultats du modèle bayésien *BAM*. En effet, ceux-ci permettent d'obtenir le bon ordre dans les priorités sur les attaques les plus probables, même avec une incertitude sur les paramètres en entrée. En revanche, concernant les valeurs absolues des probabilités, plusieurs des paramètres sont très sensibles sur leur intervalle d'utilisation usuel (par exemple, `false-positive` ou `probability-attack-source`). Même avec une petite incertitude sur ces paramètres, les probabilités de compromission des états peuvent beaucoup changer.

Ainsi, le modèle *BAM* est un modèle bayésien qui peut être construit à partir de n'importe quel modèle d'attaque (cyclique ou acyclique) pouvant être spécifié sous la forme d'un modèle d'attaque générique *GAM*. Il est efficace pour des modèles jusqu'à des tailles moyennes. Ses résultats sont assez précis pour classer les probabilités de compromission des états, même avec de l'incertitude sur les paramètres. Cependant, il est nécessaire d'avoir une connaissance très précise sur la valeur des paramètres pour avoir des résultats précis. Un autre défaut de ce modèle est qu'il mélange l'augmentation de probabilité due aux attaques qui se sont déjà produites et des futurs possibles.

TABLE A.1 – Sensibilité des résultats du *BAM*, en fonction de ses paramètres.

Nom	**Description**	**Intervalle de variation**	**Influence sur le classement**	**Influence sur la probabilité**
`false-positive`	Taux de faux positif des sondes de détection.	$[0-1]$	Faible impact.	Impact important, uniquement sur les machines compromises, pour les faibles valeurs du paramètre.
`false-negative`	Taux de faux négatif des sondes de détection.	$[0-1]$	Faible impact.	Impact moyen, uniquement sur les machines compromises, pour les valeurs moyennes du paramètre.
`probability-attack-source`	Probabilité à priori qu'une attaque commence depuis un état.	$[0-1]$	Impacte le classement quand d'autres états ont une probabilité d'être source d'attaque plus forte que la vraie source d'attaque.	Impact important sur les probabilités des noeuds non compromis.
`probability-unknown-attack`	Probabilité qu'une attaque inconnue permette d'atteindre un état, sans passer par une transition existante.	$[0-0.5]$	Impact uniquement pour les valeurs élevées du paramètre.	Impact moyen sur les états compromis ou non-compromis.
`nbSteps`	Paramètre du *GAM* qui limite la profondeur du nombre de transitions à ajouter aux *BAT*.	$[[0-7]]$	Presque pas d'impact.	Impact moyen sur les probabilités des états compromis.
`probability-new-transition`	Probabilité qu'un attaquant propage son attaque en une nouvelle transition d'attaque.	$[0-1]$	Impact uniquement aux valeurs extrêmes du paramètre.	Impact important sur les probabilités des états compromis.

A.5 HRAM, le modèle hybride d'évaluation dynamique de risque

Nous avons présenté un modèle d'attaque bayésien, *BAM*, qui peut être utilisé pour l'analyse dynamique de risque. Mais celui-ci possède plusieurs limitations: (1) il ne peut être utilisé que pour des systèmes de taille petite ou moyenne, (2) il ne prend pas en compte l'ordre dans lequel les alertes sont reçues, (3) pour chaque état, la sortie du *BAM* est une unique valeur qui cumule la probabilité d'être déjà compromis, en fonction des alertes reçues, et celle de l'être dans un futur proche. Il n'est pas possible de faire la distinction entre ces deux informations, alors qu'elles peuvent toutes deux être utiles pour un opérateur de sécurité. Le modèle que nous proposons désormais est un modèle hybride, *HRAM*, qui combine lui aussi les modèles d'attaque et les réseaux bayésiens pour faire de l'évaluation dynamique de risque. Il est constitué de la combinaison de deux types de modèles: (1) Les modèles de corrélation dynamique de risque (*DRCM*) qui corrèlent une chaine d'alertes avec la connaissance du système pour analyser les attaques qui ont été détectées et produisent les probabilités de compromission des états. Ces modèles visent à évaluer la probabilité des menaces, pour savoir d'où viennent les attaques. (2) Les modèles d'évaluation de risque des futurs possibles (*FRAM*) prennent en compte le statut actuel d'attaque du système (donné par les modèles *DRCM*), ainsi que les transitions possibles (connues dans le modèle d'attaque), afin d'évaluer quelles sont les attaques futures qui sont susceptibles de se produire. Ces modèles visent à atténuer les menaces, en corrigeant les prochaines étapes d'attaque qui sont les plus risquées. La séparation des problèmes

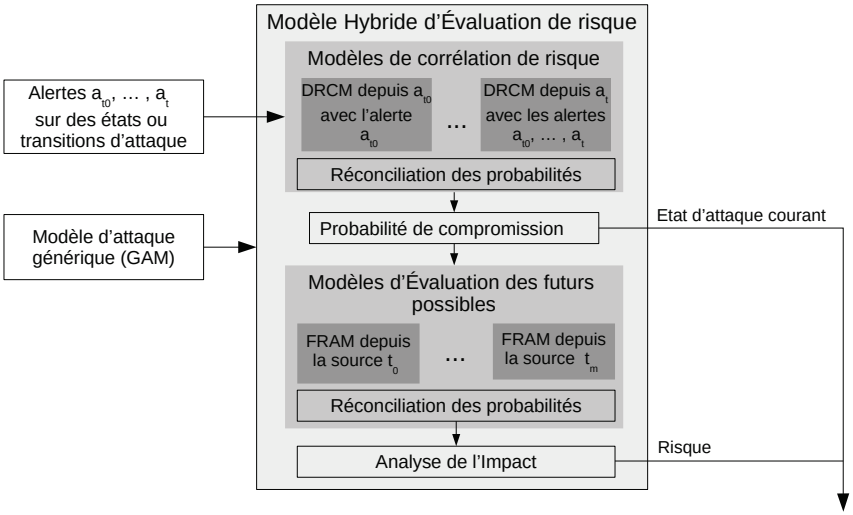


FIGURE A.4 – Architecture du modèle d'attaque hybride *HRAM*

en deux modèles distincts permet une nette amélioration des performances en termes de nombre d'états qu'il est possible de traiter.

L'architecture du modèle d'attaque hybride *HRAM* est présentée dans la Figure A.4. Comme le modèle *BAM*, le modèle *HRAM* est construit à partir d'un *GAM*, et d'un ensemble d'alertes. Le premier type de modèles, les *DRCMs* sont construits à partir des alertes entrantes. À chaque nouvelle alerte est construit un modèle qui essaye d'expliquer l'alerte la plus récente reçue, en tenant compte des transitions possibles dans le système et des alertes qui ont été reçues auparavant. Si des alertes ne sont pas retrouvées dans le modèle construit à partir de l'alerte la plus récente, c'est qu'elles ne font pas partie de la même attaque, ou bien qu'une attaque conjointe n'a pas encore convergé. Dans ce cas, on conserve plusieurs *DRCMs* en parallèle, correspondant aux différentes attaques en parallèle dans le système. La réconciliation des probabilités permet de remonter les probabilités de compromission des états du système, à partir des *DRCMs* gardés en parallèle. Pour chaque état du système qui peut être considéré comme une source potentielle d'attaque (soit à cause de sa probabilité à priori, soit à cause de son état courant dans le système issu de l'analyse des *DRCMs*), on construit un modèle des futurs possibles (*FRAM*). À l'aide de la fonction d'analyse d'impact, le *FRAM* va permettre de savoir les futurs probables les plus risqués.

Chaque *DRCM* est construit à partir de la sonde de détection correspondant à l'alerte la plus récente reçue à l'instant présent. On ajoute alors récursivement toutes les transitions permettant d'expliquer la levée de cette alerte. Pour des raisons des performances, comme il n'est pas possible de garder tous les chemins d'attaque possibles, des heuristiques permettent de ne garder que les chemins les plus probables (les chemins sur lesquels il n'y a pas beaucoup de sondes de détection qui n'ont pas levé d'alertes à la suite). Au contraire, un *FRAM* est construit à partir d'une source d'attaque possible. Ensuite on ajoute toutes les transitions possibles qui ne se sont pas déjà produites (une transition déjà détectée n'est plus un futur possible). Comme il est irréaliste de vouloir prédire les attaques futures de nombreuses étapes à l'avance, il n'est pas nécessaire d'ajouter un grand nombre de transitions successives au modèle. Celui-ci pourra être mis à jour, en fonction des nouvelles alertes qui arrivent. Enfin, les probabilités des futures attaques possibles sont réconciliées par une fonction qui garde le maximum des probabilités correspondant à un état du *GAM*, ce qui correspond au pire cas possible de compromission de cet état. Comme pour le *BAM*, c'est cette probabilité qui pourra être utilisée pour calculer le risque associé à cet état.

TABLE A.2 – Sensibilité des résultats du *HRAM*, en fonction de ses paramètres.

Nom	**Description**	**Intervalle de variation**	**Influence sur le classement**	**Influence sur la probabilité**
`false-positive`	Taux de faux positif des sondes de détection.	$[0-1]$	Pas d'impact.	Impact faible.
`false-negative`	Taux de faux négatif des sondes de détection.	$[0-1]$	Impact important au-delà de 0.05.	Impact important pour les états non compromis.
`probability-attack-source`	Probabilité à priori qu'une attaque commence depuis un état.	$[0-1]$	Presque pas d'impact.	Impact moyen sur les états non compromis.
`probability-unknown-attack`	Probabilité qu'une attaque inconnue permette d'atteindre un état, sans passer par une transition existante.	$[0-0.5]$	Pas d'impact.	Impact faible sur l'intervalle de variation le plus fréquent.
`max-number-no-info-to-keep`	Paramètre du *DRCM* qui limite le nombre successif de sondes sans informations à garder.	$[[0-10]]$	Pas d'impact.	Pas d'impact quand la valeur est strictement supérieure au nombre de détections non détectées dans les scénarios d'attaque.
`max-number-no-alert-to-keep`	Paramètre du *DRCM* qui limite le nombre successif de sonde qui n'ont pas levées d'alertes à garder.	$[[0-10]]$	Pas d'impact.	Pas d'impact.
`nbSteps-possible-futures`	Paramètre du *FRAM* qui limite le nombre successif de transitions d'attaque dans chaque *FRAM*.	$[[0-4]]$	Pas d'impact.	Impact très faible.
`probability-new-transition`	Paramètre du *FRAM* qui représente la probabilité qu'une attaque inconnue permette d'atteindre un état, sans passer par une transition existante.	$[0-1]$	Pas d'impact.	Impact fort sur quelques-uns des états non compromis (les futurs possibles).

De même que pour le *BAM*, pour construire le modèle hybride *HRAM*, 8 paramètres principaux entrent en jeu. Ceux-ci peuvent être, soit des paramètres probabilistes du modèle *GAM* pris en entrée, soit des paramètres intrinsèques au modèle bayésien hybride. Le tableau A.2 présente les paramètres impliqués et résume les résultats de l'étude de sensibilité des résultats du *HRAM* en fonction de variation de ces paramètres.

Aucun paramètre n'a d'impact significatif sur le classement des probabilités de compromission des états, sur leur intervalle d'utilisation le plus courant. Seul un paramètre (`false-negative`) a un impact important sur la valeur absolue des probabilités de compromission. Trois paramètres ont un impact moyen (`probability-attack-source`, `probability-unknown-attack` et `probability-new-transition`). Mais ceux-ci peuvent être estimés assez précisément par les résultats d'une analyse de risque antérieure (par exemple, `probability-attack-source`), ou bien ils permettent d'ajuster le niveau de sécurité voulu par un administrateur dans son système (`false-negative`, `probability-unknown-attack` et `probability-new-transition`). Ainsi, pour le modèle *HRAM*, le classement des probabilités de compromission des états n'est pas impacté par la variation des paramètres. La valeur absolue de ces probabilités est seulement moyennement impactée, si l'incertitude sur les paramètres n'est pas trop forte, mis à part pour le paramètre `false-negative` qui peut ajouter quelques faux positifs (donner des probabilités de compromission fortes à des états non compromis), s'il est trop élevé.

Donc, le modèle hybride *HRAM* est un modèle bayésien qui peut être construit à partir de n'importe quel modèle d'attaque (cyclique ou acyclique) pouvant être spécifié sous la forme d'un modèle d'attaque générique *GAM*. Il est efficace pour des modèles jusqu'à des tailles plus grandes que le *BAM*. Ses résultats permettent de classer exactement les probabilités de compromission des états et d'avoir des valeurs de probabilité assez précises, même avec de l'incertitude sur les paramètres. Ce modèle ajoute des d'améliorations par rapport au *BAM*, qui sont permises par la séparation des modèles *DRCM* pour analyser le passé et les *FRAM* pour analyser les futurs possibles.

A.6 Application au domaine de la sécurité informatique

Nous avons développé deux modèles d'analyse dynamique de risque: *BAM*, le modèle d'attaque bayésien et *HRAM*, le modèle hybride d'évaluation dynamique de risque. Ceux-ci sont construits à partir du modèle d'attaque générique *GAM*. Afin d'évaluer et de comparer la pertinence des résultats fournis par ces modèles, nous les appliquons désormais au domaine de la sécurité informatique et réseau. Dans ce domaine, les attaques potentielles sont généralement spécifiées à l'aide d'un graphe d'attaque, soit logique, soit topologique. Les graphes d'attaque logiques sont plus détaillés que les graphes d'attaque topologiques, mais ils sont par conséquent aussi beaucoup plus gros. Ainsi, ils ne peuvent pas être utilisés pour les grands systèmes d'information. Nous nous intéresserons donc ici à des modèles d'attaque génériques, construits à partir de graphes d'attaque topologiques.

La première étape de ce processus est de générer le graphe d'attaque topologique. Ce graphe topologique est constitué de noeuds topologiques (qui représentent par exemple des machines) et d'étapes d'attaque (qui représentent par exemple l'exploitation d'une vulnérabilité sur une machine). Un graphe d'attaque topologique peut être généré soit depuis un graphe d'attaque logique, tel que ceux générés par exemple avec MulVAL [OGA05a] ou directement à l'aide d'un moteur de graphe d'attaque topologique tel que TVA [JNK+11]. Cependant, les moteurs de graphes d'attaque actuels ont plusieurs limitations: (1) ils ont une représentation soit logique, soit topologique des attaques, mais il n'est pas possible d'avoir les deux en même temps, (2) ils ne passent pas à l'échelle (ils ne peuvent pas être distribués et sont limités en nombre de machines et vulnérabilités), (3) pour les moteurs topologiques, il n'est pas possible de changer les règles d'attaque.

Ainsi, nous avons développé un nouveau moteur de graphe d'attaque, *SAGE* qui résout ces problèmes. (1) c'est un moteur topologique, mais qui, pour chaque étape d'attaque, conserve les conditions qui ont été nécessaires pour qu'elle se produise, (2) il peut être distribué sur plusieurs noeuds de calcul pour passer à l'échelle, (3) les règles d'attaque peuvent être modifiées. La Figure A.5 résume l'architecture du moteur *SAGE*. Celui-ci prend deux types d'entrées: (1) des entrées statiques (les paramètres de configuration du moteur et les règles d'attaque) qui sont chargées au lancement du moteur, (2) des entrées dynamiques (les vulnérabilités, la topologie réseau et la matrice de flux) qui sont stockées dans une base de données orientée graphe et qui peuvent être mises à jour dynamiquement. Le graphe est calculé en fonction des requêtes effectuées par l'opérateur, souvent par l'intermédiaire de l'interface graphique.

À partir du graphe topologique, il est possible de construire un modèle d'attaque générique *GAM*, permettant ensuite l'utilisation des modèles d'analyse dynamique de risque. Pour cela, les états et les transitions du *GAM* sont les noeuds topologiques et les étapes d'attaque du *BAM*. Les conditions sont celles associées aux étapes d'attaque. Dans un graphe d'attaque topologique, les conditions sont généralement la présence d'une vulnérabilité sur une machine. Les sondes de détection sont celles qui sont déployées dans le système, configurées pour alerter en cas d'exploitation d'une vulnérabilité connue.

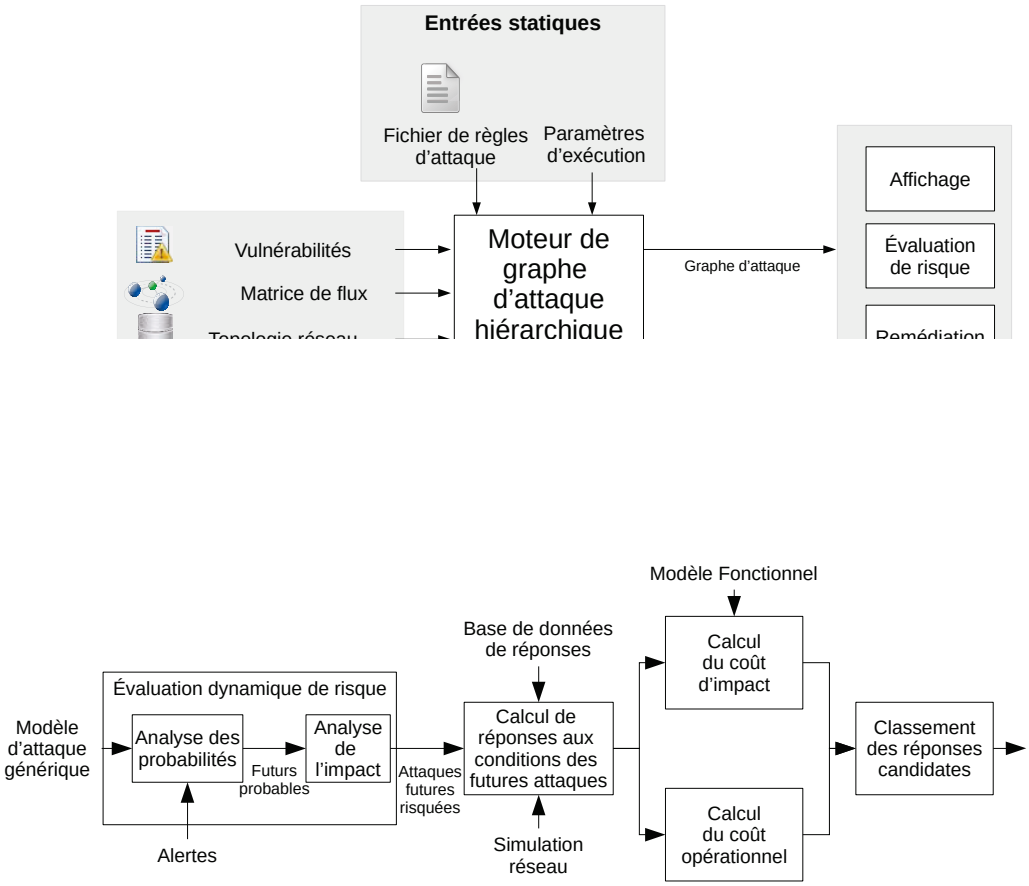


FIGURE A.6 – Processus de calcul de réponses

Une fois que le modèle générique *GAM* est construit, il est possible d'effectuer l'analyse dynamique de risque, à l'aide des deux modèles construits précédemment. Pour cela, nous commençons par générer plusieurs topologies réseau aléatoires de taille variable. Sur ces topologies nous établissons des scénarios d'attaque de plusieurs étapes successives auxquelles correspondent des scénarios de détection (comprenant en plus des faux positifs et faux négatifs). Il est alors possible de comparer les résultats théoriques connus dans les scénarios d'attaque avec les résultats pratiques, les probabilités de compromission calculées par les modèles d'évaluation de risque. Les résultats produits par le modèle bayésien *BAM* sont bons, mais uniquement pour les petites topologies (< 25 machines): pour de telles topologies, ils permettent de distinguer les machines compromises des machines non compromises. Les résultats du modèle hybride *HRAM* sont meilleurs, car ils sont très stables, quelle que soit la taille de la topologie. Pour l'analyse du passé, le *HRAM* donne une claire séparation entre les machines effectivement compromises et celles qui ne le sont pas. En prédiction des futurs possibles, cette séparation est moins nette, car il est évidemment plus compliqué de prédire les attaques futures, mais le modèle permet de prédire, parmi toutes celles qui sont possibles, celles qui ont le plus de chance de se produire.

Une fois que nous avons validé l'intérêt et les résultats produits par les modèles bayésiens (en particulier, ceux du modèle hybride *HRAM*), nous pouvons analyser comment ces modèles peuvent être utilisés pour calculer des réponses à des attaques informatiques en cours. Pour cela, nous généralisons la méthodologie de calcul de remédiation en une méthodologie de calcul de réponses.

La Figure A.6 représente le processus de calcul de réponses pour les attaques en cours. En comparaison avec la méthodologie de calcul de remédiation, il y a deux composants principaux qui changent:

L'extraction de chemins d'attaque d'attaque et score, qui est remplacée par l'évaluation dynamique de risque. En effet, pour le calcul de remédiation, il n'y a pas d'attaques en cours, on ne corrige donc que les attaques les plus risquées (impactantes et probables). Au contraire, pour le calcul de réponse, il s'agit de corriger les attaques en cours et leurs futurs probables. Le composant d'évaluation dynamique de risque qui a été décrit précédemment repose sur l'un des modèles bayésiens (*BAM* ou *HRAM*). Il calcule les futurs probables et à l'aide d'une fonction d'analyse d'impact en déduit les attaques futures risquées.

La remédiation des préconditions et l'extraction des préconditions suffisantes, qui sont remplacées par le calcul de réponses aux conditions des futures attaques. Ce nouveau composant a pour but de calculer les réponses permettant d'empêcher en partie ou complètement une attaque future potentielle. La solution que nous proposons est de s'appuyer sur les modèles des futurs possibles *FRAMs* du modèle hybride, afin de protéger les chemins d'attaque qui engendrent la plus grande probabilité de compromission de l'actif visé. Pour cela, nous parcourons les conditions présentes sur ce chemin de la "plus nécessaire" à la "moins nécessaire" et calculons les réponses possibles, à l'aide de la base de données de réponse.

Nous avons donc appliqué les modèles d'évaluation dynamique de risque développés précédemment au domaine de la sécurité informatique. Pour cela, nous avons généré un graphe topologique à l'aide du moteur *SAGE*, puis l'avons transcrit au format générique du *GAM*, pour pouvoir construire les modèles d'analyse dynamique de risque. Nous avons pu évaluer que les résultats calculés par ces modèles étaient satisfaisants. Enfin, ces modèles nous ont permis de généraliser le processus de calcul de remédiation en un processus de calcul de réponses aux attaques en cours.

A.7 Conclusion

Pour conclure, nous avons développé dans cette thèse de doctorat un système d'analyse statique et dynamique de risque basé sur les modèles d'attaque. Celui-ci prend en compte la connaissance à priori sur le système et les alertes qui sont levées dans celui-ci. La première contribution principale de cette thèse est la définition de modèles d'évaluation dynamique de risque, qui combinent l'analyse du passé (expliquer les alertes reçues jusqu'à l'instant présent et en déduire le niveau de compromission des actifs du système), et l'analyse des futurs possibles (essayer de prévoir les attaques à venir qui sont les plus probables ou risquées). Ces modèles possèdent au moins trois améliorations significatives par rapport à l'état de l'art. La première est la possibilité de les construire depuis des modèles d'attaque contenant des cycles, ce qui correspond à la majorité des modèles d'attaque réels. Deuxièmement, le modèle hybride *HRAM* sépare dans des modèles appropriés l'analyse du passé et du futur, ce qui évite de confondre la probabilité d'être déjà compromis avec celle de l'être dans un futur proche. Enfin, il permet aussi de distinguer plusieurs attaques simultanées distinctes. La deuxième contribution de ces travaux est l'amélioration du passage à l'échelle des modèles d'évaluation dynamique de risque. En utilisant des heuristiques permettant de tenir compte uniquement des chemins d'attaque significatifs, nous limitons l'impact négatif de l'explosion des cycles. La troisième contribution de cette thèse est la définition d'une méthodologie permettant de calculer des réponses en s'appuyant sur les modèles d'attaque. Tout d'abord, nous avons construit cette méthodologie pour corriger les chemins d'attaque potentiels qui peuvent exister dans un réseau. Puis, nous l'avons étendue aux attaques en cours dans un réseau, en nous basant sur les résultats de l'analyse dynamique de risque.

Quatre perspectives principales peuvent être données à ces travaux. La première concerne la difficulté d'estimer les paramètres probabilistes du modèle. Même si l'analyse de sensibilité des paramètres a montré que la plupart des paramètres n'étaient pas trop sensibles, au moins pour le classement des probabilités de compromission, il faut faire attention au choix de certains paramètres pour obtenir des résultats précis. La première piste de solution est de trouver d'autres sources de données pour compléter les paramètres pris en entrée par le modèle. La deuxième est d'utiliser les réseaux bayésiens pour faire de l'apprentissage des paramètres du réseau, par exemple après la confirmation/infirmation d'une attaque par un opérateur. La deuxième perspective concerne le passage à l'échelle qui, s'il est bien meilleur que l'état de l'art, n'est pas encore suffisant pour gérer de très gros réseaux. Une piste est de regrouper les noeuds qui possèdent des configurations similaires (vulnérabilités, accès autorisés, *etc.*). Il serait aussi possible de paralléliser les calculs effectués sur les réseaux bayésiens. La troisième perspective vise à compléter la méthodologie de calcul de réponse s'appuyant sur les modèles dynamiques de risque. L'approche que nous avons présentée est assez simple et devrait être validée avec des données d'attaque réelle pour s'assurer que les contre-mesures proposées sont bonnes. Enfin, la dernière perspective est d'appliquer les modèles d'évaluation dynamique de risque à d'autres domaines que la sécurité informatique classique. En effet, le fait de se baser sur un modèle d'attaque générique pour construire nos modèles permet d'appliquer ceux-ci à d'autres types de modèles d'attaque, par exemple des scénarios d'attaque de plus haut niveau ou encore de les appliquer à la modélisation d'attaques logiques-physiques. Ainsi, il faudrait valider que les modèles que nous avons construits peuvent s'appliquer et donnent des résultats cohérents sur ces autres domaines d'activité.