Diagnosing accidental and malicious events in industrial control systems

Edwin Bourget

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

THÈSE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPÉRIEURE MINES-TÉLÉCOM ATLANTIQUE
BRETAGNE PAYS DE LA LOIRE - IMT ATLANTIQUE

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : Informatique

Par

Edwin BOURGET

Diagnosing accidental and malicious events in industrial control systems

Thèse présentée et soutenue à Cesson-Sévigné, le 16 juin 2020
Unité de recherche : LabSTICC
Thèse N° : 2020IMTA0179

Rapporteurs avant soutenance :

Mohamed Mosbah	Professeur, LABRI
Guy Pujolle	Professeur, Sorbonne Université

Composition du Jury :

Président :	Abdelmalek Benzekri	Professeur, Université Paul Sabatier de Toulouse
Examinateurs :	Olivier Festor	Directeur de recherche INRIA, Télécom Nancy
	Mohamed Mosbah	Professeur, Bordeaux INP
	Guy Pujolle	Professeur, Sorbonne Université
Dir. de thèse :	Frédéric Cuppens	Professeur, IMT Atlantique
Co-dir. de thèse :	Nora Cuppens	Directrice de recherche, IMT Atlantique

Invité(s) :

Samuel Dubus	Ingénieur de recherche, Nokia

LIST OF PUBLICATIONS

International conferences

- Edwin Bourget, Frédéric Cuppens, Nora Cuppens-Boulahia, Samuel Dubus, Simon N. Foley, Youssef Laarouchi. **Probabilistic Event Graph to Model Safety and Security for Diagnosis Purposes**. DBSec 2018: 38-47
- Simon N. Foley, Fabien Autrel, Edwin Bourget, Thomas Clédel, Stephane Grunenwald, Jose Rubio-Hernan, Alexandre Kabil, Raphael Larsen, Vivien M. Rooney, Kirsten Vanhulst. **Science Hackathons for Cyberphysical System Security Research: Putting CPS testbed platforms to good use**. CPS-SPC@CCS 2018: 102-107
- Hélène Le Bouder, Gaël Thomas, Edwin Bourget, Mariem Graa, Nora Cuppens, Jean-Louis Lanet. **Theoretical Security Evaluation of the Human Semantic Authentication Protocol**. ICETE (2) 2018: 498-505

RÉSUMÉ

Ce que l'on appelle Industrie 4.0 (d'après la quatrième révolution industrielle), Industrie du futur ou Smart Factory est un changement de paradigme qui s'opère dans les outils de production industrielle : leur informatisation. L'Industrie 4.0 correspond à une automatisation des processus industriels à travers des échanges de données toujours plus volumineux, et l'intégration de nouvelles technologies telles que l'infonuagique ou l'internet des objets. Une conséquence de cela est la généralisation de systèmes *cyber-physiques* pour contrôler et superviser ces processus.

De nombreux avantages accompagnent ce nouveau paradigme, mais également de nouveaux défis à traiter. En ce qui concerne la sécurité en particulier, les changement sont conséquents.

Historiquement, les systèmes industriels étaient conçus pour être relativement fermés du monde extérieur. On pourrait dire qu'une simple cloture et un gardien étaient nécessaires pour en assurer la sécurité. En effet, un attaquant devait avoir un accès physique au système pour pouvoir l'endommager. Les différents composants des systèmes industriels ont donc été conçu avec le postulat que cette protection physique était déployée et ils n'embarquaient aucune forme de sécurité.

Les systèmes cyber-physiques, cependant, outrepassent le besoin d'être physiquement présent dans le système pour pouvoir l'influencer. Un attaquant peut donc utiliser les mêmes canaux que le système de contrôle pour accéder au système et l'attaquer. Des attaques sur des systèmes cyber-physiques ont déjà eu lieu par le passé comme l'attaque STUXNET sur le programme nucléaire iranien [27, 69, 1] en 2010 ou l'attaque sur le système de distribution d'électricité ukrainien en 2015 [68, 28].

Ce doctorat a été réalisé dans le cadre de la chaire Cyber-CNI, Cybersécurité des Infrastructures Nationales Critiques. Cette chaire a pour but d'améliorer la sécurité des systèmes critiques. Les travaux présentés dans cette thèse se concentrent sur le diagnostic d'évènements de sûreté et de sécurité. Autrement dit lorsque des incidents, qui peuvent être de nature accidentelle ou malveillante, surviennent dans un système industriel, nous cherchons à fournir des informations précises et pertinentes afin qu'un décisionnaire puisse adopter la réaction la plus appropriée. En raison de la nature de l'Industrie 4.0, et que les changements dans l'environnement défini par les systèmes cyber-physiques, diagnostiquer des incidents pose de nouvelles questions, en grande partie à cause des interactions entre sûreté et sécurité, qui sont bien plus importantes qu'auparavant.

D'après Piètre-Cambacédès et Bouissou [95], la sûreté et la sécurité ont quatre types d'interdépendances : dépendance conditionnelle, renforcement, antagonisme et indépendance. La sûreté et la sécurité sont conditionnellement dépendantes lorsque l'une est un prérequis pour l'autre. Le renforcement a lieu lorsqu'elles sont chacune renforcé par le même procédé. L'antagonisme survient lorsque l'augmentation de l'une cause une diminution de l'autre. L'indépendance correspond à une non affectation de l'une par une modification de l'autre.

Un example simple montrant comment sûreté et sécurité peuvent entrer en conflit est l'exemple de la sortie de secours. Si le feu se déclenche dans un bâtiment, il faut que la sortie de secours soit ouverte afin que les gens puisse se mettre à l'abri. Néanmoins, un terroriste pourrait tirer parti de ce mécanisme en déclenchant lui-même une alarme à incendie afin de déverrouiller les sorties de secours et entrer dans le bâtiment. Dans cette situation, la sûreté et la sécurité sont chacune capable de gérer la situation mais arrivent à une conclusion opposée qui peut, dans les deux cas, mettre en péril la sécurité des personnes à l'intérieur du bâtiment. L'objectif dans cette situation est de fournir des informations qui permettent de choisir la réponse la plus appropriée lorsqu'un incident implique à la fois sûreté et sécurité. Dans le cas de la sortie de secours, une question à laquelle le diagnostic doit répondre est « l'alarme est-elle légitime ou bien a-t-elle été déclenchée par un attaquant ? ».

Historiquement, les modèles de sûreté et de sécurité sont monolithiques : ils ne considère que soit la sécurité, soit la sûreté mais pas les deux à la fois. C'est une conséquence de l'absence de besoin de sécurité dans les équipements industriels évoquée lors des paragraphes précédents. Cependant, en raison de la désormais interaction entre la sûreté et la sécurité, il est bien évidemment nécessaire d'utiliser les modèles capables de les considérer en même temps.

Mélanger sûreté et sécurité n'est pas un problème trivial. Les objectifs sont différents et les techniques mis en oeuvre pour les atteindre exploitent des spécificités que la sûreté et la sécurité ne partagent pas nécessairement. Ces comportements peuvent aisément être représentés par des modèles dédiés mais cela rend toute généralisation presque impossible. Gibaudo *et al.* [50] citent par exemple que l'ordre dans lequel des évènements surviennent est fondamental en sécurité alors qu'il peut généralement être ignoré en sûreté de fonctionnement. Gibaudo *et al.* [50] notent également que les modèles de sûreté étant structurellement proches du système, ils ne varient que peu lors de modifications mineures de celui-ci contrairement aux modèles de sécurité qui nécessitent des révisions importantes après le moindre changement. La littérature a parfois tenté d'utiliser des combinaisons de modèles de sûreté et de sécurité [47, 85]. De telles approches ont été rendues possibles au prix de restrictions quant aux interactions entre sûreté et sécurité [47], ou bien en utilisant des modèles très abstraits fournissant peut d'informations [85]. Décrire un modèle suffisamment générique pour pouvoir représenter à la fois la sûreté et la sécurité, tout en fournissant assez d'informations pour réaliser le diagnostic est le

défi relevé dans cette thèse.

La définition du diagnostic est indissociable du contexte dans lequel elle est utilisé. C'est pourquoi les définitions et les objectifs du diagnostics sont non seulement différents entre la sûreté et la sécurité, mais de profondes divergences peuvent également apparaitre entre deux modèles de sûreté ou de sécurité. C'est pourquoi il est nécessaire de définir précisément ce qu'est le diagnostic dans un contexte où sûreté et sécurité sont mélangés. Nous proposons la définitions suivante : *le diagnostic dans un contexte de sûreté-sécurité est la processus de déduire le mécanisme le plus probable ayant causé une condition observée, ainsi qu'en identifier les conséquences*. Cette définition nous permet d'identifier un ensemble de fonctions que doivent réaliser un modèle de diagnostic : de la détection d'incident, de la corrélation d'incident, de l'explication, de l'évaluation de risque, de l'évaluation de vraisemblance, de la prise en compte de contremesures et une utilisation en direct.

Il n'existe que très peu de modèles combinant sûreté et sécurité et aucun n'est dédié au diagnostic. Par conséquent, l'état de l'art présenté dans cette thèse ne se concentrent pas sur la finalité des modèles mais sur les informations qu'ils sont capables de fournir et des fonctions de diagnostic qu'ils peuvent assumer. L'objectif était à la fois de définir une taxonomie pour aider à situer de nouveaux modèles, ainsi que pour identifier des modèles existant pouvant servir de base à un modèle de diagnostic. Nous avons identifié LAMBDA/CRIM comme particulièrement intéressant pour notre étude.

LAMBDA [36] est un langage permettant de décrire des attaques. CRIM [35, 34], quant à lui, est un moteur de corrélation utilisant des modèles LAMBDA pour en obtenir des scénarios d'attaque complet. LAMBDA/CRIM fait partie des modèles de dépendances : ce sont des modèles qui expriment les dépendances entre les évènements en décrivant les conditions nécessaire pour qu'ils puissent survenir ainsi que leurs conséquences lorsqu'ils sont réalisés. Un intérêt de LAMBDA/CRIM est qu'il permet de décrire des évènements dits atomiques, c'est à dire qui ne peuvent être raffinés en deux évènements, aisément descriptibles par des experts en sécurité, puis laisser le moteur de corrélation reconstruire les scénarios en exploitant les conditions et conséquences décrites en LAMBDA. Travailler sur des évènements atomiques permet de s'affranchir de la difficulté de considérer à la fois sûreté et sécurité : les experts de sûreté et de sécurité peuvent indépendamment modéliser leurs évènements respectifs et laissent le moteur de corrélation réaliser l'intégration. Il a été nécessaire d'adapter LAMBDA pour y intégrer des évènements de sécurité mais le principe général décrit dans ce paragraphe reste le même.

LAMBDA/CRIM ne permettent pas d'obtenir des métriques importantes pour le diagnostic telles que le temps moyen avant la réalisation d'un évènement ou sa probabilité de réalisation après un certain temps. Pour cela, il a été nécessaire d'ajouter une couche probabiliste à LAMBDA/CRIM, ce qui a aboutit à la création de PROS^{2}E. PROS^{2}E associe à chaque évène-

ment une variable aléatoire représentant le temps nécessaire pour que l'évènement ait lieu. En utilisant des formules probabilistes décrites et prouvées dans cette thèse, il devient possible d'obtenir, à partir des variables aléatoires de chaque évènement atomique, celle associée au temps de réalisation d'un scénario complet. Cela permet donc de calculer toutes les métriques temporelles ou probabilistes nécessaires à la réalisation du diagnostic. PROS^{2}E est ensuite amélioré en lui rajoutant la capacité de considérer des contremesures, puis en améliorant ses capacités de modélisation temporelles et séquentielles. Ces capacités sont illustrées sur des scénarios de diagnostic tout au long de la thèse.

En plus d'avoir produit un modèle d'évènements novateur proposant une méthode innovante pour prendre en compte les probabilités temporelles, cette thèse a abouti à la publication de deux articles de conférences sans compter un autre article de conférence et un article de journal en soumission.

CONTENTS

1 Introduction **15**

1.1 The new paradigm of the Industry 4.0 . . . 16
1.2 Redefining the relationship between safety and security . . . 17
1.2.1 What are safety and security? . . . 17
1.2.2 The intricacy of interactions . . . 17
1.2.3 The difficulty of mixing safety and security . . . 18
1.3 Different questions for diagnosis . . . 19
1.3.1 The different types of diagnosis . . . 19
1.4 Contributions . . . 24
1.5 Organisation of the thesis . . . 24

2 State of the Art **27**

2.1 Chapter content . . . 28
2.2 Safety models . . . 28
2.2.1 Fault tree based models . . . 30
2.2.2 Bayesian Networks . . . 34
2.2.3 Petri nets . . . 37
2.2.4 Safety models summary . . . 39
2.3 Security models . . . 39
2.3.1 Attack trees . . . 41
2.3.2 Other attack graph based approaches . . . 46
2.3.3 Bayesian Networks For Security . . . 47
2.3.4 Dependencies models . . . 48
2.3.5 Petri nets for security . . . 50
2.3.6 Security models summary . . . 51
2.4 Hybrid models . . . 51
2.4.1 Boolean-logic Driven Markov Processes . . . 53
2.4.2 Integration of fault trees and attack trees . . . 55
2.4.3 Hybrid models summary . . . 56
2.5 Classifying models based on their contributions to diagnosis . . . 57
2.5.1 Description of categories . . . 57
2.5.2 Classification of studied models . . . 59

3 PROS^{2}E: A new probabilistic representation of events **61**
3.1 Introduction . . . 62
3.2 Preliminaries: logical dependencies . . . 62
3.2.1 Conditions and consequences . . . 63
3.2.2 Semi-explicit correlation to obtain scenarios . . . 63
3.2.3 Raising alerts for an on-line use . . . 63
3.2.4 Advantages of this approach . . . 64
3.3 Preliminaries: Probabilistic theory . . . 65
3.3.1 Atomically modelling the time . . . 65
3.3.2 Propagating the probability . . . 66
3.3.3 Probabilistic Equivalences . . . 68
3.3.4 Obtaining relevant values for the distributions . . . 68
3.3.5 Recap . . . 69
3.4 PROS^{2}E . . . 70
3.4.1 The event model . . . 70
3.4.2 Obtainable information . . . 71
3.5 An example on how to use PROS^{2}E . . . 72
3.5.1 Taum Sauk Hydroelectric Power Station . . . 72
3.5.2 The scenario . . . 72
3.6 Conclusion . . . 80

4 Handling countermeasures **81**
4.1 Introducing countermeasures . . . 82
4.1.1 Defining countermeasures . . . 82
4.1.2 What makes countermeasures different from regular events? . . . 82
4.1.3 Countermeasures in the dependency model . . . 83
4.1.4 Probabilistic representations of countermeasures in other models . . . 83
4.2 Motivating example . . . 84
4.3 Probabilistic modelling of countermeasures . . . 86
4.3.1 Computing probabilities with countermeasures, splitting the cases . . . 86
4.3.2 Simplifying the situations . . . 86
4.3.3 Probabilistic expression . . . 87
4.4 Application to the study case . . . 89
4.4.1 Modelling events and obtaining the graph . . . 89
4.4.2 Computing metrics . . . 90
4.5 Conclusion . . . 92

5 Improving PROS^{2}E with enhanced and accurate representation of the time 93
5.1 Motivating example . . . 94
5.1.1 First diagnosis example: processing new alerts . . . 95
5.1.2 Second diagnosis example: Purely accidental? . . . 96
5.2 Why PROS^{2}E needs improvement . . . 96
5.2.1 Needing more sequentiality . . . 96
5.2.2 Knowing when time is ticking or not . . . 97
5.3 Modelling sequences . . . 98
5.3.1 Correlation with a SEQ-AND . . . 99
5.3.2 Probabilistic expression of the SEQ-AND . . . 99
5.3.3 Nature of events: not just safety and security . . . 102
5.4 Managing the wear . . . 102
5.5 Application to the study case . . . 104
5.5.1 Modelling events and obtaining the graph . . . 104
5.5.2 First diagnosis example: processing new alerts . . . 104
5.5.3 Second diagnosis example: Purely accidental? . . . 105
5.6 Conclusion . . . 106

6 Conclusion 109
6.1 Contributions of this thesis . . . 110
6.2 Perspectives . . . 111

LIST OF FIGURES

2.1 An example of a fault tree . . . 31
2.2 A BN with the corresponding conditional probability tables . . . 34
2.3 Temporal dependencies in a DBN . . . 36
2.4 An example of a Petri net . . . 38
2.5 An example of an attack tree . . . 42
2.6 Outline of an LAMBDA event . . . 49
2.7 An example of a BDMP . . . 54

3.1 Events A and B happening in sequence and their equivalence C . . . 66
3.2 An AND gate and its equivalence . . . 67
3.3 An OR gate and its equivalence . . . 67
3.4 Situation output by CRIM . . . 68
3.5 Equivalence of Figure 3.4 . . . 68
3.6 Situation output by CRIM . . . 68
3.7 Equivalence of Figure 3.6 . . . 68
3.8 Hard-drive failure . . . 71
3.9 Scenario output by CRIM . . . 71
3.10 Access to the Operator Control Network . . . 73
3.11 Compromising of the communication link . . . 73
3.12 Scenario output by CRIM . . . 75
3.13 Evolution of the probabilities when no event has occurred . . . 76
3.14 First case: accidental . . . 76
3.15 table . . . 77
3.16 Evolution of the probabilities in case 6 . . . 77
3.17 Second case: attack . . . 78
3.18 Third case: mixed . . . 79

4.1 Fischertechnik-based CPS testbed . . . 84
4.2 Fischertechnik-based CPS testbed network architecture . . . 85
4.3 Countermeasure situation . . . 87
4.4 Countermeasure situation and equivalences . . . 88
4.5 Scenario considered in this section . . . 89
4.6 Model of the countermeasure of the scenario . . . 90

4.7 Scenario considered in this section 91

5.1 Milling operation in the Fischertechnik-based CPS testbed 94
5.2 First incorrect model to incorrectly place a part 97
5.3 First incorrect model to incorrectly place a part 97
5.4 Timeline of the functioning of the conveyor belt 97
5.5 Model of the conveyor belt failure 98
5.6 Recombination example 100
5.7 g 101
5.8 $\dot{g}$ 101
5.9 Model of the conveyor belt failure 104
5.10 Scenario considered in this section 105
5.11 Aftermath of the attack 107

LIST OF TABLES

1.1 Diagnosis scope in safety oriented articles 21

2.1 Summary of information available in safety models for diagnosis 40
2.2 Summary of information available in safety models for diagnosis 51
2.3 Summary of information available in safety models for diagnosis 57
2.4 Capabilities of the models 59

3.1 Events in the graph 74

4.1 Events in the graph 90
4.2 Success probabilities after one year 91
4.3 Success probabilities after one year 92

5.1 Events in the graph 106

CHAPTER 1

INTRODUCTION

The work presented in this thesis addresses the problem of diagnosis in industrial systems. We describe a framework to represent incidents that can happen in complex systems, how they influence each other to lead to undesired events, and adjoin it with a probabilistic model. This allows us to obtain various metrics that we use to provide diagnosis of ongoing incidents. We illustrate this by showcasing the use of our framework on various examples and use cases.

1.1 The new paradigm of the Industry 4.0

The Industry 4.0 paradigm can have many names such as Industry of the Future, Smart Factories or Industry 4.0 (as in the Fourth Industrial Revolution) but it corresponds to the same principle: *the computerization of manufacturing*. Industry 4.0 leads to increasingly automated industrial processes through ever more data exchange and the integration of new types of technologies such as the cloud computing or the internet of things. This results in the generalization of complex *cyber-physical systems* to control and monitor manufacturing processes.

This shift in the way industrial systems are designed and handled brings several benefits but is also accompanied with new issues that need to be addressed. Concerning security, specifically, the changes are substantial.

Industrial systems were initially designed to be relatively sealed from the outside world. One could say that the security was ensured by building a fence around the factory and hiring a guard at the gate. This is overstating the point but not too far off the truth though. Indeed, a physical access to the industrial system was once necessary for an attacker to provoque any perturbation to it. Industrial equipment such as Programmable Logical Controllers (PLC), sensors or actuators were developed with the assumption that this physical protection was deployed. They therefore did not need to embed any form of security.

Cyber-physical systems, however, bypass the need to be physically present in the system to be able to influence it. An attacker can use the same channels as the control system to access and attack the industrial system. Cyber attacks on industrial systems are a reality and have already made the news in several occasions such as the STUXNET attack on the Iranian nuclear program in 2010 [27, 69, 1] or the attack on the Ukrainians power plants in 2015 [68, 28].

This doctorate has been realised as part of the chair Cyber-CNI, Cybersecurity of Critical National Infrastructure, that aims at enhancing the security of critical systems. This entails multifarious research subjects such as resilience [29, 30], system architecture [108, 53], defence strategy [54], intrusion detection [109, 107, 106, 105, 110], alert reporting and visualisation [61, 60], or the evaluation of the impact of attacks [49], as well as the human and psychological aspect of security [84, 102, 104, 103]. Our study focuses on diagnosis of safety and security related events. In other words, when incidents occur in an industrial system, these incidents can be of accidental or malicious nature, and we aim at providing precise and relevant information

in order, for a decision maker, to select the most appropriate reaction. Due to the nature of the Industry 4.0, and the changes in the environment set by cyber-physical systems, diagnosing incidents raises new questions, mostly because of the interactions between safety and security that are much more important than before. The next section is focused on these interactions between safety and security.

1.2 Redefining the relationship between safety and security

Depending on the field of application, safety and security can have various perimeters. Before discussion about their interactions, it is necessary to define the terms for our study. In that purpose, we use the SEMA (System-Environment Malicious-Accidental) framework defined by Pièrre-Cambacédès and Bouissou and Bouissou [95]. It is a compilation of several definitions and scopes of safety and security used in various environments.

1.2.1 What are safety and security?

In this thesis, we use the *Malicious vs. Accidental* definition: "Security typically addresses malicious risks while safety addresses purely accidental risks". This is a distinction about the origin of an incident. Indeed, we are interested in achieving a better comprehension of an incident when it is initially unclear whether its nature is accidental or malicious. This is the relevant definition for our study since the *accidental* side has been studied for decades when the *malicious* is fairly recent for industrial systems and is a consequence of Industry 4.0.

With these definitions being given, we can move on to the next point of the discussion and see how safety and security interact and create non-trivial situations.

1.2.2 The intricacy of interactions

According to Pièrre-Cambacédès and Bouissou [95], safety and security have four types of interdependencies: conditional dependencies, reinforcement, antagonism and independence [94]. Safety and security are conditionally dependent when one is a requirement for the other. For example, ensuring that an attacker cannot blow up a pipeline is part of ensuring the safety of the pipeline. Reinforcement occurs when both are strengthened by the same process. This is the case when isolating an industrial system in a dedicated network for security reason will also increase its bandwidth for more interactions with the supervision. Antagonism happens when an increase in one causes a decrease in the other. This occurs when providing a remote access to an industrial system's supervision increases the attack surface of an attacker. Independence is when a modification of one has no impact on the other. Using https instead of http on the embedded server of a controller yields a better security without impacting the

safety. Understanding an ongoing incident, and deploying the most appropriate response, is a non-trivial problem, particularly in case of antagonism.

A simple example of how security and safety can conflict with each other is the problem of the fire door. If a fire occurs in a building, the fire doors should automatically unlock and open to ensure people's safety by allowing them to exit the building. However, a terrorist could take advantage of this safety response by triggering the alarm of the building for the doors to unlock, allowing him to get inside. In this situation, safety and security supervision are both capable of handling the situation but with opposite response that will jeopardize the safety of the persons in the building if wrongfully deployed. The objective in this situation would be to provide information that will allow for the most sensible and appropriate response to an incident involving potentially both safety and security. In the case of the fire door, a question diagnosis should be providing an answer to could be "is the alarm legit or is it triggered by an attacker?".

This thesis focuses on providing an answer to such questions since they are part of the process we call diagnosis. This diagnosis is obviously to be based on a model that captures the incidents regardless of their nature. This is actually a major issue and the subject of the next paragraphs.

1.2.3 The difficulty of mixing safety and security

Historically, safety and security models are monolithic: they only consider safety or security but not both. This is another by-product of a separation by design between safety and security discussed in the previous paragraphs. Due to interactions between safety and security, it is obviously necessary to use models that can consider them both.

Blending safety with security is not a trivial problem. Their objectives are different and the techniques developed to achieve them exploit some of specific aspects that they do not necessarily share. In safety, for example, the failure rate of a component is generally a well known value, obtained experimentally from the failure of other components. Whereas in security, estimating the time necessary for an attack step is a difficult problem and the consensus is that it has not yet been addressed. On the contrary, in security, probabilities are more used in terms of choice that an attacker can make between multiple options than on the time taken to perform them. This raises another point where in security, events happen because a person chose to cause them, while in safety, a component failure is generally accidental.

Dedicated models will capture these specific behaviours that allow them to achieve their objective but make almost impossible any generalisation. Gibaudo et al. [50] also cite the fact that the order of occurrence of events is fundamental in security when it can be mostly ignored in safety, as well as fault models being structurally similar to the system and therefore not much subject to change when attack model will need important revisions after even minor modifications.

The literature has sometimes tried to use combinations of a safety model with a security model [47, 85]. Such approaches are possible with narrowing the scope of interactions safety and security can have [47], or using very abstract models with few computing possibilities [85]. Describing a model generic enough to capture both safety and security but enabling enough extraction of information is a challenge. In this thesis, we discourse on how it can be done for diagnosis.

The second chapter of this thesis is dedicated to the study of models that consider safety and/or security in order to provide information for diagnosis. But for now, we still have to precisely define what diagnosis in a safety and security environment means.

1.3 Different questions for diagnosis

So far, we have several times used the term diagnosis in a broad sense. However and much like safety and security, it can have many definitions depending on the environment or context of application. In this section, we discuss about existing definitions and we give our own.

1.3.1 The different types of diagnosis

Studying safety diagnosis models and security diagnosis models was a necessary preliminary step in understanding the whys and wherefores of what can be considered diagnosis in the literature. And it appeared that these definitions are not necessary relevant for a safety-security context. Therefore, it is beyond the scope of our study to provide an in-depth review of the state of the art of these models since they are, in retrospect, unfit for the problem we want to address. However, we mention them in this section as we discuss about the various definitions of diagnosis and provide a suitable one for a mixed safety-security perspective.

Diagnosing incidents in an environment where both accidents and attacks are considered is very different from an environment where only accidents or only attacks can happen. Indeed, when dealing with both safety an security, one will require information such as the accidental or malicious nature of the incident, its causes, the consequences and the nature of the consequences, etc. Such information is not needed when performing legacy diagnosis of purely accidental faults or malicious attacks, or has a different meaning.

Techniques dedicated to purely accidental or malicious incidents achieve their objective but cannot provide the information we believe necessary for a mixed-nature diagnosis. This type of diagnosis needs a different perspective to perform an analysis that requires different information. Instead of articulating our study around existing diagnosis models that are unfit for the considered problem, we have decided to focus on means to obtain this information and how it will enable diagnosis of accidental and malicious incidents. In the upcoming paragraphs,

we discuss about the definitions accepted for the different types of diagnosis and provide a justification for the definition we retain.

Diagnosing accidental faults

In the field of safety, diagnosis is typically considered as the defined of three tasks: fault detection, fault isolation and fault identification [90, 119]. Fault detection is the action of discovering that the system has switched from a normal state to an abnormal one. Fault isolation is the process of finding which components are at fault. Fault identification is the discovery of the nature of the fault.

However, diagnosis is sometimes defined as simply fault isolation and fault identification [26, 76]. In that case, fault detection constitutes a singular process out of the scope of the diagnosis. In other cases, diagnosis is going to be fault detection and fault isolation, fault identification not being mentioned [5, 48]. Table 1.1 summarises various scopes of what is called diagnosis in several works on the subject.

Nonetheless, two families of approaches are identified: logic-based and behavioural [119]. The logic-based approach detects faults by comparing the inputs of a component to its outputs. The behavioural basically monitors the evolution of physical measures in the system and detects faults based on the deviation of these measures from a normal state. Overall, safety diagnosis is about determining which components are at fault. And it makes sense since, in an industrial system, it is safe to assume that the propagation of an incident is deterministic when the original faults are known. However, when security is also a parameter, new situations occur. First, all of the steps of a security attack do not necessarily correspond to the exploitation of a vulnerability, the security counterpart of a fault. For example, an attacker could take advantage of an auto-update mechanism to push a back-door in the new version of a software. The update in itself is not a vulnerability, but it becomes part of an attack scenario. With such neutral events, the semantics of the model is then violated in a way that may render its result questionable. Of course, this would require further investigation on a case-by-case basis. Yet, even if the semantics was adaptable enough to provide a meaningful result, another issue arises.

The information sought for a diagnosis with security involved is very different from a purely safety one. An example of that is the nature of the incident. If safety alone is considered, an incident can obviously only be accidental and there is therefore no need for a means to evaluate its nature. With security involved, an incident could be identified as a fault but be actually caused by an attack, thus needing to trigger a different response from the system. For instance, a component could be breaking out of natural decay, or because its wear was sped up by an attacker. If it is the later, replacing the component without preventing the attacker from doing it again does not represent a perennial solution. Safety-oriented diagnosis models do not provide any means to evaluate this because it is not relevant in the context they were designed for.

References	Detection	Isolation	Identification
[5, 48]	x	x	
[26, 76]		x	x
[90, 119]	x	x	x

Table 1.1: Diagnosis scope in safety oriented articles

Diagnosing malicious attacks

When studying the security of a system, diagnosis is sometimes understood as intrusion detection. Jackson *et al.* [55] proposed an intrusion diagnosis model able to detect an ongoing attack as well as classifying it, based on some predefined symptoms. Diagnosis is then an on-line process that happens at the same time of the attack.

For Elsaesser and Tanner [44], "diagnosis is the process of deducing the most likely mechanism that caused an observed condition". In their article, oriented towards computer forensics, they also provide a five step approach to achieving diagnosis: observe abnormality, generate possible explanations, eliminate impossible explanations, eliminate implausible explanations, and then rate the remainder. They consider diagnosis as being an off-line investigation operation.

For Arshad *et al.* [9], diagnosis "is traditionally defined as the process to investigate the cause of a successful intrusion". In their paper, though, it is defined as "the process of evaluating the severity of an intrusion for a monitored host".

In these definitions, a diagnosis model may have to raise alerts corresponding to an attack but are mostly designed to identify its causes, or its consequences. This is a major difference with safety diagnosis models which focused on identifying which components are at fault but not the cause of these faults.

Safety-security diagnosis

Before giving a definition of diagnosis in a safety-security environment, let us first develop the differences between the objectives of the different types of diagnosis. Piètre-Cambacédès and Bouissou have identified four types of interactions between safety and security [94]. The intricacy of such interactions are what makes diagnosis in a safety-security environment different.

Let us illustrate this on the following scenario, inspired from the Stuxnet attack. We consider a programmable logic controller (PLC) connected to a SCADA server. The PLC controls, amongst other, a centrifuge. An attacker has a remote access to the SCADA server and is able to send orders to the centrifuge, as well as conceal these orders from the supervision. The attacker wants to destroy the centrifuge and does so by rapidly varying its rotation speed. Upon destruction of the centrifuge, the industrial supervision will call for its replacement, not aware of

the presence of the attacker since safety diagnosis does not acknowledge for such situations. However, the attacker still being in the SCADA server, replacing the centrifuge is hardly a solution since he will manage to break it again. Let us now consider the security supervision, which has detected the presence of a rogue agent in the SCADA system. Security diagnosis has also identified what vulnerability has been exploited and even that a patch exists to solve this issue. However, deploying a patch requires qualification, scheduling and disruption of ongoing operation. In the mean time, from the security point of view, the most sensitive thing to do is to switch off the SCADA server. This is of course an unacceptable solution from the point of view of the industrial process. This situation illustrates that safety and security have to be supervised as a whole, and cannot simply rely on legacy diagnosis. What we expect from diagnosis in this situation is, first, to correlate the attacker present in the system with the failure of the centrifuge, but also to be able to consider potential solutions and ponder their consequences.

To summarize, we can say that legacy diagnosis is about raising alerts when safety-security diagnosis has to got much further. However, it will rely on alerts raised thanks to this legacy diagnosis. Safety-security diagnosis' objective is not to redo what is done by these existing models because it is not about raising alerts. On the contrary, it is about using these alerts to understand an ongoing situation and conciliate safety and security in providing explanation and identifying solutions.

Elsaesser and Tanner's definition [44] is close to what diagnosis in a safety-security environment should be since it is about identifying how an incident has occurred and has spread to the system. However, we also believe diagnosis should go further and also evaluate the risk associated with the ongoing incidents. Indeed, safety and security responses for the same incident can enter into conflict with each other, or could impact one another. Knowing the outcome that carries the higher overall risk for the system and not simply from a security or a safety point of view will help choosing the most appropriate response. Therefore, we adapt this definition for the identified needs:

Definition 1 *Diagnosis in a safety-security context is the process of deducing the most likely mechanism that caused an observed condition, and identifying the future outcome*

As such, and unlike Elsaesser and Tanner, we consider this type of diagnosis to be an online process. From now and until the end of this thesis, unless explicitly mentioned, the term "diagnosis" will have this definition.

To perform its intended use, a diagnosis model needs to realise several functions:

- *Incident detection*. This is the preliminary step to any diagnosis operation. In practice, incident detection is mostly an input and not performed by the diagnosis model since it is efficiently performed by legacy diagnosis tools.

- *Incident correlation*. In an industrial system, when a component fails, it will most likely affect other sub-systems that will generate incidents as well. An attack is also generally executed in several steps, each of them susceptible to generate alerts. Moreover, faults and attacks can have effect on each others. Therefore, a diagnosis model is expected to understand if several incidents are related or correspond to independent situations.

- *Explanation*. A diagnosis model aims at generating hypothesis on the origin and the outcome of an incident. A scenario is a composition of incidents that lead to an undesired event and a diagnosis model should provide possible scenarios corresponding to observations.

- *Risk evaluation*. A supervisor of a system will not necessarily rely on the most probable outcome when selecting an appropriate response. Indeed, it is sometimes the one provoking the highest impact that will be remedied to. Thus, it is necessary to evaluate the probability and the risk associated with identified explanations.

- *Likelihood evaluation*. When a very unlikely incident happens, it is generally the sign that it was wrongfully identified. Whether it be a false positive or mistaken for something else, it is the sign that further analysis is required. When dealing with a security model whose perimeter can, therefore, not realistically consider every possible incident, it is a very important feature.

- *Countermeasures*. In reaction to an incident, the system will deploy countermeasures. These responses will remedy to the incident, meaning that they will cancel its effects, or at least mitigate it, transitioning the system back to a functioning state different from before the incident. However, they may have side-effects on the system such as creating new opportunities for an attacker or disrupting the industrial process. Moreover, even without side-effects, deploying these countermeasures will change the state of the system and are needed to be registered when performing live analysis. A diagnosis model needs to be able to consider countermeasures into their scenarios.

- *On-line*. A diagnosis model is used to identify the causes and consequences of an incident, and will therefore be used to identify the most appropriate response to an incident. It is therefore necessary that relevant and intelligible information is provided in time for the most appropriate countermeasure to be deployed.

In the next chapter, we analyse models that are able to provide information to realise such functions. Even though they are not always explicitly designed to perform diagnosis, they could be used, to some extent, for it. We are interested in the metrics they are able to compute and how these metrics represent valuable information towards diagnosis.

Chapter 2 focuses on models used respectively for safety, security, and both. We provide an overview of relevant models and metrics. We then propose a classification of these models, and compare them to what we consider a diagnosis model could have, based on what exists in the literature.

1.4 Contributions

This thesis presents two major contributions: a discussion on the ins and outs of diagnosis in a safety and security environment, and a model to perform this diagnosis.

The first contribution leads to a definition of diagnosis, and the identification of several functions desirable in a diagnosis model. It is accompanied with a review of existing models that can fulfil these functions by studying them under the angle of the metrics and information they are able to provide. These models are not diagnosis models *per se* but could be used to partially perform diagnosis, or could be used as the fundamentals for a more elaborate and comprehensive diagnosis model.

The second contribution is PROS^{2}E, an event model used to several metrics and information to diagnosis incidents in complex systems. This contribution was incremental, with a first version that laid out the mathematical foundations upon which more elaborate iterations were built. These subsequent contributions to PROS^{2}E allowed to model more situations more accurately, enhancing the diagnosis capacities of the model.

1.5 Organisation of the thesis

The remainder of the thesis is divided in five chapters.

Chapter 2 is a state of the art in terms of obtaining the relevant metrics for diagnosis. As we discussed in the previous paragraphs, there exists no diagnosis model for safety and security. However, we have identified the information necessary to perform the diagnosis and this chapter is a review of existing models that provide these metrics.

Chapter 3 presents PROS^{2}E, an model used to represent the events that can occur in a system, keep track of there occurrence in real-time and provide relevant metrics for diagnosis on the fly.

Chapter 4 focuses on the integration of countermeasures in PROS^{2}E. Countermeasures are particular events that prevent a scenario from happening. As such, they require a reflection on how they impact the various metrics computed by the model, an consequently a specific probabilistic modelling.

Chapter 5 improves the modelling and diagnosis capabilities of PROS^{2}E by adding the Sequential-AND gate and reworking the time processing. This leads to more complex system

that can be modelled, and more accurate metrics that can be computed for previously existing situations.

Chapter 6 concludes the work presented in this thesis by summarizing it, and by presenting perspectives for future works regarding PROS^{2}E and diagnosis in cyber-physical systems in general.

CHAPTER 2

STATE OF THE ART

2.1 Chapter content

Since diagnosis for an environment where both safety and security are to be considered is a mostly unexplored field, it is not surprising that no model have been developed for it. In Chapter 1, we have however identified several functions that such a model should realise. These functions can be realised, even partially, by existing models that are the subject of the this chapter.

Due to the different nature of safety and security events, we have organised this chapter in three sections: safety models, security models, and hybrid models. Because these models are not diagnosis models per se, they are not studied under the scope of how they can perform diagnosis but what they can provide for diagnosis. We discuss about various metrics or information that these models can provide and that can be valuable when establishing the diagnosis of an incident.

We do not simply focus on the metrics but also more generally on which functions they can realise, and summarize this in a classification of the models.

2.2 Safety models

In this section, we study safety models and the information they are able to provide for diagnosis. As a preamble, we illustrate what sort of information we look for by presenting some simple scenarios and raising some questions that diagnosis should provide an answer to.

First, let us consider a cream enrichment plant designed to produce cream from milk. The separation of cream from the milk is realised by two centrifuges in series. When a centrifuge encounters a problem, it is possible to increase the rotation speed of the other centrifuge to compensate but it also speeds up its deterioration. The occurrence of such incident raises several questions that diagnosis can address.

A question one might ask is, if the functioning centrifuge increases its rotation speed to compensate for the failure of the other one, how long until it wears out? To answer this question, it is necessary to have a measure of the time until the next failure in the model. It is usually expressed as a *Mean Time To Failure* (MTTF). However, for this specific question, the model needs to handle the fact that this MTTF depends on the rotation speed of the centrifuge, or more generally on the state of the system. A sister metrics to the MTTF is the *reliability* of the system. It is defined as the probability that the system works for a given period. Similar to reliability, but particularly useful for repairable systems, the *availability* is the probability that the system works at a given time. Repairable systems can also gain from knowing their Mean Time Between Failures (MTBF). Since the repairing of the failed centrifuge will take some time, they are important information to balance risk and benefits between deciding to stop the system or

increase the rotation speed.

Another information one might want to know is how long until the system is fully functional again. This is typically computed by a Mean Time To Repair (MTTR), which expresses the average time it takes to address a specific failure. In our scenario, it will correspond to the time taken to repair the broken centrifuge or to increase the rotation speed of the remaining one.

Sometimes, repairing a component is not sufficient to restore the system to a normal state. We now consider another scenario of an incinerator supplied in combustive, for example dioxygen, by a bellows. In an incinerator, the energy produced by the combustion of already present waste is sufficient to ignite additional waste in a sort of self-sustained reaction. However, like any combustion, it needs a supply of combustive. If the bellows providing the combustive fails, the combustion stops. Replacing the bellows will not be sufficient to restore the system to a functioning state because the combustion will need to be restarted. In this scenario, having the MTTR and the repair cost of the pump is not sufficient to have the one of the system. Indeed, kick-starting the reaction again will have to be modelled independently, most likely by a dedicated countermeasure event, and will induce an additional cost, therefore impact and risk, on the system.

Let us now consider a water plant that drills water from a phreatic table. The water plant relies on seven drills that can be switched on or off by a controller. After being drilled, the water is filtered, exposed to UV lamps, and mixed with chlorine in a processing basin. Water in then transferred from this basin to a water reservoir with six pumps that can be individually switched on or off by a controller. In this process, the failure of the system could require the failure of several components at the same time. *Minimal cut sets* are minimal sets of events causing a system failure. They are useful because they point out critical components of the system that need to be closely monitored, as well as reduce the complexity of required computation since it becomes possible to focus on smaller sets of events. *Minimal path sets* represent minimal sets of components that will keep the system running as long as they do not fail, providing emphasis on critical components, as well as reducing computation complexity if demanded.

The seven drills and six pumps of the water plant are used to mitigate water production depending on the demand. They also act as a convenient redundancy mechanism when a pump or drill would malfunction. However, all of the drills and pumps are of the same model, and therefore suffer from the same weaknesses: the failure of one component could mean a higher risk for remaining ones. Such behaviour is called *Common Cause Failures* (CCF) and is extremely important to spot in order not to overestimate the reliability of the system. Spotting CCF allows to correlate events as well as update the failure probability of remaining components.

Finally, when providing a possible explanation or a possible outcome, the ability for a model to give a confidence interval of its result is greatly appreciated. Indeed, on the one hand, the

return on experience is not always the same for each component of the system. On the other hand, depending on the position of an event in the scenario, a small variation of the probability of occurrence of this event could have either little or important consequences on the MTTF, reliability or other probabilistic computations. This consequence that a variation of a random variable associated with an event has on the probability of the scenario is called the *sensitivity* of the random variable, or of the associated event.

Having introduced important metrics that are sought in for diagnosing purely safety situations, we will now see if such information is present in existing models.

2.2.1 Fault tree based models

Fault Tree analysis (FT) is one of the most widely used techniques to compute risks concerning industrial systems. They were developed as a means to evaluate the launch system of the Minuteman Missile in 1962. After almost two decades of development for avionics and nuclear plants, the Fault Tree Handbook [120] was published in 1981, still being a well respected reference on the subject. Fault trees are widely used for probability risks assessments in industrial systems. Several different variations, which we discuss about in this section, have then been developed to suit specific needs. However, they all share a common origin that is a mature and reliable model to analyse risks.

In this section, we give the general principles of fault trees and fault tree analysis, as well as some close variations, and the metrics they are able to produce that are potentially interesting for diagnosis.

Fault tree structure

A fault tree has a tree structure of which the root is the undesired event being investigated, leaves are safety events, branches model failure propagation, and nodes are logical gates conditioning this propagation. Figure 2.1 is an example of a fault tree.

To create a fault tree, one can use five types of events. First, there are the *basic events* that correspond to a basic initiating fault. Then there are *intermediate events* that verbosely describe the combination of its child events but have no effect on the analysis. There are also *undeveloped events* whose development into smaller events is not relevant for the study or impossible due to insufficient information. The last two types of events are *transfer ins* and *transfer outs* used to cut the tree into several trees when the FT becomes too large.

The events are then recombined into scenarios using four types of logical gates. There is the *and gate* describing a conjunction of events, the *or gate* describing a disjunction of events, the k/N *gate* that is activated when k out of its N input are active, and the *inhibit gate* whose output can only be active if a condition is verified.

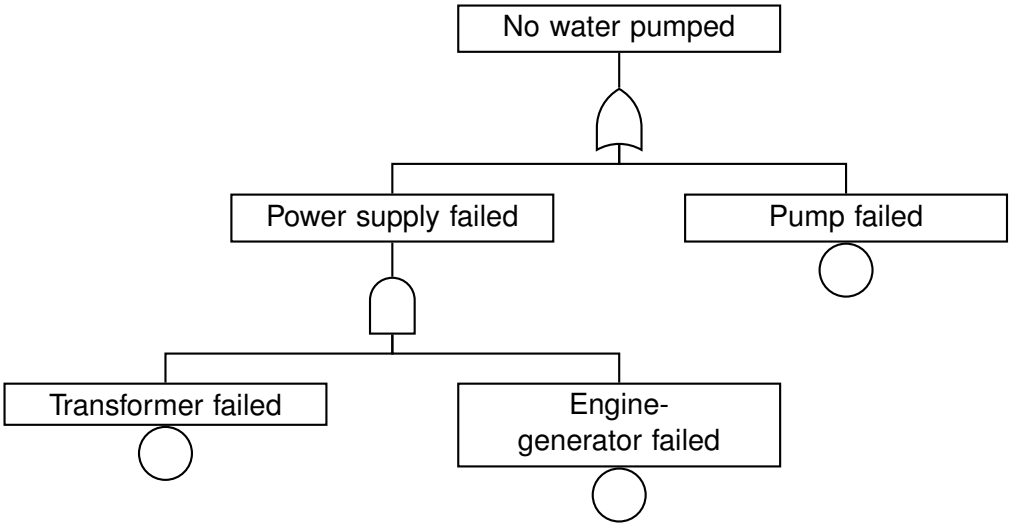


Figure 2.1: An example of a fault tree

Other gates and event types are defined and used by extensions of FT to adapt to specific situations [41, 71, 14].

Information available

Fault trees usually perform risk analysis of a system. They are built by system experts that investigate possible failure of components and how these failures can cause damage to the system, damage represented by the top event of the tree. Once built, they provide analysts with valuable information for risk analysis.

First, from the structure of the tree, one can extract several information. Minimal cut sets are available [32, 124, 24]. Ali [2] showcases the use of minimal path sets to evaluate the reliability of a system. CCF, however, cannot be analysed by fault trees. They can nevertheless be modelled using OR gates, but rely on external information added by the system expert [111].

Fault trees are also able to provide so called "single time" metrics. In these situations, probabilities are associated with each basic event, describing the confidence that the components will fail during a predetermined period of time. They allow for computation of a *failure propagation*: the failure rate of all basic event being known, the failure of each intermediate event can be computed using the ones of his children according to the type of gate that recombines them. By iterating this process on the whole tree, one can obtain the failure rate of the top event. Lastly, fault trees can be converted into Bayesian networks for further analysis [13]. The capacities of Bayesian networks for safety analysis is discussed in section 2.2.2.

Having probabilistic capabilities is, as stated in section 1.3.1, mandatory for diagnosis. Single time metrics are, however, quite limited. The failure rate is valid for a predefined amount of

time, which is not suited for an on-line use. However, there exists counterpart to these metrics that are function of the time. They are called continuous time metrics.

The goal of continuous time analysis is to assess the evolution of the safety of the system over time. To do so, a probability distribution function D_e representing the evolution of the failure rate over time is assigned to each event. For repairable components, a reparation distribution RD_e is also assigned. Then a random variable X_e, function of the time, which values are in $\{0, 1\}$, represents the state of the component : 1 if at fault and 0 else. Similar random variables X_g can also be computed for the gates from the random variables X_e. With that additional mathematical tool, several metrics can be computed such as reliability [120, 13, 42], MTTF [4], availability [42, 11] or MTBF [4, 113] if the model account for the repairability of the system. The *sensitivity* of a random variable, and therefore the basic event it is associated with, can also be computed.

With continuous time analysis, an on-demand estimation of the time until the failure of the system is available, that can be used to evaluate the risk undergone by the system. By assigning the probabilities of realised events to 1, as it corresponds to the certainty that a fault has happenned, a fault tree could adapt itself to an undergoing situation. Fault trees are also able to provide a probability of failure for any given period, which are, once again, useful for the risk. Lastly, having a probabilistic evaluation for each basic event, they are potentially able to evaluate the likelihood of occurrence of any failure, or combination of failures. This would make fault trees both able to quantify their confidence in the diagnosis provided, as well as raise suspicion on a system being improperly modelled. Indeed, an event occurring despite an extremely low probability might mean that it was wrongfully modelled or detected.

Limits

Fault Trees can only consider two-state components: working and not-working. However, it is sometimes relevant for a component to have more. For instance, a component can be in an overclocked state, in order to compensate for a failed component, in which case the failure rate is increased. Or at the contrary, a downgraded state when it has suffered some damage but not enough to render it completely inoperative.

Some common behaviour of industrial systems cannot be modelled by a standard fault tree. For example, in the system modelled by the fault tree represented on figure 2.1, the engine-generator is a backup in case the transformer fails. Therefore, it would not be started unless needed, and as such, its failure rate would depend on the state of the transformer. Although it can be acceptable for a purely logical representation, it leads to inadequate and misleading results when computing temporal metrics such as MTTF or reliability.

Moreover, there would most likely be a switch between the transformer and the engine-generator and the order in which the components fails would have a consequence on the out-

come. If the switch fails after the transformer, it is not an issue since the engine-generator was successfully started. However, if the switch fails before the transformer, the engine-generator cannot be started and the water cannot be pumped. Standard fault trees cannot acknowledge such elementary behaviour in which the temporal order of occurrence of the faults has an impact on the scenario. Dynamic fault trees were developed to address this issue.

Dynamic fault trees

As we've just seen, in some cases, a specific order in which the components fail is necessary for top event to be realised. Regular fault trees cannot take this behaviour into account. Dynamic fault trees were developed in 1990 to address this issue [41].

Dynamic Fault Trees (DFT), as defined by Dugan *et al.* [41] add three types of gates to regular fault trees: functional dependency, cold spares and priority-AND.

The functional dependency gate has one output, one trigger input and one or more regular inputs. The occurrence of the trigger event forces all of the input events to occur. An example of a functional dependency gate could be with components communicating through a network. The input events would be the ability for components to communicate on the network, and the trigger event would be the network failing. Indeed, if the network fails, regardless of the state of each component, they would not be able to communicate.

Cold spares are backup components that are initially unpowered and that are only activated upon failure of a functioning one. Cold spare gates are gates with one output, and several inputs. Each input corresponds either to the primary input, or an alternative input. Alternative inputs are ordered and are only allowed to fail if all of the previous ones have failed. The first alternative input can only fail if the primary input, the only one initially powered on, has failed.

Priority-AND is a regular AND gate with the additional condition that failure has to happen in a specific order. If the order of occurrence is not respected, the failures are ignored by the gate. This is particularly useful to model the situation we previously described with the transformer, the engine-generator and the switch.

Reliability [41, 15, 83, 31], availability [41, 15, 31], MTTF are computable in a DFT, and even MTTR and MTBF if the DFT is repairable [15]. Minimal cut sets do not capture sequential information but they can still be computed by swapping priority-AND gates and cold spare gates for AND gates and functional dependency gates for OR gates. To preserve this sequential information, Tang *et al.* [126] defined cut sequences, which are basically ordered cut sets, and means to compute them. Several other computation or representation methods were then developed [125, 73, 74, 82, 99, 25].

Analysis of a DFT can also be done by converting it into a Dynamic Bayesian Network [83, 97], a model we discuss in section 2.2.2.

2.2.2 Bayesian Networks

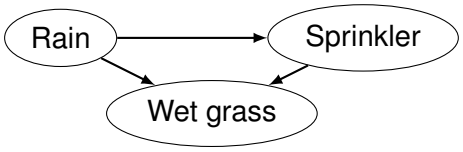
Although Bayesian networks (BN) functioning is based on Bayes' theorem (2.1), named after Thomas Bayes (1701-1761) and formulated by Pierre-Simon de Laplace in 1812, Bayesian networks were introduced by Judea Pearl as late as 1986 [89]. In 2010, Weber *et al.* [122] proposed a bibliographical overview of various uses of Bayesian networks in industrial systems. Their study showcases early proposals in the 90s and a huge increase of contributions in the years 2000s, especially after 2005. Bayesian networks can now be considered a mature method to evaluate different aspects of safety in industrial systems.

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)} \tag{2.1}$$

A Bayesian network is a graphical model representing random variables laid out in a directed acyclic graph. The graph (V, E) is composed of edges E and vertices V. Each vertex, or node, represents a random variable and each edge represents a dependency between the two random variables of the corresponding edges. Each node is also associated with a probability distribution function that takes the values of the parent nodes as an input, and outputs the probability of the variable of the considered edge. The joint probability of the network is effectively (2.2), with $pa(x)$ being the parents of variable x. Figure 2.2 displays a classical Bayesian network and the conditional probability tables of the random variables.

$$P(V) = \prod_{x \in V} P(x|pa(x)) \tag{2.2}$$

Rain	
T	F
0.3	0.7



	Sprinkler	
Rain	T	F
T	0.1	0.9
F	0.6	0.4

		Wet Grass	
Rain	Sprinkler	T	F
T	T	0.99	0.01
T	F	0.9	0.1
F	T	0.8	0.2
F	F	0.01	0.99

Figure 2.2: A BN with the corresponding conditional probability tables

Bayesian networks are useful for safety and security modelling because they can represent dependencies between faults and/or attacks in global scenarios. Under the hypothesis that

the marginal probability of each event can be known, it is then possible to obtain a model for a probabilistic evaluation of the safety and the security of a given system. Then, when a set of events occurs, this information can be transmitted to the corresponding nodes and then propagate in the system. In a nutshell, Bayesian networks are a means to obtain, spread and extract knowledge. For diagnosis, specifically, BN are particularly useful because they can compute the probability distribution of variables based on the observation of others. This means that they can provide an explanation for an observed phenomena.

Information available

According to Bobbio *et al.* [13], diagnosis with BN generally represents three types of inference: the computation of the posterior marginal probability distribution of each component, the computation of the posterior joint probability distribution of subset of components, and the computation of the posterior joint probability distribution on the set of all nodes, but the evidence ones. They all correspond to the propagation of the information after a phenomena has been observed, and the showcase of the most probable explanation. This is, for example, showcased in Mengshoel *et al.* [81]. They can therefore estimate the same reliability [13, 117, 16] of single-time fault tree analysis that presents limited interest for diagnosis compared to continuous time analysis, as discussed in section 2.2.1.

Unlike fault trees, BN can easily model dependencies between events such as common cause failures on mutually exclusive events, standby dependencies or components supporting loads where the occurrence of one event changes the probabilities of other events [117].

By adding decision and utility nodes, in which case the Bayesian network is called an influence network, *cost* can be evaluated, proportionally with the probability of the various events in the model [51]. Cost can be either damage cost, which can be considered as impact, or, for instance, the cost of installing a new equipment. Cost is represented by so called *chance nodes*, Therefore, the cost is not associated with each event but only with terminal ones. Moreover, the cost is not exactly seen as a numerical parameter but is represented by the label of the chance nodes. This hybrid perspective of cost being an event makes it questionable for providing a measure of impact for diagnosis. Similarly, Kang and Golay [62] propose to use BN to evaluate the impact that actions taken by operators, when responding to a fault, can have on the system.

Just like fault trees, BN are able to compute what combinations of events can lead to an undesired one. They are able to compute cut-sets [13] in that regard.

Limits

One of the major drawback of BN is their impossibility to model temporal dependencies. Therefore, BN cannot consider the order in which events have to happen. Moreover, a BN cannot

make a forecast of what is going to happen if the variables change over time. In an environment where some events can only happen if other events are realised and where time is an important parameter, this is a major issue.

Importing the time dependencies, such as the ones found in a DFT, in a BN have been done by Boudali *et al.* [16] by discretising the time of the study in several intervals and adding as many states in each node as there are intervals. They are able to produce similar results as a DFT at the expense of a considerable increase of complexity.

Several other models have been developed to address this issue, as we discussed in the following sections. Nonetheless, adding temporality to a BN is a tedious task that adds a lot of complexity to the model and therefore, is a constant trade-off between the accuracy of the model and its computability [8].

Building BN has been identified as being difficult for safety experts [18, 62] which is an obstacle towards a wider adoption. Automatic transformations from a more common model such as FT has however been considered [13].

Dynamic Bayesian Networks

Dynamic Bayesian networks (DBN), that can also be called two-timeslice Bayesian networks (2TBN), are an extension of regular BN designed to address the temporal issue [38]. A DBN is defined as a couple (B_1, B_{2d}). B_1 is a regular BN that describes the initial distribution of the variables. B_{2n} is a two-step DBN that describes the transition from time $t-1$ to time t; it contains $P(x_t|x_{t-1})$ for each variable x in the network. The joint probability of a time-step is then (2.3), only now, the parents of x_t can be in the time $t-1$. Figure 2.3 showcases the temporal dependencies in a DBN.

$$P(V_t|V_{t-1}) = \prod_{x \in V} P(x_t|pa(x_{t-1})) \tag{2.3}$$

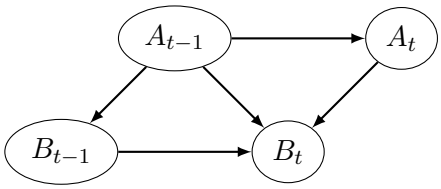


Figure 2.3: Temporal dependencies in a DBN

DBN are a solution to the temporal problem addressed in section 2.2.2. However, to our knowledge, they were never used to compute MTTF or similar metrics. The reason behind this is unsure, whether the model cannot provide such information or because the formalism is too complicated for a computer to provide results in resonable time.

Moreover, $P(x_t|x_{t-1})$ being the same for any t, the time, expressed in number of steps, taken for any event to be realised follows a binomial distribution. It is impossible to use a Dirac distribution to describe an event that occurs after a specific amount of time, or a Weibull distribution which is much more fit to model hard-drive failure [114].

Temporal Nodes Bayesian Networks

Temporal nodes Bayesian networks (TNBN) [8] were designed to address the lack of temporal capabilities of regular BN. In a BN, a node is associated with a random variable that represents the state of the node. In a TNBN, the random variable of the node represents the temporal interval in which a change of state occurs.

With this modification, they are able to predict and diagnose events and give an estimation of the time needed for these events to happen.

TNBN constitutes an interesting approach in addressing a major disadvantage BN suffers from compared to other models: time modelling. However, they do it by multiplying the number of values the random variables in the nodes can have, therefore rendering the model much more difficult to establish for a system expert, and increasing the computing time. We note that the establishment of the model aspect was eased by an algorithm based on machine learning techniques [52].

2.2.3 Petri nets

Petri newtorks were developed by Carl Adam Petri in his Ph. D. thesis in 1962. A Petri network (PN) is a directed bipartite graph with two types of nodes: places and transitions. Places are graphically displayed as circles and transitions as rectangles. When used in dependability analysis, places represent the states of the components and the transitions describe how these states can change. Places can be marked with tokens. Edges are associated with a multiplicity, a natural non null number. A transition is enabled if every place connected to input edges contain as many token as their respective multiplicity. An enabled transition can be fired, therefore removing as many token in input places as the respective multiplicity of input edges, and add as many token in output places as the respective multiplicity of output edges. The marking of a Petri net, *i.e.* a distribution of tokens amongst places, describes one state of the modelled system. Graph 2.2.3 is a Petri net modelling the failure and the repair of two components *c1* and *c2*. The token in the place *c1 working* represents a marking when the component *c1* is functioning.

A Stochastic Petri Net (SPN) is an extension of a Petri net where the time taken to fire a transition after it is enabled is modelled by a random variable. In practice, these random variables are described by exponential distributions and each transition is therefore associated

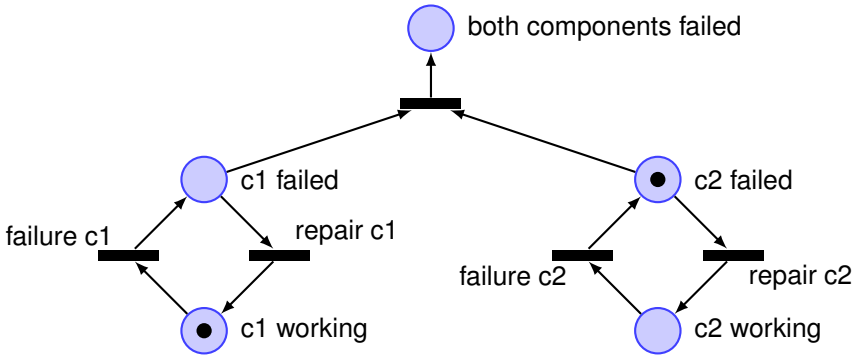


Figure 2.4: An example of a Petri net

with parameter λ called the *firing rate*. With this addition, a stochastic Petri net is actually a particular continuous time Markov chain. Ajmone-Marsan *et al.* defined Generalised Stochastic Petri Nets (GSPN) that allow immediate transition corresponding to a firing time of zero [78].

Petri nets are a general formalism used for process analysis. As the literature shows, they are able to model faults in an industrial system, but they are also able to model the system as a whole. This can be done at a much higher cost than simply focusing on failures, but could provide much more detailed about the circumstances of these faults or combinations of them. From a diagnosis perspective, this is an extremely interesting characteristic. However, to the extent of our knowledge, such detailed modelling has not been given to the literature.

Overall, designing a Petri net can be a tedious task, particularly for complex systems with a lot of dependencies between the components. However, they tend to retain a topology close to the one of the system which simplifies the update process.

Information available

In terms of quantitative analysis, the literature shows examples of computing reliability [43, 121, 31] and MTTF [43], using Monte Carlo simulations [43].

The versatility of Petri nets in terms of modelling capabilities is extremely useful because they can natively model reparation process and therefore compute associated metrics such as MTTR or MTBF.

Limits

The distribution associated with the firing of a transition can only be exponential. As shown previously, exponential distributions do not accurately model every type of failure and this constitutes an important limit in terms of obtaining a correct evaluation of the metrics required for diagnosis. Also, as stated earlier, the complexity of Petri nets that increases with the scale of

the system constitutes a major drawback in a broad use of the model. However, there have been proposition for a conversion process from a simpler model such as DFT to Petri nets [31].

Even though Petri nets are mathematicaly able to assimilate the modifications the occurence of events have on the system, which they can do by updating the markings, that would require the development of external tools to process such information. As such, we couldn't find any example of a live usage of Petri nets to track the evolution of ongoing incidents and couldn't evaluate their behaviour ina live usage.

Stochastic Reward Networks

In 1993, Ciardo *et al.* proposed stochastic reward nets (SRN). They associate reward rates to the network which corresponds to a reward rate associated with each marking of the net. With this reward rate, another way of computing the reliability of the system becomes available [77].

The major advantage of stochastic reward nets is a reduction in the size of the network, compared with a GSPN, since some behaviour of the system is no longer needed to be explicitly expressed using places and transitions but can rather be captured with the reward rates. They are therefore able to resonably model more complex system and would provide information for the diagnosis of a wider range of model than regular GSPN.

2.2.4 Safety models summary

Table 2.1 summarizes features available in safety models and useful for diagnosis. The table does not account, however, for some enhanced modelling capacities some models have against others. For example, dynamic fault trees can, contrary to regular fault trees, take into account the order in which events have to happen. They are therefore allowed to model more scenarios or yield more accurate results than their counterparts.

2.3 Security models

Information sought after in case of a purely security incident can be similar to the ones considered when a purely safety incident occurs. The names of the information differ but sometimes correspond to equivalent notions. These names, for that matter, are most of the time from the point of view of the attacker. For example, in that context, "success" would mean success for the attacker and achievement of his objectives.

Let us consider an intrusion detected in a workstation. The workstation is operated by a salesperson whose job is to overview and manage the sales of the company. The attacker, upon exploitation of vulnerabilities, has acquired a remote access to the workstation and escalated his privileges to those of the salesperson. He is therefore now able to add, delete or modify

Feature	Model
Minimal Cut Sets	Fault trees [32, 124, 24] Bayesian networks [13]
Cut Sequences	Dynamic fault trees [126]
Minimal Path Sets	Fault trees [2]
Reliability	Fault trees [120, 13, 42] Dynamic fault trees [41, 15, 83, 31] Bayesian networks [13, 117, 16] Dynamic Bayesian networks [38] Petri nets [43, 121, 31]
Mean Time To Failure	Fault trees [4] Dynamic fault trees [15] Petri nets [43]
Availability	Fault trees [42, 11] Dynamic fault trees [41, 15, 31]
Mean Time To Repair	Dynamic fault trees [15]
Mean Time Between Failures	Fault trees [4, 113] Dynamic fault trees [15]
Common Cause Failure	Bayesian networks [117]
Cost	Bayesian networks [51, 62]

Table 2.1: Summary of information available in safety models for diagnosis

information critical for the company. Having detected the intrusion, an IDS is able to inform about the presence of an attacker in this workstation. However, the alert raised by the IDS does not contain any information about the next possible actions an attacker can take such as modifying the amount of a given product to be shipped for a given sale. This represents important information to be able to evaluate the criticality of an intrusion and to understand the objectives of the attacker. Diagnosis being, amongst other things, about establishing the risk of an incident, getting this data represents valuable information that need to be provided by a diagnosis model.

It is also possible that an attacker raises several alerts when performing its attack. Those alerts need to be correlated to obtain the most accurate picture of the ongoing situation in order to provide the most sensible explanation. It is also possible that several intruders are present in the system at the same time, either contributing to the same attack or being completely independent. A diagnosis model should be able to consider and differentiate all of these situations.

Based on this qualitative analysis, a quantitative one can take place. Similarly to safety, an evaluation of the remaining time until completion of the scenario is still interesting and is called *Mean Time To Success* (MTTS) or *Mean Time To Compromise* (MTTC) and is the security tantamount to MTTF. The reliability finds its equivalent in the *success probability*. It represents the probability that an attacker achieves either an attack step or a complete scenario. It can be

done with or without taking time into account. Both of these measures can be used to determine how much time is available to deploy a countermeasure, or to estimate whether the threat is serious or not. They can also be used to estimate the expertise of an attacker. Indeed, while the MTTF of a component is most of the time deduced from feedback, the MTTS of an attack is much more controversially estimated by a security expert, and is function of the skill and experience of the attacker: a veteran attacker should take less time to perform an attack that a rookie one. Therefore, MTTS can be compared to the actual time it took an attacker to perform an action, and adjust the expected time of the future ones to have a more accurate MTTS of the overall scenario.

The repairability, however, is not a major concern in security since the restoration of the information system might come with a reconfiguration to remove the vulnerabilities exploited by the attacker when the restoration of an industrial system is more linked with the reparation of defective components. Also, deploying countermeasures often have the objective of preventing the attacker from progressing in its attack, and therefore, act before the scenario is completed. Thus, the availability, MTTR, MTBF do not necessarily have a relevant equivalent in security diagnosis. However, considering countermeasures is obviously a requisite of the model, but so is being able to model as accurately as possible the sequence in which the scenario has to happen.

We will now present several models used in purely security contexts able to provide one or more of these pieces of information.

2.3.1 Attack trees

The term "attack trees" designates a family of models that have a wide range of applications. This paper focuses on diagnosis and analyses models under that scope. As such, attack trees, in their most basic form, present few features exploitable for that purpose. In order to address specific issues, several refinements were produced.

In fault trees, the number and nature of features available depend on the mathematical tool used for the computation and a differentiation between mathematical tools is out of the scope of this study. In attack trees, the features rely on the variations of the models, and it is impossible to mention these capacities without mentioning the model associated with them. A common structure between all of these models exists, and we consider every model that align itself with this structure as being an attack tree. As such, the structure of this section is slightly different to the others of this survey: we mention the various models along with the feature they provide towards diagnosis.

Attack Trees (AT) are a widely used family of models to represent and analyse security in an information system. They were first mentioned by Schneider in 1999 [21]. They are derived from Threat logic trees [123], themselves inspired from fault trees. Compared to FT, they are

therefore fairly recent. However, their strong versatility combined with an increased need for security in information system triggered a rapid development of AT and AT-based models.

Attack trees are used to identify the steps necessary for an attack to be realised. They are created with a top-down process that stems from the root of the tree, the main goal of the attacker. The immediate steps that lead to the root are added to the tree as the children of the root. Each child is then expanded with the same process until no further refinement is required. The resulting attack tree is an extensive representation of the options an attacker has to realise the considered attack, and therefore highlights vulnerabilities that may be important to remedy.

AT-based model do not generally have the strict structure a FT has. Every node can loosely represent actions, events, goals or objectives. Contrary to FT where failures can only happen in leaf nodes, actions can happen at any level in the graph. Children are connected to their parents through logical connections (OR, AND, XOR, etc...) that are not always made explicit with logical gates. Figure 2.3.1 is an example of an attack tree where an attacker wants to modify an order placed by a client to a company. The attacker can modify the order either through the ERP software or the database directly.

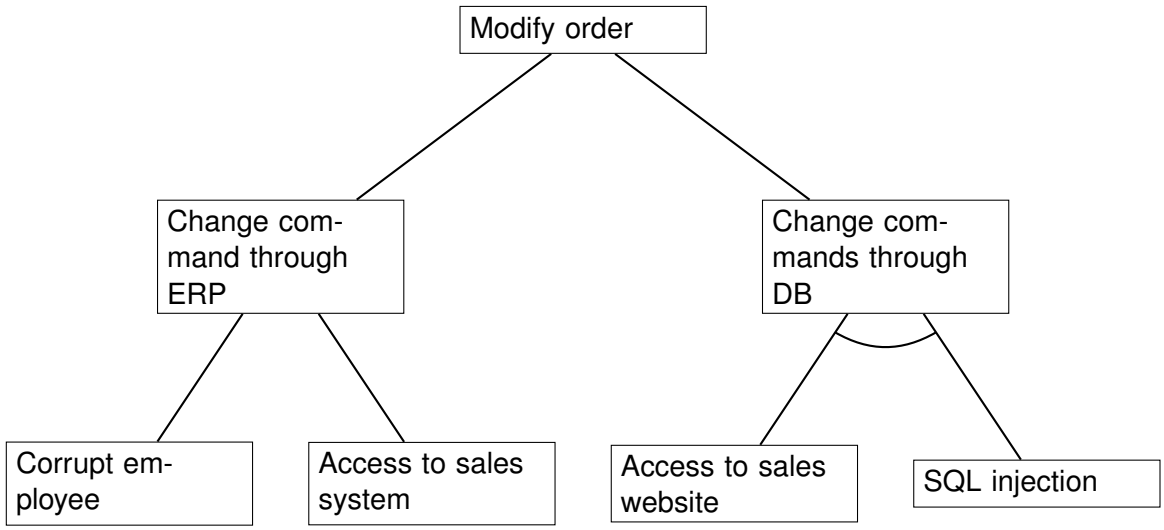


Figure 2.5: An example of an attack tree

This structure is the only greatest common divisor between various models that claim to be attack trees or are inspired of them. Most analysis capacities of attack trees are achievable through specificities provided by those models. We will therefore exhibit a range of models that cover metrics identified in the introduction of this section, such as Mean Time To Success, success probability, impact or risk.

Attack steps order

Most of the time in an attack scenario, actions need to be achieved in a certain order. For example, in the attack tree of figure 2.3.1, the SQL injection must happen after the attacker accesses the sales website. In a regular attack tree, such sequential behaviour cannot be modelled. A few models add this feature to attack trees.

In 2009, Khand [64] proposed to overcome static limitations of attack trees by using gates developed for FT and DFT. Even though Khand does not explicitly name its model, Kordy *et al.* [65] named it Dynamic Fault Trees for Security (DFTS). Khand proposes to use Priority-AND (PAND), k-out-of-n (k/n), Conditional subordination (CSUB), Sequence enforcing (SEQ) and Housing nodes. PAND and k/n and SEQ nodes are identical to the one in DFT, CSUB corresponds to FDEP gates. Housing nodes are used to switch on or off a specific subtree. They represent some different configurations of the system in which some actions may or may not be possible. No quantification is showcased in Khand's article, so it is unclear if the computing capabilities of DFT are inherited by this model.

Enhanced attack trees (EAT) [23] also provide a means to express this behaviour using an Ordered-AND gate, similar the the Priority-AND one.

Evaluating the progress of an attack

Some models add features to attack trees in order to evaluate how much of the attack scenario has been done and how far the attacker is from reaching his goal.

Augmented attack trees (AAT) [100] use the same general structure of an AT and add labels to the nodes of the tree. These labels quantify how many subgoals were compromised on an attack path and how many are to compromise to reach the root of the tree. The ratio of these two quantities expresses how close the attacker is from reaching his objective.

Similarly to the labels of an AAT, EAT [23] also add an attack level attribute, corresponding to how much of the attack has been accomplished at any given node. On top of Ordered-AND gates and an attack level, EAT add a Time-To-Live (TTL) feature to the nodes representing a time span outside of which the attack is considered failed. Their objective is to assist an IDS in the detection of ongoing attacks with the specific concern of reducing the number of false positives. In such, EAT are designed to function and analyse data in real-time. Their TTL feature is singular and interesting for diagnosis purposes. Nonetheless, even if they add time ordering of attacks, and TTL, they do not use the time when estimating how much has been done or is left.

AAT and EAT represent an interesting step towards computing risk by providing a measure of the attacker's success probability, this approach has some important limits. First, they do not take time into account when computing the probability, even though it is a major parameter to

consider when assessing the risk.

We must note that two other definitions of AAT were given but they do not provide additional information for diagnosis [96, 40].

Probabilistic success evaluation

Multi-Parameters Attack Trees (MPAT), proposed by Buldas *et al.* [22], do not take time into account. They analyse the behaviour of the attacker from an economics point of view and analyse several parameters, including the probability of success. To do so, they decorate the nodes of a standard attack tree with six parameters: the cost of the attack, the success probability, the probability of getting caught if the attack was unsuccessful, the expected penalty in case the attack was unsuccessful, the probability of getting caught if the attack was successful, and the expected penalty in case the attack was successful. These parameters are then used to compute the overall gain of the attacker. The probability of getting caught if the attack was unsuccessful and the probability of getting caught if the attack was successful are more interesting from an intrusion detection point of view than a diagnosis one.

In 2008, Jürgenson and Willemson gave a modified version of this model with only two parameters [57]. This contribution addressed several limits identified in the original model such as an overestimation of the gain of the attacker or the impossibility for the attacker to try several atomic attacks. They also noted that this model in which all of the atomic attacks can happen simultaneously is unrealistic [57]. Standard AT and AT-based models are parallel models. They address this issue by developing what they call a Serial Model for Attack Tree (SMAT) [58]. They claim to be able to compute the same results as MPAT, but with better accuracy since they consider a more realistic model for the attacker's behaviour. They were not included in section 2.3.1 because they model a sequential behaviour of the attacker but not the order in which the actions must happen.

For models taking time into account, we may cite Arnold *et al.* [7] who defined Time Dependant Attack Trees (TDAT) in 2014. Their objective is to evaluate the evolution of the success probability of an attack over time. To do so, they associate each leaf node with a probability distribution function. They can be any acyclic phased distribution (APH), and since APH distributions are topologically dense, any continuous distribution can be arbitrarily closely approximated by one [56]. They consider three types of gates: AND, OR and SEQ gates and provide formulas to recombine the distributions of child nodes into one for the gate. The process can then be iterated until the root of the tree has been processed and a distribution function for the overall attack is available.

It is to be noted that they consider non-leaves nodes as simple gates and not associated with an actual action. They are therefore, in that aspect, closer to fault trees than some other AT-based models. However, they constitute a powerful enrichment of regular attack trees in

providing them a means for temporal evaluation of the probability. They also provide actual mathematical and exact expressions of the result contrary to stochastic methods.

Impact and risk

To compute the risk, after having the probability of success, the impact is required. In that case, we compute risk with the formula $risk = impact \times probability$. Several models provide a way to compute either just the impact, which can be named "cost", or the risk.

Multi-Parameter Attack Trees, with their various parameters, provide attack trees with a means of computing gains for the attacker, which can be used to evaluate the risk. And therefore, Serial Model for Attack Tree can too.

Improved Attack (IAT) Trees, proposed in 2011 by Wen-ping and Wei-min [75], add various parameters to leaf nodes and use them to compute risks evaluation of parents nodes until reaching the root. They provide with an evaluation of the overall probability, impact and risk of the attack. However, they seem to adopt a formalism closer to the one of fault trees than attack trees. Indeed, their formulas make us believe that nodes that are not leaves are just logical gates and do not correspond to an actual action by the attacker. Therefore, they seem to have provided a way to compute risk at the expense of weaker modelling capabilities.

Countermeasures

As we developed earlier, being able to take countermeasures into account is important to model complete scenarios.

Defence trees (DT) are a way to add countermeasures to an attack tree [12]. Sets of countermeasures are added to the leaf nodes. They also provide a framework to evaluate the cost and the impact, from an economics perspective, of the various choices attackers and defenders can make. Their behaviour are then analysed through the scope of game theory.

We may also cite Countermeasure Graphs [10], a framework based on attack trees that aims at selecting a set of countermeasures to deploy in order to maximize the efficiency in terms of effectiveness versus cost.

Recap of information available

If we consider all of the mentioned models as attack trees, we are able to say that attack trees are able to compute several metrics such as sequential modelling [64, 58], success probability [100, 75, 7], MTTS [7], the impact [10, 75, 22, 57, 58], the risk [123, 22, 57, 75]. It shows that attack trees can be used live [23] and take countermeasures into account [10, 12].

2.3.2 Other attack graph based approaches

Other graphic based approaches can be loosely related to attack trees. They do not, however, have the same structure as the family previously discussed and were therefore isolated from the rest of the models.

Compromise Graphs

Compromise Graphs (CG) were introduced by McQueen *et al.* in 2006 [79]. They are directed graphs where nodes are attack steps and edges represent the achievement of attack steps. The edges are weighted with the time estimated to be necessary to realise the corresponding step. CG are used to compute Time To Compromise (TTC) in the form of the shortest path from the input node to the output node being the attacker's objective. They are used to compute the effects of security measures on the TTC.

CG have a very simple modelling that makes them relatively easy to used, but that does not allow for thorough quantitative analysis. They only produce a TTC estimation although taking into account several parameters including the skill of the attacker or the number of vulnerabilities for a same attack step.

Vulnerability Cause Graphs

Vulnerability Cause Graphs (VCG) were designed in 2006 with the objective of designing more secured software [6]. They are used to analyse the causes of a vulnerability. A VCG is a directed acyclic graph whose nodes are vulnerabilities and causes, and edges are logical connections between nodes. Nodes without children are vulnerabilities and other nodes are causes and represent events that can cause vulnerabilities.

Although they are meant to be used during the design of a software, they represent an interesting way of linking vulnerabilities, and therefore attacks exploiting these vulnerabilities, with causes. VCG are also general enough to be potentially applied to safety scenarios. However, they do not offer quantitative analysis and, in order to be used for a diagnosis purpose, would need to be integrated in another model.

Nzoukou *et al.*, 2013 [88]

In 2013, Nzoukou *et al.* [88] have proposed a framework to compute MTTC in information system. Their model is based on attack graph to obtain the dependencies between atomic attacks and a BN to represent the conditional dependencies between events. They map MTTC to each atomic attack and give a formula to obtain the one of the overall attack based on the conditional dependencies.

Their work puts a lot of emphasis on the acquisition of the atomic MTTC, which is a recurrent issue in security. They base the computation of the MTTC on the Common Vulnerability Scoring System (CVSS) [80] for each documented attack. When no vulnerability documented, they also provide a means to estimate its MTTC corresponding to the one of a zero-day attack. Lastly, they take into consideration the fact that the re-exploitation of a previously exploited vulnerability is quicker, and update the MTTC accordingly.

It is unclear how they generate the attack graph and therefore to identify a probable contribution on that part. However, they are able to obtain any information present in an attack graph such as a description of scenarios or dependencies between attacks. MTTC also becomes available for overall scenarios after being modelled for each atomic event. Using a BN, they are able to update the state of the system based on evidence that attacks happened, and compute up to date MTTC.

We must stress that the main contribution of this article relies on the use of CVSS to obtain MTTC. Indeed, as we stated in Section 2.3, the relevance of using MTTC in security is questioned because unlike in safety, there is no reliable way to obtain this information. The information is most of the time provided by an expert and is therefore subject to bias. Using CVSS, a standardized and reasonably trusted system, to calculate an MTTC improves the confidence one would have in this information.

2.3.3 Bayesian Networks For Security

As developed in section 2.2.2, Bayesian Networks are used to evaluate safety. The literature also shows that they can be used for security in various configurations to obtain different information.

Qin and Lee [98], use Bayesian networks as a predictive tool to anticipate an attacker's behaviour. They propose a set of procedures to transform an attack tree into a Bayesian network and estimate the probabilities of the nodes of the Bayesian network. Using the network, they can estimate the likelihood for a node of the network to be a goal of the attacker, based on observed evidence. Using BN, their results are devoid of any temporal evaluation. Gribaudo *et al.* [50] or Dantu *et al.* [37] also use BN as a mathematical tool to compute probabilities based on an attack graph.

Bayesian networks are not always derived from attack trees and can also be used directly. For example, Feng *et al.* [45] have used Dynamic Bayesian Networks to predict goals when events, in the form of system calls, happen. Althebyan and Panda [3] use BN in conjunction with dependency graphs and knowledge graph to anticipate the risk that an attacker might obtain some confidential information based on a detected attack. However, they amalgamate risk and success probability so they do not quantify the impact on the system.

2.3.4 Dependencies models

Unlike safety models that remain topologically close to the system, security models lose the ability to quickly associate an element of the model with a specific component. Therefore, applying the consequences of a modification in the system to the associated model is a difficult task that often results with the analysis of the system being remade from scratch. The quality of a model does not solely rely on its metrics or features but also on its convenience of use. When establishing models to evaluate the security of a system, this is a particularly important parameter to consider. In order to address this problem, a range of models, that we call *dependencies models*, were produced. Their general principle is the same: elementary attacks are described using a specific language, and then fed to a correlation engine to obtain complete scenarios. Indeed, from a security expert's point of view, it is much easier to associate a component with its vulnerabilities and attacks than to do it on a system scale. By automating the generation of the scenarios, a lot of resources can be saved and human errors avoided.

These languages describe an attack by its conditions and consequences. The attack cannot happen unless its conditions are met and, if realised, have the described consequences on the system. Conditions and consequences can have different names, such as preconditions and postconditions [36, 87] or prerequisite and consequences [86], but always represent the same thing. They are typically coupled with a correlation engine that apply the following principle: if the consequences of an attack A are the conditions of another attack B, then attack B depends on attack A to be realisable. The correlation engine tries to connect every two pairs of attacks and then output an attack graph, a more general model than attack trees.

We illustrate this section with LAMBDA and CRIM since they are used in the model presented in this thesis in Chapters 3, 4 and 5 but we mention other models of this family and some of their specificities.

LAMBDA [36] is a language used to describe attacks with describing the preconditions and the postconditions. It uses three languages to model an attack. First, the language L_1 used for the state description corresponds to the logic of predicates. It describes preconditions and postconditions as conjunctions, disjunctions and negation of predicates. Then, the language L_2, similar to event calculus, describes the transitions to the occurrence of events using logical operators $\neg$ and $land$ and the equality operator $=$. Last, the language L_3 describe the combination of events based on the event calculus algebra. LAMBDA also has the objective to assist IDS in detecting those attacks. That is why an attack in LAMBDA also has parameters to detail how the attack can be detected, and to check that it effectively took place. The languages L_2 and L_3 are also used to describe how to detect the attack and how to verify that it effectively took place in order to assist intrusion detection.

CRIM [35, 34] is a correlation engine taking LAMBDA models as inputs and outputting complete scenarios. Indeed, LAMBDA is convenient to describe atomic attacks independently form

attack $attack\_name(arg_1, arg2, ...)$
pre: $cond \in L_1$
post: $cond \in L_1$
scenario: $expr \in L_3$
where $cond \in L_2$
detection: $expr \in L_3$
where $cond \in L_2$
verification: $expr \in L_3$
where $cond \in L_2$

Figure 2.6: Outline of an LAMBDA event

each others, and CRIM then uses them to construct global scenarios. The general principle behind the construction is that if one of the postconditions of an attack A matches one of the preconditions of another attack B, then an attacker could perform attack A in order to later perform attack B. After trying to match every couple of attacks, CRIM produces an attack graph consisting of all the possible attack paths for an attacker to fulfil his objectives.

However, CRIM is not limited to attack graph generation. CRIM is usable live thanks to a powerful instantiation system. When an attack is detected by an IDS, for example, an alert can be raised. CRIM understands alerts in the IDMEF format, and uses them to instantiate the element in the graph corresponding to the attack detected. Instantiation consists in fixing the variables of the predicates of the event, and propagate these values to connected events. If two alerts are raised and the instantiation of the parameters generate conflict, they are considered as being part of two different attacks. If not, the two alerts are deemed correlated and as considered as part of a more global attack.

Upon other languages that function with the same principle, we may cite STRIPS and its evolution Concurrent STRIPS [20] that focuses on the modelling of concurrent attacks.

Ning *et al.* [86] defined Hyper-alerts in 2002 that describe elementary attacks by means of facts, prerequisites and consequences. These hyper alerts can then be correlated in an hyper-alert correlation graph. An initial difference between hyper-alerts and LAMBDA is that hyper-alerts were, from the beginning, defined to describe elementary attacks to be correlated when LAMBDA only achieve this aspect with CRIM. But *in fine*, Hyper-alerts and LAMBDA/CRIM are very similar in that regard. Noel *et al.* [87] also have a similar model.

FIGARO [19] is a language developed by EDF[1]. From a diagnosis point of view, it has the advantage to embed probabilistic information in each elementary event. FIGARO was initially developed for safety management but was later adapted to consider security. An event in FIGARO is still described in terms of preconditions and postconditions, but is also associated with a probabilistic distribution that represents the time necessary for the event to be realised if the

1. Électricité de France (EDF) is the third largest electric utility supplier world wide and the first in Europe.

preconditions are met. A FIGARO model is then fed to a framework named KB3 that can output various models such as fault trees (see Section 2.2.1) or BDMP (see Section 2.4.1). However, KB3 is a proprietary framework and little to no information was abled to be gathered about it.

Information available

Dependencies models are very powerful to obtain complete scenarios: they are able to capture the dependencies between the various components and describe how their associated vulnerabilities can be combined for an attacker to achieve identified objectives. Their most valuable output is the generation of a complete attack graph.

Unfortunately, the attack graph output by LAMBDA allows for no quantitative analysis. The strength of LAMBDA/CRIM is twofold. First, it is designed to be used live and in conjunction with IDS or similar systems. CRIM is therefore able to correlate events when they are happening and record any change to the state of the system. Secondly, designing attack graphs is increasingly difficult with the complexity of the system. Instead of asking security experts to design the whole tree, with the risk of forgetting dependencies between events and having to redesign most of it when the system changes, LAMBDA and CRIM offer a simpler process. Experts simply have to design atomic attacks, and the attack graph is automatically generated.

Limits

Dependencies models were not developed with the objective of enabling quantitative security analysis. Therefore, few metrics can be extracted from these models. We note, however, that Kanoun *et al.* [63] have proposed an extension of LAMBDA/CRIM to compute MTTS, named Mean Time to Intrusion Objective in the paper. To do so, they transform the attack paths produced by CRIM in a Markov model where transitions between states correspond to an attacker achieving an attack step. They limit their analysis to exponential distributions.

Being able to consider several attackers simultaneously in the system is not a common feature in security models. LAMBDA/CRIM or other models correlating meta-information from alerts raised by IDS are able to do it, to some extent. However, coordinated attacks are a subset of those and are not modelled by LAMBDA/CRIM. However, a few models such as Concurrent STRIPS [20] or LICCAS [112], were designed to address this issue.

2.3.5 Petri nets for security

As presented is section 2.2.3 Petri nets are a very general formalism that can also be used for security purposes. We do not dwell to much on the use of Petri nets for security since they are identical to the safety applications. But for the sake of extensiveness, we have to mention that

generalized stochastic Petri nets can also be used to evaluate the probability of occurrence of a scenario [116, 72] or MTTS [72].

Petri nets benefit from their general formalism to be able to model seamlessly safety and security. However, the limits identified for safety are still valid for security, with a scalability problem and the obligation to consider exponential distributions.

2.3.6 Security models summary

Table 2.2 summarizes information available in security models and useful for diagnosis.

Information	Model
Sequential order	Attack trees [64, 23]
Attack progress	Attack trees [100, 23]
Success probability	Attack trees [22, 57, 58, 7, 75] Bayesian networks [50, 37] Petri nets [116, 72]
Mean Time To Success	Attack trees [7] Compromise graphs [79] Nzoukou *et al.* [88] Petri nets [72]
Impact	Attack trees [10, 75, 22, 57, 58]
Risk	Attack trees [123, 22, 57, 75] Bayesian networks [3]
Countermeasures	Attack trees [12, 10] CRIM [36, 35, 34]

Table 2.2: Summary of information available in safety models for diagnosis

2.4 Hybrid models

In this section, we discuss about models that are explicitly used in the literature to address situations where both safety and security are considered. Bayesian networks, for example, are showcased in the literature to model both safety and security but not simultaneously and are therefore excluded from this section. Indeed, we have evidence that they can provide useful information for diagnosis, as mentioned in the previous sections, but there is no contribution on how they handle the interactions safety and security have, which is the focal point of this section. That is why we focus on the specificities of mixing safety and security and not linger on

metrics useful for purely safety or security situations. It is obvious that they are still needed for diagnosis but they are not the subject of the following paragraphs.

Before studying such models that we call "hybrid", we will first develop on the differences between safety and security modelling,

A major difference between component faults and security attacks is that attacks require an action from an attacker when component faults happen spontaneously or without premeditation [50]. As a direct consequence, the modelling of events will not be the same, particularly from a probabilistic point of view. Some similarities still exist but dedicated models generally exploit specificities of situations they address and become therefore not general enough. EAT [23], for instance, have a time to live feature not really relevant for component faults. Another consequence of this difference caused by the spontaneousness of safety events compared to security events is that safety events are somehow bound to happen, whereas attackers can be stopped before they can harm the system: the objective of safety is to maximize the availability of the system when the objective of security is to limit the damages of attackers. An example of how safety and security can look for adversary objectives was given in section 1.3.1. A sensible safety/security diagnosis model has to acknowledge for that difference of objectives and reconcile them.

As demonstrated with the example of section 1.3.1, when considered independently, security and safety can come up with unacceptable solutions. Even though it is not the responsibility of the diagnosis to select a response, it is necessary to take them into account to detect and report those conflicts, and identify the consequences of considered responses to an incident. And since these countermeasures modify the system, they may have undesired side effects that open new opportunities for an attacker or decrease the reliability of a system. As such, they may not simply be considered as terminations of scenarios, like in defence trees, but need to be fully integrated into them like in LAMBDA/CRIM.

However, we must point out that countermeasures are not the only way safety and security interact with each other:

- An attacker can obviously want to cause a failure: sending orders to an automate to disrupt the industrial process and cause a catastrophic failure.
- An expected failure can provide new opportunities to an attacker: a malfunctioning pressure regulation system can be exploited to cause a catastrophic explosion.
- An attack can cause an unplanned failure: an attacker executing a denial of service on the gateway of a company can disable any remote supervision of the industrial system and let it shift to a breakdown.

The ramifications of these interactions are potentially infinite, with an attack causing a failure itself allowing an attack, etc... Therefore, deporting the security modelling as an input to an

attack tree like extended fault trees do [47] is not sufficient to cover all scenarios. We discuss about extended fault trees in section 2.4.2. Being able to represent interactions between safety and security incident is an essential step towards obtaining an efficient diagnosis model.

Indeed, a mission of diagnosis is to find the origin of a problem. If we consider the example of section 1.3.1 where an attacker would destroy centrifuges. The outcome of the scenario is the destruction of a means of production and would be handled by the safety supervision. However, the origin of the incident is a security issue. In today's architectures, safety and security being supervised separately, this discovery of the origin of this attack is not an automatic process. Diagnosis models have to provide a means to achieve that and it goes with representing these interactions between safety and security.

The required metrics for safety/security diagnosis are the same as the ones defined in the two previous sections 2.2 and 2.3. Some of them are defined for a specific context but have a counterpart in the other, as seen in the introduction to section 2.3. For example, the MTTF of a safety events can be mixed with the MTTS of a security event to obtain the mean time to realisation of a combination of the two of them. These metrics are therefore homogenous, since they both represent a duration. However, determining MTTF for safety is done through return on experience that provide with a high level of confidence whereas MTTC for security is often appreciated by security experts and subject to discussion (hence Nzoukou *et al.* [88]). Therefore, mixing MTTC with MTTF, albeit mathematically sound, will dilute the confidence one have in the values output by the system. As a result, one must be cautious and check the rationality of mixing two metrics that seam similar. For instance, the impact is defined both for safety and for security but some model measure it from a system point of view and others from an attacker point of view. Mixing both will result in modelling and diagnosis errors.

2.4.1 Boolean-logic Driven Markov Processes

Boolean-logic Driven Markov Processes (BDMP) were introduced in 2003 by Bouissou and Bon [17]. It was presented as a synthetic way to model complex systems but still keeping the mathematical properties inherited from Markov processes. They were initially designed as a tool to model safety and later adopted to security [93].

BDMP are a powerful graphical model with an underlying probabilistic component. The graphical model is similar to a fault tree with leaves consisting of events, other nodes being logical gates and the root also corresponds to the undesired events. However, a new graphical element is added: triggers. Triggers are arrows that point from an origin node to a destination node. Its meaning is that the event, or combination of events, represented by the destination node, cannot happen unless the origin node has been realised. It is a way of introducing sequence modelling and dependencies between events. Figure 2.7 is an example of a BDMP.

Each leaf corresponds to an event that can be in several modes and each mode can be

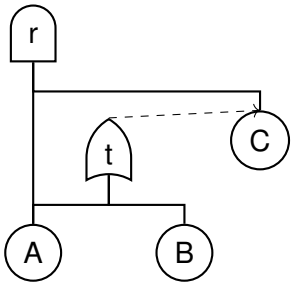


Figure 2.7: An example of a BDMP

in several states. Each mode is associated with a Markov process. Probabilistic distribution functions, called transfer functions, are used to describe how an event can switch between modes. A more thorough definition was given by *Pietre-Cambacedes and Bouissou, 2010* [91]. This approach enables the modelling of a potential repairability of the component represented by a transfer function from a damaged mode to a functional mode.

BDMP are used to analyse the dependability of modelled systems. They are used to compute the general probability of occurrence of a scenario as well as the importance of each fault, or atomic attack, in term of contribution to the general failure. The initial contribution stands in a simpler and more readable modelling of Markov processes but still keeping their powerful mathematical outputs. In 2010, with the addition of security [93], it became one of the few models to propose a semantics for representing both safety and security events. Kriaa *et al.*, 2014 [66] showcases how BDMP can be used to measure the influence of security vulnerabilities in an industrial system.

Contributions to diagnosis

BDMP describe how individual events interact with each others to form complete scenarios. They have an explicit way to describe these logical interactions but have also a framework to express sequential temporal dependencies between events. Therefore, they are not only able to provide usual structural information such as minimal cut sets [17] but are also able to consider sequences of events when analysing the past or generate hypothesis on the future.

From a probabilistic point of view, BDMP are capable of computing usual metrics such as reliability or MTTF [17, 93, 66], meaning that they can provide on demand probabilistic evaluation of future events and of how much time it will take to realise some sequences of events. BDMP are able to measure the contribution of each event to the overall probability of success [17, 93, 66]. BDMP are therefore able to compute the likelihood of occurrence of past or future events, raising suspicion if relevant, as well as giving the most probable explanation from a probabilistic point of view.

Limits

BDMP cannot model every type of scenario. In many cases, restoring a faulty component to a functioning state ends up, after eventual propagation, to a functioning state of the system. However, in some cases such as a knowledge gain by an attacker, providing a countermeasure to the only abnormal component of the system does not prevent the scenario from going forward. To our comprehension and experimentations of the model, the way the triggers and the process selectors function mean that such situation cannot be handled by BDMP.

BDMP do not have an embedded metrics for the impact of events or sequences of events on the system and are therefore unable to compute risk.

On more general aspects, even though BDMP require little to no other knowledge than the one already acquired by safety experts, they are quite tedious to construct and suffer from a poor adaptability of the model in case of a modification of the system. Automatic generation of BDMP are possible using the FIGARO language [19] and the KB3 platform.

Generalized Boolean-logic Driven Markov Processes

Generalized Boolean-logic Driven Markov Processes (GBDMP) are a recent extension of BDMP [92]. They aim at addressing three limits identified in BDMP: triggers are too restrictive in the way components interact with each others, components can only have two modes while an overclocked or degraded mode can sometimes be relevant, and triggers never fail. The BDMP framework is modified to include two new elements: switches and Moore machines. Switches connect several nodes and contain a Moore machine that outputs a reconfiguration strategy that depends on the state of the input nodes.

GBDMP are used to provide a means to evaluate the impact of various reconfiguration strategies and the effects of a potential failure in these reconfigurations. This is achieved at the cost of a model even more complex than BDMP.

2.4.2 Integration of fault trees and attack trees

Extended Fault Trees

Fovino *et al.* [47] mixed FT and AT, creating extended fault trees (EFT). Their reasoning is that the top event of an attack tree corresponds to an undesired event, similar to a basic event in a fault tree. Therefore, they basically plug the attack tree at the bottom of the fault tree, where the realisation of the attack objective could disrupt the system. To do so, they first adapt the attack tree so its structure complies with the fault tree framework and then connect it to the corresponding event or logical gate of the FT.

EFT represent an ingenious way to model both security and safety at the same time, but keeping the analysis tools of fault trees. However, they also inherit the same limits and are not able to overcome them (no time modelling, etc.).

Attack-Fault Trees

Attack-Fault Trees (AFT) were invented by Kumar and Stoelinga in 2017 [67]. Their proposal is to decompose an undesired event in smaller sub-goals until reaching basic component failures, basic attack steps or instant failures. The subgoals are described using logical gates taken from dynamic fault trees and attack trees such as the AND, OR, SAND, VOT(k)/n, PAND, FDEP and SPARE gates. In a way, attack-fault trees can be seen as a mix between fault trees and attack trees. The resulting model is then translated in several stochastic timed automata to compute some identified safety-security metrics: probability of disruption, expected cost of malicious disruption and mean-time to malicious disruption.

In their mixing of security and safety events, Kumar and Stoelinga are conscious to capture specificities of each events, much like BDMP do it. Where BDMP associate different Markov processes with each type of event, AFT do the same with stochastic timed automata. As such, they present a rather reasonable way of mixing safety events and security events. However, they still suffer from the inability to consider other probability distributions than the exponential one. Indeed, as we saw in section 2.2.2 other distributions are more suited to represent some types of events [114]. They also do not consider countermeasures or events that are neither failure nor attacks. Indeed, such events can provide meaningful context by tracking a relevant change in the system.

2.4.3 Hybrid models summary

Table 2.3 summarizes information available in hybrid models and useful for diagnosis.

Information	Model
Cut sets	BDMP [17, 93]
Sequential order	BDMP [17, 93] Extended fault trees [47] Attack-fault trees [67]
Success probability	BDMP [17, 93] Extended fault trees [47] Attack-fault trees [67]
Mean Time To Success/Failure	BDMP [17, 93] Attack-fault trees [67]

Table 2.3: Summary of information available in safety models for diagnosis

2.5 Classifying models based on their contributions to diagnosis

In this section we propose a taxonomy to characterize several capacities of models and discriminate them either from their ability to be used live during an incident or from the information they provide. Being used live encompasses the categories adaptative and real-time. Provided information considers information inferred from the incidents with categories hybrid, descriptive, explicative, evaluative and remediative.

2.5.1 Description of categories

Hybrid A model belongs to the *hybrid* category if it is designed to model both safety and security events. The literature has to explicitly showcase the use of the model in hybrid contexts; we do not simply assume such capacity.

Descriptive A model is said *descriptive* when it explicitly decomposes a global scenario in several elementary events, and provide a logical structure to describe in which way these elementary events compose the global scenario. For the most simple cases, when an event can only be in a binary state, that is realised or not, a Boolean expression would be extractable from the model. A descriptive model can therefore provide information about events to monitor and describe the state of the system when alerts are raised. The vast majority of models presented in this thesis are descriptive.

Explicative An *explicative* model considers the temporal order in which elements have to take place. It does not necessarily quantify the time but it is at least able to organise events

relatively to each others. This is an important feature for several steps of diagnosis. First, during the correlation of incidents, it allows to know which ones are supposed to happen before others: alerts being raised in the wrong order could for example mean that they are actually not related. Then, when performing explanation, knowing the order in which events arrive will help to investigate the origin of the incident, as well as generate hypothesises for future events.

Evaluative A model is considered *evaluative* if it is able to compute either a probability of occurrence or the impact of an event or a series of events over a period of time. It is usually done by associating each atomic event with a probability distribution function, failure rates or costs, and then using recombination formulas to obtain the compositions of events. The perks of an evaluative model are that it is able to provide a probabilistic evaluation of how much time is available to deploy a countermeasure or how much is at stake. Moreover, when an event occurs, it is able to compute the probability that this event should have happened. When that probability is unreasonably low, it might point out a suspicious behaviour. This is reasoning beyond the model but evaluative models have the ability to raise such issues.

Remediative A *remediative* model considers countermeasures as part of the scenario. This is particularly useful for scenarios mixing security and safety because they can be antagonistic [94]. Therefore, knowing the side effects of a safety response on the security, and vice versa, might cause system administrators to reconsider such deployment. In any case, this is valuable information for the diagnosis.

Adaptative *Adaptative* models are evaluative models with the ability to adjust probabilities based on observed events. It means both updating the probabilities based on events that have already happened, and taking into account the elapsed time for unrealised events. Evaluative models using failure rates usually lack this sort of flexibility necessary to provide the most practical information.

Real-time Establishing a link between an event and an alert raised by a SIEM[2], a SCADA or any other agent is fundamental to keep track of events that have occurred in the system. But being able to do it live greatly increases the value of a diagnosis model in an environment where reactivity and swiftness are key. *Real-time* models are able to track occurred events in order to update their picture of the system and provide up-to-data analysis. If the model is also remediative, it should also be able to monitor the deployment of countermeasures.

2. Security Information and Event Management, it is a system responsible for handling security events in an information system.

2.5.2 Classification of studied models

Models described in Sections 3, 4 and 5 are added to their respective categories as displayed on Table 2.4. For the sake of conciseness, when all of the variations of a particular model pertain to the same category, only the parent model is listed. For example, all the variations of attack trees reviewed are descriptive: only "attack trees" are listed in table 2.4.

Category	Description	Models
Hybrid	Able to consider both safety and security events	Bayesian networks [89] BDMP [17, 93] Extended fault trees [47]
Descriptive	Decomposing a high-level event in elementary ones and making explicit logical connections between the elementary events that lead to the high-level one	Fault Trees [120] Bayesian networks [89] Attack trees [21] LAMBDA/CRIM [36, 35, 34] Nzoukou *et al.* [88] BDMP [17, 93] Extended fault trees [47]
Explicative	Able to provide an explanation on what events have taken place, and in what order, as well as conjecturing on future events and probable future outcomes	Dynamic fault trees [41] Dynamic fault trees for security [64] Extended attack trees [23] Time dependant attack trees [7] LAMBDA/CRIM [36, 35, 34] BDMP [17, 93]
Evaluative	Able to provide a probabilistic evaluation of each atomic event, when described, and/or the occurrence probability of the top event	Fault trees [120] Bayesian networks [89] Advanced attack trees [100] Time dependant attack trees [7] Defense trees [12] Nzoukou *et al.* [88] BDMP [17, 93] Extended fault trees [47]
Remediative	Ability to consider countermeasures and their consequences	Defense trees [12] LAMBDA/CRIM [36, 35, 34]
Adaptative	Can update the variable of the model based on observation	LAMBDA/CRIM [36, 35, 34] Nzoukou *et al.* [88]
Real-time	With embedded capacities to catch and analyse alerts raised by agents such as SCADA, IDS or SIEM	Extended attack trees [23] LAMBDA/CRIM [36, 35, 34]

Table 2.4: Capabilities of the models

Table 2.4 illustrates that very few models are suited for a live analysis of a problem. This is probably reinforced by the fact that there is no standard for safety alert exchanged between agents, unlike security where the IDMEF format exists [39]. Indeed, safety alerts are always raised and handled by the SCADA. In the rare cases where the SCADA have to spread this information, a custom mechanism has to be created. This is a major issue that need to be

addressed for diagnosis model to use information contained in the SCADA.

We also notice that the consideration of countermeasures is not deemed important in most existing models. We have however pointed out in Sections 1 and 2 that they can have important consequences on the system and, therefore, radically change the selected response to an incident.

To conclude this chapter, we begin by remembering that legacy diagnosis questions and techniques are not suited for cyber-physical systems. We define diagnosis in a safety-security context as "the process of deducing the most likely mechanism that caused an observed condition, and identifying the future outcome". Based on this definition, we identify seven functions that have to be covered by a diagnosis model: incident detection, incident correlation, explanation, risk evaluation, likelihood evaluation, countermeasures and on-line.

Based on this observation, we have analysed existing models not necessarily oriented towards legacy diagnosis but rather providing relevant and useful information for diagnosis in a safety security context. Before dwelling on the models, we first illustrate on practical examples what information is sought after and how the models can provide it. After that analysis, we classify the models, according to their contributions to diagnosis, throughout seven categories: hybrid, descriptive, explicative, evaluative, remediative, adaptative and real-time.

First, we can deduce from our classification that none of the analysed model fill all of the functionalities previously identified. However, we acknowledge that the needs of a diagnosis model are contextual. We have established this classification with the most general approach of the problem. Yet, for specific needs, some functionalities might not be needed and a model would not need to pertain to all categories. Therefore, our classification can still be used to identify both the needs of a specific context and a suitable model.

We must nonetheless point out that there exists relatively few models that consider both safety and security, which limits the pool of available models for safety-security diagnosis. The literature shows that works are being conducted to define new models or to adapt existing ones. However, the process of including safety events in a security model, and vice-versa, is far from being trivial, and studies are being conducted to address this issue.

CHAPTER 3

PROS^{2}E: A NEW PROBABILISTIC REPRESENTATION OF EVENTS

3.1 Introduction

This chapter presents the first iteration of PROS^{2}E, a Probabilistic Representation Of Safety and Security Events. It is a first approach to providing information for diagnosis. PROS^{2}E is used to model individual events from a logic and probabilistic point of view. It is more designed as a proof of concept than a practical model but constitutes a necessary step to explore the whys and wherefores of diagnosis as well as lay the mathematical foundations on which dwells the second version of PROS^{2}E, whose Chapters 4 and 5 are dedicated to. Of course, this model has to respect the same constraints that we identified in the previous chapters and that we recall as a preamble.

Concerning its capacities, a good diagnosis model should be able to provide several metrics such as Mean Time To Success/Failure, Success Probability, Likelihood, etc. It should be able to be used live and process new information in real-time in order to keep track of ongoing situations. It should have advanced modelling capabilities in order to consider complex scenarios, by addressing the side effects of counter measures for example. But the functionalities of the model are not the only criterion to consider. Indeed, we want to provide with a model that can be easily handled by system experts. As such, we have added the constraint that it should reuse as much available knowledge as possible, and not call for unrealistic or unorthodox methods. Moreover, since cyber-physical systems tend to evolve at a much higher rate than traditional industrial systems, a good diagnosis model should consider this and offer a simple way for system experts to update an existing model. This first version of PROS^{2}E presented in this chapter does not have the advanced modelling capabilities yet, this will come in the next chapter. But we carefully designed PROS^{2}E to cope with the rest of the constraints listed here, as well as having an architecture that can be easily adapted to enhance its functionalities if required.

PROS^{2}E is adapted from LAMBDA and CRIM, whose functions that are relevant for our model are presented. This chapter also contains the new probabilistic layer of the model, along with the corresponding proofs. It is concluded by a use case that presents various metrics that can be obtained, and how they can be used for diagnosis.

3.2 Preliminaries: logical dependencies

As we stated in the introduction of this chapter, we use the methodology of dependencies models in order to obtain a representation of the events that can occur in the system, and particularly the one of LAMBDA/CRIM [36, 35, 34]. LAMBDA/CRIM was designed with a security purpose, but can be adapted to represent more general event. This section presents the functionalities of LAMBDA/CRIM that were reused for PROS^{2}E, namely the preconditions and postconditions, the semi-explicit correlation of CRIM, and the detection and alert management.

3.2.1 Conditions and consequences

LAMBDA is a language that models attacks using the conditions for them to be achievable and their consequences if performed. An event being defined as a modification in the state of the system, it still happens under certain conditions and have an impact on the system. The general formalism of LAMBDA can easily be extended to safety and security events. The conditions and consequences are expressed using first order logic, and we reuse its vocabulary in the upcoming paragraphs.

The state of the system, or system state, is a list of variables. These variables can be checked or allocated using predicates. Saying that an event can happen if the event is in a specific state is therefore equivalent as saying that in can happen if a set of variables meet certain conditions. The conditions on these variables are listed in the *preconditions set*. Similarly, the consequences on the system can also be unequivocally expressed using a *postconditions set*. Both preconditions and postconditions sets are predicates combined with the logical connectives $\wedge$, $\vee$ and $\neg$.

3.2.2 Semi-explicit correlation to obtain scenarios

CRIM has a correlation engine that is used to obtain complete scenarios from LAMBDA models. It draws links from postconditions of a LAMBDA model to the preconditions of another. The idea is that if the consequences of an event are also the conditions for another one, the realisation of the first event might trigger the second. It is therefore relevant to consider that there are sequential dependencies between the two events.

To express this formally, let us consider two LAMBDA models A and B. Let $Post(A)$ and $Pre(B)$ respectively the postcondition set of A and precondition set of B. $Post(A)$ and $Pre(B)$ are sets of predicates. A and B are said to be correlated if there exists predicates $pred_A$ in $Post(A)$ and $pred_B$ in $Pre(B)$ such that $pred_A$ and $pred_B$ are unifiable through a most general unifier (mgu) θ.

Two correlated events correspond to two events that can potentially happen in sequence. They can therefore be represented in a directed graph to visually express this sequentiality, or dependency, for example.

Using this principle, by trying to correlate every two pairs of LAMBDA models, CRIM is able to obtain the dependencies between every pair of events, and thus obtain complete scenarios out of atomically modelled events.

3.2.3 Raising alerts for an on-line use

CRIM is not only able to generate scenarios but is also able to process alerts. It is designed to work with an IDS or a SIEM, software able to generate and centralise alerts. An alert contains

information that can be used by CRIM to *instantiate* a LAMBDA model. An instantiated LAMBDA model is a model whose variables in the predicates are allocated. It corresponds to an event that has occurred.

A LAMBDA model is generally associated with an alert number. This number corresponds to an alert that can be raised by an agent in the system. When it is raised, the alert can be captured by CRIM and its metainformation extracted to instantiate a corresponding alert. When two alerts corresponding to correlated LAMBDA models are raised, CRIM checks with the mgu θ that their instantiated variables are coherent. If it is the case, the two alerts are part of the same ongoing scenario. If not, CRIM considers two scenarios occurring in parallel: it may correspond to two attackers performing independent attacks, for example.

3.2.4 Advantages of this approach

The first major reason for using LAMBDA/CRIM is its correlation engine. For an expert, not having to generate complete scenarios by themselves like they would do with attack of fault trees is a major time-saver. Indeed, when the topology of the event model does not match the one of the system, it can be extremely tedious to update a model after even the slightest change in the system. These situations actually often result in a new modelling from scratch. By defining atomic events and letting CRIM do the rest of the correlation, experts can be much more efficient. Moreover, it provides a common language for safety, security, and system experts to interact. Having several specialities that work on the same complex model can be challenging and using LAMBDA/CRIM can simplify that.

Another reason is the instantiation. With it, CRIM is able to track the occurring of events in the system. It provides us with a photography of the system at any given time, which is fundamental for an on-line use. And as seen previously, this is an important aspect of diagnosis.

Finally, LAMBDA/CRIM is adaptable to our problem. To perform the correlation and the instantiation, CRIM uses the preconditions and postconditions, and an alert number of a LAMBDA model. LAMBDA is a more complex language but the aforementioned parts of the language are sufficient to exploit the capacities of CRIM, and are relevant to model safety and security events instead of just attacks.

At this point, we have the tools to model and track the realisation of events in a system, but we need to reason about the probabilities and the time of realisation of the events in order to obtain important metrics for diagnosis. This is the object of the next section.

3.3 Preliminaries: Probabilistic theory

Thanks to LAMBDA and CRIM, we are able to generate complete scenarios out of atomic events. We now need to obtain a probabilistic evaluation about the realisation of events. Once again, using atomic events is a huge benefit: it is much easier for an expert to provide accurate information about atomic events than a combination of several of them. We therefore exploit this by providing a means to associate atomic events with this probabilistic measure and then establish the mathematical tools to propagate this measure to complete scenarios.

3.3.1 Atomically modelling the time

Risk analysis models often provide with a means to provide information about risk. Fault trees, for example, associate each failure with a failure rate. Some versions of attack trees do the same. Petri nets have a similar technique to probabilistically represent the failure of a component. However, these approaches of using failure rates are incompatible with diagnosis: the failure rates are established for a predetermined timespan. For a live process such as diagnosis, these timespans represent a problem: what happens when we want to start diagnosing in the middle of a timespan? We have adopted a different approach to overcome this problem. But these risk analysis methods tell us that experts are used to provide failure rates in their models and the solution we propose remains familiar with their usual work processes.

Instead of a failure rate, we associate a probabilistic distribution function (PDF) with each event. More formally, each event is associated with a continuous random variable that represents the time at which the event is realised. This random variable is described by a PDF that is to be determined by the experts modelling the system. There is no limitations on the PDF that can be used: they can be standard ones (gaussian, exponential, etc) or totally custom ones. We discuss about the choice of the PDF in section 3.3.4 but let us focus on the mathematical theory for now.

Using continuous random variables instead of failure rates gives us more flexibility and allows us for richer computations. Indeed, we have mathematical functions that we can evaluate, integrate, combine with each other in order to obtain information. The computations are not done through stochastic processes such as Monte-Carlo simulations, but provide exact mathematical results which are obviously more accurate.

Having these functions for each event is the first step but we need to obtain the PDF for any combination of events, including complete scenarios, for them to be practical. This is the subject of the following paragraphs.

3.3.2 Propagating the probability

Once the logical dependencies have been established between the different events and a scenario has been selected, its PDF can be computed. To do so, the local PDF associated with each event needs to be recombined in order to express the global PDF of the scenario. There exist three different situations: the sequence, the AND, and the OR.

On a side note, we will assume that the random variables associated with two events are always independent. It means that the time taken to realise one event has no influence on the time taken to realise another event.

The Sequence

Let us consider that events A and B happen in sequence. C corresponds to "A happens then B happens". C is not a LAMBDA event. It is just an abstraction used to compute the PDF associated to a set of events. It has no meaning outside of the scope of the probabilistic calculations. A, B, and C are respectively associated with the PDF f_A, f_B, and f_C. The probability that C happens in a given timespan $\mathcal{I}$ given that nothing has happened yet is actually the probability that both A and B happen in $\mathcal{I}$. This can be written:

$$\begin{aligned} f_C : \mathbb{R} &\to \mathbb{R} \\ x &\mapsto \int_{-\infty}^{+\infty} f_A(t) f_B(x-t) dt \end{aligned}$$

In other words, f_C is the convolution of f_A and f_B: $f_C = f_A * f_B$. Figure 3.1 represents the equivalent situation.

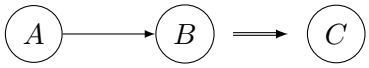


Figure 3.1: Events A and B happening in sequence and their equivalence C

This can also be expressed in a more "probabilistic" way. C is the sum of A and B: the time necessary for C to be realised is the sum of the times for A and B to be realised. The PDF associated with the sum of two independent random variables is the convolution of their respective PDF. A and B being independent, hence the result.

The AND

Let us consider that both events A and B need to happen for C to happen. A and B are respectively associated with the PDF f_A and f_B, the CDF F_A and F_B (the cumulative distribution

function evaluated at x is the probability that the event will be realised at x), and with random variables X and Y. The probability that A and B happen in a given timespan $\mathcal{I}$ can therefore be written $P(X \in \mathcal{I}, Y \in \mathcal{I})$. X and Y being independent, we have:

$$P(X \in \mathcal{I}, Y \in \mathcal{I}) = P(X \in \mathcal{I})P(Y \in \mathcal{I})$$

If we call g the PDF associated with the event "A and B has happened", we obtain:

$$g = f_A F_B + F_A f_B$$

Figure 3.2 represents the equivalent situation.

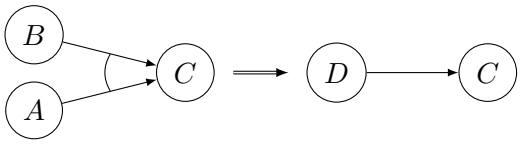


Figure 3.2: An AND gate and its equivalence

The OR

The case of the OR is similar to the one of the AND, with the difference that the AND is the set intersection when the OR is the set union. The resulting PDF, using the same notations as the AND case is:

$$g = f_A + f_B - f_A F_B - F_A f_B$$

Figure 3.3 represents the equivalent event.

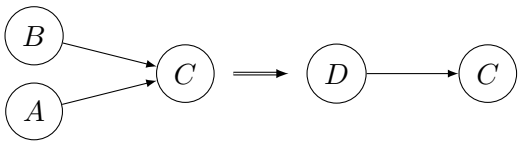


Figure 3.3: An OR gate and its equivalence

The nature of the calculus used to obtain the PDF corresponding to the three situations makes it so that they can be done in any order. For instance, the PDF associated with $(A\,or\,B)\,or\,C$ is the same as $A\,or\,(B\,or\,C)$.

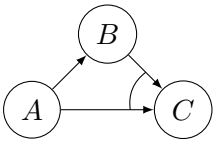


Figure 3.4: Situation output by CRIM

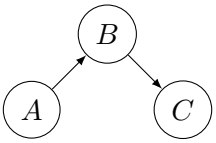


Figure 3.5: Equivalence of Figure 3.4

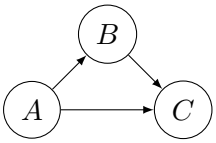


Figure 3.6: Situation output by CRIM

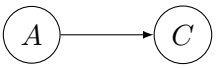


Figure 3.7: Equivalence of Figure 3.6

3.3.3 Probabilistic Equivalences

Depending on the pre/postconditions in the events, CRIM might output a graph such as in figure 3.4 where an event C can only happen after both B and A have happened, and B can only happen after A has happened.

In terms of order of occurrence of the events, this situation is equivalent to the one represented in figure 3.5, where C can only happen after B that can also only happen after A. The situation is then a simple sequence that can be computed using the formulas of paragraph 3.3.2.

Another situation is represented in figure 3.6 where an event C can only happen after either B or A has happened, and B can only happen after A has happened.

The occurrence of B has no consequence whatsoever on C and therefore can be removed from the probabilistic calculations, as represented in the equivalent graph in figure 3.7.

These equivalences are only to be used in the probabilistic calculations of the scenarios. To have the best understanding of the logical dependencies between the events, the event graph should be kept as is.

3.3.4 Obtaining relevant values for the distributions

Having a mathematical model is one thing but it being practical is another one. The strength of our theory revolves around the probabilistic distribution functions and it is therefore legitimate to wonder how convenient it is to provide these PDF. This is the discussion of this paragraph.

Establishing a PDF for an event might seem like a complex task when it actually is not. Most standard PDF are defined using parameters. These parameters have a "physical" meaning. The exponential distribution is defined by its parameter λ which is actually the inverse of its expected value. The normal distribution (or Gaussian) is defined by its expected value μ and its variance σ^2. Establishing the PDF for a given event is actually answering two questions: what

is the most appropriate distribution and what are its parameters.

Concerning the distribution to use, it depends generally on the nature of the event. For example, a Weibull distribution might be well suited for mechanical failures when an electromagnetic failure would correspond to an exponential distribution. Indeed, a mechanical failure happens after the wear of a component and would tend to be more common during a specific timeframe. Conversely, an electromagnetic failure would be more unpredictable and not based on the wear of the component: the exponential distribution being memoryless, it would seem to be more suited for that sort of failure. Likewise, concerning attacks, knowing that an attacker has not performed an attack yet does not give information about its future realisation: the exponential distribution is once again a good candidate. Using PDF to model failures being a rather unusual method, the literature is scarce on the matter but some studies can still be found [114]. In the end, it remains the choice of the expert who models the event but empirical studies and experience allow to narrow down the realistic possibilities.

Parameters are statistical measures. As such, they can easily be obtained empirically but this stresses out a great divide between safety and security. It is quite easy to obtain reliable values for failures and accidents since the return on experience is important for them. This is most of the time how the failure rates for usual risk analysis methods are determined. Getting the parameters for attacks is a much trickier problem because there are much more parameters to consider. For example, the level of the attackers, their means, or their motives are as many factors that influence the time necessary for an attack to be performed. Studies such as REF have been conducted to provide with a reliable way to estimate these parameters but the literature is not mature yet. It is however encouraging to see that this problem has been identified and that efforts are being made to solve it. Nevertheless, for now, estimating parameters for attacks is mostly an arbitrary decision by the security expert.

3.3.5 Recap

In this section, we have presented a probabilistic theory that can be used to model the time necessary for an event to occur. We have established the practicability of this method that calls on knowledge already used by risk analysis methods. It requires an expert to associate a probability distribution function with each atomic event and the model would take care of the recombination to provide PDF for composition of events, including complete scenarios. The next section describes PROS^{2}E, the event model built with this probabilistic theory and the logical one of section 3.2.

3.4 PROS^{2}E

In this section, we present PROS^{2}E, the event model that reuses the theoretical principles previously described in this chapter.

3.4.1 The event model

An event is characterised by its attributes: a set of preconditions, a set of postconditions, the nature of the event, a realisation process and a detection process.

The **preconditions set** and the **postconditions set** are inherited from LAMBDA. They are composed of predicates combined with the logical connectives $\wedge$, $\vee$ and $\neg$. The former set is used to describe the value that the variables of the system state must have for the event to be feasible. The later set is used to describe the value that the variables of the system state will have after the event has occurred.

The **nature** of the events is a label used to quickly know the type of the event: so far, we only used safety and security but any other type of events could potentially be added, such as countermeasures or regular events whose occurrence is part of the industrial process.

We define the **realisation process** as a probability distribution function (pdf). It is used to describe the evolution of the occurrence probability of the event over time, given that all of the conditions for the event to happen are met. Any pdf can be used, even custom ones. This is where the difference in modelling security and safety events lies in our model. Indeed, their realisation process is different and is represented, in this model through the pdf. With this approach, the model can therefore acknowledge for any kind of propagation: when modelling a specific event, one simply has to use its most appropriate pdf.

The **detection process** is used to link the modelled event to an alert $\mathcal{A}$ collected by the SCADA or SIEM system. It is used to inform the model that the event has been realised in order to update the system state and trigger instantiation of alert $\mathcal{A}$.

Figure 3.8 gives the model of the failure of a hard-drive. The pdf associated with the event is a Weibull distribution. The distributions and their parameters are chosen by the experts modelling the system, but can be derived from sizeable return on experience. For instance, when it comes to hard-drive failures, [114] has determined that the Weibull distribution is much more suited than the exponential distribution. Modelling this event with a pdf allows us to compute the mean time to failure of the event (5 years) or the probability of failure after ninety days (7.52×10^{-9}) for instance.

After several atomic events have been modelled, CRIM can be used to obtain complete scenarios and display them in an event graph such as the one of figure 3.9, taken from the use case of the following section.

Name	Hard-drive failure
Preconditions	$\neg failed(HardDrive)$
Postconditions	$failed(HardDrive)$
Nature	Safety
Realisation	Weibull distribution ($lambda = 5.516$, $k = 4$)
Detection	Server operating system raises alert

Figure 3.8: Hard-drive failure

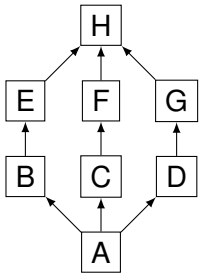


Figure 3.9: Scenario output by CRIM

3.4.2 Obtainable information

Given an event, or a set of events, associated with the random variable X and the pdf f, the **Mean Time To Failure**, **Mean Time To Success** or **Mean Time To Realisation** of this event is the expected value of X:

$$E[X] = \int_{-\infty}^{+\infty} xf(x)dx$$

Given an event, or a set of events, associated with the random variable X and the pdf f, the **Success probability** between t_0 and t is the difference of the cdf between these two times:

$$P(X \in [t_0, t_1]) = F(t_0) - F(t_1) = \int_{t_0}^{t_1} f(x)dx$$

In practice, t_0 is generally the current time when the calculation is done.

The **likelihood** of an event, or a set of events is computed the same way as the Success probability, but with $t_0 = -\infty$.

Cut sets can be computed with the topology of the event graph output by CRIM. They can stress minimal sets of events to closely monitor, as well as reduce computation time by reasoning on a smaller subset of events.

3.5 An example on how to use PROS^{2}E

In this section we give a more practical sense of the model and metrics we have previously discussed. We make use of a study case to illustrate more concretely how PROS^{2}E and its metrics can be used to provide an explanation when several events occur in a system.

3.5.1 Taum Sauk Hydroelectric Power Station

The case study is based on an actual power station: Taum Sauk Hydroelectric Power Station. An accident occurred there in 2005 that led to the destruction of a part of the power station [101, 115, 59]. This accident was purely due to safety issues but could have possibly been triggered by an attacker. The Taum Sauk Hydroelectric Power Station has previously been used to showcase the use of BDMP on scenarios that mix safety and security [66].

The Taum Sauk power station is primarily composed of an upper reservoir, a lower reservoir, a pump and a turbine. The reservoirs are filled with water. During low energy demand periods, typically at night, the system pumps water from the lower reservoir to the upper one, storing potential energy. Reciprocally, during high energy demand periods, water is allowed to flow back from the upper reservoir to the lower one, driving the turbine and producing electricity.

Safety mechanisms were designed to prevent the upper reservoir from overflowing, but they failed in December 2005 when water overtopped the upper reservoir, eroding the relief upon which it was built and subsequently causing a massive breach in the dam.

The system architecture supposed to prevent the upper reservoir from overflowing is the following. Three Druck pressure transducers, two Warrick sensors and two PLCs (the Common PLC and the Upper Reservoir PLC). The Druck pressure transducers convert pressure into water level. The three values output by the Druck transducers are transmitted to both PLCs, are averaged and then used to trigger a normal smooth automatic shut down of the operations if needed. If for whatever reason the operations are not shut down and the water level continues to rise, the Warrick sensors are activated. They send a signal to the PLCs (each sensor is connected to a different PLC) to trigger a hard emergency stop.

3.5.2 The scenario

Following the methodology of our model, in order to get scenarios, we first describe all of the elementary attacks or failures. We present two elementary events with figures 3.10 and 3.11: an intruder gaining access to the OCNet and an attacker compromising the communication link between the Common PLC and the Pump. Data about all of the events are summarized in table 3.1.

Name	**Access Operator Control Network**
Preconditions	encryption(OC_Net, null)
Postconditions	remoteAccess(A, OC_Net)
Nature	attack
Realisation	exponential distribution $(1/\lambda = 3years)$
Detection	IDS detects unknown IP address

Figure 3.10: Access to the Operator Control Network

Name	**Compromise Common PLC communication link**
Preconditions	remoteAccess(A, OC_Net) & vulnerable(Common_PLC, cve-2004-1234)
Postconditions	manInTheMiddle(A, Common_PLC, Pump)
Nature	attack
Realisation	exponential distribution $(1/\lambda = 10min)$
Detection	IDS detects ARP spoofing

Figure 3.11: Compromising of the communication link

Ref	Description	PDF	Nature
A_1	Access OC net	Exponential($1/\lambda = 3y$)	Attack
A_2	Compromise Common_PLC communication link	Exponential($1/\lambda = 8min$)	Attack
A_3	Compromise UR_PLC communication link	Exponential($1/\lambda = 8min$)	Attack
A_4	Compromise OCC communication link	Exponential($1/\lambda = 10min$)	Attack
A_5	Send false order (Common_PLC)	Exponential($1/\lambda = 5min$)	Attack
A_6	Send false order (UR_PLC)	Exponential($1/\lambda = 5min$)	Attack
A_7	Send false order (OCC)	Exponential($1/\lambda = 7min$)	Attack
A_8	Link with pump compromised	Dirac δ_0	Attack
B_1	Warrick_2 failure	Exponential($1/\lambda = 5y$)	Accident
B_2	Common_PLC failure	Exponential($1/\lambda = 15y$)	Accident
B_3	Drucks failure	Exponential($1/\lambda = 5y$)	Accident
B_4	UR_PLC failure	Exponential($1/\lambda = 15y$)	Accident
B_5	Warrick_1 failure	Exponential($1/\lambda = 5y$)	Accident
B_6	Common_PLC and sensors failure	Dirac δ_0	Accident
B_7	Both PLC failure	Dirac δ_0	Accident
B_8	Water level sensor failure	Dirac δ_0	Accident
B_9	UR_PLC and sensors failure	Dirac δ_0	Accident
O	Pump does not stop	Dirac δ_0	Objective

Table 3.1: Events in the graph

After defining all of the events, they are fed to the correlation engine that outputs scenarios. Several scenarios can lead to the failure of the upper reservoir. For the sake of providing a clear and concise example, we have selected one displayed on figure 3.12. It is focused on overtopping the upper reservoir. It can be done by an attacker intercepting all stop orders sent to the pump while it is active. Three sources can produce the stop orders : the Common PLC, the Upper Reservoir PLC and the Operator Control Center. All of the orders can be intercepted if one compromises the communication links between the order sources and the pump. Finally,

all of the communication links can be compromised if one has access to the Operator Control Network. For reference, all of the probability distributions are chosen exponential, except for the event “Pump does not stop” which is a Dirac delta function. The upper reservoir can also fail if the PLCs fail, if both the Druck and Warrick sensors fail, or if the Druck sensors and a combination of a Warrick sensor and a PLC fail.

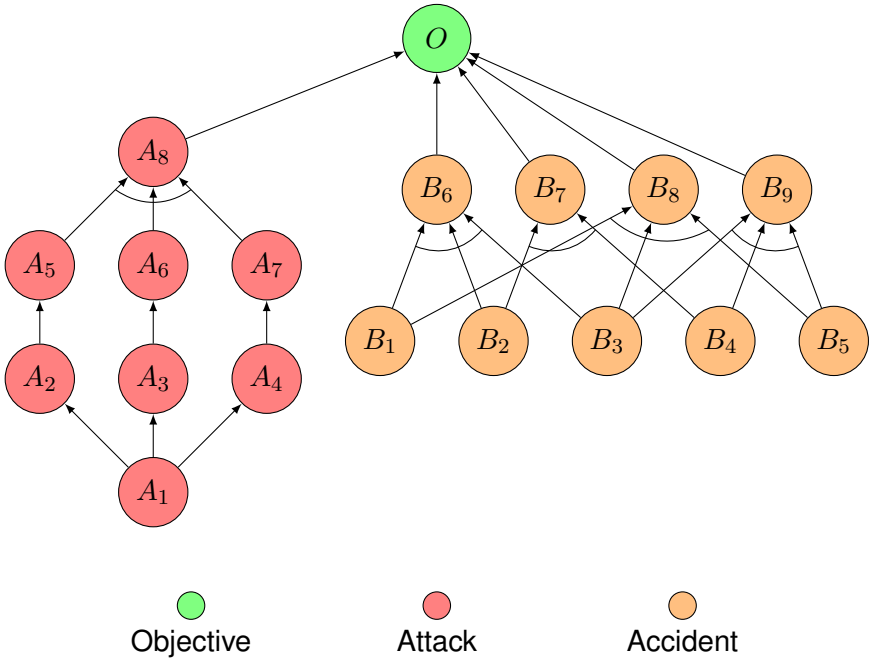


Figure 3.12: Scenario output by CRIM

Let us first consider a case where no alerts have been raised yet. Therefore, the considered graph is the one presented in Figure 3.12. By recombining the PDF, we obtain the expression of the distribution associated with the following event : “the pump stops given that no event has occurred yet”. We will call h the PDF associated with this event and H the cdf. Having h, one is now able to compute the MTTF of the specified undesired event: $mttf(h) = \int_{-\infty}^{+\infty} xh(x)dx = 3y23min27sec$. Using H, we can compute the probability that the event is realised after a given time. H is ploted on Figure 3.13 and we can read, for example, the probability of occurrence after one year $H(1y) = 28,3\%$, three years $H(3y) = 63.2\%$, or five years $H(5y) = 81.1\%$.

Now we consider a situation where the two PLCs have failed and the reservoir has been effectively overtoped. The SCADA system has issued an alert for each failure of a PLC as well as the overtopping of the reservoir. The alerts have been captured and processed by PROS^{2}E that can immediately identify the origin of the incident. Indeed, as displayed on graph 3.14

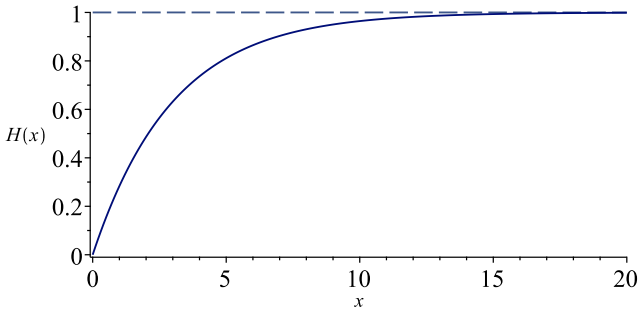


Figure 3.13: Evolution of the probabilities when no event has occurred

which corresponds to this situation, only safety events have been triggered and PROS^{2}E will say that the undesired incident is of accidental origin and can rebuild the sequence of events that led to it: the failure of a PLC, followed by the failure of the other PLC that lead to the failure of the upper reservoir.

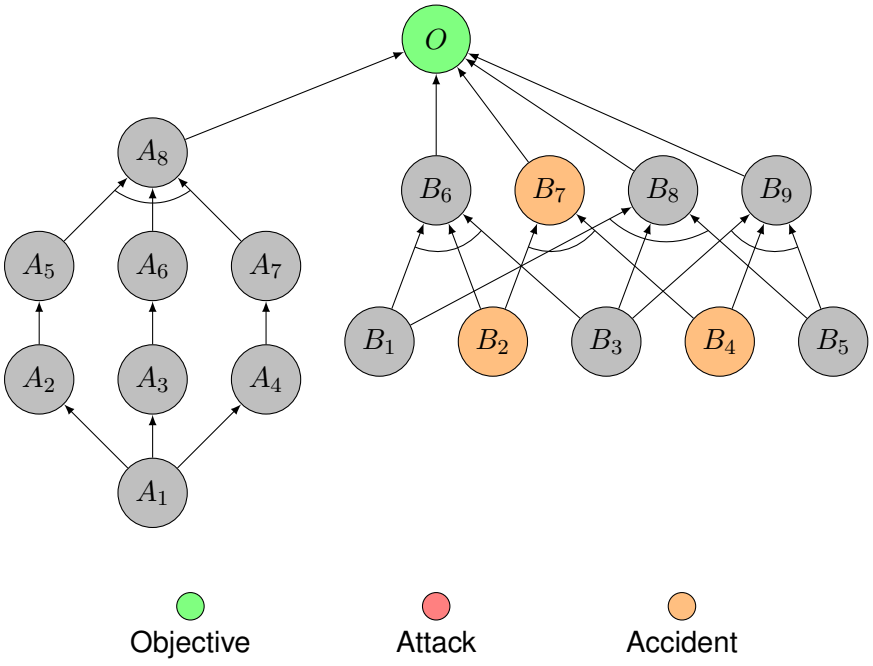


Figure 3.14: First case: accidental

In another situation, events A_1, A_2, A_3, A_4 and A_7 have occurred as displayed on figure 3.17. Each step that the attacker realises triggers an alert by an IDS that is captured by PROS^{2}E.

CRIM analyses the metadata of the alerts and correlates as they satisfy the most general unifiers corresponding to the dependency links. In that case, the undesired event has not occurred yet, but PROS^{2}E can still identify that the system is under attack due to the nature of the events that have occurred. PROS^{2}E can also provide various metrics required by a system administrator to take a decision as for the severity of the threat a response to it. The MTTF of O after event A_7 has been realised is $7min30sec$. The probability of failure after $5min$ is 40.0% or $10min$ is 74.8%. The evolution of the probability of occurrence of O given that A_1, A_2, A_3, A_4 and A_7 have happened is displayed in Figure 3.16.

The MTTF has been computed for various sets of alerts raised in Table 3.15. All of the computations were done using the software Maple and took around a second for each required value.

Case	Alerts raised	MTTS
1	$\emptyset$	3y 23min 27sec
2	A_1	23min 27sec
3	A_1, A_2	21min 13sec
4	A_1, A_2, A_5	20min 54sec
5	A_1, A_3, A_4	14min 54sec
6	A_1, A_2, A_3, A_4, A_7	7min 30sec

Figure 3.15: table
MTTS associated with various cases

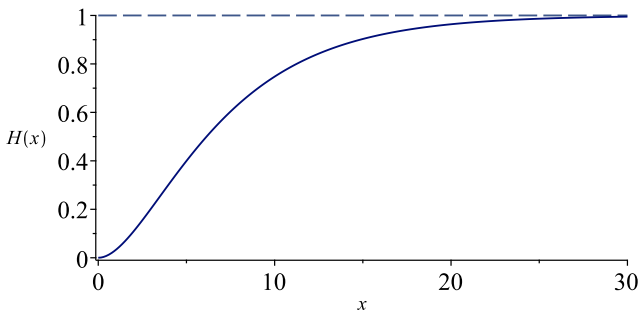


Figure 3.16: Evolution of the probabilities in case 6

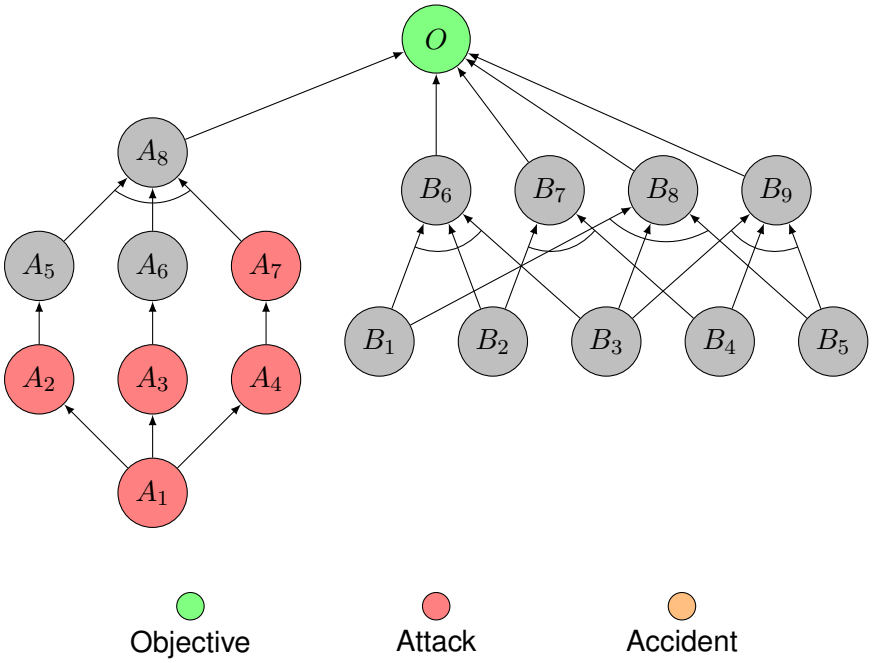


Figure 3.17: Second case: attack

If we consider a situation where all of the attack events have been realised, and events B_2 and B_5 have occurred, we may wonder if the overtoping was due to an attack, an accident, or both. If we look at the graph 3.18, we notice that the undesired event O has been triggered by the security events and not the safety one. PROS^{2}E is able to quickly identify the origin of the incident, even if alerts of different nature have been raised.

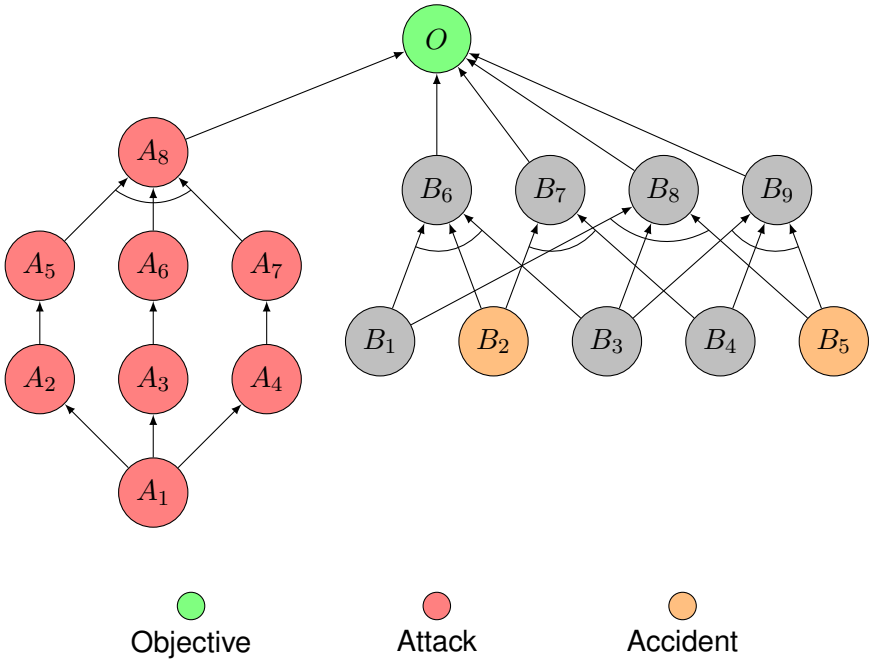


Figure 3.18: Third case: mixed

With several different cases over one scenario, we have shown that the model adapts to real-time scenarios: the probabilities of the undesired events evolve over the realisation of subsequent ones. This is important because, for example, the choices of the attacker may change which are the most probable scenarios. And the appropriate response may differ depending on what parts of the system are at risk. Safety and security supervisors obtain precious information from the model in the form of probabilistic evaluations of the evolution of the failure/compromise rate.

3.6 Conclusion

In this chapter we have presented PROS^{2}E, a framework to model and monitor incidents. We have illustrated how it can be used to perform diagnosis on a real-life study case: the Taum Sauk Hydroelectric Power Station. We have used some situations to showcase some of the capacities of PROS^{2}E.

Unfortunately, PROS^{2}E suffer from limitations. Some events can be triggered by the operators of the system in response to an ongoing incident. Such events will hinder the progression of the incident and need special probabilistic modeling. PROS^{2}E as presented in this chapter cannot capture this behaviour. The modeling of the time is also rather basic and need some refinement to express more complex situations, and provide more accurate metrics. The next two chapters are dedicated to these improvements.

All of the probabilistic computations presented in this chapter have been done using independent Maple spreadsheets but were not part of a monolithic solution. We have proven that it is technologically possible to perform the computations in real-time, and an integrated solution that correlates LAMBDA models, tracks incidents and delivers metrics in real-time can be developed. We regret not having been able to do it so far, but are still expecting to be able to work on it.

CHAPTER 4

HANDLING COUNTERMEASURES

4.1 Introducing countermeasures

In the version of PROS^{2}E presented in the previous Chapter 3, countermeasures are not included. This is identified as an issue since countermeasures can have side effects and therefore lead to undesired incidents. This Chapter is dedicated to their integration in PROS^{2}E. This process required to define what are countermeasures and how specific they are, which we discuss in the following paragraphs of this introduction. In Section 4.2 we provide a case study with countermeasure to motivate this contribution. Once countermeasures are defined, their probabilistic modelling became possible by providing a new probabilistic formula for their recombination with other events in order to follow the methodology developed in Chapter 3. This is the subject of Section 4.3. Section 4.4 is an application of the contributions of this Chapter on its study case. Section 4.5 is the conclusion of this Chapter.

4.1.1 Defining countermeasures

Countermeasures are actions taken to oppose an action, an effect, an event, or to prevent them. They can take many forms such as replacing a defective part, switching to a spare component, modifying the processing speed of a component to increase its life expectancy or compensate the failure of another component, etc.

One could argue that countermeasures are measures taken *after* an undesirable event have taken place and are outside of the scope of diagnosis. We argue that being able to measure the consequence of a countermeasure goes hand in hand with evaluating the severity of an incident: a serious incident with a good mitigating countermeasure is not necessarily as bad as a benign incident with no countermeasure. Moreover and this is particularly the case when mixing safety and security: countermeasures have side-effects that affect the system. And since safety and security are most of the time considered independently, it is not trivial to measure the impact a safety countermeasure will have on the security of the system, and vice versa. As such, countermeasures are integral part of scenarios and have to be modelled and taken into account in the probabilistic computations.

4.1.2 What makes countermeasures different from regular events?

Countermeasures are a particular type of events that are very different from regular attacks or accidents. First, the system do not endure them but they are voluntarily triggered. That means that when forecasting the outcome of a scenario, several different situations can be considered depending on whether the countermeasures are deployed or not.

Secondly, unlike regular events, countermeasures impede the realisation of a scenario. It means that if a countermeasures is deployed, it is possible that an objective is not reached.

From a probabilistic point of view, this poses a challenge. Indeed, with the current formulas, it is implied that the realisation of an event triggers the next one. With countermeasures, it is the opposite.

4.1.3 Countermeasures in the dependency model

Countermeasures have already been introduced in LAMBDA [33]. At the same time, *anti-correlation* was defined. To express this formally, let us consider two LAMBDA models A and B. Let $Post(A)$ and $Pre(B)$ respectively the postcondition set of A and precondition set of B. $Post(A)$ and $Pre(B)$ are sets of predicates. A and B are said to be anti-correlated if there exists predicates $not(pred_A)$ in $Post(A)$ and $pred_B$ in $Pre(B)$, or $pred_A$ in $Post(A)$ and $not(pred_B)$ in $Pre(B)$, such that $pred_A$ and $pred_B$ are unifiable through a most general unifier (mgu) θ.

Identifying countermeasures with anti-correlation is particularly interesting when safety and security interact. As we have discussed before, the interactions between safety and security sometimes lead to conflict between them. That is why an event A can be anti-correlated with a security event B and corelated with a safety event C. A is therefore a countermeasure in the scenario of B but an integral part of the scenario of C. Labelling events with a "countermeasure" tag is not sufficient to identify them in scenarios and anti-correlation provides just that.

Thanks to anti-correlation, we have a means to identify events that act as countermeasures in any given scenario. We still need to translate this behaviour in probabilistic formulas. This is the main contribution of this chapter.

4.1.4 Probabilistic representations of countermeasures in other models

The literature generally considers the cost of deploying countermeasures and not their effect on the probability the scenario's outcome [12, 118, 70]

In classical safety models such as fault trees of Petri nets, countermeasures are generally the repair of a component. To quickly summarize their computations, being done using Monte-Carlo simulation, they discretize a timespan in a certain number of intervals. At each interval they roll probabilities to know which components have failed and which have been repaired in that interval. This has the advantage to be able to look at very long timespans with several failures and repairs of components whereas our approach is more "single occurrence". However, and as stated earlier, Monte-Carlo simulations are not suited to diagnosis, and it is not possible to adapt their probabilistic handling of countermeasures to our model.

Kanoun *et al.* [63] used LAMBDA/CRIM to select countermeasures in a security context. They mapped a Markovian model on a scenario output by CRIM, associating exponentially distributed random variables to each attack, in order to compute the probability of occurrence of various outcomes, and select the countermeasure depending on those probabilities. This

approach was not satisfactory for the specificities of diagnosis because we need to be able to use more distribution functions than the exponential one, and because they only consider countermeasure that would hinder the objective of the attacker. In our approach, we want to consider situations when a countermeasure could prevent an attacker from going further a certain point but would not have any effect when the attacker is already beyond.

4.2 Motivating example

In order to illustrate the importance and the difficulty of correctly handling countermeasures, we describe a case study from which we extract a few situations.

It is based on a Fischertechnik-based CPS testbed presented in [46]. This test-bed consists of a virtualized IT network, three industrial PLCs Crouzet Millenium 3 XD26 and Fischertechnik actuators and sensors miniaturising an industrial process of machining parts. The network is organised in two networks. First, the IT networks consisting of a programming workstation used to program the controllers, and a supervision workstation that supervises the industrial process. Then the OT network containing the three controllers. The two networks are connected with a firewall. The IT network is connected to the internet through another firewall. The network diagram is represented on figure 4.2.

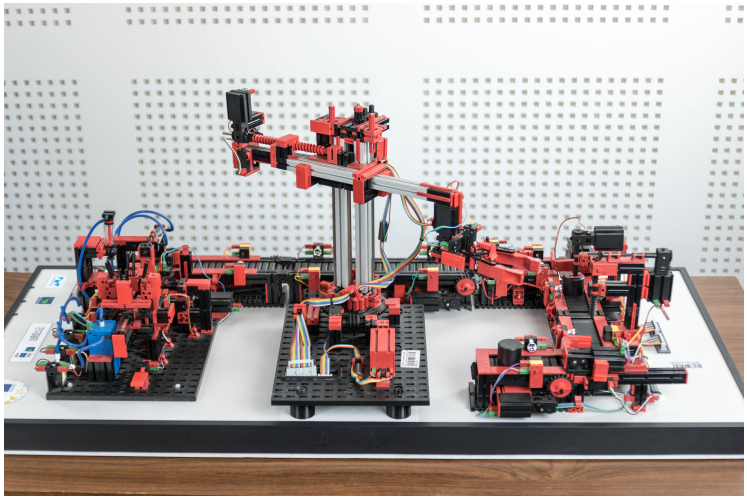


Figure 4.1: Fischertechnik-based CPS testbed

An attack developed for this testbed consists of an attacker tricking a legitimate employee into opening an infected spreadsheet that gives him a remote connexion to the PLC programming workstation. From there, the attacker can perform two actions. The first one is a Denial Of Service (DOS) on the network modules of the PLC using malformed ModBus packets. That

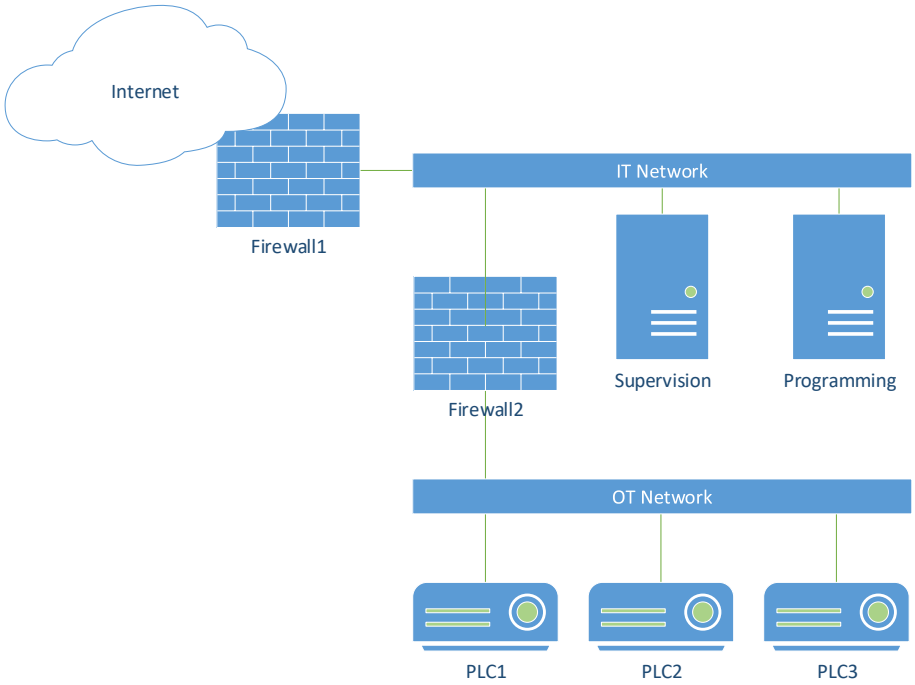


Figure 4.2: Fischertechnik-based CPS testbed network architecture

way, the industrial process keeps going, since the program in the controllers is not affected, but the supervision cannot contact the controllers anymore and no feedback can be collected by the supervision. For the second attack, the attacker sends orders to the PLC and disrupt the industrial process by opening the arm at an inappropriate time for example.

A system administrator is able to counteract on those attacks. The firewall is technically able to spot this suspicious connection from the PLC programming workstation to an unknown IP address and potentially reset it. The administrator could call for the reset of the connexion in order to stop the attack and preserve the integrity of the industrial system. A reset of the connexion would eventually stop the DOS attack and give back access to the supervision. For the second attack, however, a reset after the attacker has already disrupted the industrial process would not cancel the consequences.

To model the entirety of the scenario, we need the capacity to model the countermeasure. Standard metrics such as MTTS or success probability are required for the diagnosis and means to compute them has to be provided. The following section is about modifying PROS^{2}E in order to address this issue.

4.3 Probabilistic modelling of countermeasures

4.3.1 Computing probabilities with countermeasures, splitting the cases

When expressing a probabilistic computation, it is essential to precisely understand what is being computed. The human language is extremely dense in unsaid and implied things that are nevertheless necessary to be made explicit in the probabilistic problem. This is particularly the case in the MTTS.

The Mean Time To Success is the average time necessary for a set of events to occur. When computing it, it is implied that the event will indeed happen. The mean time of something that can't happen does not make sense. Probabilistically, the mean time is computed using the expected value which weighs all of the cases to output a result. It is important that it only considers situations in which the event happens. A countermeasure generates cases in which the considered event will not happen. It should therefore never be considered in the probabilistic computations of a MTTS.

On the other hand, the success probability represents the chance that an event will occur. A countermeasure will obviously lower this success probability and, thus, have to be considered in the computation.

In a nutshell, computing metrics for the same scenario can result in several different situations being considered. It is important to know what is implied and which events are relevant for the metrics computed. As a consequence, a single mathematical formula cannot compute all of the metrics but we need several formulas for several metrics. This does not invalidate the probabilistic theory developed in chapter 3, on the contrary, all of the formulas are still valid. But it is necessary to have in mind that a probabilistic model always has a "given that something" and this "something" can lead to errors and mistakes if not understood.

In the next section, we develop the case when a countermeasure has to be considered, for a success probability for instance, and the formulas that need to be used in order to recombine several events into one, in the same principle of chapter 3.

4.3.2 Simplifying the situations

Let us consider a single countermeasure for the moment. A countermeasure is going to be activated by one or more events, and will prevent one or more events from being realisable. Figure 4.3 represents this situation. This situation is similar to the one of our motivating example with an event providing an attacker with a remote access necessary for several events to occur, and cancelled by a countermeasure.

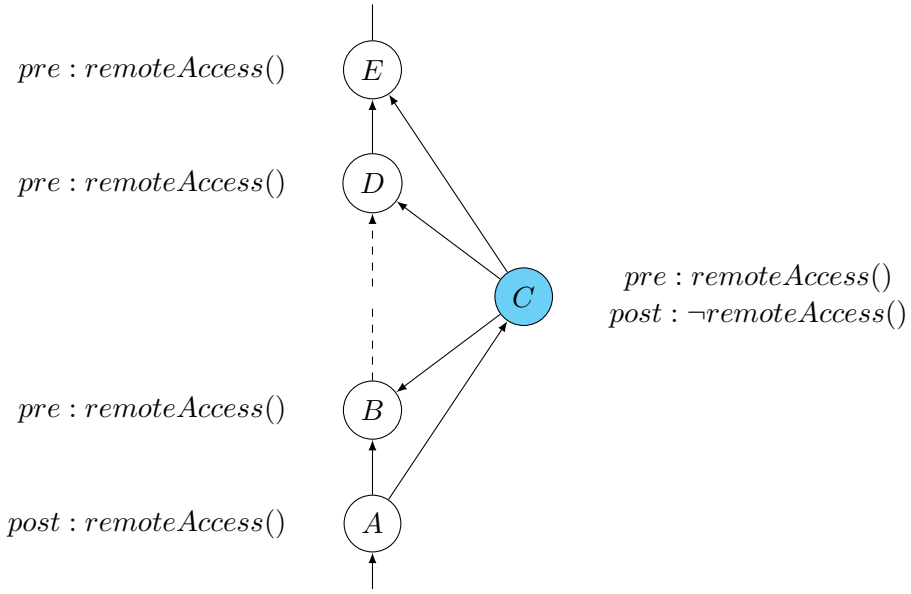


Figure 4.3: Countermeasure situation

4.3.3 Probabilistic expression

In figure 4.4, event A represents the first event that can trigger the countermeasure C. Event E is the last event affected by countermeasure C. It is important to consider the first event that can trigger the countermeasure because it is the one that can start the timer for the deployment of the countermeasure. Several events will be affected by the countermeasure and considering the last one, meaning the deepest one in the scenario, is important because any event that leads to this one will have no consequence on the general outcome if the countermeasure is deployed. Using the recombination formulas of chapter 3, we obtain a meta-event E' which we can use to obtain another meta-event C'.

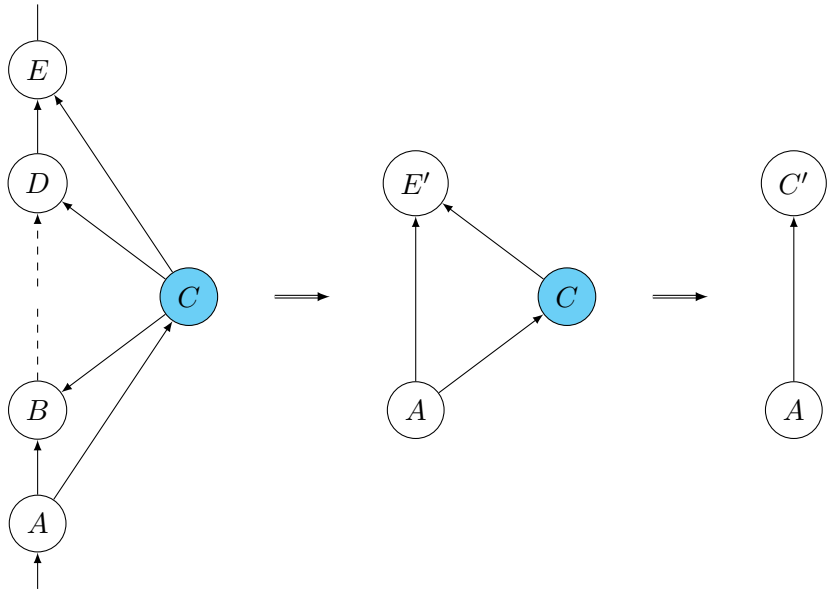


Figure 4.4: Countermeasure situation and equivalences

Event C' corresponds to "E' occurs and C does not occur". Let us respectively associate events E', C and C' with random variables X, Y and Z, with PDF f, g and h, and cdf F, G and H. Probabilistically, given any real t, "E' occurs and C does not occur" translates into:

$$
\begin{aligned}
P(Z < t) &= P(X < t \cap Y > t) \\
P(Z < t) &= P(X < t)P(Y > t) \\
P(Z < t) &= P(X < t)(1 - P(Y < t)) \\
H(t) &= F(t) - F(t)G(t) \\
h(t) &= f(t) - f(t)G(t) - F(t)g(t)
\end{aligned}
$$

These formulas can be used to compute the success probability of an event given that a countermeasure is deployed. However it is important to note that H is not an actual CDF. By definition, a probability distribution function is a *measure* whose *mass* is 1. This means that the sum of the probability of all of the events is 1. In other words that the probability of occurrence of an event after an infinite time is 1. However, in the case of an event that can be hindered by a countermeasure, after an infinite time, both the event and the countermeasure have been deployed, resulting in a probability of success after an infinite time of 0: $H(\infty) = F(\infty) - F(\infty)G(\infty) = 1 - 1 \times 1 = 0$. If h were a probability distribution function, and therefore

H a cumulative distribution function, we would have $H(\infty) = 1$. This does not prevent us from using these formulas in order to compute the success probability, but it is another evidence that they should not be used recklessly for any probabilistic computation, expected values for MTTS in particular.

4.4 Application to the study case

In this section, we show how countermeasures are used to model scenarios in PROS^{2}E and how to compute metrics used for diagnosis.

4.4.1 Modelling events and obtaining the graph

After modelling each individual event, we obtain the scenario represented in figure 4.5. Data about the nodes of the graph is summarized in table 4.1. We provide a LAMBDA model in figure 4.6.

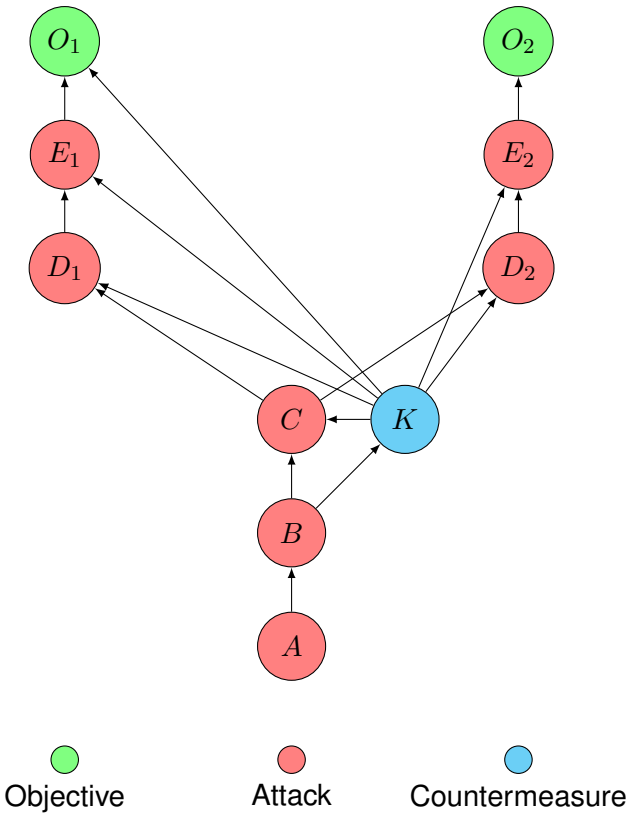


Figure 4.5: Scenario considered in this section

Ref	Description	PDF	Nature
A	Infected mail received	Exponential($1/\lambda = 1y$)	Attack
B	Infected mail opened; remote access established	Exponential($1/\lambda = 1h$)	Attack
C	ModBus scan	Exponential($1/\lambda = 5min$)	Attack
D_1	Malformed Modbus packet sent	Exponential($1/\lambda = 5min$)	Attack
E_1	Denial of Service	Dirac δ_0	Attack
O_1	Supervision can't reach PLC	Dirac δ_0	Objective
D_2	Download controllers programs	Exponential($1/\lambda = 5min$)	Attack
E_2	Send malicious order	Exponential($1/\lambda = 5min$)	Attack
O_2	Industrial process disturbed	Dirac δ_0	Objective
K	Reset connexion	Exponential($1/\lambda = 2min$)	Countermeasure

Table 4.1: Events in the graph

Name	Reset connexion
Preconditions	$remoteAccess(Attacker, ProgrammingWorkstation)$
Postconditions	$\neg remoteAccess(Attacker, ProgrammingWorkstation)$
Nature	Countermeasure
Realisation	Exponential distribution ($1/\lambda = 2min$)
Alert	Countermeasure #001

Figure 4.6: Model of the countermeasure of the scenario

4.4.2 Computing metrics

For the first situation, we consider that no event has taken place yet. The MTTS for the occurrence of O_1 is $1year$ and for O_2 is also $1year$. As presented in section 4.3, countermeasure K was not included in the computation of the MTTS. It has, however, an influence on the Success Probability after one year as shown in table 4.2.

Objective	O_1	O_2
Success Probability with K	1.14×10^{-7}	3.27×10^{-8}
Success Probability without K	0.632	0.632

Table 4.2: Success probabilities after one year

For the second situation, let us consider the case where events A, B and C have happened, as represented in Figure 4.7.

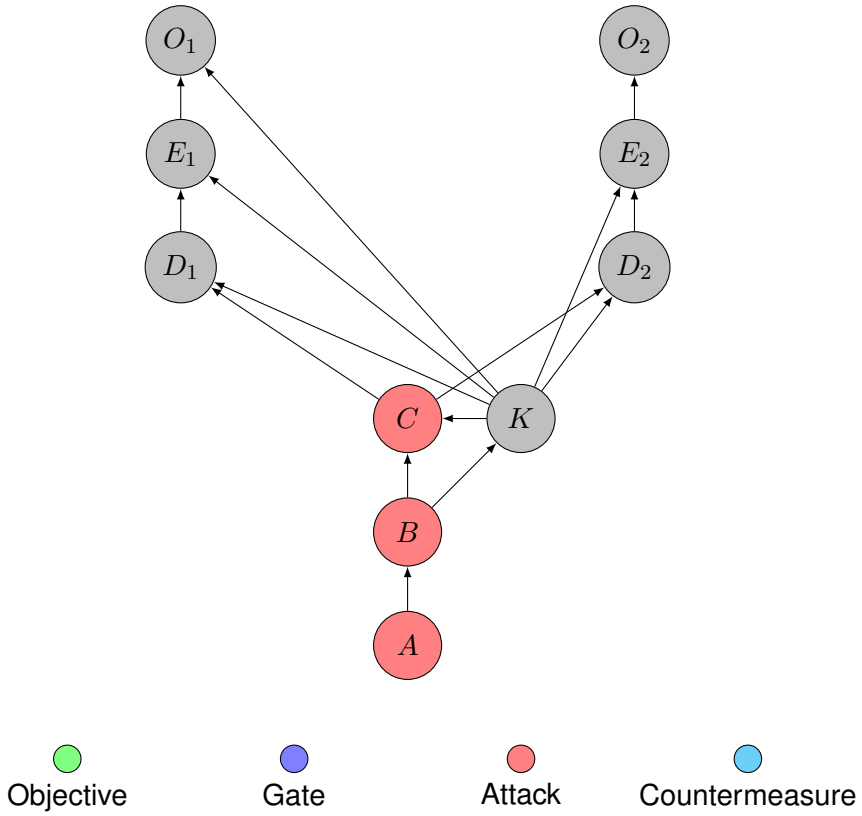


Figure 4.7: Scenario considered in this section

In that case, the MTTS for O_1 is $5min$ and for O_2 is $10min$. For both cases, the Success Probability after $7min30sec$ is shown in table 4.3.

Objective	O_1	O_2
Success Probability with K	0.0183	0.0104
Success Probability without K	0.777	0.442

Table 4.3: Success probabilities after one year

4.5 Conclusion

In this chapter, we have presented countermeasures, why they are important to consider in order to establish more complex and realistic scenarios, and how to integrate them to PROS^{2}E. We have illustrated this on a case study based on the Fischertechnik platform.

In terms of the limit of our approach, we concede that we have presented a very black and white situation: an event is either a countermeasure or it is not. This is perfectly fine to establish the probabilistic theory and give a precise idea of what is being considered, but in reality the situation is more complex. Some events might not be flagged as countermeasures per se, but will impede the realisation of other events. A denial of service attack, for example, will prevent another attacker from reaching a remote server. In order to know when to use the probabilistic formulas of a countermeasure, instead of using the nature of the event, the anticorrelation already present in CRIM will probably produce more suitable candidates. Some research is necessary to confirm this in order to obtain a more practical implementation. Nevertheless, this does not challenge the correctness of the formulas developed in this chapter, and the overall utility of including countermeasures in the model.

CHAPTER 5

IMPROVING PROS^{2}E WITH ENHANCED AND ACCURATE REPRESENTATION OF THE TIME

This Chapter is dedicated to another improvement of PROS^{2}E, this time adding improved sequentiality and more accurate time representation in order to achieve better modelling and diagnosis. We have identified that depending on the behaviour of the components, their wear might be incorrectly tracked and measured by PROS^{2}E. PROS^{2}E also suffers from an incapability to cover relatively usual situations due to the semantics of its logical gates that are not rich enough. In this Chapter, we begin to motivate these issues before solving them with rethinking the way the time is counted and with the addition of and SEQ-AND gate. In Section 5.1 we describe a case study used to justify the need for improvements in PROS^{2}E in Section 5.2. Sections 5.3 and 5.4 are dedicated to the modifications in PROS^{2}E, respectively for the sequentiality and the time representation. Section 5.5 applies those new capacities on the use case.

5.1 Motivating example

We use the Fischertecknik platform again in order to illustrate the need for further improvements on PROS^{2}E. These improvements concern time representation and are required to model more situations with better accuracy, and to avoid obtaining false numerical results that would lead to wrong diagnosis.

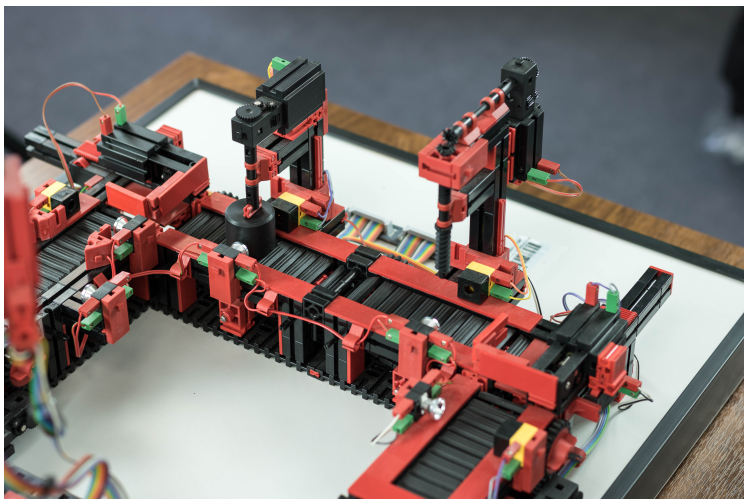


Figure 5.1: Milling operation in the Fischertechnik-based CPS testbed

The scenario we provide is focused on the milling operation. In this operation, a part to be processed is brought under a milling machine by a conveyor belt. Upon receiving the information from an optical sensor that a part is in position, one of the PLCs stops the conveyor belt and starts the milling process. After a few seconds, the milling stops and the part is conveyed

to the next operation.

For our scenario, we consider sets of events that can lead to a part being wrongly placed under the milling machine. This can obviously happen if a problem occurs with the sensor, by accident or intentionally. However, there are other situations in which that can happen. Indeed, there is a slight delay between the activation of the optical sensor and the effective stopping of the conveyor belt. This means that the part is not in position at the activation of the optical sensor and travels a short distance that depends on the speed of the conveyor belt. This distance can also be impacted if the conveyor belt is slippery. In case of a high required precision of the milling, it can have important consequences.

A wrong speed of the engine can happen for two reasons. The first reason is that the engine of the conveyor belt is malfunctioning due to its natural wear. Since it is continuously started and stopped, this requires to monitor its functioning time with accuracy. The second reason is when an attacker manages to modify the speed of the conveyor belt. This can be done by plugging an electrical circuit, to the power supply of the motor of the conveyor belt. An attacker can social engineer an operator of the industrial system to plug the electrical device and then modify the speed of the engine through radio frequencies such as GSM or LORA.

The conveyor can be stopped at the wrong time if either the optical sensors is malfunctioning, if the PLC is malfunctioning or if the attacker infiltrates the RFCOMM networks and sends wrong information to the PLC. However, according to its program, the PLC will only stop the conveyor belt upon the activation of the optical sensor if it has previously received the information that a switch has been activated. That switch corresponds to a previous actuator being back to its stand-by mode. Therefore, the attacker will actually need to send two pieces of information, in the correct order, to the PLC in order to successfully stop the conveyor belt.

The adherence of the conveyor belt can only be accidentally modified by oil spilling which cause is out of the scope of this scenario but always considered accidental.

5.1.1 First diagnosis example: processing new alerts

As a first example, we consider an attacker that has setup an RFCOMM antenna within range of the industrial system and the system raises an alert after detecting the presence of this antenna thanks to an unknown MAC address. The attacker then sends the messages of activation button Sw6 and photodiode Ph11, in that order. With these actions, the attacker manages to wrongfully place the part under the milling machine.

With this first situation, we can evaluate how a model manages to process new alerts raised by an IDS. These new events should orientate the diagnosis towards a pure security breach since all of the events are either attacks or direct consequences of an attack.

5.1.2 Second diagnosis example: Purely accidental?

For our second example, an attacker manages to damage a conveyor so that it breaks. The attacker is able to perform this attack without disrupting the milling process and therefore remain stealthy. This attack has not been anticipated by the experts and is therefore not part of the models they produced. This event has happened after the conveyor has been deployed for precisely four weeks.

This situation is used to illustrate the importance of accurate time modelling thanks to a failure probability of the engine that can be misleading if the functioning time is improperly measured. It is as well used to evaluate how a model behaves when it is confronted to a situation not anticipated by its designer, in this situation an attack, and how information can still be provided to avoid a wrong diagnosis.

5.2 Why PROS^{2}E needs improvement

Let us use the two situations to justify the need to improve PROS^{2}E.

5.2.1 Needing more sequentiality

In the example of Section 5.1.1, by successively sending activation messages of button Sw6 and photodiode Ph11, the attacker is able to stop the conveyor belt at an inappropriate time and therefore place the part in an incorrect position. The attacker has to send the message for Sw6, and only then the message Ph11 in order to achieve the attack. This seems to be naturally modelled in PROS^{2}E by two events happening in sequence as represented by figure 5.2. However, this means that the event *Message Ph11* could not happen before event *Message Sw6* has been sent, and specifically that the attacker could not start to send the second message before the alert of the first have been processed. Yet the attacker is capable of sending the *Message Ph11* at the same time as *Message Sw6*: using sequential situation would result in an *inaccurate estimation of the time and a wrong probability*.

It is then natural to try to use an AND gate to model this situation, as displayed on figure 5.3. But in this situation, receiving *Message Ph11* before *Message Sw6* would still trigger event *Part incorrectly placed* in the model even though it would not correspond to the success of the attack resulting in the *wrong identification of a situation* and an *incorrect probability of success*.

The only way to correctly model such dependency between events is to modify PROS^{2}E. In Chapter 2, we have discussed about several models that have similar issues and that have been improved to solve them. In particular, attack and fault tree are not able to model our example, but have been extended into dynamic attack trees [64] and dynamic fault trees [126] in order to address this issue. They use a new gate for this: the Sequential-AND (SEQ-AND)

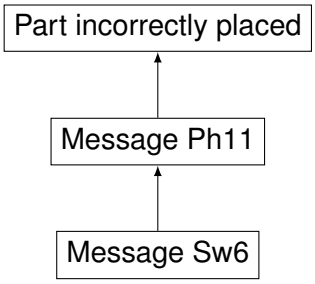


Figure 5.2: First incorrect model to incorrectly place a part

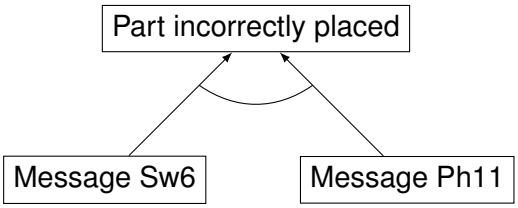


Figure 5.3: First incorrect model to incorrectly place a part

gate. It describes that two events have to happen in a given order for the gate to be fired. We have decided to incorporate a Sequential-AND gate into PROS^{2}E since it is a well-known and commonly used gate, whose signification is familiar to system experts.

5.2.2 Knowing when time is ticking or not

In the example of Section 5.1.2, we have a conveyor belt that is only started when it needs conveying a part. We will do the hypothesis that it cannot spontaneously fail upon starting or stopping: the probabilities of failure after ten continuous minutes of functioning and ten sessions of one minute are the same. Let us say that the conveyor belt works in cycles of one minute period during which it is on for the first twenty seconds and off for the remaining forty as represented in figure 5.4. The model associated with the failure of the conveyor belt is presented in figure 5.5.

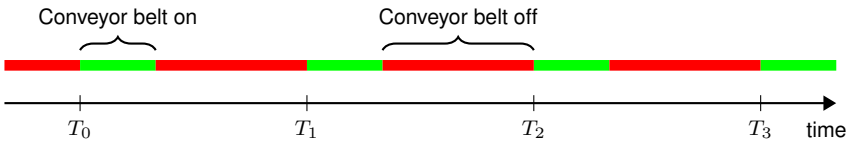


Figure 5.4: Timeline of the functioning of the conveyor belt

Name	**Conveyor belt failure**
Preconditions	$\neg failed(ConveyorBelt) \wedge$ $atWork(ConveyorBelt)$
Postconditions	$failed(ConveyorBelt) \wedge$ $\neg atWork(ConveyorBelt)$
Nature	Safety
Realisation	Weibull distribution ($\lambda = 5$, $k = 4$)
Alert	SCADA #003

Figure 5.5: Model of the conveyor belt failure

When the system starts the conveyor belt for the first time at T_0, the preconditions of the event **Conveyor belt failure** are met and the time is being counted. After twenty seconds, the predicate $atWork(ConveyorBelt)$ is not true anymore, the preconditions of the event are not met anymore and the event is not realisable anymore. This poses a problem concerning the time: how are the twenty seconds of period $[T_0, T_1]$ counted? Since this situation was not specified in the first version of PROS^{2}E, we could have a clock that never stops ticking resulting in the full minute of $[T_0, T_1]$ being counted, or a clock resetting each time the preconditions are met resulting of a maximum counted time of twenty seconds, regardless of the actual wear of the conveyor belt. Just to illustrate how problematic this situation is, we computed the probability of failure after nine years. The actual probability of failure is 7.772%, according to the probability distribution function specified, the one for a clock that never stops ticking is 99.86%, and the one for a clock that is constantly reset is $7.474 \times 10^{-24}\%$. This would obviously lead to *incorrect diagnosis* and *incorrect probability of failure*.

Rethinking and formalising the way time is being counted is necessary to address this issue. This example highlights the need for PROS^{2}E to have such feature, and we illustrate later in this chapter how important this is for diagnosis.

5.3 Modelling sequences

In order to overcome the issues previously identified, we modified the pre/postconditions and CRIM to support the *Sequential-AND*, we extended the nature of events, and we propose a more accurate counting of the time in order to capture the wear of components, **and we had to rework the probabilistic formulas**.

5.3.1 Correlation with a SEQ-AND

In the first situation described in paragraph 5.1.1, the messages of activation button Sw6 and photodiode Ph11 have to be sent in that order for the attack to work. If the messages are reversed, the attack does not work. In order not to consider impossible situations, and to accurately model possible ones, we introduce *Sequential-AND* gate (*SEQ-AND*) in addition to the regular *AND* and *OR* gates, taking example on the solution proposed to overcome this problem in fault trees and attack trees [41]. However, adding this SEQ-AND gate is not trivial since it requires to accurately model its semantics both from the event correlation layer of the model, and the probabilistic one.

CRIM [34] can be seen as a two-steps process: first the generation of the attack graph, and then the instantiation of alerts. The sequentiality of events is a temporal notion to be applied during the instantiation of alerts, but it has no impact on the generation of the graph other than displaying a specific logic gate.

In order to generate an attack graph, CRIM correlates events according to their predicates: two events are correlated if there exists a most general unifier to unify the preconditions of one event with the postconditions of the other[1]. These predicates are described using the language L_1 of LAMBDA [36]. In order to express the sequentiality of event, we propose to add a SEQ-AND operator $\not\rightarrow$ to this language: $expr_A \not\rightarrow expr_B$ describes that $expr_A$ and $expr_B$ must be verified but also that $expr_A$ must be verified before $expr_B$.

The SEQ-AND is a regular AND that adds a temporal condition. However, time does not take part of the generation of the attack graph. That is why, for the acquisition of unifiers between events, the SEQ-AND can be treated as a regular AND. However, the SEQ-AND must be apparent on the resulting event graph because it will be used for further operations such as the instantiation of alerts and probabilistic computations. Indeed, when two alerts associated with two events connected with a SEQ-AND are raised, CRIM has to check that the unifiers match, but also that the timestamps of the alerts are consistent with the temporality expressed by the SEQ-AND. Inconsistency noted on the unifiers or the temporality then results in uncorrelated events, from the point of view of the scenario.

5.3.2 Probabilistic expression of the SEQ-AND

The semantics of the SEQ-AND also need to be expressed in the probabilistic computations of the model because using either regular AND or pure sequentiality between such events will result in inaccurate results.

Let us consider the following situation: event O is gated after events A and B and we try to obtain a virtual event C corresponding to "A and B occur and A occurs before B". A, B and C

1. More information in Cuppens *et al.* (2002)[34]

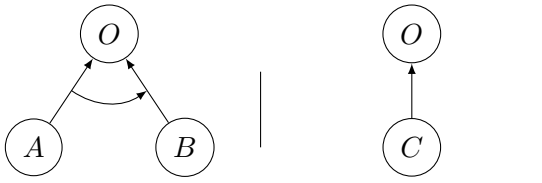


Figure 5.6: Recombination example

are respectively associated with random variables X, Y and Z, probability distribution functions f, g and h, and cumulative distribution functions F, G and H. The expressions of f and g are known and we want to get the expression of h. By definition of the SEQ-AND, and using Bayes Law, we obtain for any real time t:

$$\begin{aligned} H(t) = P(Z < t) &= P(X < t \cap Y < t \cap X < Y) \\ &= P(X < t | X < Y \cap Y < t) P(X < Y \cap Y < t) \\ &= P(X < Y | Y < t) P(Y < t) \\ &= P(X - Y < 0 | Y < t) G(t) \end{aligned}$$

The probability $P(X - Y < 0 | Y < t)$ is not known because we have an additional condition on Y. Indeed, we now know that event B will occur before time t. Therefore, $P(Y > t) = 0$ and g does not describe Y any more. We need a new probability distribution function for Y. We will discuss about this new function in the next paragraphs but for the moment, let us assume that we have it and denote it $\dot{g}_t$. Let us define:

$$\begin{aligned} \dot{g}_t^* : \mathbb{R} &\to \mathbb{R} \\ x &\mapsto \dot{g}_t(-x) \end{aligned}$$

Given that the probability distribution function of the sum of two variables in the convolution of their probability distribution functions, we obtain:

$$\begin{aligned} H(t) &= P(X - Y < 0 | Y < t) G(t) \\ &= G(t) \int_{-\infty}^{0} f * \dot{g}_t^*(x) dx \end{aligned}$$

We obtain an expression of H with no unknown elements provided that $\dot{g}_t$ is known. H can then be derived to obtain h. However, there exists no rule to obtain $\dot{g}_t$ from g. The possibility space of Y is modified and an assumption needs to be made to obtain $\dot{g}_t$. Indeed, let us take the

example of a hard drive. A hard drive can suffer from mechanical or electromagnetic failures. Let us assume that electromagnetic failures are accurately described by an exponential distribution when a mechanical failure is accurately described by a Weibull distribution [114]. The default distribution to model the time until failure is therefore a combination of both. We will also assume that the older the drive, the more prevalent mechanical failures are. The information that a fault is occurring on a recent drive will probably mean that the default distribution does not fit to model it, and the exponential one, corresponding to an electromagnetic failure, will be much more accurate. This is an example why an absolute transformation from g into $\dot{g}_t$ cannot be given. For the sake of this Chapter, we propose a way to obtain a $\dot{g}_t$ from g but we also stress that this does not fit every situation.

The assumption we make is that when it is known that Y is inferior to t, for any real t, the residual probability in g after t is equidistributed before t. We obtain the following expression for $\dot{g}_t$:

$$\dot{g}_t : \mathbb{R} \to \mathbb{R}$$
$$x \mapsto \begin{cases} \alpha g(x) & \text{if } x < t \\ 0 & \text{else} \end{cases} \text{, with } \alpha = \left(\int_{-\infty}^{t} g(x)dx \right)^{-1}$$

To better understand the transformation, figures 5.7 and 5.8 respectively plot g and $\dot{g}_t$.

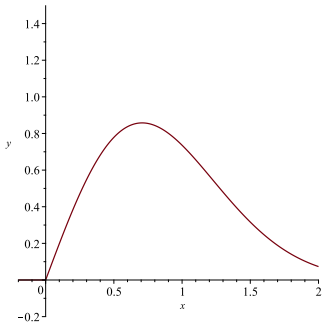


Figure 5.7: g

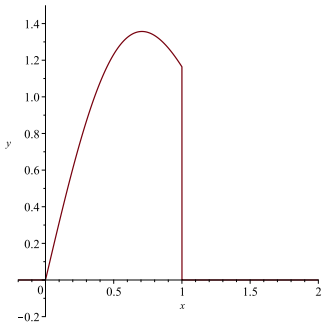


Figure 5.8: $\dot{g}$

As we saw in Chapter 4, the probabilistic expression of the problem depends on what is actually computed. This expression is the one to be used when computing the probability that an event will happen after a given amount of time. However, when computing a MTTS or a similar metrics, this is computed with the hypothesis that the event will indeed happen, and therefore that all of its conditions are verified. In the case of the SEQ-AND, that means that we *know* that A happens before B, and the formula becomes:

$$
\begin{aligned}
H(t) = P(Z < t) &= P(X < t \cap Y < t \cap X < Y | X < Y) \\
&= P(X < t \cap Y < t | X < Y) \\
&= P(Y < t | X < Y) \\
&= P(Y < t | X < Y) \\
&= \frac{P(X - Y < 0 | Y < t) G(t)}{P(X < Y)}
\end{aligned}
$$

In a nutshell, the probabilistic semantics of the SEQ-AND depends on the event being modelled. Indeed, it adds new information on the random variables that cannot have a generic expression. However, this information can be translated in new probability distribution functions, based on new assumptions. Therefore, the SEQ-AND can still be expressed from a probabilistic point of view and be integrated in PROS^{2}E.

5.3.3 Nature of events: not just safety and security

In the initial version of PROS^{2}E, the nature of events is simply a static label that has no particular behaviour. This is troublesome when an event can happen for various reasons. For example, the wrong speed of the conveyor belt can happen either because of a failure of the conveyor belt or because of an incorrect attack. To address this issue, we introduce *neutral* events. A neutral event is an event whose nature is to be determined by its children upon its instantiation: the nature of the children that activate the preconditions is transmitted to the neutral event. In the case of the incorrect rotation speed, the neutral event could become a *safety* or a *security* event depending on which child triggered it.

5.4 Managing the wear

Modelling repairable systems is a major concern for industrial experts. All the more for a model that is designed to be used in real-time: the enable, disable, or repair of a component is to be handled by the system. The probabilistic part of the model aims at providing with an accurate estimation of the time necessary for a subset of events to happen. The evaluation is based on failure and attack probabilities that are function of the time, described in the probability distribution function. The assumption made in the previous chapters is that the probability starts changing when the preconditions associated with the event are met. However, this does not correspond to an accurate description of many situations.

Let us consider the case of the conveyor belt: its failure only impacts the system when it is conveying a part, and the precondition associated would look like $atWork(ConveyorBelt) \wedge carried(Part, ConveyorBelt)$. This means that the probability would only start changing when it is both working and conveying. However, the belt only requires to be working in order to wear out and fail. Therefore, not considering the timespan in which the belt is working on but not conveying will result in an inaccurate description of its failure probability.

Another situation poorly handled in the previous chapters is when the conveyor belt is being started and stopped. At first it is up and can fail: its probability of failure evolves. Then it is switched off and therefore cannot fail. It is then eventually restarted when a part needs moving. After the restart and using CRIM, we shall have two possibilities: the time of origin of the event would either be reset and every wear that has occurred in the first span of functioning would be voided, or the clock would still have ticked when the component was stopped. Both situations would lead to errors in the time counted. Moreover, upon repair of a component, it is required that this reset happens. We therefore need to add to the model a means to express this reset of the probability.

In order to address the issues raised in the previous paragraph, we modify LAMBDA to add the concept of a local stopwatch to each event. This can be seen as a module that, if not used, does not hinder the previous functioning of the model. A stopwatch is basically three buttons: start, stop and reset.

The start button tells when the clock starts ticking and is implemented with the addition of a new *time-condition*, being a set of predicates expressed in the language L_1 just like preconditions and postconditions. The time-conditions express the state of the system in which the event can happen. However, the occurrence of the event will not have consequences on a possible scenario until the preconditions are met. The stop button corresponds to the time-conditions not being valid any more. Time-conditions are not to be correlated by CRIM.

The reset button describes the conditions upon which the counter is to be reset. It most likely corresponds to the repair of a component, that can be announced to the network just like its failure. It therefore corresponds to an alert that will, upon reception by the model, trigger the reset of the counter. In the model, it is described by a "Reset" attribute that is an alert number corresponding to an alert raised by the system. Figure 5.9 corresponds to the failure of the conveyor belt, modelled with the new elements presented in this section.

Figure 5.9 corresponds to the failure of the conveyor belt, modelled with the new elements presented in this section.

Name	Conveyor belt failure
Preconditions	$\neg failed(ConveyorBelt) \wedge$ $atWork(ConveyorBelt)$
Postconditions	$failed(ConveyorBelt) \wedge$ $\neg atWork(ConveyorBelt)$
Time-condition	$atWork(ConveyorBelt)$
Nature	Safety
Realisation	Weibull distribution ($\lambda = 5$, $k = 4$)
Alert	SCADA #003
Repair	SCADA #503

Figure 5.9: Model of the conveyor belt failure

5.5 Application to the study case

In this section, we illustrate the added modelling capacities on the use case described in Section 5.1, as well as showcase how it is possible to provide diagnosis of incidents using our PROS^{2}E.

5.5.1 Modelling events and obtaining the graph

After modelling each individual event and feeding them to CRIM, we obtain the scenario represented in figure 5.10. Data about the nodes of the graph is summarized in table 5.1. We provide a LAMBDA model in figure 5.9.

5.5.2 First diagnosis example: processing new alerts

We apply PROS^{2}E to the example of Section 5.1.1, event C7 is initially triggered. From this situation, we are able to obtain the PDF of the general scenario and therefore compute an MTTS of $7min29s$, or a probability of occurrence after fifteen minutes of 45%. The MTTS is computed as the expected value of the PDF and the probability is the evaluation of the cdf at the time considered.

During the second step of the scenario, an IDS associates the activation messages with the unknown antenna previously identified and raises an alert corresponding to events C5 and C6. The consequence on the model is the firing of events C2, C1 and O. Since O and C1 were only triggered by security events, they are themselves coloured as security events. The model is therefore able to provide the origin and the consequences of the incident.

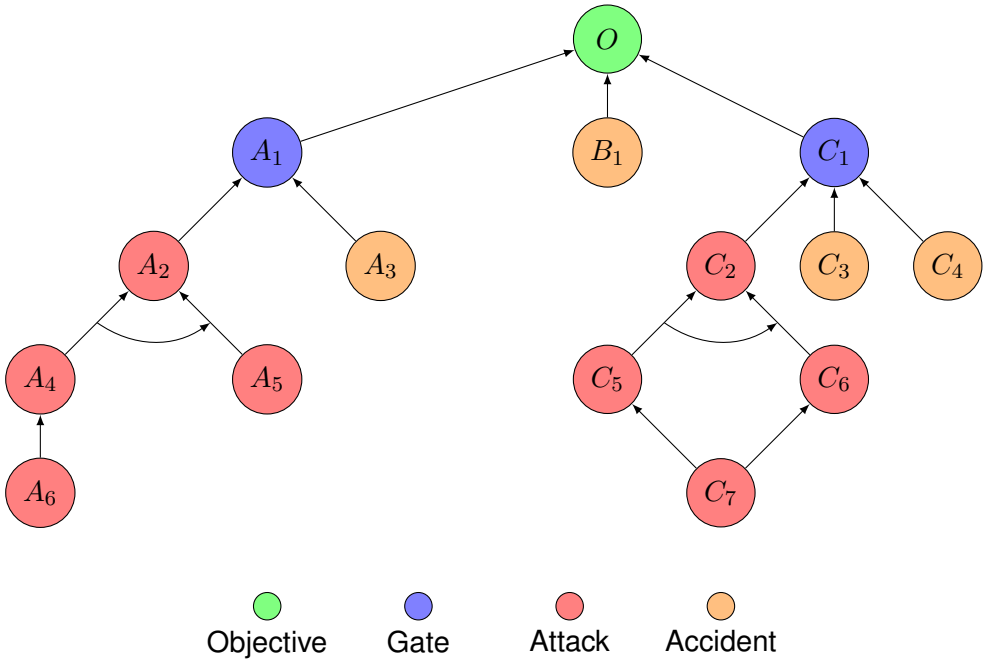


Figure 5.10: Scenario considered in this section

5.5.3 Second diagnosis example: Purely accidental?

In this situation, the only nodes activated in the graph are safety ones (A_3, A_1 and O), indicating a pure accident. However, PROS^{2}E can compute an important metrics in this situation: the likelihood. Indeed, it computes the probability that the engine fails during its first four weeks. Given that the engine is active 30% of the time, that corresponds to 1.2 weeks of functioning with a probability of failure of 0.331%. Due to such a low probability, the fact that the situation has been correctly identified by the model is extremely unlikely and this raises a lot of suspicion on this event and calls for a more thorough analysis of its origin. The diagnosis here is: accidental but highly suspicious.

This also demonstrates the importance of a correct measurement of the time. Indeed, without being able to properly monitor the wear of the conveyor belt, the likelihood evaluation would be way off its actual value : 7.47×10^{-24} corresponding to five seconds of wear for a clock that resets at every start of the engine. Even though 7.47×10^{-24} is still a value that would raise suspicion, it corresponds to a behaviour of the model that would raise suspicion for every failure of the conveyor belt, even "normal" ones. Therefore the importance of correctly tracking the system state and accurately measuring the time. Once again, the computations were done using the software Maple and took around a second on a desktop computer for each required value.

Ref	Description	PDF	Nature
O	Wrong part placement	Dirac δ_0	Objective
$A1$	Wrong conveyor belt speed	Dirac δ_0	Neutral
$A2$	Attacker decreases conveyor belt speed	Dirac δ_0	Attack
$A3$	Conveyor belt failure	Weibull($\lambda = 5w$, $k = 4$)	Accident
$A4$	Circuit set up	Exponential($1/\lambda = 2d$)	Attack
$A5$	Order sent to circuit	Exponential($1/\lambda = 1d$)	Attack
$A6$	Corruption of employee	Exponential($1/\lambda = 3w$)	Attack
$B1$	Bad grip on the conveyor belt	Exponential($1/\lambda = 10w$)	Accident
$C1$	Conveyor belt stopped at inappropriate time	Dirac δ_0	Neutral
$C2$	Attacker provokes stop	Dirac δ_0	Attack
$C3$	PLC failure	Exponential($1/\lambda = 10y$)	Accident
$C4$	Position sensor failure	Weibull($\lambda = 1w$, $k = 4$)	Accident
$C5$	Activation of button Sw6 by attacker	Exponential($1/\lambda = 5min$)	Attack
$C6$	Activation of photodiode Ph11 by attacker	Exponential($1/\lambda = 5min$)	Attack
$C7$	Attacker sets up RFCOM antenna	Exponential($1/\lambda = 3w$)	Attack

Table 5.1: Events in the graph

With these experiments, we illustrate that PROS^{2}E is able to model complex scenarios with elaborate sequentiality expressed in the relationships between the elements. This sequentiality is then translated into more accurate metrics such as MTTS or probability of occurrence. PROS^{2}E is able to provide information from past events such as identifying the nature of the situation or raising suspicion when relevant, but also to anticipate future events thanks to its graph as well as precious metrics that can be transmitted to system operators for them to take the best solution to the ongoing problem.

5.6 Conclusion

In this chapter, we have improved PROS^{2}E with advanced modelling and diagnosis capabilities. Being able to consider the sequential order of events is actually a necessity in most event models, such as fault trees, attack trees, or Bayesian networks. It broadens the range of scenarios that can be modelled and increases the accuracy of metrics that can be computed from them. We also have demonstrated that tracking the actual wear of a component is paramount

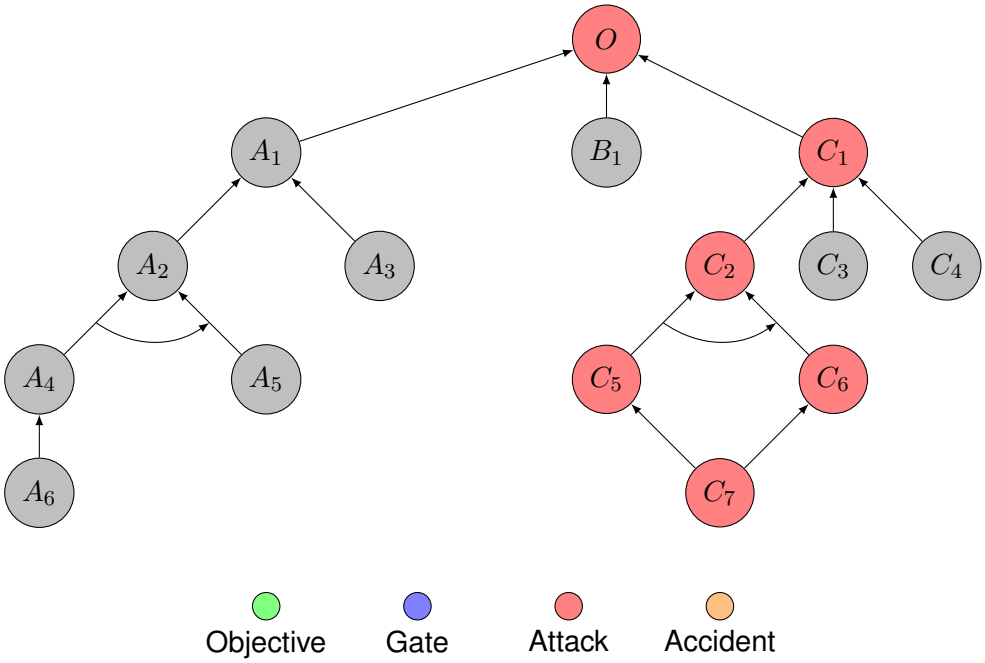


Figure 5.11: Aftermath of the attack

to obtaining relevant information when evaluating the likelihood of an event or forecasting future ones.

The trade-of for the addition of the SEQ-AND gate is more work being required when establishing the probability distribution function. An additional condition on the time of the incident sometimes calls for a completely different probability distribution function. Deciding on this is not trivial an can be a source of subjectivity. The computation is also heavier but, according to our experience in the subject, they remain very satisfactory.

We have displayed how PROS^{2}E can be used for diagnosis. Our use case is based on a research platform conceived with actual industrial components and allowed us to study the evolution of computed metrics and their inferred diagnosis with ongoing events.

CHAPTER 6

CONCLUSION

6.1 Contributions of this thesis

In this thesis, we have presented our contributions to diagnosis in cyber-physical systems.

In Chapter 1 we discuss about the whys and wherefores of diagnosis in a safety and security environment. We provide our own definition for diagnosis in this particular context. We establish that no diagnosis model exists for safety/security environment and instead identify the functionalities a diagnosis model should have in order to base the Chapter 2 on.

Chapter 2 is a review of several models able to provide information that can be used to perform diagnosis. The review is divided in three parts, focusing on models that can consider respectively only safety events, only security events, or both at the same time. This chapter was concluded by the establishment of a classification in order to identify more easily the functionalities of existing models.

In Chapter 3, we present PROS^{2}E, a framework to model and monitor incidents. It allows for metrics to be computed and information to be obtained in order to perform diagnosis. PROS^{2}E does not call for new knowledge to model the elementary events which is an important point for a practical use. Indeed, safety and security experts are used to modelling elementary events with fault trees or attack trees, for example, and estimating the probability of occurrence is also a regular exercise. Chapter 3 presents a basic model of PROS^{2}E that is improved in the following chapters.

Chapter 4 is dedicated to countermeasures. These events that hinder the overall realisation of a scenario require specific probabilistic modelling to be included in PROS^{2}E, which we have provided. We also discuss the importance of including these countermeasures in scenarios. Since safety and security are historically managed separately, the side effects of a safety countermeasure on the security are not necessarily anticipated, et vice versa. All the more reasons to include them in scenarios, as well as diagnosing both safety and security at the same time in order not to trigger a countermeasure for a wrongfully identified incident.

Chapter 5 focuses on improving the time capabilities of PROS^{2}E with two principal aspects: sequentiality and counting time. Sequentiality is a common problem that required widely-used models such as fault trees or attack trees to evolve: it occurs when the order in which events happen have a consequence on the overall outcome. We have tackled this problem by adding a Sequential AND gate, and expressing it both logically and probabilistically. This increased the number of scenarios that could be represented, and provided more accurate results. Counting time was troublesome when a component could be stopped, restarted, and replaced. We introduced the concept of a stopwatch in the LAMBDA models to address this issue. This improvement yields much more accurate results to avoid incorrect diagnosis.

If we look back at the classification established in Chapter 2, we conclude that PROS^{2}E is a *hybrid* model since it considers both safety and security events, it is *descriptive*, *explicative*,

adaptative and *real-time* thanks to the event graph and the ability to process alerts, it is *evaluative* due to the probabilistic theory, *remediative* with the improvements of Chapter 4. PROS^{2}E ticks all the boxes of the classification. However and this is addressed in the perspectives section, PROS^{2}E is not perfect and can be improved.

6.2 Perspectives

One of the most natural way to continue our work is to automate PROS^{2}E. For now, the probabilistic part is done in independent Maple spreadsheets that served their purpose as a proof of concept and proof of feasibility but an integrated solution is most desirable. Beyond obvious programming work necessary to integrate the parts developed in this thesis, two important aspects need to be discussed.

First, there are the inputs of PROS^{2}E. PROS^{2}E relies on capturing alerts in order to know what events are happening in the system. Ideally, these alerts need to be transmitted using a standard format. For security, standards such as IDMEF exist. For safety, this is not the case. For now, when a SCADA system needs to communicate with another entity, such as an Enterprise Resource Planning (ERP), ad-hoc solutions are implemented. PROS^{2}E cannot rely on these ad-hoc solutions and, even beyond diagnosis, cyber-physical systems would gain from an alert exchange format developed for safety too.

Secondly, a capacity to interrogate PROS^{2}E is required for a complete automation. This can take the form of a query language for example. This would allow an operator to interrogate PROS^{2}E with simple questions such as "What are the N most probable scenarios in the next T minutes?", "What sequence of events lead to the current situation?" ou "How likely was this event?". These questions would be translated into logic and probabilistic computations to output the required response in the clearest way possible. But it has to be preceded by a study of what information is required and how to present it. Providing exhaustive but raw data is probably not the best option. Indeed, diagnosis is most likely performed in time of stress and the information delivered should be as clear and useful as possible.

In the same vein as providing meaningful result, comes the problem of establishing the probability distribution functions and particularly measuring the uncertainty. As discussed in the thesis, anticipating an attack with precision is still an ongoing issue that PROS^{2}E is not designed to solve. However, we know that the distribution functions are established with various degrees of confidence. Providing a way to express this confidence and carrying it to complete scenarios will modulate the results output by PROS^{2}E will improve the trust that we have in the results output by the model. Moreover, there is a concept of volatility of a random variable: it is the impact that a change in this random variable has on the overall result. A random variable with a high volatility corresponds to an event that has a great influence on the result and should

be monitored more thoroughly. Being able to express this volatility would enrich the diagnosis.

Lastly, we regret not having been able to pair our probabilistic evaluation with an impact measure in order to output the risk associated with scenarios. The probability of occurrence events is not enough when having to chose which one to act against. Indeed, a less probable event with an important impact might be more interesting to prevent than a probable one with little to no impact. This choice is of course to be made by the system experts, but we believe a PROS^{2}E, and diagnosis models in general, should provide such information.

We must also keep in mind that security and safety are proprieties of the system and do not represent finalities to achieve. Trying to improve the security in industrial systems is met with their huge inertia when it comes to change. The constraints of industrial systems and the concerns of the people that work in them are different from those of conventional information systems. This is also one of the reason behind many choices we made regarding PROS^{2}E and diagnosis in general. And it should always be considered when continuing our work on diagnosis, or most universally when proposing solution to improve industrial systems' security.

From a technical point of view, the convergence of information technologies and operational technologies also poses numerous tricky challenges. In the sate of the art of this thesis, we have studied several safety and security models and we have measured how difficult it is to mix safety and security representations. Considerable efforts will be necessary to develop models that meet both the expectations of safety and security experts, as well as capture the semantics peculiar to their respective domains.

In a nutshell, understanding the needs of two so far independent functions of systems, with different perspectives and objectives, and with irreconcilable legacy techniques creates a mostly uncharted domain that was very exciting to explore during this thesis. This domain is still not mature and is full of interesting and rewarding challenges that we look forward to tackle in the near future.

BIBLIOGRAPHY

[1] AL-RABIAAH, S. The “Stuxnet” Virus of 2010 As an Example of A “APT” and Its “Recent” Variances. In *2018 21st Saudi Computer Society National Computer Conference (NCC)* (Apr. 2018), pp. 1–5.

[2] ALI, D. M. Applications of minimal path set and dual fault tree approach for piecewise reliability evaluation of large-scale electrical power systems. In *2008 12th International Middle-East Power System Conference*, pp. 11–17.

[3] ALTHEBYAN, Q., AND PANDA, B. A knowledge-based bayesian model for analyzing a system after an insider attack. Proceedings of The Ifip Tc 11 23rd International Information Security Conference, Springer US, pp. 557–571.

[4] AMARI, S. V., AND AKERS, J. B. Reliability analysis of large fault trees using the vesely failure rate. In *Reliability and Maintainability, 2004 Annual Symposium - RAMS*, pp. 391–396.

[5] AMIN, S., LITRICO, X., SASTRY, S., AND BAYEN, A. M. Cyber security of water scada systems-part i: Analysis and experimentation of stealthy deception attacks. *Ieee Transactions on Control Systems Technology 21*, 5 (2013), 1963–1970.

[6] ARDI, S., BYERS, D., AND SHAHMEHRI, N. Towards a structured unified process for software security. In *Proceedings of the 2006 international workshop on Software engineering for secure systems* (1137630), ACM, pp. 3–10.

[7] ARNOLD, F., HERMANNS, H., PULUNGAN, R., AND STOELINGA, M. *Time-Dependent Analysis of Attacks*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014, pp. 285–305.

[8] ARROYO-FIGUEROA, G., SUCAR, L. E., AND VILLAVICENCIO, A. Probabilistic temporal reasoning and its application to fossil power plant operation. *Expert Systems with Applications 15*, 3 (1998), 317–324.

[9] ARSHAD, J., TOWNEND, P., AND XU, J. An automatic intrusion diagnosis approach for clouds. *International Journal of Automation and Computing 8*, 3 (2011), 286.

[10] BACA, D., AND PETERSEN, K. Prioritizing countermeasures through the countermeasure method for software security (cm-sec). Product-Focused Software Process Improvement, Springer Berlin Heidelberg, pp. 176–190.

[11] BARLOW, R. E., AND PROSCHAN, F. *Statistical theory of reliability and life testing: probability models*. Florida State Univ Tallahassee, 1975.

[12] BISTARELLI, S., FIORAVANTI, F., AND PERETTI, P. Defense trees for economic evaluation of security investments. In *First International Conference on Availability, Reliability and Security (ARES'06)*, p. 8 pp.

[13] BOBBIO, A., PORTINALE, L., MINICHINO, M., AND CIANCAMERLA, E. Improving the analysis of dependable systems by mapping fault trees into bayesian networks. *Reliability Engineering & System Safety 71*, 3 (2001), 249–260.

[14] BOBBIO, A., AND RAITERI, D. C. Parametric fault trees with dynamic gates and repair boxes. In *Annual Symposium Reliability and Maintainability, 2004 - RAMS*, pp. 459–465.

[15] BOUDALI, H., CROUZEN, P., AND STOELINGA, M. A compositional semantics for dynamic fault trees in terms of interactive markov chains. Automated Technology for Verification and Analysis, Springer Berlin Heidelberg, pp. 441–456.

[16] BOUDALI, H., AND DUGA, J. B. A new bayesian network approach to solve dynamic fault trees. In *Annual Reliability and Maintainability Symposium, 2005. Proceedings.*, pp. 451–456.

[17] BOUISSOU, M., AND BON, J.-L. *A new formalism that combines advantages of fault-trees and Markov models: Boolean logic driven Markov processes*, vol. 82. 2003.

[18] BOUISSOU, M., MARTIN, F., AND OURGHANLIAN, A. Assessment of a safety-critical system including software: a bayesian belief network for evidence sources. In *Annual Reliability and Maintainability. Symposium. 1999 Proceedings (Cat. No.99CH36283)*, pp. 142–150.

[19] BOUISSOU, M., VILLATTE, N., BOUHADANA, H., AND BANNELIER, M. Knowledge modelling and reliability processing: presentation of the figaro language and associated tools. Report, Electricite de France (EDF), December 1991.

[20] BRAYNOV, S., AND JADLIWALA, M. Representation and analysis of coordinated attacks. In *Proceedings of the 2003 ACM workshop on Formal methods in security engineering* (1035434, 2003), ACM, pp. 43–51.

[21] BRUCE, S. Attack trees. *Dr. Dobb's Journal* (1999).

[22] BULDAS, A., LAUD, P., PRIISALU, J., SAAREPERA, M., AND WILLEMSON, J. *Rational Choice of Security Measures Via Multi-parameter Attack Trees*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 235–248.

[23] CAMTEPE, S. A., AND YENER, B. Modeling and detection of complex attacks. In *2007 Third International Conference on Security and Privacy in Communications Networks and the Workshops - SecureComm 2007*, pp. 234–243.

[24] CARRASCO, J. A., AND SUNE, V. An algorithm to find minimal cuts of coherent fault-trees with event-classes, using a decision tree. *IEEE Transactions on Reliability 48*, 1 (1999), 31–41.

[25] CHAUX, P.-Y., ROUSSEL, J.-M., LESAGE, J.-J., DELEUZE, G., AND BOUISSOU, M. Towards a unified definition of minimal cut sequences. *IFAC Proceedings Volumes 46*, 22 (2013), 1–6.

[26] CHEN, J., ROBERTS, C., AND WESTON, P. Fault detection and diagnosis for railway track circuits using neuro-fuzzy systems. *Control Engineering Practice 16*, 5 (2008), 585–596.

[27] CHEN, T. M., AND ABU-NIMEH, S. Lessons from Stuxnet. *Computer 44*, 4 (Apr. 2011), 91–93.

[28] CHEREPANOV, A., AND LIPOVSKY, R. BLACKENERGY – WHAT WE REALLY KNOW ABOUT THE NOTORIOUS CYBER ATTACKS. *Virus Bulletin October* (2016), 8.

[29] CLÉDEL, T., FOLEY, S. N., CUPPENS, N., CUPPENS, F., KERMARREC, Y., DUBOIS, F., LAAROUCHI, Y., AND LE COMTE, G. Towards the evaluation of end-to-end resilience through external consistency. Cyberspace Safety and Security, Springer International Publishing, pp. 99–114.

[30] CLÉDEL, T., CUPPENS, N., AND CUPPENS, F. Managing the overestimation of resilience. In *Proceedings of the 14th International Conference on Availability, Reliability and Security* (New York, NY, USA, 2019), ARES '19, Association for Computing Machinery.

[31] CODETTA-RAITERI, D. The conversion of dynamic fault trees to stochastic petri nets, as a case of graph transformation. *Electronic Notes in Theoretical Computer Science 127*, 2 (2005), 45–60.

[32] COUDERT, O., AND MADRE, J. C. Fault tree analysis: 10/sup 20/ prime implicants and beyond. In *Annual Reliability and Maintainability Symposium 1993 Proceedings*, pp. 240–245.

[33] CUPPENS, F., AUTREL, F., BOUZIDA, Y., GARCIA, J., GOMBAULT, S., AND SANS, T. Anti-correlation as a criterion to select appropriate counter-measures in an intrusion detection framework. *Annales Des Télécommunications 61*, 1 (2006), 197–217.

[34] CUPPENS, F., AUTREL, F., MIEGE, A., AND BENFERHAT, S. Recognizing malicious intention in an intrusion detection process. In *HIS*, pp. 806–817.

[35] CUPPENS, F., AND MIEGE, A. Alert correlation in a cooperative intrusion detection framework. In *Proceedings 2002 IEEE Symposium on Security and Privacy*, pp. 202–215.

[36] CUPPENS, F., AND ORTALO, R. *LAMBDA: A Language to Model a Database for Detection of Attacks*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000, pp. 197–216.

[37] DANTU, R., LOPER, K., AND KOLAN, P. Risk management using behavior based attack graphs. In *International Conference on Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004.*, vol. 1, pp. 445–449 Vol.1.

[38] DEAN, T., AND KANAZAWA, K. A model for reasoning about persistence and causation. *Computational Intelligence 5*, 2 (1989), 142–150.

[39] DEBAR, H., CURRY, D., AND FEINSTEIN, B. The intrusion detection message exchange format (idmef). Rfc, RFC Editor, March 2007.

[40] DEWRI, R., RAY, I., POOLSAPPASIT, N., AND WHITLEY, D. Optimal security hardening on attack tree models of networks: a cost-benefit analysis. *International Journal of Information Security 11*, 3 (2012), 167–188.

[41] DUGAN, J. B., BAVUSO, S. J., AND BOYD, M. A. Fault trees and sequence dependencies. In *Annual Proceedings on Reliability and Maintainability Symposium*, pp. 286–293.

[42] DURGA RAO, K., GOPIKA, V., SANYASI RAO, V. V. S., KUSHWAHA, H. S., VERMA, A. K., AND SRIVIDYA, A. Dynamic fault tree analysis using monte carlo simulation in probabilistic safety assessment. *Reliability Engineering & System Safety 94*, 4 (2009), 872–883.

[43] DUTUIT, Y., CHÂTELET, E., SIGNORET, J. P., AND THOMAS, P. Dependability modelling and evaluation by using stochastic petri nets: application to two test cases. *Reliability Engineering & System Safety 55*, 2 (1997), 117–124.

[44] ELSAESSER, C., AND TANNER, M. C. Automated diagnosis for computer forensics. *The Mitre Corporation* (2001), 1–16.

[45] FENG, L., WANG, W., ZHU, L. N., AND ZHANG, Y. Predicting intrusion goal using dynamic bayesian network with transfer probability estimation. *Journal of Network and Computer Applications 32*, 3 (2009), 721–732.

[46] FOLEY, S. N., AUTREL, F., BOURGET, E., CLEDEL, T., GRUNENWALD, S., HERNAN, J. R., KABIL, A., LARSEN, R., ROONEY, V. M., AND VANHULST, K. Science hackathons

for cyberphysical system security research: Putting cps testbed platforms to good use. In *Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy* (3264897, 2018), ACM, pp. 102–107.

[47] FOVINO, I. N., MASERA, M., AND DE CIAN, A. Integrating cyber attacks within fault trees. *Reliability Engineering & System Safety 94*, 9 (2009), 1394–1402.

[48] FRANK, P. M. Fault diagnosis in dynamic systems using analytical and knowledge-based redundancy: A survey and some new results. *Automatica 26*, 3 (1990), 459–474.

[49] GONZALEZ-GRANADILLO, G., RUBIO-HERNAN, J., AND GARCIA-ALFARO, J. A pyramidal-based model to compute the impact of cyber security events. In *Proceedings of the 13th International Conference on Availability, Reliability and Security* (New York, NY, USA, 2018), ARES 2018, Association for Computing Machinery.

[50] GRIBAUDO, M., IACONO, M., AND MARRONE, S. Exploiting bayesian networks for the analysis of combined attack trees. *Electronic Notes in Theoretical Computer Science 310* (2015), 91–111.

[51] GULVANESSIAN, H., AND HOLICKÝ, M. Determination of actions due to fire: recent developments in bayesian risk assessment of structures under fire. *Progress in Structural Engineering and Materials 3*, 4 (2001), 346–352.

[52] HERNANDEZ-LEAL, P., GONZALEZ, J. A., MORALES, E. F., AND ENRIQUE SUCAR, L. Learning temporal nodes bayesian networks. *International Journal of Approximate Reasoning 54*, 8 (2013), 956–977.

[53] ISMAIL, Z., KIENNERT, C., LENEUTRE, J., AND CHEN, L. A game theoretical model for optimal distribution of network security resources. In *Decision and Game Theory for Security* (Cham, 2017), S. Rass, B. An, C. Kiekintveld, F. Fang, and S. Schauer, Eds., Springer International Publishing, pp. 234–255.

[54] ISMAIL, Z., LENEUTRE, J., BATEMAN, D., AND CHEN, L. A methodology to apply a game theoretic model of security risks interdependencies between ict and electric infrastructures. pp. 159–171.

[55] JACKSON, C., LEVITT, K., ROWE, J., KRISHNAMURTHY, S., JAEGER, T., AND SWAMI, A. A diagnosis based intrusion detection approach. In *MILCOM 2015 - 2015 IEEE Military Communications Conference*, pp. 929–934.

[56] JOHNSON, M. *The Denseness of Phase Distributions*. School of Industrial Engineering, Purdue University, 1988.

[57] JÜRGENSON, A., AND WILLEMSON, J. *Computing Exact Outcomes of Multi-parameter Attack Trees*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 1036–1051.

[58] JÜRGENSON, A., AND WILLEMSON, J. *Serial Model for Attack Tree Computations*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 118–128.

[59] K., T., S., R., AND S., B. Before the public service commission state of missouri - staff's initial incident report. Report, October 24 2007.

[60] KABIL, A., DUVAL, T., CUPPENS, N., COMTE, G. L., HALGAND, Y., AND PONCHEL, C. From cyber security activities to collaborative virtual environments practices through the 3d cybercop platform. In *Information Systems Security - 14th International Conference, ICISS 2018, Bangalore, India, December 17-19, 2018, Proceedings* (2018), V. Ganapathy, T. Jaeger, and R. K. Shyamasundar, Eds., vol. 11281 of *Lecture Notes in Computer Science*, Springer, pp. 272–287.

[61] KABIL, A., DUVAL, T., CUPPENS, N., COMTE, G. L., HALGAND, Y., AND PONCHEL, C. Why should we use 3D Collaborative Virtual Environments for Cyber Security? In *2018 IEEE Fourth VR International Workshop on Collaborative Virtual Environments (3DCVE)* (Mar. 2018), pp. 1–2. ISSN: null.

[62] KANG, C. W., AND GOLAY, M. W. A bayesian belief network-based advisory system for operational availability focused diagnosis of complex nuclear power systems. *Expert Systems with Applications 17*, 1 (1999), 21–32.

[63] KANOUN, W., CUPPENS-BOULAHIA, N., CUPPENS, F., DUBUS, S., AND MARTIN, A. Success likelihood of ongoing attacks for intrusion detection and response systems. In *2009 International Conference on Computational Science and Engineering* (2009), vol. 3, pp. 83–91.

[64] KHAND, P. A. System level security modeling using attack trees. In *2009 2nd International Conference on Computer, Control and Communication*, pp. 1–6.

[65] KORDY, B., PIÈTRE-CAMBACÉDÈS, L., AND SCHWEITZER, P. Dag-based attack and defense modeling: Don't miss the forest for the attack trees. *Computer Science Review 13-14* (2014), 1–38.

[66] KRIAA, S., BOUISSOU, M., COLIN, F., HALGAND, Y., AND PIETRE-CAMBACEDES, L. *Safety and Security Interactions Modeling Using the BDMP Formalism: Case Study of a Pipeline*. Springer International Publishing, Cham, 2014, pp. 326–341.

[67] KUMAR, R., AND STOELINGA, M. Quantitative security and safety analysis with attack-fault trees. In *2017 IEEE 18th International Symposium on High Assurance Systems Engineering (HASE)*, pp. 25–32.

[68] LABS, A. Comprehensive Analysis Report on Ukraine Power System Attacks - Antiy Labs | The Next Generation Anti-Virus Engine Innovator, 2016.

[69] LANGNER, R. Stuxnet: Dissecting a Cyberwarfare Weapon. *IEEE Security Privacy 9*, 3 (May 2011), 49–51.

[70] LEE, W., FAN, W., MILLER, M., STOLFO, S. J., AND ZADOK, E. Toward cost-sensitive modeling for intrusion detection and response. *J. Comput. Secur. 10*, 1–2 (July 2002), 5–22.

[71] LEE, W. S., GROSH, D. L., TILLMAN, F. A., AND LIE, C. H. Fault tree analysis, methods, and applications - a review. *IEEE Transactions on Reliability R-34*, 3 (1985), 194–203.

[72] LI, B., LU, R., CHOO, K.-K. R., WANG, W., AND LUO, S. On reliability analysis of smart grids under topology attacks: A stochastic petri net approach. *ACM Trans. Cyber-Phys. Syst. 3*, 1 (2018), 1–25.

[73] LIU, D., XING, W., ZHANG, C., LI, R., AND LI, H. Cut sequence set generation for fault tree analysis. Embedded Software and Systems, Springer Berlin Heidelberg, pp. 592–603.

[74] LIU, D., ZHANG, C., XING, W., LI, R., AND LI, H. Quantification of cut sequence set for fault tree analysis. High Performance Computing and Communications, Springer Berlin Heidelberg, pp. 755–765.

[75] LV, W., AND LI, W. Space based information system security risk evaluation based on improved attack trees. In *2011 Third International Conference on Multimedia Information Networking and Security*, pp. 480–483.

[76] MA, J. *Methods and Systems for Fault Diagnosis in Nuclear Power Plants*. Thesis, 2015.

[77] MALHOTRA, M., AND TRIVEDI, K. S. Dependability modeling using petri-nets. *IEEE Transactions on Reliability 44*, 3 (1995), 428–440.

[78] MARSAN, M. A., CONTE, G., AND BALBO, G. A class of generalized stochastic petri nets for the performance evaluation of multiprocessor systems. *ACM Trans. Comput. Syst. 2*, 2 (1984), 93–122.

[79] MCQUEEN, M. A., BOYER, W. F., FLYNN, M. A., AND BEITEL, G. A. Quantitative cyber risk reduction estimation methodology for a small scada control system. In *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06)*, vol. 9, pp. 226–226.

[80] MELL, P., SCARFONE, K., AND ROMANOSKY, S. Common vulnerability, scoring system. *Ieee Security & Privacy 4*, 6 (2006), 85–89.

[81] MENGSHOEL, O. J., DARWICHE, A., AND UCKUN, S. Sensor validation using bayesian networks. In *Proceedings of the 9th international symposium on artificial intelligence, robotics, and automation in space (iSAIRAS-08)*.

[82] MERLE, G. *Algebraic modelling of Dynamic Fault Trees, contribution to qualitative and quantitative analysis*. Thesis, 2010.

[83] MONTANI, S., PORTINALE, L., AND BOBBIO, A. Dynamic bayesian networks for modeling advanced fault tree features in dependability analysis. pp. 1414–1422.

[84] N., F. S. A grounded theory approach to security policy elicitation. 454–471.

[85] NIGAM, V., PRETSCHNER, A., AND RUESS, H. Model-based safety and security engineering. 2018.

[86] NING, P., CUI, Y., AND REEVES, D. S. Constructing attack scenarios through correlation of intrusion alerts. In *Proceedings of the 9th ACM conference on Computer and communications security* (586144, 2002), ACM, pp. 245–254.

[87] NOEL, S., ROBERTSON, E., AND JAJODIA, S. *Correlating Intrusion Events and Building Attack Scenarios Through Attack Graph Distances*. 2004.

[88] NZOUKOU, W., WANG, L., JAJODIA, S., AND SINGHAL, A. A unified framework for measuring a network's mean time-to-compromise. In *2013 IEEE 32nd International Symposium on Reliable Distributed Systems*, pp. 215–224.

[89] PEARL, J. Fusion, propagation, and structuring in belief networks. *Artificial Intelligence 29*, 3 (1986), 241–288.

[90] PERNESTÅL, A. *A Bayesian approach to fault isolation with application to diesel engine diagnosis*. Thesis, 2007.

[91] PIETRE-CAMBACEDES, L., DESFLESSELLE, Y., AND BOUISSOU, M. Attack and defense dynamic modeling with bdmp - extended version.pdf. 2010.

[92] PIRIOU, P. Y., FAURE, J. M., AND LESAGE, J. J. Generalized boolean logic driven markov processes: A powerful modeling framework for model-based safety analysis of dynamic repairable and reconfigurable systems. *Reliability Engineering & System Safety 163* (2017), 57–68.

[93] PIÈTRE-CAMBACÉDÈS, L., AND BOUISSOU, M. Beyond attack trees: Dynamic security modeling with boolean logic driven markov processes (bdmp). In *Proceedings of the 2010 European Dependable Computing Conference* (1827752), IEEE Computer Society, pp. 199–208.

[94] PIÈTRE-CAMBACÉDÈS, L., AND BOUISSOU, M. Modeling safety and security interdependencies with bdmp (boolean logic driven markov processes). In *2010 IEEE International Conference on Systems, Man and Cybernetics*, pp. 2852–2861.

[95] PIÈTRE-CAMBACÉDÈS, L., AND CHAUDET, C. The sema referential framework: Avoiding ambiguities in the terms "security" and "safety". *International Journal of Critical Infrastructure Protection 3*, 2 (2010), 55–66.

[96] POOLSAPASSIT, N., AND RAY, I. *Investigating Computer Attacks Using Attack Trees*. Springer New York, New York, NY, 2007, pp. 331–343.

[97] PORTINALE, L., RAITERI, D. C., AND MONTANI, S. Supporting reliability engineers in exploiting the power of dynamic bayesian networks. *International Journal of Approximate Reasoning 51*, 2 (2010), 179–195.

[98] QIN, X., AND LEE, W. Attack plan recognition and prediction using causal networks. In *20th Annual Computer Security Applications Conference*, pp. 370–379.

[99] RAUZY, A. B. Sequence algebra, sequence decision diagrams and dynamic fault trees. *Reliability Engineering & System Safety 96*, 7 (2011), 785–792.

[100] RAY, I., AND POOLSAPASSIT, N. *Using Attack Trees to Identify Malicious Attacks from Authorized Insiders*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 231–246.

[101] ROGERS, J., AND M. WATKINS, C. *OVERVIEW OF THE TAUM SAUK PUMPED STORAGE POWER PLANT UPPER RESERVOIR FAILURE, REYNOLDS COUNTY, MO*. 2008.

[102] ROONEY, V. M., AND FOLEY, S. N. An online consent maturity model: Moving from acceptable use towards ethical practice. In *Proceedings of the New Security Paradigms Workshop* (New York, NY, USA, 2018), NSPW '18, Association for Computing Machinery, p. 64–79.

[103] ROONEY, V. M., AND FOLEY, S. N. What Users Want: Adapting Qualitative Research Methods to Security Policy Elicitation. In *Computer Security* (Cham, 2018), S. K. Katsikas, F. Cuppens, N. Cuppens, C. Lambrinoudakis, C. Kalloniatis, J. Mylopoulos, A. Antón, and S. Gritzalis, Eds., Lecture Notes in Computer Science, Springer International Publishing, pp. 229–249.

[104] ROONEY, V. M., AND FOLEY, S. N. What you can change and what you can't: Human experience in computer network defenses. In *Secure IT Systems* (Cham, 2018), N. Gruschka, Ed., Springer International Publishing, pp. 219–235.

[105] RUBIO-HERNAN, J., DE CICCO, L., AND GARCIA-ALFARO, J. Event-triggered watermarking control to handle cyber-physical integrity attacks. In *Secure IT Systems* (Cham, 2016), B. B. Brumley and J. Röning, Eds., Springer International Publishing, pp. 3–19.

[106] RUBIO-HERNAN, J., DE CICCO, L., AND GARCIA-ALFARO, J. On the use of watermark-based schemes to detect cyber-physical attacks. *EURASIP Journal on Information Security 2017*, 1 (2017), 8.

[107] RUBIO-HERNAN, J., DE CICCO, L., AND GARCIA-ALFARO, J. Adaptive control-theoretic detection of integrity attacks against cyber-physical industrial systems. *Transactions on Emerging Telecommunications Technologies 29*, 7 (2018), e3209. e3209 ett.3209.

[108] RUBIO-HERNAN, J., SAHAY, R., DE CICCO, L., AND GARCIA-ALFARO, J. Cyber-physical architecture assisted by programmable networking. *Internet Technology Letters 1*, 4 (2018), e44.

[109] RUBIO HERNAN, J. M. *Detection of attacks against cyber-physical industrial systems*. Theses, Institut National des Télécommunications, July 2017.

[110] RUBIO-HERNÁN, J., D. CICCO, L., AND GARCÍA-ALFARO, J. Revisiting a watermark-based detection scheme to handle cyber-physical attacks. In *2016 11th International Conference on Availability, Reliability and Security (ARES)* (Aug 2016), pp. 21–28.

[111] RUIJTERS, E., AND STOELINGA, M. Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools. *Computer Science Review 15-16* (2015), 29–62.

[112] SAMARJI, L., CUPPENS-BOULAHIA, N., CUPPENS, F., PAPILLON, S., KANOUN, W., AND DUBUS, S. Coordination and concurrency aware likelihood assessment of simultaneous attacks. International Conference on Security and Privacy in Communication Networks, Springer International Publishing, pp. 524–529.

[113] SCHNEEWEISS, W. G. On the polynomial form of boolean functions: Derivations and applications. *IEEE Trans. Comput. 47*, 2 (1998), 217–221.

[114] SCHROEDER, B., AND GIBSON, G. A. Disk failures in the real world: what does an mttf of 1,000,000 hours mean to you? In *Proceedings of the 5th USENIX conference on File and Storage Technologies* (1267904), USENIX Association, p. 1.

[115] TEAM, F. T. S. I. Report of findings on the overtoppingand embankment breach of the upper dam - taum sauk pumped storage project,. Report, April 28 2006.

[116] TEN, C., LIU, C., AND MANIMARAN, G. Vulnerability assessment of cybersecurity for scada systems. *IEEE Transactions on Power Systems 23*, 4 (2008), 1836–1846.

[117] TORRES-TOLEDANO, J. G., AND SUCAR, L. E. Bayesian networks for reliability analysis of complex systems. Progress in Artificial Intelligence — IBERAMIA 98, Springer Berlin Heidelberg, pp. 195–206.

[118] TOTH, T., AND KRUEGEL, C. Evaluating the impact of automated intrusion response mechanisms. In *18th Annual Computer Security Applications Conference, 2002. Proceedings.* (Dec 2002), pp. 301–310.

[119] TRAVÉ-MASSUYÈS, L. Bridging control and artificial intelligence theories for diagnosis: A survey. *Engineering Applications of Artificial Intelligence 27* (2014), 1–16.

[120] VESELY, W. E., GOLDBERG, F. F., ROBERTS, N. H., AND HAASL, D. F. Fault tree handbook. Report, DTIC Document, 1981.

[121] VOLOVOI, V. Modeling of system reliability petri nets with aging tokens. *Reliability Engineering & System Safety 84*, 2 (2004), 149–161.

[122] WEBER, P., MEDINA-OLIVA, G., SIMON, C., AND IUNG, B. Overview on bayesian networks applications for dependability, risk analysis and maintenance areas. *Engineering Applications of Artificial Intelligence 25*, 4 (2012), 671–682.

[123] WEISS, J. D. A system security engineering process. In *14th Annual NCSC/NIST National Computer Security Conference*, vol. 2, pp. 572–581.

[124] XIANG, J., YANOO, K., MAENO, Y., TADANO, K., MACHIDA, F., KOBAYASHI, A., AND OSAKI, T. Efficient analysis of fault trees with voting gates. In *2011 IEEE 22nd International Symposium on Software Reliability Engineering*, pp. 230–239.

[125] ZHANG, H., ZHANG, C., LIU, D., AND LI, R. A method of quantitative analysis for dynamic fault tree. In *2011 Proceedings - Annual Reliability and Maintainability Symposium*, pp. 1–6.

[126] ZHIHUA, T., AND DUGAN, J. B. Minimal cut set/sequence generation for dynamic fault trees. In *Annual Symposium Reliability and Maintainability, 2004 - RAMS*, pp. 207–213.

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Titre : Diagnostic de défaillance et de malveillance dans les systèmes de contrôle industriels

Mot clés : Diagnostic, Sûreté de fonctionnement, Sécurité, Systèmes industriels

Résumé : La convergence des systèmes d'informations et des systèmes industriels entraine un changement de paradigme dans la gestion des incidents accidentels et malveillants. Sûreté de fonctionnement et sécurité doivent désormais interagir, ce qui change le périmètre et les problématiques du diagnostic.

Après avoir défini ce nouveau périmètre du diagnostic, cette thèse fournit une analyse des modèles existants permettant de fournir des informations nécessaires au diagnostic. Elle propose ensuite PROS^{2}E, un nouveau modèle d'évènements sur lequel s'appuyer pour diagnostiquer des incidents dans des systèmes industriels. Il a été spécifiquement conçu pour réutiliser l'expertise déjà présente dans les différents métiers de la sûreté de fonctionnement et de la sécurité. PROS^{2}E est ensuite amélioré pour représenter des incidents plus complexes et fournir des informations avec plus de précision. Plusieurs exemples illustrent les possibilités de diagnostic du modèle.

Title: Diagnosing accidental and maliciious events in industrial control systems

Keywords: Diagnosis, Safety, Security, Industrial systems

Abstract: The convergence of information and industrial systems triggered a paradigm shift in the management of malicious and accidental events. Safety and security must now interact and it changes the perimeters and the issues of diagnosis.

After defining this new perimeter, this thesis provides an analysis of existing models that provide necessary informations for diagnosis. It then proposes PROS^{2}E, a new event model upon which safety and security diagnosis can be performed in industrial systems. It was specificaly designed to exploit experience already present in the fields of safety and security management. PROS^{2}E is then improved to represent more complex incidents and provide more accurate information. Several examples illustrate the diagnosis capacities of the model.