



## MASTER RESEARCH INTERNSHIP



## BIBLIOGRAPHIC REPORT

---

### Design of libraries for Attack Tree Synthesis

---

**Domain:** Languages and Automata Theory - Symbolic Computation

*Author:*  
Paul LAURENT

*Supervisor:*  
Sophie PINCHINAT  
LogicA

**Abstract:** write your abstract here

## Contents

<b>Introduction</b>	<b>1</b>
<b>1 Notations préliminaires</b>	<b>1</b>
1.1 Modélisation des systèmes . . . . .	1
1.2 Traces . . . . .	1
1.3 Attack trees . . . . .	1
1.4 Libraries . . . . .	1
<b>2 Modélisation d'attaques à l'aide de modèles graphiques de sécurité</b>	<b>2</b>
2.1 Avantages des modèles graphiques de sécurité . . . . .	2
2.2 Principaux modèles graphiques de sécurité . . . . .	2
2.2.1 Arbres d'attaque . . . . .	2
2.2.2 Graphes d'attaque . . . . .	3
2.3 Comparaison entre arbres et graphes d'attaque . . . . .	4
2.3.1 Traduction d'ATs en AGs . . . . .	4
2.3.2 Traduction pertinente de graphes en arbres d'attaque . . . . .	4
2.3.3 Analyse quantitative : ATs vs AGs . . . . .	5
<b>3 Construire des arbres d'attaque pertinents</b>	<b>5</b>
3.1 Approches de génération d'arbres d'attaque . . . . .	5
3.1.1 Génération d'AT par analyse syntaxique du système . . . . .	6
3.1.2 Génération d'AT par atteignabilité dans un graphe . . . . .	6
3.1.3 Génération d'AT par model checking . . . . .	7
3.2 Guided design of attack trees . . . . .	7
3.3 Library-based attack tree synthesis . . . . .	7
<b>4 Attack Trees for Information Systems</b>	<b>7</b>
4.1 Manual modeling of information systems and attacker capabilities . . . . .	7
4.2 Building libraries from databases of known attacks . . . . .	7
4.3 Towards automatic attack tree generation from logs . . . . .	8
<b>Conclusion</b>	<b>9</b>

# Introduction

## 1 Notations préliminaires

Cette section présente les notations et définitions de base utilisées dans le reste du document.

### 1.1 Modélisation des systèmes

**Définition 1** (Système de transition étiqueté (LTS)). *Pour représenter l'évolution des états d'un système, nous utilisons des systèmes de transition étiquetés (LTS - Labeled Transition Systems). Soit  $\text{Prop}$  l'ensemble des propositions sur un ensemble d'états  $S$ . Un LTS est un triplet  $(S, \rightarrow, \lambda)$  où  $\rightarrow \subseteq S \times A \times S$  est une relation de transition étiquetée par des actions issues d'un ensemble fini  $A$ , et  $\lambda : S \rightarrow 2^{\text{Prop}}$  est une fonction de valuation qui associe à chaque état un ensemble de propriétés atomiques vraies dans cet état.*

**Définition 2** (Asset-Based System (ABS)).

### 1.2 Traces

La littérature distingue deux sémantiques principales données aux traces utilisées pour modéliser les scénarios d'attaque dans les modèles de sécurité [16, 20] : les traces basées sur les actions (ou action-based), comme définies dans [22] et les traces basées sur les états (ou state-based), comme présentées dans [3, 18, 2].

**Définition 3** (Traces action-based)). *Une action-based sur les actions est une séquence finie d'actions  $a_1 a_2 \dots a_n$  où chaque action  $a_i$  appartient à  $\mathbb{B}$  l'ensemble des actions élémentaires de l'attaquant, issu de l'ensemble  $\rightarrow$  des transitions du LTS.*

**Définition 4** (Traces state-based). *Une trace state-based est une séquence finie de valuations  $v_1 v_2 \dots v_n$  où chaque valuation  $v_i$  est un ensemble de propositions atomiques vraies dans un état du LTS.*

**Définition 5** (Concaténation synchrone).

**Définition 6** (Shuffle de traces).

**Définition 7** (Composition parallèle).

### 1.3 Attack trees

**Définition 8** (Arbre d'attaque).

### 1.4 Libraries

**Définition 9** (Bibliothèque d'arbres d'attaque).

## 2 Modélisation d'attaques à l'aide de modèles graphiques de sécurité

### 2.1 Avantages des modèles graphiques de sécurité

Les modèles graphiques de sécurité, tels que les arbres d'attaque [20] ou les graphes d'attaque [16], sont largement utilisés pour représenter et analyser les scénarios d'attaque contre des systèmes informatiques. Il s'agit de structures mathématiques qui modélisent les différentes étapes qu'un attaquant peut suivre pour compromettre un système, en décomposant les objectifs d'attaque en sous-objectifs plus petits.

Ces modèles offrent plusieurs avantages clés :

- **Communication** : Ils fournissent une visualisation intuitive des scénarios d'attaque, et sont définis avec une sémantique formelle, facilitant la communication entre les parties prenantes, y compris les experts en sécurité et les scientifiques.
- **Raisonnement** : Leur sémantique formelle permet leur mécanisation, permettant leur génération automatique, l'identification automatique des vulnérabilités par test d'atteignabilité, ou encore l'analyse numérique des scénarios d'attaque (coût, probabilité de succès, etc.).

### 2.2 Principaux modèles graphiques de sécurité

#### 2.2.1 Arbres d'attaque

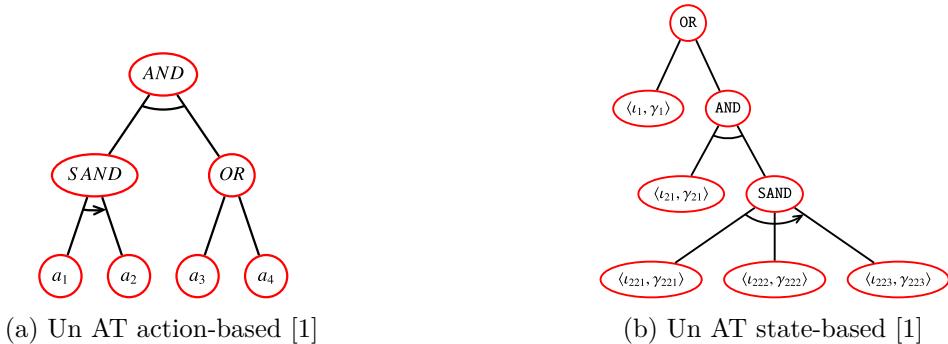


Figure 1: Exemples d'arbres d'attaque.

Les arbres d'attaque (AT) sont des structures arborescentes où chaque nœud représente un sous-objectif d'attaque, pouvant être réduit à effectuer une action élémentaire. Les nœuds internes sont connectés par des opérateurs logiques tels que AND, OR, et parfois des opérateurs séquentiels comme SAND (AND séquentiel) pour modéliser des dépendances temporelles entre les actions.

La Figure 1 présente un exemple d'arbre d'attaque action-based (à gauche), et state-based (à droite).

Dans la littérature, les arbres d'attaque existent sous plusieurs variantes, différent par leur sémantique. On distingue principalement deux approches :

- **Basé sur les actions (Action-based)** : où les feuilles de l'arbre sont étiquetées par des actions.
  - Dans [8], une feuille ne reconnaît que la trace composée uniquement de l'action  $a$ .
  - Dans INSÉRER CITATION, une feuille reconnaît toutes les traces contenant au moins l'action  $a$  à n'importe quelle position.
- **Basé sur les états (State-based)** : Utilisée dans [3, 18], où les feuilles de l'arbre sont un couple  $\langle \iota, \gamma \rangle$  avec  $\iota, \gamma \in \text{Prop}$ , représentant respectivement les conditions initiales et finales du système. Dans certains cas,  $\iota$  peut être omis si aucune condition initiale spécifique n'est requise, ou si elle est triviale.

### 2.2.2 Graphes d'attaque

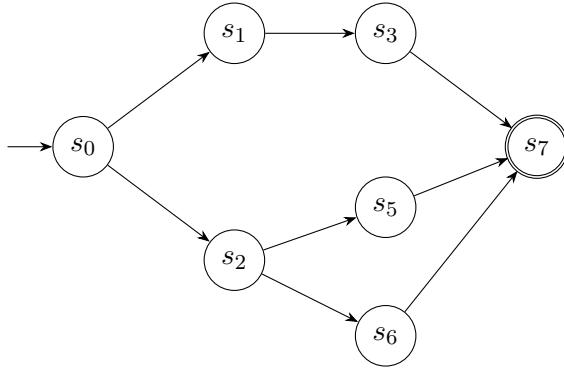


Figure 2: Exemple d'un AG

Dans [21] un graphe d'attaque (AG) est défini ainsi :

**Définition 10** (Graphe d'attaque). *Un graphe d'attaque est un tuple  $(S, \rightarrow, S_0, S_s)$ , où  $S$  est un ensemble de nœuds représentant les états du système,  $\rightarrow \subseteq S \times S$  est une relation de transition,  $S_0 \subseteq S$  correspond à l'ensemble des états initiaux, et  $S_s \subseteq S$  désigne l'ensemble des états cibles que l'attaquant cherche à atteindre.*

La Figure 2 illustre un exemple simple de graphe d'attaque, qui admet les modélisé les attaques dont le but est d'atteindre l'état  $s_7$  à partir de l'état initial  $s_0$ . Des états intermédiaires sont inclus pour représenter les étapes clés de l'attaque.

Une autre définition proposée par [17] impose que les transitions soient étiquetées par des actions de l'attaquant :

**Définition 11** (Graphe d'attaque étiqueté). *Un graphe d'attaque étiqueté est un tuple  $(S, \delta, S_0, S_s)$ , où  $S$ ,  $S_0$  et  $S_s$  sont définis comme pour un graphe d'attaque classique, et*

$\delta : S \times \mathcal{A} \sqcup \sqcup \rightarrow S$  est une fonction de transition étiquetée par des éléments de  $\mathcal{A} \sqcup \sqcup$ , représentant les actions de l'attaquant.

Cette définition offre une représentation exhaustive des états du système, plus adaptée pour le l'analyse automatique. Cependant, on perd la possibilité de ne représenter que les états clés atteints durant le déroulement de l'attaque, ce qui nuit à la lisibilité du modèle par les humains.

## 2.3 Comparaison entre arbres et graphes d'attaque

### 2.3.1 Traduction d'ATs en AGs

Dans [3], un arbre d'attaque  $\tau$  induit un automate  $\mathcal{A}_\tau = (Q, A, \delta, I, F)$  tel que les traces admissibles par  $\mathcal{A}_\tau$  coïncide avec l'ensemble des traces admissibles de  $\tau$ , où :

- $Q$  est l'ensemble des états de l'automate, construit à partir des nœuds de l'AT ;
- $A$  est l'alphabet des actions de l'attaquant ;
- $\delta : Q \times A \rightarrow Q$  est la fonction de transition, définie en fonction des relations entre les nœuds de l'AT ;
- $I \subseteq Q$  est l'ensemble des états initiaux ;
- $F \subseteq Q$  est l'ensemble des états finaux.

À partir de cet automate, on peut définir un AG qui admet les mêmes traces que  $\tau$ .  $G_\tau = (Q, \rightarrow_G, I, F)$ , où la relation de transition  $\rightarrow_G$  est induite par la fonction de transition  $\delta$ .

Cependant, dans le graphe ainsi construit, toutes les étapes intermédiaires parcourues par l'attaquant pour passer d'un état A à un état B sont explicitement représentées. Cela va à l'encontre de l'objectif principal des graphes d'attaque, qui est de ne retenir que les états clés atteints durant le déroulement de l'attaque.

### 2.3.2 Traduction pertinente de graphes en arbres d'attaque

En revanche, dans [17], une traduction d'un graphe d'attaque en un arbre d'attaque est proposée, en conservant les mêmes traces.

La méthode prend en entrée :

1. Un graphe d'attaque  $G = (S, \delta, S_0, S_s)$ , où toutes les transitions de  $G$  sont étiquetées par des actions issues de  $\mathcal{A} \sqcup \sqcup$  ;
2. Une hiérarchie d'actions de haut niveau  $\mathbb{H} = (\{\mathcal{H}_k\}_{0 \leq k \leq K}, \mathcal{R})$ , où :
  - $\mathcal{H}_0 = \mathcal{A} \sqcup \sqcup$  est l'ensemble des actions élémentaires de l'attaquant ;
  - $\mathcal{H}_k$  pour  $0 < k \leq K$  est un ensemble d'actions de plus haut niveau, chacune étant définie comme une séquence d'actions appartenant à  $\bigcup_{0 \leq j < k} \mathcal{H}_j$ .  
Pour la suite, on note  $\mathcal{H} = \bigcup_{0 \leq k \leq K} \mathcal{H}_k$  l'ensemble de toutes les actions. ;
  - $\mathcal{R} \subseteq \mathcal{H} \times \mathcal{H}^*$  est une relation définissant la décomposition des actions de plus haut niveau en séquences d'actions niveau inférieur.

Et produit en sortie un arbre d'attaque  $\tau_G$ , labellé par des actions de  $\mathcal{H}$ , tel que le langage reconnu par  $\tau_G$  coïncide avec l'ensemble des traces admissibles de  $G$ .

Cependant, cette transformation requiert une entrée externe pour regrouper plusieurs actions élémentaires en actions de plus haut niveau, afin d'obtenir un AT pertinent.

Ainsi, bien que des traductions existent entre les ATs et les AGs, elles ne conservent pas l'objectif principal de chacun des modèles sans l'introduction d'informations analogues. Cela souligne la différence dans les informations que chaque modèle cherche à capturer et à représenter : les états clés atteints au cours de l'attaque pour les AGs, et la décomposition de l'attaque en sous-objectifs organisés de manière hiérarchique pour les ATs.

### 2.3.3 Analyse quantitative : ATs vs AGs

les différences d'expressivité entre les arbres et AGs citées précédemment se concrétisent principalement dans le cadre de l'analyse quantitative des scénarios d'attaque.

En effet, par leur structure hiérarchique et acyclique, les ATs permettent une analyse plus directe et efficace des métriques telles que le coût total de l'attaque, la probabilité de succès, ou encore le temps nécessaire pour mener à bien l'attaque. Dans ces analyses, des valeurs numériques sont attribuées aux feuilles de l'arbre, puis sont agrégées vers la racine en fonction des opérateurs logiques utilisés (AND, OR, SAND). Le survey [23] offre un aperçu complet des différentes méthodes d'analyse quantitative appliquées aux ATs. Cependant, cela ne les empêche pas de supporter des analyses quantitatives complexes, notamment basées sur des automates temporisés pondérés (priced timed automata) comme dans [13, 2].

En revanche, par leur structure concentrée sur les états du système modélisé, les AGs permettent une représentation détaillée des informations quantitatives au cours de l'attaque. L'analyse quantitative nécessite souvent des techniques probabilistes avancées, telles que les graphes d'attaque bayésiens ou des méthodes d'inférence approximative/exacte [19, 14]. Chaque nœud peut être associé à une probabilité de compromission et un impact, et les risques sont propagés à travers les chemins du graphe pour calculer des métriques globales comme la probabilité de compromis du système ou la perte attendue.

Dans la suite de ce rapport, nous nous concentrerons principalement sur les arbres d'attaque, en raison de leur popularité et de leur efficacité pour l'analyse qualitative des scénarios d'attaque.

## 3 Construire des arbres d'attaque pertinents

### 3.1 Approches de génération d'arbres d'attaque

Le survey [12] propose un panorama des différentes approches de génération automatique des modèles graphiques de sécurité. Ces méthodes diffèrent principalement par la manière dont les vulnérabilités du système sont détectées. On peut distinguer trois grandes catégories d'approches : l'analyse syntaxique, la recherche d'atteignabilité dans

un graphe, et le model checking. Ces méthodes prennent en entrée une description du système, ainsi qu'un objectif d'attaque, et produisent en sortie un arbre d'attaque représentant les différentes façons dont un attaquant peut atteindre cet objectif.

### 3.1.1 Génération d'AT par analyse syntaxique du système

La méthode introduite dans [22] est l'une des premières à proposer une génération automatique d'arbres d'attaque. Le système étudié est représenté dans le cadre du *Value Passing Calculus* [15], une extension du  $\pi$ -calculus, qui permet de décrire les composants du système comme des *processus* inter-communicants.

Dans cette approche, chaque interaction possible avec le système peut être interprétée comme une étape conditionnée par une forme de contrôle de sécurité. Autrement dit, certaines actions ne deviennent accessibles que si l'attaquant dispose des informations nécessaires.

Ces informations sont représentées par des *canaux de communication*. Accéder à un comportement particulier du système revient à posséder la connaissance des canaux appropriés, tout comme un attaquant doit posséder les bons secrets ou accès pour progresser dans son attaque.

Une fois le modèle du système et l'objectif de l'attaque spécifiés, une formule propositionnelle est inférée à l'aide d'une approche de type chaînage arrière (backward chaining). Cette formule dépend des propositions atomiques qui représentent la compromission ou la possession par l'attaquant de ressources, de secrets ou de capacités élémentaires du système. Un AT action-based est finalement inféré de cette formule.

Cette approche constitue une limite, car elle fournit une sous-approximation des attaques réelles due au niveau d'abstraction du  $\pi$ -calcul, qui modélise la connaissance des canaux mais pas l'usage répété de ceux-ci. L'analyse indique ainsi quels canaux doivent être devinés, sans prendre en compte le nombre d'interactions nécessaires, ce qui peut conduire à sous-estimer l'effort réel de l'attaque.

Bien que la complexité théorique soit exponentielle dans le pire des cas, ce coût dépend de la structure du processus et n'est pas systématique, contrairement à la vérification de modèles basée sur l'exploration de l'espace d'états.

### 3.1.2 Génération d'AT par atteignabilité dans un graphe

On peut regrouper les approches de [5], et du projet européen TREsPASS [7, 6] dans cette catégorie.

La méthode présentée dans [5] prend en entrée un système décrit sous forme d'un graphe orienté, où une attaque est un chemin menant d'un noeud initial  $s$  à un noeud cible  $t$ . Tous les chemins de  $s$  à  $t$  sont extraits à l'aide d'une recherche en profondeur (depth-first search) depuis  $t$ , puis convertis en un arbre d'attaque en regroupant les sous-chemins communs.

Le projet TREsPASS [7, 6] propose une approche similaire, mais en labellisant les noeuds

du graphe avec des propriétés informations supplémentaires. Un nœud peut représenter *endroit*, un *acteur*, un *processus* ou un *objet*, et peut contenir des *ressources*, qui peuvent être des *objets*, ou des *données*. Les conditions selon lesquelles des actions peuvent être réalisées par des acteurs ou des processus sont définies par des politiques. Une politique peut exiger des identifiants, constitués de données, d'objets ou de prédictats.

Les arbres d'attaque générés dans ce travail décrivent les moyens par lesquels un attaquant peut invalider une politique dans un système donné. Leur construction repose sur une invalidation récursive des politiques [10, 9], consistant à identifier les acteurs potentiels ainsi que les couples action-localisation conduisant à l'invalidation de la politique.

Ces deux approches identifient les scénarios d'attaque par exploration de chemins dans un graphe représentant le système. Cependant, elles restent limitées en termes de généralisation. Elles nécessitent un graphe du système dans un formalisme spécifique, souvent différent d'un travail à l'autre, ce qui restreint la réutilisabilité des modèles. De plus, leur complexité est exponentielle dans le pire des cas, en raison de l'exploration exhaustive des chemins possibles entre les nœuds source et cible.

### 3.1.3 Génération d'AT par model checking

Parler des 3 catégories d'approches de génération d'arbres d'attaque [12]

[4] : Règles de raffinement déduites du système, mais non labellisées par des objectifs réels. Approche de génération top-down permise par leurs règles générées mais ne marche pas si un expert spécifie une librairie. Limitée à OR et SAND

## 3.2 Guided design of attack trees

Parler des techniques permettant de créer un arbre d'attaque correct et utile [3, 2]

Ouvrir sur la sous-section suivante – comment raffiner une feuille quand on sait qu'elle est utile.

## 3.3 Library-based attack tree synthesis

Présentation de l'algorithme, approche bottom up [18]

Parler de l'extension de cette approche avec le langage étendu au shuffle.

# 4 Attack Trees for Information Systems

## 4.1 Manual modeling of information systems and attacker capabilities

Parler de la spécification des systèmes sous forme d'Asset-Based Systems (ABS).

Parler des pouvoirs supplémentaires pouvant être accordés à l'attaquant symbolique.

## 4.2 Building libraries from databases of known attacks

Peut-on créer des librairies pour systèmes d'informations depuis les bases de données d'attaques connues (CAPEC, CVE, MITRE ATT&CK) ?

### **4.3 Towards automatic attack tree generation from logs**

Méthode de labellisation semi-automatique de logs réseaux et systèmes avec des techniques de la MITRE ATT&CK database [11]

## **Conclusion**

## References

- [1] Maxime Audinot. “Assisted design and analysis of attack trees”. 2018REN1S082. PhD thesis. 2018. URL: <http://www.theses.fr/2018REN1S082/document>.
- [2] Maxime Audinot, Sophie Pinchinat, and Barbara Kordy. “Guided Design of Attack Trees: A System-Based Approach”. In: *31st IEEE Computer Security Foundations Symposium, CSF 2018, Oxford, United Kingdom, July 9-12, 2018*. IEEE Computer Society, 2018, pp. 61–75. DOI: 10.1109/CSF.2018.00012. URL: <https://doi.org/10.1109/CSF.2018.00012>.
- [3] Maxime Audinot, Sophie Pinchinat, and Barbara Kordy. “Is My Attack Tree Correct?” In: *Computer Security - ESORICS 2017 - 22nd European Symposium on Research in Computer Security, Oslo, Norway, September 11-15, 2017, Proceedings, Part I*. Ed. by Simon N. Foley, Dieter Gollmann, and Einar Snekkenes. Vol. 10492. Lecture Notes in Computer Science. Springer, 2017, pp. 83–102. DOI: 10.1007/978-3-319-66402-6\\_7. URL: [https://doi.org/10.1007/978-3-319-66402-6%5C\\_7](https://doi.org/10.1007/978-3-319-66402-6%5C_7).
- [4] Olga Gadyatskaya et al. “Refinement-Aware Generation of Attack Trees”. In: Sept. 2017, pp. 164–179. ISBN: 978-3-319-68062-0. DOI: 10.1007/978-3-319-68063-7\_11.
- [5] Jin Bum Hong, Dong Seong Kim, and Tadao Takaoka. “Scalable Attack Representation Model Using Logic Reduction Techniques”. In: *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*. 2013, pp. 404–411. DOI: 10.1109/TrustCom.2013.51.
- [6] Marieta Ivanova et al. “Transforming Graphical System Models to Graphical Attack Models”. In: vol. 9390. Feb. 2016, pp. 82–96. ISBN: 978-3-319-29967-9. DOI: 10.1007/978-3-319-29968-6\_6.
- [7] Marieta Georgieva Ivanova et al. “Attack Tree Generation by Policy Invalidation”. In: *Information Security Theory and Practice*. Heraklion Crete: Springer-Verlag, 2015, pp. 249–259. ISBN: 978-3-319-24017-6. DOI: 10.1007/978-3-319-24018-3\_16. URL: [https://doi.org/10.1007/978-3-319-24018-3\\_16](https://doi.org/10.1007/978-3-319-24018-3_16).
- [8] Ravi Jhawar et al. “Attack Trees with Sequential Conjunction”. In: *CoRR* (2015). arXiv: 1503.02261. URL: <http://arxiv.org/abs/1503.02261>.
- [9] Florian Kammüller and Christian W. Probst. “Combining Generated Data Models with Formal Invalidation for Insider Threat Analysis”. In: *2014 IEEE Security and Privacy Workshops*. 2014, pp. 229–235. DOI: 10.1109/SPW.2014.45.
- [10] Florian Kammüller and Christian W. Probst. “Invalidating Policies using Structural Information”. In: *2013 IEEE Security and Privacy Workshops*. 2013, pp. 76–81. DOI: 10.1109/SPW.2013.36.

- [11] Sébastien Kilian et al. “CasinoLimit: An Offensive Dataset Labeled with MITRE ATT&CK Techniques”. In: *Proceedings of the 28th International Symposium on Research in Attacks, Intrusions and Defenses*. Gold Coast, Australia, Oct. 2025. URL: <https://hal.science/hal-05224264>.
- [12] Alyzia-Maria Konsta et al. “Survey: Automatic generation of attack trees and attack graphs”. In: *Computers & Security* 137 (2024), p. 103602. ISSN: 0167-4048. DOI: <https://doi.org/10.1016/j.cose.2023.103602>. URL: <https://www.sciencedirect.com/science/article/pii/S0167404823005126>.
- [13] Rajesh Kumar, Enno Ruijters, and Mariëlle Stoelinga. “Quantitative Attack Tree Analysis via Priced Timed Automata”. In: *International Conference on Formal Modeling and Analysis of Timed Systems*. 2015. URL: <https://api.semanticscholar.org/CorpusID:13738717>.
- [14] Luis Muñoz-González et al. “Efficient Attack Graph Analysis through Approximate Inference”. In: *ACM Trans. Priv. Secur.* 20.3 (July 2017). ISSN: 2471-2566. DOI: 10.1145/3105760. URL: <https://doi.org/10.1145/3105760>.
- [15] Hanne Riis Nielson, Flemming Nielson, and Roberto Vigo. “A Calculus for Quality”. In: *Formal Aspects of Component Software, 9th International Symposium, FACS 2012, Mountain View, CA, USA, September 12-14, 2012. Revised Selected Papers*. Ed. by Corina S. Pasareanu and Gwen Salaün. Vol. 7684. Lecture Notes in Computer Science. Springer, 2012, pp. 188–204. ISBN: 978-3-642-35861-6. DOI: 10.1007/978-3-642-35861-6\_12. URL: [http://dx.doi.org/10.1007/978-3-642-35861-6\\_12](http://dx.doi.org/10.1007/978-3-642-35861-6_12).
- [16] Cynthia Phillips and Laura Painton Swiler. “A graph-based system for network-vulnerability analysis”. In: *Proceedings of the 1998 Workshop on New Security Paradigms*. NSPW ’98. Charlottesville, Virginia, USA: Association for Computing Machinery, 1998, pp. 71–79. ISBN: 1581131682. DOI: 10.1145/310889.310919. URL: <https://doi.org/10.1145/310889.310919>.
- [17] Sophie Pinchinat, Mathieu Acher, and Didier Vojtisek. “Towards Synthesis of Attack Trees for Supporting Computer-Aided Risk Analysis”. In: vol. 8938. Sept. 2014. ISBN: 978-3-319-15200-4. DOI: 10.1007/978-3-319-15201-1\_24.
- [18] Sophie Pinchinat, François Schwarzenbuber, and Sébastien Lê Cong. “Library-Based Attack Tree Synthesis”. In: *Graphical Models for Security - 7th International Workshop, GramSec 2020, Boston, MA, USA, June 22, 2020 Revised Selected Papers*. Ed. by Harley Eades III and Olga Gadyatskaya. Vol. 12419. Lecture Notes in Computer Science. Springer, 2020, pp. 24–44. DOI: 10.1007/978-3-030-62230-5\_2. URL: [https://doi.org/10.1007/978-3-030-62230-5%5C\\_2](https://doi.org/10.1007/978-3-030-62230-5%5C_2).
- [19] Nayot Poolsappasit, Rinku Dewri, and Indrajit Ray. “Dynamic Security Risk Management Using Bayesian Attack Graphs”. In: *IEEE Transactions on Dependable and Secure Computing* 9.1 (2012), pp. 61–74. DOI: 10.1109/TDSC.2011.34.
- [20] Bruce Schneier. “Attack Trees: Modeling Security Threats”. In: *Dr. Dobb’s Journal* (Dec. 1999).

- [21] O. Sheyner et al. “Automated generation and analysis of attack graphs”. In: *Proceedings 2002 IEEE Symposium on Security and Privacy*. 2002, pp. 273–284. DOI: 10.1109/SECPRI.2002.1004377.
- [22] Roberto Vigo, Flemming Nielson, and Hanne Riis Nielson. “Automated Generation of Attack Trees”. In: *2014 IEEE 27th Computer Security Foundations Symposium*. 2014, pp. 337–350. DOI: 10.1109/CSF.2014.31.
- [23] Wojciech Widel et al. “Beyond 2014: Formal Methods for Attack Tree-based Security Modeling”. In: *ACM Comput. Surv.* 52.4 (2019), 75:1–75:36. DOI: 10.1145/3331524. URL: <https://doi.org/10.1145/3331524>.