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

INSA

Abstract: write your abstract here

Table des matières

Introduction

1 Notations

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 $\mathcal{F}$ 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 $\lambda : S \rightarrow 2^{\mathcal{F}}$ 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 : les traces basées sur les actions (ou action-based), comme définies dans [13] et les traces basées sur les états (ou state-based), comme présentées dans [3, 10, 2].

Définition 3 (Taces 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, modélisé par l'ensemble $\rightarrow$ des transitions du LTS.*

Définition 4 (Traces state-based)**.** *Une trace state-based est une séquence finie de vulations $\nu_1 \nu_2 \dots \nu_n$ où chaque valuation ν_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 ou les graphes d'attaque, 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, facilitant la compréhension et la communication entre les parties prenantes, y compris les experts en sécurité, les développeurs et les gestionnaires.

— **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

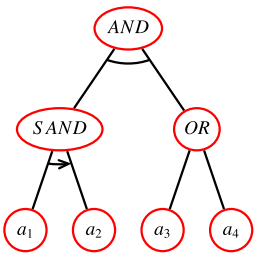

FIGURE 1 – Un AT action-based [1]

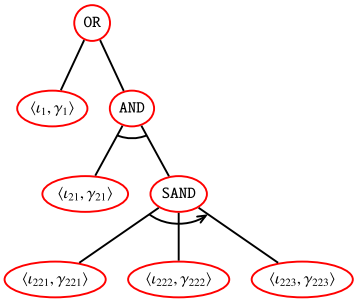

FIGURE 2 – Un AT state-based [1]

Les arbres d'attaque (AT) sont des structures arborescentes où chaque nœud représente un objectif d'attaque, et les feuilles représentent des actions élémentaires que l'attaquant peut entreprendre pour atteindre ces objectifs. Leur but est de mettre en évidence 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.
Les figures 1 et 2 présentent un exemple d'arbre d'attaque action-based (à gauche), et state-based (à droite). Cet exemple montre comment une attaque peut être décomposé en sous-objectifs, reliés par des opérateurs logiques `AND`, `OR` et `SAND`.

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

— **Basé sur les actions (Action-based)** : où les traces de l'arbre sont une séquence d'actions de l'attaquant. Cette approche met l'accent sur les transitions d'états du système.
Ainsi, une feuille de l'arbre, correspondant à un ensemble contenant une trace composée d'une seule action élémentaire de l'attaquant, qui est modélisée par une transition entre états du système.

— **Basé sur les états (State-based)** : introduite dans [3] où les traces de l'arbre sont une séquence de valuations sur l'état du système. Cette approche se concentre sur les conditions du système avant et après les actions de l'attaquant.

2.2.2 Graphes d'attaque

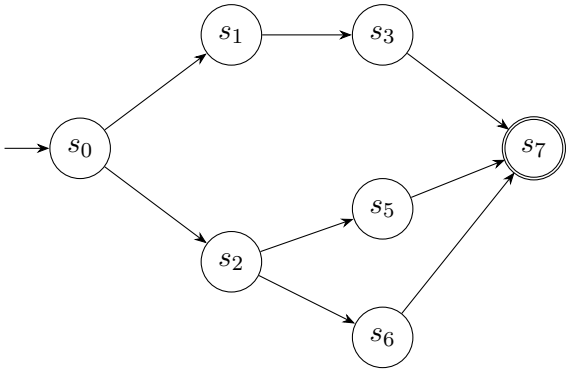

FIGURE 3 – Exemple d'un AG

Dans [12] un graphe d'attaque (AG) est défini comme 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 3 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 [9], modélise un graphe d'attaque comme un tuple (S, δ, S_0, S_s), où $\delta : S \times \mathbb{B} \rightarrow S$ est une fonction de transition étiquetée par des éléments de $\mathbb{B}$, représentant les actions de l'attaquant. Cette formulation impose que chaque arête soit explicitement associée à une action spécifique, tandis que la première autorise un niveau d'abstraction plus élevé en ne contraignant pas les transitions à être directement liées à une seule action.

2.3 Comparaison entre arbres et graphes d'attaque

2.3.1 Traduction d'ATs en AGs

Soit un AT τ défini sur un LTS $(S, \rightarrow, \lambda)$.
Dans [3], un arbre d'attaque τ induit un automate $\mathcal{A}_\tau = (Q, A, \delta, I, F)$ tel que les traces admisses par $\mathcal{A}_\tau$ coïncide avec l'ensemble des traces admissibles de τ, 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 τ. $G_\tau = (Q, \rightarrow_G, I, F)$, où la relation de transition $\rightarrow_G$ est induite par la fonction de transition δ.
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 [9], 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 $\mathbb{B}$;
2. Une hiérarchie d'actions de haut niveau $\mathbb{H} = (\{\mathcal{H}_k\}_{0 \leq k \leq K}, \mathcal{R})$, où :
   - $\mathcal{H}_0 = \mathbb{B}$ 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 de niveau inférieur.

Et produit en sortie un arbre d'attaque τ_G, labéllé par des actions de $\mathcal{H}$, tel que le langage reconnu par τ_G coïncide avec l'ensemble des traces admissibles de G.
Cependant, cette transformation requiert une intervention humaine 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 [14] offre un aperçu complet des différentes méthodes d'analyse quantitative appliquées aux ATs. Cependant, cela ne les empêchent pas de supporter des analyses quantitatives complexes, notamment basées sur des automates temporisés pondérés (priced timed automata) comme dans [7, 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 [11, 8]. 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

[6, 14] présentent un panorama des différentes approches de génération d'arbres d'attaque jusqu'à 2024. Les approches sont classées selon les entrées utilisées pour la génération des arbres d'attaque. 3 grandes catégories y sont identifiées : Model-Driven, Analysis-Driven, Vulnerability-Driven.

3.1.1 Approches basées sur des modèles (Model-Driven)

Les approches Model-Driven génèrent des arbress d'attaque avec pour seule entrée un modèle du système à analyser.
[13] propose une approche de génération d'arbres d'attaque en utilisant l'algèbre de processus.

3.1.2 Approches basées sur l'analyse (Analysis-Driven)

3.1.3 Approches basées sur les vulnérabilités (Vulnerability-Driven)

Parler des 3 catégories d'approches de génération d'arbres d'attaque [6]

- Analysis-Driven
  INPUT : Description du système + propriétés de sécurité
- Vulnerability-Driven
  INPUT : Description du système + informations sur sa vulnérabilité
  [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 [10]
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 [5]

Conclusion

Références

[1] Maxime Audinot. « Assisted design and analysis of attack trees ». 2018REN1S082. Thèse de doct. 2018. URL : http://www.theses.fr/2018REN1S082/document.

[2] Maxime Audinot, Sophie Pinchinat et 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, p. 61-75. DOI : 10.1109/CSF.2018.00012. URL : https://doi.org/10.1109/CSF.2018.00012.

[3] Maxime Audinot, Sophie Pinchinat et 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*. Sous la dir. de Simon N. Foley, Dieter Gollmann et Einar Snekkenes. T. 10492. Lecture Notes in Computer Science. Springer, 2017, p. 83-102. DOI : 10.1007/978-3-319-66402-6_7. URL : 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, p. 164-179. ISBN : 978-3-319-68062-0. DOI : 10.1007/978-3-319-68063-7_11.

[5] Sébastien Kilian et al. « CasinoLimit : An Offensive Dataset Labeled with MITRE ATT&amp ;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.

[6] 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.

[7] Rajesh Kumar, Enno Ruijters et 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.

[8] Luis Muñoz-González et al. « Efficient Attack Graph Analysis through Approximate Inference ». In : *ACM Trans. Priv. Secur.* 20.3 (juill. 2017). ISSN : 2471-2566. DOI : 10.1145/3105760. URL : https://doi.org/10.1145/3105760.

[9] Sophie Pinchinat, Mathieu Acher et Didier Vojtisek. « Towards Synthesis of Attack Trees for Supporting Computer-Aided Risk Analysis ». In : t. 8938. Sept. 2014. ISBN : 978-3-319-15200-4. DOI : 10.1007/978-3-319-15201-1_24.

[10] Sophie Pinchinat, François Schwarzentruber et 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*. Sous la dir. d'Harley Eades III et Olga Gadyatskaya. T. 12419. Lecture Notes in Computer Science. Springer, 2020, p. 24-44. DOI : 10.1007/978-3-030-62230-5_2. URL : https://doi.org/10.1007/978-3-030-62230-5%5C_2.

[11] Nayot Poolsappasit, Rinku Dewri et Indrajit Ray. « Dynamic Security Risk Management Using Bayesian Attack Graphs ». In : *IEEE Transactions on Dependable and Secure Computing* 9.1 (2012), p. 61-74. DOI : 10.1109/TDSC.2011.34.

[12] O. Sheyner et al. « Automated generation and analysis of attack graphs ». In : *Proceedings 2002 IEEE Symposium on Security and Privacy*. 2002, p. 273-284. DOI : 10.1109/SECPRI.2002.1004377.

[13] Roberto Vigo, Flemming Nielson et Hanne Riis Nielson. « Automated Generation of Attack Trees ». In : *2014 IEEE 27th Computer Security Foundations Symposium*. 2014, p. 337-350. DOI : 10.1109/CSF.2014.31.

[14] 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.