



Gestion de l'incohérence en détection d'intrusions

—
Délivrable n°03

Salem BENFERHAT
CRIL

Safa YAHI
CRIL



1 Introduction

La sécurité d'un système d'informations, qui figure parmi les enjeux majeurs de nos jours, revient à garantir la confidentialité et l'intégrité des données de ce système, ainsi que la disponibilité de ses services. Parmi les mécanismes mis au point en vue de sécuriser les systèmes d'informations, on peut distinguer les systèmes de prévention tels que l'authentification, dont l'objectif est de prouver l'identité des utilisateurs, le contrôle d'accès (RBAC [15], OR-BAC [1], etc), qui consiste à définir les droits accordés aux utilisateurs sur les données et les pare-feux dont le rôle est de filtrer l'accès aux services du système d'informations vis-à-vis du monde extérieur.

Cependant, ces mécanismes ne sont pas suffisants pour protéger les systèmes contre des attaques malveillantes ! En effet, les systèmes informatiques présentent souvent des vulnérabilités, c'est-à-dire des failles de conception, d'implémentation et de configuration, ce qui permet à des attaquants de contourner les mécanismes de prévention. De plus, un certain nombre de ces systèmes se focalisent sur la protection des attaques extérieures, alors qu'une grande partie des attaques proviennent de l'intérieur. Étant donné que les systèmes de prévention ne suffisent pas, une seconde couche de sécurisation s'avère nécessaire, à savoir la détection d'intrusions.

La détection d'intrusions [2] vise à détecter les attaques contre le système surveillé. Néanmoins, à son tour, elle souffre de certains problèmes. En effet, certaines attaques peuvent passer inaperçues, et dans ce cas, on parle de faux négatifs. En revanche, certaines alertes sont générées par rapport à des attaques qui n'ont pas eu lieu, ce qui correspond à des faux positifs.

Par ailleurs, la détection d'intrusion coopérative, qui implique plusieurs systèmes de détection d'intrusions ou IDSs (pour Intrusion Detection System) et d'autres systèmes (tels que les analyseurs réseaux, scanners de vulnérabilités, etc), offre de nombreux avantages. En effet, elle permet une vision globale du système surveillé, et donc des points de vue complémentaires. Toutefois, étant donné que les systèmes utilisés dans la coopération ne sont pas totalement fiables, souvent des conflits et des incohérences surviennent. De ce fait, il est indispensable de gérer ces incohérences afin d'exploiter cette coopération.

Ce livrable présente une nouvelle approche de corrélation d'alertes dans un contexte de détection d'intrusion coopérative qui permet de corréler les informations issues des différents systèmes utilisés, en vue de réduire le nombre d'alertes, en particulier les faux positifs.

L'idée de cette approche de corrélation est de raisonner, sans trivialisations, à partir des informations issues des différents systèmes utilisés conjointement dans le processus de surveillance. Par ailleurs, étant donné que les informations manipulées en détection d'intrusion sont de nature structurée (souvent exprimées en XML), nous proposons de les représenter en logiques de description. Les logiques de description ou DLs (pour Description Logics) sont bien adaptées pour représenter des informations structurées, et de plus, elles garantissent la décidabilité du raisonnement. Avant de décrire notre approche de corrélation, nous rappelons le principe de la détection d'intrusions. Ensuite, nous donnons une introduction aux logiques de description.

2 Principe de la détection d'intrusions

La détection d'intrusions est apparue au début des années 80, suite aux travaux de Anderson [2] et à ceux de Denning [7]. Un système de détection d'intrusions (IDS pour Intrusion Detection System) est destiné à détecter automatiquement des actions non autorisées ou malicieuses menées par des personnes internes ou externes au système d'informations surveillé.

Un IDS regroupe deux types de composants : un capteur et un analyseur. Alors que le capteur est chargé de collecter les données où se manifeste l'activité des utilisateurs, l'analyseur se charge de l'analyse, comme son nom l'indique, des événements produits par les capteurs afin de déterminer si une activité est intrusive ou non, autrement dit s'il s'agit d'une attaque.

Les IDSs peuvent être classifiés en général par rapport à deux critères, à savoir la nature des données analysées et la méthode d'analyse [6].

Ainsi, relativement à la source des données, on distingue trois catégories d'IDSs :

- les **IDSs réseaux ou NIDS** (pour Network-Based IDS) munis d'un dispositif matériel qui permet de capturer le trafic du réseau qu'ils surveillent,
- les **IDS systèmes ou HIDS** (pour Host-based IDS) qui analysent des données d'audit produites par le système d'exploitation des hôtes sur lesquels ils sont installés,
- les **IDS applicatifs** dont la source est les audits applicatifs. Donc, les données à analyser sont produites directement par une application, par exemple un serveur web, ftp, de base de données.

En ce qui concerne les méthodes d'analyse, on distingue d'une manière générale, deux catégories : les approches par signatures et les approches comportementales.

- Les **IDSs basés-approche par signatures** nécessitent une connaissance a priori sur les attaques détectées. Autrement dit, tout ce qui n'est pas explicitement interdit est autorisé. L'approche la plus fréquemment utilisée s'appuie sur des algorithmes de pattern matching. Les motifs recherchés dans les sources de données (chaîne de caractères, taille anormale, accès à un fichier système), censés caractériser des attaques sont appelés des signatures. D'autres approches ont été proposées, par exemple, Mé [13] propose d'appliquer des algorithmes génétiques pour analyser les audits systèmes. Lunt et al [12] introduisent une approche basée sur des systèmes experts. Parmi de tels IDSs, citons le NIDS Snort qui est actuellement le plus utilisé en détection d'intrusions.
- Pour les **IDSs basés-approche comportementale**, on définit préalablement un modèle de comportement normal ou autorisé du système surveillé. Au cours de la surveillance, le comportement courant de l'entité est mesuré, et toute déviation significative du comportement courant par rapport au comportement de référence génère une alerte. Contrairement à l'approche par signature, tout ce qui n'est pas explicitement autorisé est interdit. La

construction du modèle de référence se fait le plus souvent par apprentissage. Debar [5] propose par exemple d'utiliser des réseaux de neurones, Forrest et al [10] proposent pour leur part d'appliquer une approche immunologique. Des méthodes statistiques ont aussi été suggérées [9]. Le modèle peut aussi être construit à partir de spécifications de comportement d'applications ou par spécification de politiques de sécurité, comme dans l'approche de Zimmerman et al. [17].

Les logiques de description sont une famille de formalismes de représentation de connaissances qui permettent de représenter les connaissances d'un domaine d'application d'une manière structurée et formelle. Ces logiques sont issues de modèles graphiques de représentation de connaissances, notamment les réseaux sémantiques. En général, dans un réseau sémantique, on distingue des concepts (dénotés par des nœuds génériques), des individus (dénotés par des nœuds d'individus), des liens classe-fille/classe-mère et des liens de propriétés. En utilisant les liens classe-fille/classe-mère, les concepts peuvent être classés dans une hiérarchie, et en utilisant les liens de propriétés, des propriétés peuvent être assignées aux concepts. Toutefois, ces liens sont dépourvus de sémantique. De ce fait, ils peuvent être interprétés de différentes manières. Afin de pallier cette ambiguïté, d'autres formalismes ont été développés tels que les réseaux d'héritage structuré qui ont été implémentés dans le système KL-ONE [4]. Par la suite, KL-ONE a été doté d'une sémantique ce qui a fixé de manière précise la signification de ses constructeurs graphiques, et a conduit à la définition de la première logique de description [11], appelée aussi à cette époque langage terminologique, langage conceptuel ou encore langage basés KL-ONE.

D'un point de vue pratique, les logiques de description sont assez prometteuses. En fait, elles ont été utilisées dans l'implémentation de nombreuses applications dans divers domaines. Sans être exhaustifs, nous citons par exemple le Web Sémantique pour la représentation d'ontologies et la recherche d'information basée sur la logique, la médecine où l'objectif est la construction et la maintenance de très grandes ontologies médicales, les librairies numériques, les systèmes d'informations basés web, le traitement du langage naturel, la gestion de bases de données, le génie logiciel, etc.

2.1 Logique de description \mathcal{AL}

Une base de connaissances basée sur la logique de description¹ comprend deux composants, la TBox et la ABox. La TBox introduit la terminologie, c'est-à-dire le vocabulaire du domaine d'une application. Quant à la ABox, elle contient des assertions par rapport à des individus nommés en termes de ce vocabulaire.

Le vocabulaire consiste en des concepts, qui dénotent des ensembles d'individus, et des rôles qui désignent des relations binaires entre des individus. En plus des concepts atomiques et des rôles atomiques (les noms des concepts et des rôles), tous les systèmes DL permettent de construire des descriptions de concepts et de rôles plus complexes.

Les descriptions élémentaires sont les concepts atomiques et les rôles atomiques à partir desquels des descriptions complexes peuvent être générées via les constructeurs de concepts et les constructeurs de rôles. Dans ce qui suit, on

¹Pour une description plus complète des logiques de description (incluant IDMEF et M4D4, voir le livrable 03.

utilisera les lettres A et B pour désigner des concepts atomiques, la lettre R pour un rôle atomique et les lettres C et D pour les descriptions de concepts. Les logiques de description sont distinguées par rapport aux constructeurs qu'elles utilisent. La logique de description \mathcal{AL} (pour *Attributive Language*) a été introduite dans [16] comme étant la logique de description la moins expressive ayant un intérêt pratique. Les descriptions de concepts dans la logique \mathcal{AL} sont générées selon la règle syntaxique suivante:

$C, D \rightarrow$	$A \mid$	(concept atomique)
	$\top \mid$	(concept universel)
	$\perp \mid$	(concept bottom)
	$\neg A \mid$	(négation atomique)
	$C \sqcap D \mid$	(intersection)
	$\forall R.C \mid$	(restriction de valeur)
	$\exists R.\top$	(quantificateur existentiel limité)

Afin d'illustrer cette règle, prenons l'exemple suivant qui nous donne une idée sur ce qui peut être exprimé dans la logique \mathcal{AL} :

Exemple 1 Soient **Person** et **Female** deux concepts atomiques. Donc, $\text{Person} \sqcap \text{Female}$ et $\text{Person} \sqcap \neg \text{Female}$ sont des concepts décrivant intuitivement les personnes qui sont féminines et les personnes qui ne sont pas féminines. En outre, étant donné un rôle atomique **hasChild**, on peut construire les concepts $\text{Person} \sqcap \exists \text{hasChild}.\top$ et $\text{Person} \sqcap \forall \text{hasChild}.\text{Female}$ dénotant respectivement les personnes qui ont au moins un enfant et les personnes dont tous les enfants sont féminines. Enfin, en utilisant le concept \perp , on peut également décrire les personnes qui n'ont pas d'enfants par $\text{Person} \sqcap \forall \text{hasChild}.\perp$.

En vue de définir une sémantique formelle des concepts \mathcal{AL} , on considère une interprétation \mathcal{I} définie par la donnée d'un ensemble non vide $\Delta^{\mathcal{I}}$ (le domaine d'interprétation) et d'une fonction d'interprétation, qui assigne à chaque concept atomique A un ensemble $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ et à chaque rôle atomique R une relation binaire $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. Une fonction d'interprétation est étendue aux descriptions de concepts par la définition inductive suivante :

$$\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\
\perp^{\mathcal{I}} &= \emptyset \\
(\neg A)^{\mathcal{I}} &= \Delta^{\mathcal{I}} - A^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(\forall R.C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \forall b, (a, b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\} \\
(\exists R.\top)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \exists b, (a, b) \in R^{\mathcal{I}}\}
\end{aligned}$$

Deux concepts C et D sont dits équivalents, $C \equiv D$, si et seulement si $C^{\mathcal{I}} = D^{\mathcal{I}}$ pour toute interprétation \mathcal{I} . Par exemple, on peut aisément vérifier que les concepts $\forall \text{hasChild}.\text{Female} \sqcap \forall \text{hasChild}.\text{Student}$ et $\forall \text{hasChild}.\text{(Female} \sqcap \text{Student)}$ sont équivalents.

2.2 Au delà de la logique \mathcal{AL}

D'autres logiques, plus expressives, peuvent être définies en rajoutant d'autres constructeurs à la logique \mathcal{AL} à savoir:

- L'union de concepts, désignée par la lettre \mathcal{U} , notée par $C \sqcup D$ et interprétée par:

$$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}.$$

- La quantification existentielle complète, désignée par la lettre \mathcal{E} , notée par $\exists R.C$ et interprétée par:

$$(\exists R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \exists b, (a, b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}.$$

- Les restrictions de nombres, désignées par la lettre \mathcal{N} et notées par $\geq nR$ (restriction au moins) et par $\leq nR$ (restriction au plus) où n représente un entier positif. Ces restrictions de valeurs sont interprétées respectivement comme suit:

$$(\geq nR)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} : |\{b \mid (a, b) \in R^{\mathcal{I}}\}| \geq n\},$$

et

$$(\leq nR)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} : |\{b \mid (a, b) \in R^{\mathcal{I}}\}| \leq n\}.$$

- La négation de concepts arbitraires, désignée par la lettre (C) , notée par $\neg C$ et interprétée de la façon suivante :

$$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} - (C)^{\mathcal{I}}.$$

Exemple 2 Avec ces nouveaux constructeurs, on peut par exemple décrire les personnes ayant pas plus d'un enfant ou bien au moins trois enfants dont un est féminine comme suit:

$$\text{Person} \sqcap (\leq 1 \text{ hasChild} \sqcup (\geq 3 \text{ hasChild} \sqcap \exists \text{hasChild.Female})).$$

Étendre la logique \mathcal{AL} par n'importe quel sous ensemble des constructeurs précédents résulte en une nouvelle logique désignée par une chaîne de caractères de la forme :

$$\mathcal{AL}[\mathcal{U}][\mathcal{E}][\mathcal{N}][C],$$

où une lettre dans l'appellation reflète la présence du constructeur correspondant. Par exemple, $\mathcal{AL}\mathcal{E}\mathcal{N}$ est l'extension de \mathcal{AL} par la quantification existentielle complète et les restrictions de nombres.

D'un point de vue sémantique, on a $C \sqcup D \equiv \neg(\neg C \sqcap \neg D)$ et $\exists R.C \equiv \neg \forall R. \neg C$. Par conséquent, l'union et la quantification universelle complète peuvent être exprimées via la négation et vice versa. Ainsi, on utilisera la lettre \mathcal{C} au lieu des lettres \mathcal{UE} dans les noms des logiques. Par exemple, on écrit $\mathcal{AL}\mathcal{C}\mathcal{N}$ au lieu de $\mathcal{AL}\mathcal{U}\mathcal{E}\mathcal{N}$.

2.3 Les terminologies TBox

Une terminologie ou TBox en abrégé est un ensemble d'axiomes terminologiques. En général, les axiomes terminologiques sont de la forme

$$C \sqsubseteq D \quad (R \sqsubseteq S)$$

ou

$$C \equiv D \quad (R \equiv S)$$

où C, D sont des concepts (et R, S sont des rôles). Les axiomes du premier type sont appelés inclusions alors que ceux du second type sont dits équivalence.

Sémantiquement, une interprétation \mathcal{I} satisfait une inclusion $C \sqsubseteq D$ si et seulement si $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ et satisfait une équivalence $C \equiv D$ si et seulement si $C^{\mathcal{I}} = D^{\mathcal{I}}$. Si Σ est un ensemble d'axiomes, alors \mathcal{I} satisfait Σ si et seulement si \mathcal{I} satisfait chaque élément de Σ . Si \mathcal{I} satisfait un axiome (resp. un ensemble d'axiomes), alors \mathcal{I} est dite modèle de cet axiome (resp. l'ensemble d'axiomes). Deux axiomes ou deux ensembles d'axiomes sont équivalents si et seulement s'ils ont les mêmes modèles.

Une équivalence dont le membre gauche est un concept atomique est une définition. Les définitions sont utilisées pour affecter des noms symboliques à des descriptions complexes. Par exemple, par l'axiome:

$$\mathbf{Mother} \equiv \mathbf{Woman} \sqcap \exists \mathbf{hasChild}.\mathbf{Person}$$

on associe à la description $\mathbf{Woman} \sqcap \exists \mathbf{hasChild}.\mathbf{Person}$ le nom \mathbf{Mother} . Les noms symboliques peuvent être utilisés comme des abréviations dans d'autres descriptions. Si, par exemple, on définit le concept \mathbf{Father} par analogie au concept \mathbf{Mother} , on peut définir le concept \mathbf{Parent} comme suit:

$$\mathbf{Parent} \equiv \mathbf{Mother} \sqcup \mathbf{Father}.$$

On appelle un ensemble de définitions Σ une terminologie ou une TBox si aucun nom symbolique n'est défini pas plus d'une fois, c'est-à-dire que pour chaque concept atomique A , il existe au plus un axiome dans Σ dont le membre gauche est A .

Exemple 3 *La terminologie suivante exprime des concepts liés à des relations familiales:*

$$\begin{aligned} \mathbf{Woman} &\equiv \mathbf{Person} \sqcap \mathbf{Female} \\ \mathbf{Man} &\equiv \mathbf{Person} \sqcap \neg \mathbf{Woman} \\ \mathbf{Mother} &\equiv \mathbf{Woman} \sqcap \exists \mathbf{hasChild}.\mathbf{Person} \\ \mathbf{Father} &\equiv \mathbf{Man} \sqcap \exists \mathbf{hasChild}.\mathbf{Person} \\ \mathbf{Parent} &\equiv \mathbf{Mother} \sqcup \mathbf{Father} \\ \mathbf{GrandMother} &\equiv \mathbf{Mother} \sqcap \exists \mathbf{hasChild}.\mathbf{Parent} \\ \mathbf{MotherWithoutDaughter} &\equiv \mathbf{Mother} \sqcap \forall \mathbf{hasChild}.\neg \mathbf{Woman} \end{aligned}$$

Par ailleurs, les concepts d'inclusion sont utilisés pour exprimer l'appartenance d'un concept à un autre. On appelle une inclusion où le membre gauche est un concept atomique une spécialisation telle que $\mathbf{Woman} \sqsubseteq \mathbf{Person}$.

2.4 Les assertions ABox

Le second composant d'une base de connaissances exprimée en logiques de description est la description du monde ou la ABox. La ABox décrit un état spécifique du domaine en termes de concepts et de rôles. Plus précisément, dans une ABox, on introduit des individus (en leur donnant des noms) ainsi que leur propriétés. Les individus sont notés par a, b, c . Soit C un concept et R un rôle. On peut former des assertions comme suit: $C(a)$ et $R(b, c)$.

La première assertion, dite assertion de concept, signifie que a appartient à (l'interprétation de) C . La deuxième, appelée assertion de rôle, signifie que c est un remplisseur du rôle R pour b .

Par exemple, si PETER, PAUL et MARY sont des noms d'individus, alors **Father** (PETER) signifie que PETER est un père et **hasChild**(MARY, PAUL) signifie que PAUL est enfant de MARY. Une ABox notée \mathcal{A} est un ensemble fini de telles assertions.

Une ABox peut être vue comme étant une instance d'une base de données relationnelle avec seulement des relations unaires et binaires. Toutefois, contrairement à la sémantique du monde clos des bases de données classiques, la sémantique des ABox est une sémantique monde ouvert étant donné que les systèmes de représentation de connaissances sont appliqués dans des situations où l'on peut supposer que l'information est incomplète.

La sémantique des ABox est définie par l'extension des interprétations aux noms d'individus. Donc une interprétation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ n'assigne pas uniquement des ensembles et des relations binaires aux concepts et aux rôles mais aussi assigne à chaque nom d'individu a un élément $a^{\mathcal{I}}$ de $\Delta^{\mathcal{I}}$.

D'autre part, une interprétation \mathcal{I} satisfait l'assertion de concept $C(a)$ si $a^{\mathcal{I}} \in C^{\mathcal{I}}$. Elle satisfait l'assertion de rôle $R(a, b)$ si $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$ et elle satisfait une ABox \mathcal{A} si elle satisfait chaque assertion dans \mathcal{A} . Dans ce cas, \mathcal{I} est dite modèle de l'assertion ou de la ABox. Enfin, une interprétation \mathcal{I} satisfait une assertion ou une ABox par rapport à une terminologie Σ si en plus d'être modèle de l'assertion ou de la base, elle est également modèle de la terminologie Σ .

2.5 Raisonnement en logiques de description

Intuitivement, au lieu de concevoir de nouveaux algorithmes en DLs, on peut essayer de réduire le problème qu'on veut résoudre à un problème d'inférence connu en logique classique. Par exemple, la décidabilité du problème d'inférence en logique \mathcal{ALC} peut être obtenu en ayant la décidabilité de la logique \mathcal{L}^2 (un fragment de la logique des prédicats du premier ordre à deux variables) en sachant que tout concept \mathcal{ALC} peut être traduit dans la logique \mathcal{L}^2 . Prenons l'exemple suivant:

Exemple 4 Une traduction directe du concept $\forall R.(\exists R.A)$ génère la formule

$$\forall y.(R(x, y) \rightarrow (\exists z.(R(y, z) \wedge A(z))))).$$

Comme la sous formule $\exists z.(R(y, z) \wedge A(z))$ ne contient pas la variable x , cette dernière peut être réutilisée à la place de z . Ce renommage génère la formule équivalente suivante:

$$\forall y.(R(x, y) \rightarrow (\exists x.(R(y, x) \wedge A(x))))$$

qui utilise uniquement deux variables.

Toutefois, la complexité des procédures de décision obtenues ainsi est souvent plus élevée qu'il ne faut ce qui montre l'intérêt de développer de nouveaux algorithmes spécifiques aux logiques de description.

Par ailleurs, tous les problèmes d'inférence peuvent être réduits au problème de satisfiabilité en sachant que la logique de description en question permet la négation et la disjonction. Pour les logiques de description qui n'utilisent pas ces deux opérateurs en même temps la subsomption de concepts peut être calculée par des algorithmes de subsomption structurelle, c'est-à-dire des algorithmes qui comparent la structure syntaxique des descriptions de concepts.

Bien que ces algorithmes soient très efficaces, ils ne sont complets que pour des logiques simples peu expressives. En particulier, les logiques de description avec négation et disjonction ne peuvent pas être traitées par ce type d'algorithmes. En revanche, pour de telles logiques, les algorithmes basés-tableaux se trouvent très utiles. Le premier algorithme basé-tableaux en logique de description a été proposé par [16] dans l'optique de tester la satisfiabilité des concepts \mathcal{ALC} . Depuis, cette approche a été utilisée pour tester la satisfiabilité de nombreuses logiques de description qui étendent \mathcal{ALC} .

Les algorithmes de tableaux réduisent le problème de subsomption au problème de satisfiabilité. En fait, on sait que $C \sqsubseteq D$ si et seulement si $C \sqcap \neg D$ est insatisfiable. Enfin, citons quelques moteurs d'inférences basés sur les DLs tels que FaCT, Racer, Pellet, FaCT++, F-OWL.

3 Représentation en DLs des connaissances en détection d'intrusions

En détection d'intrusions, les informations que l'on manipule sont de nature très structurée. Souvent, ces informations sont représentées en XML. Par exemple, le format d'alertes IDMEF (pour Intrusion Detection Message Exchange Format) décrit les alertes en XML. Il est de même pour les descriptions des vulnérabilités données par OVAL (Open Vulnerability and Assessment Language). Néanmoins, XML est limité à une représentation syntaxique. Étant donné que cette représentation est dénuée de sémantique, un formalisme logique s'impose. Dans ce cas, la logique propositionnelle n'est pas vraiment adéquate, car elle ne permet pas de représenter les informations de manière structurée. D'où la nécessité d'aller au delà de cette logique en termes d'expressivité.

Nous proposons alors de considérer un fragment de la logique du premier ordre, à savoir les logiques de description. Le choix de telles logiques est justifié tout d'abord par le fait qu'elles conviennent à la représentation des informations structurées. Aussi, elles sont décidables. De plus, on dispose actuellement d'un nombre considérable de logiques de description variant en termes d'expressivité, et pour lesquelles la complexité du raisonnement est bien connue. Par ailleurs, de nombreux raisonneurs en DLs ont été développés comme Fact++. La plupart de ces derniers utilisent des techniques d'optimisation sophistiquées. De ce fait, ces raisonneurs sont généralement efficaces en pratique, notamment par rapport à des problèmes réels.

Les informations nécessaires à notre approche de corrélation impliquent tout d'abord les alertes générées par les IDSs. Pour la description d'alertes, nous nous

sommes basés sur la description des alertes selon le format IDMEF, qui est largement utilisé en détection d'intrusions. Aussi, nous avons besoin d'informations contextuelles à savoir la topologie ainsi que la cartographie. Pour ce dernier point, nous nous sommes essentiellement inspirés du modèle M4D4 [14]. Enfin, notre approche de corrélation requiert la description des vulnérabilités. Pour ce faire, nous avons partiellement utilisé le modèle M4D4, tout en considérant d'autres sources de description de vulnérabilités, en particulier OVAL.

3.1 Représentation de l'IDMEF en logiques de description

L'IDMEF (pour Intrusion Detection Message Exchange Format) est un format d'alertes proposé par le groupe IDWG (Intrusion Detection Working Group). Il s'agit d'un modèle qui a été implémenté en XML. Une description détaillée de ce format et sur laquelle nous sommes basés est fournie par le RFC 4765². En particulier, une alerte en IDMEF admet les caractéristiques suivantes:

- Identifier : son identifiant,
- CreateTime : l'instant de la création de l'alerte par l'IDS,
- DetectTime : l'instant de la détection de l'attaque,
- AnalyseTime : l'heure courante de l'IDS,
- Analyser : l'IDS ayant généré cette alerte
- Source : désigne l'attaquant,
- Target : correspond à la victime,
- Classification : représente l'attaque,
- Assesement : indique par exemple la gravité de l'attaque,
- AdditionalData : ce champ peut comprendre tout type d'informations en dehors des informations précédentes.

Nous proposons de représenter le vocabulaire de l'IDMEF en logiques de description. Pour ce faire, nous utilisons une TBox construite à partir d'une vingtaine de concepts et d'une soixantaine de rôles. Cette TBox comprend des axiomes de définitions ainsi que des axiomes d'inclusion.

Par exemple, le concept d'une alerte est donné par la figure 1. Un tel axiome signifie qu'une alerte admet un seul identifiant qui est de type chaîne de caractères, un seul champ "detecttime" de type "time", un seul champ "create-time" de type "time", et au plus un seul champ "analysertime" de type "time". De plus, à une alerte peut correspondre une source, voire même plusieurs. De même, il peut lui correspondre une cible ou plusieurs. En outre, une alerte admet une seule classification, au plus un élément "asement" et un champ "additional data".

De la même manière, nous définissons par exemple le concept d'une source tel que décrit par la figure 2.

²<http://www.ietf.org/rfc/rfc4765.txt>

```

Alert  ⊆  ∀messageId.String ⊠ = 1 messageId ⊠
        ∀hasCreateTime.Time ⊠ = 1 hasCreateTime ⊠
        ∀hasDetectTime.Time ⊠ ≤ 1 hasDetectTime ⊠
        ∀hasAnalyserTime.Time ⊠ ≤ 1 hasAnalyserTime ⊠
        ∀hasAnalyser.Analyser ⊠ = 1 hasAnalyser ⊠
        ∀hasSource.Source ⊠
        ∀hasTarget.Target ⊠
        ∀hasClassification.Classification ⊠ = 1 hasClassification ⊠
        ∀hasAssesement.Assessment ⊠ ≤ 1 hasAssessment ⊠
        ∀hasAdditionalData.AdditionalData

```

Figure 1: Concept Alert

```

Source  ⊆  ∀sourceId.String ⊠ ≤ 1 sourceId ⊠
          ∀spoofed.{yes,no,unknown} ⊠ ≤ 1 spoofed ⊠
          ∀interface.String ⊠ ≤ 1 interface ⊠
          ∀hasNode.Node ⊠ ≤ 1 hasNode ⊠
          ∀hasProcess.Process ⊠ ≤ 1 hasProcess ⊠
          ∀hasUser.User ⊠ ≤ 1 hasUser ⊠
          ∀hasService.Service ⊠ ≤ 1 hasService

```

Figure 2: Concept Source

Considérons par exemple la figure 3 qui décrit une description IDMEF d'une alerte dans le format XML. Cet exemple est tiré du RFC 4765³ de l'IDMEF. La description de cet exemple en logiques de description génère une ABox qui comprend les assertions suivantes :

- **Différents concepts**

- Alert(ALR1)
- Analyser(ANL1)
- Source(SRC1)
- Target(TRG1)
- Classification(CLS1)
- Node(NOD1)
- Node(NOD2)
- Node(NOD3)
- Reference(RFC1)
- Address(ADR1)
- Address(ADR2)

- **L'alerte**

- messageId(ALR1, "abc123456789")
- hasCreateTime(ALR1, 2000-03-09T10:01:25.93464-05:00)

³<http://www.ietf.org/rfc/rfc4765.txt>

```

<?xml version="1.0" encoding="UTF-8"?>
<idmef:IDMEF-Message xmlns:idmef="http://iana.org/idmef"
  version="1.0">
  <idmef:Alert messageid="abc123456789">
    <idmef:Analyzer analyzerid="hq-dmz-analyzer01">
      <idmef:Node category="dns">
        <idmef:location>Headquarters DMZ Network</idmef:location>
        <idmef:name>analyzer01.example.com</idmef:name>
      </idmef:Node>
    </idmef:Analyzer>
    <idmef:CreateTime ntpstamp="0xbc723b45.0xef449129">
      2000-03-09T10:01:25.93464-05:00
    </idmef:CreateTime>
    <idmef:Source ident="a1b2c3d4">
      <idmef:Node ident="a1b2c3d4-001" category="dns">
        <idmef:name>badguy.example.net</idmef:name>
        <idmef:Address ident="a1b2c3d4-002"
          category="ipv4-net-mask">
          <idmef:address>192.0.2.50</idmef:address>
          <idmef:netmask>255.255.255.255</idmef:netmask>
        </idmef:Address>
      </idmef:Node>
    </idmef:Source>
    <idmef:Target ident="d1c2b3a4">
      <idmef:Node ident="d1c2b3a4-001" category="dns">
        <idmef:Address category="ipv4-addr-hex">
          <idmef:address>0xde796f70</idmef:address>
        </idmef:Address>
      </idmef:Node>
    </idmef:Target>
    <idmef:Classification text="Teardrop detected">
      <idmef:Reference origin="bugtraqid">
        <idmef:name>124</idmef:name>
        <idmef:url>http://www.securityfocus.com/bid/124</idmef:url>
      </idmef:Reference>
    </idmef:Classification>
  </idmef:Alert>

```

Figure 3: Exemple d'alerte

- hasAnalyzer(ALR1, ANL1)
- hasSource(ALR1, SRC1)
- hasTarget(ALR1, TRG1)
- hasClassification(ALR1, CLS1)

- **L'analyser**

- analyzerId(ANL1, "hq-dmz-analyzer01")
- hasNode(ANL1, NOD1)

- **Le noeud N1**

- category(NOD1, "dns")
- location(NOD1, "Headquarters DMZ Network")
- name(NOD1, analyzer01.example.com)

- **La source**

- hasNode(SRC1, NOD2)
- hasIdent(SRC1, a1b2c3d4-002)

- **Le noeud N2**

- hasIdent(NOD2, "a1b2c3d4-001")
- category(NOD2, "dns")
- name(NOD1, badguy.example.net)
- hasAddress(NOD2, ADR1)

- **L'adresse ADR1**

- hasIdent(ADR1, "a1b2c3d4-002")
- category(ADR1, "ipv4-net-mask")
- address(ADR1, 192.0.2.50)
- netmask(ADR1, 255.255.255.255)

- **Le nœud N3**

- hasIdent(NOD3, "d1c2b3a4")
- category(NOD3, "dns")
- hasAddress(NOD3, ADR2)

- **La cible**

- hasIdent(TRG1, "d1c2b3a4")
- hasNode(TRG1, NOD3)

- **L'adresse ADR2**

- category(ADR2, "ipv4-addr-hex")
- address(ADR2, 0xde796f70)

- **La classif**

- text(CLS1, "Teardrop detected")
- hasReference(CLS1, RFC1)

- **La reference**

- origin(RFC1, "bugtraqid")
- name(RFC1, 124)
- url(RFC, <http://www.securityfocus.com/bid/124>)

3.2 Topologie

Décrire la topologie permet par exemple de déduire si un IDS est capable ou non de détecter une alerte. La topologie concerne les nœuds ainsi que leurs interconnexions. Dans le modèle M4D4, on considère que chaque réseau admet une seule adresse. Nous traduisons cette information en DLs de la façon suivante :

$$\text{Network} \sqsubseteq \forall \text{netaddress}.\text{String} \sqcap = 1 \text{ netaddress}$$

Les nœuds représentent n'importe quelle machine connectée au réseau. Un nœud admet une adresse et appartient à un réseaux.

$$\text{Node} \sqsubseteq \forall \text{nodeaddress.String} \sqcap = 1 \text{ nodeaddress} \sqcap \\ \forall \text{hasNodeNet.Network}$$

Les passerelles sont des nœuds particuliers dont l'objectif est de connecter des réseaux. Clairement, une passerelle appartient à plus d'un réseau.

$$\text{Gateway} \sqsubseteq \text{Node} \sqcap > 1 \text{ hasNodeNet} \\ \text{Node} \sqcap \neg \text{Gateway} \sqsubseteq = 1 \text{ hasNodeNet}$$

Les passerelles directement accessibles par un nœud sont désignées par:

$$\text{Node} \sqsubseteq \forall \text{hasNodeGateway.Gateway} \sqcap \\ \forall \text{hasNodeSystemname.String} \sqcap = 1 \text{ hasNodeSystemname}$$

Enfin, un nœud peut être en plus caractérisé par son nom système:

$$\text{Node} \sqsubseteq \forall \text{hasNodeSystemName.String} \sqcap = 1 \text{ hasNodeSystemName}$$

3.3 Cartographie en DL

Selon le modèle M4D4, un produit est caractérisé par un seul nom, une seule version, un seul type et par une seule architecture. En logiques de description, ceci correspond au concept produit que nous exprimons en DLs de la manière suivante :

$$\text{Software} \sqsubseteq \forall \text{softwareName.String} \sqcap = 1 \text{ softwareName} \sqcap \\ \forall \text{softwareVersion.String} \sqcap = 1 \text{ softwareVersion} \sqcap \\ \forall \text{softwareType.String} \sqcap = 1 \text{ softwareType} \sqcap \\ \forall \text{softwareArchitecture.String} \sqcap = 1 \text{ softwareArchitecture} \sqcap$$

Un nœud héberge un logiciel:

$$\text{Node} \sqsubseteq \forall \text{hosts.Software}$$

Un processus est un produit exécuté par un utilisateur.

$$\text{Process} \sqsubseteq \forall \text{hasSoftware.Software} \sqcap = 1 \text{ hasProduct} \sqcap \\ \forall \text{hasUser.User} \sqcap = 1 \text{ hasUser}$$

Un service est un processus qui écoute sur un port:

$$\text{Service} \sqsubseteq \forall \text{hasProcess.Process} \sqcap = 1 \text{ hasProcess} \sqcap \\ \forall \text{port.Integer} \sqcap = 1 \text{ port}$$

3.4 Les vulnérabilités en DLs

En général, telle que décrite par exemple dans M4D4, une vulnérabilité est caractérisée par son degré de sévérité, le niveau d'accès nécessaire afin de l'exploiter, ses conséquences et sa date de publication. Nous représentons alors le concept vulnérabilité en DL comme suit :

Vulnerability \sqsubseteq \forall severity. $\{high, medium, low\}$
 \forall requires. $\{remote, local, user\}$
 \forall losstype. $\{confidentiality, integrity, availibility, privilege_escalation\}$
 \forall published.Date

Dans le modèle M4D4, on considère qu'une vulnérabilité affecte tout simplement une liste de produits. Cette liste est appelée configuration. Par ailleurs, les caractéristiques de vulnérabilité peuvent être extraites de plusieurs sources. Par exemple, la base de données NVD⁴ (National Vulnerability Database), la base de données OSVDB⁵ Open Source Vulnerability Database ainsi que le projet OVAL⁶ (Open Vulnerability and Assesment Language) sont des initiatives indépendantes visant à structurer les informations relatives aux vulnérabilités. En analysant de plus près ces sources, en particulier OVAL qui est un standard, nous avons constaté que le fait qu'un nœud soit affecté d'une vulnérabilité revient généralement à vérifier des conditions logiques plus compliquées impliquant des opérateurs de conjonction, de disjonction et de négation. Par exemple, considérons la description (selon OVAL) de la vulnérabilité CVE-2008-0082 donnée par la figure 4. Dans cette description, on voit clairement que la condition de la vulnérabilité est donnée sous forme d'une formule logique composée.

Definition Id: oval.org.mitre.oval.def:5995		Date: 2008-09-19	
Title:	Windows Messenger Information Disclosure Vulnerability		
Description:	An ActiveX control (Messenger.UIAutomation.1) in Windows Messenger 4.7 and 5.1 is marked as safe-for-scripting, which allows remote attackers to control the Messenger application, and "change state," obtain contact information, and establish audio or video connections without notification via unknown vectors.		
Version:	1	Class:	vulnerability
Status:	ACCEPTED	Reference(s):	CVE-2008-0082
Family:	windows		
Platform(s):	Microsoft Windows 2000 Microsoft Windows XP Microsoft Windows Server 2003	Product(s):	Windows Messenger 4.7 Windows Messenger 5.1
Definition Synopsis:			
<ul style="list-style-type: none"> • Windows Messenger 4.7 is installed • AND the version of msgsc.dll is less than 4.7.0.3002 • OR • Windows Messenger 5.1 is installed • AND the version of msgsc.dll is less than 5.1.0.715 			

Figure 4: Vulnérabilité CVE-2008-0082

Par conséquent, la notion de configuration proposée dans M4D4 n'est pas suffisante afin de décrire une vulnérabilité. Nous proposons alors de tirer profit des sources précédentes pour décrire les vulnérabilités.

Ainsi, le fait qu'une machine soit par exemple vulnérable relativement à CVE-2008-0082 est décrit comme suit :

$$\exists$$
vulnerableTo. $(\exists$ hasReference.CVE – 2008 – 0082) \sqsubseteq
 $(\exists$ host. $(\exists$ hasName.WindowsMessenger4.7) \sqcap
 \exists host. $((\exists$ hasName.msgsc.dll) \sqcap $(\exists$ hasVersion. \leq 4.7.0.3002))) \sqcup
 $(\exists$ host. $(\exists$ hasName.WindowsMessenger5.1) \sqcap
 \exists host. $((\exists$ hasName.msgsc.dll) \sqcap $(\exists$ hasVersion. \leq 5.1.0.715)))

⁴<http://nvd.nist.gov/>

⁵<http://www.osvdb.org>

⁶<http://oval.mitre.org/>

4 Gestion de l'incohérence en détection d'intrusions

L'approche de corrélation d'alertes que nous proposons se situe dans un cadre d'une détection d'intrusions coopérative. Dans ce cas, plusieurs IDSs et d'autres analyseurs tels que les scanners de vulnérabilités et réseaux sont déployés à travers le réseau, et coopèrent dans le processus de détection. L'idée derrière est d'avoir une vision globale et une meilleure couverture du réseau et aussi plusieurs points de vue qui peuvent être complémentaires.

De plus, nous considérons pour les attaques exploitant des vulnérabilités connues, une base de description de vulnérabilités pour vérifier à quel point une attaque signalée a vraiment eu lieu et ce dans le sens où la présence d'une vulnérabilité renforce la présence d'une attaque alors que son absence vérifie l'inverse.

Par ailleurs, il est bien connu que les dispositifs utilisés en détection d'intrusions ne sont pas totalement fiables. En effet, pour les IDSs, les faux positifs (fausses alertes) et les faux négatifs (attaques non détectées) en témoignent. En ce qui concerne les analyseurs réseaux, la récolte des informations se fait en général hors ligne. De ce fait, à un moment donné, la configuration du réseau peut changer sans pour autant mettre à jour les informations de l'analyseur réseau. De plus, en général, on a affaire à des informations incomplètes : l'analyseur est incapable de déterminer la version de tel ou tel logiciel.

Par conséquent, la coopération dans ce cas peut facilement générer des incohérences. Par exemple, un IDS alarme quant à l'existence d'une attaque envers un système exploitant une certaine vulnérabilité alors que le scanner réseau dit que la cible n'est pas vulnérable. Un autre exemple est qu'un IDS signale une attaque alors qu'un autre IDS, qui se trouve capable de détecter une telle attaque selon sa visibilité topologique et fonctionnelle, ne le fait pas.

Nous proposons alors de résoudre les conflits issus de cette coopération par le biais d'une approche logique de raisonnement en présence d'incohérence. Naturellement, nous adoptons une approche basée sur la restauration de la cohérence [3]. En particulier, nous considérons le cadre de bases de croyances partiellement préordonnées. En effet, en détection d'intrusions, certains dispositifs sont comparables entre eux. A titre d'exemple, citons le travail de thèse [8] qui consiste à évaluer, et donc à comparer des IDSs. Néanmoins, d'autres dispositifs ne le sont pas : comparer un IDS et les informations fournies par un scanner réseau, surtout si on tient compte de la dynamique du réseau n'a pas de sens. Par conséquent, on voit clairement la nécessité de considérer des bases de croyances partiellement préordonnées.

La question qui se pose maintenant est la suivante : quelle relation d'inférence parmi celles dédiées au raisonnement en présence d'incohérence à partir de bases de croyances partiellement préordonnées convient-il de choisir ? Ces relations d'inférence diffèrent essentiellement relativement à leur degré de prudence, notre réponse à cette question est de déléguer un tel choix à l'administrateur réseau. En effet, tout dépend de la nature du système sous surveillance et du degré de prudence qu'il souhaite assigner à son système. Autrement dit, il va falloir

trouver un compromis entre le taux de réduction d’alertes souhaité et le risque de passer à côté d’une vraie attaque (ce risque n’est pas nul car les informations déduites sont plutôt plausibles). Nous pensons par exemple que la surveillance d’un système dans un centre de recherche en énergie atomique nécessite l’approche la plus sceptique possible, par exemple les inférences possibilistes.

Pour résumer, les entrées de notre approche sont :

- les alertes générées par les différents IDSs,
- la topologie ainsi que la cartographie du réseau,
- les caractéristiques des vulnérabilités.

Le résultat est une base de croyances partiellement préordonnées. Plus précisément, une base de croyances exprimées en logiques de description. Ensuite, en appliquant une inférence à partir de bases de croyances partiellement préordonnées selon le degré de prudence requis par l’administrateur, on détermine la plausibilité qu’une certaine machine soit la cible ou non d’une éventuelle attaque signalée. Dans le cas où l’on détermine que cette attaque n’est pas plausible, l’alerte correspondante est éliminée (ou est affectée un degré faible) ce qui réduit le nombre d’alertes à traiter.

5 Cas d’utilisation

Illustrons l’approche de corrélation proposée avec un cas d’utilisation simple. Pour ce faire, nous considérons un système de détection d’intrusions coopérative où sont impliqués deux IDSs de type Snort I_A et I_B en plus d’un scanner de réseaux S . Supposons que l’IDS I_A a le même degré de fiabilité que l’IDS I_B . En outre, ces deux IDSs sont incomparables par rapport au scanner de réseaux S .

Supposons maintenant ce qui suit :

- L’IDS I_A génère une alerte signalant une attaque A ayant pour nom ”NET-BIOS DCERPC ISystemActivator bind attempt” contre la machine M . La traduction de cette information en DL donne l’assertion suivante :

$$A_1 : \text{hasName}(A, \text{”NETBIOS DCERPC ISystemActivator bind attempt”}) \\ \sqcap \text{attackedBy}(M, A).$$

- L’IDS I_B n’a pas généré d’alerte relative à l’attaque A par rapport à la machine M après un certain temps t ce qui signifie que cette alerte ne sera jamais émise. En DL, on obtient :

$$A_2 : \neg \text{attackedBy}(M, A). \tag{1}$$

- D’après le scan effectué par l’analyseur réseaux S , la machine M héberge Microsoft Windows NT ainsi que le Patch Q823980. La description correspondante en DL est donnée par l’assertion A_3 :

$$A_3 : \text{hasName}(S_1, \text{”MicrosoftWindowsNT”}) \sqcap \text{host}(M, S_1) \\ \sqcap \text{hasName}(S_2, \text{”PatchQ823980”}) \sqcap \text{host}(M, S_2).$$

- Par ailleurs, on sait d'une manière certaine, selon la documentation de Snort, plus précisément selon la signature suivante :

```

alert tcp EXTERNAL_NET any -i HOME_NET 135 ("msg:NETBIOS
DCERPC ISystemActivator bind attempt"; flow:to_server,established;
content:"—05—"; distance:0; within:1; content:"—0b—"; distance:1; within:1;
byte_test:1,&,1,0,relative; content:"—A0 01 00 00 00 00 00 00 C0 00 00 00
00 00 00 46—"; distance:29; within:16; reference:cve,CAN-2003-0352;
classtype:attempted-admin; sid:2192; rev:1;)

```

que l'attaque A exploite une vulnérabilité ayant pour référence CVE 2003-0352.

Nous exprimons cette information en DL via l'axiome d'inclusion suivant :

$$T_1 : \exists attackedBy. (\exists hasName. NETBIOSDCERPCISystemActivatorbindattempt)$$

$$\sqsubseteq \exists vulnerableTo. (\exists hasReference. CVE2003 - 0352)$$

- Enfin, selon OVAL, une machine est vulnérable par rapport à CVE 2003-0352 si et seulement si elle vérifie les conditions suivantes :
 - La machine héberge Microsoft Windows NT.
 - La machine n'héberge pas le Patch Q823980.
 - La version de rpcss.dll sur cette machine doit être inférieure à 4.0.1381.7224.

En DL, ces conditions sont résumées via l'axiome d'équivalence suivant :

$$\begin{aligned}
T_2 : \exists vulnerableTo. (\exists hasReference. CVE2003 - 0352) &\sqsubseteq \\
\exists host. (\exists hasName. MicrosoftWindowsNT) &\sqcap \\
\neg \exists host. (\exists hasName. Patch Q823980) &\sqcap \\
\exists host. ((\exists hasName. rpcss.dll) \sqcap (\exists hasVersion. \leq 4.0.1381.7224)) &
\end{aligned}$$

En résumé, on obtient une base de croyances DL partiellement préordonnée $((T, A), \preceq)$ telle que :

- $T = \{T_1, T_2\}$,
- $A = \{A_1, A_2, A_3\}$.

Le préordre partiel \preceq sur cette base est décrit par la figure 5.

Voyons maintenant lesquelles des croyances $attackedBy(M, A)$ et $\neg attackedBy(M, A)$ est par exemple une conséquence p-lexicographique de (T, A) .

Tout d'abord, les sous-bases maximales cohérentes de $attackedBy(M, A)$ contenant les informations certaines T_1 et T_2 sont A et B telles que :

- $A = \{T_1, T_2, A_2, A_3\}$,

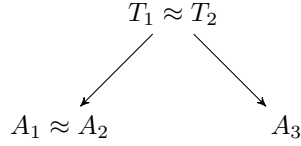


Figure 5: Le préordre partiel \preceq sur (T, A)

- $B = \{T_1, T_2, A_1\}$.

Il est clair que $A \prec_{plex} B$ ce qui implique que l'inférence p-lexicographique à partir (T, A) équivaut l'inférence classique à partir de la sous-base A . De ce fait, on infère $\neg attackedBy(M, A) : (T, A) \vdash_{plex} attackedBy(M, A)$ ce qui est intuitivement attendu. Par conséquent, l'alerte liée à cette attaque peut être supprimée, ce qui réduit le nombre d'alertes qui doivent être analysées et traitées par l'administrateur. Elle peut également se voir assigner un degré de plausibilité faible, ce qui permet de la traiter quand même par prudence, une fois toutes les attaques les plus plausibles traitées.

Néanmoins, ce résultat peut être considéré comme étant aventureux pour un administrateur réseau dans un contexte critique. Dans ce dernier cas, il peut opter par exemple pour une approche plus prudente telle que les extensions possibilistes qui ne permettent de rien déduire dans ce cas particulier, mais qui peuvent bien évidemment inférer des conséquences plus prudentes dans d'autres situations, en modifiant par exemple la relation de fiabilité entre les IDSs.

6 Conclusion

Dans ce livrable, nous avons tout d'abord montré la pertinence du raisonnement en présence d'incohérence, en particulier, à partir de bases de croyances partiellement préordonnées, en détection d'intrusions coopérative. Nous avons alors proposé une nouvelle approche logique de corrélation d'alertes où les informations sont exprimées en logiques de description. Cette approche est paramétrée par une des relations d'inférence à partir de bases de croyances partiellement préordonnées. Comme ces relations d'inférence diffèrent essentiellement par rapport à leur prudence, nous déléguons le choix de la relation d'inférence à l'administrateur réseau pour l'effectuer d'une manière subjective et contextuelle. L'étape suivante dans ce cadre consiste à implémenter et à expérimenter cette approche de corrélation d'alertes.

References

- [1] A. AbouElKalam, S. Benferhat, A. Miège, R. ElBaida, F. Cuppens, C. Saurel, P. Balbiani, Y. Deswarte, and G. Trouessin. Organization based access control. In *Policies for Distributed Systems and Networks (POLICY'03)*, pages 120–128, 2003.
- [2] J. P. Anderson. Computer security threat monitoring and surveillance. Technical Report 98–17, James P Anderson Co., FortWashington, Pennsylvania, USA, April 1980.

- [3] Salem Benferhat and Safa Yahi. Complexity and cautiousness results for reasoning from partially preordered belief bases. In *ECSQARU*, pages 817–828, 2009.
- [4] R. J. Brachman and J. G. Schmolze. An overview of the kl-one knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.
- [5] H. Debar. *Application des réseaux de neurones à la détection d'intrusions sur les systèmes informatiques*. PhD thesis, Université de Paris 6, 1993.
- [6] H Debar, M. Dacier, and Andreas Wespi. A revised taxonomy for intrusion-detection systems, 2000.
- [7] Dorothy E. Denning. An intrusion-detection model. *IEEE Trans. Softw. Eng.*, 13(2):222–232, 1987.
- [8] Mohammed El-Sayed Gadelrab. *Évaluation des système de détection d'intrusion*. Thèse de doctorat, Université Paul Sabatier, Toulouse, France, décembre 2008.
- [9] Teresa F. Lunt Ann Tamaru Mabry Tyson Harold S. Javitz, Alfonso Valdez and John Lowrance. Next generation intrusion detection expert system (nides) - 1. statistical algorithms rationale - 2. rationale for proposed resolver. technical report a016–rationales. In *SRI International*, 333 Ravenswood Avenue, Menlo Park, CA, 1993.
- [10] Steven A. Hofmeyr, Stephanie Forrest, and Anil Somayaji. Intrusion detection using sequences of system calls. *J. Comput. Secur.*, 6(3):151–180, 1998.
- [11] H. J. Levesque and R. J. Brachman. Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, 3:78–93, 1987.
- [12] Teresa F. Lunt and R. Jagannathan. A prototype real-time intrusion-detection expert system. *Security and Privacy, IEEE Symposium on*, 0:59, 1988.
- [13] L. Mé. *Audit de sécurité par algorithmes génétiques*. PhD thesis, Université de Rennes 1, 1994.
- [14] Benjamin Morin, Ludovic Mé, Hervé Debar, and Mireille Ducassé. A logic-based model to support alert correlation in intrusion detection. *Information Fusion*, 10(4):285–299, 2009.
- [15] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.
- [16] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artif. Intell.*, 48(1):1–26, 1991.
- [17] Jacob Zimmermann, Ludovic Mé, and Christophe Bidan. An improved reference flow control model for policy-based intrusion detection. In *ESORICS*, pages 291–308, 2003.