N^o d'ordre : D-04-01

Thèse

présentée devant

l'**Institut National des Sciences Appliquées de Rennes**

pour obtenir le grade de

Docteur

spécialité : *Informatique*

par

Benjamin Morin

Corrélation d'alertes issues d'outils de détection d'intrusions avec prise en compte d'informations sur le système surveillé

A soutenir le 5 février 2004 devant la commission d'examen :

Rapporteurs	**S. Benferhat**	Professeur des Universités	Université d'Artois
	B. Le Charlier	Professeur des Universités	Université catholique de Louvain
Examinateurs	**M. Ducassé**	Professeur des Universités	INSA de Rennes
	F. Cuppens	Professeur	ENSTB
	L. Mé	Professeur	Supélec
	H. Debar	Expert	France Télécom R&D

Table des matières

Remerciements

Adolescent peu rebelle, c'est pourtant à cette époque que remonte ma passion pour le piratage informatique. Aussi téméraire que rebelle, la détection d'intrusion m'a permis de pratiquer cet "art" en toute légalité, et c'est à Ludovic Mé que je dois mes premiers pas dans le domaine. Je souhaite le remercier pour son soutien permanent, ses qualités humaines exceptionnelles et pour ses conseils toujours pertinents et constructifs.

Je souhaite aussi exprimer toute ma gratitude envers Hervé Debar, qui m'a fait partager ses connaissances, ainsi que pour la confiance qu'il m'a manifesté tout au long de cette thèse et au delà.

Je veux remercier Mireille Ducassé pour l'ensemble de ses conseils et commentaires, tout particulièrement pour ceux qui m'ont fait râler.

Je tiens à remercier Salem Benferhat et Baudouin Le Charlier d'avoir accepté d'être mes rapporteurs, ainsi que Frédéric Cuppens qui m'a fait l'honneur d'être dans mon jury.

Je souhaite remercier l'ensemble des membres de l'équipe SSR de France Télécom R&D qui, par leur bonne humeur quotidienne, ont contribué à ce que cette thèse se passe dans les meilleures conditions. Je ne saurais tous les nommer de peur d'oublier certains ; pour résumer, je dirais "*Merci, France Télécom...*" [76].

Je remercie de même tous les membres de l'équipe SSIR de Supélec, qui m'ont toujours accueilli avec autant de chaleur, malgré mes absences prolongées.

Merci à mes parents qui m'ont toujours soutenu, dans toutes les occasions, sans qui cette thèse n'aurait pas vu le jour, de toute évidence.

Je souhaite aussi remercier tous mes *potes* de Valognes, de Caen et d'ailleurs, qui à leur manière ont aussi contribué à cette thèse, en me changeant les idées à l'occasion de ces nombreuses soirées passées ensemble. Je profite de cette page pour leur rappeler une fois de plus que je ne suis pas vigile et que je ne travaille ni à La Poste, ni à la SNCF.

Enfin, mes remerciements les plus chaleureux vont vers celle qui m'a toujours encouragé et soutenu depuis le début de cette thèse. Anne-Sophie, je te remercie de tout mon coeur pour m'avoir supporté dans les moments de doute et d'avoir partagé avec moi les moments de joie. Nous allons continuer ensemble à combattre les virus, chacun à notre manière.

à Anne-Sophie,

Chapitre 1

Introduction

Dans un contexte de connectivité croissante des réseaux informatiques, la sécurisation des systèmes d'informations est devenue un enjeu majeur. La sécurité a pour objectif d'assurer la disponibilité des services, la confidentialité et l'intégrité des données et des échanges.

De nombreux mécanismes ont été développés pour assurer la sécurité des systèmes d'informations et tout particulièrement pour *prévenir* les intrusions. Citons par exemple l'authentification, qui consiste pour les utilisateurs à prouver leur identité, le contrôle d'accès, qui consiste à définir les droits accordés aux utilisateurs sur les données, les pare-feux qui filtrent l'accès aux services du système d'informations vis-à-vis de l'extérieur.

Malheureusement, ces mécanismes ont des limitations. En effet, les systèmes informatiques présentent des failles de conception, d'implémentation et de configuration qui permettent à des attaquants de contourner les mécanismes de prévention. De plus, un certain nombre de ces systèmes protègent des attaques extérieures, alors que plusieurs études ont montré que 60 à 70% des attaques proviennent de l'intérieur des systèmes d'informations.

Comme les systèmes de prévention d'intrusions ne suffisent pas, une seconde ligne de défense est nécessaire, la détection d'intrusions. Les outils de détection d'intrusions proposés à l'heure actuelle présentent des défauts et de nombreux travaux tentent à l'heure actuelle d'apporter des solutions pour pallier ces défauts. Parmi ces solutions, la corrélation d'alertes semble être une voie de recherche particulièrement prometteuse. Cette thèse se propose d'explorer cette voie en prenant en compte des informations qui ne sont pas présentes dans les alertes, mais qui décrivent des caractéristiques du système surveillé.

1.1 Détection d'intrusions

La détection d'intrusions est apparue au début des années 80, suite aux travaux de Anderson [6] et à ceux de Denning [24], qui posent les fondations de la détection d'intrusions. Cantonnée jusqu'au début des années 90 au domaine de la recherche, les premiers produits commerciaux sont alors apparus. Dans [13], Cuff comptabilise main-

tenant une centaine d'outils commerciaux, dans le domaine public ou bien prototypes de recherche.

Il n'est pas envisageable de détecter les intrusions manuellement car le volume d'informations à analyser est immense. Les sondes de détection d'intrusions (IDS, *Intrusion Detection System*) sont des outils chargés de détecter automatiquement des actions non autorisées (on parlera aussi d'attaques ou d'actions malicieuses) effectuées par des personnes internes ou externes au système d'informations surveillé.

Une sonde de détection d'intrusions regroupe deux types de composants : un capteur et un analyseur. Fonctionnellement parlant, ces deux composants sont séparés, mais structurellement parlant, ils sont généralement situés dans le même outil. Le capteur est chargé de collecter les données où se manifeste l'activité des utilisateurs. L'analyseur est chargé d'analyser les événements produits par les capteurs pour déterminer si une activité est intrusive.

Les deux critères les plus importants pour classer les sondes sont la nature des données analysées et la méthode d'analyse [21].

1.1.1 Nature des données analysées

Les outils de détection d'intrusions ne détectent pas directement les actions intrusives, mais leurs manifestations lors d'interactions avec les entités du système d'informations. Les capteurs observent ces interactions par le biais des sources de données présentées ci-dessous. La source de données utilisée pour détecter les intrusions est une caractéristique essentielle des IDS. On distingue trois catégories de sources : le trafic réseau, les audits système et les audits applicatifs.

a) Réseau

Les sondes réseau (NIDS, *Network-Based IDS*) sont munies d'un dispositif matériel qui permet de capturer le trafic réseau. Ces sondes permettent de détecter les intrusions qui se manifestent par une interaction réseau entre l'attaquant et la victime.

L'avantage majeur de cette approche réside avant tout dans sa simplicité d'utilisation et son innocuité vis-à-vis de son environnement. A condition d'être positionnée à un endroit stratégique, une seule sonde peut surveiller l'activité des utilisateurs dans l'ensemble d'un réseau. Le travail de ces sondes n'a aucun impact sur les performances des entités surveillées car l'analyse est effectuée sur une machine dédiée. Enfin, une majorité d'attaques (deux vulnérabilités sur trois[1]) impliquent une interaction au niveau du réseau et sont donc détectables à ce niveau. Ces avantages font qu'à l'heure actuelle, la plupart des outils de détection d'intrusions déployés dans des réseaux d'exploitation sont des IDS réseau.

Pour autant, cette source de données présente un certain nombre d'inconvénients. Tout d'abord, bien entendu, seules les activités impliquant des communications réseau peuvent donner lieu à des alertes. Ensuite, l'analyse exhaustive des trames réseau est rendue de plus en plus difficile avec la montée en débit des réseaux. Les sondes actuelles sont capables d'analyser la totalité du trafic sur des liens à 10Mbit/s, mais dès

[1] estimation selon les bases de vulnérabilités CVE

100Mbit/s, les premières limitations se font sentir et l'analyse sur des réseaux 1Gbit/s est très limitée. En outre, le remplacement progressif des concentrateurs par des commutateurs[2] dans les réseaux limite la couverture des sondes. De plus, le chiffrement des données transitant sur les réseaux empêche l'analyse du contenu des paquets par les sondes réseaux. Enfin, plusieurs techniques d'évasion ont été proposées par Ptacek et Newsham [68], ainsi que Handley *et al* [37], qui permettent à un attaquant de contourner des sondes de détection d'intrusions réseau. Ainsi, les premières sondes réseau ne réassemblaient pas les paquets IP, si bien que les attaques n'étaient pas détectées si l'attaquant prenait soin de fragmenter ses paquets.

b) Audit système

Les sondes système (HIDS, *Host-based IDS*) analysent des données d'audit produites par le système d'exploitation des hôtes sur lesquels elles sont installées. Ces données correspondent au niveau de sécurité C2 spécifié dans le livre orange [1] (*Orange Book*), du département de la défense américain. La plupart des systèmes d'exploitation actuels sont équipés de ce type d'audit.

L'avantage des données système réside dans leur fiabilité et leur granularité fine, qui permettent un diagnostic précis des actions effectuées sur un hôte par un attaquant. De plus, la quasi-totalité des attaques provoquent *in fine* une interaction avec le système et sont donc théoriquement détectables à ce niveau.

Malheureusement, le volume d'événements généré par les systèmes d'audits peuvent être considérables, ce qui peut avoir un impact très néfaste sur les performances de la machine surveillée. De plus, la couverture de l'IDS est limitée à un seul hôte, si bien que les administrateurs de sécurité sont amenés à déployer plusieurs sondes, ce qui complexifie la tâche d'administration.

Une catégorie particulière d'IDS systèmes, qualifiés d'hybrides, analysent les données réseau à destination d'un hôte particulier. Ils utilisent à cet effet la pile protocolaire (TCP/IP) du système surveillé. Le problème du chiffrement évoqué dans le cas des sondes réseau se trouve résolu car les données à analyser sont déchiffrées par l'hôte. De même, les problèmes liés à l'utilisation croissante des réseaux commutés et à l'augmentation du débit s'en trouvent aussi résolus *de facto*, puisque la sonde ne surveille qu'un seul hôte.

c) Audit applicatif

La troisième catégorie de source de données est constituée des audits applicatifs. Les données à analyser sont produites directement par une application (*e.g.* un serveur web, ftp, de base de données). L'avantage de cette approche est que les données produites sont très synthétiques ; elles sont sémantiquement riches et leur volume est modéré. L'analyse faite par les sondes peut ainsi être plus complexe.

Par exemple, une requête HTTP à un serveur web et sa réponse donne lieu à une séquence de plusieurs paquets IP au niveau du réseau, alors qu'elle ne donne lieu qu'à

[2] contrairement aux concentrateurs qui diffusent les trames à l'ensemble des hôtes qui y sont connectés, les commutateurs n'envoient aux hôtes que le trafic qui leur est destiné

```
alert tcp $EXTERNAL_NET any -> $INTERNAL_NET 80
  (msg:"WEB-PHP edit_image.php access";
   flow:established,to_server;
   uricontent:"/edit_image.php";
   reference:nessus,11104;
   reference:cve,CVE-2001-1020;
   classtype:web-application-activity;
   sid:1999;
   rev:1;)
```

FIG. 1.1 – Une signature Snort

une seule ligne dans le fichier d'audit de ce même serveur web.

Malheureusement, les applications doivent être instrumentées pour produire des données d'audit, ce qui n'est pas toujours le cas. De plus, la couverture de la sonde en terme de surveillance est encore plus réduite que dans le cas des informations système puisqu'ici l'activité d'une application seulement est surveillée.

1.1.2 Méthodes d'analyse

Il existe deux types d'approches pour détecter les intrusions dans les systèmes informatiques. L'une est basée sur un modèle constitué des actions interdites dans le système d'informations ; nous qualifions l'analyse correspondante de *morphosciente*. L'autre est basé sur un modèle constitué des actions autorisées ; l'analyse correspondante est qualifiée de comportementale.

a) Analyse morphosciente

Cette analyse est généralement désignée par le terme anglais *misuse*, qui signifie littéralement *mauvaise utilisation*. La traduction française généralement adoptée est analyse *par signatures* ou *par scénarios*. Nous trouvons la première trop restrictive ; elle désigne une technique particulière et non l'approche en général. Quant à la seconde, nous préférons conserver la notion de scénario à la corrélation d'alertes. Ne trouvant pas de traduction satisfaisante dans la littérature du domaine, nous proposons le terme *morphoscient*, c'est-à-dire *qui connaît la forme*. En effet les analyseurs morphoscients possèdent une base de connaissances des manifestations des attaques au niveau des sources de données. Ainsi, dans cette approche, tout ce qui n'est pas explicitement interdit est autorisé.

L'approche la plus répandue est basée sur des algorithmes de *pattern matching*. Les motifs recherchés dans les sources de données, censés caractériser des attaques sont appelés des signatures. D'autres approches ont été proposées ; par exemple, Mé [57] propose d'appliquer des algorithmes génétiques pour analyser les audits systèmes. Lunt *et al* [51] proposent une approche basée sur des systèmes experts.

Snort[3] est un analyseur morphoscient réseau. C'est le produit actuellement le plus utilisé en détection d'intrusions. La Figure 1.1 est un exemple de signature extrait de la base de Snort, qui en compte approximativement 2000. Pour que cette signature donne lieu à une alerte pour un paquet IP donné, il faut que le contenu de ce paquet satisfasse l'ensemble des contraintes exprimées dans cette signature, à savoir : l'adresse IP source (resp. destination) doit être externe (resp. interne), le protocole de transport doit être TCP et le port destination doit être 80. Enfin, le paquet doit contenir la chaîne `/edit_image.php`. Les notions d'extérieur et d'intérieur sont désignées par l'administrateur sous la forme d'adresses de réseaux. Nous ne détaillons pas les autres informations qui ne sont pas des contraintes ; elles sont évoquées dans le chapitre 2.

Comme le paradigme de détection morphoscient est avant tout basé sur la reconnaissance des attaques, son avantage majeur réside dans le caractère très détaillé des alertes. L'IDS est capable de désigner quelle vulnérabilité est exploitée, quelle action est entreprise par un attaquant, ce qui permet une contre mesure adaptée et efficace par l'opérateur de sécurité.

L'inconvénient majeur d'une telle approche réside dans son incapacité à détecter des attaques inconnues.

Parmi ces sondes, on peut distinguer celles qui appliquent leur analyse à un seul événement, que nous qualifions de mono-événementielles, de celles qui appliquent leur analyse sur plusieurs événements, dites multi-événementielles. Snort est un exemple de sonde mono-événementielle. Plusieurs travaux, dont ceux de Pouzol et Ducassé proposent une sonde système multi-événementielle [66, 67]. Notons que cette dichotomie est parfois ambiguë car le caractère mono ou multi-événementiel d'une sonde dépend aussi de la source de données utilisée. Par exemple, pour analyser une requête HTTP et la réponse correspondante, une sonde réseau doit être multi-événementielle car cette requête se traduit par une succession de paquets IP ; une analyse mono-événementielle suffit pour une sonde applicative.

La quasi-totalité des sondes déployées en milieu opérationnel sont morphoscientes et mono-événementielles.

b) Analyse comportementale

Dans l'analyse comportementale, un modèle de comportement normal ou autorisé du système surveillé est préalablement construit. Au cours de la surveillance, le comportement courant de l'entité est mesuré ; toute déviation significative du comportement courant par rapport au comportement de référence donne lieu à une alerte. A contrario de l'approche précédente, tout ce qui n'est pas explicitement autorisé est interdit.

La construction du modèle de référence se fait le plus souvent par apprentissage. Debar [20] propose par exemple d'utiliser des réseaux de neurones ; Forrest *et al* [38] proposent pour leur part d'appliquer une approche immunologique. Des méthodes statistiques ont aussi été suggérées [42]. 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* [84].

[3] `http://www.snort.org`

L'avantage de l'analyse comportementale est avant tout lié à sa capacité à détecter des attaques inconnues. Les inconvénients sont que les alertes sont plus difficiles à appréhender qu'avec une analyse morphosciente car l'alerte ne désigne pas une attaque connue. De plus, ces outils ont tendance à générer un grand nombre de faux positifs (c'est-à-dire des alertes en l'absence d'attaque) si le modèle de référence n'est pas exhaustif. Des attaques peuvent aussi ne pas être détectées si le corpus utilisé pour l'apprentissage contient des activités intrusives.

1.2 Corrélation d'alertes

Le déploiement à grande échelle et en milieu opérationnel d'IDS à la fin des années 90 a mis en évidence leurs défauts. Leur problème majeur réside avant tout dans l'excès d'alertes qu'ils produisent. L'opérateur de sécurité chargé d'analyser les alertes se trouve rapidement débordé et, à terme, ne consulte plus qu'épisodiquement les listes d'alertes enregistrées laissant ainsi éventuellement passer des attaques. L'utilisation des IDS est alors similaire à des caméras de surveillance dont on ne visionne les enregistrements que lorsqu'un problème est survenu.

Le deuxième défaut des IDS réside dans la sémantique pauvre des alertes. Même dans le cas des alertes produites par une sonde morphosciente, censées être détaillées, l'opérateur de sécurité ne peut généralement pas déterminer la sévérité d'une alerte sans avoir recours à une analyse manuelle des événements qui ont provoqué l'alerte.

Enfin, les outils de détection d'intrusions sont confrontés au problème des faux négatifs, c'est-à-dire l'absence d'alerte en présence d'attaque. Ces trois problèmes sont discutés dans le chapitre 2 de ce mémoire.

D'une manière générale, Jackobson et Weissman définissent la corrélation d'alertes comme l'*interprétation conceptuelle de plusieurs alertes visant à la leur assigner une meilleure sémantique et à réduire la quantité globale d'alertes* [41]. La corrélation d'alertes est donc envisagée comme l'une des clés de l'évolution des systèmes de détection d'intrusions, au moins pour résoudre le problème d'excès d'alertes et l'amélioration de leur contenu.

Les travaux concernant la corrélation d'alertes dans le domaine de la détection d'intrusions sont relativement récents. Une majorité de ces travaux est issue d'observations et d'expérimentations faites sur le terrain ; la base théorique est encore en construction aujourd'hui. Les produits commerciaux existants ne proposent pas de fonction de corrélation à proprement parler, à l'exception de Risk Manager [23]. Ils proposent essentiellement des systèmes de stockage d'alertes dans une base de données relationnelle, accompagnée d'une console d'affichage des alertes pour faciliter le travail de tri par l'opérateur.

Les approches de corrélation exploitent essentiellement des informations contenues dans les alertes, tout en faisant plus ou moins explicitement référence à des connaissances annexes, nécessaires pour corréler les alertes.

En effet la corrélation d'alertes ne doit pas se limiter aux seules informations contenues dans les alertes fournies par les outils de détection d'intrusions. La corrélation doit aussi exploiter des connaissances sur le système d'informations surveillé, sur les

attaques et sur les outils de sécurité.

Les problèmes des IDS sont notoires, mais mal identifiés. Nous proposons donc dans le chapitre 2 de résumer l'ensemble des problèmes sous la forme d'une taxinomie. Le chapitre suivant est dédié à l'état de l'art des techniques de corrélation d'alertes existantes. Dans le chapitre 4, nous proposons un modèle baptisé $\mathbf{M_2D^2}$, qui fédère les informations qui nous semblent nécessaires à la corrélation d'alertes. Dans les chapitres 5 et 6, nous proposons deux approches de corrélation d'alertes qui exploitent les informations formalisées dans le modèle $\mathbf{M_2D^2}$. Enfin, nous concluons ce mémoire et évoquons les travaux ultérieurs que nous envisageons.

Chapitre 2

Problématique

L'expérience acquise au cours de cette thèse dans l'analyse des alertes montre que le problème majeur des systèmes de détection d'intrusions réside dans l'excès d'alertes. Comme se pose de surcroît le problème des faux négatifs, les administrateurs sont amenés à multiplier les sondes et donc à augmenter encore la quantité d'alertes produites. De plus, la sémantique des alertes, c'est-à-dire l'information qu'elle portent, est pauvre et ne permet pas à l'opérateur de se faire une idée de la gravité des alertes.

Dans cette section, décrivons d'abord la structure des alertes produites par les sondes de détection d'intrusions. Nous détaillons ensuite les trois problèmes identifiés des sondes de détection d'intrusions, sous la forme d'une classification : excès d'alertes, pauvreté de la sémantique et faux négatifs.

2.1 Description des Alertes

IDMEF [17] (*Intrusion Detection Message Exchange Format*), est le format d'alerte proposé par l'IDWG (*Intrusion Detection Working Group*), le groupe de travail sur le thème de la détection d'intrusions à l'IETF (*Internet Engineering Task Force*). IDMEF divise les informations contenues dans les alertes en quatre catégories : l'attaque, l'attaquant, la victime et la date. Ces quatre classes sont communes à la quasi-totalité des alertes ; elles permettent à un opérateur de comprendre ce qui a été fait (l'attaque), par qui (l'attaquant), contre qui (la victime) et quand (l'estampillage temporel de l'alerte).

Nous détaillons d'abord ces informations et leur différentes modalités dans les quatre sections suivantes. Dans la section 2.1.5, nous présentons brièvement d'autres informations disponibles dans les alertes.

2.1.1 Attaques

Dans une alerte, l'attaque désigne une action effectuée par un attaquant dans le système d'information surveillé. Le terme *attaque* dénote une action intrusive. En fait, il s'agit d'un abus de langage car par définition, une intrusion est un enchaînement d'actions qui conduisent à une violation de la politique de sécurité. Les actions qui composent une intrusion peuvent être des attaques ou des actions légitimes. Malgré

cela, dans la suite de ce mémoire, nous utilisons le terme *attaque* pour désigner l'action référencée dans une alerte.

Dans les approches de détection comportementales, les alertes font référence à une déviation par rapport au comportement de référence de l'entité surveillée. Les attaques sont alors qualifiées par une mesure de divergence, ainsi que par l'ensemble des événements qui ont contribué à la divergeance du modèle.

Les approches de détection morphoscientes reconnaissent les attaques grâce aux signatures idoines. Une signature est associée à chaque attaque et chaque signature possède un identifiant unique auquel est associé un message textuel destiné à l'opérateur de sécurité. Dans la suite, nous appelons ce message la documentation de la signature. Les identifiants de signatures constituent ainsi un moyen élémentaire pour désigner les attaques.

Comme dans le cas de l'analyse morphosciente les attaques sont connues, les informations disponibles pour les décrire sont potentiellement plus riches. C'est d'ailleurs un des atouts majeurs de cette approche car le message d'alerte est précis et l'opérateur peut réagir plus rapidement qu'avec une alerte produite par un analyseur comportemental. Les informations disponibles pour qualifier les attaques dans le cas d'une analyse morphosciente sont précieuses pour la corrélation d'alertes car elles permettent de traiter la sémantique des alertes. D'ailleurs, à notre connaissance, les approches de corrélation existantes utilisent exclusivement des alertes produites par des analyseurs morphoscients. Dans la suite de cette section, les identifiants d'attaques qui sont évoquées impliquent généralement que l'attaque soit connue.

Par exemple, dans la signature Snort donnée en Figure 1.1, page 12 l'identifiant de signature est `sid:1999`, et sa documentation est `WEB-PHP edit_image.php access`.

Le format des identifiants de signatures et de leur documentation sont propriétaires. Il dépendent du modèle de l'analyseur qui les produit. Ceci constitue un premier obstacle à la coopération d'IDS hétérogènes car les attaques ne sont pas décrites dans le même langage. Notons que les formats d'alertes définissent une syntaxe commune, mais pas un vocabulaire commun.

Par définition, les attaques exploitent une vulnérabilité. Les attaques peuvent ainsi être désignées par un identifiant de vulnérabilité. Plusieurs organismes référencent les vulnérabilités lorsqu'elles sont découvertes. Le plus connu de ces organismes est le Mitre[1], qui propose une liste de vulnérabilités baptisée CVE (*Common Vulnerabilities and Exposures*). Citons aussi les identifiants BugTraq[2], Nessus[3] et Xforce[4]. La plupart des IDS actuels associent à leurs signatures la ou les références CVE des vulnérabilités correspondantes. La liste CVE constitue donc un référentiel commun de nommage des attaques. Dans la signature donnée en Figure 1.1, deux références de vulnérabilités sont données, une Nessus (`nessus,11104`) et une CVE (`reference:cve,CVE-2001-1020`). Tout comme la documentation des signatures, une description informelle est généralement associée aux identifiants de vulnérabilités.

L'utilisation des vulnérabilités pour désigner les attaques présente toutefois quelques

[1] http ://www.mitre.org/
[2] http ://www.securityfocus.com/bid
[3] http ://www.nessus.org/
[4] http ://xforce.iss.net/

inconvénients. Toutes les signatures ne font pas référence à une vulnérabilité. Certaines signatures sont destinées à détecter des activités qui n'exploitent pas de vulnérabilité au sens propre du terme, comme par exemple des actions potentiellement illicites en regard de la politique de sécurité du système d'information (utilisation de protocoles non sécurisés par exemple). Quand bien même une vulnérabilité est exploitée, elle peut ne pas être connue. A titre indicatif, seules 33% des signatures de Snort font référence à une vulnérabilité CVE[5].

De plus, les informations liées aux vulnérabilités ne reflètent pas les propriétés d'une instance d'attaque. Une vulnérabilité peut être exploitée de différentes manières en fonction de l'objectif de l'attaquant. Par exemple, la signature Snort en Figure 1.1 détecte les accès à un script réputé pour être vulnérable, et non les exploitations effectives de la vulnérabilité.

Enfin, certains IDS associent aux signatures une catégorie d'attaque. Ces catégories peuvent être utilisées par les approches de corrélation d'alertes pour qualifier les attaques. La catégorie d'attaque de la signature Snort de la Figure 1.1 est `web-application-attack`. Elle dénote une attaque contre une application de type web. Comme les identifiants de signatures, les catégories sont propres à un IDS particulier.

En résumé, les description d'attaques utilisées dans les systèmes de détection d'intrusions ne sont pas structurées. Ce sont des messages textuels destinés aux opérateurs de sécurité. L'absence de structure de cet attribut limite le traitement automatisé de la sémantique des alertes car cette sémantique est principalement portée par l'attaque.

2.1.2 Victimes

Les victimes sont les entités du système d'informations contre lesquelles sont dirigées les attaques. Elles ne doivent pas être confondues avec le type de composant vulnérable à l'attaque, qui fait partie des spécifications d'une attaque. Il existe cinq grandes classes d'identifiants de victimes : les nœuds du réseau, les services, les processus, les utilisateurs, les fichiers.

Les nœuds du réseau correspondent à des hôtes et des périphériques d'interconnexion (routeurs, passerelles). Les sondes réseau les identifient par leur(s) adresse(s) ou un nom issu d'un mécanisme de résolution de type DNS (*Domain Name Service*). Etant donnée la prédominance du protocole IP, les adresses des nœuds sont généralement au format IPv4. L'adresse MAC peut aussi être utilisée car elle identifie de manière unique une interface réseau, mais elle n'est accessible qu'aux sondes situées dans le même réseau local que la victime.

Les sondes système ou applicatives identifient les victimes par leur nom système. Un nom système et un nom réseau sont deux notions différentes. Le nom système est propre à l'hôte, alors que le nom réseau est lié à une adresse IP. L'utilisation conjointe de DHCP[6] et DDNS[7] tend à se généraliser dans les réseaux et permet d'assurer une

[5] Ce chiffre prend en compte la totalité des 1964 signatures disponibles en juin 2003

[6] *Dynamic Host Configuration Protocol*, permet d'affecter aux hôtes d'un système une adresse IP à la demande (le nom système peut être joint à la requête

[7] *Dynamic Domain Name Service*, service de résolution de nom capable de prendre en compte les associations entre nom réseau et adresse IP dynamiquement

certaine cohérence entre les noms d'hôtes et les noms réseau mais cette cohérence n'est pas garantie.

Une majorité d'attaques exploitent une faille dans un logiciel. Une manière de désigner la victime d'une attaque est donc d'utiliser un identifiant du logiciel visé. Lorsque le logiciel est un serveur applicatif, les sondes réseau désignent le logiciel par son numéro de port d'écoute[8], voire par le nom du protocole correspondant. Par exemple, un serveur web peut être identifié par le numéro de port TCP 80, ou bien par le nom de service http.

Les sondes système désignent les logiciels par leur identifiant de processus (PID, *Process ID*) et surtout par le nom de la commande correspondante parce que du point de vue d'un opérateur, le PID d'un processus n'a aucune signification. Seules les sondes système ont accès à ce type d'information. Un nom de processus est le pendant système du numéro de service utilisé par les sondes réseau.

Un identifiant d'utilisateur désigne l'identité logique d'un utilisateur dans un système informatique. Un identifiant d'utilisateur peut être utilisé comme identifiant de victime en cas d'attaque visant à usurper une identité, ou a voler des ressources appartenant à l'utilisateur en question.

Les ressources des utilisateurs se présentent en particulier sous la forme de fichiers. Un identifiant de fichier n'est pas une victime à proprement dit mais plutôt une cible. L'objectif d'un grand nombre d'attaques consiste *in fine* à manipuler les fichiers d'un système, c'est-à-dire les lire, les effacer, les modifier ou les exécuter.

Les types de cibles ne sont pas mutuellement exclusives, au contraire. Par exemple, une attaque peut exploiter une vulnérabilité propre à un serveur web présent sur un nœud du réseau et qui permet d'accéder à certains fichiers du système. L'alerte doit alors référencer le service, l'hôte et le fichier.

A condition que l'administrateur de sécurité ne s'intéresse qu'aux attaques dirigées contre le système d'information, les victimes ne désignent que des entités qui appartiennent au système d'information.

Les identifiants de victimes utilisés par les sondes réseau sont peu fiables. Les sondes système ou applicatives sont capables *de facto* de désigner les victimes des attaques sans ambiguïté car elles sont intégrées au système surveillé. En revanche, dans le cas des sondes réseau qui peuvent être éloignées de la victime, certaines architectures utilisées dans les réseaux rendent ambigus ou imprécis les identifiants de victimes. Prenons l'exemple d'un réseau où deux sondes surveillent le trafic, l'une située en amont et l'autre en aval d'un routeur effectuant de la translation d'adresses statique (NAT). Par définition de la translation d'adresse, l'adresse des hôtes n'est pas la même en amont et en aval du routeur. Les adresses des victimes contenues dans les alertes émises par les sondes, relatives aux attaques transitant via le routeur, ne sont donc pas les mêmes. Pourtant, il s'agit bien de la même entité. Cet exemple illustre le besoin de prendre en compte les propriétés du système d'information dans le processus de corrélation.

[8]Le numéro de port est présent dans les entêtes de la couche transport utilisée dans les communications réseau

2.1.3 Attaquant

L'attaquant représente l'origine d'une activité intrusive détectée dans le système d'information. Dans le contexte de la corrélation d'alertes, cet attribut possède *a priori* un pouvoir corrélatif fort. Il paraît en effet légitime de grouper les attaques qui proviennent de la même source car elles peuvent être les étapes d'un scénario d'intrusion composé.

L'attaquant désigne très rarement une personne physique car cette information n'est pas disponible dans les sources de données. En effet, beaucoup de cibles ne requièrent pas d'authentification. Quand bien même une authentification est nécessaire, l'objectif des attaques consiste en particulier à contourner le mécanisme d'authentification. En résumé, si en théorie la source d'une intrusion implique une ou plusieurs personnes physiques, en pratique l'identité de ces personne n'est pas disponible.

Nous avons donc recours à des identifiants logiques pour désigner les attaquants. Ces identifiants sont fortement liés à la nature de la source de donnée utilisée par les capteurs. Il existe quatre catégories d'identifiants [83] : les nœuds réseau, les services réseau, les identifiants d'utilisateurs, et les identifiants de processus.

Nous ne redéfinissons pas ces quatre identifiants d'attaquants qui sont les mêmes que les victimes. Nous commentons simplement leurs différences en tant qu'attaquant.

Comme les capteurs utilisés en milieu opérationnel sont essentiellement réseau, les attaquants sont très souvent désignés par l'adresse d'un hôte.

La fiabilité des adresses IP en tant qu'identifiant d'attaquant n'est pas toujours garantie. Certaines classes d'attaques permettent à l'attaquant de forger l'adresse IP source, lui ôtant toute signification. La validité des adresses IP source en tant qu'identifiant est aussi limitée dans le temps : du fait du dynamisme croissant des réseaux, l'adresse IP d'un attaquant peut subir des variations intentionnelles ou imposées par l'environnement de l'attaquant. Enfin, certains mécanismes réseaux comme la translation d'adresses rendent les attaquants anonymes *de facto*. Par conséquent, le pouvoir corrélatif fort de la source des attaques évoqué plus haut est donc à tempérer.

Lorsqu'une adresse IP est externe au système d'information, les seules informations disponibles au sujet de l'attaquant proviennent des bases de données du *Internet Assigned Numbers Authority* (IANA)[9], organisme en charge de l'affectation des plages d'adresses IP dans le monde.

Lorsqu'une adresse IP est interne, en revanche, les informations disponibles à son sujet sont potentiellement plus riches et fiables, puisque l'affectation des adresses IP aux hôtes est géré par les administrateurs du système d'information.

Il est important de noter que l'adresse IP d'un attaquant n'est pas équivalente à l'adresse IP source d'une communication. En effet, certaines attaques sont détectables par la réponse que la victime à l'attaquant, auquel cas l'adresse source désigne la victime. La correspondance entre les entités impliquées dans une communication réseau et les notions d'attaquant/victime dépend donc de la façon dont est détectée l'attaque.

Notons toutefois pour certaines catégories d'attaques, la distinction entre victime et attaquant n'est pas toujours triviale. Dans les attaques de type ver comme Nimda[10] ou

[9] `http://www.iana.org`

[10] `http://www.cert.org/advisories/CA-2001-26.html`

Blaster[11], par exemple, l'attaquant est avant tout une *victime* d'une attaque préalable. Dans ce type d'attaque, l'hôte interne à l'origine d'une attaque contre un hôte externe est tout de même qualifié de victime, bien que l'intrusion vienne de lui. Les attaques par dénis de service distribué constituent un autre exemple. Ces attaques sont détectées en particulier par des communications entre un maître et un esclave. Dans ce cas, le maître et l'esclave interviennent en qualité d'attaquant dans l'alerte.

Comme les victimes, les attaquants peuvent être identifiés par un numéro de port. Notons qu'en qualité d'attaquant, un numéro de port n'est pas associé à un nom de protocole, puisque le numéro de port affecté à l'initiateur d'une communication réseau est aléatoire. C'est le numéro de port destination qui est associé à un type de protocole.

Un attaquant est très rarement désigné par un identifiant d'utilisateur. Comme la détection des attaques se fait dans l'environnement des victimes, un attaquant ne peut être désigné par un identifiant d'utilisateur que si son attaque est exécutée depuis l'hôte victime et qu'il s'est préalablement authentifié. Même dans ces conditions, rien n'empêche un attaquant d'avoir usurpé l'identité d'un utilisateur pour effectuer son attaque.

Tout comme les identifiants d'utilisateurs, les identifiants de processus désignent plutôt des victimes d'attaques, car cette information n'est disponible que si l'analyseur surveille l'activité de l'hôte à partir duquel l'attaque est effectuée.

2.1.4 Horodatage

Pour le cadre de la coopération de plusieurs analyseurs, les managers peuvent prendre en considération l'estampillage temporel des alertes ou bien l'ordre de réception des alertes. L'estampillage temporel nécessite que les horloges des acteurs impliqués dans la chaîne de détection (capteurs, analyseurs) soient synchronisées, pour que l'écart entre les dates d'occurrence des alertes soit négligeable. Cette contrainte est difficile à satisfaire dans le contexte de la détection d'intrusion où les managers sont confrontés à des rafales d'alertes rapprochées dans le temps. Dans le cas de la date de réception, l'ordre entre les alertes est total, mais le manager est tributaire des délais d'analyse et de transmission des alertes.

Comme défini dans l'IDMEF, on peut distinguer deux types d'informations temporelles dans les alertes :

- la date de réception du premier événement qui compose une attaque par un analyseur (*detect time*),
- la date de création de l'alerte par l'analyseur (*create time*).

Il existe une autre information temporelle définie dans l'IDMEF, l'horloge de l'analyseur (*analyser time*), utilisée pour calculer des dérives d'horloges entre les analyseurs.

L'horodatage des événements est fourni par les capteurs. Lorsqu'une attaque implique plusieurs événements, l'alerte correspondante utilise l'horodatage de l'événement le plus ancien pour indiquer à l'opérateur la date du début de l'attaque.

Le délai qui sépare la date d'occurrence du premier événement de la date de création de l'alerte correspond bien sûr au temps d'analyse, mais aussi au temps nécessaire

[11] `http://www.cert.org/advisories/CA-2003-20.html`

à l'occurrence d'autres événements impliqués dans l'analyse de la sonde qui peuvent infirmer ou confirmer le caractère intrusif d'une activité.

2.1.5 Autres informations

D'autres informations sont généralement disponibles dans les alertes produites par les sondes de détection d'intrusions. Citons :

- L'identité de la sonde à l'origine de l'alerte. Il s'agit en général d'un identifiant affecté par l'opérateur de sécurité qui identifie de manière unique une sonde du système d'information.
- Un numéro de série d'alerte. Couplé à l'identifiant de la sonde, le numéro de série constitue un identifiant unique d'alerte.
- Un niveau de sévérité de l'alerte. Il s'agit en général d'un entier sur une échelle de valeurs indiquant la gravité de l'alerte.
- le succès ou l'échec de l'attaque référencée par l'alerte.

Les deux dernières sont détaillées dans la section 2.3, qui traite de la sémantique des alertes.

2.2 Excès d'alertes

Dans la section précédente, nous avons présenté les problèmes liés au contenu des alertes produites par les sondes de détection d'intrusions. Dans cette section et dans les suivantes, nous présentons les problèmes des alertes dans leur globalité. Cette section traite plus particulièrement du problème de l'excès d'alertes.

La quantité excessive d'alertes générées par les outils de détection d'intrusions est notoire. Nous décrivons ici les causes d'excès d'alertes, nous les illustrons par des observations effectuées dans le réseau de Supélec et nous définissons les fonctions de corrélations qui peuvent solutionner les problèmes. Cependant, il est difficile de mesurer objectivement l'excès d'alertes. En effet, le nombre et la nature des alertes dépend d'un grand nombre de facteurs, éventuellement dictés par des considérations subjectives telles que la politique de sécurité du système d'informations surveillé. Nos mesures sont donc uniquement indicatives.

Parmi les facteurs liés aux sondes qui influencent la quantité d'alertes, citons :

- La configuration des sondes. Elle se traduit en particulier par la liste des signatures appliquées à une sonde morphosciente, qui est dictée par la politique de sécurité propre au site.
- Le nombre de sondes déployées dans le système d'informations, qui dépend de la taille et de la topologie du réseau.
- La quantité d'événements à traiter.

La quantité d'événements à traiter dépend elle même de plusieurs paramètres :

- La nature des événements produits par le capteur : une capteur réseau est à amené à capturer l'activité de plusieurs acteurs, selon plusieurs protocoles, alors qu'une sonde applicative n'a à analyser que l'activité de l' application surveillée.
- L'activité de la source de donnée : elle dépend de la localisation du capteur dans le système d'informations, de l'intérêt que présente l'entité surveillée pour des

		Sonde			Total
		F	D	I	
Nombre d'alertes		37425	14346	235317	287088
Nombre de triplets distincts		11810	3762	145	15707
Faux positifs	Inadéquation				
	Environnement	27%	40%	37.5%	
Multiplication	Redondance	38%/0%	99%/0%	0%/0%	5%
	Récurrence	20%	62%	99%	
	Division	-	-	-	-
Granularité		40%	2.7%	0%	

FIG. 2.1 – Mesures effectuées dans le réseau de Supélec. F = sonde située devant le pare-feu, D = sonde située dans la DMZ, I = sonde située dans le réseau interne

attaquants potentiels et de la présence d'équipements spécifiques dans l'environnement surveillé (e.g. pare-feux)

L'annexe B décrit les paramètres de l'expérimentation menée dans le réseau de Supélec, notamment la topologie du réseau, la localisation des sondes et leur configuration. On peut noter que ces paramètres sont tout à fait classique et en ce sens représentatifs d'un grand nombre de systèmes d'informations.

Le tableau 2.1 résume les observations effectuées pendant 15 jours à l'aide de trois sondes Snort déployées dans le réseau. Une sonde est située à l'entrée du réseau (devant le pare-feu), une autre dans la DMZ (*DeMilitarized Zone*), et la dernière dans le réseau interne. Le nombre d'alertes désigne le nombre d'instances d'alertes produites par chaque sonde. Les 287088 alertes générées par les 3 sondes, représentent une moyenne d'environ 20000 alertes par jour. Ce résultat concorde avec les "milliers d'alertes quotidiens" évoquées par Manganaris [52]. En projetant les alertes selon leur composantes attaque, attaquant, victime et en mesurant le nombre de triplets distincts, nous obtenons une mesure de l'hétérogénéité des alertes. La dernière ligne indique le taux estimé de faux positifs. Nous détaillons chacune des entrées dans la suite de cette section.

L'excès d'alertes résulte d'une combinaison de trois phénomènes. Premièrement, les sondes génèrent une majorité de fausses alertes. Deuxièmement, la quantité globale d'alertes est inutilement multipliée. Enfin, les informations sur les intrusions sont réparties dans des alertes trop granulaires en terme de contenu sémantique.

2.2.1 Faux positifs

Un faux positif résulte du jugement erroné d'un analyseur, qui qualifie une activité normale d'intrusive. Comme les activités normales sont répétitives et qu'en outre elles constituent la majorité de l'activité globale, les faux positifs contribuent en grande partie à l'excès global d'alertes.

Dans le tableau 2.1, les taux de faux positifs sont des estimations. Il n'existe pas de critère commun pour qualifier une alerte de faux positif ; ce choix est laissé

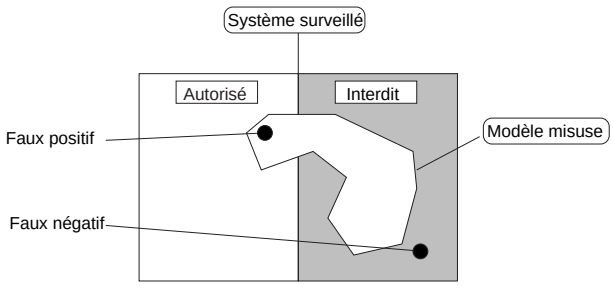

FIG. 2.2 – Problèmes des IDS *misuse*

à l'appréciation de l'opérateur. Dans ces conditions, il ne nous est pas possible d'affirmer avec certitude que, d'une part les alertes que nous estimons être des faux positifs le sont réellement, et d'autre part que les alertes que nous jugeons être des vrais positifs le sont réellement aussi.

Comme on peut le constater, les taux de faux positifs mesurés avoisinent 40%. La proportion de fausses alertes évoquée dans plusieurs articles publiés entre 1999 et 2002 varie entre 90 et 99% [27, 49, 43, 45]. Plusieurs remarques s'imposent donc pour expliquer cette différence.

Tout d'abord, la sonde située en amont du pare-feu est particulièrement exposée aux attaques sauvages issues d'outils d'attaques automatisés. Ces attaques augmentent considérablement le taux de vrais positifs.

Deuxièmement, la proportion de faux positifs observée au niveau de la sonde située dans la DMZ est elle aussi relativement faible ; une proportion importante (41%) d'attaques constatées au niveau du pare-feu sont de type web, or le pare-feu ne bloque pas les communications HTTP, si bien que la sonde située dans la DMZ est elle aussi très exposée aux attaques extérieures.

Troisièmement, on constate que seulement 37.5% des alertes produite par la sonde interne sont des faux positifs. Cependant, deux triplets distincts représentent à eux seuls 45% des instances de vrais positifs. Si nous ne les comptabilisons pas, le taux de faux positifs devient égal à 92.5%. Nous pensons que les taux de faux positifs évoqués dans les publications concernent des sondes situées dans des réseaux internes.

Les faux positifs proviennent soit d'une inadéquation du modèle de référence avec le système réel, soit de l'absence de prise en compte de l'environnement par les sondes.

a) Inadéquation du modèle de référence

La détection des intrusions peut se faire en modélisant soit les activités interdites (analyse morphosciente), soit les activités autorisées (analyse comportementale) d'un système d'informations. Comme l'illustrent les Figures 2.2 et 2.3, les faux positifs proviennent soit de l'incomplétude du modèle dans le cas d'une analyse comportementale, soit de l'empiètement du modèle sur les activités autorisées dans le cas d'une analyse morphosciente.

Par exemple, une des signatures de Snort est basée sur la présence du motif

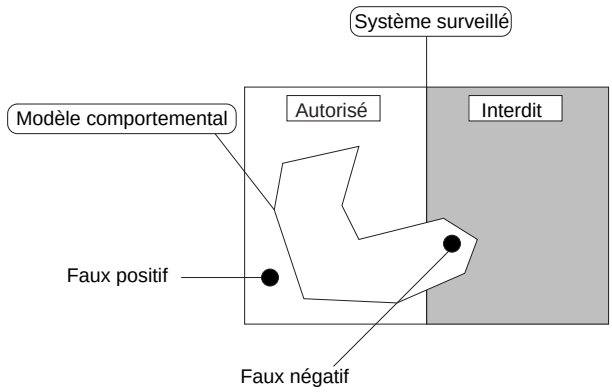

FIG. 2.3 – Problèmes des IDS comportementaux

`/count.cgi` dans les requêtes HTTP car une version du logiciel `count.cgi` présente une faille. Cette signature est trop générique car n'importe quelle requête au script donne lieu à une alerte. Dans le cas des observations menées à Supélec, cette signature engendre 6578 fausses alertes car toutes les requêtes adressées à la page d'accueil du site web font implicitement appel au script `counter.cgi`, responsable de la comptabilisation du nombre d'accès à une page.

L'élimination les faux positifs devrait passer prioritairement par l'amélioration des techniques de détection. Toutefois, les techniques de détection plus élaborées peuvent s'avérer trop coûteuses en temps d'analyse et entraîner un risque accru de faux négatifs, dont les conséquences peuvent être plus graves que les faux positifs. Le déport de l'analyse fine des alertes au niveau du manager se justifie donc pour identifier les fausses alertes.

b) Absence de Prise en compte de l'environnement

Chaque système d'informations présente des caractéristiques propres qui constituent l'environnement d'une sonde de détection d'intrusions. L'activité normale des entités incluses dans l'environnement d'une sonde est parfois assimilable à une activité intrusive, indépendamment de la pertinence des mécanismes de détection ou de la configuration des sondes.

Prenons l'exemple des serveurs mandataires web (ou *proxies*). Le rôle des serveurs mandataires est de relayer vers l'extérieur les requêtes des utilisateurs internes qui leur sont adressées. Vis-à-vis de l'extérieur, les requêtes semblent ne provenir que d'un seul hôte. Ce type de composant peut provoquer deux types de fausses alertes. Tout d'abord, comme plusieurs utilisateurs l'utilisent simultanément, le serveur mandataire effectue un grand nombre de connexions rapprochées dans le temps, vers des serveurs distincts, toujours sur le même port. Le fonctionnement normal d'un serveur mandataire correspond à une attaque de type *balayage IP*, dont l'objectif est de trouver une victime potentielle en interrogeant plusieurs serveurs, sur un même port. L'autre type de fausse alerte liée aux serveurs mandataires provient du fait que, du point de vue d'un IDS qui

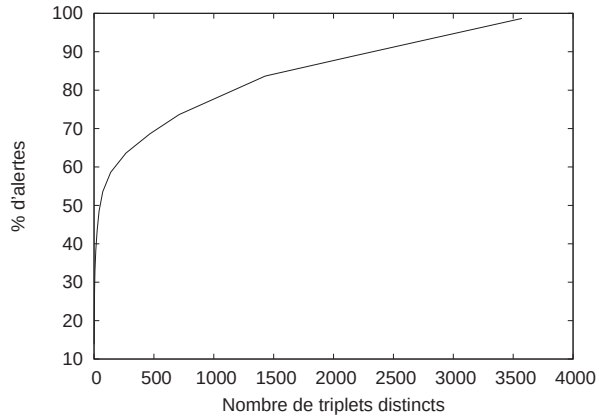

FIG. 2.4 – Proportion d'alertes en fonction du nombre de triplets distincts (sonde DMZ)

surveille les activités internes, les requêtes effectuées par les utilisateurs via le serveur mandataire sont destinées au serveur mandataire. Si les requêtes sont jugées malicieuses par l'IDS, elle sont assimilées à une attaque interne contre l'hôte hébergeant le serveur mandataire, alors que ces requêtes sont destinées à l'extérieur.

La fausse alerte liée aux requêtes `counter.cgi` évoquée précédemment peut aussi être considérée comme un cas de non prise en compte de l'environnement. En effet, ce script fait partie de l'environnement de Supélec, puisqu'il est implicitement appelé par le serveur à chaque requête. Si les systèmes de détection d'intrusions intégraient cette connaissance, les requêtes au script ne seraient pas qualifiées d'attaques.

2.2.2 Multiplication des alertes

Le tableau 2.1 montre que le nombre de triplets distincts est petit devant le nombre d'instances. Cette mesure illustre le caractère répétitif des alertes. Nous la détaillons à l'aide de trois courbes représentant le pourcentage d'alertes en fonction du nombre de triplets distincts, pour chaque sonde (voir courbes 2.4, 2.5, 2.6). On constate tout d'abord que la répétition des alertes varie en fonction des emplacements des sondes. Dans le cas de la sonde interne, deux triplets distincts représentent à eux seuls plus de 50% des alertes. Les alertes des deux autres sondes sont plus hétérogènes, mais on note tout de même que pour la sonde située en amont du pare-feu (resp. dans la DMZ), 8% (resp. 13%) des triplets distincts représentent 65% des alertes.

Nous identifions trois types de phénomènes qui multiplient la quantité globale d'alertes et par conséquent contribuent à l'excès d'alertes : la *récurrence*, la *redondance* et la *divison*. La multiplication des alertes résulte d'une combinaison de ces trois phénomènes.

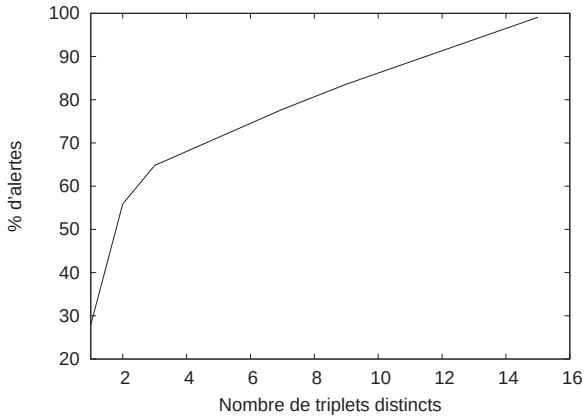

FIG. 2.5 – Proportion d'alertes en fonction du nombre de triplets distincts (sonde interne)

a) Récurrence

Des activités répétitives, d'origine malicieuse ou non, engendrent des alertes récurrentes. Comme les activités normales sont souvent répétitives, les alerte récurrentes s'avèrent être des fausses alertes dans de nombreux cas.

Par exemple, dans le corpus d'alertes de Supélec, on constate que 21016 alertes sont provoquées par un hôte qui effectue des requêtes SNMP jugées litigieuses par Snort à un serveur du réseau. Ces requêtes sont issues d'une machine d'administration et ne sont donc pas des attaques.

Certaines classes d'attaques engendrent aussi des activités répétitives. Dans le même corpus, on peut constater qu'un hôte précédemment infecté par un cheval de Troie et esclave d'un dénis de service distribué provoque 131474 alertes en tentant de contacter son maître, soit plus d'un tiers du nombre total d'alertes.

Nous appelons *fusion* la fonction de corrélation qui traite les alertes récurrentes. La fusion pourrait être effectuée directement au niveau des sondes, par l'analyseur. Nous concevons plutôt les analyseurs comme des processus chargés de filtrer et qualifier les événements dignes d'intérêt vis-à-vis de la sécurité du système d'information. En ce sens, un analyseur doit rapporter au manager chaque événement d'un phénomène récurrent. La fonction de fusion appartient alors au manager.

b) Redondance

Le phénomène de redondance des alertes vient de la multiplication des sondes de détection d'intrusions. Si une activité intrusive est détectée par plusieurs analyseurs, elle donne lieu à plusieurs alertes indépendantes.

La multiplication des sondes de détection d'intrusions dans le système d'information augmente la couverture en terme de surveillance et limite ainsi le risque de faux négatifs. Elle permet aussi d'obtenir des points de vue complémentaires sur les attaques dans le cas où les analyseurs diffèrent dans leur méthode d'analyse ou leur source de donnée.

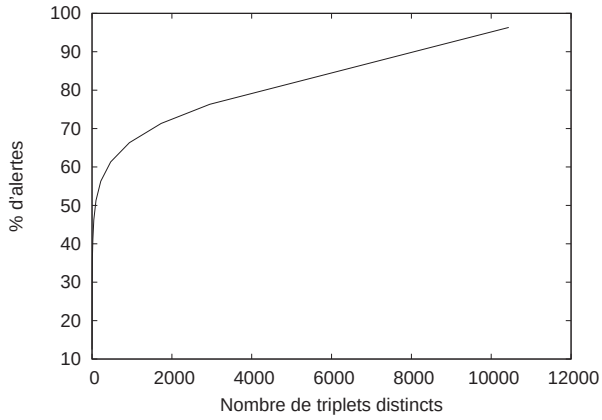

FIG. 2.6 – Proportion d'alertes en fonction du nombre de triplets distincts (sonde en amont pare-feu)

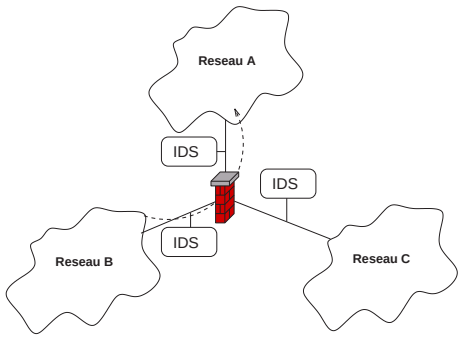

FIG. 2.7 – Multiplication des sondes dans un réseau

La figure 2.7 montre une topologie typique de réseau dans laquelle des sondes réseau sont déployées. Le réseau de Supélec correspond à ce type de topologie. On voit qu'en fonction de son trajet, une attaque (symbolisée par une flèche pointillée) n'est pas détectée par les mêmes sondes. Dans le corpus, la quasi-totalité des 14346 alertes produites par la sonde située dans la DMZ sont redondantes avec des alertes produites par la sonde située en amont du pare-feu.

Nous appelons *synthèse* la fonction de corrélation qui traite les alertes redondantes. Elle consiste à grouper les alertes qui font référence à la même activité intrusive et à générer une alerte synthétique, incluant toutes les informations contenues dans les alertes.

L'activité malicieuse est le dénominateur commun d'alertes redondantes. Il est malheureusement difficile d'utiliser ce dénominateur commun pour synthétiser les alertes car une même activité se traduit par des événements très hétérogènes en fonction des sources de données, et même au niveau de deux sources de données similaires. Par exemple, une requête malicieuse adressée à un serveur web peut être détectée sous

forme de paquets au niveau du réseau, sous forme d'une séquence d'appels système sur le système, ou bien sous forme d'une entrée dans un fichier d'audit applicatif. Ces formes sont difficilement comparables entre elles et rend la synthèse d'alerte à partir des événements très difficile. On est donc amené à synthétiser les alertes redondantes non pas en fonction des événements qui les ont provoqué, mais en fonction de la description des alertes.

Comme la synthèse implique plusieurs analyseurs, elle doit être implantée au niveau du manager.

c) Division

Chaque signature d'un analyseur morphoscient concerne une propriété particulière des événements fournis par les capteurs. L'activation d'une signature engendre l'émission d'une alerte. Lorsqu'un événement présente plusieurs propriétés suspecte, une alerte synthétique contenant l'ensemble des propriétés suspectes devrait être composée, de sorte qu'*au plus* une alerte soit générée par événement. Malheureusement, la plupart des analyseurs dissocient les alertes liées à un même événement. Nous appelons ce phénomène la division.

Prenons l'exemple de la requête http suivante :

```
http://webserveur/cgi-bin/phf?Qalias=x%0a/bin/cat%20/etc/passwd
```

Une version du script phf présente une vulnérabilité permettant à un attaquant de faire exécuter des commandes illicites à la victime [12]. A condition que le script utilisé sur le serveur `webserver` soit vulnérable, la requête précédente permet d'accéder au fichier `/etc/passwd`, qui contient des informations sur les utilisateurs du système. Certains IDS génèrent deux alertes pour cette requête : l'une concernant la présence `phf?` dans la requête et l'autre concernant `/etc/passwd`. Du point de vue du manager, les deux alertes sont dissociées, l'opérateur de sécurité est chargé de d'analyser les événements pour comprendre que les deux alertes sont liées.

La division concerne aussi les analyseurs comportementaux. En effet, si une déviation affecte plusieurs variables modélisant un système, alors un IDS dont l'analyse est statistique, multivariables peut générer plusieurs alertes.

Le tableau 2.1 ne contient pas de mesures de division car seules de sondes Snort sont utilisées dans cette expérimentation. Or, les sondes Snort émettent au plus une alerte par événement car l'analyse s'arrête dès qu'une signature correspond à l'événement analysé.

La *composition* est la fonction de corrélation chargée de réunir les alertes concernant le même événement et émises par une même sonde. Cette fonction doit être implantée au sein des analyseurs car on dispose de toutes les informations nécessaires à leur niveau.

2.2.3 Granularité des alertes

D'une manière très générale, une intrusion effectuée par un attaquant est composée de plusieurs étapes. Ces étapes permettent par exemple à l'attaquant d'acquérir dans un

[12] `http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0067`

premier temps des informations sur le système d'information, pour adapter les attaques ultérieures et atteindre son objectif (acquisition de privilèges, vol de données, dénis de service).

Les outils actuels dissocient les alertes afférentes aux différentes étapes d'une intrusion. C'est à l'opérateur qu'incombe l'identification d'une stratégie d'attaque à partir de l'ensemble des alertes. Cette tâche est rendue d'autant plus délicate que les alertes sont noyées au sein du flot d'alertes global. L'un des objectifs de la corrélation est donc de regrouper les alertes relatives aux étapes d'une intrusion.

Notons que dans certains cas, la frontière entre la division et la granularité est ténue. Elle réside essentiellement dans la nature du capteur utilisé. Prenons l'exemple d'un capteur A agissant au niveau d'envois TCP et d'un capteur B agissant au niveau d'une ligne d'audit applicatif d'un serveur web. A et B produisent chacun deux alertes, l'une afférente à l'attaque, l'autre afférente à son succès ou à son échec. Le succès ou l'échec d'une attaque web peut être déterminé par la réponse fournie par le serveur web. La réponse du serveur est contenue dans la même ligne d'audit que l'attaque, alors qu'elle fait l'objet d'un envoi séparé dans le cas de TCP. Par conséquent, dans le cas du capteur A, le problème est attribué à la granularité trop fine des alertes ; dans le cas du capteur B elle est attribuée à la division des alertes.

Dans un premier temps, l'un des rôles de la corrélation est de regrouper les alertes correspondant à une même étape. Dans un second temps, la corrélation doit permettre de regrouper les alertes correspondant à une même intrusion.

Le tableau 2.1 montre que l'excès d'alertes lié à la granularité varie en fonction de la position de la sonde dans le réseau. La sonde située en amont du pare feu présente un taux élevé car elle est confrontée à un grand nombre d'attaques par balayage, qui sont bloquées par le pare-feu. Les autres sondes ne voient donc pas ce trafic. Une attaque par balayage consiste lancer une attaque donnée sur chacune des adresses IP d'une plage. Une alerte est alors générée pour chaque adresse IP.

2.3 Sémantique des alertes

Nous avons vu dans les paragraphes précédents que la corrélation d'alertes doit réduire le flot d'alertes soumis à l'opérateur. Suivant la définition de Jackobson et Weissman [41] (page 14), la corrélation doit aussi améliorer la sémantique des alertes. De fait, la pauvreté sémantique des alertes constitue le deuxième problème majeur des systèmes de détection d'intrusions. Les informations contenues dans les alertes sont le plus souvent obscures et ne permettent pas à un opérateur d'appréhender les alertes sans procéder à une analyse manuelle du contexte de l'alerte. Ce contexte comprend les événements ayant provoqué l'alerte, les alertes périphériques, la documentation des signatures et des vulnérabilités, les caractéristiques des entités du système d'informations. Cette analyse est longue et fastidieuse, par conséquent l'opérateur de sécurité est rapidement submergé par les alertes car sa capacité d'analyse est limitée.

L'amélioration sémantique des alertes se décline suivant trois axes : l'enrichissement du contenu des alertes, la mesure de la sévérité et la qualification des groupes d'alertes.

2.3.1 Enrichissement des alertes

Dans la section 2.1, nous avons vu qu'en fonction de type d'analyse et du modèle de sonde, de la source de données et de la localisation du capteur, les attributs des alertes ne sont désignés de la même manière.

La première fonction de la corrélation d'alerte est donc de normaliser les alertes, c'est-à-dire enrichir leur contenu pour permettre leur manipulation de manière homogène, indépendamment du type d'analyse, du modèle de sonde, de la source de données et de la localisation du capteur.

Cette normalisation nécessite de prendre en compte des informations périphériques liées aux attributs des alertes (attaques, victimes et attaquants).

2.3.2 Evaluation de la sévérité

L'amélioration sémantique des alertes doit permettre d'évaluer la sévérité des alertes, c'est-à-dire le degré de gravité de l'attaque correspondante. Actuellement, les informations contenues dans les alertes ne permettent pas à un opérateur d'évaluer la sévérité des attaques sans recourir à une analyse manuelle des événements à l'origine des alertes.

Dans les IDS, comme dans le format IDMEF, la sévérité des alertes est donnée sous la forme d'un indice appartenant à une échelle de gravité. Pour les analyseurs morphoscients, les indice sont liés aux signatures. Cependant, la sévérité d'une alerte dépend des modalités d'attaques et non de la signature chargée de détecter ces attaques, qui concerne les propriétés statiques des attaques.

Reprenons l'exemple de l'attaque sur le script `phf` évoquée précédemment. Nous considérons les quatre requêtes suivantes :

```
http://webserveur/cgi-bin/phf?Qalias=value
http://webserveur/cgi-bin/phf?
http://webserveur/cgi-bin/phf?Qalias=x%0a/bin/cat%20/etc/passwd
http://webserveur/cgi-bin/phf?Qalias=x
        %0a/usr/bin/X11R6/bin/xterm%20-ut%20-display%20pirate:0.0
```

La première requête est légitime ; la seconde est litigieuse car sa syntaxe n'est pas conforme aux spécifications du script (un argument devrait être fourni) ; la troisième et la quatrième sont des attaques avérées qui consistent à faire exécuter du code illégitime à la victime. En cas de succès, dans le troisième cas, le fichier `/etc/passwd` est affiché, dans le quatrième cas une ligne de commande interactive est fournie à l'attaquant. L'exécution de code illégitime sur une victime constitue la conséquence la plus grave pour une attaque.

Les effets de ces requêtes ne sont pas les mêmes et pourtant la plupart des IDS génèrent la même alerte dans les quatre cas. En effet, les signatures sont généralement basées sur la présence de la chaîne `cgi-bin/phf?` dans les requêtes. Deux politiques d'affectation des indices de gravité aux signatures peuvent être adoptées :

- soit l'indice est faible car la signature ne détecte finalement qu'un accès au script, auquel cas les attaques avérées ne donnent pas lieu à des alertes sévères ;

– soit l'indice de gravité est maximal car le gain potentiel d'une attaque est grave, auquel cas toutes les requêtes légitimes donnent lieu à des alertes sévères.

On voit qu'aucune des solutions n'est satisfaisante. Nous pensons que la sévérité des alertes est fonction de trois paramètres :

1. Le gain maximal de l'attaque. Nous proposons trois catégories de gain, de gravité croissante : dénis de service, obtention d'information et exécution de code. Il désigne le gain le plus grave en cas de succès d'une attaque. Ce paramètre peut être associé directement aux signatures.
2. La modalité de l'attaque. Nous proposons deux grandes catégories : le test de vulnérabilité ou l'attaque avérée.
3. Le succès ou l'échec de l'attaque.

Le gain maximal des quatre attaques énumérées ci-dessus est le même, l'exécution de code. La modalité est le test de vulnérabilité pour les deux premières, et l'attaque avérée pour les deux dernières. Le succès ou l'échec des attaques dépend de la vulnérabilité de la victime.

La vulnérabilité d'une victime peut se faire en croisant les alertes avec des rapports issus de scanners de vulnérabilité, soit en ayant une connaissance précise des logiciels hébergés par une victime. Nous identifions trois types de statuts de vulnérabilité :

– effective : l'attaque exploite une vulnérabilité présente sur la cible,
– passée : l'attaque exploite une vulnérabilité anciennement présente sur la cible,
– impossible : l'attaque exploite une vulnérabilité qui ne peut pas exister sur la cible. Par exemple, une tentative d'exploitation d'une vulnérabilité liée à un logiciel Microsoft sur une machine Unix est impossible.

En résumé, un niveau de sévérité sous la forme d'une mesure quantitative figée à la conception des signatures ne suffit pas. Le niveau de sévérité d'une alerte dépend des modalités d'attaques et devrait être mesuré qualitativement.

2.3.3 Qualification de groupes d'alertes

L'une des fonctions de la corrélation est de regrouper les alertes, mais le regroupement en tant que tel ne suffit pas, il doit s'accompagner d'une qualification. La qualification permet de décrire de manière synthétique les groupes d'alertes, justifier la raison du regroupement.

La qualification peut se faire en reconnaissant un phénomène (malicieux ou non) à l'origine d'un groupe d'alerte. Par exemple, on peut voir dans le corpus que Supélec plusieurs manifestations du ver Nimda. Une seule tentative d'attaque par le ver se traduit par plusieurs dizaines de requêtes litigieuses à un serveur web, donnant chacune lieu à une alerte. L'amélioration sémantique réside ici dans l'étiquetage du groupe par le qualificatif "Nimda", qui est bien le phénomène reconnu à l'origine de la séquence d'alertes.

La qualification peut aussi se faire en utilisant des termes abstraits, pour résumer le contenu des groupes. Par exemple, dans le cas de l'alerte relative au accès répétés au script `counter.cgi`, il serait souhaitable que la liste des attaquants présumés soit résumée pour que l'opérateur comprenne d'emblée que le phénomène est global (i.e. les requêtes proviennent de n'importe quelle source).

2.4 Faux Négatifs

Le problème des faux négatifs n'est pas abordé dans ce document. Nous l'évoquons malgré tout au titre de l'énumération des problèmes des sondes de détection d'intrusions.

En dépit de l'excès d'alertes générées par les sondes de détection d'intrusions, un grand nombre d'attaques ne sont pas détectées. Les faux négatifs sont bien entendu difficiles à quantifier en milieu opérationnel, puisque ces attaques ne donnent lieu à aucune alerte.

Le taux de faux négatifs varie en fonction des sondes, du taux de faux positifs tolérés, de l'âge et du type des attaques.

La capacité d'un IDS à détecter une attaque dépend de la présence au niveau de la source de donnée d'un capteur et de la capacité de l'analyseur à identifier l'activité comme étant intrusive.

Dans le premier cas, les opérateurs sont amenés à multiplier les sondes de détection d'intrusion afin d'obtenir une couverture aussi complète que possible du système d'information. La multiplication des sondes multiplie aussi la quantité globale d'alertes et renforce le besoin de réduction d'alertes évoqué précédemment. En outre, avec le débit croissant des réseaux les capteurs sont amenés à abandonner des paquets, et manquer ainsi des attaques.

Dans le second cas, l'assouplissement des règles de détection des analyseurs permet de réduire le taux de faux négatifs, au prix d'une quantité de faux positif accrue. Ceci est illustré en figure 2.2 (page 25), où l'on voit l'assouplissement d'une règle, qui se traduit par une signature trop générique par exemple, provoque un empiètement du modèle sur les activités autorisées et engendre des fausses alertes.

Les tests d'IDS, en particulier ceux effectués par la DARPA [49] révèlent que pour une tolérance de 10 fausses alertes par jour, en moyenne 40% des attaques ne sont pas détectées par les IDS. Malheureusement, ces tests ne sont pas effectués en environnement opérationnel. Si l'on considère que plusieurs milliers d'alertes sont générées en milieu opérationnel [53] et que 90 à 99% des alertes sont des fausses alertes [43], alors on voit qu'une tolérance de 10 fausses alertes par jour est peu réaliste.

En ce qui concerne l'âge des attaques, dans le cas des analyseurs morphoscients, une attaque très récente n'est pas détectée parce qu'elle n'est pas encore référencée dans les bases de signatures. A l'inverse, des attaques trop anciennes ne sont plus détectées non plus parce que la probabilité qu'une cible soit vulnérable est jugée faible. Par exemple, Snort ne détecte pas les attaques contre les serveurs Apache de version inférieure à 1.3.26.

Nos travaux de corrélation d'alertes s'appuient sur les observations faites par les analyseurs. Nous ne nous préoccupons pas par la suite du problème des faux négatifs qui, selon nous, doivent prioritairement faire l'objet d'améliorations au niveau des analyseurs. Notons toutefois que dans l'approche de corrélation d'alertes proposée par Cuppens et Miège, des faits non-observés sont inférés à partir de scénarios d'attaques préétablis [15]. Dans leur approche, la corrélation est donc impliquée dans la résolution des faux négatifs. Nous évoquons cette approche plus en détail en section section 3.2.1.

Chapitre 3

Etat de l'art

Dans ce chapitre, nous présentons d'abord les approches qui ont été proposées pour améliorer le contenu des alertes, puis nous décrivons les approches de corrélation d'alertes existantes.

3.1 Contenu des alertes

Nous avons vu que les informations contenues dans les alertes doivent être agrémentées par des informations extérieures, notamment environnementales. Plusieurs auteurs proposent d'intégrer l'environnement dans le processus de corrélation d'alertes.

Nous nous focalisons d'abord sur les travaux proposés pour améliorer la description des attaques, puis nous évoquons les travaux visant à prendre en compte les informations environnementales dans les alertes.

3.1.1 Description des attaques

Dans la section 2.1.1, nous avons vu d'une part que l'hétérogénéité des identifiants d'attaques constitue un obstacle à la coopération d'IDS hétérogènes. Nous avons vu d'autre part que l'absence de structure des descriptions informelles associées aux identifiants va à l'encontre d'un traitement automatisé de la sémantique des alertes.

Une solution triviale pour permettre aux analyseurs morphoscients de coopérer consiste à construire un référentiel commun d'identifiants de signatures. C'est l'objectif de la liste CVE, mais comme nous l'avons dit les entrées de la liste CVE sont des vulnérabilités et non des signatures, or toutes les signatures ne font pas référence à une vulnérabilité. Cuppens propose de constituer une telle table d'associations entre signatures [14]. Au cours d'une phase préliminaire de tests d'IDS, la table d'association entre les identifiants de signatures de plusieurs IDS et un identifiant fédérateur est établie. La phase de test consiste à confronter les IDS évalués à un ensemble d'attaques et à observer les signatures activées par chaque IDS, pour chaque attaque.

Malheureusement, il est très difficile de constituer et de maintenir une base de données d'attaques couvrant l'ensemble des signatures activables de plusieurs IDS : plusieurs nouvelles attaques sont découvertes chaque jour, la manière d'exploiter l'at-

taque n'est pas toujours disponible, les procédures de test ne sont pas toutes automatisables et le nombre d'IDS disponibles est grand.

La structuration des descriptions d'attaques a fait l'objet de nombreuses publications. Dans la littérature consacrée au sujet, on peut distinguer trois approches : les taxinomies, les ontologies (qui sont une généralisation des précédentes) et les langages. L'adoption d'une structure de description d'attaques par l'ensemble des acteurs intervenant dans le processus de détection des intrusions permettrait d'une part de traiter le contenu sémantique des alertes mais aussi de faire coopérer les sondes.

a) Taxinomies d'attaques

De nombreux travaux proposent une taxinomie des attaques, c'est-à-dire un ensemble de critères qui permettent de conceptualiser et catégoriser les attaques. On pourra se réferer à la taxinomie proposée par Howard et Longstaff [39] pour une synthèse de ces travaux. Nous évoquons simplement les propriétés générales de ces taxinomies.

Les classifications proposées par les auteurs tentent d'avoir les caractéristiques suivantes :

- mutuellement exclusives : une attaque n'appartient qu'à une seule catégorie
- exhaustives : toutes les attaques peuvent être décrites,
- non ambigues : la classification d'une attaque est toujours la même, indépendemment de la personne qui la classifie,
- concensuelles : les catégories sont acceptées par une majorité des membres de la communauté,
- utiles : les catégories doivent être utilisables dans le domaine.

Les taxinomies les plus simples n'utilisent aucun critère de classification particulier et proposent simplement un ensemble de catégories qui couvrent des aspects variés des attaques. Des catégories relatives au mode opératoire (par exemple "cheval de Troie") ou aux conséquences des attaques (par exemple "Dénis de service") sont typiquement rencontrées dans ces approches. Elles sont établies de manière empirique. Le nombre de catégories proposées varie selon les auteurs : Cohen propose 97 catégories [12], alors que Cheswick et Bellovin n'en proposent que 7 [11].

D'autres auteurs proposent un ou plusieurs critères de classification :

- Effets : impact d'une attaque réussie sur le système. Les valeurs peuvent être relatives au type de privilèges acquis par l'attaquant, comme proposé par Kendall [46] : `remote-to-local` (accès utilisateur à partir d'un accès réseau), `user to root` (accès administrateur à partir d'un accès utilisateur), `remote-to-root` (accès administrateur à partir d'un accès réseau). Les catégories `dénis de service` et `révélation d'informations` font aussi partie des effets.
- Technique d'attaque : nature de la vulnérabilité exploitée. Elle peut être de configuration, d'implémentation ou de spécification [39, 46]. L'ingénierie sociale fait aussi partie des techniques d'attaques.
- Action : nature de l'action correspondant à l'attaque, comme par exemple `balayage`, `lecture`, `suppression`.

Les classifications d'attaques peuvent être élaborées du point de vue du de l'attaquant ou bien du point de vue du système attaqué, comme c'est le cas dans la taxi-

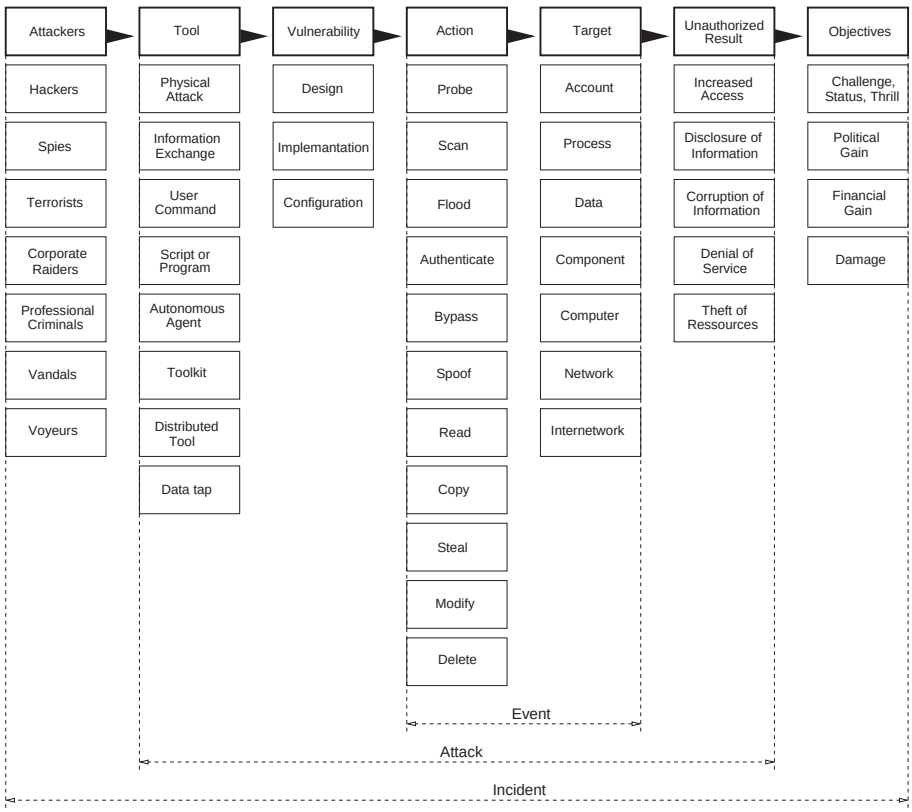

FIG. 3.1 – Taxinomie de Howard

nomie de Lindqvist et Jonsson [48]. Comme le suggèrent Allen *et al* [4], les taxinomies centrées autour de la cible des attaques sont adaptées à la corrélation d'alerte et la détection d'attaque car les observations proviennent de capteurs situés dans le système surveillé ou à proximité et non au niveau de l'attaquant.

La taxinomie d'incidents proposée par Howard et Longstaff reprend la majorité des critères de classification identifiés par les auteurs [39]. A titre illustratif, nous présentons cette taxinomie en Figure 3.1.

Selon les définitions de Howard et Longstaff, un incident est un ensemble d'attaques menées par un attaquant et dirigée par un objectif. Une attaque est effectuée à l'aide d'un outil (*tool*) qui exploite une vulnérabilité et a un effet (*unauthorized result*) sur le système attaqué. Une attaque se manifeste par des événements. Un événement est une action dirigée contre une cible (*target*).

Le champ d'application de cette taxinomie est volontairement vaste car elle n'est pas destinée qu'à la corrélation d'alertes. Certains des critères de classification ne sont donc pas pertinents pour la corrélation, en particulier ceux dédiés aux incidents. Cette remarque se généralise d'ailleurs à la plupart des taxinomies proposées, qui sont trop générales, si bien que la description des attaques perd en précision.

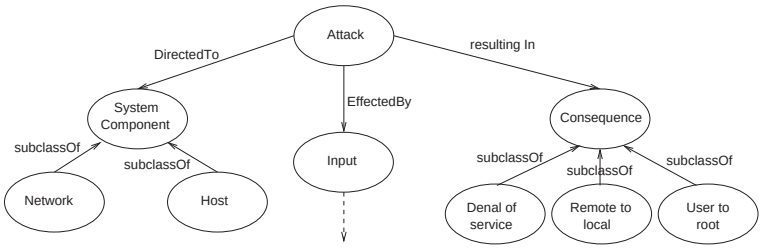

FIG. 3.2 – Une ontologie des attaques

b) Ontologies d'attaques

Une ontologie est *une spécification concensuelle et formelle de conceptualisations* [35]. Une ontologie procure une connaissance commune et partagée sur un domaine et peut être communiquée à des personnes et des logiciels applicatifs. En l'occurrence, les attaques constituent le domaine d'étude et composants de détection d'intrusions (analyseurs et managers) sont les logiciels. Les travaux décrits dans le paragraphe précédent proposent des taxinomies d'attaques, c'est-à-dire des critères permettant de classifier les attaques. Une taxinomie est une forme particulière d'ontologie, simple pour l'homme qui fonctionne souvent par associations et abstractions. Les ontologies permettent à une machine de raisonner sur les termes qui composent une ontologie.

D'une manière générale, la description ontologique d'un objet est censée se substituer à l'objet lui même dans sa manipulation. Par exemple, dans le contexte de la corrélation d'alertes, les termes de l'ontologie qui qualifient une attaque se substituent à l'identifiant de l'attaque dans le processus de corrélation. L'utilisation d'une ontologie en détection d'intrusion permet non seulement de décrire les attaques, mais surtout de détecter des attaques et corréler des alertes à partir des description des données analysées.

Dans [40], Joshi *et al* proposent une ontologie des attaques, du point de vue du système attaqué. La Figure 3.2 illustre un sous ensemble de l'ontologie proposée par Joshi *et al.* Avec cette ontologie, une attaque est caractérisée par le type de système visé (une réseau, un hôte), ses effets (dénis de service, accroissement de privilèges) et le mode opératoire de l'attaque (*input*). Ces termes constituent les nœuds du graphes, qui sont reliés entre eux par des relations (par exemple *effectedBy*). On peut remarquer par exemple que les effets d'une attaque reprennent la taxinomie proposée par Kendall [46].

c) Langages de description d'attaques

Plusieurs auteurs proposent des langages de description d'attaques. Toutefois, ces langages ne qualifient pas les propriétés des attaques. Leur objectif est plutôt de décrire une attaque par l'enchaînement des étapes qui la compose. Des taxinomies d'attaques sont implicitement utilisées dans ces langages mais leur but premier n'est pas de proposer une ontologie des attaques. Comme ils s'inscrivent dans un processus de corrélation d'alertes, nous les décrivons dans la Section 3.2.1, dédiée aux approches de corrélation d'alertes explicites.

d) Synthèse sur les identifiants d'attaque

En dépit du nombre de travaux portant sur la description des attaques, aucune proposition n'a fait l'objet d'un consensus dans la communauté de détection d'intrusions. Ceci tient peut être au fait que, à notre connaissance, ces structures n'ont pas été appliquées à bases documentaires d'attaques réelles, comme on peut en trouver dans les descriptions informelles disponibles dans les signatures d'analyseurs morphoscients.

Les techniques de corrélation d'alertes doivent donc utiliser des informations extérieures, telles que les vulnérabilités, ou bien définir leurs propres qualificatifs d'attaques, mais les descriptions ne couvrent pas l'ensemble des attaques existantes. Par exemple, dans l'approche de Dain et Cunnigham [18, 19], ainsi que celle de Valdes et Skinner [78], les catégories sont définies par rapport aux effets des attaques, comme par exemple `dénis de service`, `sondage`, `violation de privilège`, etc. Nous verrons en section 3.2 l'utilisation qui est faite de ce type d'information.

Le NIST (*National Institute of Standards and Technology*) propose une base de données descriptive des vulnérabilités référencées dans la liste CVE, baptisée ICAT[1]. Le schéma de base de données reprend certains des critères de classification proposés par les auteurs de taxinomies évoqués ci-dessus. Les vulnérabilités sont classées dans la base à partir de la description fournies par le Mitre.

3.1.2 Informations environnementales

La description des alertes fait apparaître la nécessité d'impliquer des données qui ne sont pas présentes dans les alertes pour corréler les alertes. Nous appelons ces données les informations environnementales.

Comme nous le verrons dans la section 3.2, dédiée aux approches de corrélation, les auteurs font implicitement appel à des connaissances expertes et environnementales pour corréler les alertes. Dans cette section, nous les présentons les quelques travaux qui on été menés, visant à structurer ces informations.

a) *Netstat Network Fact Base*

Dans [80, 81], Vigna et Kemmerer proposent de combiner les alerte et les propriétés des entités du système d'information surveillé. Les propriétés sont stockées dans une base de faits du réseau (NFB, *Network Fact Base*) qui inclue la topologie du réseau et les services hébergés par les hôtes.

La partie topologique est inspirée des travaux antérieurs de Vigna [79] sur un modèle formel de topologie d'un réseau TCP/IP. Les autres informations sur la modélisation de l'environnement dans NFB sont simplement données par les auteurs de manière informelle, à l'aide d'exemples. Par conséquent, à l'exception de la topologie, le modèle NNFB n'est ni formel, ni complet.

[1] `http://icat.nist.gov/`

b) *Intrusion Reference Model*

Goldman *et al* [34] proposent eux aussi de prendre en compte les propriétés du système d'information, en particulier la politique de sécurité, dans le processus de corrélation des alertes. Leur modèle est baptisé NERD (*Network Entity Relationship Database*).

Comme NFB, la partie topologie réseau de NERD est aussi inspirée des travaux de Vigna [79]. La description des entités et des relations entre les entités de NERD est plus complète que dans NFB. Cependant, la structure formelle de NERD donnée dans [34] est sommaire. De plus, pour autant que sachions, NERD n'a pas évolué depuis sa conception en 2001.

c) *M-Correlator*

Dans des travaux parallèles aux nôtres, Porras *et al* [65] proposent d'impliquer les connaissances expertes sur les attaques et sur le système d'information surveillé dans le calcul de sévérité des alertes. Les auteurs proposent aussi d'associer aux alertes des événements provenant d'autres outils que les IDS, baptisés de manière générique périphériques INFOSEC. Ces connaissances sont contenues dans une base de données (IHFB, *Incident Handling Fact Base*), qui fait partie d'un système de corrélation baptisé *M-Correlator*.

Les auteurs donnent essentiellement des indications sur la manière de collecter les informations contenues dans la IHFB, mais ils ne précisent pas sa structure générale.

3.1.3 Contribution

Les informations contenues dans les alertes ne sont pas suffisantes. Pour cette raison, nous proposons un modèle qui fédère les informations que nous jugeons nécessaires pour corréler les alertes issues des sondes de détection d'intrusions. Ce modèle est baptisé $\mathbf{M_2D^2}$, suivant les initiales de ses auteurs.

L'état de l'art évoqué ci-dessus montre que le modèle doit structurer les connaissances portant sur :

- l'environnement, c'est-à-dire la topologie du réseau, mais aussi les propriétés des entités du système d'informations,
- les vulnérabilités que peuvent exhiber ces entités et les attaques qui exploitent les vulnérabilités,
- les alertes et les événements qui font référence aux instances d'attaques,
- les outils de sécurité en général (dont font partie les IDS) qui surveillent les entités et produisent des alertes.

$\mathbf{M_2D^2}$ est un modèle qui fédère l'ensemble de ces concepts et les relations qui les lient ; il peut être perçu comme une infrastructure sur laquelle les systèmes de corrélation peuvent s'appuyer pour effectuer leur analyse.

Suivant les recommandations de McHugh selon qui les progrès en détection d'intrusions viendront des efforts de formalisation de la part des membres de la communauté,$\mathbf{M_2D^2}$ est un modèle formel.

$\mathbf{M_2D^2}$ constitue la contribution majeure de cette thèse. Il est décrit en chapitre 4 et a fait l'objet d'une publication à RAID 2002 (*Recent Advances in Intrusion Detection*).

3.2 Approches de Corrélation

Les approches de corrélation d'alertes en détection d'intrusions peuvent être divisées en trois grandes familles : les approches de corrélation *explicites*, *semi-explicites* et *implicites*.

La corrélation explicite part du principe que les intrusions contre les systèmes informatiques sont dirigées par un but et constituées de plusieurs étapes permettant d'atteindre ce but. La corrélation consiste alors à élaborer des scénarios d'intrusions *a priori* et à mettre en correspondance les alertes issues de analyseurs avec ces scénarios. D'une certaine manière, la corrélation explicite est aux alertes ce que l'analyse morphosciente est aux événements. D'ailleurs, la frontière entre les deux approches est parfois ténue car la distinction entre les événements et les alertes est toujours litigieuse dans la communauté de détection d'intrusions.

La corrélation semi-explicite est une généralisation de l'approche explicite. Elle exploite aussi les connaissances expertes sur les attaques, mais au lieu de reconnaître des scénarios préétablis, elle tente de détecter des cohérences dans les pré-requis et les conséquences des attaques détectées, révélatrices d'évolutions dans une intrusion.

Les deux approches précédentes tentent de reconnaître ou constituer des scénarios d'attaques. Les approches de corrélation implicite sont des approches de type *data-mining* : elle tentent d'extraire des connaissances implicites, d'identifier des tendances d'un ensemble d'alertes. L'objectif final est de réduire le volume d'informations soumis à l'opérateur pour analyse, tout en améliorant la sémantique des alertes. Contrairement aux approche explicite qui traitent chaque alerte individuellement, les approches implicites traitent les alertes dans leur globalité. D'une certaine manière, les approches implicites sont plus pragmatiques que les approches explicites car elles tentent avant tout de résoudre les problèmes des sondes de détection d'intrusions.

Il faut bien comprendre que la différence entre ces trois approche réside la modélisation plus ou moins explicite des *relations* entre les alertes : la corrélation explicite *reconnaît* les liens ; la corrélation implicite *découvre* les liens. La différence ne réside pas l'association de connaissances expertes pour décrire les alertes ; comme nous allons le voir, qu'elles soient implicites ou explicites, les approches sont amenées à utiliser des connaissances expertes sur les attaques ou l'environnement pour corréler les alertes. C'est d'ailleurs ce constat qui a motivé initialement la conception de $\mathbf{M_2D^2}$.

Après la présentation les travaux liés à la corrélation d'alertes en détection d'intrusions, nous évoquons l'existant en matière de corrélation d'alertes dans le domaine du diagnostic de pannes, ainsi que les différences entre ces deux domaines.

Le tableau 3.2 fait une synthèse des approches de corrélation d'alertes en détection d'intrusions, avec leur propriétés. Nous nous y réferons par la suite.

Auteurs	Réf.	Type de Corrélation				
		Implicite			Semi-explicite	Explicite
		Agrégation	Synthèse	Qualification		
Cuppens *et al*	[16]					√
	[14]	√	√			
	[15]				√	
Valdes et Skinner	[78]	√	√			
Julisch	[43, 45]	√	√	√		
Michel et Mé	[60]					√
Ning *et al*	[63, 62]				√	
Dain et Cunningham	[18, 19]	√	√			
Manganaris *et al*	[52]	√	√			
Qin et Lee	[69]	√	√			
Debar et Wespi	[23]	√	√			

3.2.1 Corrélation explicite

La corrélation explicite part du principe que les intrusions contre les systèmes informatiques sont dirigées par un but et constituées d'un ensemble d'étapes permettant d'atteindre ce but. Chaque étape est une action ou une attaque qui participe à l'accomplissement des suivantes. Par exemple, l'acquisition d'informations sur la cible d'une attaque constitue une des prémices d'une attaque exploitant une vulnérabilité découverte.

Dans ce contexte, la corrélation explicite consiste à envisager *a priori* différentes stratégies d'intrusions du point de vue d'un attaquant et à les modéliser sous la forme de scénarios. La corrélation consiste alors à mettre en correspondance le flux d'alertes avec les étapes définies dans les scénario. Lorsque l'ensemble des étapes d'un scénario sont observées dans le flux d'alertes et que les contraintes logico-temporelles liant ces étapes sont respectées, le scénario est reconnu et une alerte de haut niveau est soumise à l'opérateur.

Les opérateurs logico-temporels typiquement proposés dans les scénarios pour combiner les événements sont les suivants (nous utilisons les notations proposées par Cuppens [16]) :

- La séquence $a = a_1; a_2$: a est reconnue si une observation de a_1 est suivie d'une observation de a_2.
- Le choix $a = a_1?a_2$: a est reconnu si a_1 ou a_2 sont observés
- La négation $a = \bar{b}$: a est reconnu si b n'est pas observé,
- L'ordre partiel $a = a_1 \& a_2$: a est reconnu si a_1 et a_2 sont observés, indépendamment de l'ordre.

Dans [16], Cuppens et Ortalo proposent un langage de spécification de scénarios

scenario : $(A_1; (A_2 \& A_3) \& A_4 \& A_5); A_6$

where
$$\begin{aligned}
&action(A_1) = rpcinfo(DestIP) \quad \wedge \\
&action(A_2) = showmounte(DestIP) \quad \wedge \\
&action(A_3) = showmounta(DestIP) \quad \wedge \\
&action(A_4) = finger(DestIP) \quad \wedge \\
&action(A_5) = create_account(Username, Userid) \quad \wedge \\
&action(A_6) = mount(P, /\text{mnt})
\end{aligned}$$

FIG. 3.3 – Un exemple de scénario

d'attaques baptisé Lambda.

Un scénario spécifié en Lambda contient à la fois une description de l'état du système avant et après l'intrusion ainsi que de l'enchaînement des étapes qui constituent l'intrusion.

Chaque étape est une collection d'attributs, exprimés dans le langage des prédicats. Les attributs sont par exemple l'action effectuée, l'acteur et la date d'occurrence de l'étape. Les étapes sont combinées entre elles par les opérateurs évoqués ci-dessus. Un exemple de scénario est donné en figure 3.3. Ce scénario est composé de cinq étapes $A_1, \ldots, A_5$, dont l'attribut *action* est défini. On constate que l'identifiant de l'action est le foncteur (*showmounte* par exemple) d'un prédicat dont les variables sont les paramètres de l'action (en l'occurrence, l'adresse IP $DestIP$ de la cible). L'enchaînement des actions est spécifié en entête du scénario.

L'état du système avant l'intrusion est décrit par une collection de prédicats. Ces prédicats constituent les pré-conditions du scénario. Ils permettent de spécifier les conditions nécessaires au succès de l'intrusion, relatives soit à l'attaquant soit au système attaqué.

De la même manière, les post-conditons permettent de décrire les conséquences d'une intrusion sur l'état du système et de l'attaquant.

Par exemple, l'une des pré-condition du scénario proposé en Figure 3.3 est $use\_service(DestIP, \text{fingerd})$ qui indique que l'hôte dont l'adresse IP est $DestIP$ doit exhiber le service `fingerd` afin que l'action `finger(DestIP)` réussisse. Une post-condition du scénario est $can\_access(P)$ qui indique que l'attaquant a accès à la partition P de l'hôte.

Les pré-conditions et les post-conditions sont décrites plus en détail dans la section suivante, dédiée à la corrélation semi-explicite.

Le scénario d'attaque contient l'enchaînement des actions à effectuer, du point de vue de l'attaquant. Comme certaines de ces actions ne sont ni observées (parce qu'elles ne constituent pas une attaque en soi), ni observables (parce qu'elles sont effectuées dans l'environnement de l'attaquant), un scénario Lambda contient en plus un scénario de détection. Le scénario de détection est la combinaison des actions détectables par les analyseurs. Ce sont donc en fait les actions du scénario de détection qui permettent d'instancier les variables utilisées dans le scénario d'attaque. Par exemple, dans le scénario décrit en Figure 3.3, l'action $create\_account(Username, Userid)$ n'est pas

observable car elle est effectuée sur la machine de l'attaquant. Elle n'apparaît donc pas dans la combinaison d'action détectables par les analyseurs.

Notons que dans [15], Cuppens évoque la corrélation abductive, qui consiste à générer des alertes virtuelles dont l'occurrence permettrait de corréler des alertes observées pour reconnaître un scénario complet d'attaque.

Enfin, une section vérification permet de spécifier les actions à entreprendre pour vérifier le succès de l'intrusion. Nous ne la détaillons pas ici.

Dans cet état de l'art, nous nous sommes volontairement cantonnés à la description détaillée des approches de corrélation explicite. Il faut noter qu'il existe des similitudes fortes entre les scénarios utilisés en corrélation d'alertes explicite et les signatures complexes proposées par des analyseurs morphoscients tels que ASAX, proposé par Habra *et al.* [36], LogWeaver, proposé par Roger et Goubault-Larreq [71] ou bien Sutekh, proposé par Ducassé et Pouzol [66, 67]. Les événements analysés par ces outils sont des appels système produits par les systèmes d'exploitation des hôtes. Pour leur part, Michel et Mé proposent aussi un langage de signatures baptisé Adèle [60] pour analyser des événements hétérogènes.

Ces similitudes ne sont pas surprenantes. En effet, comme nous l'avons évoqué en introduction de cette section (page 41), la corrélation explicite peut être vue comme une transposition des approches d'analyse morphosciente au niveau des alertes. La question suivante revient d'ailleurs de manière récurrente lors de discussions sur le sujet de la corrélation explicite : lorsque c'est possible, pourquoi déporter au niveau des alertes une analyse qui peut être effectuée au niveau des événements ? En effet, il semble naturel de détecter les attaques au plus proche de la source de données (i.e. dans un analyseur). Ce sont essentiellement des problèmes de performances qui justifient le déport d'une analyse complexe à un plus haut niveau. En ce qui concerne les données système, la simple collecte des données s'avère coûteuse et a un impact sensible sur les performances des hôtes. Si à cette collecte s'ajoute l'analyse, alors les pertes de performances peuvent devenir intolérables sur des serveurs en exploitation. Concernant les données réseau, la collecte n'a pas d'importance puisque la surveillance est effectuée par des machines dédiées. Cependant, plusieurs études [72, 47] montrent que les sondes présentent des limitations liées au débit des réseaux, dès 100 mbit/s (ce qui est relativement modeste) et ce, même avec une analyse simple (Snort). En conséquence, nous concevons plutôt les analyseurs comme des *filtres* d'événements intrusifs, produisant des alarmes destinées à un manager en charge de l'analyse élaborée.

Cependant, la différence essentielle entre une alerte et un événement réside dans le niveau d'analyse auquel ils sont soumis : une alerte est le produit d'une analyse ayant pour objectif de déterminer si une action malveillante a été entreprise ou non ; un événement n'a pas subi d'analyse. Indépendament de la nature du composant qui les produit, on peut donc dire que les alertes sont des types d'événements particuliers, qui font référence à d'autres événements dont le caractère intrusif est avéré (selon les critères de l'analyseur). Cette différence correspond bien aux définitions de IDMEF [83], selon lequel un événement est *une occurrence dans une source de données, produite par un capteur à destination d'un analyseur, qui* peut *donner lieu à une alerte.* Une alerte est *une notification formattée de l'occurrence d'un événement pertinent [du point de vue de la sécurité] produite par un analyseur, à destination d'un manager.*

Par conséquent, dans la mesure les opérateurs logico-temporels utilisés dans les signatures des outils évoqués ci-dessus sont les mêmes que ceux utilisés dans les scénarios, à savoir la conjonction, la négation, la disjonction et la séquence, rien n'empêche *a priori* d'utiliser ces outils non plus pour analyser des événements, mais pour analyser des alertes.

Dans cette thèse, nous avons choisi d'appliquer le formalisme des chroniques, pour lequel il existe des bases théoriques solides et une implémentation efficace. Cette approche est décrite dans le chapitre 6.

3.2.2 Corrélation semi-explicite

La reconnaissance d'un scénario d'attaque dépend de la correspondance exacte des actions d'un attaquant avec celles prévues dans le scénario. Du fait du nombre d'actions envisageables, le nombre de scénarios d'attaques possibles est gigantesque. La spécification *a priori* de l'ensemble des scénarios par un expert nous parait donc illusoire. A ce constat s'ajoute le caractère malicieux des attaquants, qui ont précisément intérêt à adapter leur comportement de manière à tromper le système de reconnaissance.

Ce défaut des approches par scénarios tient au fait que les actions des attaquants sont des constantes dans les scénarios, contrairement aux autre attributs des alertes qui sont des variables instanciables par le système de reconnaissance (l'attaquant et la victime, par exemple).

L'évolution logique des approches par scénarios classiques consiste donc à modéliser les actions de l'attaquant par des variables. L'unification des variables correspondant aux actions est contrainte par une mise en correspondance entre les pré-requis et les conséquences des actions. De telles correspondances dénotent soit une évolution dans l'intrusion d'un attaquant, soit des tentatives répétées pour atteindre un même objectif.

Cuppens caractérise ce type de corrélation de semi-explicite [15] : des connaissances expertes explicites sont toujours nécessaires pour qualifier les effets et les pré-requis des actions, mais la notion de scénario prédéfini disparaît. En effet, l'objectif de la corrélation n'est plus de mettre en correspondance un flux d'alertes avec un ensemble d'actions prédéfinies, mais de lier des alertes lorsqu'une cohérence dans les actions des attaquants est constatée.

Une famille de travaux est basée sur ce type d'approche. Dans [75], Templeton and Levitt proposent un langage de description d'attaques, Jigsaw, centré autour de l'expression des pré-requis d'une attaque et de ses effets vis à vis de l'attaquant. Il est important de noter que Jigsaw est destiné à décrire des attaques, c'est-à-dire les briques de base d'une intrusion. Jigsaw n'est pas conçu pour décrire des scénarios d'intrusions. À notre connaissance, les auteurs ne proposent pas de sémantique opérationnelle à leur langage.

Le langage Lambda proposé par Cuppens et Ortalo [16], décrit en Section 3.2.1, permet de spécifier les pré-requis et les effets des attaques. Les pré-requis et les effets sont définis respectivement dans les section pré-condition et post-condition d'un scénario. Dans le cadre de l'approche semi-explicite proposée par Cuppens et Miège [15], le corps d'un scénario se trouve réduit à une seule attaque (ou action), puisque l'objectif de la

pre : remote_access(A,DestIP) ∧ use_service(DestIP, `portmapper`)
post : knows(use_service(DestIP,`portmapper`))
scenario :A_1
where action(A_1)=rpcinfo(DestIP) ∧
actor(A_1) = A

FIG. 3.4 – Une action individuelle d'un scénario

corrélation semi-explicite n'est plus de mettre en correspondance les alertes avec les étapes prévues dans le corps d'un scénario, mais bien de mettre en correspondance les pré-requis avec les effets des attaques.

Par exemple, la Figure 3.4 illustre la spécification d'une étape d'attaque en Lambda. Il s'agit d'une brique élémentaire du scénario donné en Figure 3.3. L'attaque est une requête à un hôte dont l'adresse IP est DestIP. L'accès distant par l'attaquant A à l'hôte (remote_access(A,DestIP) et le fait que l'hôte exhibe le service `portmapper` use_service(DestIP, portmapper) constituent les pré-conditions de l'attaque. Le résultat de l'attaque est un accroissement de connaissance de la cible par l'attaquant (knows(use_service(DestIP,`portmapper`)))

Cuppens propose deux types de corrélations :

- La corrélation *post-pré* consiste à lier une attaque dont les post-conditions satisfont les pré-conditions d'une autre alerte. Ce type de corrélation dénote une évolution dans l'intrusion de l'attaquant.
- La corrélation *post-post* consiste à lier deux attaques dont les effets coïncident. Ce type de corrélation dénote des tentatives différentes d'un même attaquant pour atteindre un même objectif.

Formellement, une attaque A_1 est directement corrélable avec une attaque A_2 si au moins une des post-conditions de A_1 est unifiable avec au moins une des pré-condition de A_2.

Prenons l'exemple des vulnérabilités `CAN-2001-0476`[2] et `CAN-2001-1384`[3]. La première est une faille du script cgi Aspseek qui permet à un attaquant distant d'obtenir des privilèges utilisateur. La deuxième est une vulnérabilité des noyaux Linux qui permet à un attaquant d'obtenir les privilèges administrateurs, qui requiert des privilèges utilisateurs pour être exploitée. Une post-condition de `CAN-2001-0476` est donc $post($`CAN-2001-0476`$) = user\_access(T1)$ où la variable T représente la cible de l'attaque. Une pré-condition de `CAN-2001-1384` est $pre($`CAN-2001-1384`$= user\_access(T2)$. On constate que les la pré-condition de `CAN-2001-0476` et celle de `CAN-2001-0476` sont unifiables. Deux alertes faisant référence à des exploitation successives de `CAN-2001-0476` et `CAN-2001-1384` contre la même cible sont donc corrélables.

La condition de corrélation directe est étendue aux cas où les effets d'une attaque se traduisent par un gain de connaissance de l'attaquant sur sa cible : une attaque dont une post-condition est $knows(p)$ est corrélable avec une attaque dont la pré-condition est p.

[2] `http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0476`
[3] `http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-1384`

Considérons par exemple l'attaque `nmap-o` qui permet à l'attaquant de connaître le type de système d'exploitation utilisé sur un hôte. Une post-condition de cette attaque est modélisée par le fait $knows(use\_os(C, O))$, qui signifie qu'un attaquant sait qu'un hôte C utilise un système d'exploitation O. L'attaque `WinNuke` est un dénis de service qui concerne les machines Windows. Une pré-condition de cette attaque est $use\_os(C, windows)$. `nmap-o` est corrélable avec `WinNuke` car $use\_os(C, O)$ est unifiable avec $use\_os(C, windows)$.

Cuppens propose en outre d'impliquer des règles ontologiques fournies par un expert dans les conditions de corrélation. Ces règles se présentent sous la forme de pré-conditions et post-conditions. A titre d'illustration, remplaçons l'attaque `nmap-o` de l'exemple précédent par l'attaque `TCPScan`. `TCPScan` consiste à scanner les ports d'un hôte pour connaître les services qu'il exhibe. Une des post-conditions de cette action est modélisée par le prédicat $knows(use\_service(C, netbios))$, qui signifie que la cible C exhibe le service $netbios$. Ces deux attaques ne sont donc pas corrélables, puisque $use\_service(C, netbios)$ n'est pas unifiable avec $use\_os(C, windows)$. Cependant, le service netbios est caractéristique des machines Windows. Cette connaissance experte se traduit par une règle ontologique dont la pré-condition est $use\_service(H, netbios)$ et la post-condition est $use\_os(H, Windows)$. La condition de corrélation est étendue pour prendre en compte ces règles ontologiques : deux attaques sont corrélable si il existe une chaîne de règles ontologiques directement corrélables entre elles. Dans l'exemple, `TCPScan` est corrélable à `WinNuke` via la règle ontologique car $use\_os(C, windows)$ est unifiable avec $use\_os(H, windows)$ et $use\_service(C, netbios)$ unifiable avec $use\_service(H, netbios)$.

La spécification des pré-conditions et des post-conditions pour chaque attaque permet de générer automatiquement les scénarios envisageables. On se ramène alors à une corrélation explicite classique, comme décrite en Section 3.2.1. L'avantage de cette approche est que la rédaction des scénarios n'est pas à la charge de l'opérateur de sécurité. Il faut malgré tout noter qu'envisager de manière exhaustive l'ensemble des pré- et post-conditions possibles de toutes des attaques peut s'avérer difficile.

Dans [63, 62], Ning *et al* proposent une approche de corrélation semi-explicite tout à fait similaire à celle de Cuppens. Nous ne la détaillons donc pas ici.

3.2.3 Corrélation implicite

La corrélation explicite évoquée dans la section précédemment nécessite une spécification *a priori* des relations entre les alertes. Nous avons vu que cette contrainte était difficile à satisfaire. De plus, la proportion élevée de faux positifs ainsi que le caractère redondant et récurrent des alertes ne sont pas adaptés à une reconnaissance de scénarios d'attaques.

La corrélation implicite traite les alertes dans leur globalité, afin de fournir à l'opérateur de sécurité une vision synthétique de l'ensemble des alertes. On peut isoler trois fonctions principales de corrélation implicite : l'*agrégation*, la *synthèse* et la *qualification*. Le vocabulaire employé pour désigner ces trois fonctions de corrélation varie en fonction des auteurs.

L'agrégation consiste à regrouper les alertes selon différents critères. La synthèse

consiste créer une *méta-alerte*, qui réunit les informations contenues dans un groupe d'alerte. La qualification consiste à ajouter des informations aux alertes synthétiques, afin d'améliorer leur sémantique.

L'agrégation des alertes et leur synthèse réduisent quantitativement le nombre d'alertes soumise à l'opérateur, mais n'améliorent pas la qualité des alertes : la qualification d'une méta-alerte repose alors sur l'expertise de l'opérateur. Peu d'approches qualifient les méta-alertes.

Comme on peut le constater sur la figure **??**, l'agrégation et la synthèse sont des fonctions généralement couplées. En effet, dans plusieurs approches de corrélation implicite, l'agrégation d'une alerte dans un groupe existant repose sur une mesure de similarité entre l'alerte et la méta-alerte qui représente le groupe. Dans ces approches, la mesure de similarité est une combinaison des mesures de similarité entre les attributs des alertes. Nous les détaillons dans le premier paragraphe suivant.

Parmi les attributs utilisés pour comparer les alertes, l'attaque présente un intérêt particulier, car c'est elle qui porte la sémantique de l'alerte. Certaines approches se situent à la frontière entre les approches implicites et les approches semi-explicites. En effet, un des critères de similarité utilisé entre les attaques repose sur une correspondance entre leurs conséquences et leurs prérequis. Nous étudions en détail ces approches dans le deuxième paragraphe suivant.

a) Agrégation et synthèse d'alertes similaires

Le principe de base des approches d'agrégation consiste d'abord à définir une mesure de similarité entre deux alertes ou entre une alerte et une méta-alerte. Chaque occurrence d'alerte est alors comparée aux méta-alertes existantes. Si la similarité entre une alerte et une méta-alerte est forte, alors l'alerte est intégrée à la méta-alerte ; si aucune méta-alerte n'est jugée similaire, alors l'alerte forme un nouvelle méta-alerte.

Les alertes sont des $n-$uplets, dont les attributs sont classiquement un identifiant d'attaque, une source d'attaque, une cible, une date et l'identité de la sonde à l'origine de l'alerte. La fonction de similarité entre les alertes est une combinaison des mesures de similarité entre les attributs des alertes. Le succès de ces approches dépend en grande partie de la prise en compte, au niveau de la mesure de similarité, des particularités de chaque attribut que nous avons évoqué en section 3.1.

L'approche de Valdes et Skinner [78] est probabiliste. La similarité entre une alerte X et une méta-alerte Y est définie comme la moyenne pondérée des mesures de similarité entre les attributs communs à X et Y :

$$sim(X,Y) = \frac{\sum_i e_i sim_i(X_i, Y_i)}{\sum i e_i}$$

e_i est la pondération associée à l'attribut i ; X_i est la valeur de l'attribut i de l'alerte X. Y_i est une liste de valeurs prises par l'attribut i des alertes participant à la méta-alerte Y. La fonction sim_i est la fonction de similarité associée à l'attribut i, à valeur dans $[0;1]$.

L'opération de synthèse consiste à compléter les ensembles Y_i de la méta-alerte avec les valeurs des attributs X_i de l'alerte. Il s'agit donc d'une union au sens ensembliste

du terme.

Les attributs retenus pour représenter les alertes sont l'identifiant d'attaque, les adresses IP source et destination, l'identifiant de la sonde ayant généré l'alerte et la date d'occurrence. L'identifiant d'attaque est un terme désignant une catégorie d'attaque. Les auteurs proposent 14 catégories d'attaques, du type "dénis de service", "sondage", "violation de privilège", etc. L'affectation des identifiants d'attaques fournis par les sondes de détection d'intrusions à telle ou telle catégorie dépend de connaissances expertes. Notons que dans cette approche, les attaques n'appartiennent qu'à une seule catégorie, ce qui ne n'est généralement pas le cas dans la réalité.

Les fonctions de mesure de similarité sim_i entre attributs prennent en compte des caractéristiques propres à chaque attribut. Par exemple, deux adresses IP sources seront similaires si elles appartiennent au même sous-réseau. Concernant les identifiants d'attaques, une matrice de similarité entre les différente catégories d'attaques reflète la vraisemblance qu'une catégorie soit suivie d'une autre. Par exemple, la probabilité qu'une attaque de type "sondage", qui permet à un attaquant d'acquérir de l'information sur la cible, soit suivie d'une attaque de type "dénis de service" est élevée. La similarité de ces deux attaques est donc jugée forte.

Les pondérations e_i utilisées dans le calcul de similarité des alertes symbolisent la similarité *attendue* des attributs. Cette pondération permet d'accroître ou au contraire de réduire l'importance d'un attribut dans le calcul de similarité de l'alerte avec la méta-alerte. Cette pondération est dépend des autres attributs de l'alerte considérée, ainsi que des alertes précédemment incluse dans la méta-alerte. Par exemple, dans le cas d'une attaque dont la source peut être forgée, le poids associé à la mesure de similarité de la source est faible. En effet, si la source peut prendre n'importe quelle valeur, son influence vis-à-vis de la mesure de similarité globale devient négligeable. Dans cet exemple, l'attribut correspondant au type d'attaque a une influence sur l'attribut correspondant à la source.

En plus des pondérations utilisées pour diminuer ou accroître l'influence de certains paramètres, la similarité entre les alertes est contrainte par des seuils propres à chaque attribut. Si la similarité d'un attribut est inférieure son seuil, alors la similarité entre l'alerte et la méta-alerte est nulle. Ces seuils permettent d'influer sur la nature des méta-alertes formées. Par exemple, en diminuant le seuil lié à l'identité de l'analyseur, les méta-alertes peuvent impliquer des alertes issues de sondes hétérogènes ; en diminuant le seuil correspondant à la classe d'attaque, les méta-alertes peuvent impliquer des alertes qui correspondent aux différentes aux différentes étapes d'une intrusion.

Les valeurs de seuils et celles utilisées dans les fonctions de similarité sont empiriques.

Le critère d'agrégation proposé par Dain et Cunningham [18, 19] est très similaire à celui de Valdes et Skinner, mais seuls le type d'attaque, l'adresse IP source et la date d'occurrence sont utilisés comme attributs. La similarité inter-alertes est le produit des similarités inter-attributs. Il n'existe pas de notion explicite de similarité attendue comme dans l'approche de Valdes et Skinner, mais le type d'attaque a une influence sur la nature de fonction de similarité entre les dates d'occurrence des alertes. Par exemple, dans un dénis de service par engorgement (*flooding*), deux alertes sont d'autant moins similaires que les dates d'occurrence des alertes sont éloignées.

Comme dans l'approche de Valdes et Skinner, Dain et Cunnigham n'intègrent une occurrence d'alerte qu'au groupe qui maximise la fonction de similarité. En effet, le nombre total de groupes d'alertes possibles croît exponentiellement avec le nombre d'alertes[4], le système d'agrégation ne peut donc pas maintenir la totalité des sous groupes d'alertes possibles. Les groupes d'alertes ne peuvent donc que croître par ajouts successifs d'alertes ; ils ne peuvent pas être divisés et une alerte ne peut appartenir qu'à un seul groupe. Toutefois, contrairement à Valdes et Skinner, les groupes d'alerte ne sont pas synthétisés. Par conséquent, les occurrences d'alertes ne sont pas comparées à des méta-alertes, mais à la dernière alerte intégrée à chaque groupe. Ce choix est discutable car rien ne permet d'assurer que la dernière alerte ajoutée à un groupe est celle qui va maximiser la similarité globale.

Dans [14], Cuppens propose aussi une technique d'agrégation et de synthèse d'alertes. La différence entre l'approche de Cuppens et celles précédemment évoquées réside dans l'utilisation de règles logiques pour définir la similarité inter-alertes, à la place de fonctions de similarité probabilistes. Un peu à la manière des pondérations utilisées dans l'approche de Valdes et Skinner, les règles qui régissent la similarité des entre les alertes dépendent de la nature des attaques.

Notons que dans l'approche de Cuppens, une occurrence d'alerte peut être intégrée à plusieurs groupes existants. Par conséquent, des groupes d'alertes séparés peuvent se trouver réunis par ajout d'une alerte commune.

Les attributs des méta-alertes sont l'union des attributs des alertes. Notons que la synthèse des alerte est dissociée de l'agrégation : les groupes d'alertes sont construits incrémentalement et les méta-alertes sont constituées à partir des groupes. Dans l'approche de Valdes et Skinner, seules les méta-alertes sont constituées ; les groupes d'alertes ne sont pas conservés. Les alertes perdent donc leur individualité lors de la synthèse.

Manganaris *et al* [52] proposent de découvrir des règles d'association d'alertes au sein de bases de données d'alertes. La découverte de règles d'associations est un problème bien connu en fouille de donnée, en particulier dans le domaine de l'analyse de transactions commerciales [3], pour identifier les combinaisons fréquentes d'achats effectués par les clients d'un supermarché.

Partant du principe que les alertes fréquentes, réparties sur une longue période, sont généralement le fait de phénomènes normaux (i.e. non intrusifs), Manganaris propose de modéliser le comportement normal d'un système d'information par des règles d'associations issues des bases d'alertes.

Dans l'approche de Manganaris, une *transaction* est constituée de pics fréquentiels d'alertes, c'est-à-dire de groupes d'alertes dont le délai inter-alerte est faible. Une transaction est l'unité de base de l'analyse. L'analyse des transaction révèle des règles d'association entre les alertes d'un même groupe et permettent d'établir le profil de comportement normal d'un analyseur. Ensuite, les déviations constatées entre les alertes générées par l'analyseur et son profil permettent d'isoler les activité intrusives.

Dans cette approche, l'agrégation des alertes n'est donc pas basée sur une mesure de similarité sur la valeur des attributs, mais sur des correspondances fréquentielles.

[4] ce nombre correspond au nombre de parties d'un ensemble

Les groupes d'alertes sont qualifiés par le fait qu'ils dévient ou non du profil de comportement de l'analyseur.

Dans [69], Qin et Lee proposent une approche permettant de grouper des alertes causalement liées, sans connaissance *a priori* des propriétés des attaques. Leur approche est basée sur l'analyse de séries temporelles de variables et plus spécifiquement le test de causalité de Granger (TCG). Suivant l'approche de causalité de Granger, une variable Y est causée par une variable X si Y est mieux prédite par les valeurs passées de X et Y que par les valeurs passées de Y seulement.

Dans le contexte de la corrélation d'alertes, les séries temporelles sont des hyper-alertes, c'est-à-dire des séquences de méta-alertes identiques. Les méta-alertes proviennent de l'agrégation d'alertes identiques, survenues dans une fenêtre temporelle étroite (de l'ordre de la seconde). Le TCG permet de trouver l'ensemble des hyper-alertes causalement liées à une hyper-alerte donnée. Les méta-alertes sont ordonnées par ordre de gravité afin d'effectuer TCG sur les alertes prioritaires. La gravité d'un groupe d'alertes est fonction de la correspondance entre le type de cible d'une attaque et la cible réelle. Elle est calculée à l'aide d'un réseau Bayésien naïf qui modélise les propriétés des attaques et celles de l'environnement. La gravité d'un groupe d'alertes est d'autant plus haute que le type de cible d'une attaque coïncide avec la cible réelle.

L'atout présumé de cette approche réside dans la possibilité de lier causalement des alertes, sans connaissances *a priori* sur les propriétés des attaques. En effet, certaines alertes (en particulier celles fournies par des sondes comportementales) font référence à des identifiants d'attaques inconnus. Les approches de corrélation basées sur les correspondances entre les pré-requis et les conséquences d'attaques sont tributaires de la disponibilité de ces données.

Notons tout de même que l'auteur exploite des informations sur les propriétés des attaques, en particulier pour évaluer la gravité des attaques. De plus, du fait de son caractère statistique, un lien causal établi entre des hyper-alertes nécessite qu'un grand nombre d'alertes soient impliquées dans les hyper-alertes pour être significatif. En d'autres termes, une analyse statistique peut découvrir et reconstruire un scénario d'intrusions à partir des alertes à condition que le scénario soit répétitif. Or le non déterminisme des intrusions constitue justement l'une des critiques faite aux approches de corrélation explicite et qui justifie les approches de corrélation implicites.

Les alertes agrégées selon cette approche peuvent ensuite être soumises à des techniques de corrélation explicites, permettant de reconnaître des scénarios d'attaques connus. Ceci illustre la complémentarité des deux types de corrélation.

b) Méta-alertes en tant que *vues* des alertes

L'approche de corrélation de Debar et Wespi, décrite dans [23], est la première implémentation de solution de corrélation d'alertes dans un outil commercial (*Risk Manager*).

Dans le composant d'agrégation et de corrélation (ACC) de Risk Manager, les groupes d'alertes sont constitués d'alertes qui ont la même projection selon un certain nombre d'axes, les axes étant représentés par les attributs des alertes. Ces groupes sont appelés situations.

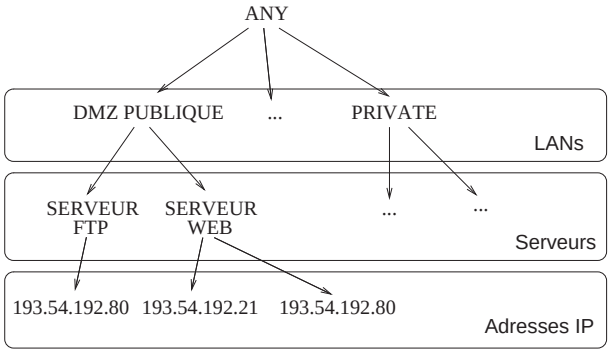

FIG. 3.5 – Taxinomie pour les cibles des attaques

Les situations soumises à l'opérateur sous forme de méta-alertes sont celles dont la sévérité est importante et dont le nombre d'alertes sous-jacentes est significatif.

Le nombre et la nature des axes utilisés pour la projection ont une signification particulière. Par exemple, des alertes ayant la même source et le même nom peuvent être révélatrices d'un attaquant tentant d'exploiter une même vulnérabilité, quelle que soit la victime ; des alertes ayant la même source et la même cible peuvent être révélateurs d'un attaquant intéressé par un hôte donné (un serveur web précis, par exemple).

Dans [43, 45], Julisch propose une variante d'une méthode de fouille de donnée appelée AOI (*Attribute-Oriented Induction*), qui permet de grouper les alertes en généralisant progressivement leur attributs. Les généralisations sont basées sur des connaissances expertes exprimées sous la forme de taxinomies propres à chaque attribut. Les taxinomies permettent de segmenter plus finement les sous-espaces de projections d'alertes, donnant ainsi aux méta-alertes plus de spécificité que dans l'approche de Debar et Wespi. En ce sens, l'approche de Julisch est une extension de celle de Debar et Wespi.

Une taxinomie est un ensemble fini de valeurs muni d'un ordre partiel. L'ordre partiel définit le niveau d'abstraction relatif des éléments de l'ensemble. Les attributs des alertes émises par les sondes de détection d'intrusions appartiennent aux éléments les plus spécifiques de la taxinomie, c'est-à-dire ceux pour qui il n'existe pas de valeur moins abstraite.

Dans l'approche de Julisch, les alertes sont des quadruplets constitués d'un identifiant d'attaque, d'une source, d'une cible et d'une date d'occurrence. Une taxinomie est définie pour chaque type d'attribut. La Figure 3.5 schématise la taxinomie associée à la cible des attaques. Plusieurs niveaux d'abstractions sont représentés sur cette figure : adresses IP, serveurs, LANs. Au niveau des alertes, les cibles sont identifiées par des adresses IP, et la valeur `Serveur Web` est plus abstraite que la valeur `193.54.192.80`. La valeur la plus abstraite est *any*, qui signifie *n'importe quelle cible*.

L'AOI consiste à abstraire progressivement les valeurs des attributs des données (les alertes) en suivant les taxinomies associées aux aux attributs et à fusionner les données rendues égales par l'abstraction. De cette manière le nombre total d'alertes est diminué et les valeurs des attributs sont plus abstraits. Par exemple, deux alertes

dont les cibles respectives sont `193.54.192.21` et `193.54.192.80` sont généralisées en remplaçant la cible par `Serveur Web`. Les deux alertes sont fusionnées, à condition que les autres attributs soient identiques.

Dans l'algorithme d'AOI conventionnel, les données sont abstraites jusqu'à ce que le nombre total de données devienne inférieur à un seuil fixé. Ce critère d'arrêt est mal adapté à la corrélation d'alertes car le nombre souhaitable d'alertes issues du processus d'abstraction n'est pas connu au préalable. De plus, dans le processus d'AOI conventionnel, l'abstraction d'un attribut est fait sur l'ensemble des données, même si cette abstraction n'est pas pertinente pour certaines données. Cette abstraction sauvage conduit à une sur-généralisation des données.

Julisch propose un algorithme d'AOI adapté. A chaque itération, cet algorithme traite un ensemble $\mathcal{L}_i$; un attribut k est sélectionné et les alertes de $\mathcal{L}$ sont généralisées en fonction de cet attribut, puis fusionnées, formant ainsi de nouvelles alertes α_k. Chaque alerte α_k représente ainsi une vue synthétique d'un sous-ensemble $L \subseteq \mathcal{L}_0$ de l'ensemble initial d'alertes. Les alertes α_k représentant des ensembles d'alertes de cardinalité supérieure à un seuil fixé sont soustraites de l'ensemble $\mathcal{L}_i$, puis soumises à l'opérateur car elles sont jugées suffisament abstraites. Le processus de généralisation est réitéré sur l'ensemble des alertes restantes. De cette manière, les alertes ne sont pas sur-généralisées.

Cette approche permet par exemple de constituer des alertes du type "un **serveur proxy** génère des **scans de ports** vers l'**extérieur**" (les termes en gras sont les attributs des alertes générées par le mécanisme de corrélation). **Serveur proxy** est la source issue de l'abstraction de l'adresse IP d'un serveur proxy ; **scan de port** est un identifiant d'attaque n'ayant subi aucune abstraction ; **extérieur** est la destination de l'attaque, issu de plusieurs abstractions de la cible.

L'approche de Julisch nous semble intéressante, car elle permet de traiter l'excès d'alertes, tout en améliorant le contenu des alertes. Les taxinomies permettent d'intégrer des connaissances environnementales, comme la topologie par exemple et de les exploiter pour qualifier les alertes. En effet, les alertes abstraites fournies à l'opérateur sont décrites par les attributs des taxinomies. Ces attributs abstraits constituent une information supplémentaire présentée à l'opérateur. Les alertes abstraites sont donc sémantiquement plus riches que les méta-alertes produites par les approches citées précédemment, dont les attributs sont la réunion des valeurs des alertes.

3.2.4 Corrélation d'alertes et diagnostic de pannes

Ces vingt dernières années, la corrélation d'alertes a fait l'objet de nombreuses études dans le domaine du diagnostic de *pannes* en milieu industriel. On pourra se référer à la thèse de Meira [58] pour un état de l'art des techniques de corrélation d'alertes dans le milieu des réseaux de télécommunications.

La corrélation d'alertes dans le diagnostic de pannes consiste à reconnaître une panne à partir de ses symptômes. Les symptômes sont des séquences d'événements générés par des capteurs chargés de surveiller l'état du système.

Les séquences d'alertes à reconnaître sont fournis par un expert, ou bien issus d'une phase d'apprentissage supervisé. Le système de corrélation compare le flux

d'événements avec les motifs de scénarios et effectue les actions spécifié lorsqu'une instance de scénario est reconnue.

De cette manière, le nombre d'alertes est réduit et leur sémantique est améliorée car les séquences d'événements reconnues sont qualifiées.

La transposition des techniques de corrélation issues du diagnostic de pannes à la corrélation d'alertes en détection d'intrusions consiste à faire une analogie entre les pannes et les intrusions. Toutefois, cette transpostion se heurte à certaines caractéristiques propres aux intrusions. Nous passons en revue ces caractéristiques et dans chaque cas, nous les discutons.

Non déterminisme Les manifestations de pannes connues sont des phénomènes déterministes. En détection d'intrusions, la simple présence d'un acteur humain dans la boucle, l'attaquant, introduit du non-déterminisme dans les activités analysées. Qui plus est, les attaquant ont *intérêt* à camoufler leurs actions afin de ne pas être détecté. Si les mêmes briques sont utilisées par des attaquant distincts pour construire un scénario d'attaques –ce qui justifie la détection morphosciente–, en revanche, nous pensons que l'agencement de ces briques n'est pas déterministe.

Toutefois, l'analyse des flots d'alertes produits en milieu opérationnel révèle que de plus en plus d'attaques sont le fait d'outils automatisés. Les séquences d'alertes engendrées par ces outils sont déterministes. L'utilisation de techniques de reconnaissance d'attaques est d'autant plus justifiée que ces outils d'attaques engendrent des séquences d'alertes récurrentes, dont la fréquence est élevée. Le regroupement et la qualification de ces alertes a alors un impact sensible sur la quantité globale d'alertes.

Faux positifs et faux négatifs Dans la majorité des cas, les capteurs chargés de détecter des pannes sont fiables : il génèrent peu de fausse alertes et les symptômes de pannes sont tous détectés. Comme nous l'avons montré dans la problématique, les sondes de détection d'intrusions émettent un majorité de fausses alertes. De plus, l'inverse, les sondes de détection d'intrusions ne détectent pas toutes les attaques.

Toutefois, une majorité des attaques ne donnent lieu qu'à un seul événement. Si cet événement peut être confondu avec la manifestation d'une activité normale, alors des fausses alertes sont générées en masse. Dans ce cas, comme aucun autre événement caractéristique de l'attaque n'est disponible, il est possible d'exploiter des événements caractéristiques d'activités anodines, afin de discriminer les vrais positifs des faux positifs. La reconnaissance d'une séquence d'alertes permet alors d'infirmer une hypothèse d'attaque. Une fois de plus, les techniques explicites montrent leur utilité.

Nombre de symptômes Le nombre de symptômes distincts manipulés pour le diagnostic de pannes est de l'ordre de quelques dizaines et ce nombre est relativement statique. En détection d'intrusions, les symptômes sont les identifiants d'attaques, dont font partie les identifiants de signatures. Les IDS morphoscients actuels possèdent plusieurs centaines de signatures (Snort en possède 2000). De plus, plusieurs nouvelles vulnérabilités sont découvertes chaque jour, qui donnent lieu à autant de signatures dans les IDS.

L'utilisation d'ontologies permet de réduire le nombre d'identifiants d'attaques. Toutefois, les séquences d'alertes évoquées ci-dessus impliquent des attaques très spécifiques, qu'une classification n'identifierait pas avec autant de précision.

3.3 Conclusion

Dans cet état de l'art, nous avons d'abord décrit les travaux portant sur une amélioration du contenu des alertes, c'est-à-dire des attaques et des identifiants de victimes et d'attaquants. Si l'ensemble des auteurs semblent s'accorder sur la nécessité de prendre en compte des informations supplémentaires, paradoxalement, peu propositions ont été formulées pour structurer ces informations dans un modèle.

Notre première contribution est de proposer un tel modèle, baptisé $\mathbf{M_2D}^2$. Nous le décrivons dans le chapitre 4.

Nous avons ensuite passé en revue l'ensemble des approches de corrélation d'alertes. On peut diviser ces approches en trois familles : explicite, semi-explicite et implicite.

La corrélation explicite consiste à mettre en correspondance les alertes avec des scénarios d'intrusions pré-établis. Comme le nombre d'attaques envisageables est grand, il est impossible à un opérateur de sécurité d'envisager tous les scénarios possibles.

La corrélation semi-explicite est une généralisation de la corrélation explicite, dans le sens où les actions des attaquants sont des variables du système de corrélation. Les scénarios intrusifs sont établis par le système de corrélation lorsque des coïncidences entre les pré-requis des attaques et leurs conséquences sont trouvées. Cette approche requiert des connaissances expertes sur les effets et les pré-requis des attaques.

Les deux approches précédentes sont relativement prospectives car les informations qu'elles exploitent ne sont pas disponibles à l'heure actuelle dans les alertes fournies par les outils de détection d'intrusions.

Les approches de corrélation implicites traitent les informations intrinsèques des alertes fournies par les outils de détection d'intrusion actuels et tentent de mettre en évidence des *tendances*, c'est-à-dire des regroupements intéressants d'alertes. Le regroupement des alertes *similaires* et la qualification des groupes permettent à l'opérateur de sécurité d'avoir une vue synthétique des ensembles d'alertes.

Nos deux autres contributions portent sur des approches de corrélation exploitant les informations modélisées dans $\mathbf{M_2D}^2$. La première contribution est une approche de corrélation implicite, qui est une application de l'analyse de concepts logique proposée par Ferré et Ridoux [30]. Nous la présentons en chapitre 5.

Notre troisième contribution est une approche de corrélation explicite basée sur les chroniques. Les chroniques sont un formalisme issu du diagnostic de pannes, proposé par Dousson [25]. Nous la présentons en chapitre 6.

Chapitre 4

M_2D^2

4.1 Introduction

L'étude des approches de corrélation existantes montre que quelle que soit l'approche adoptée (implicite, semi-explicite ou explicite), des connaissances supplémentaires, absentes des alertes, sont nécessaires pour corréler les alertes.

Bien que la plupart des auteurs s'accordent implicitement sur la nécessité d'utiliser ces informations, peu de travaux proposent de les modéliser de manière cohérente et aussi exhaustive que possible.

Dans ce chapitre, nous proposons un modèle baptisé $\mathbf{M_2D^2}$[1], qui fédère les informations que nous jugeons nécessaires à la corrélation des alertes. $\mathbf{M_2D^2}$ constitue ainsi une infrastructure sur laquelle les approches de corrélations peuvent se baser pour collecter les informations nécessaires à la corrélation d'alertes. La structure générale de $\mathbf{M_2D^2}$ est schématisée en figure 4.1. On peut classer les connaissances nécessaires en quatre catégories :

- les propriétés des entités du système d'information surveillé, que nous appelons la cartographie,
- les propriétés des attaques et des vulnérabilités,
- la configuration des outils de sécurité,
- les événements et en particulier les alertes.

Dans [56], McHugh suggère que la détection d'intrusions bénéficierait d'un effort de formalisation de la part de la communauté. Pour répondre à ce besoin, $\mathbf{M_2D^2}$ est un modèle relationnel de données, formalisé à l'aide des notations de la méthode B et Z [2]. Un résumé de ces notations est disponible en annexe A.

Dans les quatre premières sections de ce chapitre, nous décrivons les quatre catégories de concepts évoqués ci-dessus. Nous les présentons dans l'ordre suivant : propriétés du système d'informations, attaques et vulnérabilités, outils de sécurité et événements. Nous optons pour cet ordre car un système d'informations est constitué d'entités physiques et logiques qui présentent des vulnérabilités que des attaques exploitent. Les outils de sécurité surveillent le système d'informations afin d'empêcher ou de détecter les attaques. Ces outils de sécurité consomment et produisent des

[1] l'acronyme est issu des initiales des auteurs

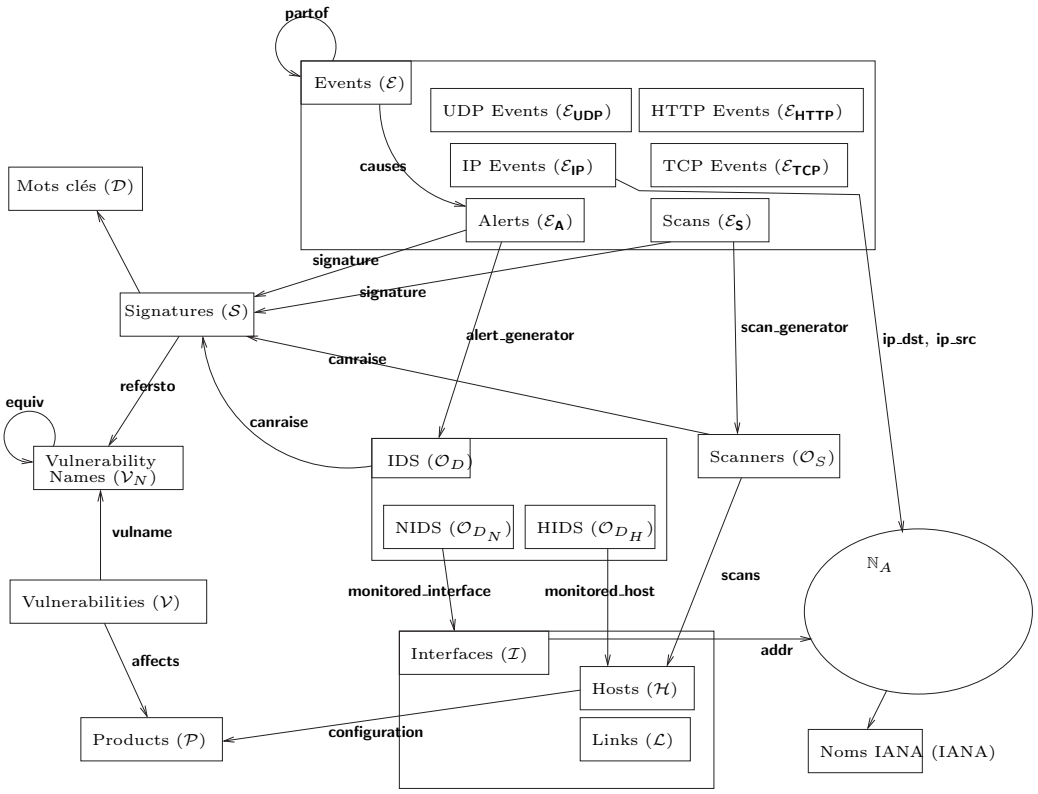

FIG. 4.1 – $\mathbf{M_2D^2}$

événements.

Chacune des parties dédiées à la description des concepts de $\mathbf{M_2D^2}$ est divisée en quatre sections. Une section **Description** décrit de manière informelle les concepts et les relations entre les concepts ; une section **Acquisition des données** décrit comment les données sont collectées ; en effet, la conception de $\mathbf{M_2D^2}$ a été en partie régie par la disponibilité des informations additionnelles permettant d'améliorer la sémantique. Une section **Formalisation** formalise les concepts décrit préalablement ; la section **Limitations** donne des informations sur les éventuels problèmes subsistants dans le modèle.

Dans la dernière nous concluons et évoquons les perspectives et améliorations nécessaires de $\mathbf{M_2D^2}$.

4.2 Cartographie

Dans le contexte de la corrélation d'alertes, nous définissons la cartographie comme l'inventaire des propriétés du système d'informations surveillé. Ces propriétés regroupent à la fois les informations sur la topologie du réseau, les outils logiciels existant sur les hôtes du réseau, ainsi que leur configuration.

Justifions d'abord l'utilisation d'informations cartographiques par rapport aux objectifs de la corrélation d'alertes exprimées dans la problématique (section 2.3).

Evaluation de la sévérité Les propriétés du système d'information sont indispensables pour la corrélation d'alertes car elles permettent d'enrichir le contenu des alertes et évaluer leur sévérité en comparant les pré-requis d'une attaque avec les propriétés effectives des victimes. L'évaluation de la sévérité peut aussi avoir un impact sur le volume d'alertes présenté aux opérateurs qui veulent uniquement être alertés des attaques susceptibles d'aboutir. En supprimant les alertes relatives à des attaques qui échouent, le volume global d'alertes présenté à l'opérateur diminue de manière très importante.

Enrichissement du contenu des alertes Comme nous l'avons évoqué dans la section 2.1.2, les cibles des attaques sont désignées sous des formes variées dans les alertes, peuvent être entachées d'erreurs et sont peu informatives vis-à-vis d'un opérateur humain. Il est donc nécessaire d'associer les informations cartographiques à la corrélation d'alertes pour identifier précisément les cibles et les sources des attaques.

Fausses alertes L'observation des alertes en milieu opérationnel montre que des configurations particulières peuvent provoquer un volume important de fausses alertes dans un système d'informations. Ces configurations impliquent des entités du système d'informations ayant des fonctions particulières. Le cas des serveurs mandataires évoqué en section 2.2.1 (page 26) est un exemple de fonction. La connaissance de la présence de telles entités permet d'identifier des alertes réputées pour être des fausses alertes.

Visibilité des sondes La cartographie permet de modéliser la visibilité topologique d'une sonde, c'est-à-dire son aptitude à détecter les manifestations d'une attaque par sa localisation dans le système d'informations. La visibilité topologique permet d'optimiser la couverture des IDS (minimiser le nombre de sondes pour éviter les redondances d'alertes tout en maximisant la couverture). Elle permet aussi de gérer des conflits entre sondes, c'est-à-dire être capable de justifier l'absence de réaction d'une sonde, relativement à une alerte générée par une autre sonde, par le fait que les manifestations d'une attaque ne sont pas accessibles à la première.

4.2.1 Description de la cartographie

Nous divisons la cartographie en deux grandes parties : la topologie et les éléments logiciels. La topologie désigne les entités physiques du système d'informations (hôtes, routeurs, passerelles, liens) et leur interconnexions. Les éléments logiciels sont les outils logiques (systèmes d'exploitations, services) qui sont hébergés par les hôtes.

Les informations cartographiques sont centralisées et utilisées au niveau d'un manager (au sens IDMEF du terme [83]). On pourrait envisager de les distribuer au niveau des sondes mais ces informations peuvent être volumineuses et on ne sait pas *a priori* quelles informations seront nécessaires sur quelle sonde.

a) Topologie

Les concepts du modèle de topologie concernent essentiellement des entités internes du système d'informations. Ce sont en effet les seules sur lesquelles les administrateurs et les opérateurs de sécurité ont des connaissances. Les seules informations disponibles pour désigner des attaquants externes autrement que par leur adresse IP consiste à utiliser les noms des autorités propriétaires de ces adresses. Ces informations sont fournies par les branches régionales du *Internet Assigned Numbers Authority* (IANA). Les branches régionales du IANA sont des organismes en charge de l'attribution des plages d'adresses IP à des autorités (fournisseurs d'accès, entreprise commerciales, universités, etc.). Elles sont au nombre de quatre, l'ARIN, le RIPE, l'APNIC et le LACNIC, qui gèrent respectivement l'Amérique et l'Afrique du sud, l'Europe, l'Asie et l'Amérique Latine.

Comme l'illustre IDMEF [83], le concept de cible d'une attaque recouvre des notions de niveaux de granularité variés : processus, compte utilisateur, hôte ou réseau. Le concept d'hôte est central dans $\mathbf{M_2D^2}$. En effet, un hôte exécute des processus, est utilisé par des utilisateurs, appartient à des réseaux.

Les entités du système d'informations sont interconnectées par l'intermédiaire d'une interface et forment un réseau. La notion de topologie d'un réseau s'applique théoriquement à toutes les couches OSI (physique à applicative). Indépendemment de la couche considérée, les modèles topologiques existants sont assez naturellement basés sur des graphes. C'est la sémantique associée aux noeuds et arcs du graphe qui apportent aux réseaux leur spécificité.

La littérature du domaine concerne essentiellement les topologies des couches 2 et 3 du modèle OSI, c'est-à-dire respectivement la topologie physique qui implique les liaisons de données (e.g. Ethernet) et la topologie logique qui implique la couche réseau et le routage (e.g. IP). Dans le cadre de $\mathbf{M_2D^2}$, nous nous focalisons sur les réseaux Ethernet/IP/TCP car ce sont les plus classiques.

Dans le cas des modèles de topologie physique, les nœuds du graphe représentent les équipements réseau (hôtes, routeurs, concentrateurs, commutateurs, passerelles) ; les arcs représentent les connexions physiques entre les équipements.

Dans le cas des modèles de topologie logique, les nœuds du graphe représentent les adresses IP du réseau. Il existe un arc entre deux nœuds si les deux adresses IP se situent à un saut l'une de l'autre.

Le choix du modèle est donc dicté par le type de topologie requis. Dans le cas de $\mathbf{M_2D^2}$, la topologie doit permettre en particulier d'évaluer la visibilité topologique d'une sonde. Or, pour analyser un datagramme, une sonde réseau doit être située sur la route empruntée par ce datagramme, ce qui implique une connaissance de la topologie logique (voir page 75).

Toutefois, pour avoir accès au trafic qui ne lui est pas destiné, une interface d'écoute doit être située soit sur le même concentrateur, soit sur le port mirroir d'un commutateur[2] situé sur la route du datagramme. Commutateurs et concentrateurs font partie des outils d'interconnexion de niveau 2. Le modèle cartographique doit donc inclure

[2] un commutateur peut être configuré de telle sorte que le trafic destiné transitant par un port particulier soit dupliqué sur un autre port, dit port mirroir

des informations de topologie physique. Le modèle de topologie de $\mathbf{M_2D^2}$ est inspiré de celui de Vigna [79], que nous décrivons en section 4.2.3.

b) Eléments logiques

Une majorité d'attaques exploitent des vulnérabilités liées à des erreurs d'implémentation, de configuration ou de conception des logiciels. Les descriptions des vulnérabilités fournies par le Mitre ou BugTraq font références aux versions de logiciels vulnérables. Ainsi, en comparant les spécifications d'une vulnérabilité avec la configuration effective de l'hôte cible, nous pouvons évaluer les risques de succès d'une attaque.

Considérons par exemple la description de la vulnérabilité `CAN-2003-0083` suivante :

Apache 1.3 before 1.3.25 and Apache 2.0 before version 2.0.46 does not filter terminal escape sequences from its access logs, which could make it easier for attackers to insert those sequences into terminal emulators containing vulnerabilities related to escape sequences, a different vulnerability than CAN-2003-0020.

Cette vulnérabilité concerne les versions 1.3.25 à 2.0.46 du serveur Web Apache. Pour reprendre les catégories proposées en section 2.3.2 (page 33), cette vulnérabilité serait qualifiée d'impossible sur un hôte hébergeant un serveur IIS. La sévérité de l'alerte correspondante serait ainsi réduite.

Pour mesurer la menace réelle que représente une attaque exploitant une vulnérabilité, il est donc nécessaire de connaître la nature et la version des logiciels présents sur les hôtes du système d'informations.

Dans $\mathbf{M_2D^2}$, les logiciels sont appelés des produits. Dans d'autres approches comme celles de Vigna et Kemmerer [80, 81] ou bien celle de Goldman *et al* [34], les logiciels sont appelés des services.

4.2.2 Acquisition des données cartographiques

Dans cette partie, nous décrivons comment sont obtenues les données cartographiques. Comme dans la section précédente, nous présentons d'abord l'acquisition des informations topologiques puis l'acquisiton des informations sur les produits.

a) Topologie

Dans un contexte où la topologie des réseaux est de plus en plus dynamique, l'acquisition et la maintenance de la topologie doit être automatisée, *a fortiori* lorsque le réseau modélisé est de grande taille et où les administrateurs n'ont pas toujours une maîtrise totale des éléments du réseau. L'acquisition automatisée de la topologie des réseaux est appelée découverte topologique.

L'élaboration de techniques de découverte topologique n'entre pas dans le cadre de cette thèse. Nous évoquons simplement des travaux existants ayant trait à cette discipline. Dans la version actuelle de $\mathbf{M_2D^2}$, les informations topologiques de doivent être fournies manuellement par l'administrateur du système d'informations.

Les travaux sur la découverte de topologies physiques sont relativement récents. Breitbart *et al* proposent un algorithme de découverte de topologie physique [10] qui repose sur SNMP (*Simple Network Management Protocol*). Cette approche permet

d'opérer une découverte de réseaux hétérogènes de manière fiable, à condition que l'ensemble des nœuds du réseau soient équipées d'agents SNMP, ce qui n'est pas toujours le cas. La RFC 2922 proposée par L'IETF décrit les informations nécessaires à la découverte de topologies physiques [9].

Lowekamp [50] propose un autre algorithme de découverte de topologie physique qui ne repose pas sur SNMP, mais sur des heuristiques exploitant des informations retournées par des outils standard tels que ping, traceroute ou des transfers de zone DNS, par exemple. Malheureusement, les heuristiques et la qualité des informations exploitées par cette approche la rendent peu fiable.

Contrairement aux outils d'interconnexion de niveau 2 du modèle OSI (comme les concentrateurs) qui n'ont pas connaissance de leurs voisins, les outils d'interconnexion de niveau 3 (les routeurs) ont explicitement connaissance de leurs voisins pour effectuer leur fonction de routage. La découverte de topologies logiques et la constitution du graphe sous-jacent s'appuie sur les informations disponibles au sein même des routeurs.

En ce qui concerne les informations externes au système d'information, les branches régionales du IANA mettent publiquement à disposition les bases de données contenant des informations sur les propriétaires des plages d'adresses IP. Nous nous intéressons aux champ `netname` qui associe un identifiant d'autorité aux bornes inférieure et supérieure des plages d'adresses IP affectées à ces autorités. Une même autorité peut posséder plusieurs plages disjointes. C'est le cas en particulier de l'autorité `IP2000-ADSL-BAS`, qui est un fournisseur d'accès à internet, auquel sont associées 2310 plages d'adresses. D'autres autorités comme `FR-RPN-HOLDING` possède une plage d'adresses réduite à un singleton.

b) Outils Logiciels

Comme pour l'acquisition des informations topologiques, il existe deux techniques d'inventaire logiciel. L'une repose sur une plate-forme dédiée et l'autre sur des heuristiques.

Dans la première approche, un agent est installé sur chaque hôte du réseau et fournit à la demande la configuration de l'hôte. Cette solution présente l'avantage d'être fiable car les agents ont directement accès aux informations voulues, mais un agent est nécessaire par hôte, ce qui peut représenter une tâche d'administration non négligeable.

La seconde approche consiste à utiliser un *scanner* pour interroger les services d'hôtes à distance et deviner leur configuration par des heuristiques. Par exemple, le scanner Nmap[3] permet de deviner le système d'exploitation d'un hôte en se basant sur des imprécisions des spécifications de TCP/IP, qui donnent lieu à des implémentation différentes en fonction des systèmes d'exploitation. En soumettant à un hôte des datagrammes particuliers et en analysant la réponse, il est possible de d'approcher la version de système d'exploitation. C'est le choix adopté par Valdes *et al* dans [65].

Comme l'interrogation se fait à distance, aucune modification des hôtes n'est nécessaire et un seul *scanner* permet d'acquérir la configuration d'un ensemble d'hôtes. Cette approche présente donc essentiellement l'avantage d'être plus simple à mettre en œuvre que de déployer des agents sur l'ensemble des hôtes du réseau. Cependant les

[3]http ://www.insecure.org/nmap/

résultats obtenus sont imprécis, voire erronés. De plus, seuls les logiciels serveurs sont identifiables par ces outils, puisque l'inventaire est fait à distance. Enfin, les actions effectuées engendrent un trafic important et particulièrement intrusif, susceptible de faire réagir les outils de détection d'intrusions.

4.2.3 Formalisation

Dans cette section, nous formalisons les concepts liés à la cartographie dans $\mathbf{M_2D}^2$, c'est-à-dire la topologie et les logiciels. La formalisation de la topologie est donnée en figure 4.2.

a) Topologie

Comme nous l'avons vu, $\mathbf{M_2D}^2$ doit permettre de modéliser des détails complexes d'interconnexion physiques. Le modèle de topologie est donc basé sur la notion d'interface réseau.

Un réseau est un hypergraphe défini sur l'ensemble des interfaces du réseau, noté $\mathcal{I}$. Il s'agit donc d'un modèle de topologie physique. Rappelons que dans un graphe conventionnel, les arcs sont des couples de sommets ; dans un hypergraphe, les arêtes sont des sous-ensembles de sommets, appelées hyperarêtes.

L'ensemble des arêtes de l'hypergraphe est constitué de deux sous-ensembles qui forment chacun une partition de l'ensemble des interfaces. Ces deux sous-ensemble sont les hôtes ($\mathcal{H}$) et les liens ($\mathcal{L}$) du réseau. Dans la terminologie des réseaux, un lien est un réseau *local*, c'est-à-dire un LAN (*Local Area Network*). Par définition d'une hyperarête, un hôte et un lien sont formellement des sous-ensembles d'interfaces. De fait, un hôte peut contenir plusieurs interfaces et par définition, un lien relie plusieurs interfaces.

A titre d'exemple, dans la famille des réseaux de la norme IEEE 802.3, un lien du graphe modélise un concentrateur. Un hôte est une station de travail, un serveur, un routeur, une passerelle ou un commutateur.

La Figure 4.3 illustre le graphe d'un exemple de réseau. Les sommets du graphe sont les interfaces ($\mathcal{I}$), représentées par des points. Les arêtes sont les hôtes ($\mathcal{H}$), représentés par des cercles et les liens ($\mathcal{L}$), représentés par des segments. L_1 et L_2 sont des concentrateurs, H_4 est une passerelle qui relie les liens L_1 et L_2. Les autres hôtes sont des stations de travail.

La topologie logique du réseau est obtenue à partir de la topologie physique en définissant une injection **addr** qui à une interface fait correspondre une adresse IP, appartenant à l'ensemble $\mathbb{N}_A$.

Le routage est défini à partir d'une fonction **next** qui à un couple de liens $(l_i, l_j) \in \mathcal{L}^2$ associe l'ensemble des liens l_k tels qu'il existe un chemin de l_k à l_j et $l_i \cap l_k \neq \emptyset$. Si aucune route n'existe entre l_i et l_j ou si $l_i = l_j$, alors l'ensemble des liens est vide.

La fonction qui à deux interfaces associe l'ensemble des liens participant à un chemin de l_i à l_j est la fonction **route**. Elle est définie inductivement par

$$\mathbf{route}(l_i, l_j) = \{l_i\} \bigcup_{l_k \in \mathbf{next}[l_i, l_j]} \mathbf{route}(l_k, l_j)$$

Concepts

$\mathbb{N}_A$	est l'ensemble des adresses IPv4
$\mathcal{I}$	est l'ensemble des interfaces réseau
$\mathcal{H} \subseteq \mathcal{P}(\mathcal{I})$	est l'ensemble des hôtes ($\mathcal{P}(\mathcal{I})$ désigne l'ensemble des parties de $\mathcal{I}$)
$\mathcal{L} \subseteq \mathcal{P}(\mathcal{I})$	est l'ensemble des liens (*i.e.* réseaux locaux)
HN	est l'ensemble des noms système des hôtes
NN	est l'ensemble des noms réseau des hôtes
DN	est l'ensemble des noms de réseaux locaux
IANA	est l'ensemble des identifiants d'autorités dans la base IANA

Relations

$$\begin{aligned}
\textsf{addr} &\in \mathcal{I} \rightarrowtail \mathbb{N}_A \\
\textsf{nat} &\in \mathbb{N}_A \nrightarrow \mathbb{N}_A \\
\textsf{host} &\in \mathcal{I} \rightarrow \mathcal{H} \\
\textsf{hostname} &\in \mathcal{H} \rightarrow \mathrm{HN} \\
\textsf{dns} &\in \mathrm{NN} \rightarrowtail \mathbb{N}_A \\
\textsf{lanname} &\in \mathcal{L} \twoheadrightarrow \mathrm{DN} \\
\textsf{iana} &\in \mathbb{N}_A \nrightarrow \mathrm{IANA} \\
\textsf{next} &\in \mathcal{L}^2 \nrightarrow \mathcal{P}(\mathcal{L}) \\
\textsf{route} &\in \mathcal{L}^2 \nrightarrow \mathcal{P}(\mathcal{L})
\end{aligned}$$

Définitions

Un réseau est un hypergraphe $(\mathcal{I}, N)$ défini sur $\mathcal{I}$ où $N = (E_1, E_2, \ldots, E_m)$ est l'ensemble des hyperarêtes $E_i \subset \mathcal{I}$.

$\mathbb{N}_A = \{0, \ldots, 2^{32} - 1\}$

$$\textsf{route}(l_i, l_j) = \{l_i\} \bigcup_{l_k \in \textsf{next}[l_i, l_j]} \textsf{route}(l_k, l_j)$$

Propriétés

Les hôtes et les liens partitionnent chacun l'ensemble des interfaces :

$$\bigcup_{i=1}^{p} H_i = \mathcal{I} \text{ and } H_i \cap H_j = \emptyset \ (\forall i, j \in \{1, 2, \ldots, p\} \text{ and } i \neq j)$$
$$\bigcup_{i=1}^{q} L_i = \mathcal{I} \text{ and } L_i \cap L_j = \emptyset \ (\forall i, j \in \{1, 2, \ldots, q\} \text{ and } i \neq j)$$

$\textsf{next}(l_i, l_j) = \emptyset$ si $l_i = l_j$ ou si il n'existe pas de chemin de l_i à l_j

FIG. 4.2 – Modèle de topologie dans $\mathbf{M_2D^2}$

Nous contraignons les interface à avoir une et une seule adresse IP, donc **addr** est une injection. Certains systèmes d'exploitations permettent d'affecter plusieurs adresses IP à une même interface, mais ce type de configuration est suffisamment exceptionnelle pour que nous la négligions. Une interface peut ne pas avoir d'adresse IP, mais elle ne peut pas faire l'objet d'interactions au sein du réseau. Nous négligeons donc aussi cette situation. **addr** n'est pas une bijection car l'ensemble $\mathbb{N}_A$ désigne la totalité des adresses

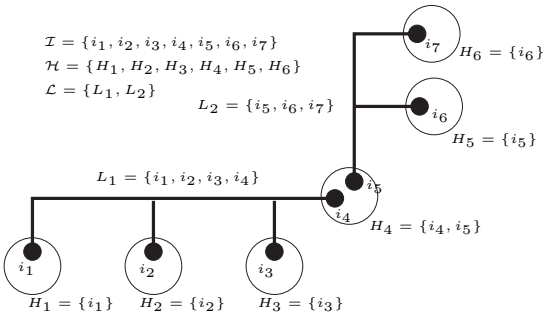

FIG. 4.3 – Un exemple de réseau

IP, qui ne sont pas toutes associées à une interface, bien entendu.

La fonction de translation d'adresses (NAT, *Network Address Translation*) statique utilisée dans les réseaux est modélisée par la fonction partielle **nat**, qui à une adresse IP associe une autre adresse IP en laquelle elle est translatée. Plusieurs adresses peuvent être translatées en une même adresse, mais une même adresse ne peut être translatée en plusieurs autres, c'est pourquoi **nat** est une fonction partielle.

L'hôte correspondant à une interface est donné par la fonction **host**. **host** est une fonction totale parce que l'ensemble des hôtes forme une partition de l'ensemble des interfaces. **host** est définie de la manière suivante :

$$\mathbf{host}[i] = h \in \mathcal{H} : i \in h$$

L'ensemble HN représente les noms systèmes des hôtes. Chaque hôte possède un et un seul nom, mais plusieurs hôtes peuvent posséder le même nom, ce qui justifie l'emploi d'une fonction totale.

L'ensemble NN représente l'ensemble des noms réseaux. Un nom réseau est un identifiant sous lequel est connu un hôte dans le réseau par une mécanisme de résolution de nom. La relation **dns** (DNS signifie *Domain Name Service*) modélise un tel mécanisme en associant un nom à une adresse IP et inversement. **dns** est une injection partielle car une adresse IP peut n'être associée à aucun nom réseau ; en revanche les noms réseau sont associés à une et une seule adresse IP. En pratique, les mécanismes de résolution de nom permettent d'associer plusieurs noms à une même adresse IP, mais un seul sert de référence pour l'adresse IP, les autres noms étant définis comme des équivalents du nom de référence. Toutefois, comme d'une part les outils de sécurité ne manipulent pas les noms équivalents et que d'autre part la résolution se fait dans le sens IP vers nom, nous ne jugeons pas nécessaire de prendre en compte les noms équivalents qui alourdiraient inutilement le modèle.

L'ensemble DN est constitué des noms de réseaux locaux. Nous supposons que tous les réseaux locaux ont un et un seul nom, donc la fonction **lanname** est une bijection.

L'ensemble IANA contient les identifiants d'autorités de la base IANA (champ *netname*). La fonction partielle **iana** associe à une adresse IP l'identifiant de l'autorité qui possède la plage d'adresse qui l'inclut. **iana** est une fonction partielle car il existe des plages d'adresses IP qui ne sont affectées à aucun organisme.

Concepts

$\mathcal{P}$	est l'ensemble des produits
VN	est l'ensemble des identifiants de produits
PN	est l'ensemble des noms de produits
PV	est l'ensemble des versions de produits
PT	est l'ensemble des types de produits.

Relations

$$\begin{aligned} \textbf{prodname} &\in \mathcal{P} \rightarrow PN \\ \textbf{version} &\in \mathcal{P} \rightarrow PV \\ \textbf{prodtype} &\in \mathcal{P} \rightarrow PT \\ \textbf{configuration} &\in \mathcal{H} \rightarrow \mathcal{P}(\mathcal{P}) \\ <\in &\; PV \leftrightarrow PV \end{aligned}$$

Définitions

$PT = \{\texttt{OperatingSystem}, \texttt{LocalApp}, \texttt{httpServ}, \texttt{ftpServ}, \ldots, \texttt{other}\}$

$PV = \mathbb{N}^p$ ($\mathbb{N}$ est l'ensemble des entiers)

$\forall v_1, v_2 \in \mathbb{N}^p, v_1 < v_2 \iff \exists j | \forall i, 1 \leq i \leq j, v_1[i] = v_2[i] \wedge v_1[j] < v_2[j]$

FIG. 4.4 – Produits dans $\mathbf{M_2D^2}$

b) Logiciels

La formalisation des produits (i.e. les logiciels) est donnée en Figure 4.4.

Un produit est un élément logiciel exécuté par un hôte du réseau. L'ensemble des produits est noté $\mathcal{P}$. A chaque produit est associé un et un seul nom appartenant à l'ensemble PN. Le nom est une chaîne de caractère qui identifie le logiciel et éventuellement l'entreprise qui le développe. Le nom d'un produit est fourni par la fonction totale **prodname**.

A chaque produit est aussi associée une version appartenant à l'ensemble PV, obtenue par la fonction totale **version**. Une version est un $n-$uplet $(n_1, ..., n_p)$ où p est un paramètre fixé *a priori* ; $p = 3$ suffit généralement pour désigner les versions des logiciels. PV est muni d'une relation d'ordre total, permettant de comparer les versions. Notons que deux versions distinctes d'un même produit sont modélisées par deux éléments différents dans l'ensemble $\mathcal{P}$, ce qui explique pourquoi **version** n'est pas une relation.

Enfin, nous associons à chaque produit un type, fourni par la fonction totale **prodtype**. Le type d'un produit est un élément de l'ensemble `{OperatingSystem,xxxServer,LocalApp,other}` où `xxxServer` représente n'importe quel logiciel de type serveur en écoute sur le réseau ; `xxx` doit être remplacé par une chaîne qui identifie le protocole applicatif correspondant (`http`, `ftp`, `snmp`, etc.). `OperatingSystem` désigne les systèmes d'exploitation et `LocalApp` concerne les produits logiciels avec lesquels seules des interactions locales sont possibles (i.e. aucune interaction *via* le réseau).

Par exemple, (`ApacheGroup/Apache`, `1.3.26`, `httpServer`) identifie le serveur Web

Apache produit par ApacheGroup, dont la version est 1.3.26.

La relation **configuration** lie un produit à un hôte qui l'exécute. Comme un hôte exécute plusieurs produits et qu'un produit peut être exécuté par plusieurs hôtes, la relation **configuration** est une relation. **configuration**(h) désigne l'ensemble des produits exécutés par h et est appelé la configuration de h.

4.2.4 Limitations

Dans le modèle actuel, les informations qui composent la cartographie d'un système d'informations sont atemporelles, contrairement aux événements (cf 4.5). Dans la réalité, les changements que subissent ces informations sont de plus en plus dynamiques. Par exemple, les routes dans un réseau peuvent changer[4], les logiciels subissent des changements de version, etc.

Bien sûr, il est possible de mettre à jour les informations contenues dans $\mathbf{M_2D}^2$, mais l'historique des modifications subies par les entités est perdue. Il est donc impossible de savoir, par exemple, si une cible ayant subi une mise à jour *était* vulnérable à la date d'une attaque donnée.

Dans le cadre des travaux futurs, il nous semble nécessaire de prendre en compte le dynamisme des informations cartographiques en ajoutant les informations temporelles sur les changements qui interviennent dans le système d'information.

Prenons l'exemple des alertes générées par les sondes réseau. Les hôtes impliqués dans les alertes sont identifiés par leur adresse IP. Les systèmes d'informations utilisent de plus en plus le protocole DHCP pour affecter aux hôtes du réseau une adresse IP. Au cours du temps, un même hôte peut se voir affecter des adresses IP distinctes. Un opérateur qui souhaite connaître l'ensemble des alertes dans lesquelles un hôte donné est impliqué ne peut donc pas se fier à l'adresse IP. Deux solutions sont possibles pour résoudre ce problème : soit mettre à jour les associations entre les adresses et les interfaces des hôtes et lier directement les alertes à l'hôte au moment de l'insertion de l'alerte, soit rendre événementielles les associations entre les adresses IP et les interfaces. Nous opterons pour la seconde solution.

Concernant les logiciels, comme nous l'avons vu, l'acquisition des données passe soit par le déploiement d'agents sur l'ensemble des hôtes du système, soit par l'utilisation de *scanners* qui interrogent les hôtes à distance et utilisent des heuristiques pour deviner la configuration des hôtes. La première approche implique une tache d'administration importante et surtout une maitrîse de l'ensemble des équipements du réseau, ce qui est d'autant plus difficile que les équipes d'administration et de sécurité ne sont pas forcément les mêmes et que les équipements peuvent être géographiquement séparés. C'est tout de même l'approche qui tend à se mettre en place dans certains réseaux. Au moment de la rédaction de cette thèse, c'est par exemple le cas dans les réseau de l'IRISA et de France Télécom R&D. Nous n'avons donc pas pu profiter des informations relevées par les agents installés à France Télécom R&D dans le cadre de nos expérimentations à cause de leur installation tardive.

[4]Notons qu'au sein du réseau d'un système d'information, les changements de routage sont peu fréquents donc le caractère statique du modèle topologique est raisonnable

La quantité la non innocuité du trafic engendré par la seconde approche d'acquisition cartographique la rendent difficile à utiliser dans des réseaux opérationnels de grande taille.

A l'avenir, nous envisageons d'étudier une variante passive de cette dernière technique d'acquisition de la cartographie. Elle consiste à inférer les informations cartographiques non plus en interrogeant les hôtes, mais en observant les transactions réseau dans lesquelles les hôtes sont impliqués, à la manière des sondes de détection d'intrusions. Les problèmes de volume et de nocivité du trafic s'en trouvent *de facto* résolues puisque l'approche ne produit aucun trafic.

4.3 Attaques

Dans la section précédente, nous avons formalisé les connaissances sur les caractéristiques du système d'informations. Nous formalisons maintenant les connaissances relatives aux attaques.

4.3.1 Description des attaques

Nous cherchons un moyen de traiter automatiquement la sémantique des attaques référencées dans les alertes. Comme nous l'avons vu en section 2.1.1, nous identifions trois catégories d'informations permettant de désigner les attaques : les identifiants de signatures des sondes de détection d'intrusions, les mot-clés issus de la documentation des signatures et les noms de vulnérabilités. $\mathbf{M_2D^2}$ intègre l'ensemble de ces informations.

a) Identifiants de signatures

Les identifiants de signatures des analyseurs morphoscients constituent un moyen élémentaire pour désigner les attaques. Ils présentent en outre l'avantage d'être des identifiants uniques pour les signatures, ce qui facilite la modélisation. Ces identifiants peuvent se présenter sous la forme d'entiers (e.g. Snort) ou bien de chaînes de caractères (e.g. Dragon). Lorsqu'il s'agit d'entiers, les identifiants ne sont pas fournis tel quel à l'opérateur ; c'est la documentation qui leur est associée qui est utilisée.

b) Mots-clé

Nous avons vu en section 2.1.1 que plusieurs taxinomies d'attaques ont été proposées par différents auteurs. A notre connaissance, aucune de ces taxinomies n'a été utilisée pour structurer les informations contenues dans la documentation des sondes de détection d'intrusions existantes. Ceci peut éventuellement expliquer qu'aucune des taxinomies proposées n'ait été adoptée par l'ensemble des membres de la communauté.

Dans $\mathbf{M_2D^2}$, nous proposons une approche empirique pour décrire les attaques, qui consiste à élaborer une taxinomie des attaques à partir des mots-clé de la documentation des signatures de sondes morphoscientes. Ainsi, on s'assure que la taxinomie est directement utilisable à partir des alertes, puisque les identifiants de signatures sont directement associés à un ensemble de mots-clé.

Identifiant Snort	Documentation
1425	`WEB-PHP content-disposition`
1739	`WEB-PHP DNSTools administror authentication bypass attempt`
1740	`WEB-PHP DNSTools authentication bypass attempt`
1741	`WEB-PHP DNSTools access`
803	`WEB-CGI HyperSeek hsx.cgi directory traversal attempt`
1076	`WEB-IIS repost.asp access`
1110	`WEB-MISC apache source.asp file access`
340	`FTP EXPLOIT overflow`
1247	`WEB-FRONTPAGE rad overflow attempt`

FIG. 4.5 – Exemple de documentation associée à des signatures de Snort

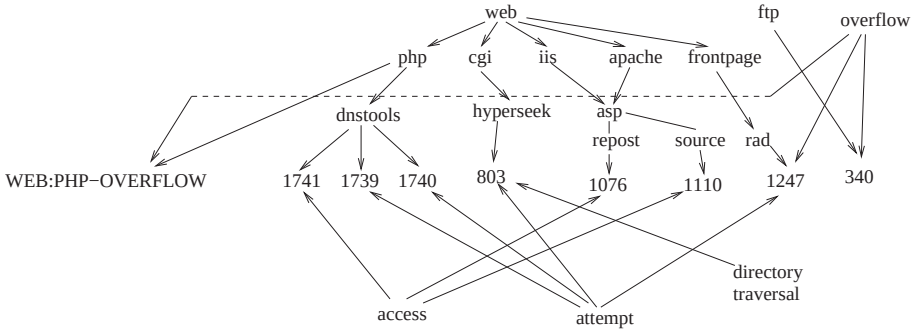

FIG. 4.6 – Une taxinomie des attaques

Prenons un exemple pour illustrer nos propos. Le tableau 4.5 contient la documentation associée à des identifiants de signature Snort. La figure 4.6 représente ces mêmes informations, agencées dans une taxinomie. Comme la documentation n'obéit à aucune format, l'identification des mots-clé pertinents et des relations taxinomiques repose uniquement sur les connaissances d'un expert.

Sur cette figure, une flèche d'un terme a à un terme b signifie que a est sémantiquement plus abstrait que b. Du point de vue du type de cible visée, il y a deux grandes classes d'attaques, `web` et `ftp`. À l'exception des qualificatifs `overflow`, `directory traversal`, `access` et `attempt`, les autres qualificatifs sont des sous catégories de `web` et `ftp`. `iis` et `apache` dénotent des modèles de serveurs `web` ; `php` et `cgi` sont indépendantes du type de serveur ; `dnstools` et `hyperseek` sont respectivement des scripts `php` et `cgi` ; le module `asp` peut être utilisé par des serveurs de type `apache` ou `iis`. Il n'y a qu'une seule attaque de type `ftp` schématisée ici, correspondant à la signature 340 de Snort. On peut voir que `overflow` est un type d'attaque indépendant du type de cible (`web` ou `ftp`) visé. Les attaques par `overflow` perme-

ttent à l'attaquant de faire exécuter des commandes illicites à la victime ; ce type d'attaque concerne n'importe quel type de processus, c'est la raison pour laquelle il est indépendant de la nature (`web` ou `ftp`) de la cible. Il en va de même pour le qualificatif `directory traversal`. Enfin, `access` et `attempt` désignent respectivement un accès à une cible et une tentative avérée d'exploiter une vulnérabilité. Encore une fois, ces deux qualificatifs peuvent s'appliquer à presque toutes les attaques.

Cette taxinomie s'applique aussi aux signatures utilisées par d'autres sondes de détection d'intrusions. Par exemple, la signature Dragon[5] `WEB:PHP-OVERFLOW` s'insère facilement dans la portion de taxinomie schématisée en Figure 4.6. On peut noter que contrairement aux identifiants de signature Snort qui sont des entiers, les identifiants Dragon sont des chaînes de caractères qui contiennent directement les mots-clé utiles.

c) Vulnérabilités

Les vulnérabilités offrent non seulement un référentiel de nommage unique entre les IDS, mais aussi une description des caractéristiques des attaques qui les exploitent.

Shirey et Howard définissent une vulnérabilité comme *une faille dans la conception, l'implémentation ou la configuration d'un logiciel ou matériel, susceptible d'être exploitée pour contourner les mécanismes censés assurer la sécurité* [39, 73]. Dans cette définition, le concept de logiciel correspond à un logiciel de $\mathbf{M_2D^2}$(cf page 66).

Comme on peut le constater en consultant leur description, les vulnérabilité n'affectent généralement pas qu'un seul produit. La vulnérabilité d'un hôte est souvent le résultat de la présence conjuguée de plusieurs produits sur l'hôte. Par exemple, un serveur web est vulnérable à condition d'héberger une version spécifique du serveur Apache *avec* une version spécifique d'un système d'exploitation Linux. Une faille dans la conception d'un système peut affecter l'ensemble de ses implémentations. Par exemple, la vulnérabilité `CAN-2002-0012` référence une faille dans les spécifications du protocole SNMP qui entraîne la vulnérabilité de l'ensemble de ses implémentations.

Arlat *et al* définissent une vulnérabilité comme *une erreur latente présente sur une cible* [7], où une *cible* correspond au concept d'*hôte* de $\mathbf{M_2D^2}$. Nous ne définissons pas de relation directe entre les vulnérabilité et les hôtes dans $\mathbf{M_2D^2}$. Cette relation est obtenue de manière transitive, par l'intermédiaire des produits. Autrement dit, un hôte est dit vulnérable si sa configuration correspond aux spécifications de la vulnérabilité.

En plus des informations sur les configurations affectées, les vulnérabilités fournissent d'autres propriétés des attaques. Compte tenu des informations disponibles dans les bases de données de vulnérabilité, nous avons retenu deux types de propriétés : les privilèges requis pour exploiter la faille et les privilèges acquis en cas d'attaque réussie. Ces deux critères permettent d'identifier des scénarios intrusifs dont les étapes se manifestent par des attaques logiquement liées (voir section 3.2.2).

Les vulnérabilités considérées dans $\mathbf{M_2D^2}$ sont celles référencées dans la liste CVE (*Common Vulnerabilities and Exposures*), proposée par le Mitre. La liste CVE a été initiée par Mann et Christey [54] afin de fournir un référentiel de nommage et de description des vulnérabilités. Les identifiants CVE sont les plus utilisés pour référencer des vulnérabilités dans la documentation associée aux signatures des sondes.

[5] `http://dragon.enterasys.com`

Les noms de vulnérabilités CVE sont formées d'un préfixe `CVE` ou `CAN`, suivi de l'année de découverte et d'un indice. Le préfixe indique indique si la vulnérabilité est candidate (CAN) ou officielle (CVE).

Il existe d'autres systèmes de nommage des vulnérabilités, qui sont utilisés dans la documentation des signatures des sondes de détection d'intrusions. Citons BugTraq[6], les avis du CERT[7], les identifiants Nessus[8] et Xforce[9].

4.3.2 Acquisition des descriptions d'attaques

Nous reprenons ici chacun des types d'identifiants d'attaque et nous indiquons comment ils sont obtenus.

a) Identifiants de signatures

L'acquisition des identifiants de signatures se fait en analysant les fichiers de configuration des sondes déployées dans le système d'informations.

b) Qualificatifs d'attaques

La taxinomie d'attaques utilisée dans le cadre de $\mathbf{M_2D^2}$ est construite à partir de la documentation des signatures de Snort. Deux raisons justifient ce choix. D'abord, de part le nombre de ses contributeurs, la base de signatures de Snort est la plus complète et la documentation des signatures d'autres IDS s'insére dans cette taxinomie. Ensuite, nous avons choisi d'utiliser Snort dans le cadre de nos expérimentations, il était donc logique d'élaborer la taxinomie à partir de ses signatures.

La documentation des signatures Snort est définie dans le champ `msg:` des signatures (voir par exemple page 12).

Notons que le temps nécessaire à un expert pour constituer la taxinomie est raisonnable. A titre indicatif, dans le cadre de nos expérimentations, la construction manuelle de la structure taxinomique de 98 identifiants de signatures nécessite de l'ordre de 4h de travail à une personne. La construction de la taxinomie pour les 1400 signatures de la base Snort nécessiterait quelques jours de travail.

c) Vulnérabilités

Comme le font remarquer Mann et Christey, CVE est une liste de noms et non une ontologie des vulnérabilités [54] ; l'objectif du Mitre n'est pas de fournir une description structurée des vulnérabilités.

Heureusement, des organismes mettent à disposition des bases de données qui structurent les descriptions des vulnérabilités fournies par le Mitre. Ainsi, le NIST (*National*

[6] `http://www.securityfocus.com/bid`
[7] `http://www.cert.org`
[8] `http://www.nessus.org/`
[9] `http://xforce.iss.net/`

Portée	Conséquence	Type	Composant visé
Locale	Disponibilité	Erreur de validation d'entrée	Système d'exploitation
Extérieure	Intégrité	Erreur de validation d'accès	Pile protocolaire
	Confidentialité	Erreur de configuration	Logiciel utilisateur
	Accès administrateur	Situation de compétition	Logiciel administrateur
	Accès utilisateur	Erreur de conception	Matériel
		Condition exceptionnelle	Protocole de communication
		Erreur d'environnement	Module de chiffrement

FIG. 4.7 – Informations ICAT sur les vulnérabilités

Institute of Standards and Technology) propose la base ICAT[10], que nous utilisons dans $\mathbf{M_2D^2}$. Citons aussi OSVDB[11] (*Open Source Vulnerability Database*).

Le schéma relationnel de la base ICAT donne accès aux propriétés des vulnérabilité ainsi qu'à la définition des configurations logicielles vulnérables. Le tableau 4.7 résume les propriétés utilisées pour classer les vulnérabilités dans le cas de la base ICAT.

$\mathbf{M_2D^2}$ne contient que les descriptions des vulnérabilités CVE car ce sont les seules fournies par la base ICAT. Toutefois, $\mathbf{M_2D^2}$ prend en compte les équivalences entre les différents systèmes de nommage. De cette manière, il est possible de manipuler des alertes faisant référence à l'exploitation d'une vulnérabilité, indépendemment du nom de vulnérabilité utilisé par la sonde.

Le Mitre met à disposition les équivalences entre le nommage CVE et d'autres systèmes de nommages répandus. En fait, il ne s'agit pas d'une équivalence *stricto sensu* : comme le fait remarquer Shirey [54], il existe des identifiants faisant référence à plusieurs vulnérabilités CVE et un même identifiant CVE peut être référencé par plusieurs identifiants d'autres systèmes de nommage.

Les associations entre les identifiants de signatures et les noms de vulnérabilité sont disponibles dans les définitions de signatures des sondes de détection d'intrusions.

4.3.3 Formalisation

Le modèle formel des descriptions d'attaques est résumé en Figure 4.8.

$\mathcal{S}$, $\mathcal{D}$ et $\mathcal{V}$ désignent respectivement les ensembles d'identifiants de signatures, les mots-clés d'attaques issus de la documentation des IDS et les vulnérabilités CVE. L'ensemble $\mathcal{V}_N$ est l'ensemble des noms de vulnérabilités de plusieurs organismes (dont les noms CVE, bien entendu).

La relation **descr** associe à un identifiant de signature l'ensemble des mots-clés présents dans sa description. L'ensemble des mots-clés est muni d'une relation d'ordre partiel $\prec$ qui définit la structure taxinomique.

Les associations entre identifiants de signatures sont modélisées par la relation **refersto**. **refersto** est une relation car une signature peut ne faire référence à aucune

[10] `http://icat.nist.gov`
[11] `http://osvdb.org`

Concepts

$\mathcal{S}$	est l'ensemble des identifiants d'attaques
$\mathcal{D}$	est l'ensemble des mots-clés d'attaques
$\mathcal{V}$	est l'ensemble des vulnérabilités CVE
$\mathcal{V}_N$	est l'ensemble des identifiants de vulnérabilités (CVE et autres)

Relations

refersto $\in$	$\mathcal{S}$	$\leftrightarrow$	$\mathcal{V}_N$
descr $\in$	$\mathcal{S}$	$\leftrightarrow$	$\mathcal{D}$
$\prec\in$	$\mathcal{D}$	$\leftrightarrow$	$\mathcal{D}$
affects $\in$	$\mathcal{V}$	$\leftrightarrow$	$\mathcal{P}(\mathcal{P})$
con $\in$	$\mathcal{V}$	$\rightarrow$	$\{$`Disponibilité`, `Integrité`, `Confidentialité`, `AccèsAdmin`, `AccèsUtilisateur`$\}$
req $\in$	$\mathcal{V}$	$\rightarrow$	$\{$`Extérieur`, `Local`$\}$
vulname $\in$	$\mathcal{V}$	$\rightarrowtail$	$\mathcal{V}_N$
equiv $\in$	**vulname**$[\mathcal{V}]$	$\leftrightarrow$	$\mathcal{V}_N$

Définitions

vulname$[\mathcal{V}]$ est l'ensemble des vulnérabilités CVE.

Commentaires

affects est une relation parce qu'une vulnérabilité peut affecter plusieurs configurations et une même configuration peut être affectée par plusieurs vulnérabilités.

FIG. 4.8 – Modèle de vulnérabilité de $\mathbf{M_2D^2}$

vulnérabilité aussi bien qu'à plusieurs et une vulnérabilité peut être associée à plusieurs signatures.

Si nous reprenons l'exemple de la signature Snort donnée en figure 1.1, page 12, nous avons les relations suivantes :

refersto(1999, cve-2001-1020)
refersto(1999, nessus-11104)

L'ensemble $\mathcal{V}$ est constitué des vulnérabilités CVE. Les identifiants de vulnérabilités (tout systèmes de nommage confondus) sont contenus dans l'ensemble $\mathcal{V}_N$. Chaque vulnérabilité de $\mathcal{V}$ possède un et un seul nom dans $\mathcal{V}_N$, donné par la surjection **vulname**.

L'ensemble $\mathcal{V}_N$ est muni d'une relation **equiv** qui fournit l'ensemble des équivalences entre les noms CVE et les noms d'autres systèmes de nommage. Cette relation permet de grouper les noms de vulnérabilités qui font référence à une seule et même vulnérabilité. Idéalement, **equiv** devrait être une relation d'équivalence entre les vulnérabilités CVE et les autres noms. Mais dans la réalité, ça n'est pas le cas. Une vulnérabilité non CVE peut donc être en relation avec plusieurs vulnérabilités CVE.

Dans $\mathbf{M_2D^2}$, une vulnérabilité affecte une *configuration*, c'est à dire un ensemble

de produits dont la présence conjuguée sur un hôte le rend vulnérable. Cette relation est modélisée par la relation **affects**. De cette manière, un hôte est affecté par une vulnérabilité si la configuration de l'hôte *inclut* la configuration vulnérable, c'est à dire si l'ensemble de produits qui constituent la combinaison vulnérable est incluse dans l'ensemble des produits constituant la configuration de l'hôte :

$$vulnerable(h, v) \iff \textbf{affects}(v) \subseteq \textbf{configuration}(h)(h \in \mathcal{H}, v \in \mathcal{V})$$

Par rapport aux critères de classification de ICAT (tableau 4.7), seules les caractéristiques relatives à la portée et les conséquences des vulnérabilité sont utilisées dans $\mathbf{M_2D^2}$. Les autres ne nous semblent pas nécessaires à la corrélation d'alertes. Par exemple, pour un opérateur de sécurité, savoir qu'une attaque permet d'acquérir des privilèges administrateur est utile, mais savoir que l'attaque résulte d'une situation de compétition (*race-condition*) ou d'une erreur de validation d'entrée (*input validation*) importe peu.

La fonction totale **req** fournit les privilèges nécessaires à l'exploitation d'une vulnérabilité et correspond aux valeurs de *portée* de la base ICAT. La fonction totale **con** fournit les conséquences de l'exploitation d'une vulnérabilité. Si l'on se réfère au gain maximum évoqué dans la problématique (page 33), on peut associer les valeurs `AccèsUtilisateur` et `AccèsAdmin` à l'exécution de code, la valeur `Confidentialité` à l'obtention d'information et les valeurs `Disponibilité` et `Intégrité` au dénis de service.

4.3.4 Limitations

Un inconvénient du modèle tient au fait que la totalité des informations utilisées pour décrire les attaques proviennent d'analyseurs morphoscients. Les alertes issues de sondes comportementales s'insèrent mal dans $\mathbf{M_2D^2}$. En effet, par définition, les sondes comportementales ne *reconnaissent* pas les attaques ; elles ne peuvent donc pas faire référence à un identifiant de signature ni à une vulnérabilité.

4.4 Outils de sécurité

Nous avons vu dans les section précédentes que les propriétés de l'environnement surveillé doivent être prises en compte dans le processus de corrélation d'alertes. La corrélation d'alertes ne doit pas non plus se cantonner aux informations fournies par les outils de détection d'intrusions. $\mathbf{M_2D^2}$ modélise d'autres outils chargés d'assurer la sécurité du système d'information.

4.4.1 Description des outils de sécurité

Les outils de sécurité sont au centre de $\mathbf{M_2D^2}$: en surveillant les entités du système d'information (*i.e.* la cartographie), ils détectent des vulnérabilités latentes ou exploitées, émettent ou consomment des événements.

Les outils de sécurité correspondent à ce que Valdes *et al* [78, 65], mais aussi Qin et Lee [69], appellent les périphériques INFOSEC, c'est-à-dire tout outil susceptible de générer de l'information relative à la sécurité du système d'information.

Ces périphériques comprennent, mais ne se limitent pas, aux sondes de détection d'intrusions, aux scanners de vulnérabilité et aux pare-feux.

La liste des outils modélisés est amenée à être complétée si des informations pertinentes pour la corrélation d'alertes sont fournies par d'autres outils.

On divise les sondes de détection d'intrusions en deux catégories, en fonction de la localisation des événements utilisés par l'analyseur : les NIDS (événements réseau) et les HIDS (événements système ou applicatifs).

La *visibilité* d'une sonde concerne son aptitude à détecter une attaque. Connaître la visibilité d'une sonde permet de gérer des conflits entre sondes. Il y a conflit lorsqu'une sonde génère une alerte et qu'une autre sonde n'en génère pas alors que sa visibilité le lui permet. Nous déclinons la visibilité en deux notions, la visibilité *topologique* et la visibilité *opérationnelle*.

La *visibilité topologique* se réfère à la capacité d'une sonde à détecter une attaque par sa localisation dans le système d'informations. La visibilité topologique d'une sonde système correspond à l'hôte qui est surveillé par la sonde. La visibilité topologique d'une sonde réseau correspond à l'ensemble des hôtes connectés sur les liens de la route suivie par une attaque (voir la section dédiée à la cartographie 4.2).

La *visibilité opérationnelle* se réfère à la capacité d'une sonde à détecter une attaque par sa configuration. La configuration d'une sonde est l'ensemble des signatures activées au niveau de la sonde. La configuration d'une sonde reflète la politique de sécurité, c'est pour cette raison que toutes les signatures ne sont pas systématiquement activée. Un administrateur peut aussi choisir de n'activer qu'une partie des signatures pour accroître les performances d'une sonde qui surveille un système fortement chargé. Enfin, la liste des signatures activables peut aussi être limitée par la nature des événements analysés par une sonde. Par exemple, une sonde applicative surveillant l'activité d'un serveur web ne détecte que les attaques contre serveurs web.

A l'opposé des sondes de détection d'intrusions qui détectent les exploitations effectives de vulnérabilités, les scanners de vulnérabilités détectent les vulnérabilités latentes. Les notions de visibilité opérationnelle et topologique s'appliquent aussi aux scanners car ces outils sont configurés pour tester l'existence d'une vulnérabilité donnée (similaire aux signatures de sondes), sur un ensemble d'hôtes.

Les pare-feux sont des outils de prévention d'intrusion. Leur rôle est d'empêcher des connexions indésirables dans un réseau. La définition des types de communications autorisées est fonction du protocole utilisé, de l'origine de la connexion, de sa destination. Les pare-feux sont très largement utilisés pour la sécurité des réseaux informatiques à l'heure actuelle.

4.4.2 Acquisition des informations sur les outils de sécurité

L'acquisition des informations relatives à la visibilité topologique des sondes de détection d'intrusions dépend de leur nature. La visibilité topologique d'une sonde système est donnée *de facto* par l'hôte sur lequel le capteur de la sonde est installé.

Concepts

$\mathcal{O}$	est l'ensemble des outils de sécurité,
$\mathcal{O}_D$	est l'ensemble des IDS,
$\mathcal{O}_{D_N}$	est l'ensemble des NIDS,
$\mathcal{O}_{D_H}$	est l'ensemble des HIDS,
$\mathcal{O}_S$	est l'ensemble des scanners de vulnérabilité,
$\mathcal{O}_F$	est l'ensemble des pare-feux,

Relations

tooltype $\in$	$\mathcal{O}$	$\rightarrow$	{FireWall, NetIDS, HostIDS,Scanner}
meth $\in$	$\mathcal{O}_D$	$\rightarrow$	{Misuse, Anomaly}
monitored_interface $\in$	$\mathcal{O}_{D_N}$	$\rightarrow$	$\mathcal{I}$
monitored_host $\in$	$\mathcal{O}_{D_H}$	$\rightarrow$	$\mathcal{H}$
scans $\in$	$\mathcal{O}_S$	$\leftrightarrow$	$\mathcal{H}$
canraise $\in$	$\mathcal{O}_D \cup \mathcal{O}_S$	$\leftrightarrow$	$\mathcal{S}$

Définitions

$\mathcal{O}_{D_N}$	$= \textbf{tooltype}^{-1}[\{\texttt{NetIDS}\}]$
$\mathcal{O}_{D_H}$	$= \textbf{tooltype}^{-1}[\{\texttt{HostIDS}\}]$
$\mathcal{O}_D$	$= \mathcal{O}_{D_H} \bigcup \mathcal{O}_{D_N}$
$\mathcal{O}_S$	$= \textbf{tooltype}^{-1}[\{\texttt{Scanner}\}]$
$\mathcal{O}_F$	$= \textbf{tooltype}^{-1}[\{\texttt{FireWall}\}]$

FIG. 4.9 – Modèle des outils de sécurité de $\mathbf{M_2D^2}$

La visibilité topologique d'une sonde réseau dépend de la présence sur le chemin suivi par le datagramme suspect, de l'interface utilisée par le capteur pour écouter le trafic. Elle est donc fonction de la topologie du réseau, de la source et de la destination d'un datagramme particulier.

L'acquisition des informations relatives à la visibilité opérationnelle des sondes morphoscientes et des scanners de vulnérabilité se fait en analysant les fichiers de signatures.

4.4.3 Formalisation

Le modèle formel des outils de sécurité dans $\mathbf{M_2D^2}$ est schématisé en figure 4.9.

L'ensemble $\mathcal{O}$ contient l'ensemble des outils de sécurité déployés dans le système d'informations. Les outils de sécurités sont qualifiés par leur type *via* la fonction **tooltype**. $\textbf{tooltype}^{-1}$ partitionne $\mathcal{O}$ en plusieurs parties $\mathcal{O}_{D_H}$, $\mathcal{O}_{D_N}$, $\mathcal{O}_S$ et $\mathcal{O}_F$ qui désignent respectivement les HIDS, les NIDS, les scanners de vulnérabilités et les pare-feux. $\mathcal{O}_D$ est l'ensemble des sondes de détection d'intrusions (toutes sources de données confondues). Comme il est envisageable qu'un système d'information ne soit équipé que de sondes réseau (ou système), les ensembles $\mathcal{O}_{D_N}$ et $\mathcal{O}_{D_H}$ peuvent être vides.

La fonction **meth** indique la méthode d'analyse utilisée par une sonde pour détecter les intrusions.

Les sondes réseau surveillent l'activité du système d'information au travers d'une interface réseau. La visibilité topologique d'une sonde réseau est donc modélisée à partir de la fonction totale **monitored_interface** qui lie le capteur de la sonde à l'interface d'écoute. La fonction est totale parce qu'il n'y a pas lieu d'inclure une sonde qui n'écoute pas le réseau.

La visibilité toplogique des sondes système ou applicatives se traduit par la fonction totale **monitored_host** qui lie la sonde à l'hôte qu'elle surveille.

La visibilité topologique des scanners de vulnérabilités est modélisée par la relation **scans**. Un scanner surveille périodiquement plusieurs hôtes du système, explicitement désignés par l'administrateur. Ceci explique que **scans** soit une relation et non une fonction.

La visibilité opérationnelle d'une sonde est modélisée dans $\mathbf{M_2D^2}$ par la relation **canraise**. **canraise**(x, y) signifie que la signature y est activée au niveau de la sonde x. Ainsi, à condition que sa visibilité topologique le lui permette, la sonde x est capable de reconnaître des attaques détectées par la signature y. Notons que le domaine de la relation **canraise** est constitué de l'union des sondes morphoscientes et des scanners de vulnérabilité.

La composition des relations **canraise** et **refersto** fournit l'ensemble des vulnérabilités qu'une sonde ou un scanner de vulnérabilité est apte à détecter.

4.4.4 Limitations

La visibilité opérationnelle d'une sonde de détection d'intrusions comportementale n'est pas définie puisque par définition la visibilité opérationnelle est une relation entre une sonde et un identifiant de signature.

Savoir si la visibilité topologique d'une sonde réseau permet à cette sonde de capter une activité intrusive à partir d'un datagramme capté par une autre sonde nécessite de connaître la route suivie par le datagramme. Ceci est possible à condition que cette route soit statique. Cette contrainte est notable. On peut toutefois la modérer en notant que la topologie des réseaux d'entreprises est peu changeante.

4.5 Evénements

Les concepts de $\mathbf{M_2D^2}$décrits jusqu'ici modélisent les connaissances extérieures aux alertes, que nous jugeons nécessaires de prendre en compte pour corréler les alertes. Nous présentons maintenant les événements, qui contiennent non seulement les alertes produites par les outils de sécurité, mais aussi les événements consommés par les capteurs et qui donnent lieu à des alertes.

4.5.1 Description des événements

IDMEF [83] définit les événement et les alerte par rapport aux entités qui les produisent et qui les consomment : un événement est une occurrence dans le flot de sortie d'un capteur, à destination d'un analyseur et une alerte est une notification formatée

de l'occurrence d'une activité intrusive, générée par un analyseur, à destination d'un *manager*.

Dans $\mathbf{M_2D^2}$, le concept d'alerte, ainsi que l'ensemble des types d'événements manipulés par les composants d'un système de détection d'intrusions héritent du concept générique événement. Ainsi, une alerte comme un paquet IP sont des types particuliers d'événements. Par la suite, nous qualifions de *bruts* (i.e. n'ayant pas encore été analysés) les événements issus des capteurs si la distinction avec les événements en général est nécessaire.

Du point de vue de la hiérarchie des composants de IDMEF, une implémentation de $\mathbf{M_2D^2}$ est un *manager*. Une implémentation est donc alimentée par des analyseurs. On peut donc se demander pour quelle raison des événements bruts sont modélisés dans $\mathbf{M_2D^2}$. En fait, nous avons besoin de modéliser les événements bruts car ce sont eux qui contiennent les informations qui désignent la victime et l'attaquant d'une attaque. Contrairement à d'autres schémas de description d'alertes comme IDMEF par exemple, dans $\mathbf{M_2D^2}$, les alertes ne contiennent pas de référence directe aux attaquants et aux victimes ; ces notions sont indirectement accessibles par les propriétés des événements qui sont à l'origine de l'alerte. Par exemple, dans IDMEF, une alerte comprend un champ attaquant dont une modalité possible est une adresse IP ; dans $\mathbf{M_2D^2}$, une alerte est liée à un autre événement de type IP, dont un des champs est une adresse. Pour autant, seuls les événements bruts liés à des alertes sont contenus dans $\mathbf{M_2D^2}$. Les événements bruts n'ont d'existence que par les alertes qui qui y font référence.

Evénements Un événement est produit par un composant appartenant aux outils de sécurité. Selon IDMEF, fonctionnellement parlant, la notion de producteur dépend du type de composant considéré : un capteur produit des événements bruts et les analyseurs produisent des alertes. Les capteurs devraient donc faire partie de l'ensemble des outils de sécurité. Toutefois, comme seuls les événements bruts liés à des alertes sont contenus dans $\mathbf{M_2D^2}$ et que, structurellement parlant, les analyseurs et les capteurs sont généralement couplés (pour former une sonde), l'identité du producteur d'un événement brut correspond à l'identité de l'analyseur qui le consomme pour produire une alerte.

Un événement est daté. Dans les spécifications de IDMEF [83], deux types de dates sont définies pour les événements :

1. le *Detect Time* représente la date d'occurence supposée d'une attaque,
2. le *Create Time* est la date de détection de l'attaque,

Ces deux types de dates sont donc implicitement modélisés dans $\mathbf{M_2D^2}$ car *in fine*, le type de date d'un événement dépend du type de l'événement : l'horodatage d'un événement brut (*e.g.* un datagramme) est affecté par un capteur et correspond à un *Detect Time* ; l'horodatage d'une alerte est défini par un analyseur et correspond à un *Create Time*.

Plusieurs relations entre les événements sont modélisées ; elles permettent en particulier de structurer les événements dans un graphe, comme suggéré par l'IDMEF avec la classe *Correlation Alert*. Une *Correlation Alert* est une alerte particulière qui contient des références à d'autres alertes. Cette définition est récursive, une *Correlation*

Alert peut elle-même être incluse dans une autre *Correlation Alert*, constituant ainsi un graphe d'alertes. Ces relations sont détaillées dans la section formalisation.

Alertes Du point de vue du manager qui les consomme pour former des alertes de plus haut niveau, les alertes sont comparables aux événements bruts consommés par les analyseurs pour former des alertes. Les alertes partagent tous les attributs du concept événement. Pour ces deux raisons, les alertes sont modélisées dans $\mathbf{M_2D^2}$ comme un type particulier d'événement.

Les alertes sont composées d'un identifiant d'attaque et sont générées par une sonde de détection d'intrusion ou par un scanner de vulnérabilité. Elle sont liées le cas échéant aux événements (pouvant être d'autres alertes) qui ont provoqué leur émission.

Dans d'autres formats, IDMEF en particulier, les alertes possèdent un attribut source et cible. Source et cible prennent des formes différentes en fonction de la source de données analysée. Dans $\mathbf{M_2D^2}$, ils n'apparaissent pas explicitement dans les alertes, mais sont présents dans les événements produits par les capteurs et liés à l'alerte. Par exemple, la source d'une attaque détectée par un IDS réseau se manifeste sous la forme de l'adresse source contenue dans l'entête d'un datagramme. La cible d'une attaque peut être le lieu de sa détection. Par exemple, les alertes émises par les IDS systèmes ou applicatifs concernent des attaques dont la cible est l'hôte surveillé.

Scans Les scans sont des messages issus de scanners de vulnérabilité, informant de la présence d'une vulnérabilité sur un système. Les scans sont similaires aux alertes : ils relient un hôte à un identifiant de vulnérabilité propre au scanner de vulnérabilité (faisant éventuellement référence à une vulnérabilité connue).

Messages de pare-feux Les messages générés par les pare-feux indiquent l'action entreprise par le pare-feu vis-à-vis d'une connexion réseau. Deux actions sont envisageables : soit la connexion est autorisée, soit elle est interdite. Les connexions sont des datagrammes IP.

Evénements IP, TCP, UDP, HTTP Les événements bruts actuellement modélisé dans $\mathbf{M_2D^2}$ sont des événements réseau car les sondes utilisées en milieu opérationnel analysent essentiellement des événements réseau. D'autres types d'événement bruts peuvent bien sûr venir compléter ceux déjà présents dans $\mathbf{M_2D^2}$. Ils sont ajoutés par héritage du concept événement.

Ces événements sont constitués d'un sous-ensemble des informations contenues dans les entêtes du protocole réseau idoine. Ainsi, un datagramme IP est constitué des adresses de l'interface émettrice et de l'interface destinatrice, ainsi que du numéro d'identification. Rien n'empêche de compléter $\mathbf{M_2D^2}$ avec d'autres attributs, mais ils ne servent *a priori* pas pour la corrélation, alors que les adresses IP permettent d'identifier les attaquants et les cibles. Le numéro d'identification peut être utilisé comme identifiant unique de datagramme (dans une fenêtre temporelle de quelques heures). Ainsi, si des sondes distinctes font référence à un datagramme donné dans une alerte,

le numéro d'identification permet de s'assurer qu'il s'agit du même datagramme. Un datagramme IP encapsule d'autres protocoles comme TCP ou UDP.

Les segments TCP ou UDP sont qualifiés par les numéros de ports utilisés au cours d'une transaction. Les événements TCP contiennent en outre les numéros d'ordre et d'accusé de réception afin d'identifier les segments appartenant à une même session. A leur tour, les protocoles TCP et UDP encapsulent les protocoles applicatifs comme HTTP.

Les événements HTTP sont des parties d'une transaction puisqu'ils sont contenus dans les données d'un datagramme. Ainsi, un événement HTTP contient de manière exclusive une url et une méthode (requête du client) ou un statut (réponse du serveur).

Nous proposons aussi un événement correspondant à une ligne d'audit HTTP générée par un serveur Web. Dans ce dernier cas, toutes les informations sur la transaction HTTP (méthode, url et statut) sont disponibles dans la ligne d'audit. Un événement d'audit HTTP est donc lié par une relation spécifique à l'ensemble des événements de type HTTP qui composent la transaction.

4.5.2 Acquisition des événements

La notion de producteur fait partie intégrante des événements : les alertes sont produits par des sondes de détection d'intrusions, les scans sont produits par des scanners de vulnérabilité et les messages sont produits par des pare-feux.

4.5.3 Formalisation

Les événements sont formalisés en figure 4.10 et 4.11.

Les événements sont les éléments de $\mathcal{E}$. Les types d'événements sont des parties de $\mathcal{E}$:

- $\mathcal{E}_{\mathsf{A}}$ est l'ensemble des alertes,
- $\mathcal{E}_{\mathsf{IP}}$ est l'ensemble des datagrammes,
- $\mathcal{E}_{\mathsf{TCP}}$ est l'ensemble des segments TCP,
- $\mathcal{E}_{\mathsf{UDP}}$ est l'ensemble des segments UDP,
- $\mathcal{E}_{\mathsf{HTTP}}$ est l'ensemble des requêtes/réponses HTTP,
- $\mathcal{E}_{\mathsf{LOG}}$ est l'ensemble des lignes d'audit HTTP,
- $\mathcal{E}_{\mathsf{F}}$ est l'ensemble des messages envoyés par un firewall.

L'héritage entre les différents types d'événements et le concept événement est modélisée par cette relation d'inclusion.

Le domaine des relations communes à tous les types d'événements est $\mathcal{E}$. Ces relations sont l'estampillage temporel, le composant producteur, la causalité et l'agrégation. La date d'occurrence d'un événement est donné par la fonction totale **tstamp**. La fonction totale **generator** associe à un événement son producteur.

La relation **causes** modélise la relation qui lie des événements à des alertes : la détection d'un datagramme suspect provoque (i.e. *est la cause de*) l'émission d'une alerte. Une événement peut être la cause de plusieurs alertes soit parce qu'il participe à des hypothèses concurrentes de corrélation, soit parce qu'il présente plusieurs propriétés

Concepts

$\mathcal{E}$	est l'ensemble des événements,
$\mathcal{E}_{\mathsf{A}}$	est l'ensemble des alertes,
$\mathcal{E}_{\mathsf{S}}$	est l'ensemble des scans,
$\mathcal{E}_{\mathsf{IP}}$	est l'ensemble des datagrammes IP,
$\mathcal{E}_{\mathsf{TCP}}$	est l'ensemble des segments TCP,
$\mathcal{E}_{\mathsf{UDP}}$	est l'ensemble de segments UDP,
$\mathcal{E}_{\mathsf{HTTP}}$	est l'ensemble des portions de requêtes HTTP,
$\mathcal{E}_{\mathsf{F}}$	est l'ensemble des messages issus de pare-feux,
$\mathcal{E}_{\mathsf{LOG}}$	est l'ensemble des lignes d'audit de serveurs web.

Relations

tstamp $\in$	$\mathcal{E}$	$\rightarrow$	$\mathbb{N}$
partof $\in$	$\mathcal{E}$	$\leftrightarrow$	$\mathcal{E}$
generator $\in$	$\mathcal{E}$	$\rightarrow$	$\mathcal{O}$
signature $\in$	$\mathcal{E}_{\mathsf{A}} \cup \mathcal{E}_{\mathsf{S}}$	$\rightarrow$	$\mathcal{S}$
scan_host_target $\in$	$\mathcal{E}_{\mathsf{S}}$	$\rightarrow$	$\mathcal{H}$
scan_port_target $\in$	$\mathcal{E}_{\mathsf{S}}$	$\rightarrow$	$\mathbb{N}$
causes $\in$	$\mathcal{E}$	$\leftrightarrow$	$\mathcal{E}_{\mathsf{A}}$

Propriétés

$\mathcal{E}_{\mathsf{A}}, \mathcal{E}_{\mathsf{S}}, \mathcal{E}_{\mathsf{IP}}, \mathcal{E}_{\mathsf{TCP}}, \mathcal{E}_{\mathsf{UDP}}, \mathcal{E}_{\mathsf{HTTP}}, \mathcal{E}_{\mathsf{LOG}}$ et $\mathcal{E}_{\mathsf{F}}$ forment une partition de $\mathcal{E}$
generator$[\mathcal{E}_{\mathsf{A}}] \subseteq \mathcal{O}_D$
generator$[\mathcal{E}_{\mathsf{S}}] \subseteq \mathcal{O}_S$
generator$[\mathcal{E}_{\mathsf{F}}] \subseteq \mathcal{O}_F$

Commentaires

Les alertes peuvent être causées par d'autres alertes, par conséquent **dom**(**causes**) $\subset \mathcal{E}$ et non $\mathcal{E} - \mathcal{E}_{\mathsf{A}}$
ran(**causes**) $\subsetneq \mathcal{E}_{\mathsf{A}}$ parce que les alertes peuvent ne pas être liées à un événement causal.

FIG. 4.10 – Modèle formel d'événements de $\mathbf{M_2D^2}$

suspecte, chacune donnant lieu à une alerte. Pour illustrer ceci, considérons l'attaque suivante :

```
http://webserveur/cgi-bin/phf?Qalias=x%0a/bin/cat%20/etc/passwd
```

Certains IDS génèrent trois alertes pour cette requête : une relative à la présence de `/cgi-bin/phf` (réputé pour être vulnérable), une relative à la présence de `/etc/passwd` dans l'URL et une relative à la réponse donnée par le serveur web (contenu du fichier `/etc/passwd`).

A l'inverse, une alerte peut être liée à plusieurs événements. Pour ces raisons, **causes** est une relation. Enfin, une alerte peut n'être liée à aucun événement. Ceci ne signifie pas que l'alerte n'a aucune cause, mais que les événements à l'origine de l'alerte ne sont pas fournis par la sonde.

Concepts

$\mathcal{U}$ est l'ensemble des URL jugées suspectes par les outils de détection d'intrusions.

Relations

ip_src, **ip_dst**, **idt** $\in$	$\mathcal{E}_{\mathsf{IP}}$	$\rightarrow$	$\mathbb{N}_A$
ipayload $\in$	$\mathcal{E}_{\mathsf{IP}}$	$\nrightarrow$	$\mathcal{E}_{\mathsf{TCP}} \cup \mathcal{E}_{\mathsf{UDP}}$
sp, **dp** $\in$	$\mathcal{E}_{\mathsf{TCP}} \cup \mathcal{E}_{\mathsf{UDP}}$	$\rightarrow$	$\mathbb{N}$
seq, **ack** $\in$	$\mathcal{E}_{\mathsf{TCP}}$	$\rightarrow$	$\mathbb{N}$
tpayload $\in$	$\mathcal{E}_{\mathsf{TCP}} \cup \mathcal{E}_{\mathsf{UDP}}$	$\rightarrowtail$	$\mathcal{E}_{\mathsf{HTTP}}$
httpmeth $\in$	$\mathcal{E}_{\mathsf{HTTP}}$	$\rightarrow$	{`GET`,`POST`,`HEAD`}
status $\in$	$\mathcal{E}_{\mathsf{HTTP}}$	$\rightarrow$	$\mathbb{N}$
url $\in$	$\mathcal{E}_{\mathsf{HTTP}}$	$\rightarrow$	$\mathcal{U}$
firemesstype $\in$	$\mathcal{E}_{\mathsf{F}}$	$\rightarrow$	{`Block`, `Accept`}
firemess $\in$	$\mathcal{E}_{\mathsf{F}}$	$\rightarrow$	$\mathcal{E}_{\mathsf{IP}}$

FIG. 4.11 – $\mathbf{M_2D^2}$ Propriétés des événements IP, TCP, UDP et HTTP

Comme le domaine de **causes** est $\mathcal{E}$, un ensemble d'alertes peut être la cause d'autres alertes. Cette propriété permet de définir la structure arborescente ($\mathcal{E}_{\mathsf{A}}$, **causes**) entre les alertes, comme suggéré par la classe `Correlation Alert` de IDMEF. Dans cette arborescence, les alertes de haut niveau générées par les outils de corrélation sont liées aux alertes de plus bas niveau, générées par les sondes, par la relation **causes**. Cette structure arborescente utilisée dans le chapitre 6.

La relation **partof** permet de modéliser l'agrégation d'événements dans d'autres événements. Pour l'instant, cette relation n'est utilisée que pour encapsuler des événements HTTP dans un événement de type audit HTTP.

Alert L'identifiant d'attaque contenu dans une alerte est obtenu *via* la fonction totale **signature**. Le générateur d'une alerte est une sonde de détection d'intrusions.

Scan Comme les alertes, les scans font référence à un identifiant d'attaque via la fonction totale **signature** et sont générés par un scanner, fourni par **generator**.

Les scans portent explicitement une référence à l'hôte impliqué dans l'analyse de vulnérabilité, ainsi que le service (numéro de port). Le premier est donné par la fonction totale **scan_host_target**, le second par la fonction totale **scan_port_target**. Contrairement aux alertes, les scans ne sont pas causés par un événement car les scanners détectent des vulnérabilités latentes. Par conséquent, l'identité des cibles n'est pas contenue dans les événements sous forme d'addresses IP, par exemple. C'est pour cette raison que l'identité des cibles doit être explicitement spécifiée dans l'événement scan.

Messages de pare-feux Un pare-feu autorise ou interdit une connexion de manière exclusive. L'action effectuée par le pare-feu est modélisée par la fonction totale

firemesstype. La connexion concernée est un événement IP identifié par la fonction totale **firemess**. Tout message d'un pare-feu fait référence à un et un seul événement IP.

Événements bruts Les fonctions partielles **ipayload** et **tpayload** désignent respectivement les données encapsulées par des événements IP et TCP ou UDP. Ce sont des fonctions partielles parce que les sondes ne sont pas toujours capable d'extraire les contenus de toutes les couches protocolaires d'un datagramme. La plupart des sondes n'analysent que la couche transport (e.g. TCP), pas les protocoles applicatifs (e.g. HTTP).

Les autres relations identifient des attributs contenus dans les entêtes des protocoles correspondants. Conformément aux spécifications de ces protocoles, toutes ces relations sont des fonctions totales : elles sont définies pour tous les éléments du domaine de définition et chacun de ces éléments a au plus une image.

ip_src, **ip_dst**, **idt** désignent respectivement l'adresse source, l'adresse destination et le numéro d'identification d'un datagramme IP. **sp**, **dp** désignent respectivement le numéro de port source et destination d'un événement TCP ou UDP. **seq**, **ack** désignent respectivement le numéro de séquence et d'acquittement d'une session TCP. **httpmeth** est la méthode d'une requête HTTP, **status** son statut et **url** l'URL.

4.5.4 Limitations

Dans la version actuelle de $\mathbf{M_2D^2}$, seuls les événements bruts les plus utilisés par les sondes sont modélisés. Pour pouvoir utiliser des sondes système, par exemple, il est nécessaire d'ajouter les événements bruts correspondants, par héritage du concept événement.

4.6 Conclusion et perspectives

Dans ce chapitre, nous avons présenté un modèle qui fédère les informations que nous jugeons nécessaires à la corrélation d'alertes. Ces informations portent sur la cartographie du système d'information surveillé, sur les attaques, sur les outils de sécurité et enfin sur les événements et les alertes.

Au titre des travaux futurs, $\mathbf{M_2D^2}$ devra prendre en compte le caractère de plus en plus dynamique de la cartographie. La cartographie doit aussi être complétée par d'autres informations, notamment par la politique de sécurité en place dans un système d'informations. En effet, la pertinence des alertes dépend en partie de la politique de sécurité en place dans le système d'informations surveillé. Par exemple, certaines signatures d'IDS morphoscients permettent de détecter l'utilisation de protocoles non sécurisés (comme telnet par exemple), qui engendrent un volume important de faux positifs dans des environnements où de tels protocoles sont autorisés.

La modélisation de la politique de sécurité du site dans $\mathbf{M_2D^2}$ permettrait en outre de vérifier que celle-ci est bien complète et cohérente vis-à-vis de la configuration des sondes, c'est-à-dire que toute violation de la politique de sécurité donne lieu à une alerte et que les signatures n'émettent une alerte que pour une violation de politique

de sécurité. En effet, en théorie, c'est la politique de sécurité qui définit les signatures appliquées aux sondes et se traduit par la relation **canraise** de $\mathbf{M_2D^2}$. Dans la pratique, le niveau de granularité des informations contenues dans les signatures n'est pas en adéquation avec le niveau d'abstraction des règles d'une politique de sécurité ; c'est la configuration des sondes qui définit implicitement la politique de sécurité. Le développement d'outils qui détectent des violations de politiques de sécurité à proprement parler constituent un sujet de recherche à part entière. On pourra se référer aux travaux de Zimmerman *et al* pour un exemple d'un tel outil [84].

Chapitre 5

Analyse conceptuelle d'alertes

Nous avons vu dans le chapitre 3 que les approches de corrélation implicites consistent *in fine* à regrouper les alertes pour pallier principalement à l'excès d'alertes. Ce type d'approche doit faciliter le travail de l'opérateur en limitant la quantité d'informations qui lui est soumise pour analyse.

Les composants de gestion d'alertes existants comme ACID[1] ou *Dragon Enterprise Manager*[2] affichent les alertes issues des sondes dans une interface web qui est connectée à un système de gestion de base de données relationnelle (SGBDR) où les alertes sont stockées. L'opérateur de sécurité soumet des requêtes au SGBDR *via* l'interface, qui en retour fournit un ensemble d'alertes qui satisfont les contraintes exprimées dans la requête. Ces contraintes portent sur les propriétés des alertes (*e.g.* adresse IP de la victime, de l'attaquant, type de l'attaque, etc.) ; en SQL, les requêtes se présentent sous la forme

$$\texttt{SELECT * FROM alerts WHERE } att_1 \texttt{ = } val_1 \texttt{ AND } att_2 = ...$$

où les att_i sont des noms d'attributs sur lesquels porte la sélection et val_i la valeur recherchée.

Le nombre d'alertes retournées peut être grand, si bien que l'opérateur est amené à raffiner sa requête pour ne sélectionner qu'un sous-ensemble d'alertes qui l'intéressent.

Comme le fait remarquer Ferré [28], la modification d'une requête n'est pas simple car cela suppose d'inférer une modification *intensionnelle*, c'est-à-dire portant sur la formule logique de la requête, à partir d'une réponse *extensionnelle*, c'est-à-dire portant sur l'ensemble d'alertes retourné ; il n'y a pas de relation simple entre une variation dans la requête et la variation correspondante de la réponse. En résumé, la modification de la requête requiert de la part de l'opérateur de la chance et/ou de l'intuition pour identifier la modification de requête appropriée. Cette modification de requête nécessite en outre de la part de l'opérateur une connaissance du langage de requête, en l'occurrence SQL.

Le système ne devrait donc pas seulement fournir des alertes en réponse aux requêtes, mais aussi proposer à l'opérateur des *liens de navigation* (*i.e.* des critères de modification) pour raffiner ses requêtes et limiter ainsi le volume d'alertes retournées.

[1] `http://www.andrew.cmu.edu/~rdanyliw/snort/snortacid.html`

[2] `http://dragon.enterasys.com`

Les réponses aux requêtes contiennent ainsi non seulement de éléments extensionnels (i.e. des alertes), mais aussi des éléments intensionnels (i.e. des éléments de descriptions d'alertes). Le rôle de l'opérateur ne consiste alors plus à inférer une nouvelle requête, mais à choisir le lien de navigation le plus adapté. Comme il est plus simple de reconnaître ce que l'on cherche que de le décrire, le travail de l'opérateur s'en trouve facilité [28].

Comme l'indique Ferré [28] dans sa thèse, interrogation et navigation constituent les deux approches de la recherche d'informations, qui englobe la représentation, l'organisation, le stockage et l'accès à l'information.

L'interrogation consiste à élaborer une requête au système d'information, qui en retour fournit un ensemble d'objets satisfaisant la requête, susceptibles d'être reconnus par l'utilisateur. Les requêtes SQL sont de ce type. D'une part, le nombre d'objets retournés peut être grand et il est difficile pour l'utilisateur de s'y retrouver ; d'autre part, ce type d'approche requiert de la part de l'utilisateur une connaissance *a priori* de l'organisation des informations (*e.g.* le schéma relationnel), du langage de requête et du contenu.

La navigation consiste à rechercher l'information en se déplaçant dans un graphe au sein duquel les objets sont structurés, un peu à la manière de la navigation dans des documents hypertextuels. L'inconvénient de ces structures est qu'elles sont figées dès la conception ; elles sont donc difficiles à faire évoluer. Notons que les liens hypertextuels qui composent les interfaces web des systèmes de gestion d'alertes ne sont pas des liens de navigation au sens de la recherche d'informations. Ce sont des raccourcis vers des requêtes SQL prédéfinies qui évitent à l'opérateur d'avoir à élaborer la requête.

Nous abordons ici le problème de la corrélation implicite comme un problème de recherche d'information et nous souhaitons combiner la recherche et la navigation, afin de fournir à l'opérateur un outil flexible de manipulation d'alertes. Nous justifions ce choix en section 5.1.

Comme le montrent Missaoui *et al* dans [70], l'analyse de concepts constitue une approche de la recherche d'information qui permet de combiner la navigation et l'interrogation. A partir d'un contexte donné, constitué d'objets et de leur description, l'analyse de concepts construit automatiquement le treillis de Galois (appelé treillis de concepts) des concepts du contexte. Chaque nœud du treillis est un concept, c'est-à-dire un ensemble d'objets partageant des propriétés communes ; les arcs dénotent des relations de spécialisation/généralisation entre les nœuds.

L'analyse de concepts est brièvement résumée en section 5.3. Avant cela, nous précisons d'abord les besoins liés à une approche de corrélation implicite. Nous détaillons ensuite en quoi l'analyse de concepts permet de répondre à ces besoins. La section 5.4 est dédiée à l'application de l'analyse de concepts aux alertes. En section 5.5, nous décrivons une implémentation possible, puis nous donnons des résultats expérimentaux.

5.1 Besoins de la corrélation

Les regroupements d'alertes effectuées par les approches de corrélation implicites existantes sont globalement tous basés sur une mesure de ressemblance entre alertes, qui est une combinaison des mesures de ressemblance entre les propriétés des alertes. Comme ces attributs sont qualitatifs, les mesures de similarité font appel à des connaissances expertes permettant de juger du degré de ressemblance de deux attributs. Par exemple, la ressemblance de deux adresses IP dépend de leur appartenance à un même sous-réseau.

Les approches proposées par Cuppens [14], Valdes et Skinner [77], Dain et Cunningham [18, 19] ou bien Debar et Wespi [23] sont de cette nature. On pourra se référer à la section 3.2.3 pour plus de détails sur ces techniques.

Nous pensons que ce type d'approche est tout à fait pertinent, mais aucune des techniques existantes ne présente l'ensemble des propriétés que nous souhaitons, à savoir : une description intrinsèque des groupes d'alertes, des groupes d'alertes non mutuellement exclusifs et enfin une formation incrémentale de la structure qui contient les groupes d'alertes. Dans la suite de cette section, nous justifions la nécessité de ces propriétés.

5.1.1 Description des groupes

Nous pensons que l'inconvénient majeur des approches de corrélation implicite réside dans l'absence de description des groupes d'alertes. Plus précisément, dans les approches existantes, les groupes d'alertes sont munis de propriétés qui sont l'union des propriétés des alertes appartenant au groupe.

Si la quantité d'informations soumise à l'opérateur est bien réduite, il n'en reste pas moins que l'opérateur est souvent ramené à consulter le contenu des groupes car l'absence de description conduit à des regroupements qui peuvent sembler arbitraires. Notons au passage que la consultation du contenu des groupes (assimilable à la navigation) n'est pas toujours permise car certaines approches fusionnent les alertes, si bien que seules les méta-alertes obtenues par fusion sont disponibles ; le lien contenant/contenu entre méta-alerte et les alertes disparaît.

Prenons un exemple pour illustrer nos propos. L'analyse des alertes générées par les sondes déployées dans le réseau de Supélec a révélé un grand nombre d'attaques de type web, provenant de l'intérieur du réseau, contre le serveur mandataire (*proxy* web) de l'école. Sans l'aide de la corrélation, ce phénomène se traduit par une liste plane d'alertes provenant des sondes qui surveillent intérieur du réseau, comme celle présentée en figure 5.1 (les attributs sont respectivement un identifiant de signature Snort, l'adresse de l'attaquant, l'adresse de la victime et la date).

Une première étape de corrélation, le regroupement, consiste à mettre en évidence l'appartenance des alertes à un même phénomène. Le groupe résultant possède une description du type de celle présentée en Figure 5.2. Les attributs du groupe sont l'union des valeurs d'attributs des alertes. C'est le cas, par exemple de l'approche de Valdes et Skinner [77].

La seconde étape consiste à fournir à l'opérateur une description synthétique des

```
...
1606    192.168.4.22    192.168.5.10    05/10/2003 15:05:16
943     192.168.3.15    192.168.5.10    05/10/2003 15:24:58
1485    192.168.2.14    192.168.5.10    05/10/2003 15:44:33
1606    192.168.5.23    192.168.5.10    05/10/2003 16:05:27
1602    192.168.8.122   192.168.5.10    05/10/2003 17:15:15
1485    192.168.0.232   192.168.5.10    05/10/2003 17:56:11
1606    192.168.2.14    192.168.5.10    05/10/2003 18:32:53
1537    192.168.4.22    192.168.5.10    05/10/2003 18:14:44
...
```

FIG. 5.1 – Liste plane d'alertes

```
[1606, 943, 1485, 1602, 1485, 1537]
[192.168.4.22, 192.168.3.15, 192.168.2.14,
      192.168.5.23, 192.168.8.122, 192.168.0.232]
[192.168.5.10]
[05/10/2003-05/10/2003 18:14:44]
```

FIG. 5.2 – Groupe d'alertes

groupes. Cette description synthétique devrait précisément ressembler à la phrase que nous avons utilisé ci-dessus pour décrire informellement le phénomène, en utilisant des termes plus ou moins abstraits : les "attaques web" constituent une abstraction des divers identifiants d'attaques, le "réseau interne" est une abstraction de la liste des adresses IP des attaquants, le "proxy web" est une abstraction de l'adresse IP de la victime. Quant à l'information temporelle, elle n'est pas évoquée, ce qui sous-entend que le phénomène en question est permanent. On peut considérer que *any* [time] est l'abstraction des dates d'attaques.

Le phénomène que nous avons décrit correspond en fait à un ensemble de fausses alertes. Un serveur mandataire joue le rôle de relais pour les requêtes `http` destinées à l'extérieur, effectuées par les utilisateurs du réseau. Les requêtes que le serveur mandataire reçoit sont donc bien destinées à l'extérieur, mais du point de vue d'un IDS surveillant le trafic interne, ces requêtes sont destinées à un hôte interne, le serveur mandataire. Les sondes sont généralement configurées pour ne détecter que les attaques menées contre les hôtes du réseau. Comme dans le cas présent les sondes croient que les requêtes sont adressées à un hôte interne, une alerte est générée si les requêtes sont jugées malicieuses. Comme beaucoup de signatures détectent des requêtes à des applications web réputées malicieuses et non des attaques réelles contre ces applications, un grand nombre de faux positifs est ainsi généré.

Pour un opérateur de sécurité, le caractère bénin d'une alerte décrite par un message du type "attaques web de l'intérieur contre proxy" est évident. En revanche, si le phénomène se présente sous la forme d'une liste d'alertes telle que celle présentée en figure 5.1, le diagnostic nécessite plus d'investigations de la part de l'opérateur.

L'exemple précédent montre que nous avons besoin d'un langage de description des alertes. Ce langage doit permettre de renseigner l'opérateur sur l'attaque, l'attaquant, la victime et la date de l'attaque. En effet, nous avons vu en section 2.1, page 17, que ces quatre informations sont nécessaires à l'appréhension d'une alerte par un opérateur.

Comme le suggère Julisch dans [43, 45], les éléments des domaines des quatre attributs sont structurés dans des taxinomies afin de pouvoir décrire les groupes d'alertes avec des termes plus ou moins abstraits. Les termes abstraits permettent non seulement de synthétiser l'information, mais aussi d'ajouter de l'information, ce qui contribue à l'amélioration sémantique des alertes. Par exemple, le fait de désigner une victime par sa fonction dans le réseau ou par son nom plutôt que par son adresse IP est souvent plus évocateur pour un opérateur de sécurité.

Les taxinomies utilisées dans l'approche de Julisch sont des arbres, c'est-à-dire qu'un attribut ne peut être abstrait qu'en une seule valeur. Ceci limite l'expressivité des relations entre les valeurs des attributs. Par exemple, dans le cas de l'abstraction des adresses IP en fonction d'hôte, un hôte ne peut avoir plusieurs fonctions. Une autre abstraction consiste à généraliser des adresses IP en identifiants de réseaux (*e.g.* DMZ). La localisation d'un hôte dans un réseau est indépendante de sa fonction : rien n'empêche un serveur web d'être dans un réseau interne d'entreprise, en dehors de la DMZ. Encore une fois, on voit que la structure taxinomique de Julisch ne permet pas d'exprimer ce type de relation. Notons que dans une publication récente [44], il semble que Julisch propose des graphes acycliques dirigés comme taxinomies d'attributs.

5.1.2 Groupes d'alertes non mutuellement exclusifs

Une autre propriété souhaitable de l'approche de corrélation implicite réside dans son aptitude à constituer des groupes d'alertes qui ne soient pas mutuellement exclusifs. En assimilant les groupes d'alertes à des hypothèses de corrélation, il est tout à fait envisageable qu'une alerte appartienne à plusieurs hypothèses de corrélations concurrentes, donc à plusieurs groupes.

Par exemple, si nous imaginons une organisation des alertes dans une structure hiérarchique triviale dans laquelle les arcs correspondent aux attributs des alertes, il n'est pas possible de savoir *a priori* si il faut classer les alertes d'abord par victime, puis par attaquant ou bien l'inverse. Cela dépend des besoins de l'opérateur de sécurité à un instant donné.

De nombreuses techniques de regroupement de données proposent de structurer des groupes d'individus dans une hiérarchie. C'est le cas des approches de classification non supervisée hiérarchiques, dont on peut trouver une liste dans [8]. Dans une telle structure, les intersections entre les groupes sont nulles. Dans l'approche de Julisch en particulier, les alertes sont structurées dans une hiérarchie.

5.1.3 Constitution incrémentale des groupes

Les alertes issues des sondes de détection d'intrusions sont produites de manière continue. Les groupes d'alertes sont donc par nature dynamiques : des alertes sont ajoutées au groupes existants, modifiées et supprimées à la demande de l'opérateur.

Les modifications requises par l'opérateur portent sur des groupes. Par exemple, pour reprendre l'exemple du serveur mandataire (p. 87), l'opérateur peut demander à ce que l'ensemble des alertes correspondant à la description soient supprimées.

La technique de regroupement doit prendre en compte l'aspect dynamique des alertes. Pour autant que nous sachions, l'approche de Julisch s'applique à un journal d'alertes préalablement enregistré. En revanche, dans les approches de Cuppens [14] ou bien Valdes et Skinner [78], les alertes sont traitées en ligne.

5.2 Notre approche

La classification conceptuelle permet de traiter les données qualitatives, c'est-à-dire dépourvues de fonctions de distance ou de similarité. Plusieurs approches ont été proposées dans la littérature [59]. L'algorithme CLUSTER/2 proposé par Michalski et Stepp construit une hiérarchie conceptuelle à partir d'un ensemble de données. Les classes obtenues, dont le nombre doit être fourni en paramètre, sont disjointes. De plus, l'approche n'est pas incrémentale. Cette approche ne nous convient donc pas. Fisher propose un algorithme, COBWEB [31], capable de construire les groupes de manière incrémentale. Des extensions (*e.g.* LABYRINTH) prennent en compte des données munies de taxinomies. Malheureusement, les classes obtenues sont aussi disjointes. Dans [33], Godin *et al* proposent une approche de classification conceptuelle fondée sur les treillis de Galois qui remplit l'ensemble de nos conditions.

Étant donné un contexte formé d'un ensemble d'objets, d'un ensemble d'attributs et d'une relation binaire qui lie les objets à leurs attributs, l'analyse de concept construit un treillis de concepts, qui est un treillis de Galois. Dans l'analyse de concepts *formelle*, chaque concept est une paire composée d'une *extension*, représentant un sous-ensemble d'objets, et d'une *intension*, représentant les propriétés communes à ces objets. Dans notre approche, les objets sont des alertes.

C'est la structure de treillis qui permet de combiner la navigation et l'interrogation. Un nœud du treillis représente un emplacement, obtenu par interrogation. Les liens connectés à ce nœud sont les liens de navigation pertinents, relatifs à l'emplacement considéré.

L'utilisation d'une relation binaire entre les objets et leurs attributs dans l'analyse de concepts formelle est trop restrictive en terme d'expressivité. Notre contexte présente par exemple des attributs valués (e.g. des adresses IP), dont le domaine forme une taxinomie.

Ferré propose de généraliser l'analyse de concepts *formelle* en analyse de concepts *logique* (ACL). Cette généralisation permet d'utiliser les formules d'une logique presque quelconque pour décrire les objets à la place de la relation binaire. En fait, la logique utilisée pour décrire les objets du contexte est un paramètre de l'ACL. En section 5.4.2, nous définissons la logique utilisée pour décrire les alertes, et qui nous permet d'appliquer l'ACL aux alertes. Ainsi, chaque sous-ensemble d'alertes (extension d'un concept) est décrit par une formule de la logique utilisée (l'intension correspondante), qui intègre des termes de niveaux d'abstraction variés du vocabulaire utilisé dans le langage. Notons que les taxinomies associées à nos attributs sont des graphes acycliques dirigés

dont le pouvoir expressif est meilleur que les arbres. Ces taxinomies sont décrites en détail en section 5.4.1. Le langage descriptif et les taxinomies répondent au problème évoqué dans la section 5.1.1.

La structure de treillis permet d'avoir des groupes d'alertes qui ne sont pas mutuellement exclusifs. Le treillis obtenu par analyse de concepts met en évidence de façon exhaustive les regroupements potentiellement intéressants par rapport aux observations. Ceci apporte une solution au problème évoqué en section 5.1.2.

Enfin, plusieurs auteurs proposent des techniques de construction incrémentale du treillis de concepts. De plus, contrairement à d'autres approches de classification conceptuelle, la structure du treillis est indépendante de l'ordre d'occurrence des observations. Ceci apporte une solution au problème évoqué dans la section 5.1.3.

5.3 Contexte théorique : l'analyse de concepts

Cette section sert à décrire le contexte théorique dans lequel se situe notre étude. Nous décrivons l'analyse de concepts *formelle* (ACF), puis nous évoquons l'analyse de concepts *logique* (ACL), proposée par Ferré et Ridoux [30, 29], qui constitue une généralisation de l'analyse de concepts formelle.

5.3.1 Analyse de concepts formelle

En analyse de concepts, un contexte met en correspondance des objets avec des propriétés. Un contexte est un triplet $K = (\mathcal{O}, \mathcal{A}, I)$, où

- $\mathcal{O}$ est un ensemble d'objets,
- $\mathcal{A}$ est un ensemble d'attributs,
- $I \subseteq \mathcal{O} \times \mathcal{A}$ est une relation binaire établissant quels attributs sont associés à quels objets.

Un concept[3] est composé d'une *extension*, c'est-à-dire un ensemble d'objets et d'une *intension*, c'est-à-dire un ensemble de propriétés. La cohérence réciproque entre les objets et leurs propriétés est garantie : l'extension contient tous les objets vérifiant l'intension et pas plus ; l'intension contient toutes les propriétés communes aux objets de l'extension et pas plus.

Formellement, un *concept* d'un contexte K est une paire $c = (O, A)$ où $O \subseteq \mathcal{O}$ et $A \subseteq \mathcal{A}$ telle que $\sigma_K(O) = A$ et $\tau_K(A) = O$ où les fonctions σ_K et τ_K sont définies de la manière suivante :

$$\begin{aligned} \sigma_K : \quad & 2^{\mathcal{O}} \to 2^{\mathcal{A}} \\ & \sigma_K(O) := \{a \in \mathcal{A} : \forall o \in O, (o, a) \in I\} \end{aligned}$$

$$\begin{aligned} \tau_K : \quad & 2^{\mathcal{A}} \to 2^{\mathcal{O}} \\ & \tau_K(A) := \{o \in \mathcal{O} : \forall a \in A, (o, a) \in I\} \end{aligned}$$

[3] La notion de concept utilisée ici n'est pas celle utilisée pour désigner les objets dans $\mathbf{M_2D}^2$

O est l'*extension* de c et A est l'*intension* de c.

L'ensemble $C(K)$ des concepts d'un contexte K est ordonné par la relation d'ordre partiel suivante :

$$(O_1, A_1) \leq (O_2, A_2) \Longleftrightarrow O_1 \subseteq O_2 \Longleftrightarrow A_1 \subseteq A_2$$

Les concepts sont donc des ensembles maximaux d'objets partageant les mêmes attributs. On prouve que l'ensemble ordonné des concepts $(C_K, \leq)$ forme un treillis complet, le treillis de concepts.

Un treillis de concepts est une *connexion de Galois* (cf annexe D) , qui met en correspondance des *ensembles* d'objets et des *ensembles* d'attributs. Le treillis de concepts est généré automatiquement par l'analyse de concepts formelle, à partir d'un contexte [82, 32].

Les liens de navigation sont les arcs du diagramme de Hasse (cf annexe D) du treillis de concepts. A partir d'un concept, on peut se déplacer vers un sous-concept, c'est-à-dire un concept dont l'intension est plus grande (la description contient plus de propriétés, elle est plus donc plus précise), mais dont l'extension est plus petite (moins d'objets possèdent la description plus précise). En posant une requête à partir d'un concept, on se déplace vers le concept dont l'extension est l'intersection de l'extension du concept d'origine et de l'ensemble des objets vérifiant la requête.

A titre d'exemple, la Figure 5.3 représente le treillis de concepts du contexte $K = (\{1,2,3,4,5,6\}, \{a,b,c,d,e,f,g,h,i\}, I)$, où I est défini par la matrice suivante :

	a	b	c	d	e	f	g	h	i
1	x		x			x		x	
2	x		x				x		x
3	x			x			x		x
4		x	x			x		x	
5		x			x		x		

Un concept peut ainsi être vu comme un *endroit* par son extension et comme une *requête* par son intension.

L'organisation d'une relation binaire d'un contexte sous la forme d'un treillis de concepts fait apparaître des groupements d'objets privilégiés, qui facilite l'appréhension des données.

L'ensemble des concepts extraits d'un contexte dans le cadre de l'ACF offre une structure de navigation compatible avec l'interrogation, grâce à la dualité extension/intension des concepts.

5.3.2 Analyse de concepts logique

Les domaines d'application variés de l'analyse de concepts (dont on peut trouver une liste dans [28]) nécessitent des contextes plus sophistiqués que la simple présence/absence d'attributs. En particulier, des attributs *valués* s'avèrent souvent nécessaires.

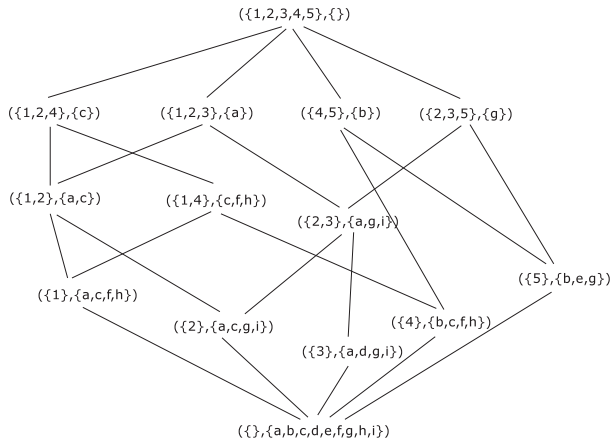

FIG. 5.3 – Exemple de treillis de concept

Ferré et Ridoux proposent une généralisation logique de l'analyse de concept formelle. L'analyse de concepts logique constitue une généralisation de l'analyse de concepts formelle car la description des objets n'est plus restreinte à une conjonction de propriétés ; les objets peuvent être décrits par une formule d'une logique presque quelconque. La logique utilisée pour *décrire* les objets constitue un paramètre de l'ACL. La logique a comme rôle, en plus de représenter les connaissances factuelles, d'intégrer et de formaliser les connaissances logiques.

Dans l'ACF, la relation I d'un contexte associe un ensemble d'attributs à chaque objet. Dans l'ACL, cette relation est remplacée par une fonction qui à un objet associe une formule de la logique utilisée pour décrire les objets.

Les propriétés des fonctions σ_K et τ_K de l'analyse de concepts formels sont maintenues dans l'analyse de concepts logique, pourvu qu'on établisse les correspondances suivantes entre les attributs et les formules :

Ensembles d'attributs	$\rightarrow$	formules
$G \subseteq F$	$\rightarrow$	$F \models G$
$F \bigcap G$	$\rightarrow$	$F \vee G$
$F \bigcup G$	$\rightarrow$	$F \wedge G$

L'inclusion ensembliste est remplacée par la conséquence logique : si un ensemble d'attributs G est inclu dans F, cela signifie que G est moins restrictif que F pour décrire les objets, c'est-à-dire en termes de logique que G est une conséquence logique de F. L'intersection ensembliste est remplacée par la disjonction : lorsqu'on enlève des éléments d'un ensemble de propriétés, la description devient plus large, ce qui revient à faire un *ou* logique entre des formules logiques. L'union ensembliste est remplacée par la conjonction logique (lorsqu'on ajoute des éléments à d'un ensemble de propriétés, la description devient plus précise, ce qui revient à faire un *et* entre des formules logiques).

Nous ne détaillons pas plus l'analyse de concepts logique dans ce document. Le lecteur pourra se réferer à la thèse de Ferré pour plus de détails [28].

5.4 Application de l'ACL aux alertes

Nous proposons d'appliquer l'analyse de concepts logique aux alertes produites par les sondes de détection d'intrusions. Les alertes sont les objets de l'analyse de concepts. Dans l'analyse de concepts, la description des objets, les requêtes et les réponses du système sont formulées dans une logique, qui est un paramètre de l'ACL (voir section 5.3.2). Pour appliquer l'ACL aux alertes, il nous faut donc d'abord définir une telle logique pour les alertes. Nous appelons cette logique la logique des alertes, notée $\mathcal{L}_A$.

La logique des alertes est une logique propositionnelle multivaluée, c'est-à-dire une logique du type objet-attribut-valeur.

En effet, indépendamment de leur statut de faux ou vrai positif, les alertes sont des faits, issus d'observations effectuées par des capteurs. On peut donc décrire une alerte par une conjonction de littéraux positifs.

Les littéraux utilisés dans les descriptions d'alertes renseignent des propriétés intrinsèques des alertes. Nous avons vu en section 2.1 qu'il existe quatre catégories d'attributs fondamentaux : l'attaque, l'attaquant, la victime et la date. Le domaine de chacun de ces attributs est composé de plusieurs valeurs ; c'est pour cette raison que la logique utilisée est dite multi-valuée.

Les alertes doivent aussi pouvoir être qualifiées par des propriétés extrinsèques, affectées par l'opérateur de sécurité.

Si la conjonction de littéraux positifs suffit à décrire les alertes, le langage de requêtes utilisé par l'opérateur pour interroger le système nécessite des constructions syntaxiques plus élaborées, comme la disjonction et la négation, pour savoir quelles alertes présentent tels *ou* tels attributs, ou bien ne présentent pas tels autres.

En section 5.4.2, nous formalisons ces propriétés en donnant la syntaxe et la sémantique des formules de la logique des alertes. Avant cela, nous définissons le vocabulaire des formules de $\mathcal{L}_A$, c'est-à-dire les domaines de définition des attributs des alertes ainsi que leur structure taxinomique.

5.4.1 Attributs des alertes

Nous donnons ici la description informelle de la structure taxinomique utilisée pour chaque attribut. Les valeurs décrites vont de la plus spécifique à la plus abstraite et sont issues des informations de $\mathbf{M_2D}^2$.

Comme nous allons le voir, les valeurs utilisées imposent aux taxinomies d'être des graphe acycliques dirigés. Une valeur peut être abstraite en plusieurs valeurs. Comme nous l'avons évoqué en introduction de ce chapitre, les structures taxinomiques utilisées dans l'approche de Julisch [43, 45] sont des arbres équilibrés dont le pouvoir expressif est inférieur aux graphes.

a) Attaques

Le moyen élémentaire pour désigner l'action effectuée par un attaquant consiste à utiliser les identifiants de signature des sondes morphoscientes. La documentation

associée aux signatures permet aux opérateurs de comprendre la raison du caractère malicieux de l'action détectée.

Les vulnérabilités constituent une abstraction des identifiants de signatures. Les vulnérabilités sont un moyen de désigner les attaques indépendant des modèles de sondes, ce qui permet de faire coopérer des sondes hétérogènes. Même au sein d'une sonde, une vulnérabilité peut être associée à plusieurs identifiants de signatures, auquel cas les différentes signatures dénotent différents moyens d'exploiter la vulnérabilité. L'utilisation des vulnérabilités permet alors de mettre en évidence des intrusions où un attaquant tente d'exploiter une vulnérabilité donnée selon plusieurs méthodes. Enfin, les vulnérabilités sont associées aux types de composants vulnérables, indispensables à l'évaluation de la sévérité des attaques. L'inconvénient des vulnérabilités est que toutes les signatures ne font pas référence à une vulnérabilité (voir page 70, chapitre **??**).

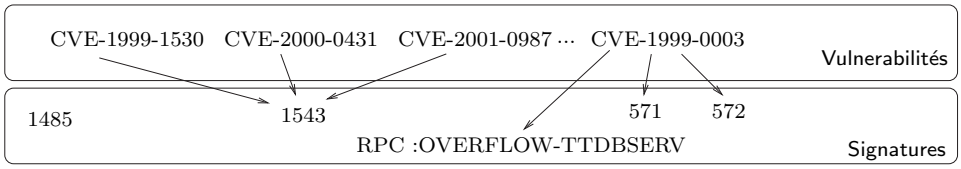

FIG. 5.4 – Portion de taxinomie des attaques

Une portion de taxinomie schématisant les relations taxinomiques entre les identifiants de signatures et les noms de vulnérabilité est donnée en figure 5.4. Dans cet exemple, on voit que la signature 1485 de Snort n'est associée à aucune vulnérabilité, alors que la signature 1543 est en relation avec trois vulnérabilités distinctes. Enfin, la vulnérabilité `CVE-1999-0003` est associée à deux signatures Snort et une signature Dragon.

Nous utilisons aussi l'ensemble des qualificatifs d'attaques issus de la classification des mots-clé de la documentation des signatures Snort dans une taxinomie, comme décrit à la page 68 du chapitre **??**. L'utilisation de cette taxinomie a trois avantages. Premièrement elle permet de décrire les groupes d'alertes en utilisant des termes de niveau d'abstraction variés et évocateurs pour l'opérateur. Deuxièmement, même si ils sont issus de la documentation Snort, ces termes peuvent être associés à des identifiants d'attaques d'autres modèles de sondes. Troisièmement, certains termes de la taxinomie peuvent être utilisés par des sondes comportementales pour décrire les attaques, comme par exemple `overflow`.

b) Victimes

Les informations disponibles sur les victimes ne concernent que les entités internes au système d'informations. Cette contrainte est raisonnable du fait que l'opérateur ne s'intéresse *a priori* qu'aux attaques contre le système d'informations qu'il gère.

Notons toutefois pour certaines catégories d'attaques, la distinction entre victime

et attaquant est ambigüe. Dans les attaques de type ver comme Nimda[4] ou Blaster[5], par exemple, l'attaquant - c'est à dire l'hôte d'où provient l'action malicieuse - est avant tout une victime d'une attaque préalable. Dans ce type d'attaque, l'hôte interne à l'origine d'une attaque contre un hôte externe est tout de même qualifié de victime. Les attaques par dénis de service distribué constituent un autre exemple. Ces attaques sont détectées par des communications entre un maître et un esclave. Dans ce cas, le maître et l'esclave interviennent en qualité d'attaquant dans l'alerte.

Les victimes d'attaques sont identifiées par leur adresse IP au niveau d'abstraction le plus bas de la taxinomie.

Les victimes d'attaques peuvent aussi être désignées par un nom d'hôte. Les hôtes peuvent posséder plusieurs adresses IP parce qu'il sont connectés à plusieurs réseaux ou bien à cause d'un mécanisme de translation d'adresses (NAT). Par exemple, lorsque deux sondes de détection d'intrusions sont déployées de part et d'autre d'un routeur effectuant de la translation d'adresse, un même hôte peut être référencé avec des adresses IP distinctes dans les alertes. Dans une telle configuration, les alertes devraient être groupées puisqu'elles font référence à la même occurrence d'attaque.

Les hôtes du réseau peuvent être abstraits en un identifiant de fonction qu'ils occupent dans le système d'informations. Un identifiant de fonction est lié au type de serveur applicatif hébergé par un hôte, comme par exemple serveur web, serveur ftp, etc. Un hôte peut héberger plusieurs serveurs applicatifs.

Les hôtes peuvent aussi être abstraits en identifiants de réseaux locaux. Plusieurs études montrent en effet une recrudescence d'attaques par balayage [22]. Une attaque par balayage consiste à lancer une attaque donnée sur chaque hôte d'un même réseau. Il est alors pertinent de désigner la victime des attaques par un identifiant de réseau, plutôt que par la liste des hôtes. Notons que comme un même hôte peut appartenir à plusieurs réseaux, un hôte peut être abstrait en plusieurs identifiants de réseaux locaux.

L'abstraction d'un hôte en identifiant réseau se fait uniquement si l'adresse IP qui est à l'origine de l'identifiant d'hôte existe réellement. En fait, certaines adresses IP impliquées dans des attaques par balayage ne sont associées à aucun hôte existant dans le réseau. Pour prendre ce type de situation en compte, les adresses IP peuvent être abstraite en plage d'adresse IP. Par exemple, l'adresse IP `193.54.192.45` est abstraite en `193.54.192.0`, qui désigne un réseau de classe C.

c) Attaquants

Au niveau d'abstraction le plus bas de la taxinomie, les attaquants sont identifiés par l'adresse IP de l'hôte depuis lequel l'attaque est menée.

Lorsque l'adresse IP d'un attaquant est externe, c'est-à-dire qu'elle n'appartient pas aux plages d'adresses du réseau surveillé, peu d'informations sont disponibles pour décrire l'attaquant. Une solution triviale pour généraliser les adresses IP consiste à ne prendre en compte que les 24 bits de poids fort de l'adresse IP (ce qui revient à prendre l'adresse du réseau de classe C correspondant à l'adresse IP). C'est par exemple la

[4] `http://www.cert.org/advisories/CA-2001-26.html`
[5] `http://www.cert.org/advisories/CA-2003-20.html`

solution adoptée par Julisch [43, 45], ainsi que par Dain et Cunnigham [18, 19] pour définir la fonction de similarité entre les alertes.

Toutefois, nous estimons que cette généralisation n'est pas satisfaisante. La première étape de généralisation des adresses doit permettre de regrouper les adresses qui appartiennent à une même autorité. Les attaquants initient leurs attaques depuis le réseau de leur entreprise, université, fournisseur d'accès à internet, au sein desquels leur adresse peut changer au cours du temps. La troncature des adresses IP sous la forme de l'adresse d'un réseau de classe C ne permet pas de prendre en compte ces autorités. En effet, depuis que l'adressage des réseaux se fait en notation CIDR[6] pour pallier aux manque d'adresses IP, plusieurs autorités distinctes peuvent se voir attribuer des portions de réseaux de classe C. Par conséquent, la troncature d'adresse peut regrouper de manière erronée des attaques provenant d'autorités différentes. A l'inverse, des autorités peuvent posséder plusieurs réseaux de classe C. Bien que provenant de la même autorité, la troncature ne peut pas regrouper deux attaques dont les adresses d'attaquant appartiennent à deux réseaux distincts.

Les identifiants d'autorités fournis par le IANA (voir chapitre 4, page 60) n'ont pas ces inconvénients. De plus, les identifiants d'autorités sont généralement plus évocateurs (ils apportent plus d'information) qu'une adresse de réseau, vis-à-vis d'un opérateur humain.

Notons que, toutefois, dans le cas des adresses IP de fournisseurs d'accès à internet, l'identifiant d'autorité peut amener à effectuer des regroupements d'adresses IP erronés. En effet, chaque utilisateur d'un réseau de fournisseur d'accès est assimilable à une autorité individuelle. Malheureusement, à notre connaissance, il n'existe pas de niveau de granularité intermédiaire entre les adresses IP et les identifiants du IANA disponibles publiquement.

Une portion de la taxinomie des attaquants externes est schématisée en Figure 5.5.

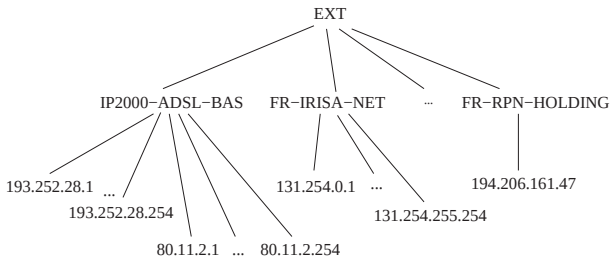

FIG. 5.5 – Portion de taxinomie des attaquants (extérieurs)

Lorsque les adresses IP des attaquants sont internes, les informations disponibles pour les qualifier sont potentiellement plus riches, puisque les hôtes sont sous la responsabilité de l'administrateur du système d'information. Ainsi, comme pour les victimes, les adresses des attaquants peuvent être abstraites en un nom d'hôte et/ou par une fonction d'hôte.

[6] la notation CIDR (*Classless Inter Domain Routing*) propose une identification réseau orientée bit. Il n'y a plus la limitation liée aux classes.

L'utilisation de fonctions d'hôtes en tant qu'identifiant d'attaquant peut permettre d'identifier des motifs de faux positifs courants ou évidents. Reprenons l'exemple des serveurs mandataires (*proxies*) http. L'étude des alertes générées par les sondes de détection d'intrusions révèle que les serveurs mandataires sont souvent impliqués en tant qu'attaquants dans des alertes de type "balayage IP sur port 80". Ce type d'attaque consiste à initier des connexions sur le port 80 d'un ensemble d'hôtes afin de trouver un serveur web potentiellement vulnérable. Ce type d'attaque se caractérise donc par une série de connexions initiées par un même hôte, sur le port 80 de plusieurs hôtes. Le comportement normal d'un serveur mandataire est donc tout à fait similaire à celui d'un attaquant effectuant un balayage IP sur le port 80. En effet, les serveurs mandataires relaient les requêtes des utilisateurs ; ils sont donc amenés à effectuer plusieurs requêtes simultanées vers des serveurs différents, si plusieurs utilisateurs utilisent le serveur mandataire en même temps. Une version abstraite de l'ensemble de ces alertes peut donc être "[balayage IP sur port 80] par les [proxy web] contre [serveurs web]". Les termes entre crochets sont respectivement l'attaque, l'attaquant et la victime.

Comme pour les victimes, les hôtes internes du réseau agissant en qualité d'attaquant peuvent être abstraits en domaines du réseau (DMZ, Réseau Élèves, etc.).

Les domaines du réseau interne et les domaines IANA pour les adresses externes sont respectivement généralisés en *interne* et *externe*.

d) Date

Comme le propose Julisch [43], la taxinomie associée à l'information temporelle portée par les alertes est constituée des différents champs qui composent une date. Nous ne prenons en compte que l'heure, le jour de la semaine et le jour du mois, qui nous semblent les plus intéressants.

Ainsi, une l'estampillage temporel des alertes, dont la résolution est de l'ordre de la milliseconde, est d'abord abstrait en heure de la journée, jour dans la semaine et numéro de jour dans le mois.

Les heures de la journée sont abstraits en `workhour` ou `outhour`, qui dénotent respectivement les heures de travail ou les heures en dehors de la période de travail. De la même façon, les jours de la semaines sont abstraits en `week-end` ou `week-day`.

Ce type de taxinomie permet d'isoler les alertes qui ne surviennent que pendant des périodes particulières (e.g. le *week-end*). Les relations d'ordre total liées au temps (i.e. avant, après, en même temps) n'ont pas lieu d'être dans la description factuelle des alertes. On pourra se réferer au chapitre 6 traitant des chroniques, ou ces relations temporelles sont utilisées de manière explicite.

5.4.2 Définition d'une logique des alertes

Nous formalisons ici la logique des alertes en donnant la syntaxe des formules de $\mathcal{L}_A$, puis leur sémantique. Ensuite, nous donnons les axiomes de la logique, définis à partir des relations taxinomiques évoquées dans la section 5.4.1. Enfin, nous définissons la notion de profils d'hôtes, basée sur les propriétés extrinsèques des alertes.

a) Syntaxe de la logique des alertes

Pour définir la syntaxe de la logique des alertes, on se donne un ensemble d'attributs $\mathcal{A} = \{\text{ATTACK}, \text{ATTACKER}, \text{VICTIM}, \text{DATE}\}$, des ensembles de valeurs $\mathcal{V}_a$ liés aux attributs et une terminologie $T = (Q, \nu)$ où Q est un ensemble de qualificatifs et ν associe à chacun des qualificatifs une formule de la logique des alertes $\mathcal{L}_A$ dont on donne la syntaxe :

$$\begin{array}{rcll} P, Q & \rightarrow & a \text{ is } v & a \in \mathcal{A}, v \in \mathcal{V}_a \\ & | & q & q \in \mathcal{Q} \\ & | & P \wedge Q & \\ & | & P \vee Q & \\ & | & \neg P & \end{array}$$

La première alternative permet de spécifier un attribut intrinsèque d'une alerte et sa valeur. Notons que pour une alerte donnée, un attribut peut prendre plusieurs valeurs et qu'un attribut peut aussi ne pas être défini. Par exemple, une alerte peut être décrite par la formule suivante :

$$\begin{array}{ll} & \text{ATTACKER is } 192.168.0.1 \\ \wedge & \text{VICTIM is } 192.168.0.2 \\ \wedge & \text{VICTIM is } 192.168.0.3 \\ \wedge & \text{ATTACK is } 1264 \\ \wedge & \text{DATE is } 25 \text{ june } 2003 \text{ } 15 : 14 : 26 \end{array}$$

Les domaines de définition des attributs sont issus des concepts de $\mathbf{M_2D}^2$. Le domaine de l'attribut ATTACK est constitué des identifiants de signatures $\mathcal{S}$, des termes descriptifs $\mathcal{D}$ et des vulnérabilités $\mathcal{V}_N$. Le domaine de l'attribut ATTACKER est constitué des adresses IP $\mathbb{N}_A$, des noms d'hôtes HN, des fonctions d'hôtes PT, des réseaux locaux DN, des identifiants IANA IANA et de l'ensemble $\{int, ext\}$. Le domaine de l'attribut VICTIM est constitué des adresses IP $\mathbb{N}_A$, des noms d'hôtes HN, des fonctions d'hôtes PT, des réseaux locaux DN, des plages d'adresses IP $\mathbb{N}_A$. Le domaine de l'attribut DATE est constitué des dates symbolisées par l'ensemble des entiers $\mathbb{N}$, des heures, des jours de la semaine, des numéros de jours dans un mois, des ensembles {weekend, weekday} et {workhour, outhour}.

La deuxième alternative permet de spécifier les caractéristiques extrinsèques des alertes permettant de qualifier les alertes. Pour l'instant, nous limitons l'ensemble $\mathcal{Q}$ à la valeur {`false-positive`}. Cette notion est détaillée dans le paragraphe d) de cette section.

Les trois dernières alternatives sont respectivement la conjonction, la disjonction et la négation de propriétés. La disjonction et la négation ne sont utilisées que pour interroger le système, pas pour décrire les alertes.

b) Sémantique de la logique des alertes

Les formules logiques ne sont pas interprétées par leur valeur de vérité, mais par des ensembles d'objets. Pour donner la sémantique de la logique des alertes, on définit

une interprétation comme une relation de $\mathcal{A}$ dans $\mathcal{V}$. La notation $\mu \models P$ signifie que l'interprétation μ satisfait la formule P :

$$\begin{array}{lll} \mu \models & a \text{ is } v & \iff v \in \mu[\{a\}] \\ \mu \models & q & \iff \mu \models \nu(q) \\ \mu \models & P \wedge Q & \iff \mu \models P \wedge \mu \models Q \\ \mu \models & P \vee Q & \iff \mu \models P \vee \mu \models Q \\ \mu \models & \neg P & \iff \mu \not\models P \end{array}$$

On définit la relation de déduction sur la logique des alertes par la formule suivante ($\mathcal{M}$ désigne le domaine des interprétations) :

$$P \models Q \iff \forall \mu \in \mathcal{M} : \mu \models P \Rightarrow \mu \models Q$$

En tant que sous-ensemble de la logique propositionnelle multivaluée, la logique des alertes $< \mathcal{L}_A, \models, \vee, \wedge >$ satisfait les conditions de l'ACL [28].

c) Axiomes de la logique et lien avec $\mathbf{M_2D^2}$

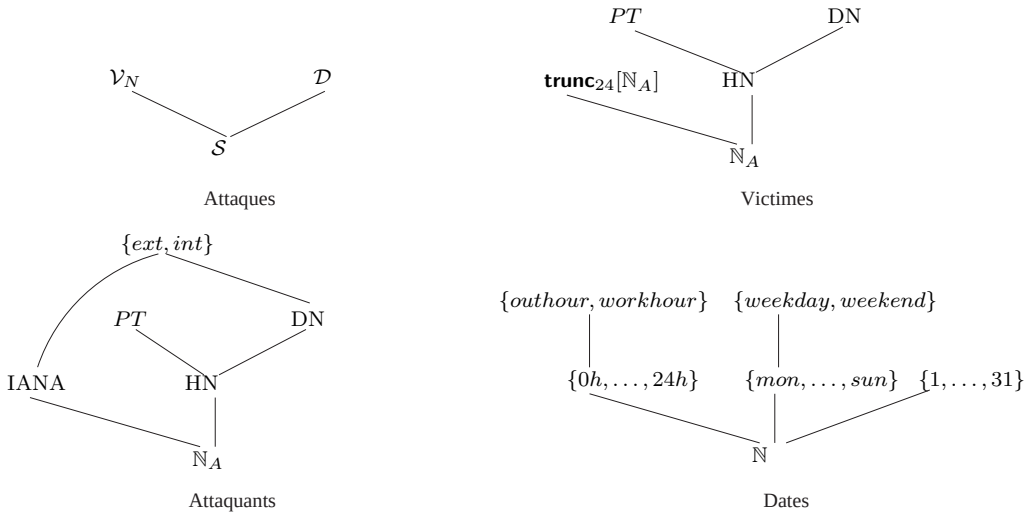

FIG. 5.6 – Taxinomies des attributs d'alertes

Les relations taxinomiques entre les valeurs des attributs, que nous avons décrites en section 5.4.1, sont résumées en Figure 5.6 avec les notations de $\mathbf{M_2D^2}$. Ces relations permettent de décrire les groupes d'alertes en utilisant en priorité les attributs les plus abstraits : si l'attribut a d'une alerte vaut v_1 et qu'il existe une relation taxinomique telle que v_0 est plus abstrait que v_1, alors la description de l'alerte soumise à l'opérateur sera prioritairement v_0.

Une relation taxinomique telle que "v_1 est plus spécifique que v_0" signifie que toute alerte qui est décrite par v_1 peut être décrite de manière plus abstraite par v_0. Les relations taxinomiques des attributs se transposent ainsi dans la logique des alertes

a	V	$S(v)$	
attack	$\mathcal{V}_N$	$\mathsf{refersto}^{-1}[v]$	1
	$\mathcal{D}$	$\{t \in \mathcal{D} \mid t \prec v\} \bigcup \{s \in \mathcal{S} \mid v \in \mathsf{descr}[s]\}$	2
victim	HN	$\mathsf{hostname}^{-1}; \mathsf{addr}[v] \cup \mathsf{hostname}^{-1}; \mathsf{addr}; \mathsf{nat}^{-1}[v]$	3
	$\mathsf{trunc}_{24}[\mathbb{N}_A]$	$\mathsf{trunc}_{24}{}^{-1}[v]$	4
	DN	$\mathsf{lanname}^{-1}; \mathsf{host}; \mathsf{hostname}[v]$	5
	PT	$\mathsf{prodtype}^{-1}; \mathsf{sub\_config}; \mathsf{configuration}^{-1}; \mathsf{hostname}[v]$	6
attacker	IANA	$\mathsf{iana}^{-1}[v]$	7
	HN	$\mathsf{hostname}^{-1}; \mathsf{addr}[v] \cup \mathsf{hostname}^{-1}; \mathsf{addr}; \mathsf{nat}^{-1}[v]$	8
	DN	$\mathsf{lanname}^{-1}; \mathsf{host}; \mathsf{hostname}[v]$	9
	PT	$\mathsf{prodtype}^{-1}; \mathsf{sub\_config}; \mathsf{configuration}^{-1}; \mathsf{hostname}[v]$	10
	$\{int\}$	DN	11
	$\{ext\}$	IANA	12
date	$\{0h, \ldots, 24h\}$	hour(v)	13
	$\{mon, \ldots, sun\}$	weekday(v)	14
	$\{1, \ldots, 31\}$	monthday(v)	15
	{weekend}	{sat,sun}	16
	{weekday}	{mon, ..., fri}	17
	{workhour}	{8h, ..., 19h}	18
	{outhour}	$\{19h, \ldots, 8h\}$	19

FIG. 5.7 – Définition des axiomes de la logique des alertes

sous la forme d'axiomes. La relation taxinomique précédente se transpose par la relation axiomatique $v_1 \models v_0$.

Cela signifie bien que dans le cas où la valeur v_1 d'un attribut a est plus spécifique que la valeur v_0 de ce même attribut, c'est-à-dire qu'on a la relation axiomatique

$$a \text{ is } v_1 \models a \text{ is } v_0$$

alors on a l'implication

$$d(o) \models a \text{ is } v_1 \Rightarrow d(o) \models a \text{ is } v_0$$

où $d(o)$ est la description d'une alerte o dans la logique des alertes.

Le schéma de définition des axiomes est le suivant :

$$\forall v \in V, \bigvee_{i \in S(v)} a \text{ is } i \models a \text{ is } v \qquad (a \in \mathcal{A}, V \subseteq \mathcal{V}_a)$$

Pour une valeur v d'un attribut a donnée, $S(v)$ représente l'ensemble des valeurs i du même attribut a qui sont directement plus spécifiques que v dans les taxinomies. La définition des axiomes de la logique se fait en instanciant les trois paramètres V, $S(v)$ et a. Le tableau 5.7 contient l'ensemble des instanciations. Pour que nous explicitions les relations, chaque ligne du tableau est indicée.

Dans la relation (1), $\mathbf{refersto}^{-1}[v]$ donne l'ensemble des signatures qui font référence à une vulnérabilité v. Par exemple, pour la vulnérabilité v = cve-1999-0039, les signatures Snort qui y font référence sont $S(v) = \mathbf{refersto}^{-1}$[cve-1999-0039] = { 1865 , 1147 , 1163 }.

Dans la relation (2), les qualificatifs d'attaques subsumés par un qualificatif v sont ceux qui sont immédiatement inférieurs à v dans la taxinomie, c'est-à-dire les valeurs $t \in \mathcal{D}$ telles que $t \prec v$ (la relation $\prec$ est définie en page 72). Il faut ajouter à cet ensemble les identifiants de signatures $s \in \mathcal{S}$ dont la description inclue v. Par exemple, dans la portion de taxinomie représentée en Figure 4.6 (page 69), les valeurs subsumées par `dnstools` sont {1741,1739,1740} ; les valeurs associées à `web` sont {`php`,`cgi`,`iis`,`apache`,`frontpage`}.

Les relations (3) et (8) associent à un nom d'hôte donné l'ensemble des adresses IP correspondant à cet hôte, c'est-à-dire celles de ses interfaces ($\mathbf{hostname}^{-1}; \mathbf{addr}[v]$) plus celles obtenues par translation d'adresse ($\mathbf{hostname}^{-1}; \mathbf{addr}; \mathbf{nat}^{-1}[v]$).

La fonction $\mathbf{trunc}_{24}[a]$ de la relation (4) donne les 24 bits de poids forts d'une adresse IP a. La valeur obtenue représente un plage d'adresses IP correspondant à un réseau de classe C.

Les relations (5) et (9) associent à un nom de réseau local l'ensemble des noms d'hôtes appartenant à ce réseau local : $\mathbf{lanname}^{-1}$ rend le réseau local associé au nom donné, c'est-à-dire un sous-ensemble d'interfaces ; la composition avec **host** retourne l'ensemble des hôtes de ce réseau ; la composition avec **hostname** rend les noms de ces hôtes.

Selon le même schéma que la relation (5), les relation (6) et (10) associent à une fonction d'hôte (*i.e.* un type de produit) l'ensemble des noms d'hôtes ayant cette fonction.

La relation (7) associe à un identifiant d'autorité la liste des IP qu'il possède.

La relation (11) dit que la valeur *int* qui symbolise l'intérieur du système d'informations, subsume l'ensemble des noms de réseaux locaux du système d'informations. Parallèlement, la relation (12) associe la valeur *ext* (extérieur du système d'information) aux identifiants d'autorités référencés par le IANA.

Les relations (13) à (19) sont des fonctions qui catégorisent les dates des alertes respectivement en heure de la journée, jour de la semaine, jour du mois, *week-end* ou jour de la semaine. Comme Julisch [43], les heures de la journée sont catégorisées en heures ouvrées ou non.

d) Qualificatifs d'alertes

Dans la section dédiée à la syntaxe de la logique des alertes (paragraphe a) de cette section), nous avons évoqué la notion de terminologie $T = (Q, \nu)$ où Q est un ensemble de qualificatifs d'alertes et ν une relation qui associe à chaque élément de Q une définition, c'est-à-dire une formule de la logique des alertes. Les éléments de Q sont les caractéristiques extrinsèques des alertes, définies par l'opérateur. Comme les définitions des qualificatifs sont des formules de la logique, elles impliquent des attaques, des attaquants, des victimes et éventuellement des périodes temporelles.

Pour l'instant, Q est réduit à un singleton, {`false-positive`}, mais il est possible d'ajouter d'autres qualificatifs, avec leur définition associée. Les définitions de {`false-positive`} sont des motifs d'alertes qui sont réputées pour être des faux positifs, engendrés par une configuration particulière du système d'informations, que nous appelons des *profils*.

Illustrons la notion de profils avec l'exemple des serveurs mandataires donné en section 5.1.1 de ce chapitre. Nous avons vu que du fait de leur fonction de relais de requêtes http, les serveurs mandataires web engendrent des fausses alertes indiquant l'occurrence d'attaques de type web provenant de l'intérieur du réseau contre le serveur mandataire, ce qui se traduit dans le langage des alertes par la formule

$$\begin{array}{ll} & \text{ATTACK is } \texttt{web} \\ \wedge & \text{ATTACK is } \texttt{access} \\ \wedge & \text{ATTACKER is } \texttt{intern} \\ \wedge & \text{VICTIM is } \texttt{http-proxy} \end{array}$$

Cette formule constitue une définition de `false-positive`. Ainsi, lorsqu'une attaque de type web survient, dont la victime est un hôte hébergeant un serveur mandataire web et dont l'attaquant est interne, alors l'alerte correspondante est implicitement qualifiée de faux-positif. On voit ici que le profil concerne une entité (mandataire web) agissant en qualité de victime. Voyons maintenant une définition de profil impliquant la même entité mais cette fois en qualité d'attaquant.

De part leur fonctionnement, les serveur mandataires de tous types sont amenés à initier un grand nombre de connexions par unité de temps vers des hôtes distincts, sur un port unique (voir page 26). Ce type d'activité est assimilable à des attaques par balayage IP et donnent lieu à des alertes. La définition suivante formalise ce type de faux positif :

$$\text{attack is } \texttt{IP scan} \wedge \text{attacker is } \texttt{proxy}$$

Lorsque des balayage IP initiés par des serveurs mandataires, les alertes correspondantes sont implicitement qualifiées de fausses alertes.

Ainsi, si l'opérateur interroge le système avec la requête $\neg$ `false-positive`, alors le système positionne l'opérateur dans le groupe des alertes qui ne sont pas réputées pour être des fausses alertes.

5.5 Stockage des alertes dans un système de fichiers logique

L'analyse de concepts logique instanciée avec une logique propositionnelle multivaluée constitue le fondement théorique de LISFS. LISFS est un système de fichiers logique, proposé par Padioleau et Ridoux [64]. LISFS offre la possibilité à n'importe quel type d'application de structurer des données (des fichiers) dans un treillis de concepts.

Dans un système de fichiers logique, les chemins des fichiers sont des formules de la logique utilisée pour décrire les fichiers. L'utilisateur du système de fichiers logique peut ainsi interroger le système de fichiers et naviguer.

Comme implémentation de l'analyse de concepts logique appliquée aux alertes, nous proposons de stocker les alertes issues des sondes de détection d'intrusions dans un système de fichiers logique.

Dans cette section, nous donnons un bref aperçu du système de fichiers logique, puis nous décrivons son utilisation pour stocker des alertes. Les résultats expérimentaux et des exemples d'interactions avec le système sont donnés dans la section suivante.

5.5.1 LISFS

Dans un système de fichiers hiérarchique conventionnel, les utilisateurs accèdent aux fichiers par leur chemin absolu. Les termes qui composent un chemin sont séparés par le caractère `/`. Un fichier ne peut se trouver à deux emplacements simultanément. Il est possible d'ajouter des liens symboliques, mais leur création et leur entretien est manuel.

Dans un système de fichiers logique, les répertoires jouent le rôle d'attributs des fichiers. Le caractère `/` dénote alors la conjonction logique. La conjonction logique est commutative, les répertoires `/a/b` et `/b/a` sont donc équivalents, ils contiennent les fichiers dont la description contient `A` et `B`.

La création d'un répertoire revient à ajouter une propriété dans le système. La commande correspondante est `mkdir`. Pour créer une relation taxinomique $v_1 \models v_0$, on utilise la commande `mkdir v0/v1`. De cette manière, lorsqu'un fichier est créé avec la propriété v_1, ce fichier possède implicitement la propriété v_0.

Comme le précisent Ferré et Ridoux [28], pour naviguer dans un système d'informations, un utilisateur doit disposer de trois commandes de base : consulter l'emplacement courante, consulter les liens de navigation disponibles depuis cet emplacement, modifier l'emplacement courant soit par sélection d'un lien, soit par formulation d'une requête.

Dans LISFS, l'emplacement courant est le répertoire de travail, désigné par `pwd`. La commande `cd` permet de changer le répertoire courant. Ainsi, `cd x` ajoute la propriété `x` à `pwd`, qui devient `pwd`$\wedge$`x`. `pwd` évolue incrémentalement en fonction des raffinements demandés par l'utilisateur. Le changement d'emplacement courant par formulation d'une requête peut se faire en spécifiant la description absolue d'un emplacement, à partir de la racine (*e.g.* `cd /nouvel/emplacement`).

La commande permettant de consulter un répertoire, c'est-à-dire d'obtenir les fichiers dont la description satisfait la requête, est la commande `ls`. La commande `ls` du système de fichiers logique de Padioleau et Ferré n'affiche pas seulement l'ensemble des fichiers satisfaisant une conjonction de propriété donnée, mais propose aussi un ensemble d'incréments *pertinents* qui permettent de raffiner la requête. La réponse à la commande `ls /propA/propB/` ne correspond donc pas à la requête SQL

```
SELECT * FROM objets WHERE att=propA AND att=propB
```

effectuée par les consoles de gestion d'alertes existantes, mais à un ensemble d'alertes *et* un ensemble d'incréments. L'ensemble d'alertes est constitué des alertes pour lesquelles

il n'existe pas de propriété permettant de les décrire plus précisément que la requête courante (`pwd`) ; l'ensemble des incréments est constitué des propriétés pertinentes qui permettent au contraire de raffiner la requête courante pour décrire les autres alertes dont la description satisfait `pwd`.

Les incréments correspondent aux liens de navigation. Sans cette fonctionnalité, la commande `ls` exécutée à la racine du système de fichiers (`pwd = /`) retournerait l'extension de la racine, c'est-à-dire l'ensemble des fichiers stockés.

Illustrons la pertinence des incréments par un exemple : si dans un emplacement `pwd` $= p$ tous les fichiers présentent la propriété a_1, quelques uns a_2 et aucun a_3, alors ni a_1 ni a_3 ne sont pertinents pour distinguer les fichiers, par contre a_2 l'est. La commande `ls` liste les sous répertoires, mais aussi les fichiers qui ne peuvent plus être distingués par aucune propriété pertinente. Dans un système de fichiers logique, les fichiers situés dans un emplacement `PWD` $= p$ sont les fichiers dont la description satisfait p, mais qui ne satisfait aucun de ses sous répertoires. Les sous-répertoires sont les incréments proposés par le système pour raffiner la recherche.

LISFS permet de prendre en compte les taxinomies d'attributs. Outre leur utilité pour désigner les propriétés des fichiers de manière abstraite, les taxinomies permettent aussi de réduire le nombre de réponses données à l'opérateur. En effet, bien que les liens de navigation réduisent le nombre de réponses à une requête, le nombre d'incréments proposés peut être important. Les incréments proposés par la commande `ls` sont donc les plus généraux, au regard des taxinomies d'attributs correspondants. Les relations taxinomiques sont définies en créant des sous-répertoire dans des répertoires.

Formellement, dans le répertoire courant `PWD` $= p$, l'ensemble des incréments (sous-répertoires) retournés par la commande `ls` est

$$\mathcal{D} = max_{\models}(\{a \in \mathcal{A} : \emptyset \subset ext(f \wedge p) \subset ext(p)\})$$

où

$$max_{\models} := \{a \in \mathcal{A} : \nexists a' \in \mathcal{A} \wedge a' \neq a \wedge a \models a'\}$$

($ext(p)$ désigne l'extension de p).

L'ensemble des fichiers retournés est :

$$\mathcal{F} = \{o \in \mathcal{O} : d(o) \models p \wedge \nexists f \in \mathcal{D} : d(o) \models f \wedge p\}$$

Dans LISFS, les attributs valués sont implémentés comme une taxinomie, où les valeurs d'un attribut sont des nœuds enfants d'un noeud représentant l'attribut lui-même.

La commande `mv` permet de modifier la description des fichiers.

5.5.2 Stockage d'alertes dans LISFS

Nous proposons de stocker les alertes produites par les sondes de détection d'intrusions dans un système de fichiers LISFS. De cette manière, l'opérateur peut effectuer des requêtes et naviguer dans les alertes de manière flexible. La manipulation d'ensembles d'alertes (modification des propriétés, suppression) est opérée à l'aide des commandes conventionnelles de manipulation de fichiers.

```
StoreAlarm(a, m_a)
let a = ⋀_{i∈{1..n}} d_i ;
for each d_i do
   CreateProperty(d_i) ;
done
cp  m_a  d_1/.../d_n/a
```

FIG. 5.8 – Algorithme de stockage d'une alerte a, de contenu m_a

```
CreateProperty(d)
if ¬exists(d) do
   D := {d_i | d ⊨ d_i} ;
   for each d_i ∈ D do
      CreateProperty(d_i) ;
   done
   mkdir d_1/.../d_n/d
done
```

FIG. 5.9 – Algorithme de création de l'arborescence de la propriété d

La procédure de stockage dans le système de fichiers d'une alerte est donnée par l'algorithme 5.8. Cette procédure prend en paramètre une alerte a dont la description est $\bigwedge_i d_i$, avec d_i de la forme a is v ($a \in \mathcal{A}$, $v \in \mathcal{V}$). L'insertion de a se fait en deux étapes :

1. L'ensemble des propriétés abstraites déduites des propriétés de l'alerte, qui n'ont pas déjà été ajoutées au cours du stockage d'autres alertes, sont ajoutées au système de fichiers. Cette étape est réalisée par l'algorithme CreateProperty. Les propriétés sont stockées sous la forme `attribut_valeur`.
2. Lorsque l'étape précédente est terminée, l'alerte est stockée dans le système de fichiers avec la commande `cp`. On considère que a désigne le nom unique donné à l'alerte. Nous utilisons le couple formé de l'identifiant de sonde et du numéro de série de l'alerte pour nommer les alertes. Cette étape est réalisée par l'algorithme StoreAlarm.

Dans l'état actuel de l'implémentation, les fichiers correspondant aux alertes contiennent le message d'alerte m_a tel que fourni par les sondes. Ce message est composé d'informations dont est issue la description $d(a)$ de l'alerte, plus d'autres informations (tels que des numéros de ports par exemple), ainsi que des traces laissées par l'attaque dans le système (paquets réseaux, audits système, lignes de log).

La procédure d'ajout de propriétés d'une alerte est donnée par l'algorithme 5.9. Cette procédure teste d'abord si la propriété fournie en paramètre existe déjà dans le système de fichiers, auquel cas la procédure se termine. Si elle n'existe pas, l'algorithme est appelé récursivement sur chaque propriété plus abstraite que celle fournie en

paramètre, puis la nouvelle propriété est ajoutée en tant que sous-répertoire des propriétés plus abstraites grâce à la commande `mkdir`. Par exemple, la création de la propriété `attack is 1602` donne d'abord lieu à la création de `attack is cve_2000_0208`, puis à la commande `mkdir attack_cve_2000_0208/attack_1602`.

5.6 Expérimentations

Dans cette section, nous présentons des expériences visant à évaluer les performances de LISFS pour le stockage d'alertes. Nous présentons d'abord l'environnement de test, puis nous donnons ensuite une évaluation des performances en mesurant le débit d'alertes toléré. Enfin, nous donnons quelques exemples de manipulation (interrogation/navigation) dans le système de corrélation.

a) Description de l'environnement

Pour tester l'approche, nous insérons les alertes issues des observations faites dans le réseau de Supélec. La description détaillée de l'environnement de test (configuration des sondes, topologie du réseau) est donnée en annexe B. Ce corpus d'alertes est constitué de 287088 alertes, produites par trois sondes Snort pendant 15 jours d'observation continue.

La machine utilisée pour l'expérimentation est dotée d'un processeur Pentium III à 700Mhz, de 128Mo de mémoire et d'un disque IDE à 7200 tours/min.

Afin de simplifier les premières expérimentations, nous avons choisi de limiter le nombre d'attributs des alertes. Pour cette raison, nous faisons abstraction de l'attribut temporel des alertes, qui ne nous intéresse pas prioritairement par rapport aux autres attributs. Les alertes sont donc exactement décrites par trois valeurs : l'identifiant de signature Snort, l'adresse IP attaquant et l'adresse IP de la victime. A ces valeurs fournies par les sondes, s'ajoutent les valeurs abstraites contenues dans les taxinomies associées aux attributs, que nous avons décrit en section 5.4.1. Toujours au titre des simplifications, seuls les identifiants de signature et les mots-clés de la documentation Snort sont utilisés pour décrire les attaques. Nous n'utilisons pas les identifiants de vulnérabilités car nous préférons tester l'approche avec les mots-clés.

Les nombres d'éléments pour chaque catégorie d'attribut sont donnés dans la table 5.10.

On peut remarquer que 3444 adresses IP différentes sont impliquées en tant qu'attaquants dans le corpus d'alertes. Lorsqu'elles sont externes, ces adresses appartiennent à 1719 noms de réseaux référencés dans la base IANA. En ce qui concerne les hôtes internes, nous avons référencé 7 fonctions d'hôtes (serveurs HTTP, FTP, SMTP, SSH, DNS, Proxy HTTP et d'impression). Le réseau est compartimenté en 7 domaines dont font partie le réseau d'enseignement, celui du personnel, la DMZ (*DeMilitarized Zone*). Seuls 111 noms d'hôtes ont pu être obtenus auprès du serveur DNS du réseau. Les adresses IP qui ne sont associées à aucun nom sont directement associées à un identifiant de domaine réseau.

La figure 5.11 représente un exemple de relations taxinomiques. Dans cet exemple, les hôtes du réseau local d'adresse 172.17.0.0/16 sont les machines dédiées à l'en-

Attribut		Nombre d'éléments
Attaque	Signature	98
	Mot-clé	212
Attaquant	Adresses IP	3444
	Netname	1719
	Nom d'hôte	111
	Fonctions	7
	Domaines	7
Victime	Adresses IP	790
	Adresses LAN	12
	Nom d'hôte	111
	Domaines	7
	Fonctions	7
Total		6525

FIG. 5.10 – Nombre d'attributs du corpus d'alertes

seignement dans le réseau de Supélec. Nous appelons ce réseau `eleves`. L'adresse IP 172.17.10.1 est la machine appelée `graal`. Vis-à-vis de l'extérieur, `graal` a comme adresse IP 193.54.194.10. `graal` a entre autres fonctions celle de serveur SSH. En revanche, ni les fonctions ni le nom de l'hôte dont une adresse est 172.17.10.46 ne sont connus.

Dans le corpus, 790 adresses IP distinctes sont impliquées en tant que victime. Ces victimes appartiennent toutes au réseau de Supélec. Les remarques concernant les attaquants internes valent aussi pour les victimes.

Le corpus contient au total, 6525 valeurs d'attribut distinctes.

Dans le cadre de cette expérimentation, les structures taxinomiques ont été construites *a priori* à partir du corpus d'alertes, suivant des algorithmes correspondant à ceux donnés en section 5.5.2. Ainsi, l'insertion d'une alerte dans le système de fichiers consiste uniquement à créer un fichier dont le nom est formé à partir de la sonde et du numéro de série, et dont le chemin est composé des trois attributs de l'alerte. La commande correspondante est la suivante :

```
cp message alertid/attackerip/victimip/sensor-serial
```

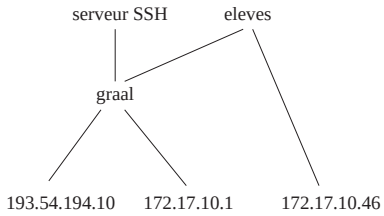

FIG. 5.11 – Exemple de taxinomie

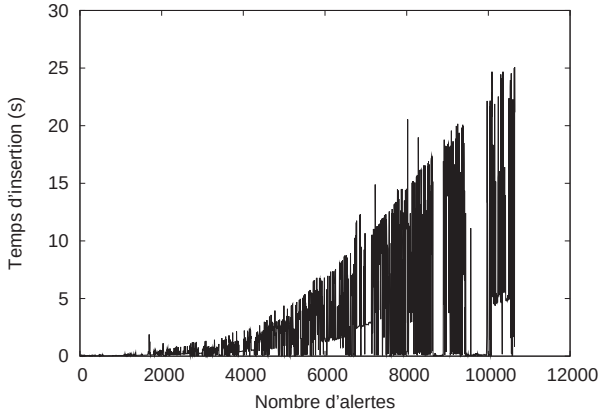

FIG. 5.12 – Durée d'insertion d'une alerte (segments)

b) Temps d'insertion

Nous insérons itérativement chaque alerte du corpus et mesurons le temps d'insertion toutes les 5 alertes. La courbe représentée en figure 5.12 représente l'évolution du temps d'insertion en fonction du nombre d'alertes précédemment stockées, pour les 10000 premières alertes.

On constate que le temps d'insertion croît globalement avec le nombre d'alertes. On peut toutefois remarquer des plages où la durée d'insertion chute brutalement. C'est le cas en particulier des insertions 6950-7050, 8700-9000 et 9500-10000. L'analyse détaillée de ces plages d'insertions révèle qu'il s'agit de séries d'alertes correspondant à des attaques par balayage IP sur les réseaux 193.54.192.0/24, 193.54.193.0/24, 193.54.194.0/24. Il s'agit donc d'alertes quasi-identiques, ne divergeant éventuellement que par l'adresse IP de la victime. Les ressemblances entre ces alertes peuvent expliquer les temps d'insertion faibles.

La durée d'insertion d'une alerte est fortement influencée par les valeurs de ses attributs. Plus le nombre de valeurs déductibles à partir d'une valeur d'attribut avec les axiomes de la logique des alertes est grand, plus la durée d'insertion est importante. Par exemple, les valeurs déductibles des signatures Snort 474 et 1807 sont schématisées en figure 5.13. La structure taxinomique de la signature 1807 est plus *complexe* que celle de la signature 474. L'attribut attaque des alertes insérées dans les plages 8700-9000

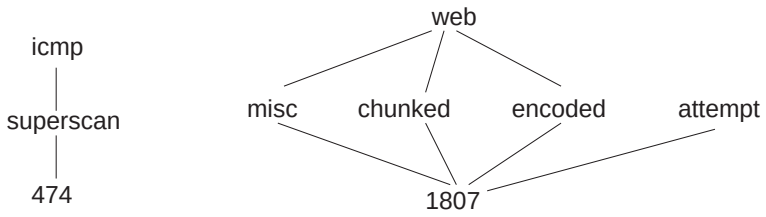

FIG. 5.13 – Deux taxinomies partielles d'attaques

et 9500-10000 est la signature 474. Les temps d'insertion des alertes dont l'attaque est 1807 sont parmi les plus élevés.

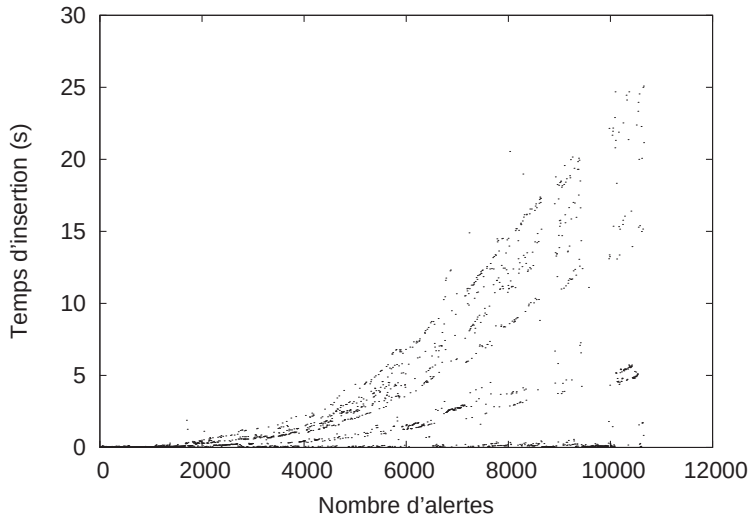

FIG. 5.14 – Durée d'insertion d'une alerte (points)

Cette hypothèse est confirmée par la courbe 5.14. Cette courbe représente les mêmes données que celle donnée que la courbe 5.12, avec des points au lieu de segments. Cette représentation permet de mettre en évidence courbes de niveaux, qui correspondent à des familles d'alertes dont les taxinomies des attributs sont de complexité analogue.

Le graphique 5.15 représente uniquement les temps d'insertion d'alertes dont la victime est le serveur web de Supélec pour les attaques dont les identifiants de signature sont respectivement 1214, 1149 et 1519. On voit qu'à valeurs d'attributs constants, l'évolution de la durée d'insertion est assez régulière. Les artefacts sont vraisemblablement liés à des défaut de pagination et aux mécanismes de cache utilisés dans l'implémentation dans LISFS.

Evoquons maintenant les durées d'insertion des alertes. Les expérimentations effectuées par Padioleau et Ferré décrites dans [64] portent en particulier sur la structuration de pages *man* dans LISFS. Les pages sont les fichiers, décrits par la rubrique `NAME` de la page de *man* associée. L'expérience porte sur 11502 pages de man, indi-

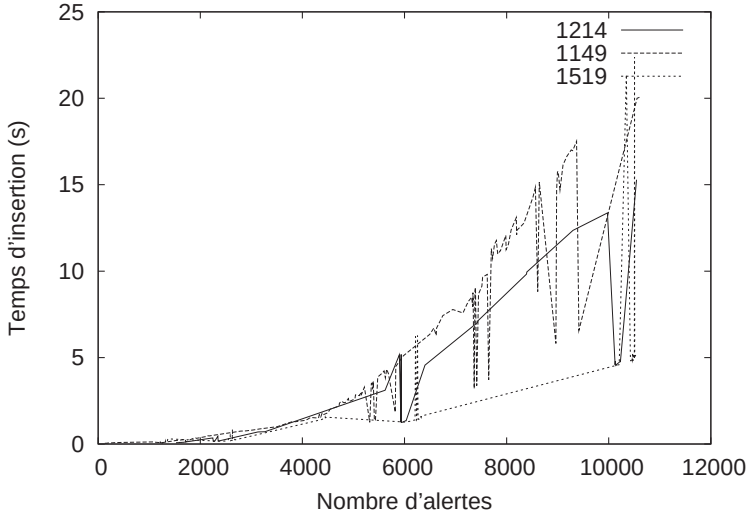

FIG. 5.15 – Durée d'insertion pour les signatures 1214,1149,1519

viduellement décrites par en moyenne 21 attributs parmi 43442. Les résultats montrent que le temps de création d'un fichier est globalement constant, de l'ordre de 0.5 à 1s. Ce temps de création se détériore (plus de 1 à 2s) lorsque le nombre de fichiers insérés avoisine 10000.

Dans notre cas, les durées d'insertion sont plus élevées que dans les expérimentations effectuées par Padioleau et Ferré. Les différences constatées s'expliquent d'abord par les taxinomies utilisées. Les taxinomies des alertes sont plus complexes que celles des pages man.

Les différences de durées s'expliquent aussi par la différence de puissance entre les machines utilisées pour les expérimentations. Dans le cas de Padioleau, une machine dotée d'un processeur Pentium IV à 2.4 Ghz et de 756 Mo de mémoire est utilisée.

La mémoire disponible et la puissance de processeur semblent être deux paramètres déterminants pour les performances globales du système. D'une part, le processeur est totalement monopolisé de l'insertion des alertes. D'autre part, on peut constater au voisinage de la 4000ème insertion une inflexion de la pente de la courbe. Cette inflexion coïncide avec l'épuisement de la mémoire centrale disponible, majoritairement occupée par le processus chargé de stocker les alertes.

Rappelons que dans le cas du réseau de Supélec, 20000 alertes sont produites chaque jour, soit une moyenne d'une alerte toutes les 5 secondes. Ce débit est toléré par LISFS jusqu'à la 6000ème alerte stockée. En utilisant une machine dotée de plus de mémoire centrale et d'un processeur plus rapide, la quantité d'alertes stockables avant que le temps de traitement ne dépasse le débit des sondes avoisine 10000. Ce chiffre est clairement insuffisant, puisqu'il représente la moitié de la production quotidienne d'alertes.

Notons qu'au temps d'insertion des alertes s'ajoute celui de la création des répertoires. Toutefois, ce surcoût semble négligeable devant celui de l'insertion d'alertes puisque la création des structures taxinomiques correspondant aux 6525 attributs est

1	V_172.16.10	3773	V_192.168.1
1	V_192.168.5	341	V_192.70.40
4270	V_193.54.192	1584	V_193.54.193
914	V_193.54.194	235	V_dmzpriv
7033	V_dmzpub	764	V_entree
2	V_ftpserver	41	V_lprserver
1	V_mailserver	10	V_personnel
194	V_proxyHTTP	3	V_ras
1584	V_rez	12	V_smtp_server
1	V_sshserver	341	V_ssir
7172	V_webserver	5669	A_access
1882	attempt	1	A_dns
68	A_dos	8702	H_ext
6	A_ftp	2054	A_icmp
2182	H_int	26	A_ip
33	A_msdtc	6	A_overflow
508	A_proxy	121	A_rpc
298	A_scan	1	A_smtp
799	A_snmp	7039	A_web

FIG. 5.16 – Incréments à la racine du système de fichiers

de l'ordre de quelques secondes.

Tout d'abord, un traitement amont visant à réduire la quantité d'alertes stockées dans le système de fichiers semble indispensable. Ce traitement peut par exemple consister à ne stocker que les descriptions d'alertes distinctes. Actuellement, nous stockons toutes les alertes, y compris celles dont la description est identique. Rappelons que les 287088 alertes ne représentent que 14139 triplets (attaque, attaquant, victime) distincts.

Ensuite, étant donnée la proportion majoritaire de fausses alertes, on peut supposer que l'opérateur purge la base d'alertes une fois que les faux positifs ont été identifiés. Si cette purge est faite quotidiennement, le système reste utilisable.

c) Navigation et Interrogation

Nous donnons maintenant quelques mesures de temps de réponse sur des requêtes effectuées dans un système de fichiers où sont stockées 10820 alertes. L'expérimentation a été interrompue car les temps d'insertion devenaient trop importants.

A titre indicatif, une requête `ls -l` à la racine du système de fichiers nécessite 43.6s de traitement. Le système propose en réponse 38 incréments et aucune alerte, dont la liste est donnée en Figure 5.16. L'absence d'alertes dans la réponse du système se justifie par le fait qu'aucune alerte ne possède de description vide.

Les nombres indiquent la taille de l'extension de la requête $\texttt{pwd} \wedge \texttt{prop}$ où `prop` est la propriété située à la droite du nombre. Les propriétés préfixés par `V_` désignent les victimes, celles préfixées par `A_` les attaques et celles préfixées par `H_` les attaquants.

Admettons que l'opérateur s'intéresse aux attaques contre les serveurs de mail. La commande `cd V_mailserver` permet de sélectionner les alertes dont la description est satisfait la propriété `victim is mail_server`.

La modification incrémentale du répertoire courant est instantanée. La liste retournée contient une seule alerte, `1-2211`. Pour visionner l'alerte, l'opérateur utilise la commande `cat 1-2211`, qui retourne le message Snort suivant :

```
[**] [1:1549:3] SMTP HELO overflow attempt [**]
[Classification: Attempted Administrator Privilege] [Priority: 1]
12/18-02:22:25.935402 192.168.1.10:1889 -> 172.16.10.3:25
TCP TTL:124 TOS:0x0 ID:51782 IpLen:20 DgmLen:48 DF
******S* Seq: 0x887A3E15  Ack: 0x0  Win: 0xFAF0  TcpLen: 28
TCP Options (4) => MSS: 1460 NOP NOP SackOK
[Xref => http://www.securityfocus.com/bid/7726]
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2000-0042]
```

La première ligne donne la description de l'attaque, suivie de la classification associée et de la sévérité. Les adresses IP de l'attaquant et de la victime sont indiquées en 3ème ligne. Les autres informations concernent l'entête du paquet IP qui a donné lieu à l'alerte, ainsi que des références vers des identifiants de vulnérabilités.

Le fait que la réponse à la requête soit dépourvue d'incréments indique qu'il n'existe qu'une seule occurrence d'attaque de type smtp. Aucun autre hôte n'a donc été victime d'une telle attaque.

La liste précédente montre qu'il existe une alerte impliquant le serveur web en tant que victime. Le contenu de cette alerte révèle qu'il s'agit d'une attaque de type scan, correspondant à la signature 474 :

```
[**] [1:474:3] ICMP superscan echo [**]
```

La commande `cd /A_474` suivie de +ls+ retourne la liste suivante :

253	V_192.70.40	15	V_193.54.192
270	V_193.54.193	262	V_193.54.194
2	V_dmzpriv	5	V_dmzpub
1	V_ftpserver	1	V_lprserver
7	V_personnel	1	V_proxyHTTP
1	V_ras	270	V_rez
1	V_smtp_server	1	V_sshserver
253	V_ssir	3	V_webserver
800	H_ext		

On voit que la totalité des 800 attaques activant la signature 474 ne proviennent que de l'extérieur (il n'existe pas d'attribut `attaquant` autre que `H_ext`). La totalité des plages d'adresses IP possédées par Supélec (192.70.40.0/24, 193.54.192.0/24, 193.54.193.0/24, 193.54.194.0/24 ont été balayées par ce type d'attaque. Des fonctions et des domaines d'hôtes apparaissent (*e.g.* `V_ftpserver`) parce que du point de vue des

taxinomies, ces valeurs ne sont pas subsumées par les plages d'adresses IP. Les nombre d'occurrences sont voisins de 255, ce qui est révélateur de ce type d'attaques car au cours d'un balayage, la totalité des adresses IP d'une plage donnée (255 pour une plage de classe C) sont touchées.

Supposons maintenant que l'opérateur s'intéresse aux attaques de type web, dont la victime est un serveur mandataire web. La commande suivante permet de sélectionner les alertes dont la description satisfait ces conditions :

```
cd /A_web/V_proxyHTTP/
```

En consultant le répertoire courant avec la commande `ls`, l'opérateur constate que le seul incrément portant sur les attaquants est `H_int`, ce qui signifie que les attaquants viennent que de l'intérieur. Nous avons vu que ce phénomène était un faux positif (cf section 5.1.1). L'opérateur peut donc décider de qualifier de `false_positive` l'ensemble des alertes dont la description satisfait la formule `A_web/H_int/V_proxyHTTP` :

```
mv A_web/H_int/V_proxyHTTP/ /false_positive/
```

Ainsi, les nouvelles alertes dont la description satisfait cette description sont qualifiées de faux positifs. Par la suite, la commande `cd /!false_positive` permet de sélectionner les alertes qui ne sont pas qualifiées de faux positif par un profil (voir section 5.4.2, d)).

5.7 Conclusion

Dans ce chapitre, nous avons proposé d'appliquer le paradigme de l'analyse de concepts aux ensembles d'alertes pour permettre à un opérateur de sécurité d'interroger et de naviguer dans base de données d'alertes de manière flexible. On peut classer cette approche dans la famille des techniques de corrélation implicites car le résultat est de constituer des groupes d'alertes partageant des propriétés communes. Notre approche améliore les techniques de corrélation implicites car :

- les groupes d'alertes sont décrits par une formule de la logique utilisée pour décrire les alertes,
- les groupes obtenus ne sont pas mutuellement exclusifs,
- la création des groupes d'alertes est incrémentale et ne nécessite pas de reconstruire la structure de stockage des alertes à chaque fois

Pour instancier l'analyse de concepts logique proposée par Ferré et Ridoux, nous avons proposé une logique des alertes servant à décrire les alertes et effectuer des requêtes au système. Cette logique est une logique propositionnelle ; il est envisageable d'utiliser des langages plus élaborés comme par exemple Lambda, mais notre logique nous semble suffisamment expressive pour les besoins de la corrélation ; en outre, plus la logique est complexe, plus la tâche d'analyse de concepts est coûteuse.

Le vocabulaire des formules de la logique des alertes est munie de termes abstraits qui permettent de décrire les groupes d'un haut niveau.

L'implémentation consiste à stocker les alertes issues des sondes de détection d'intrusions dans le système de fichiers logique LISFS, proposé par Padioleau et Ridoux.

L'utilisation du système de fichiers pour la recherche et la navigation dans un ensemble d'alertes se révèle utile. Malheureusement, les tests effectués montrent des limitations sur le nombre d'alertes manipulables, liées au temps de traitement de chaque alerte et aux durées de réponses du système aux requêtes qui lui sont soumises.

Ces limitations sont d'abord liées à la complexité inhérente à l'analyse de concepts. Cependant, Padioleau précise que l'implémentation actuelle du système de fichiers logique favorise la vitesse de consultation, au détriment de la vitesse de création des fichiers. Il serait intéressant d'étudier un algorithme capable de combiner un temps de création raisonnable et de consultation raisonnables.

Dans l'application de l'analyse de concepts logique aux alertes présentée ici, seules des connaissances explicites sont utilisées pour permettre à l'opérateur de naviguer dans le treillis de concepts. Au titre des travaux futurs, nous pouvons envisager d'utiliser l'analyse de concepts pour extraire des connaissances implicites d'un contexte, comme le suggère Ferré dans [28]. En effet, l'analyse de concepts a été appliquée dans les domaines de l'analyse de données, de la fouille de données et l'apprentissage. Les connaissances extraites par fouille de données permettent d'identifier des règles d'associations vérifiées par des données. Une règle d'association peut par exemple découvrir le fait que les attaques web internes sont *très souvent* dirigées contre le serveur mandataire, au lieu de laisser le soin à l'opérateur de le constater lui même. Dans un contexte différent de l'analyse de concepts, la découverte de règles d'associations a déjà été appliquée sur des bases d'alertes par Manganaris [52]. Son approche nous semble prometteuse et nous envisageons de la coupler à la nôtre.

Chapitre 6

Corrélation de symptômes intrusifs : une application des chroniques

Les approches de corrélation d'alertes peuvent se diviser en trois classes : implicites, semi-explicites et explicites. Les approches de corrélation explicites ont pour but de reconnaître des schémas de corrélations préétablis, alors que les approches de corrélation implicites ont pour objectif de *découvrir* des relations entre les alertes. L'approche semi-explicite, constitue une généralisation de l'approche explicite en ce que les attaques font partie des variables instanciables des schémas de corrélation.

Dans ce chapitre, nous proposons une approche de corrélation explicite basée sur le formalisme des chroniques. Nous décrivons d'abord brièvement ce formalisme, puis nous expliquons comment il est appliqué à la corrélation d'alertes en détection d'intrusions. Dans une troisième section, nous donnons des exemples de chroniques pour illustrer nos propos. Enfin, nous concluons et évoquons des extensions futures à apporter à cette approche de corrélation.

6.1 Présentation des Chroniques

Le paradigme des chroniques est élaboré au sein du laboratoire des techniques logicielles de France Télécom par Dousson [25] depuis 1994. Un outil de reconnaissance de chroniques, baptisé CRS[1] (*Chronicle Recognition System*), y est aussi développé et a été utilisé pour nos expérimentations. La présentation des chroniques qui est faite dans cette section est brève et informelle. Le lecteur est invité à se référer à la thèse de Dousson [25], ainsi qu'à [26] pour de plus amples détails.

De manière très générale, les chroniques fournissent une infrastructure de modélisation de systèmes dynamiques. Elles incluent un mécanisme de supervision de l'évolution du système modélisé, afin d'y détecter des changements.

Les chroniques ont initialement été conçues pour la reconnaissance de pannes

[1] `http://crs.elibel.tm.fr`

dans les réseaux de communications. La généricité de leur formalisme et de leur implémentation ont depuis permis de les appliquer à d'autres domaines tels que la médecine (détection d'arythmies cardiaques), la surveillance de trafic routier (modélisation du comportement de véhicules), ou encore la surveillance d'installations de turbines à gaz. Nous proposons d'appliquer ce formalisme pour corréler des alertes issues de sondes de détection d'intrusions.

Les chroniques sont basées sur un formalisme dans lequel le temps est fondamental, contrairement aux systèmes experts classiques, qui bâtissent leur raisonnement essentiellement à partir de règles et relèguent souvent l'information temporelle au second plan.

Notons que ce traitement particulier du temps est l'opposé de celui de l'approche de corrélation implicite présentée au chapitre 5 où le temps est traité comme un attribut quelconque, muni d'une structure taxinomique, ne prenant pas en compte des relations telles que *avant*, *après*, *en même temps*, etc. Ces deux traitements de l'information temporelle sont complémentaires.

Une chronique est un ensemble d'événements ponctuels, liés entre eux par des contraintes temporelles, et dont l'occurrence peut être soumise à un certain contexte.

Dans la suite de cette section, nous décrivons brièvement la représentation des chroniques, puis nous présentons le processus de reconnaissance.

6.1.1 Représentation des chroniques

En intelligence artificielle, une approche naturelle pour représenter l'information temporelle consiste à associer des assertions à des dates. La représentation des chroniques repose sur la logique temporelle réifiée [55, 5, 74]. Dans ce formalisme, les termes propositionnels sont reliés aux informations temporelles ou à d'autres termes propositionnels *via* des prédicats particuliers. Par exemple, la proposition

$$\text{HOLD}(\text{is(light,on)}, T)$$

est vraie à condition que la proposition "la lumière est allumée" (is(light,on)) soit vraie pendant une durée T. HOLD est le prédicat qui qualifie temporellement la proposition atemporelle is(light,on).

a) Représentation du temps

La représentation du temps repose sur des instants comme primitives temporelles. Un intervalle temporel est une paire $T = (t_1, t_2)$ où t_1 et t_2 représentent respectivement la borne supérieure et la borne inférieure de l'intervalle T.

On considère que la résolution des instants est suffisamment fine pour la dynamique de l'environnement. En effet, dans les chroniques, si deux événements identiques se produisent au même instant, un seul est pris en compte par le système de reconnaissance. En détection d'intrusions, nous sommes confrontés à des rafales d'alertes identiques se produisant pendant de courtes durées. Il est donc important de choisir une résolution temporelle suffisamment fine pour que chaque alerte soit prise en compte individuellement par le système de reconnaissance.

b) Attributs du domaine

Les attributs du domaine sont les propriétés atemporelles de la logique réifiée [55, 5, 74], qui servent à décrire l'environnement du système surveillé. Un attribut du domaine est une proposition d'une logique propositionnelle multivaluée classique, du type

$$P : v \qquad (v \in V)$$

où P est le nom d'attribut et v sa valeur, appartenant à un domaine V.

Il est possible d'utiliser des symboles fonctionnels à arité quelconque comme attributs du domaine à l'aide de variables. Chaque instance de la fonction est alors un domaine d'attribut indépendant des autres. Dans la syntaxe des chroniques, les variables sont préfixées par un point d'interrogation. Par exemple, l'attribut du domaine $Load(?host)$ dénote la charge d'un serveur $?host$, susceptible de changer au cours du temps ; son domaine est par exemple V ={high, normal, low}. Une instance de ce domaine d'attribut est $Load(www)$, où www est le serveur web du réseau.

Notons que contrairement aux variables utilisées pour les attributs du domaine, les symboles temporels ne sont pas préfixés par un point d'interrogation. Les instants sont toujours des variables, on ne peut pas spécifier de date sous forme de constante dans les chroniques. La variable correspondant à un instant est instanciée lors de l'occurrence d'un événement.

Les *messages* sont des attributs du domaine particuliers, auxquels aucune valeur n'est associée. Ils sont utilisés pour modéliser des alertes, par exemple.

c) Prédicats de réification

Les prédicats de réification sont utilisés pour qualifier temporellement les attributs du domaine. Quatre types de prédicats de réification sont utilisés dans les chroniques : *hold*, *event*, *noevent* et *occurs*. Leur syntaxe et leur sémantique informelle sont données en figure 6.1.

- le prédicat *hold* est une assertion représentant la persistance de la valeur d'un attribut du domaine pendant un intervalle de temps, sans que la date de changement ou d'affectation de la valeur ne soit forcément connue,
- la prédicat *event* dénote l'occurrence d'un message ou le changement *instantané* de la valeur d'un attribut du domaine à une date donnée. Les prédicats *event* et *hold* sont schématisés en figure 6.2,
- le prédicat *noevent* est une assertion qui spécifie quel motif d'événement est interdit durant un intervalle, c'est-à-dire celui dont l'occurrence invaliderait une chronique,
- le prédicat *occurs* est un prédicat permettant de compter des occurrence de messages.

La valeur ∞ peut être utilisée dans les prédicats de réification à la fois dans les compteurs d'événements (pour désigner "un nombre quelconque d'événements") et dans les intervalles temporels.

Le prédicat $occurs$ est unifiant dans le sens où les relations suivantes sont vérifiées :

$$\begin{aligned} noevent(P, (t_1, t_2)) &\equiv occurs((0, 0), P, (t_1, t_2)) \\ event(P, t_1) &\equiv occurs((1, \infty), P, (t_1, t_1 + 1)) \end{aligned}$$

Prédicat	Sémantique
$hold(P : v, (t_1, t_2))$	Assertion spécifiant que l'attribut du domaine P doit conserver sa valeur v durant l'intervalle de temps $[t_1, t_2[$.
$event(P : (v_1, v_2), t)$	Changement de la valeur de l'attribut du domaine P de v_1 à v_2 à l'instant t.
$event(P, t)$	Occurrence d'un message P à l'instant t.
$noevent(P, (t_1, t_2))$	Assertion spécifiant qu'aucun motif d'événement P ne doit survenir entre t_1 et t_2.
$occurs((n_1, n_2), P, (t_1, t_2))$	n occurrences du message P doivent être reçues par le système entre t_1 et t_2, avec $n_1 \leqslant n \leqslant n_2$.

FIG. 6.1 – Prédicats réificateurs

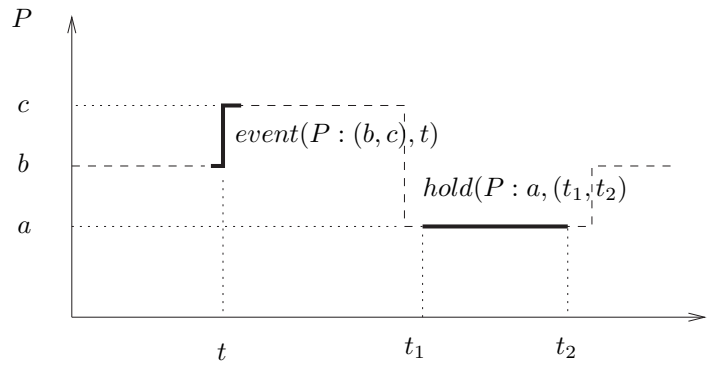

FIG. 6.2 – Prédicats *hold* et *event*

6.1.2 Reconnaissance des chroniques

Les scénarios des chroniques sont appelés des *modèles de chroniques*. Dans la suite, nous utilisons indifféremment l'un ou l'autre terme. L'utilisation des chroniques se déroule en deux étapes.

D'abord, les modèles de chroniques sont compilées afin de tester leur cohérence et de les coder sous forme de structures de données efficaces (des graphes de contraintes temporelles).

Ensuite, une *instance de chronique* est créée pour chaque modèle de chronique et le flot d'événements est traité en ligne. La reconnaissance est incrémentale (chaque événement est intégré dès son arrivée) et en une seule lecture du flot d'entrée.

Dans la suite nous détaillons les modèles et les instances de chroniques, puis le processus de reconnaissance.

```
1  chronicle example1 {
2    event(e1,t1);
3    event(e2,t2);
4    event(e3,t3);
5
6    t1<t2<t3
7    t3-t2 <= 4
8  }
```

FIG. 6.3 – Exemple de chronique

a) Modèles et instances de chroniques

Un modèle de chronique est destiné à représenter un schéma d'évolution d'une partie de l'environnement. Il est composé d'instants, d'événements, d'assertions, de contraintes temporelles entre les différents événements et d'actions à effectuer lorsque la reconnaissance de la chronique a lieu.

Un exemple de modèle de chronique est donné en figure 6.3. Le nom de la chronique est `example1`. Trois motifs d'événements `e1`, `e2` et `e3` sont utilisés. Les contraintes temporelles indiquent que `e1` doit arriver avant `e2` qui doit arriver avant `e3` et que `e3` doit arriver au plus 4 secondes après `e2`. Des exemples de scénarios appliqués à la détection d'intrusions sont donnés dans les sections suivantes.

Une instance de chronique est un produit du système de reconnaissance, c'est le résultat de la mise en correspondance des événements du flot d'entrée avec les motifs du modèle de chronique. Le système détecte tout sous-ensemble d'occurrence d'événements d'un modèle de chroniques en tenant compte des contraintes temporelles et des assertions. Un sous ensemble incomplet est donc une instance partielle de chronique, c'est-à-dire une chronique qui n'est pas encore reconnue.

b) Reconnaissance de chroniques

Une occurrence d'événement dont la partie atemporelle s'unifie avec un motif d'événement est toujours candidat pour être intégré dans une instance de chronique. Son intégration dépend de son adéquation avec les contraintes de la chronique, de sa date d'occurrence et des événements précédemment intégrés.

Les événements peuvent être partagés par plusieurs chroniques ; le système de reconnaissance gère l'ensemble des instances concurrentes. L'ensemble des instances sont gérées sous la forme de fenêtres temporelles qui sont graduellement contraintes lors l'intégration d'événements.

L'occurrence d'un événement peut entraîner l'invalidation d'instances de chroniques (et donc leur destruction) : si la limite temporelle associée à un événement est atteinte, l'ensemble des instances en attente d'un événement prévu avant cette limite temporelle sont invalidées. Une instance de chronique peut aussi être invalidée si une assertion est violée.

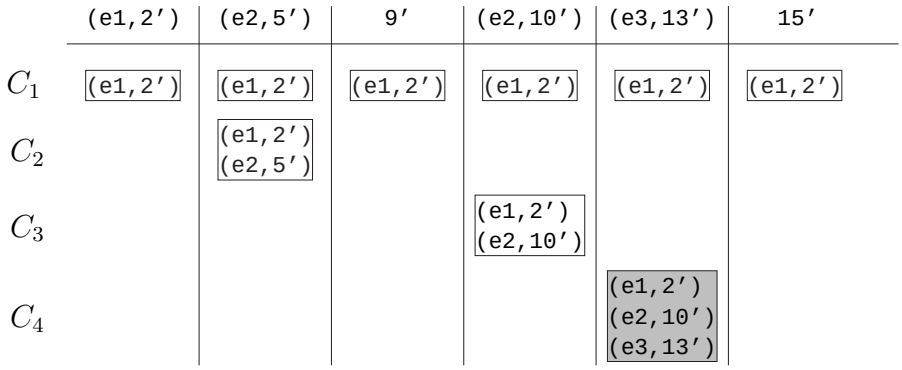

FIG. 6.4 – Reconnaissance de chronique

L'occurrence d'un événement peut aussi entraîner la suppression d'assertions devenues obsolètes.

Un élément très important des chroniques est que la reconnaissance est exhaustive. Lorsqu'un événement est intégré dans une instance, le système ne peut pas savoir *a priori* si l'événement s'intègre correctement en regard des événements à venir. Le système de reconnaissance doit maintenir l'hypothèse que la chronique dans laquelle est intégré l'événement n'est pas nécessairement celle qui sera reconnue. Les chronique dans lesquelles un événement est intégré sont dupliquées avant l'intégration. Le système maintient ainsi toutes les hypothèses afin que l'ensemble des séquences d'événements satisfaisant les contraintes des chroniques soient reconnues.

Par exemple, si la chronique schématisée en figure 6.3 est confrontée au flot d'événements ((e1,1'),(e2,3')(e2,4')(e2,5')(e3,6')), alors trois instances de la chronique `example1` sont reconnues.

Un exemple synthétique de reconnaissance de chroniques est donné en Figure 6.4. Le modèle de chronique utilisé correspond à celui donné en figure 6.3. A l'initialisation du système, une instance vide de la chronique est crée ; un événement `e1` est reçu à 2' et intégré à C_1 ; à 5', `e2` est reçu, C_1 est dupliquée et `e2` est intégré à C_2. À 9', une contrainte temporelle n'est plus satisfaisable donc C_2 est invalidée. À 13, `e2` est reçu et intégré dans C_3, duplicata de C_1. A 13, `e3` est reçu et conduit à la reconnaissance de C_4. C_3 ne peut plus être reconnue à 15' et est donc invalidée.

6.2 Chroniques pour la détection d'intrusions

L'application des chroniques à la détection d'intrusions consiste à faire une analogie entre les pannes et les intrusions, qui appartiennent respectivement aux domaines de la sûreté de fonctionnement et de la sécurité. Comme nous allons le voir, cette analogie n'est pas triviale car les manifestations de pannes sont généralement déterministes, ce qui n'est pas le cas des intrusions, du fait de la présence d'acteurs humains malicieux.

Dans la première partie de cette section, nous discutons quel point de vue –attaquant ou système surveillé– les scénarios reflètent. Nous justifions ensuite l'utilisation qui est

faite des chroniques en tant que *manager*, selon la définition de IDMEF [17].

6.2.1 Quels types de scénarios ?

Dans le contexte de la corrélation explicite, on peut envisager deux types de scénarios : des scénarios dont les étapes sont des attaques (du point de vue de l'attaquant) ou bien des scénarios dont les étapes sont des alertes (du point de vue du système attaqué. Nous appelons les premiers des *scénarios d'intrusions* et les second des *scénarios d'alertes*.

Les approches de corrélation explicites existantes comme Lambda [16] ou ADele [60] proposent les deux points de vue. Plus exactement, ces langages proposent de spécifier des scénarios d'intrusions, afin d'en dériver automatiquement des scénarios d'alertes. En effet, les alertes sont les conséquences des attaques qui constituent les intrusions ; les alertes sont le substrat utilisé pour construire les scénarios au cours de la reconnaissance.

L'avantage de la spécification de scénarios d'intrusions plutôt que de scénarios d'alertes tient au niveau d'abstraction plus élevé des premiers par rapport aux seconds. Un opérateur de sécurité se met à la place d'un attaquant et *imagine* des scénarios d'intrusions possibles dans le système et les traduit dans un langage dédié (*e.g.* Lambda [16] ou ADeLe [60]). L'opérateur n'a donc pas besoin d'analyser des fichiers d'alertes pour identifier des séquences d'alertes indicatives d'un scénario d'intrusions.

De plus, comme les scénarios d'intrusions reconnus représentent le point de vue d'un attaquant, ils sont plus informatifs pour l'opérateur de sécurité.

Enfin, à partir de scénarios d'intrusions, il est possible d'inférer des actions inobservées à partir de scénarios partiellement reconnus pour informer l'opérateur de sécurité d'attaques qui n'ont pas provoqué d'alertes. C'est ce que proposent Cuppens et Miège dans [15].

La spécification de scénarios d'attaques est séduisante mais elle présente plusieurs défauts. Premièrement, pour que la spécification de scénarios d'intrusions soit rentable, ils faut qu'ils soient doivent être déterministes et fréquents ; en d'autres termes, il faut que le scénario d'attaque imaginé par l'expert corresponde à celui utilisé par les attaquants. Or, nous pensons que les scénarios d'attaques sont imprévisibles.

Les scénarios utilisés en corrélation explicite peuvent être vus comme une transposition au niveau des alertes des signatures utilisées dans les techniques de détection d'attaques morphoscientes. On pourrait donc généraliser le reproche sur l'indéterminisme des scénarios d'intrusions aux signatures des techniques de détection morphoscientes. Pourtant, la détection d'intrusions par signatures d'attaques demeure la plus efficace et sont la plus utilisées en milieu opérationnel. En fait, la distribution à grande échelle par le biais de l'Internet de techniques d'attaques (sous la forme de fichiers compilables par exemple) font que les mêmes attaques sont réutilisées par un grand nombre d'attaquants. Les symptômes d'attaques sont donc souvent les mêmes et justifient de fait l'approche par signatures. En revanche nous croyons que l'agencement des briques d'intrusions sous forme de scénarios ne sont pas déterministes. Cette critique est aussi émise par Qin et Lee dans [69].

Deuxièmement, les sondes de détection d'intrusions ne sont pas fiables. Elles génèrent beaucoup de fausses alertes et manquent des attaques. Un système de

corrélation d'alertes explicite basé sur des scénarios d'intrusions construirait des plans d'intrusions inutiles en ce sens que les alertes utilisées pour les construire sont des fausses alertes. Le système de corrélation serait à terme débordé. A ceci s'ajoute le caractère malicieux de l'attaquant qui peut en plus profiter de la médiocrité des sondes pour engendrer des alertes et saturer les systèmes de corrélation explicite. Cette dernière remarque est aussi formulée par Pouzol et Ducassé dans [67].

Deuxièmement, l'imagination de scénarios d'attaques et leur rédaction nécessite du temps et des connaissances expertes en techniques d'attaques et sur l'architecture du système d'informations surveillé. Ce type d'expertise est rare et donc cher. Notons que dans l'approche de Cuppens *et al* [16, 15], le travail des experts doit se cantonner à la description en Lambda des *attaques* avec leur pré-conditions et leurs post-conditions. Les scénarios possibles sont construits automatiquement à partir d'une base de descriptions d'attaques, par la mise en correspondance des pré-conditions et des post-conditions.

Dans le cas des chroniques, outre le fait qu'il est difficile d'envisager *a priori* les scénarios d'intrusions, il est aussi difficile de spécifier les contraintes temporelles entre les événements qui participent à un modèle de chronique. En effet, les intervalles de temps qui séparent les actions d'un attaquant sont très variables. Qui plus est, un attaquant peut choisir d'espacer les étapes de son intrusion de manière à ce que ses actions ne soient pas corrélées.

Troisièmement, dans les faits, la dérivation des scénarios d'attaques en scénarios d'alertes est difficilement automatisable. Il n'y a pas toujours de correspondance directe entre les actions d'un attaquant et leurs manifestations au niveau du système surveillé. Non seulement la traduction d'actions abstraites en événements ou alertes est difficile, mais en plus certaines actions sont effectuées à l'extérieur du système surveillé et sont donc inobservables. Par exemple, une intrusion peut nécessiter le déchiffrement d'un mot de passe qui est effectuée par l'attaquant et ne donne lieu à aucune alerte. Par contre, l'étape préliminaire qui consiste à récupérer le mot de passe chiffré peut donner lieu à des actions plus ou moins légitimes et donc à des alertes. Si la dérivation n'est pas automatisée, la spécification des scénarios d'intrusions constitue ainsi une sur-couche qui s'ajoute à la spécification de scénarios d'alertes.

Nous ne destinons pas l'utilisation de chroniques à la modélisation de scénarios d'intrusions. Nous modélisons des scénarios d'alertes, directement à partir d'observations dans le flux d'alertes générées par les analyseurs. De fait, l'analyse des alertes produites dans un milieu opérationnel révèle des séquences récurrentes d'alertes, issues de phénomènes déterministes. Cette récurrence ne présage pas du caractère malicieux des alertes. Ainsi, contrairement aux scénarios d'intrusions, les phénomènes modélisés par nos scénarios d'alertes n'ont pas forcément pour origine une intrusion. Il peut s'agir d'activités erratiques, réputées pour engendrer des alertes.

Dans le cas de phénomènes malicieux, le déterminisme des séquences peut par exemple être le fait d'outils d'attaques automatisés. Comme l'acteur humain n'intervient plus dans le processus intrusif, le comportement des intrusions est toujours le même. L'utilisation des outils automatisés se généralise et représentent une proportion importante des alertes constatées.

Comme la plupart des analyseurs utilisés à l'heure actuelle sont mono-

événementiels[2], que chaque étape de l'intrusion est une action intrusive en soi et que les analyseurs produisent au mieux une alerte par étape, le nombre d'alertes produites est important. Ceci contribue à l'excès global d'alertes, mais aussi à l'excès de fausses alertes et à la pauvreté de leur sémantique.

Comme dans tout processus diagnostique, l'utilisation de plusieurs symptômes dans le processus de reconnaissance des attaques permet à la fois :

- de confirmer ou infirmer le caractère intrusif d'une activité, ce qui est essentiel dans le domaine de la détection d'intrusions où les fausses alertes prédominent ;
- de réduire le nombre d'alertes non seulement en éliminant les fausses alertes, mais aussi en regroupant les alertes provoquées par une même activité intrusive ;
- d'améliorer la sémantique des alertes par la *reconnaissance* du phénomène et en étiquettant les groupes d'alertes par le diagnostic.

Les chroniques répondent donc bien aux objectifs que nous nous sommes fixés dans cette thèse. Comme déjà dit, nous n'adressons par le problème des faux négatifs. Dans [15], Cuppens et Miège évoquent la possibilité d'inférer des faits inobservés dans la reconnaissance de scénario d'intrusions. En ce sens, l'approche de Cuppens et Miège contribue d'une certaine manière à la réduction des faux négatifs. Remarquons que certains des faits *inobservés* produits par leur approche sont des faits *inobservables*, qui correspondent à des actions effectuées en dehors du domaine surveillé par les capteurs. Comme nous nous positionnons du point de vue du système attaqué, la totalité des événements impliqués dans des chroniques sont observables. Leur inobservation provient éventuellement d'une défaillance des capteurs, mais nous ne traitons pas ce type de problème ici. Nous partons du principe que toute activité intrusive censée être détectée par un capteur donne lieu à au moins une alerte.

6.2.2 Chroniques en tant que manager

Jusqu'ici nous avons présenté les chroniques comme une approche de corrélation d'alertes, c'est-à-dire implantée dans un *manager*. Nous rappelons qu'un manager est un composant de détection d'intrusion muni d'une interface manager, c'est-à-dire une interface d'entrée sur laquelle sont consommées des alertes produites par des analyseurs. Nous justifions ce choix dans cette section.

Il est tout à fait envisageable d'effectuer une analyse similaire aux chroniques non pas sur des alertes, mais directement sur des événements issus de capteurs. C'est d'ailleurs l'approche choisie par exemple par Pouzol et Ducassé [66, 67] ainsi que Goubault-Larrecq et Roger [71] qui analysent des événements système.

L'amélioration des techniques d'analyse pose des problèmes de performances : que la source de données soit réseau, système ou applicative, le surcoût engendré par une analyse élaborée des événements limite le volume d'événements traités.

Pour une sonde réseau, ceci signifie qu'un nombre important de paquets sont rejetés, or l'accroissement du débit des réseaux nécessite précisément que les sondes soient plus performantes. Les contributions sur le sujet à RAID 2003 [72, 47] montre ce besoin de performances des sondes réseau.

[2] C'est-à-dire que leur analyse ne s'applique qu'à un seul événement, un paquet réseau par exemple

Pour une sonde système, le surcoût de l'analyse se traduit par un ralentissement global du système et des besoins accrus en mémoire, qui peuvent devenir intolérables pour les utilisateurs du système.

Comme les sondes réseau, les sondes applicatives peuvent aussi être amenées à ne pas traiter des événements et/ou à ralentir l'application surveillée.

Dans des environnements où les performances des sondes deviennent un facteur critique, il est donc souhaitable de cantonner le rôle des sondes de détection d'intrusions à celui de *filtre* d'événements intrusifs. Comme les chroniques consomment des alertes, nous les positionnons au niveau d'un *manager*.

Cependant, comme nous allons le voir dans les sections 6.3.4 et 6.3.3, nous avons constaté en utilisant les chroniques que nous ne pouvons pas cantonner le flot d'entrée des chroniques à des *alertes*. Le flot d'entrée doit être agrémenté d'événements qui ne dénotent pas forcément une activité intrusive pour améliorer le contenu et la sémantique des alertes. Ce constat illustre une fois de plus que la frontière entre les fonctions de *corrélation* explicite et d'*analyse* morphosciente est ténue.

Notons qu'en tant que producteur d'alertes, le système de reconnaissance de chroniques constitue aussi un *analyseur*.

6.2.3 Attributs du domaine

Les données produites et consommées sur ces interfaces constituent des attributs du domaine modélisé. Nous les détaillons ici, en précisant le lien avec le modèle $\mathbf{M_2D^2}$.

a) Evénements consommés par les chroniques

L'attribut du domaine correspondant aux alertes est défini de la manière suivante dans la syntaxe des chroniques :

```
message alarm[sensor, serial, attack, attacker, victim]
```

où `sensor` est un élément de $\mathcal{O}$ (cf page 76), qui désigne l'outil de sécurité à l'origine de l'alerte. `serial` est un numéro de série d'alerte fourni par la sonde. Le couple (`sensor`,`serial`) constitue donc un identifiant unique pour l'alerte, indispensable pour désigner sans ambiguïté les alertes. `attack` est un élément de $\mathcal{S}$ (cf page 72) identifiant une signature d'attaque. `attacker` et `victim` sont des éléments de $\mathbb{N}_A$ identifiant respectivement l'adresse IP de l'hôte agissant en qualité d'attaquant et l'adresse IP de l'hôte agissant en qualité de victime (cf page 63).

Dans $\mathbf{M_2D^2}$, les alertes ne possèdent pas explicitement d'attribut `attacker` et `victim`. Ces valeurs sont obtenues par application sur les évènements IP ayant provoqué l'alerte, des fonctions **ip_src** ou **ip_dst** (cf page 4.11). Pour simplifier, nous considérons que ces associations sont effectuées en amont de la reconnaissance par les chroniques.

Comme nous l'avons dit, le flux d'entrée des chroniques n'est pas uniquement constitué d'alertes ; il est aussi constitué d'événements que nous souhaitons associer au diagnostic. Ces événements permettent d'infirmer ou de confirmer des alertes, d'améliorer ou de compléter leur contenu. Typiquement, ces informations peuvent être de nature cartographique. En effet, comme nous l'avons évoqué dans le chapitre 4, l'un des inconvénients de $\mathbf{M_2D^2}$ réside dans le caractère statique des concepts. Les chroniques

offrent un moyen de prendre en compte les aspects dynamiques de la topologie des systèmes d'informations surveillés. Ces événements précèdent ou suivent les alertes d'un délai assez court. C'est pour cette raison que nous qualifions ces événements de *périphériques*.

Les chroniques décrites dans les sections 6.3.2 et 6.3.4 utilisent des événements périphériques auxquels sont associés des attributs du domaine particuliers.

b) Événements produits par les chroniques

Lorsqu'une instance de chronique est reconnue, des actions sont entreprises par le système de reconnaissance. Dans le cadre de l'interaction des chroniques avec $\mathbf{M_2D^2}$, nous envisageons deux types d'actions :

1. la production d'une alerte de haut niveau correspondant à l'instance de chronique reconnue (i.e. un concept dans la terminologie $\mathbf{M_2D^2}$),
2. la production de relations entre les alertes de haut niveau et les événements qui participent à l'instance.

   Comme nous l'expliquons dans le chapitre 5, afin de réduire la quantité d'alertes sans pour autant dégrader les informations disponibles, nous proposons de ne soumettre à l'opérateur que les alertes de plus haut niveau, c'est-à-dire celles qui *englobent* les autres. Il lui est possible de naviguer dans l'ensemble des alertes via les liens entre les alertes pour obtenir des détails sur les événements à l'origine des alertes.

   Dans le cas des chroniques, les alertes produites sont liées aux alertes consommées par la relation **causes** de $\mathbf{M_2D^2}$ (cf 4.10). L'action d'une chronique reconnue peut donc consister en la production d'une relation **causes** entre les alertes qui participent à l'instance et une nouvelle alerte produite.

   Notons que les événements bénins utilisés dans les chroniques ne font pas partie des événements de $\mathbf{M_2D^2}$. Ils participent à la reconnaissance d'une chronique, mais n'apparaissent pas en tant qu'événements englobés par une alerte issue d'une chronique reconnue. Ils ne sont jamais soumis à l'opérateur. C'est d'ailleurs pour cette raison qu'il ne sont munis ni d'un identifiant de sonde, ni d'un numéro de série.

L'action qui consiste à produire une alerte correspondant à une instance de chronique reconnue est la suivante :

```
emit event(chroniclealarm[chro, serial, attack,
                          attacker, victim], t)
```

On considère que `chroniclealarm` est une classe obtenue par héritage du concept alerte de $\mathbf{M_2D^2}$. Dans le formalisme de $\mathbf{M_2D^2}$, les attributs de cette classe sont l'identité de l'attaquant (`attacker`) et l'identité de la victime (`victim`). L'identifiant d'attaque (`attack`) et la date d'occurrence sont respectivement des attributs de la classe alerte et événement de $\mathbf{M_2D^2}$. Le champ `chro` est l'identité du système de reconnaissance de chronique agissant en tant que producteur d'alertes. `serial` est le numéro de série affecté par le système de reconnaissance. Ce numéro de série est obtenu par appel d'une

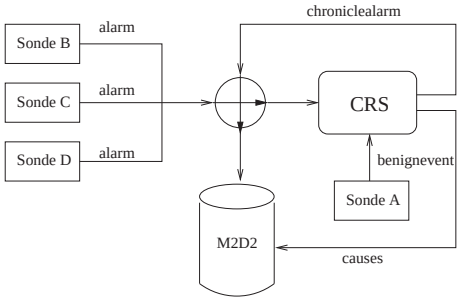

FIG. 6.5 – Interactions entre les chroniques, $\mathbf{M_2D^2}$ et les sondes

fonction `getid()`, qui retourne un numéro de série unique (cf exemples de la section suivante).

Notons que les alertes produites par le système de reconnaissance de chroniques peuvent être réutilisés en tant qu'événements consommables par ces mêmes chroniques.

L'action qui consiste à produire une relation **causes**(a_1, a_2) entre deux alertes a_1 et a_2 est la suivante :

```
store causes(sensor1, serial1, sensor2, serial2)
```

où le couple (`sensor1`,`serial1`) identifie l'alerte a_1 et le couple (`sensor2`,`serial2`) identifie l'alerte a_2.

Le schéma fonctionnel de la figure 6.5 résume les interactions entre les sondes, le système de reconnaissance des chroniques (CRS) et une implémentation de $\mathbf{M_2D^2}$. Les alertes émises par les sondes transitent par un répartiteur qui les transmet à un système de stockage d'alertes qui respecte le modèle $\mathbf{M_2D^2}$ ainsi qu'au système de reconnaissance de chroniques. Les événements bénins ne sont pas stockés dans $\mathbf{M_2D^2}$. Les alertes produites par CRS transitent par le répartiteur, si bien que ces alertes sont réutilisables par CRS.

6.2.4 Signification des chroniques reconnues

Jusqu'ici nous avons décrit l'objectif poursuivi, la nature des données consommées et des données produites. Dans cette section nous décrivons de manière informelle la signification de la reconnaissance d'une chronique. Pour cela, nous distinguons trois types de chroniques : les chroniques *positives*, *négatives* et *neutres*.

a) Chroniques *positives*

Les modèles de chroniques positives impliquent plusieurs alertes. Leur reconnaissance permet à la fois de réduire le nombre d'alertes soumises à l'opérateur et d'améliorer la sémantique. La réduction se fait en ne générant qu'une seule alerte à la place de la séquence initiale générée par les sondes. L'amélioration sémantique se fait en produisant un message explicite qui correspond exactement au phénomène reconnu.

A l'opposé, une chronique positive non reconnue signifie que les contraintes ne sont pas respectées (il manque des alertes par exemple), c'est-à-dire que le phénomène sous-jacent est différent de celui modélisé par la chronique. Dans ce cas, les alertes reçues sont soumises à l'opérateur individuellement ; elles ne sont pas englobées par l'alerte correspondant à la chronique. Comme nous l'avons dit, les événements périphériques éventuellement reçus, impliqués dans la chronique ne sont pas présentés à l'opérateur.

Les chroniques décrites en sections 6.3.1, 6.3.2 et 6.3.5 sont de ce type.

b) Chroniques *négatives*

Les problèmes des sondes de détection d'intrusions est parfois abusivement attribué à l'analyse mono-événementielle des sondes et à la simplicité excessive des signatures utilisées dans les analyseurs. En effet, l'étude des attaques montre que les éléments à la disposition des sondes pour discriminer une attaque d'une activité anodine sont parfois réduits à quelques caractéristiques peu significatives d'un seul événement (un paquet IP, par exemple). Dans ces conditions, des activités anodines peuvent facilement être confondues avec des attaques.

A titre d'illustration, le champ `content` de la signature Snort suivante contient la chaîne d'octets qui est caractéristique d'une attaque par dépassement de tampon. Cette chaîne est réduite à seulement 6 octets :

```
alert ip $EXTERNAL_NET any -> $HOME_NET $SHELLCODE_PORTS
  (msg:"SHELLCODE x86 stealth NOOP";
   content: "|eb 02 eb 02 eb 02|";
   reference:arachnids,291;
   classtype:shellcode-detect;
   sid:651; rev:5;)
```

Si dans ces situations il n'existe pas d'éléments supplémentaires permettant de discriminer une attaque d'une activité anodine, il est en revanche parfois possible d'utiliser des événements périphériques pour discriminer une activité anodine d'une attaque.

L'alerte correspondant à une chronique négative qui est reconnue porte le fait que les alertes que cette chronique englobe sont des faux positifs. Comme ces alertes sont englobées, elles ne sont pas soumises directement à l'opérateur.

Lorsqu'une instance de chronique négative n'est pas reconnue, le phénomène sous-jacent peut être réellement intrusif. L'alerte originale est alors directement soumise à l'opérateur, puisqu'elle n'est pas englobée dans une chronique reconnue.

La chronique décrite en section 6.3.3 est de ce type.

Notons qu'une chronique négative peut parfois être ramenée à une chronique positive en utilisant une contrainte sur la non occurrence (noevent) des événements périphériques utilisés pour désamorcer l'alerte.

c) Chroniques *neutres*

Les chroniques neutres complètent le contenu d'alertes avec des événements périphériques, sans qualifier la ou les alertes de vrai ou faux positif. La chronique décrite en section 6.3.4 est de ce type.

```
GET /scripts/root.exe?/c+dir
GET /MSADC/root.exe?/c+dir
GET /c/winnt/system32/cmd.exe?/c+dir
GET /d/winnt/system32/cmd.exe?/c+dir
GET /scripts/..%255c../winnt/system32/cmd.exe?/c+dir
GET /_vti_bin/..%255c../..%255c../..%255c../winnt/
                            system32/cmd.exe?/c+dir
GET /_mem_bin/..%255c../..%255c../..%255c../winnt/
                            system32/cmd.exe?/c+dir
GET /scripts/..%c1%1c../winnt/system32/cmd.exe?/c+dir
GET /scripts/..%c0%2f../winnt/system32/cmd.exe?/c+dir
GET /scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir
GET /scripts/..%c1%9c../winnt/system32/cmd.exe?/c+dir
GET /scripts/..%%35%63../winnt/system32/cmd.exe?/c+dir
GET /scripts/..%%35c../winnt/system32/cmd.exe?/c+dir
GET /scripts/..%25%35%63../winnt/system32/cmd.exe?/c+dir
GET /scripts/..%252f../winnt/system32/cmd.exe?/c+dir
```

FIG. 6.6 – Manifestation de Nimda

6.3 Exemples d'applications

Nous donnons ici des exemples d'application concrets de chroniques utilisées pour corréler des alertes.

6.3.1 Attaques virales

Dans cette section, nous illustrons l'utilisation des chroniques pour corréler des alertes correspondant à des attaques virales. Nous proposons d'abord une chronique pour la manifestation de chaque tentative d'attaque. Le caractère récurrent des alertes, inhérent à la détection de l'activité des vers, permet d'utiliser deux autres chroniques qui améliorent la chronique initiale. Nous les décrivons dans un second temps.

a) Manifestation du ver

Un ver est un programme informatique autonome qui exploite une faille dans un serveur applicatif, lui permettant d'executer des instructions illicites. Lorsqu'un ver infecte un serveur, il se réplique et tente d'infecter d'autres serveurs environnants. Leur activité est déterministe. Nous présentons un exemple d'utilisation sur le ver Nimda.

Le ver Nimda exploite une vulnérabilité des serveurs web IIS. Au cours d'une tentative d'infection, plusieurs dizaines de requêtes HTTP sont envoyés par la machine attaquante à la victime. La Figure 6.6 est un extrait de cette séquence de requêtes.

L'analyse de la séquence révèle des tentatives d'exécution de commandes (`cmd.exe`, `root.exe`). Chacune des requêtes, prise individuellement, est suspecte du point de vue

```
1  chronicle nimda[?attacker, ?victim]
2  {
3    occurs((1,2),alarm[?sensor, ?, iis_code_red_ii_root_exe,
4                       ?attacker,?victim], (t,t+2))
5    occurs((1,4),alarm[?sensor, ?, iis_decode_bug,
6                       ?attacker,?victim],(t,t+2))
7    occurs((1,14),alarm[?sensor, ?, iis_cmd_exe,
8                        ?attacker,?victim],(t, t+2))
9    occurs((1,3),alarm[?sensor, ?, web_dot_dot,
10                      ?attacker,?victim],(t,t+2))
11   occurs((1,2),alarm[?sensor, ?, iis_unicode,
12                      ?attacker,?victim],(t,t+2))
13   occurs((1,1),alarm[?sensor, ?, iis_unicode2,
14                      ?attacker,?victim],(t,t+2))
15   occurs((1,1),alarm[?sensor, ?, iis_unicode3,
16                      ?attacker,?victim],(t,t+2))
17   occurs((1,1),alarm[?sensor, ?, iis_decode_bug3,
18                      ?attacker,?victim],(t,t+2))
19   occurs((1,1),alarm[?sensor, ?, iis_decode_bug2,
20                      ?attacker,?victim],(t,t+2))
21   occurs((1,1),alarm[?sensor, ?, iis_decode_bug4,
22                      ?attacker,?victim],(t,t+2))
23
24   when recognized {
25      emit event(chroniclealarm[nimda_attempt, ?attacker,
26                                ?victim], t);
27   }
28 }
```

FIG. 6.7 – La chronique pour le ver Nimda

de la sécurité. Il est donc légitime de générer une alerte pour chacune d'elles. De fait, les IDS dont l'analyse est mono-événementielle génèrent autant d'alertes que de requêtes HTTP : en fonction des bases de signatures utilisées, de 15 à 30 alertes sont ainsi produites à chaque tentative d'infection. Cependant, lorsque la séquence de requêtes décrite en figure 6.6 est détectée, il est plus intéressant de générer une seule alerte synthétique. Cette alerte est plus intéressante non seulement du point de vue de la réduction du volume global d'alertes, mais aussi du point de vue de l'amélioration de la sémantique, puisqu'elle *identifie* explicitement le ver Nimda.

La chronique donnée en figure 6.7 permet de synthétiser les alertes issues des sondes. Les identifiants de signatures utilisés dans cette chronique correspondent à ceux de Dragon[3]. Les contraintes liées à la reconnaissance de cette chronique sont les suivantes :

– la victime (`?victim`) et l'attaquant (`?attacker`) doivent être communs à chaque

[3] `http://dragon.enterasys.com`

alerte,
- l'ensemble des alertes doit survenir dans un intervalle de deux secondes,
- l'ordre d'occurrence des alertes n'est pas important ; c'est la première alerte correspondant à l'un des motifs de la chronique qui fixe la valeur de `t`,
- en fonction des versions du ver Nimda, un nombre variable de chaque type de requêtes est constaté. Afin d'utiliser un seul modèle de chronique pour les différentes versions du ver, nous utilisons le prédicat `occurs`, qui compte le nombre d'occurrences. Par exemple, entre 1 et 14 occurrences de l'attaque `iis_cmd_exe` sont acceptées.

Lorsque ces contraintes sont respectées, une instance de chronique est reconnue. L'action associée à la chronique consiste à générer une nouvelle alerte, dont les paramètres `?attacker` et `?victim` correspondent à ceux des alertes individuelles et dont l'identifiant d'attaque est `nimda_attempt`.

Cette chronique permet de réduire le nombre d'alertes liées au ver Nimda d'un facteur 14 à 30 en fonction des IDS. L'impact sur le volume global d'alertes est variable ; il dépend de la virulence du ver Nimda. Selon nos propres observations, au plus fort du phénomène d'infection, les alertes Nimda représentaient 70 à 80% des alertes générées par les sondes de détection d'intrusions.

b) Amélioration de la chronique

La chronique précédente permet de reconnaître des tentatives d'infection par le ver Nimda et réduit le nombre d'alertes d'un facteur 14 à 30. Cependant, cette chronique est reconnue à chaque *tentative* d'attaque. Or, de part sa nature, un ver engendre un très grand nombre de tentatives d'attaques : d'une part, les machines infectées tentent périodiquement d'infecter leur voisines ; d'autre part, une machine déjà infectée reçoit toujours de nouvelles tentatives d'attaques.

Malgré la chronique précédente, un grand nombre d'alertes est donc toujours produit. Une amélioration consiste à ne générer qu'une alerte par *nouvelle* infection et à *masquer* les tentatives d'attaques des hôtes infectés derrière cette alerte.

Cette amélioration a en outre un deuxième avantage inhérent aux vers : un hôte qui attaque ses voisins est par définition une victime d'une attaque précédente. En ne générant qu'une alerte par hôte nouvellement infecté, c'est-à-dire ceux qui attaques leurs voisins alors qu'ils ne le faisaient pas avant, on limite la liste des alertes à celles qui correspondent à des infections réussies.

On bénéficie alors d'une réduction du nombre d'alertes et d'une amélioration sémantique notables.

La chronique de la figure 6.9 comptabilise les nouvelles infections. Un hôte `victim` est dit nouvellement infecté, si il est impliqué en tant qu'attaquant dans une tentative d'attaque `nimda_attempt` (lignes 4/5) *et* qu'il n'a pas été déjà comptabilisé *avant* cette tentative d'attaque (lignes 6/7). On peut remarquer en ligne 4 que la victime de la tentative d'attaque est uniquement symbolisée par un `?`, qui dénote une variable libre (ne devant pas être instanciée). En effet, la valeur de cette variable (l'identité de la victime potentielle) n'a pas d'importance. De toutes façons, si cette victime est vulnérable, elle

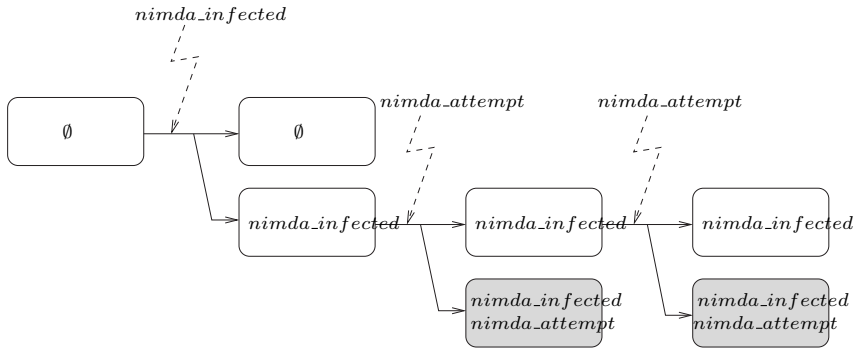

FIG. 6.8 – Duplication de la chronique part_of_infection

```
1  chronicle new_infection[?victim]
2  {
3    event(sysinit, t0)
4    event(chroniclealarm[?sensor, ?ida, nimda_attempt,
5                         ?victim, ?], t1)
6    noevent(chroniclealarm[chro, ?id_inf, nimda_infected,
7                           ?victim, ?], (t0,t1))
8
9    when recognized
10   {
11      emit event(chroniclealarm[chro, ?id=getid(), nimda_infected,
12                                ?victim, _], t1)
13      store causes(?sensor, ?ida, chro, ?id)
14   }
15 }
```

FIG. 6.9 – La chronique pour le ver Nimda

sera infectée, donc elle sera le sujet d'une autre instance de chronique `new_infection`. Le motif en ligne 3 est un événement envoyé à l'initialisation du système de reconnaissance pour définir l'origine du temps.

La chronique de la Figure 6.10 masque les tentatives d'attaques consécutives à l'infection d'un hôte. Cette chronique utilise l'exhaustivité de la reconnaissance des chroniques (voir 6.1.2). Elle est instanciée lorsqu'une alerte `nimda_infected` est reçue, suite à la reconnaissance de la chronique précédente (lignes 3/4). Chaque tentative d'attaque par l'hôte infecté est intégré à cette chronique et provoque sa reconnaissance (lignes 5/6). Du fait de l'exhaustivité de la reconnaissance, avant l'intégration d'un événement `nimda_attempt` la chronique originale est dupliquée et intégrera les tentatives ultérieures. Ce mécanisme est schématisé en figure 6.8. Les instances de chroniques sont représentées par des cadres qui contiennent les événements déjà intégrés de l'instance. L'occurrence d'événement est symbolisée par une flèche discontinue surmontée

```
1  chronicle part_of_infection
2  {
3    event(chroniclealarm[?chro, ?id_inf, nimda_infected,
4                         ?victim, ?], t0)
5    event(chroniclealarm[?sensor, ?id_al, nimda_attempt,
6                         ?victim, ?], t1)
7    when recognized
8    {
9       store causes(?sensor, ?id_al, chro, id_inf)
10   }
11 }
```

FIG. 6.10 – Chronique `part_of_infection`

du nom de l'événement. Les instances reconnues sont grisées.

Lorsque la chronique `part_of_infection` est reconnue, la relation qui lie le fait que la victime est infectée à son symptôme (*i.e.* la tentative d'attaque) est enregistrée (ligne 9). Seul le fait que la victime est infectée est soumis à l'opérateur, l'ensemble des tentatives d'attaques de la victime sont masquées par ce fait.

La figure 6.11 schématise les relations entre alertes établies lors de la reconnaissance d'une attaque Nimda. Les cadres contiennent les attributs (attaque, attaquant, victime) des alertes. Tous les symptômes d'une tentative d'attaque Nimda ne sont pas représentés car il sont trop nombreux. Les liens entre les cadres dénotent une relation **causes**. L'identité de la chronique à l'origine des liens est indiquée par les flèches en pointillés. Seule l'alerte *nimda_infected* est présentée à l'opérateur (symbolisé par un œil, au sommet de la figure).

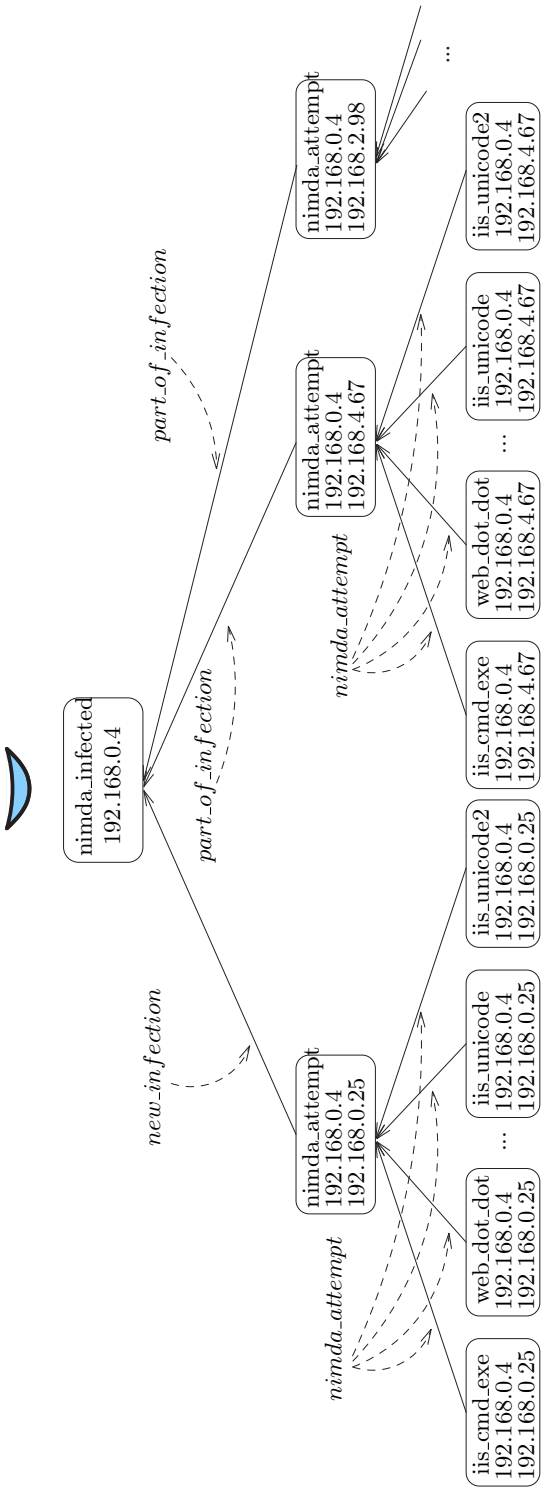

FIG. 6.11 – Relations entre les alertes engendrées par le ver Nimda

```
1  chronicle portscan[?attacker, ?victim]
2  {
3     event(alarm[?sensor, ?, portscan_begin,
4                 ?attacker, ?victim], t1)
5     occurs((1,+oo),alarm[?sensor, ?, portscan_status,
6                         ?attacker, ?victim], (t1+1,t2))
7     noevent(alarm[?sensor, ?, portscan_end,
8                   ?attacker, ?victim], (t1,t2))
9     event(alarm[?sensor, ?, portscan_end,
10                ?attacker, ?victim], t2)
11    t1<t2
12
13    when recognized {
14      emit event(chroniclealarm[portscan, ?attacker,
15                                ?victim], t2);
16    }
17 }
```

FIG. 6.12 – Chronique portscan

On notera que la chronique `part_of_infection` utilise comme motif d'événement des alertes issues de la reconnaissance des chroniques `new_infection` et `nimda_attempt` et que `new_infection` utilise la chronique `nimda_attempt`. Ceci illustre la dualité des chroniques en tant que manager et analyseur dans le processus de traitement des alertes : ils consomment des alertes issues de plusieurs analyseurs, mais aussi des alertes qu'ils produisent eux-mêmes.

6.3.2 Balayage de ports

Le ver Nimda présenté à la section précédente est un exemple de phénomène persistant, au cours duquel les symptômes se manifestent de manière quasi-continue. Cela engendre un volume important d'alertes car les sondes ont tendance à *segmenter* ces phénomènes continus, c'est-à-dire générer périodiquement des alertes identiques. Le flot d'alertes soumises à l'opérateur est ainsi pollué par des alertes récurrentes tant que la cause du phénomène n'est pas traitée (*e.g* application d'un correctif dans le cas des attaques virales). Un autre exemple de phénomène persistant est donné par les attaques par balayage de ports.

Un balayage de ports consiste à acquérir des informations sur une cible en initiant des connexions sur un grand nombre de ports de la victime, afin de découvrir quels services sont disponibles. Les balayages de ports ne sont pas des attaques à proprement parler, mais ils constituent souvent les prémices d'une intrusion. On constate aussi souvent qu'ils sont l'objet de faux positifs.

Dans le cas de Snort, un balayage de port se caractérise par les trois identifiants d'attaques suivants :

- un `portscan_begin` qui désigne le début d'un balayage ;
- plusieurs `portscan_status` qui informent l'opérateur que le balayage est en cours ;
- un `portscan_end` indiquant la fin supposée du balayage.

La fin du balayage survient quand aucune tentative de connexion n'est détectée pendant un laps de temps suffisamment long. Dans le cas de fausses alertes, on constate en fait que la séquence des trois types d'alertes se répète régulièrement.

La chronique représentée en figure 6.12 permet de synthétiser les alertes engendrées par une attaque par balayage de port. L'utilisation des variables `?victim` et `?attacker` dans tous les motifs de la chronique traduit implicitement le fait que la victime et l'attaquant sont les mêmes au cours d'un balayage de port. Un balayage de port est encadré par l'occurrence des événements `portscan_begin` (lignes 3/4) et `portscan_end` (ligne 9/10). Ce sont ces deux événements qui instancient les bornes inférieure (`t1`) et supérieure (`t2`) de la chronique. Entre ces deux instants, une nombre quelconque d'événements `portscan_status` peuvent survenir (ligne 5/6).

On notera en lignes 7/8 le motif qui oblige à ce qu'il n'y ait pas plusieurs alertes `portscan_end` entre `t1` et `t2`. Sans cette contrainte, une instance de chronique dont l'événement `portscan_begin` et un certain nombre de `portscan_status` sont instanciés serait reconnue autant de fois qu'il y a d'occurrences ultérieures de `portscan_end` du fait de l'exhaustivité de la reconnaissance des chroniques. Comme nous l'avons dit, les balayages de ports sont des phénomènes récurrents et les trois étapes reviennent régulièrement. En d'autres termes, cette contrainte dit que le `portscan_end` qui clôt un balayage de port est le *premier* qui survient après un `portscan_begin`. On voit que dans ce cas, l'exhaustivité de la reconnaissance impose au rédacteur de chroniques de prendre ce genre de précaution pour éviter que des instances de chroniques ne soient maintenues en mémoire inutilement.

Plus généralement, comme le précise Dousson (page 75 de sa thèse), les modèles de chroniques sont souvent limités dans le temps par une durée bornée entre le premier événement et le dernier. Ceci garantit que les instances ne peuvent vivre indéfiniment et que des reconnaissances ne se recouvrent. Cependant, en détection d'intrusions, il n'est pas toujours possible de spécifier de telles bornes temporelles. Dans ce cas, il est possible d'ajouter des assertions dans la chronique afin que des instances ne se multiplient pas au fil des reconnaissances. Une durée limite pour détruire des chroniques est superflue. C'est le cas de l'assertion `noevent` utilisée en lignes 7/8 de cette chronique.

La présence nécessaire de cette assertion alourdit la lecture des chroniques. Toutefois, notons que l'exhaustivité de la reconnaissance (qui nécessite cette assertion) peut être utilisée comme une fonctionnalité du système de reconnaissance, comme nous avons pu le voir pour la chronique du ver Nimda (Figure 6.9). Dans tous les cas, l'auteur de chroniques doit être conscient de ce type d'écueil lors de la rédaction de chroniques.

Quand la chronique `portscan` est reconnue, une alerte synthétique est générée et l'instant choisi pour cette alerte correspond à la fin du balayage de port (`t2`).

```
1  chronicle shellcode_mitigation[?attacker, ?victim]
2  {
3     event(periphevent[ftp_retr_request,
4                       ?attacker, ?victim], t1)
5     event(alarm[?sensor, ?aid, shellcode,
6                 ?attacker, ?victim], t2)
7     noevent(periphevent[ftp_transfer_complete,
8                         ?victim, ?attacker], (t1+1,t3-1))
9     event(periphevent[ftp_transfer_complete,
10                      ?victim, ?attacker], t3)
11
12    t1<t2<t3
13
14    when recognized {
15      emit event(alarm[chro, ?id=getid(), shellcode_mitigation,
16                       ?attacker,?victim], t2);
17      store causes(?sensor, ?aid, chro, ?id)
18    }
19 }
```

FIG. 6.13 – Chronique shellcode_mitigation

6.3.3 Elimination de fausses alertes

Les chroniques précédentes sont des chroniques positives et les motifs d'événements utilisés dénotent exclusivement des activités intrusives. Dans cette partie, nous étudions une chronique négative (*i.e.* sa reconnaissance est indicative d'un faux positif) qui exploite des motifs d'événements bénins.

La médiocrité des alertes émises par les sondes est parfois liée à l'analyse mono-événementielle des sondes et à la pauvreté des signatures. Toutefois, un certain nombre d'attaques ne sont détectables que par quelques propriétés faiblement discriminantes, si bien que des activités légitimes peuvent être confondues avec des attaques. Dans cette situation, si aucun autre élément n'est disponible pour confirmer le caractère intrusif d'une activité, il peut en revanche exister des éléments périphériques qui sont caractéristiques d'une activité légitime et permettent ainsi d'infirmer une alerte. L'objectif de cette chronique est de prendre en compte ces événements pour discriminer les faux positifs des vrais positifs.

Des attaques par débordement de zone mémoire sont régulièrement détectées par une sonde Dragon déployée dans le réseau de France Télécom. L'analyse des alertes a révélé qu'il s'agissait de faux positifs. En effet, les attaques par dépassement de zone mémoire consistent à exploiter une absence de validation de la taille d'une zone mémoire utilisée par une application pour y insérer des commandes illicites afin qu'elles soient exécutées par l'application. Les commandes `NOP`[4] sont fréquemment utilisées pour rem-

[4]Instruction indiquant au processeur de ne rien faire

plir les zone mémoire. Le code hexadécimal de l'instruction `NOP` dans les architectures x86 est `0x90`. Cette propriété des attaques par dépassement de tampon est utilisée dans les bases de signatures pour détecter ces attaques de manière générique, c'est-à-dire indépendante de l'application visée : si une séquence discontinue d'octets `0x90` est détectée dans une interaction avec une application, une alerte est émise.

Malheureusement, certaines interactions avec des services applicatifs impliquent de telles chaînes d'octets et donnent lieu à des fausses alertes. En particulier, le transferts d'images lors d'interactions avec un serveur FTP provoquent des fausses alertes parce que le format JPEG inclut des chaînes d'octets `0x90`.

La désactivation d'une telle signature n'est pas une solution car de vraies attaques peuvent être manquées. Nous proposons donc une chronique qui est reconnue lorsque des événements indicatifs d'un transfert de fichier encadrent une alerte `shellcode`, auquel cas l'alerte est qualifiée de faux positif. Cette chronique est schématisée en Figure 6.13.

La chronique `shellcode_mitigation` est reconnue lorsqu'une occurrence d'alerte `shellcode` est encadrée par une alerte dénotant une requête de transfert de fichier (`ftp_retr_request`, lignes 3/4) et une alerte dénotant la fin de transfert (`ftp_transfer_complete`, lignes 9/10). On utilise dans le cas présent un attribut du domaine `periph_event` ayant trois attribut : le premier désigne la nature de l'événement, le second l'adresse IP de l'attaquant et le troisième l'adresse IP de la victime.

Comme la chronique `portscan`, c'est la première occurrence de `ftp_transfer_complete` qui clôt la reconnaissance de la chronique, ce qui explique la présence de la contrainte `noevent` en lignes 7/8, pour invalider toutes les instances précédentes.

On notera que lors de transferts normaux (*i.e.* lorsqu'aucune alerte ne survient entre le début et la fin du transfert), le système de reconnaissance sait qu'une instance de chronique ne peut plus être reconnue dès lors qu'un événement `ftp_transfer_complete` est reçu, car la contrainte `t1<t2<t3` ne peut plus être satisfaite.

Ce type de chronique nécessite de configurer des sondes pour que les événements périphériques fassent l'objet d'une alerte. Les sondes de détection d'intrusions orientées signatures, éventuellement les mêmes que celles qui détectent les attaques, peuvent être utilisées à cet effet. La signature permettant de détecter la requête de transfert utilisera le motif "`RETR`" du protocole FTP ; la signature permettant de détecter ls fin du transfert utilisera le motif "`226`", qui est le code de retour indiquant la fin d'un transfert[5].

Bien entendu, la victime et l'attaquant apparaissant dans les motifs de la chronique sont les mêmes. On notera simplement que la victime et l'attaquant ont été inversé dans le motif de l'événement `ftp_transfer_complete` car la fin du transfert est envoyée du serveur (victime) au client (attaquant).

Au lieu de reconnaître une chronique dans le cas d'une fausse alerte, il aurait été envisageable de reconnaître la chronique seulement lorsque l'alerte `shellcode` *n'est pas* encadrée par des événements de début et de fin de transfert. La chronique n'aurait été

[5]Voir RFC 959, `http://www.ietf.org/rfc/rfc0959.txt`

```
1  chronicle ddns[?attacker, ?victim]
2  {
3     event(hasip[?name]:(?,?victim), t1)
4     hold(hasip[?name]:?victim, (t1,t2))
5     event(alarm[?sid, ?id, ?attack, ?attacker,
6                            ?victim], t2)
7
8     when recognized {
9       emit event(chroniclealarm[?sid, ?id2 = getid(), ?attack,
10                        ?attacker, ?name], t2);
11      store causes(?sid, ?id, ?sid, ?id2)
12    }
13 }
```

FIG. 6.14 – Chronique ddns

reconnue qu'en cas d'attaque avérée. N'oublions pas que nous souhaitons *qualifier* les alertes, c'est-à-dire leur ajouter du contenu ou l'améliorer. De notre point du vue, le traitement (*e.g.* suppression) appliqué aux alertes qualifiées de faux positifs, ne doit pas être appliqué au niveau des chroniques, mais à un niveau supérieur, éventuellement sur décision de l'opérateur de sécurité. Dans le cas de cette chronique, l'alerte `shellcode` est englobée dans une chronique `shellcode_mitigation` qui la qualifie de faux positif. Si une alerte est un vrai positif, alors la chronique n'est pas reconnue et l'alerte `shellcode` est soumise individuellement à l'opérateur.

Lorsque la chronique `shellcode_mitigation` est reconnue, une nouvelle alerte est générée (lignes 15/16), dont l'identifiant est `shellcode_mitigation`, l'attaquant et la victime sont les mêmes que les événements de la chronique. Sa date d'occurrence est `t2` car c'est bien à cette date que la fausse alerte est survenue. Son identifiant est retourné par la fonction `getid()` et le nom de la sonde est `chro`. La relation qui lie cette nouvelle alerte à la fausse alerte est enregistrée (ligne 17).

6.3.4 Prise en compte d'informations topologiques

La chronique présentée dans cette section ne permet pas de juger de la gravité d'une alerte (comme dans les exemples des sections 6.3.1 ou 6.3.3), ni de réduire le nombre d'alertes (comme dans 6.3.2), mais elle illustre de quelle manière la qualité des informations contenues dans les alertes peut être améliorée en prenant en compte les information topologiques dynamiques.

La pérennité des adresses IP en tant qu'identifiants de victimes est de plus en plus limitée. En effet, le protocole DHCP (*Dynamic Host Configuration Protocol*) permet d'affecter dynamiquement une adresse IP aux hôtes qui en font la demande. L'adresse d'un hôte peut alors changer au cours du temps. Lors de leur requête, les hôtes fournissent au serveur DHCP leur nom, afin que l'association entre ce nom et l'adresse IP qui leur est fournie soit automatiquement prise en compte au niveau des serveurs de

```
1  chronicle successful_shellExec[?source, ?target]
2  {
3     event(alarm[?snort, ?id2, ?comm_exec_alarm,
4                          ?source, ?target], t1);
5     event(systemalarm[?syssensor, ?id3, shell_exec,
6                       ?user, ?target], t2);
7     noevent(systemalarm[?syssensor, ?, login, ?user,
8                          ?target], (t1,t2));
9
10    ?comm_exec_alarm in {sid_1751, sid_1894}
11
12    t2 - t1 in [0,3]
13
14    when recognized {
15      emit event(alarm[chro, ?id=getid(), successful_comm_exec,
16                       ?source, ?target], t2);
17      store causes(chro, ?id, ?snort, ?id2)
18      store causes(chro, ?id, ?syssensor, ?id3)
19    }
20 }
```

FIG. 6.15 – Chronique successful_shellExec

noms.

La durée de validité des noms d'hôtes en tant qu'identifiant d'hôte est plus longue que les adresses IP. Nous proposons une chronique qui capte les changements d'association dynamiques entre noms et adresses afin de substituer l'identifiant de la victime par son nom.

Nous utilisons un événement périphérique qui prend en compte les variations d'associations IP/nom. Il est modélisé par l'attribut du domaine `hasip` (voir Figure 6.14). Son unique argument est un nom d'hôte et sa valeur est l'adresse IP qui est affectée à cet hôte. Une instance de la chronique est créée pour chaque hôte `?hote` dont un changement d'adresse est détecté (ligne 3). Lorsque le système reçoit une alerte impliquant l'adresse de `?hote` en qualité de victime (lignes 5/6), la chronique est reconnue et cette adresse est simplement remplacée par la valeur de `?hote` (lignes 9 à 11).

On notera que pour invalider la chronique quand un nouveau changement d'adresse survient, on a recours à une assertion (ligne 4) qui contraint l'attribut du domaine `hasip` à ne pas changer de valeur entre son initialisation à `t1` et la réception de l'alerte à `t2`.

6.3.5 Coopération de sondes

Les chroniques proposées dans les sections précédentes consomment des alertes générées par la même sonde. Nous proposons maintenant un exemple de chronique

permettant de faire coopérer des sondes hétérogènes.

La chronique schématisée en Figure 6.15 permet d'associer deux alertes, l'une issue d'une sonde réseau et l'autre issue d'une sonde système. La première indique l'occurrence d'une attaque permettant l'exécution illicite de commandes (lignes 3/4) et la seconde indique l'exécution d'une ligne de commande (*shell*) (ligne 5/6). Les occurrences concomitantes de ces deux événements peuvent confirmer le succès d'une attaque.

On utilise un attribut du domaine spécifique pour les alertes issues de sondes systèmes. Le premier paramètre d'un événement de type `systemalarm` est un identifiant d'attaque (ou plus généralement d'action, en l'occurrence l'exécution d'un shell), le second est l'identité de l'utilisateur à l'origine de l'exécution du *shell* et le dernier est l'adresse de l'hôte sur lequel l'action est effectuée.

La ligne 10 permet de restreindre les valeurs acceptables pour une variable. Ici, les valeurs de la variable `comm_exec_alarm` sont restreintes à quelques identifiants d'attaques (Snort) permettant l'exécution illicite de commandes.

La ligne 12 contraint les deux événements à ne pas être séparés de plus de 3 secondes. L'attaque et l'exécution du shell sont théoriquement quasi-simultanés, mais le délai de trois secondes permet de résister aux décalages inévitables entre les horloges, même en environnement synchronisé. De plus, cette contrainte borne temporellement les instances de la chronique, ce qui a pour effet d'éviter de conserver indéfiniment des instances en mémoire.

Cet exemple illustre une caractéristique intéressante des chroniques. Les chroniques considèrent que les événements peuvent ne pas être reçus dès leur occurrence. Le retard possible d'un événement peut être provoqué par des délais de transmission ou des temps de traitement par les sondes situées en amont de la reconnaissance des chroniques. Comme les retards sont variables et mal connus, la réception des événements ne se fait pas dans l'ordre chronologique de leur occurrence. Comme le système de reconnaissance est principalement basé sur les dates d'occurrence, il fait l'hypothèse que les événements sont reçus dans le désordre, mais sont correctement estampillés par leur date d'occurrence. Ceci nécessite que les horloges des sondes soient synchronisées et que les différences entre les horloges soient négligeables par rapport aux délais utilisés dans les modèles de chroniques. Le traitement des événements dans une instance garantit la reconnaissance indépendamment de la chronologie de traitement.

La connexion d'un utilisateur sous une identité identique à celle du propriétaire du shell permettrait de justifier l'exécution de ce shell et ainsi invalider le succès de l'attaque. La ligne 7 permet d'éliminer cette hypothèse : la chronique ne serait pas reconnue si la connexion (`login`) d'un utilisateur `user` survient entre la détection de l'attaque et l'exécution du shell. Cette information est fournie par la même sonde système.

Lorsque la chronique est reconnue, une nouvelle alerte englobant les deux précédent est générée, indiquant qu'une attaque probablement réussie est survenue à l'instant `t2`.

6.4 Conclusion

Les chroniques sont un outil de modélisation de systèmes dynamique proposé par Dousson [25]. Elles bénéficient de fondements théoriques solides et d'une implémentation efficace.

L'application des chroniques à la corrélation d'alertes en détection d'intrusions constitue une approche de corrélation explicite. Les scénarios modélisent des phénomènes du point de vue du système surveillé pour deux raisons principales. Tout d'abord, nous pensons que les stratégies d'intrusions ne sont pas déterministes, contrairement à des phénomènes de pannes par exemple. Qui plus est, les attaquants peuvent mettre en œuvre des stratégies pour contourner les mécanismes de détection. Ensuite, les sondes ne sont pas fiables et les fausses alertes qu'elles produisent conduiraient le système de reconnaissance de chronique à construire des scénarios d'attaques fondés sur des fausses alertes. Les événements traités par l'outil de reconnaissance de chroniques sont soit des alertes, soit des événements périphériques bénins, qui permettent d'invalider ou au contraire de confirmer une alerte.

Nous avons donné des exemples d'application des chroniques à des cas concrets rencontrés en milieu opérationnel. Les chroniques développées ont été testées sur des fichiers d'alertes produites par des sondes Dragon et Snort et les performances sont tout à fait satisfaisantes. Comme nous l'avons indiqué en introduction, les chroniques ont initialement été conçues pour gérer des alertes issues de capteurs chargés de détecter des pannes dans un réseau de télécommunications. Dans un tel contexte, l'outil de reconnaissance doit être très efficace car il est confronté à des rafales de plusieurs dizaines d'alertes par secondes. Les rafales d'alertes constatées dans le domaine de la détection d'intrusions sont au plus du même ordre de grandeur que dans le domaine de la gestion de pannes, où les chroniques ont déjà été appliquées avec succès.

Les performances du système dépendent du nombre de modèles de chroniques utilisés. Le temps requis pour intégrer un événement croit linéairement avec le nombre de modèles de chroniques. Dans [25], les expérimentations de Dousson sont effectuées avec 80 modèles de chroniques contenant approximativement 10 motifs d'événements chacune parmi 50 attributs du domaine, 20 contraintes temporelles et 4 assertions. Dans cette configuration, l'intégration d'un événement nécessite en moyenne 10 ms. Notons que ces expérimentations ont été effectuées sur du matériel datant de 1994. Des détails sur les expérimentations sont disponibles dans [25] (pages 78–81) et dans [26].

D'une manière générale, comme dans toute approche par scénario, l'impact des chroniques sur la réduction du volume global d'alertes dépend de la complétude de la bibliothèque de scénarios. Cette mesure n'est donc pas représentative des performances des chroniques. L'un des objectif de cette thèse n'est pas de développer une bibliothèque de chroniques pour la corrélation d'alertes, mais plutôt d'illustrer les apports de ce paradigme pour la corrélation d'alertes. De même, l'apport des chroniques pour l'amélioration de la sémantique des alertes est purement subjectif et ne peut donner lieu à des mesures.

L'utilisateur des chronique est soit amené à développer lui même ses propres chroniques, soit à utiliser des chroniques déjà conçues, disponibles dans une bibliothèque.

Dans le cas où l'opérateur développe lui même ses chroniques, le temps de prise en main des chroniques n'est pas négligeable. Les mécanismes d'invalidation de chroniques par ajout d'assertion ou par contraintes temporelles ne sont pas toujours intuitifs et nécessitent une validation des modèles de chroniques sur des journaux d'alertes-tests.

Au temps de développement et de validation des chroniques s'ajoute le temps d'analyse des phénomènes qui permet de mettre en évidence les séquences d'alertes observées, voire même *observables*, si des événements périphériques sont nécessaires.

Le processus de rédaction de chroniques peut ainsi s'avérer long. A titre indicatif, le développement de la chronique 6.7 nécessite au total environ une demi-heure de développement, alors que les chroniques 6.9 et 6.10 peuvent nécessiter une journée de développement à un utilisateur inexpérimenté.

Pour réduire le temps de développement des chroniques, nous envisageons d'ajouter des fonctionnalités au compilateur de chroniques existant de manière à faciliter le travail de l'opérateur. L'invalidation de chroniques par ajout d'assertions peut par exemple faire partie de ces fonctionnalités. Les propositions de Pouzol et Ducassé [67] peuvent être utilisées à cet effet.

Pour réduire le temps passé à mettre en évidence les phénomènes récurrents, nous envisageons de tester un outil, FACE, conçu par l'équipe de développement des chroniques. FACE permet de découvrir des chroniques fréquentes dans des journaux d'alertes. La fréquence d'un phénomène dépend d'un seuil fixé par l'utilisateur. L'outil n'a pas pu être testé sur des bases d'alertes car jusqu'à une version récente de l'outil, seules des chroniques sans variables pouvaient être découvertes, or certains phénomènes sont fréquents seulement à condition de faire abstraction de certains paramètres des domaines d'attributs. C'est le cas de Nimda par exemple : si l'on ne prend en compte que l'identifiant de signature (*e.g.* `iis_decode_bug4`, `iis_cmd_exe`, etc.), alors la séquence d'alertes est fréquente. En revanche, si l'on prend aussi en considération l'adresse de la victime, il se peut que la tentative d'attaque Nimda contre la victime n'apparaisse qu'une seule fois, auquel cas la chronique n'est pas jugée fréquente.

Cette approche de corrélation à base de chroniques a fait l'objet d'une publication à la conférence conférence RAID 2003 [61] et a reçu un accueil favorable par les membres de la communauté lors de sa présentation.

Chapitre 7

Conclusion

Les outils de détection d'intrusions permettent de pallier les insuffisances des mécanismes de prévention d'intrusions. Ils détectent les actions malveillantes visant à remettre en cause l'intégrité, la disponibilité et/ou la confidentialité des ressources d'un système d'informations. Malheureusement, les outils de détection d'intrusions présentent plusieurs problèmes. Les travaux présentés dans cette thèse portent sur la résolution, par corrélation d'alertes, de certains de ces problèmes : réduction du volume global d'alertes et amélioration de la sémantique de ces alertes.

Nous résumons ici les contributions de cette thèse, avant d'évoquer les perspectives qu'ouvrent nos travaux.

7.1 Contributions

Nous avons proposé dans le chapitre 2 une classification des problèmes que présentent les outils de détection. Cette classification est composée de trois catégories principales : le volume excessif d'alertes, leur pauvreté sémantique et les faux négatifs. Il s'agit là de la première contribution de cette thèse.

L'étude de l'état de l'art a mis en évidence la nécessité de prendre en compte des informations supplémentaires pour pouvoir corréler les alertes. Notre deuxième contribution est donc un modèle formel, baptisé $\mathbf{M_2D^2}$, qui fédère ces informations. $\mathbf{M_2D^2}$ constitue une infrastructure à partir de laquelle des approches de corrélation peuvent extraire des informations pour corréler les alertes.

Les informations modélisées dans $\mathbf{M_2D^2}$ sont divisées en quatre catégories :

- les caractéristiques du système d'informations surveillé,
- les vulnérabilités présentes sur les entités du système d'informations et les propriétés des attaques qui les exploitent,
- les outils de sécurité déployés dans le système d'informations, chargés de prévenir ou détecter les attaques,
- les alertes produites et les événements consommés par les outils de sécurité.

Le modèle $\mathbf{M_2D^2}$ a donné lieu à une publication à la conférence RAID 2002 (*Recent Advances in Intrusion Detection*). Une maquette implémentant la structure de $\mathbf{M_2D^2}$ a été réalisée à l'aide d'une base de données relationnelle et d'un moteur Prolog.

Nos troisième et quatrième contributions portent sur deux approches de corrélation d'alertes qui exploitent les informations de $\mathbf{M_2D^2}$ et permettent de traiter les alertes.

Nous avons proposé une technique de corrélation d'alertes explicite, basée sur le formalisme des chroniques. Les chroniques ont initialement été proposées par Dousson [25] pour le diagnostic de pannes dans le contexte de la supervision de réseaux de télécommunications. Ce formalisme bénéficie de fondements théoriques solides et d'une implémentation efficace. Les alertes sont traitées en ligne ; elles sont confrontées à un ensemble de scénarios, constitués de motifs d'alertes liés entre eux par des contraintes temporelles. La reconnaissance d'un scénario donne lieu à la production d'une alerte synthétique, qui permet ainsi d'améliorer la sémantique des alertes. Le caractère récurrent des alertes permet de réduire fortement le volume global d'alertes à l'aide d'un nombre modeste de scénarios. Afin de réduire encore le volume d'alertes, les scénarios que nous proposons impliquent aussi des événements périphériques, qui permettent de discriminer des vrais positifs de faux positifs. L'approche a été validée sur des journaux d'alertes enregistrés en milieu opérationnel et a donné des résultats satisfaisants. Cette contribution a été publiée et présentée à la conférence RAID 2003.

Dans notre quatrième contribution, nous abordons la corrélation d'alertes sous l'angle de la recherche d'information. Les alertes sont stockées dans une structure permettant de combiner l'interrogation et la navigation dans un ensemble d'alertes. L'objectif est de permettre à l'opérateur de sécurité de manipuler les alertes de manière flexible. Cette approche diffère de la précédente en ce qu'elle ne produit pas d'alertes. En outre, elle constitue une technique de corrélation implicite car elle vise à mettre en évidence des regroupements potentiellement intéressants d'alertes, sans schéma de corrélation prédéfini.

Notre approche est basée sur l'analyse de concepts logique, proposée par Ferré et Ridoux [30]. L'analyse de concepts logique est un paradigme permettant de combiner l'interrogation et de la navigation dans des ensembles de données. Afin d'appliquer l'analyse de concepts logique aux alertes, nous avons proposé une logique des alertes simple. Le vocabulaire du langage est constitué de termes issus de $\mathbf{M_2D^2}$ qui permettent de décrire les groupes d'alertes de manière plus ou moins abstraite.

Une implémentation consiste à stocker les alertes dans le système de fichier logique proposé par Padioleau [64], qui remplit les conditions du langage de description des alertes. Ainsi, l'opérateur peut interroger le système de gestion d'alertes et naviguer à l'aide des commandes conventionnelles d'un système de fichier.

Les premiers résultats expérimentaux ont mis en évidence l'intérêt de l'approche pour manipuler les alertes. Cependant, l'implémentation actuelle présente des limitations dues au temps de stockage des alertes. Il reste donc nécessaire d'étudier une implémentation différente, permettant d'optimiser le temps de traitement des alertes lors de leur stockage.

7.2 Perspectives

Les travaux effectués au cours de la thèse offrent plusieurs perspectives. Ils se situent dans la continuité des travaux effectués. Nous les détaillons dans cette section.

Nous avons vu que la corrélation d'alertes nécessite la prise en compte d'informations cartographiques. Elles permettent d'une part d'évaluer la vulnérabilité d'un hôte vis-à-vis d'une attaque, mais aussi de modéliser des phénomènes réputés pour engendrer des faux positifs. Ces phénomènes impliquent des hôtes du systèmes d'informations qui, du fait de leur fonction, se trouvent impliqués en tant que victime ou attaquant dans des alertes.

Dans le cadre de nos travaux futurs, nous souhaitons travailler à court terme sur l'acquisition des informations cartographiques. A plus long terme, nous souhaitons étudier la modélisation et le traitement des phénomènes qui engendrent des faux positifs. Nous pensons qu'il permettraient de réduire considérablement le volume d'alertes. Nous détaillons ici brièvement ces axes de recherche.

7.2.1 Acquisition passive des informations cartographiques

Dans le chapitre consacré à $\mathbf{M_2D^2}$, nous avons décrit deux techniques d'acquisition des données cartographiques. La première repose sur une plate-forme dédiée qui est difficile à administrer dans des réseaux comportant un grand nombre d'hôtes, surtout lorsque les opérateurs n'ont pas une maîtrise totale des équipements du réseaux. La seconde consiste à interroger les hôtes à distance et inférer leur configuration par des heuristiques. Cette dernière approche présente l'inconvénient d'engendrer un trafic réseau important si les interrogations sont effectuées fréquemment (ce qui est nécessaire pour maintenir à jour les informations).

Nous souhaitons mettre en œuvre une variante *passive* de cette technique d'acquisition des informations cartographiques. Elle consiste à induire la configuration des hôtes du système d'informations en analysant les transactions entre les clients et les serveurs. Cette approche est dite passive car les informations sont obtenues uniquement par écoute du trafic réseau. Ainsi, l'acquisition n'engendre aucun trafic et la mise à jour des informations se fait en continu.

7.2.2 Phénomènes engendrant des faux positifs

Dans le chapitre consacré à l'analyse conceptuelle des alertes, nous avons proposé de modéliser les phénomènes réputés pour engendrer des faux positifs sous la forme de propriétés déduites des attributs des alertes. Nous souhaitons approfondir cet axe de recherche pour traiter les alertes en ligne. L'approche s'articule autour :

- des alertes issues des sondes ;
- d'une base d'informations cartographiques analogues à celles définies dans $\mathbf{M_2D^2}$. Nous appelons ces informations cartographiques des *profils*, liés à la fonction des hôtes dans le système d'information. Un serveur mandataire est un type de profil, par exemple ;
- d'une base de définitions de phénomènes réputés pour engendrer des fausses alertes, sous la forme de règles logiques reliant les alertes aux informations cartographiques.

Les alertes, les profils et les règles sont des formules logiques identiques à celles définies dans le chapitre 5, impliquant des termes abstraits. Les alertes sont traitées en ligne

par un moteur d'inférence chargé de déterminer si une alerte est un faux positif connu, c'est-à-dire si il existe une règle dont la formule logique est une conséquence logique de la formule de l'alerte.

La base de profils peut être construite par le procédé d'acquisition passive évoqué ci-dessus, par un procédé conventionnel, ou bien encore par l'administrateur. Les règles sont définies par des experts en détection d'intrusions.

Nous pouvons aussi découvrir des éléments de ces bases par induction, à partir des alertes. En effet, les observations faites dans des milieux opérationnels révèlent que les alertes fortement récurrentes s'avèrent souvent être des faux positifs.

Lorsqu'un identifiant d'attaque est référencé dans un grand nombre d'alertes, alors il est possible d'induire des règles : si les hôtes impliqués dans les alertes référençant de tels identifiants d'attaques (en tant que victimes ou en tant qu'attaquant) partagent tous un même ensemble de propriétés cartographiques, alors il est possible d'induire une règle qui relie l'identifiant d'attaque à cet ensemble de propriétés.

De même, lorsqu'un hôte est impliqué dans un grand nombre d'alertes référençant une attaque particulière et qu'il existe une règle reliant cette attaque à des des profils, alors nous pouvons induire une relation entre l'hôte et ce(s) profil(s).

7.2.3 Apprentissage de chroniques

Dans la lignée de l'apprentissage de règles de corrélation d'alertes et des chroniques, nous souhaitons aussi évaluer l'outil FACE. FACE est un outil de découverte de chroniques fréquentes au sein de journaux d'alarmes. Le caractère récurrent des alertes permet en effet d'envisager l'existence de séquences répétitives, qui sont difficiles à déceler par une analyse manuelle. Cela est d'autant plus intéressant que le développement de chroniques nécessite une expertise de haut niveau.

Ces derniers mois, le déploiement à grande échelle de systèmes de détection d'intrusions dans les systèmes d'informations de France Télécom ont suscité un intérêt grandissant pour la corrélation d'alertes auprès des équipes de sécurité. Ces systèmes d'informations constituent autant de terrains d'expérimentation uniques, qui permettent d'envisager le développement de techniques de corrélation d'alertes répondant à des problématiques concrètes et variées, ainsi que leur validation sur des données opérationnelles.

Annexe A

Notations B/Z

L'ensemble des relations entre deux ensembles s et t est noté $s \leftrightarrow t$. Une relation représente une correspondance $(0,n)-(0,n)$ entre les éléments d'un ensemble. L'ensemble des relations partielles entre s et t est noté $s \nrightarrow t$. Une relation partielle est une correspondance $(0,n)-(0,1)$. L'ensemble des fonctions totales entre s et t est noté $s \rightarrow t$. Une fonction totale est une correspondance $(1,n)-(0,1)$. L'ensemble des injections partielles (resp. totales) entre s et t est noté $s \rightarrowtail t$ (resp. $s \rightarrowtail t$). Une injection partielle (resp. totale) est une correspondance $(0,1)-(0,1)$ (resp. $(1,1)-(0,1)$). L'ensemble des surjections partielles (resp. totales) entre s et t est noté $s \twoheadrightarrow t$ (resp. $s \twoheadrightarrow t$). Une surjection partielle (resp. totale) représente un correspondance $(0,n)-(n,1)$.

Nom	Syntaxe	Définition	Conditions
Ensemble de relations	$s \leftrightarrow t$	$\mathcal{P}(s \times t)$	
Relation inverse	r^{-1}	$\{y, x \mid (y, x) \in t \times s \wedge (x, y) \in r\}$	$r \in s \leftrightarrow t$
Domaine	$\mathbf{dom}(r)$	$\{ x \mid x \in s \wedge \exists y \cdot (y \in t \wedge (x, y) \in r)\}$	$r \in s \leftrightarrow t$
Rang	$\mathbf{ran}(r)$	$\{ y \mid y \in t \wedge \exists x \cdot (x \in s \wedge (x, y) \in r)\}$	$r \in s \leftrightarrow t$
Composition	$r \mathbf{;} q$	$q \circ r$	$r \in s \leftrightarrow t \wedge q \in t \leftrightarrow u$
Identité	$\mathbf{id}(s)$	$\{ x, y \mid (x, y) \in s \times s \wedge x = y\}$	
Restriction	$u \lhd r$	$\{ x, y \mid (x, y) \in r \wedge x \in u \}$	$r \in s \leftrightarrow t \wedge u \subseteq s$
Image	$r[w]$	$\mathbf{ran}(w \lhd r)$	$r \in s \leftrightarrow t \wedge w \subseteq s$
Fonction Partielle	$s \nrightarrow t$	$\{ r \mid r \in s \leftrightarrow t \wedge (r^{-1} \mathbf{;} r) \subseteq \mathbf{id}(t) \}$	
Fonction Totale	$s \rightarrow t$	$\{ f \mid f \in s \nrightarrow t \wedge \mathbf{dom}(f) = s \}$	
Injection	$s \rightarrowtail t$	$s \rightarrow t \cap s \nrightarrow t$	
Surjection	$s \twoheadrightarrow t$	$s \nrightarrow t \cap s \rightarrow t$	
Bijection	$s \rightarrowtail\!\!\!\!\twoheadrightarrow t$	$s \rightarrowtail t \cap s \twoheadrightarrow t$	

FIG. A.1 – Notations B/Z

Annexe B

Description de l'expérimentation à Supelec

Le système d'informations de Supélec est constitué d'un réseau de 200 à 300 hotes. La taille relativement modeste de ce réseau nous en donne une bonne maîtrise, ce qui facilite les investigations pour comprendre les causes des alertes.

La structure du réseau de Supélec est assez conventionnelle et en ce sens représentative d'autres systèmes d'informations. Comme l'illustre la Figure B.1, le réseau est compartimenté en plusieurs sous-réseaux :

- la DMZ publique contient les serveurs de Supélec, accessibles de l'extérieur : web, ftp, dns et mail ;
- la DMZ privée contient des serveurs à accès restreints, en particulier aux autres campus de Supelec (Metz et Gif-sur-Yvettes) ;
- le réseau du personnel contient les ordinateurs des membres du personnel de Supélec, tous services confondus ;
- le réseau des élèves contient les machines dédiées à l'enseignement ;
- le réseau de la résidence contient les machines personnelles des élèves de l'école ;
- le réseau xyplex contient des modems ;
- le réseau SSIR est le réseau personnel de l'équipe de sécurité des systèmes d'information et des réseaux

Les adresses IP privées des sous-réseaux sont précisés sur la figure. Le réseau *interne* est composé des réseaux du personnel et des élèves. Les adresses IP de ces deux sous-réseaux sont translatées en 192.168.4.0/24 par le routeur R_2.

Supélec possède quatre plages d'adresses IP publiques de réseaux de classe C : 193.54.192.0/24, 193.54.193.0/24, 193.54.194.0/24, 192.70.40.0/24. La totalité des adresses de la plage 193.54.193.0/24 (réservée à la résidence) est accessible depuis l'extérieur. En revanche, seules quelques adresses IP des autres plages sont accessibles depuis l'extérieur ; elles sont translatées statiquement par le routeur R_i puis routées vers des serveurs des différents sous-réseaux.

L'ensemble du réseau est protégé de l'extérieur par un pare-feu. Trois sondes de détection d'intrusions Snort (symbolisées par des caméras sur le schéma) sont déployées dans le réseau interne, la DMZ publique et devant le pare-feu. Les alertes qu'elles

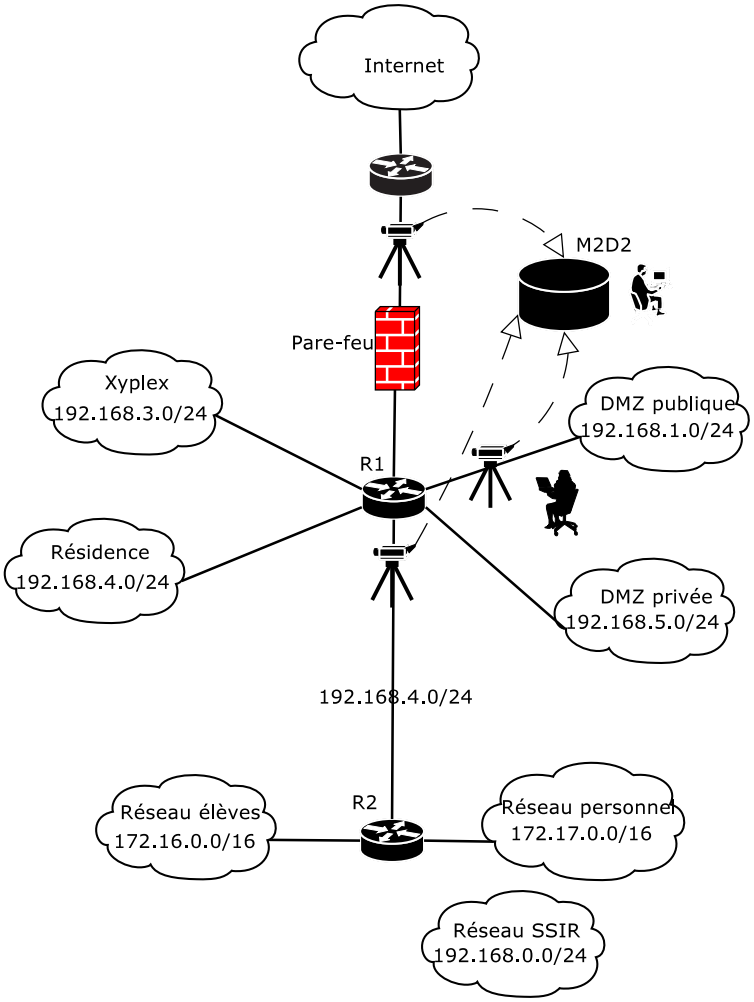

FIG. B.1 – Structure du réseau de Supelec

produisent sont stockées dans une base de donnée relationnelle qui implémente une partie de la structure de $\mathbf{M_2D^2}$. La configuration des sondes est celle fournie par défaut avec Snort.

Le corpus d'alertes utilisé dans les expérimentations est le résultat de l'observation du trafic réel pendant 15 jours, au cours de décembre 2002. Durant cette période, aucune activité notable d'attaque virale n'a été constatée. Par exemple, seulement 30 tentatives du ver Nimda ont été enregistrées contre plusieurs milliers par jour au plus fort du phénomène, datant d'août 2002. Le corpus est constitué de 287088 alertes, en majorité issues de la sonde interne, qui a détecté les tentatives répétées d'une machine esclave d'un dénis de service distribué pour contacter son maitre.

Annexe C

Glossaire de détection d'intrusions

Activité intrusive Manifestation au niveau d'une source de données d'une action qui participe à une intrusion.

Alerte Notification formattée de l'occurence d'une activité intrusive constatée dans une source de données. Une alerte est produite par un analyseur, sur son interface de sortie.

Analyseur Composant de détection d'intrusions qui possède une interface analyseur et produit des alertes à destination d'un manager.

Analyseur morphoscient Analyseur dont la méthode de détection repose sur une connaissance des caractéristiques des attaques dans un flot d'événements.

Analyseur comportemental Analyseur dont la méthode de détection repose sur une déviation significative de l'activité des entités surveillées par rapport à leur activité normale.

Attaque Action consistant à exploiter d'une vulnérabilité.

Capteur Composant de détection d'intrusions qui collecte les données provenant d'une source de donnée et produit un flot d'événements au travers d'une interface de sortie à destination d'un analyseur.

Composant de détection d'intrusions Elément de programme qui participe à la chaine de traitements durant la détection. Un composant possède au moins deux interfaces qualifiées d'entrée et de sortie, permettant respectivement de recevoir et de produire des données. Il existe trois types de composants : les capteurs, les analyseurs et les managers.

Dénis de service Attaque visant à nuire à la disponibilité d'une ressource (réseau par exemple).

Dénis de service distribué Dénis de service dans lequel un grand nombre d'attaquants sont associés.

Maître Acteur d'un dénis de service qui coordonne les attaques d'autres acteurs (les esclaves).

Esclave Acteur d'un dénis de service dont les attaques sont commandées par un maître.

Événement Occurence dans le flot de sortie d'un capteur, à destination d'un analyseur.

Faux positif Alerte en l'absence d'attaque.

Faux négatif Absence d'alerte en présence d'attaque.

Interface analyseur Interface de sortie produisant des alertes.

Interface manager Interface d'entrée recevant des alertes.

Intrusion Enchaînement d'actions qui conduisent à une violation de la politique de sécurité. Les actions qui composent une intrusion peuvent être des attaques ou des actions légitimes.

Manager Composant de détection d'intrusions qui possède une interface manager.

Politique de sécurité est l'ensemble des spécifications du point de vue de l'attribut sécurité-confidentialité d'un système d'informations [7].

Sonde Composant qui regroupe un capteur et un analyseur.

Source de données Entité d'un système d'information qui génère un flot d'informations, appelé activité, qui reflète les actions effectuées sur le système surveillé.

Système de détection d'intrusions Combinaison de plusieurs composants de détection d'intrusions.

Vrai positif Alerte en présence d'attaque.

Vrai négatif Absence d'alerte en l'absence d'attaque.

Vulnérabilité Faille de conception, d'implémentation ou de configuration d'un système logiciel ou matériel [39].

Annexe D

Rappels mathématiques

Connexion de Galois Une connexion de Galois entre deux ensembles ordonnés $(E, \leq_F)$ et $(F, \leq_F)$ est une paire de fonctions (σ, τ), $\sigma : E \to F$ et $\tau : F \to E$ telle que :

$$\forall e \in E, \forall f \in F, e \leq_E \tau(f) \iff \sigma(e) \leq_F f$$

Treillis Un treillis est un ensemble ordonné $(E, \leq)$ dans lequel deux éléments x et y ont une borne inférieure et une borne supérieure notées respectivement par les opérations binaires $\wedge$ et $\vee$. La borne inférieure d'un ensemble d'élements X est le plus grand élément inférieur aux éléments de X. La borne supérieure d'un ensemble d'élements X est le plus petit élément supérieur aux éléments de X.

Un treillis est *complet* si tout ensemble d'éléments admet à la fois une borne inférieure et une borne supérieure.

Diagramme de Hasse Le diagramme de Hasse d'un ensemble muni d'un ordre partiel $(E, \leq)$ est un graphe dont les nœuds sont les éléments de E et dont les arcs sont les éléments de la couverture de $\leq$.

Couverture d'une relation La couverture $cov(\leq)$ d'une relation d'ordre partiel $\leq$ est la réduction transitive et reflexive de $\leq$: $(x, y) \in cov(\leq) \iff \nexists z : x \leq z \leq y$.

Bibliographie

[1] US DoD Standard : Department of Defense Trusted Computer System Evaluation Criteria. DOD 5200.28-STD, Supercedes CSC-STD-001-83, 1985.

[2] J.-R. Abrial. *The B Book : Assigning programs to meanings*. Cambridge University Press, 1996.

[3] R. Agrawal, T. Imielinski, and A. N. Swami. Mining Association Rules Between Sets of Items in Large Databases. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, D.C., 26–28 1993.

[4] J. Allen, A .Christie, W .Fithen, J. M. Hugh, J Pickel, and E. Stoner. State of the practice of intrusion detection technologies (cmu/sei-99-tr-028). Technical report, Carnegie Mellon University, 2000.

[5] J. F. Allen. Toward a General Theory of Action and Time. In *Artificial Intelligence*, volume 23, pages 123–154, 1984.

[6] J.P. Anderson. Computer Security Threat Monitoring and Surveillance. Technical report, Fort Washington - Technical Report Contract 79F26400, 1980.

[7] J. Arlat, J.P. Blanquart, A. Costes, Y. Crouzet, Y. Deswarte, J.C. Fabre, H. Guillermain, M. Kaaniche, K.Kanoun, J.C. Laprie, C. Mazet, D. Powell, C. Rabejac, and P. Thévenod. *Guide de la Sureté de Fonctionnement*. Cepadues editions, 1995.

[8] P. Berkhin. Survey of Clustering Data Mining Techniques. Technical report, Accrue Software, San Jose, CA, 2002.

[9] A. Bierman. Physical Topology MIB. RFC 2922.

[10] Y. Breitbart, M. N. Garofalakis, C. Martin, R. Rastogi, S. Seshadri, and A. Silberschatz. Topology Discovery in Heterogeneous IP Networks. In *INFOCOM (1)*, pages 265–274, 2000.

[11] W. R. Cheswick and S. M. Bellovin. *Firewalls and Internet Security : Repelling the Wily Hacker*. Addison-Wesley Publishing Company, 1994.

[12] F. B. Cohen. Information System Attacks : A Preliminary Classification Scheme. In *Computer and Security*, number 1, pages 29–46, 1997.

[13] A. Cuff. Intrusion detection systems list. http ://www.networkintrusion.co.uk/.

[14] F. Cuppens. Managing Alerts in Multi-Intrusion Detection Environment. In *Proceedings of the 17th Annual Computer Security Applications Conference (ACSAC'01)*, 2001.

[15] F. Cuppens and A. Miege. Alert Correlation in a Cooperative Intrusion Detection Framework. In *Proccedings of the IEEE Symposium on Security and Privacy*, 2002.

[16] F. Cuppens and R. Ortalo. LAMBDA : A Language to Model a Database for Detection of Attacks. In H. Debar, L. Mé, and S. Felix Wu, editors, *LNCS 1907 - Proceedings of the Third International Workshop on the Recent Advances in Intrusion Detection (RAID)*, Lecture Notes in Computer Science (LNCS), October 2000.

[17] D. Curry and H. Debar. Intrusion Detection Message Exchange Format Data Model and Extensible Markup Language (XML) Document Type Definition. Internet Draft (work in progress), December 2003. http ://search.ietf.org/internet-drafts/draft-ietf-idwg-idmef-xml-10.txt.

[18] O. Dain and R. Cunningham. Building Scenarios from a Heterogeneous Alert Stream. In *Proceedings of the 2001 IEEE Workshop on Information Assurance and Security*, June 2001.

[19] O. Dain and R. Cunningham. Fusing a Heterogeneous Alert Stream into Scenarios. In *Proceedings of the 2001 ACM Workshop on Data Mining for Security Applications*, pages 1–13, November 2001.

[20] H. Debar. *Application des réseaux de neurones à la détection d'intrusions sur les systèmes informatiques.* PhD thesis, Université de Paris 6, 1993.

[21] H. Debar, M. Dacier, and A. Wespi. A Revised Taxonomy for Intrusion-Detection Systems. *Annales des Télécommunications*, 55(7-8), 2000.

[22] H. Debar and D. Lefranc. Observations on the Internet Traffic Reaching Broadband-Connected Users. In *Proceedings of EICAR Conference*, 2003.

[23] Hervé Debar and Andreas Wespi. Aggregation and Correlation of Intrusion-Detection Alerts. In Wenke Lee, Ludovic Mé, and Andreas Wespi, editors, *Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection (RAID 2001)*, number 2212 in Lecture Notes in Computer Science, pages 85–103, Davis, CA, USA, October 2001. Springer.

[24] D. Denning. An Intrusion-Detection Model. *IEEE Transactions on Software Engineering*, 13(2) :222–232, 1987.

[25] C. Dousson. *Suivi d'Évolutions et Reconnaissance de Chroniques.* PhD thesis, Université Paul Sabatier (Toulouse), 1994.

[26] C. Dousson. Extending and Unifying Chronicles with Event Counters. In *15th European Conference on Artificial Intelligence (ECAI2002)*, pages 286–291, 2002.

[27] R. Durst, T. Champion, B. Witten, E. Miller, and L. Spagnuolo. Test and Evaluation of Computer Intrusion Detection Systems. *Communications of the ACM*, 42(7), July 1999.

[28] S. Ferré. *Systèmes d'information logiques : un paradigme logico-contextuel pour interroger, naviguer et apprendre.* PhD thesis, Université de Rennes 1, October 2002.

[29] S. Ferré and O. Ridoux. Une Généralisation Logique de l'Analyse de Concepts Formels. Technical Report 3820, INRIA/IRISA, 1999.

[30] S. Ferré and O. Ridoux. A Logical Generalization of Formal Concept Analysis. *International Conference in Conceptual Structures, Lecture notes in Computer Science*, (1867) :371–384, 2000.

[31] D.H. Fisher. *Knowledge Acquisition via Incremental Conceptual Clustering.* PhD thesis, Department of Information and Computer Science, University of California, 1987.

[32] B. Ganter and R. Wille. *Formal Concept Analysis - Mathematical Fundations.* Springer, 1999.

[33] R. Godin, G. Mineau, R. Missaoui, and H. Mili. Méthodes de Classification Conceptuelle Basées sur les Treillis de Galois et Applications. In *Revue d'intelligence artificielle*, number 9, pages 105–137, 1995.

[34] R. P. Goldman, W. Heimerdinger, S. A. Harp, C. W. Geib, V. Thomas, and R. L. Carter. Information Modeling for Intrusion Report Aggregation. In *Proceedings of the DARPA Information Survivability Conference and Exposition*, June 2001.

[35] T.R. Gruber. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. In *International Journal of Human Computer Studies*, volume 5, pages 907–928, 1995.

[36] N. Habra, B. L. Charlier, A. Mounji, and I. Mathieu. ASAX : Software Architecture and Rule-based Language for Universal Audit Trail Analysis. In *Proceedings of the 2nd European Symposium on Research in Computer Security (ESORICS)*, Springer-Verlag, 1992.

[37] M. Handley, C. Kreibich, and Vern Paxson. Network Intrusion Detection : Evasion, Traffic Normalization, and End-to-End Protocol Semantics. In *Proceedings of the 10th USENIX Security Symposium*, Washington, DC, August 2001.

[38] S. Hofmeyr, S. Forrest, and A. Somayaji. Intrusion Detection Using Sequences of System Calls. In *Journal of Computer Security*, volume 6, pages 151–180, 1998.

[39] J. D. Howard and T. A. Longstaff. A Common Language for Computer Security Incidents. CERT - SAND98-8667, 1998.

[40] A. Joshi J. Undercoffer and J. Pinkston. Modeling Computer Attacks : A Target-Centric Ontology for Intrusion Detection. Submitted to RAID 2003, 2003.

[41] G. Jakobson and M. D. Weissman. Alarm Correlation. *IEEE Network Magazine*, pages 52–60, 1993.

[42] Harold S. Javitz, Alfonso Valdez, Teresa F. Lunt, Ann Tamaru, Mabry Tyson, and John Lowrance. Next generation intrusion detection expert system (NIDES) - 1. statistical algorithms rationale - 2. rationale for proposed resolver. Technical Report A016–Rationales, SRI International, 333 Ravenswood Avenue, Menlo Park, CA, March 1993.

[43] K. Julisch. Mining Alarm Clusters to Improve Alarm Handling Efficiency. In *Proceedings of the 17th Annual Computer Security Applications Conference (ACSAC)*, December 2001.

[44] K. Julisch. Clustering Intrusion Detection Alarms to Support Root Cause Analysis. *ACM Transactions on Information and System Security*, 6(4), 2003.

[45] K. Julisch and M. Dacier. Mining Intrusion Detection Alarms for Actionable Knowledge. In *Proceedings of Knowledge Discovery in Data and Data Mining (SIGKDD)*, 2002.

[46] K. Kendall. A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems. Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, June 1999.

[47] C. Kruegel and T. Toth. Using Decision Trees to Improve Signature-Based Intrusion Detection. In G. Vigna, E. Jonsson, and C. Kruegel, editors, *LNCS 2820, Recent Advances in Intrusion Detection (RAID)*, Lecture Notes in Computer Science (LNCS), pages 155–172. Springer-Verlag, 2003.

[48] U. Lindqvist and E. Jonsson. How To Systematically Classify Computer Security Intrusions. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, pages 154–163, 1997.

[49] R. P. Lippmann, D. J. Fried, I. Graf, J. W. Haines, K. R. Kendall, D. McClung, D. Weber, S. E. Webster, D. Wyschogrod, R. K. Cunningham, and M. A. Zissman. Evaluating Intrusion Detection Systems : The 1998 DARPA Off-line Intrusion Detection Evaluation. In *Proceedings of the 2000 DARPA Information Survivability Conference and Exposition (DISCEX'00)*, 2000.

[50] B. Lowekamp, D. R. O'Hallaron, and T. Gross. Topology Discovery for Large Ethernet Networks. In *SIGCOM01*, pages 237–248, 2001.

[51] T.F. Lunt and R. Jagannathan. A Prototype Real-time Intrusion Detection Expert System. In *IEEE Symposium on Security and Privacy*, pages 59–66, 1988.

[52] S. Manganaris, M. Christensen, D. Zerkle, and K. Hermiz. A Data Mining Analysis of RTID Alarms. In *First International Workshop on the Recent Advances in Intrusion Detection (RAID98)*, September 1998.

[53] S. Manganaris, M. Christensen, D. Zerkle, and K. Hermiz. A Data Mining Analysis of RTID Alarms. In *Computer Networks*, number 4, pages 571–577, 2000.

[54] D. E. Mann and S. M. Christey. Towards a Common Enumeration of Vulnerabilities. In *Proceedings of the 2nd Workshop on Research with Security Vulnerability Databases*, January 1999.

[55] D.V. McDermott. A Temporal Logic for Reasoning about Processes and Plans. In *Cognitive Science*, pages 101–155, 1982.

[56] J. McHugh. Intrusion and Intrusion Detection. *International Journal of Information Security*, July 2001.

[57] L. Mé. *Audit de sécurité par algorithmes génétiques.* PhD thesis, Université de Rennes 1, numéro d'ordre 1069, 1994.

[58] D. M. Meira. *A Model for Alarm Correlation in Telecommunication Networks.* PhD thesis, Federal University of Minas Geiras, November 1997.

[59] R.S. Michalsky and R.E. Stepp. Learning from Observation : Conceptual Clustering. In *Machine Learning : An Artificial Intelligence Approach*, 1993.

[60] C. Michel and L. Mé. ADeLe : An Attack Description Language for Knowledge-Based Intrusion Detection. In *Proceedings of the 16th International Conference on Information Security (IFIP/SEC 2001)*, pages 353–365, June 2001.

[61] B. Morin and H. Debar. Correlation of Intrusion Symptoms : an Application of Chronicles. In G. Vigna, E. Jonsson, and C. Kruegel, editors, *LNCS 2820, Recent Advances in Intrusion Detection (RAID)*, Lecture Notes in Computer Science (LNCS), pages 94–112. Springer-Verlag, 2003.

[62] P. Ning, Y. Cui, and D. S. Reeves. Analyzing Intensive Intrusion Alerts via Correlation. In *Proceedings of the 5th International Symposium on the Recent Advances in Intrusion Detection (RAID2002)*, pages 74–94, 2002.

[63] P. Ning, Y. Cui, and D. S. Reeves. Constructing Attack Scenarios Through Correlation of Intrusion Alerts. In *Proceedings of the 9th Conference on Computer and Communication Security*, 2002.

[64] Y. Padioleau and O. Ridoux. A Logic File System. In *USENIX 2003 Annual Technical Conference*, pages 99–112, 2003.

[65] P. A. Porras, M. W. Fong, and A. Valdes. A Mission-Impact-Based Approach to INFOSEC Alarm Correlation. In *Proceedings of the 5th International Symposium on the Recent Advances in Intrusion Detection (RAID2002)*, pages 95–114, 2002.

[66] J.P. Pouzol and M. Ducassé. From Declarative Signature to Misuse IDS. In Wenke Lee, Ludovic Mé, and Andreas Wespi, editors, *4th International Conference on Recent Advances in Intrusion Detection (RAID'01)*, number 2212 in Lecture Notes in Computer Science, Davis, CA, USA, October 2001. Springer.

[67] J.P. Pouzol and M. Ducassé. Formal Specification of Intrusion Signatures and Detection Rules. In S. Schneider, editor, *15th Computer Security Foudations Workshop*, pages 64–76. IEEE Press, 2002.

[68] T. H. Ptacek and T. N. Newsham. Insertion, Evasion, and Denial of Service : Eluding Network Intrusion Detection. Secure Networks, Inc, 1998.

[69] X. Qin and W. Lee. Statistical Causality Analysis of INFOSEC Alert Data. Submitted to RAID 2003, 2003.

[70] R. Missaoui R. Godin and A. April. Experimental Comparison of Navigation in a Galois Lattice with Conventional Information Retrieval Methods. In *International Journal of Man-Machine Studies*, volume 5, pages 747–767, 1993.

[71] M. Roger and J. Goubault-Larrecq. Log Auditing Through Model-Checking. In *Proceedings of the 14th IEEE Computer Security Foundations Workshop (CSFW01)*. IEEE, 2001.

[72] L. Schaelicke, T. Slabach, B. Moore, and C. Freeland. Characterizing the Performance of Network Intrusion Detection Sensors. In G. Vigna, E. Jonsson, and C. Kruegel, editors, *LNCS 2820, Recent Advances in Intrusion Detection (RAID)*, volume 2820 of *Lecture Notes in Computer Science (LNCS)*, pages 155–172. Springer-Verlag, 2003.

[73] R. Shirey. Internet Security Glossary. RFC 2828, 2000.

[74] Y. Shoham. Temporal Logics in AI : Semantical and Ontological Considerations. In *Journal of Artificial Intelligence*, pages 89–104, 1987.

[75] S. Templeton and K. Levitt. A Requires/Provide Model for Computer Attacks. In *Proceedings of the ACM New Security Paradigms Workshop*, pages 31–38, 2000.

[76] Tryo. France télécom. Sony Music, 1998.

[77] A. Valdes and K. Skinner. An Approach to Sensor Correlation. SRI International, 2000.

[78] A. Valdes and K. Skinner. Probabilistic Alert Correlation. In Wenke Lee, Ludovic Mé, and Andreas Wespi, editors, *Proceedings of the 4th International Symposium on the Recent Advances in Intrusion Detection (RAID 2001)*, number 2212 in Lecture Notes in Computer Science, Davis, CA, USA, October 2001. Springer.

[79] G. Vigna. A Topological Characterization of TCP/IP Security. Technical Report TR-96.156, Politecnico di Milano, 1996.

[80] G. Vigna and R. A. Kemmerer. NetSTAT : A Network-based Intrusion Detection Approach. In *Proceedings of the 14th Annual Computer Security Application Conference*, December 1998.

[81] G. Vigna and R. A. Kemmerer. NetSTAT : A Network-based Intrusion Detection System. *Journal of Computer Security*, February 1999.

[82] R. Wille. Restructuring Lattice Theory : An Approach Based on Hierarchies of Concepts. *Ordered Sets*, pages 445–470, 1982.

[83] Mark Wood and Mike Erlinger. Intrusion Detection Message Exchange (IDMEF) requirements. Internet Engineering Task Force - IDWG, 2003. Work in progress, expires April 22nd, 2003.

[84] J. Zimmermann, L. Mé, and C. Bidan. An Improved Reference Flow Control Model for Policy-Based Intrusion Detection. In *Proceedings of the 8th European Symposium on Research in Computer Security (ESORICS)*, 2003.