CRIM : un module de corrélation d'alertes et de réaction aux attaques

Fabien Autrel et Frédéric Cuppens
GET-ENST-Bretagne, 35576 Cesson Sévigné (France)

21 avril 2006

Résumé

Avec le besoin grandissant de protéger les systèmes d'informations, la détection d'intrusion constitue une démarche intéressante mais encore très imparfaite. Il y a actuellement deux approches principales de la détection d'intrusions : l'approche comportementale et l'approche par détection de scénarios. Aucune de ces approches n'est complètement satisfaisante. Elles génèrent souvent de trop nombreux faux positifs et les alertes sont trop élémentaires et insuffisamment précises pour être directement exploitables par un administrateur de sécurité. Une approche intéressante consiste à développer un module de coopération pour analyser et corréler les alertes, générer un diagnostic plus global et synthétique, et aider l'administrateur de sécurité dans le choix d'une contre-mesure adaptée à l'attaque détectée. Cet article présente les travaux que nous avons réalisés, dans ce contexte, pour concevoir le module CRIM (Coopération de Reconnaissance d'Intentions Malveillantes).

I. Introduction

L'objectif principal de la sécurité des systèmes d'informations (SSI) est de concevoir et développer des systèmes d'informations dont le fonctionnement se conforme à la spécification d'une politique de sécurité (PS). Une PS est un ensemble de règles qui définissent les autorisations, les interdictions et les obligations des sujets (c'est-à-dire les utilisateurs et les applications) qui peuvent accéder au système d'informations. Un attaquant (également appelé intrus) correspond à un agent malveillant qui essaye de porter atteinte à la PS. Une attaque est donc une tentative délibérée de violer la PS. Cette attaque peut avoir pour but :

- D'accéder illégalement à une information. Dans ce cas, l'attaque viole une contrainte de confidentialité exprimée dans la PS.
- De créer, modifier ou supprimer illégalement une information. Dans ce cas, l'attaque viole une contrainte d'intégrité.
- D'empêcher les autres utilisateurs d'avoir un accès légal à certains services ou ressources. Dans ce cas, l'attaque viole une contrainte de disponibilité.

Les approches classiques de la SSI sont fondées sur la définition et la mise en œuvre de moyens pour protéger le système contre les attaques. Il s'agit par exemple d'implanter des mécanismes d'authentification, de contrôle d'accès ou des protocoles cryptographiques pour empêcher l'occurrence d'attaques. Par exemple, de nombreux administrateurs de sécurité utilisent des pare-feux pour protéger l'infrastructure informatique dont ils ont la responsabilité. Un pare-feu permet d'implanter une politique de sécurité qui spécifie quels sont les sujets qui ont accès au réseau protégé par ce pare-feu. Cependant, avec le besoin croissant d'ouverture des systèmes via l'Internet, de nouvelles vulnérabilités et de nouvelles attaques exploitant ces vulnérabilités apparaissent régulièrement. On peut, par exemple, citer des attaques très efficaces contre la disponibilité telles que les dénis de service distribués (DDOS) ou celles contre la confidentialité et l'intégrité des communications telles que le détournement de trafic IP (IP hijacking). Face à ce type d'attaque, les pare-feus ne fournissent pas de solution satisfaisante.

Dans ce contexte, la détection d'intrusions [4] constitue une approche attractive. L'objectif n'est plus d'empêcher l'occurrence d'attaques mais de les détecter de manière à préparer la contre-mesure la plus adaptée. La détection d'intrusion présente l'avantage sur les approches classiques fondées sur la protection de pouvoir s'adapter plus rapidement aux nouvelles attaques. La détection d'intrusion est un domaine de recherche actif depuis une vingtaine d'années. Deux approches principales ont jusqu'à présent été proposées : l'approche comportementale et l'approche par détection de scénarios d'attaque. Intuitivement, l'approche comportementale repose sur la construction d'une base de comportements normaux des sujets. Le module de détection d'intrusion (MDI) génère alors une alerte lorsque le comportement effectivement observé dévie du comportement normal enregistré dans la base. L'approche par détection de scénarios d'attaque suppose, quant à elle, l'existence d'une base qui décrit toutes les attaques connues et le MDI analyse des fichiers d'audit à la recherche des traces révélatrices de l'occurrence d'une attaque enregistrée dans la base. Dans les premiers MDI, les données d'audit étaient fournies par les machines hôtes. Ce type de MDI est aujourd'hui appelé MDI hôte. Ces données ne sont pas suffisantes pour détecter de nombreuses attaques possibles via des connexions Internet (telles que par exemple, l'IP spoofing, l'IP hijacking, le smurfing, le flooding, etc.). C'est la raison pour laquelle la plupart des MDI actuels utilisent également des données d'audit réseau. Ces MDI, que l'on appelle MDI réseau, représentent une part importante de l'offre commerciale actuelle.

Cependant, comme nous allons le voir dans les sections II.1. et II.2., aucune des approches comportementales ou par détection de scénarios d'attaques ne fournit des résultats complètement satisfaisantes. Ces approches génèrent souvent de trop nombreuses alertes, correspondant à des faux positifs (c'est-à-dire l'occurrence d'une alerte ne correspondant pas à une attaque), alors que certaines attaques ne sont pas détectées. On parle dans ce cas de faux négatifs. C'est la raison pour laquelle l'un des premiers MDI développés, IDES (Intrusion Detection Expert System) [27], combinaient les deux approches, l'approche par scénarios (implantée en utilisant un système expert) et l'approche comporte-

mentale (implantée en utilisant des méthodes statistiques). L'idée est de profiter des avantages de chacune des approches tout en limitant leurs inconvénients.

Cette idée est intéressante. Par exemple, dans le cas d'un réseau, plusieurs MDI peuvent être installés. Chaque MDI se charge de surveiller une partie du réseau et il est alors nécessaire de les faire coopérer pour obtenir un point de vue global de l'intrusion. Un autre argument en faveur d'une approche coopérative est la possibilité de combiner des MDI qui fonctionnent de façon différente, certains utilisant l'approche comportementale et d'autre l'approche par scénario. Ces approches étant complémentaires, la coopération doit permettre de dériver un diagnostic global plus pertinent que celui fournit par les différents MDI pris séparément. Comme nous l'avons mentionné précédemment, IDES fut le premier prototype à proposer une telle approche qui fut ensuite raffiné dans le projet EMERALD [36]. Des travaux plus récents proposent d'utiliser les réseaux bayésiens [39] et la corrélation d'alertes probabiliste [40]. D'autres démarches sont celles mises en œuvre dans Tivoli Enterprise Console présentée dans [15] ou celle proposée dans le système NCR proposé par P. Ning, V. Cui et R. S. Reeves [33]. L'objectif principal de la coopération entre MDI est de corréler des alertes pour générer des alertes plus globales et éliminer les fausses alertes. Cet article a pour but de présenter nos travaux dans ce domaine. Ceux-ci ont débuté dans le cadre du projet MIRADOR financé par la DGA et se sont poursuivis dans le projet RNTL DICO. Le contexte de ce travail est un environnement de détection d'intrusions avec plusieurs MDI qui génèrent des alertes lorsque des intrusions se produisent. Dans ce contexte, nous avons développé le prototype CRIM (Coopération et Reconnaissance d'Intention Malveillante) [14] qui réalise les fonctions suivantes :

- Gestion des alertes. Cette fonction gère les différents messages d'alertes générés par les MDI dans une base de données relationnelle.
- Regroupement d'alertes. Il s'agit de générer des paquets regroupant les alertes qui correspondent à la même occurrence d'attaque.
- Fusion d'alertes. Une alerte globale est générée pour chaque paquet identifié par la fonction de regroupement.
- Corrélation d'alertes. Cette fonction permet d'identifier les différentes étapes d'un scénario d'attaque plus global.
- Reconnaissance d'intentions. Cette fonction poursuit l'analyse du scénario d'attaque détecté par la fonction de corrélation de manière à anticiper sur les objectifs malveillants de l'attaquant. Elle génère un ensemble de scénarios cohérents avec les alertes reçues et les classe par ordre de plausibilité.
- Réaction. Il s'agit d'activer la contre-mesure la plus adaptée compte tenu du diagnostic fourni par les fonctions précédentes.

Le reste de l'article est organisé de la façon suivante. Les sections II.1 et II.2 introduisent respectivement les approches comportementale et par scénario en récapitulant leurs avantages et leurs inconvénients. La section III présente l'architecture de CRIM et la section IV explique son fonctionnement. La section V expose les expérimentations effectuées pour valider notre approche. Enfin, la section VI conclut l'article.

II. Approches existantes pour la détection d'intrusion

II.1. Approche comportementale

L'approche comportementale est fondée sur une description statistique des sujets. L'objectif est de détecter les actions anormales effectuées par ces sujets (par exemple, des heures de connexion anormales, un nombre anormal de fichiers supprimés ou un nombre anormal de mots de passe incorrects fournis au cours d'une connexion).

Le comportement normal des sujets est appris en observant le système pendant une période donnée (par exemple, un mois). Le comportement normal, appelé comportement sur le long terme, est enregistré dans la base de données et comparé avec le comportement présent des sujets, appelé comportement à court terme. Une alerte est générée si une déviation entre ces comportements est observée. Dans cette approche, le comportement sur le long terme est, en général, mis à jour périodiquement pour prendre en compte les évolutions possibles des comportements des sujets.

On considère traditionnellement que l'avantage principal de l'approche comportementale est de pouvoir être utilisée pour détecter de nouvelles attaques. Cependant, cette approche présente également plusieurs inconvénients. Tout d'abord, le diagnostic fournit par une alerte est souvent flou et nécessite une analyse complémentaire. Ensuite, cette approche génère souvent de nombreux faux positifs car une déviation du comportement normal ne correspond pas toujours à l'occurrence d'une attaque.

II.2. Approche par scénario

La détection d'intrusions peut également s'effectuer selon une approche par scénario. Il s'agit de recueillir des scénarios d'attaques pour alimenter une base d'attaques (*ScenarioBase*). La fonction du MDI consiste alors à analyser les fichiers d'audit à la recherche de motifs (pattern) qui correspondent à la description d'une attaque enregistrée dans *ScenarioBase*.

Le principal avantage d'une approche par scénario est la précision des diagnostics qu'elle fournit par rapport à ceux avancés par l'approche comportementale. Il est bien entendu que l'inconvénient majeur de cette approche est qu'elle ne peut détecter que des attaques dont on a enregistré la signature dans *ScenarioBase*. Or, définir de façon exhaustive *ScenarioBase* est une des principales difficultés à laquelle se heurte cette approche. La génération de faux négatifs est à craindre particulièrement lorsque l'attaque en question possède plusieurs implémentations très voisines.

Il existe de nombreuses attaques difficiles à détecter car elles nécessitent de corréler plusieurs événements. Dans la plupart des produits commerciaux, ces attaques élaborées sont décomposées en plusieurs signatures élémentaires. Cette décomposition peut malheureusement générer de nombreux faux positifs si un

mécanisme plus global n'est pas développé pour corréler les alertes correspondant à ces différentes signatures élémentaires.

III. Principes de CRIM

Au vu des deux sections précédentes, on peut conclure qu'aucune des approches existantes de la détection d'intrusions n'est complètement satisfaisante car elles génèrent de trop nombreux faux positifs et faux négatifs. Ces approches sont néanmoins complémentaires. L'approche comportementale permet de détecter de nouvelles attaques mais ne permet pas de dégager un diagnostic précis de ce qui s'est effectivement produit dans le système. L'approche par scénario peut fournir un meilleur diagnostic mais est limitée à la détection d'attaques déjà connues. De plus, les MDI commerciaux actuels sont souvent fondés sur des opérations de filtrage et de dénombrement qui s'avèrent insuffisantes pour détecter des attaques complexes nécessitant de corréler plusieurs événements.

Par conséquent, nous nous intéressons à la conception d'un module qui permet à plusieurs MDI de coopérer. Ce module de coopération peut être intégré dans une plate-forme de détection d'intrusions pour deux raisons différentes :

1. la plate-forme intègre des MDI qui utilisent des approches de détection différentes (comportementale ou par scénario) ou des MDI fondés sur l'approche par scénario mais qui utilisent des bases d'attaques différentes,
2. la plate-forme intègre des MDI basés sur la même approche de détection mais ces MDI sont distribués sur un réseau de sorte que chaque MDI n'a pas une perception complète de l'activité du réseau.

Dans chaque cas, les objectifs principaux du module de coopération sera de combiner les alertes fournies par ces MDI pour diminuer le nombre de faux négatifs et éliminer le plus grand nombre de faux positifs possible. La figure 1 présente les principes que nous avons proposés pour développer un module de coopération, que nous avons appelé CRIM (Coopération et Reconnaissance d'Intentions Malveillantes) [12]. Ce module réalise six fonctions principales.

La fonction de gestion de la base d'alertes collecte les alertes générées par les différents MDI et les enregistre dans une base de données pour être analysés par les autres fonctions de CRIM. Nous supposons que le format de ces alertes est compatible avec celui défini par l'IDMEF (Intrusion Detection Exchange Format) [9]. L'objectif de l'IDMEF est de définir un format de données et des procédures d'échange communs pour que les MDI et les modules de réaction puissent s'échanger et partager les données de détection. En particulier, nous utilisons le schéma XML défini par l'IDMEF pour représenter des alertes sous forme de documents XML. L'approche proposée pour implanter la fonction de gestion de la base d'alertes consiste à convertir les messages XML correspondant aux alertes en un ensemble de n-uplets qui sont enregistrés dans une base de données relationnelle.

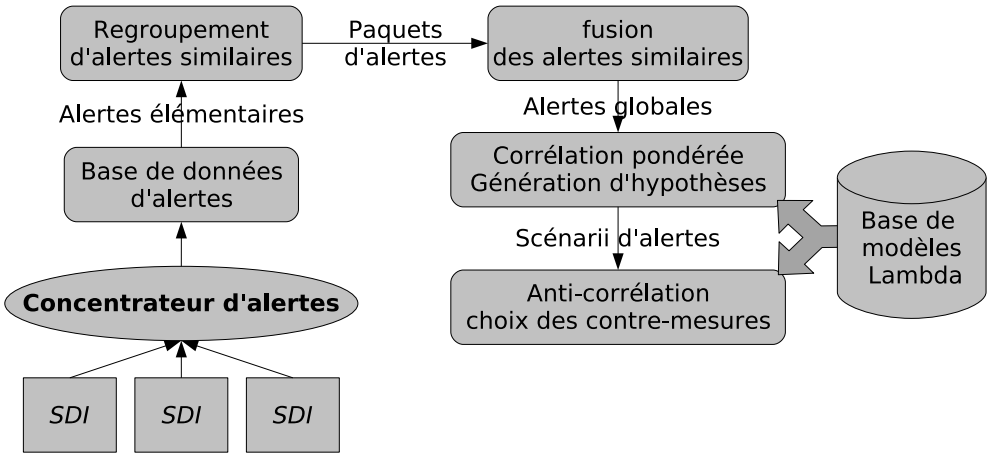

Fig. 1 – Architecture de CRIM
CRIM architecture

Le schéma de cette base de données relationnelle est dérivé du schéma XML qui décrit les messages d'alertes. La fonction de regroupement a accès à cette base d'alertes et génère des paquets d'alertes. Un paquets d'alertes est un ensemble d'alertes qui correspondent à la même occurrence d'attaque. Ces alertes peuvent être générées par le même MDI ou par des MDI différents. L'approche proposée consiste à définir une relation de similarité qui relie les alertes appartenant au même paquet. Chaque paquet est ensuite envoyé à la fonction de fusion. L'objectif de cette fonction est de créer une nouvelle alerte, appelée alerte globale, qui réalise la synthèse des différentes informations contenues dans le paquet à fusionner.

La fonction de corrélation analyse ensuite les alertes générées par la fonction de fusion. Le principe de la corrélation est de considérer que l'attaquant essaye d'atteindre un certain objectif malveillant mais il ne peut pas généralement y parvenir en effectuant une seule attaque. Il va donc, dans ce cas, effectuer plusieurs attaques correspondant à autant d'étapes d'un plan d'intrusion plus global qui lui permet d'atteindre son objectif malveillant.

Les MDI classiques ne permettent de détecter que les attaques élémentaires qui correspondent aux étapes de ce plan d'intrusion. L'objectif de la fonction de corrélation est de corréler les alertes pour reconnaître le plan que l'attaquant est en train de réaliser.

Le résultat de la corrélation est un ensemble de plans candidats qui correspondent aux intrusions de l'attaquant. Cependant, l'objectif final de l'attaquant n'est peut-être pas encore atteint. La fonction de reconnaissance d'intentions a alors pour but d'extrapoler ces plans candidats de manière à anticiper sur les intentions de l'attaquant. Cette fonction doit fournir un diagnostic global du passé (ce que l'attaquant a réalisé jusqu'à maintenant), du présent (ce que l'at-

taquant a obtenu et quel est l'état actuel du système) et du futur (prévoir comment l'attaquant va poursuivre son intrusion).

Au fur et a mesure que des alertes sont reçues et que les modèles Lambda correspondants sont instanciés, les instances de modèles sont pondérées et les scénario rangés par ordre croissant de plausibilité selon le critère exposé dans [5, 1].

Ce diagnostic est alors transmis à la fonction de réaction. L'objectif de cette fonction est d'activer une contre-mesure pour stopper l'intrusion. Actuellement, la décision d'activer une contre-mesure n'est pas automatique dans CRIM mais est contrôlée par l'administrateur de sécurité. Plus précisément, la fonction de réaction vient aider l'administrateur dans le choix de la contre-mesure la mieux adaptée, c'est-à-dire celle qui permettra effectivement d'empêcher les actions malveillantes réalisées par l'attaquant. Cependant il est possible d'automatiser certaines réactions.

La section suivante présente, de façon plus détaillée, le fonctionnement des différentes fonctions de CRIM. Les résultats d'expérimentations que nous avons effectuées pour valider notre approche sont ensuite donnés dans la section 6.

IV. Fonctionnement de CRIM

IV.1. Regroupement d'alertes

Lorsqu'un MDI génère une nouvelle alerte, la fonction de regroupement détermine, parmi les alertes présentes dans la base, celles qui ont un « lien » avec cette nouvelle alerte. La démarche consiste à définir une relation de similarité entre deux messages d'alertes. Intuitivement, une relation de similarité définit dans quels cas deux alertes peuvent être considérées suffisamment proches pour être regroupées.

Ce problème a déjà été étudié par Valdes et Skinner [40]. Ils proposent d'utiliser une approche probabiliste dans laquelle une fonction de similarité est définie pour chaque attribut d'une alerte, la similarité globale étant obtenue en combinant les fonctions de similarité élémentaires et en utilisant la notion d'espérance de similarité. Debar et Wespi proposent de définir des règles expertes d'agrégation [15] qu'ils appellent « définition de duplicata ». Une définition de duplicata spécifie quelles alertes sont attendues pour un certain événement et quelles conditions doivent être satisfaites sur les attributs de ces alertes. Dans [21], Klauss Julisch propose d'agréger les alertes en les regroupant par « root cause » ou causes premières. La motivation principale de ce travail est que, selon Julisch, 90% des alertes de détection d'intrusions sont la manifestation de problèmes de configuration du système surveillé. L'auteur propose de définir la notion de *dissimilarité* entre alertes.

Dans [13], la fonction de similarité fut définie sans utiliser une approche probabiliste. Il s'agissait plutôt d'une approche de type « système expert » dans laquelle les différentes exigences de similarité sont spécifiées par des règles expertes. Ces règles expertes sont spécifiques au domaine et il est nécessaire

d'analyser les alertes générées par plusieurs MDI pour définir correctement ces règles. Ces règles expertes spécifient dans quels cas des instances des entités suivantes sont similaires : la classification, la date, la source et la cible.

Le premier prototype de CRIM utilisait cette approche à base de règles. Les règles étant apprises à partir de résultats expérimentaux conduits sur un réseau de test, le travail nécessaire à leur écriture était ainsi minimisé. Cette approche comporte quelques inconvénients que nous tentons d'éliminer à travers la redéfinition de la fonction de similarité :

- La comparaison de deux alertes donne un résultat booléen, les alertes sont soit similaires ou différentes. Cela introduit certaines ambiguités. Considèrons par exemples 3 alertes A_1, A_2 et A_3. Supposons que A_1 et A_2 ne soient pas similaires d'après la fonction de similarité et qu'elles consituent donc deux paquets d'alertes séparés. Le système reçoit ensuite A_3, qui est jugée similaire à la fois à A_1 et A_2. Il n'est alors pas possible de choisir dans quel paquet d'alerte mettre A_3 sans informations supplémentaires.
- Un résultat booléen n'est pas bien adapté à la comparaison d'attributs continus, tels que le temps. Ainsi au lieu de considérer deux dates de détection comme non similaires lorsque leur écart a atteint un certain seuil, nous faisons décroitre le niveau de similarité entre ces dates à mesure que leur écart s'accroit.

L'approche implantée actuellement dans CRIM s'appuie sur la définition d'un opérateur de similarité Sim permettant d'évaluer, grâce au calcul d'un réel appartenant à $[0, 1]$, dans quelle mesure deux alertes se rapportent à la détection du même événement [2]. Lorsque le réel calculé est supérieur à un certain seuil, on considère que les deux alertes sont similaires et peuvent donc être regroupées dans le même paquet.

Le réel calculé par la nouvelle fonction de similarité lève l'ambiguité citée plus haut. Si l'on considère à nouveau les trois alertes de l'exemple précédent, il est possible de choisir si A_3 doit être regroupée avec A_1 ou A_2 dans le cas ou $Sim(A_1, A_3)$ et $Sim(A_2, A_3)$ sont supérieurs au seuil (et donc A_1 et A_3 sont similaires, ainsi que A_2 et A_3). En effet il convient de regrouper A_3 avec l'alerte la plus similaire.

Le format utilisé dans CRIM pour représenter les alertes est le format IDMEF, qui est un modèle orienté objet d'une alerte. Ce format définit un ensemble d'attributs qui se veut suffisamment complet pour pouvoir représenter une alerte quelque soit la sonde l'ayant générée.

Étant donné l'ensemble des attributs définis par l'IDMEF, nous avons défini un ensemble de fonctions de similarité, plus précisément une fonction pour chaque attribut du modèle. L'application de ces fonctions sur les attributs de deux alertes génère un ensemble de valeurs de similarité, ces valeurs étant des réels de l'intervalle $[0, 1]$. Pour obtenir une valeur de similarité entre les deux alertes comparées, nous sommons ces valeurs et nous les pondérons [2], ceci afin de prendre en compte les spécificités des différentes classes d'attaque. En effet en fonction de la classe d'attaque associée à une alerte, certains de ses attributs sont plus ou moins pertinents lors de la comparaison avec une autre

alerte. Par exemple une attaque consistant à envoyer des paquets contenant une adresse source forgée (spoofing) devra être traitée différemment par la fonction d'agrégation. Dans ce cas, le poids associé aux instances de classes correspondant à des adresses forgées sera nul ou très faible.

Étant donné que pour une même attaque deux MDI différents peuvent utiliser des classifications différentes, il est nécessaire de définir un dictionnaire permettant de mettre en correspondance une classification et son attaque. Ainsi grâce à ce dictionnaire et à partir de la classification d'une alerte, nous sélectionnons les poids à utiliser lors du calcul de similarité. Dans le cas ou le module d'agrégation/fusion reçoit une alerte dont la classification n'est pas connue, et donc pour laquelle l'ensemble de poids n'est pas défini, un ensemble de poids par défaut lui est associé. Cet ensemble de poids par défaut permet ainsi le traitement des alertes correspondant à des attaques inconnues lors de la phase de création des ensembles de poids par un expert. Nous proposons cette réponse au problème du traitement des alertes inconnues cité plus haut dans la précédente approche.

Le paramétrage des fonctions de similarité fait intervenir soit une connaissance experte, soit un apprentissage. Par exemple la fonction de similarité entre adresses IP se base sur la définition d'une taxonomie sur les adresses IP. Cette taxonomie peut être par exemple définie à partir de la topologie du réseau, de la politique de sécurité du système ou encore des différents types de machines du réseau. Ces informations sont données par un expert (bien que l'on puisse imaginer générer automatiquement une taxonomie à partir de la topologie du réseau, elle-même découverte automatiquement par des outils, par exemple OSSIM [35]). La fonction de comparaison de dates de détection requiert quand à elle un apprentissage. Cette fonction est paramétrée par deux délais d_1 et d_2. d_1 représente la durée minimale entre le moment ou une attaque est lancée et le moment ou la première alerte est générée. d_2 représente la durée entre le moment ou une attaque est jouée et le moment ou la dernière alerte est reçue. Ces calculs sont faits expérimentalement en jouant les attaques que l'on veut caractériser sur un réseau expérimental avec tous les MDI utilisés par la suite sur le système à surveiller.

IV.2. Fusion d'alertes

La fonction de fusion est utilisée, pour générer pour chaque paquet identifié par la fonction de regroupement, une alerte globale. Le processus général pour dériver cette alerte globale est le suivant. Supposons qu'une nouvelle alerte $alert_i$ soit générée par l'un des MDI. Il y a alors deux possibilités :

- Il n'y a pas d'alerte similaire à $alert_i$, toutes les valeurs de similarité calculées entre $alert_i$ et les paquets existants sont inférieures au seuil de similarité ; dans ce cas, un nouveau paquet contenant $alert_i$ est créé et une nouvelle alerte globale est générée pour ce paquet. Pour le moment, cette alerte globale est naturellement très similaire à $alert_i$.
- Si $alert_i$ peut être insérée dans un paquet existant, alors l'alerte globale associée à ce paquet est mise à jour en fusionnant l'$alert_i$ avec l'alerte

globale associée à ce paquet.

La génération de l'alerte globale consiste à « synthétiser » les différentes informations spécifiées dans les alertes du paquet.

Le mécanisme de synthèse implanté dans CRIM s'applique sur les attributs temporels des alertes (les instances des classes $CreateTime$ et éventuellement $DetectTime$), sur les informations concernant la (les) cible(s) et la (les) source(s) (les instances des classes $Target$ et $Source$) et sur la classification des alertes du paquet. En ce qui concerne les attributs temporels, l'alerte globale issue de la fusion possède éventuellement une instance de la classe $DetectTime$, si cette dernière est plus récente que l'instance la plus récente de la classe $CreateTime$. L'alerte globale a donc la date la plus récente des alertes contenues dans le paquet.

En ce qui concerne la fusion des informations sur la (les) cible(s) et la (les) source(s), l'opération consiste à comparer les arbres des instances des classes $Target$ et $Source$. A l'issue de cette comparaison on supprimera des instances redondantes de $Target$ et $Source$ et éventuellement fusionnera certaines instances. Par exemple, deux instances de la classe $Source$ peuvent avoir les mêmes attributs exceptée l'instance de la classe $Node$. Si les deux instances de $Node$ ne différent que par leur(s) instance(s) de la classe $Address$, on crée une seule instance de $Source$ avec une instance de $Node$ contenant les instances de $Address$. Ceci est permis par l'IDMEF qui spécifie que la relation d'agrégation entre $Node$ et $Address$ est du type $0..*$.

Enfin les informations de classification sont fusionnées en ne gardant que la classification la plus majoritaire dans le paquet.

IV.3. Corrélation d'alertes

Deux approches principales ont été identifiées pour la fonction de corrélation :

- La corrélation explicite des événements est utilisée lorsque l'administrateur est capable d'exprimer des liens entre événements malveillants. Ces liens peuvent être logiques ou topologiques. Cette approche est principalement fondée sur la connaissance des relations entre alertes et de la topologie des systèmes d'informations qui permet d'établir des liens entre équipements et applications [17]. L'approche fondée sur l'usage de chroniques présentée dans [30] est un exemple de corrélation explicite. Dans cette approche, un expert écrit l'ensemble des scénarios détectables dans le langage des chroniques. De la même manière dans [16] les auteurs proposent une représentation des scénarios à l'aide d'arbres ou chaque noeud représente un plan ou un sous-plan d'intrusion. Là encore un expert doit écrire une librairie de plans.
- La corrélation implicite des événements est utilisée pour mettre en évidence des relations (en général statistiques) entre événements. Cette approche est principalement fondée sur l'observation de groupes d'alertes et l'extraction de relations implicites entre ces alertes. Plusieurs travaux ont montré que les MDI produisent souvent des groupes d'alertes en fonction des données de configuration, du trafic et de la topologie du système d'information

surveillé. Ces travaux sont basés sur des techniques d'apprentissage (par exemple, classification [25], fouille de données [41], réseaux de neurones [26], etc...).

Nous proposons dans CRIM une approche différente appelée corrélation semi-explicite [10]. Cette approche est basée sur la description logique des attaques dans le langage LAMBDA [11]. Dans ce langage, une attaque est spécifiée en utilisant les cinq attributs suivants :

- La pré-condition : condition logique qui spécifie les conditions qui doivent être satisfaites pour que l'attaque soit un succès.
- La post-condition : condition logique qui spécifie les effets de l'attaque lorsque l'attaque est un succès.
- La détection : mise en correspondance d'un modèle LAMBDA et une alerte et instanciation des variables du modèle.
- La vérification : combinaison d'événements qui permet de vérifier que l'attaque est un succès.

Il s'agit ensuite d'analyser les descriptions d'attaques en LAMBDA pour dériver d'éventuels liens logiques entre la post-condition d'une attaque A et la pré-condition d'une attaque B. Si un tel lien existe, il est alors possible de corréler une occurrence de l'attaque A avec une occurrence de l'attaque B. Ce lien de corrélation représente le fait qu'un attaquant a d'abord réalisé l'attaque A dans le but de pouvoir ensuite réaliser l'attaque B.

La corrélation explicite est particulièrement adaptée à la reconnaissance de scénarios constants. Par exemple un script d'attaque génère la même séquence d'alerte à chaque exécution. Un ver se propageant exécute la même séquence d'action à chaque fois qu'il infecte un système. La corrélation semi-explicite peut détecter les mêmes scénarios que ceux détectés par des méthodes de corrélation explicite. Cependant elle est mieux adaptée dans le cas d'un attaquant intelligent planifiant son attaque. En effet étant donné que les liens de corrélation entre modèles d'attaques son découverts automatiquement, le travail de modélisation de chaque attaque est moins important que la modélisation de tous les scénario explicites possibles.

Pour définir formellement ce type de corrélation entre la post-condition d'une attaque et la pré-condition d'une autre attaque, soit $post(A)$ la formule logique représentant la post-condition de l'attaque A et $pre(B)$ la formule logique représentant la pré-condition de l'attaque B. Intuitivement, on dira que les attaques A et B sont corrélées si $post(A)$ et $pre(B)$ ont au moins un prédicat en commun qu'il est possible d'unifier. Nous renvoyons à [10] pour une présentation formelle de cette notion de corrélation.

Pour illustrer cette définition, considérons une action *mount* qui permet à un attaquant identifié par $User$ de monter une partition exportée par une certaine cible identifiée par $Address$. Pour réaliser cette action, l'attaquant doit savoir que la cible exporte une certaine partition. Cette condition est représentée par la formule suivante :

$$knows(User, exported\_partition(Address, Partition)).$$

Cette pré-condition apparaît dans $pre(mount)$:

$remote\_access(User, Address), exported\_partition(Address, Partition)$.

Considérons une autre action *showmount* qui permet à un attaquant identifié par $User$ de savoir quelles sont les partitions exportées par un certain système cible identifié par $Address$. L'effet de cette attaque est représenté par la formule suivante :

$$knows(User, exported\_partition(Address, Partition)).$$

Cette formule correspond à $post(showmount)$.

On peut par conséquent corréler *showmount* avec *mount* si la condition d'unification suivante entre $post(showmount)$ et $pre(mount)$ est satisfaite : les attaquants sont égaux, les cibles sont égales et les partitions sont égales.

Considérons maintenant une alerte $alert_1$ correspondant à une occurrence de $attack\_showmount$ et une $alert_2$ correspondant à une occurrence de $attack\_mount$. Les conditions d'unification précédentes signifient que l'on peut corréler $alert_1$ et $alert_2$ si les conditions suivantes sont satisfaites :

- $user(source(alert_1)) = user(source(alert_2))$, c'est-à-dire l'attaquant (physique) associé à $alert_1$ est le même que l'attaquant (physique) associé à $alert_2$.
- $address(target(alert_1) = address(target(alert_2))$, c'est-à-dire le système cible de l'attaque dans $alert_1$ est le même que celui apparaissant dans $alert_2$.
- $detecttime(alert_1) < detecttime(alert_2)$, c'est-à-dire $alert_1$ a eu lieu avant $alert_2$ (cette dernière condition doit toujours être vérifiée pour pouvoir corréler deux alertes).

On peut remarquer que la condition d'unification sur les partitions ne conduit pas à une exigence pour corréler les alertes. Ceci s'explique car nous supposons que les partitions ne sont pas des informations fournies par l'alerte correspondant à *showmount*.

Dans notre approche, les différents liens logiques sont automatiquement découverts au cours d'une phase d'analyse préalable de la base d'attaques spécifiées en LAMBDA. Ceci permet de générer une base de règles de corrélation qui sont ensuite appliquées pour corréler les alertes et construire des plans d'intrusions candidats correspondant à l'activité de l'attaquant.

IV.4. Reconnaissance d'intentions

Le principe pour déterminer les intentions de l'attaquant repose sur la définition d'objectifs d'intrusion. Intuitivement, un objectif d'intrusion est défini comme une violation de la politique de sécurité. Dans notre approche, un objectif d'intrusion est représenté par une condition exprimée en logique du premier ordre. Si un état du système surveillé satisfait cette condition, alors l'objectif d'intrusion est atteint dans cet état. Par exemple, obtenir illégalement des droits d'accès ou rendre indisponible un serveur du système sont des exemples d'objectifs d'intrusion.

La définition de la corrélation d'alerte présentée dans la section IV.3 est étendue pour corréler une attaque et un objectif d'intrusion. Ainsi si $post(A)$ est la formule logique représentant la post-condition de l'attaque A et $obj(I)$ la formule logique représentant la condition à satisfaire pour atteindre un objectif d'intrusion I, alors on dira que l'attaque A et l'objectif d'intrusion I sont corrélés si $post(A)$ et $obj(I)$ ont au moins un prédicat en commun qu'il est possible d'unifier. Nous renvoyons à [8] pour une présentation formelle de cette notion de corrélation.

Le processus de corrélation présenté dans la section IV.3 crée des ensembles d'alertes corrélées. Ces ensembles sont appellés plans candidats. Lorsqu'un plan candidat est généré par la fonction de corrélation, on commence par tester si ce plan a permis d'atteindre un objectif d'intrusion. Si tel est le cas, on dira que ce plan correspond à un scénario d'intrusion couronné de succès. Le scénario est alors immédiatement transmis à la fonction de réaction (voir section IV.5). Sinon, on considère qu'il s'agit d'un scénario d'intrusion en cours. Dans ce cas, la fonction de reconnaissance d'intention permet d'extrapoler ce scénario pour anticiper sur les différents objectifs d'intrusion que ce scénario permettraient d'atteindre si l'attaquant poursuit son intrusion. En général, plusieurs scénarios possibles, appelés scénarios potentiels, sont ainsi construits à partir du scénario en cours. [5, 1] propose alors d'adapter la définition de la corrélation proposée dans la section IV.3 pour pouvoir ordonner ces différents scénarios potentiels. Dans cette approche, il s'agit d'associer un coefficient de corrélation aux différentes étapes du scénario. Ces différents coefficients sont ensuite combinés au niveau du scénario global ce qui permet d'ordonner les différents scénarios potentiels.

IV.5. Réaction

L'objectif de la fonction de réaction est de déterminer les contre-mesures les plus adaptées pour agir sur les scénarios identifiés par les fonctions de corrélation et de reconnaissance d'intentions. Plusieurs stratégies de réaction sont possibles et nous nous sommes inspirés dans CRIM de la taxonomie proposée dans [20]. Plus précisément, nous considérons deux types de réaction :

- La correction qui correspond à une action qui modifie l'état du système pour corriger une vulnérabilité identifiée ou une mauvaise configuration de la politique de sécurité. Par exemple, l'installation d'un patch est une forme de correction.
- La compensation qui correspond à une action qui bloque l'attaque mais sans correction. Le système est toujours vulnérable mais la réaction empêche l'attaquant de réaliser son intrusion. Par exemple, il est possible d'arrêter un service vulnérable, de fermer une connexion entre l'attaquant et la cible (en utilisant un TCP-reset) ou de reconfigurer un pare-feu pour bloquer la source de l'attaque.

Notre approche est donc fondée sur une bibliothèque de réaction qui contient les différents types de contre-mesures qui peuvent être déclenchées pour stopper des intrusions. Comme il peut s'avérer difficile pour l'administrateur de sécurité de choisir la contre-mesure adaptée, la fonction de réaction fournit une aide pour

effectuer ce choix. Elle permet également d'établir les paramètres de la contre-mesure. Par exemple, lorsque la contre-mesure consiste à fermer une certaine connexion, les adresses IP de la source et de la cible doivent être convenablement fixées avant de déclencher la contre-mesure.

Dans CRIM, la réaction peut donc être vue comme une fonction d'aide à la décision. Sa conception est fondée sur une modélisation logique des contre-mesures dans un formalisme proche de celui proposé par LAMBDA pour représenter les attaques. Ainsi, une contre-mesure est spécifiée en utilisant les quatre attributs suivants :

- La pré-condition : état dans lequel doit se trouver le système pour que la contre-mesure soit couronnée de succès.
- La post-condition : état dans lequel se trouve le système après réalisation de la contre-mesure.
- L'action : action nécessaire pour réaliser la contre-mesure.
- La vérification : condition pour vérifier le succès de la contre-mesure.

En utilisant ce formalisme, il est alors possible de reconnaître les contre-mesures qui ont un effet négatif sur un objectif d'intrusion, c'est-à-dire qui permettent à l'administrateur de contrecarrer cet objectif. Pour cela, nous avons défini la notion d'anti-corrélation [3]. Si $post(C)$ est la formule logique représentant la post-condition d'une contre-mesure C et $obj(I)$ la formule logique représentant la condition à satisfaire pour atteindre un objectif d'intrusion I, alors on dira que la contre-mesure C et l'objectif d'intrusion I sont anti-corrélés si $post(C)$ et $obj(I)$ ont au moins un prédicat tel que ce prédicat apparaît dans $post(C)$ et sa négation apparaît dans $obj(I)$ ou inversement, le prédicat apparaît dans $obj(I)$ et sa négation apparaît dans $post(C)$, et il est possible d'unifier le prédicat dans les deux formules.

Une autre utilisation de la réaction consiste à agir sur un scénario en cours de réalisation. Il s'agit alors de modifier l'état du système pour rendre impossible l'une des étapes du scénario. Dans ce cas, la notion d'anti-corrélation est également utilisée pour reconnaître les contre-mesures candidates. Si A est une étape du scénario et C une contre-mesure, alors C sera sélectionnée par le processus de réaction si $post(C)$ est anti-corrélée avec $pre(A)$.

Nous donnons dans la section V.3 des exemples d'application de ces deux types de réaction.

V. Expérimentation

V.1. Regroupement et fusion d'alertes

Les expérimentations conduites sur les fonctions de regroupement et de fusion ont consisté à appliquer ces fonctions sur les alertes générées par une sonde *snort* lors de l'exécution d'attaques « élémentaires ». Par attaque élémentaire, nous désignons une attaque qui ne peut être décomposée.

Les attaques ont été lancées à partir de l'outil *Nmap*. Nous n'avons pas défini de poids spécifiques aux attaques utilisées, les poids par défaut ont donc

été appliqués pour tous les tests. Les délais d_1 et d_2 cités à la fin du paragraphe IV.1 sont respectivement fixés à 2 et 10 secondes.

Nmap [34] est un outil gratuit qui peut être utilisé pour explorer des réseaux. Il permet de balayer efficacement un grand nombre d'adresses IP et d'obtenir diverses informations. En effet *Nmap* peut déterminer, entre autres, le système d'exploitation, les noms et versions des services démarrés ainsi que les pare-feux installés sur une machine. Cet outil peut être utilisé par un administrateur système pour rapidement réunir des informations sur le système qu'il gère, mais il peut aussi être utilisé par un attaquant. En fait un attaquant peut chercher des machines exécutant des versions vulnérables de services ou de logiciels et utiliser ces informations pour mettre au point un scénario d'attaque.

Nous avons utilisé deux machines cibles dont le traffic réseau est surveillé par une sonde *Snort* et avons exécuté la même commande *Nmap* en même temps sur les deux cibles. De cette manière les alertes générées pour les deux cibles sont mélangées. Nous avons exécuté les deux commandes *Nmap* suivantes :

- *Nmap -sO* : cette commande permet d'identifier les protocoles utilisés par une machine. *Nmap* envoie des paquets IP sans l'entête de protocole à la machine cible en utilisant tous les numéros de protocoles possibles. A l'exécution de cette commande, *Snort* a généré 3776 alertes en 120 secondes. Après agrégation et fusion, nous obtenons 64 groupes d'alertes ou *clusters* d'alertes. Étant donné que nous avons exécuté la commande sur deux cibles, nous obtenons 32 clusters pour la première cible et 32 autres pour la seconde. Les deux ensembles de 32 clusters ayant la même composition, nous ne présentons les résultats que pour une cible. Ceci montre que l'outil d'agrégation a pu séparer les alertes concernant chaque cible.

  Parmi les 32 clusters, 10 clusters ont une taille allant de 30 a 400 alertes. A l'intérieur de ces clusters, l'écart maximal entre deux alertes est de 10 secondes. Pour ces clusters, la classification de l'alerte est *BAD-TRAFFIC Unassigned/Reserved IP protocol*. Ces alertes ont été générées car certains paquets ont été envoyés avec un numéro de protocole invalide. Le numéro de protocole est un entier codé sur 8 bits, ainsi *Nmap* génère des paquets en utilisant 256 numéros de protocoles possibles.

  Les autres clusters ont une taille plus faible, qui va de 1 à 6 alertes. Toutes les alertes au sein de chaque cluster ont la même classification et l'écart le plus grand entre deux alertes du même cluster ne dépasse pas deux secondes. Pour chacun de ces clusters, l'événement associé correspond à l'envoi d'un paquet utilisant un numéro de protocole réservé. Les classifications suivantes apparaissent dans ces clusters : *BAD-TRAFFIC IP Proto 55 IP Mobility*, *BAD-TRAFFIC IP Proto 103 PIM*, *BAD-TRAFFIC IP Proto 77 Sun ND* and *BAD-TRAFFIC IP Proto 53 SWIPE*.

  Cette expérimentation a montré que l'outil d'agrégation peut isoler des événements ponctuels noyés dans un ensemble d'alertes correspondant à l'utilisation de protocoles invalides. Après fusion, nous obtenons 64 alertes, une alerte par cluster. Ces 64 alertes occupent 104Ko au lieu des 3Mo des 3776 alertes initiales.

– *Nmap -sS* : cette commande effectue un balayage des ports de la machine cible en envoyant des paquets SYN sur chaque port pour voir lesquels sont ouverts. Si un port est ouvert, un paquet *SYN-ACK* est reçu, sinon un paquet RST est reçu. L'exécution de cette commande génère trois alertes. L'écart temporel entre la première alerte et la dernière est de une seconde. Il semble que le plugin IDMEF de *Snort* ne produise pas de date de détection dont la précision soit inférieure à la seconde.
Les trois alertes sont agrégées en un seul cluster car l'écart temporel est faible, les adresses sources et destination sont identiques de même que les ports source et destination. D'autre part les ports utilisés par cette commande font partie des ports *well known* (voir [2] pour la taxonomie sur les numéros de port).
Après fusion du cluster, nous obtenons une alerte avec une adresse source, une adresse destination, un port source et une liste de trois ports destination.

V.2. Résultats des tests de Corrélation

Nous avons également testé notre approche sur plusieurs attaques « complexes », qualifiées ainsi parce qu'elles nécessitent plusieurs étapes pour être réalisées. A titre d'exemple, considérons l'attaque suivante (voir figure 2) :

– Etape 1 ($rpcinfo$) : rpcinfo <target>
– Etape 2 ($showmount$) : showmount <target>
– Etape 3 ($mount$) : mount directory
– Etape 4 ($rhost\_modification$) : cat '++' > .rhost
– Etape 5 ($rlogin$) : rlogin <target>

Cette attaque est appelée « illegal nfs mount ». Elle comprend 5 étapes différentes. Nous avons expérimenté cette attaque sur un système protégé par une sonde Snort et une sonde e-trust. Lorsque cette attaque est initiée, 8 alertes sont générées : 6 par Snort et 2 par e-trust.

Notre fonction de regroupement fournit 4 paquets. En effet, aussi bien Snort que e-trust ne détecte pas l'étape 4. Le résultat du regroupement permet donc d'identifier correctement chacune des étapes détectées de cette attaque. Ces paquets sont fournis à la fonction de corrélation qui effectue une analyse plus approfondie des paquets. L'objectif est de corréler ces 4 paquets afin de déterminer l'attaque globale. Conformément à notre approche, les 5 étapes correspondant à l'attaque *illegal nfs mount* sont spécifiées en utilisant Lambda. Ces spécifications sont ensuite analysées hors-ligne afin de reconstruire les liens logiques entre ces étapes.

Les conditions de corrélation apparaissant dans la figure 2 sont les suivants (en indice apparaît l'action à laquelle la variable est associée) :

– cond1 : $Address_{rpcinfo} = Address_{showmount}$
– cond2 : $Address_{showmount} = Address_{mount} \wedge$
$User_{showmount} = User_{mount}$
– cond3 : $User_{mount} = User_{rhost\_modification} \wedge$
$Partition_{mount} = Partition_{rhost\_modification}$

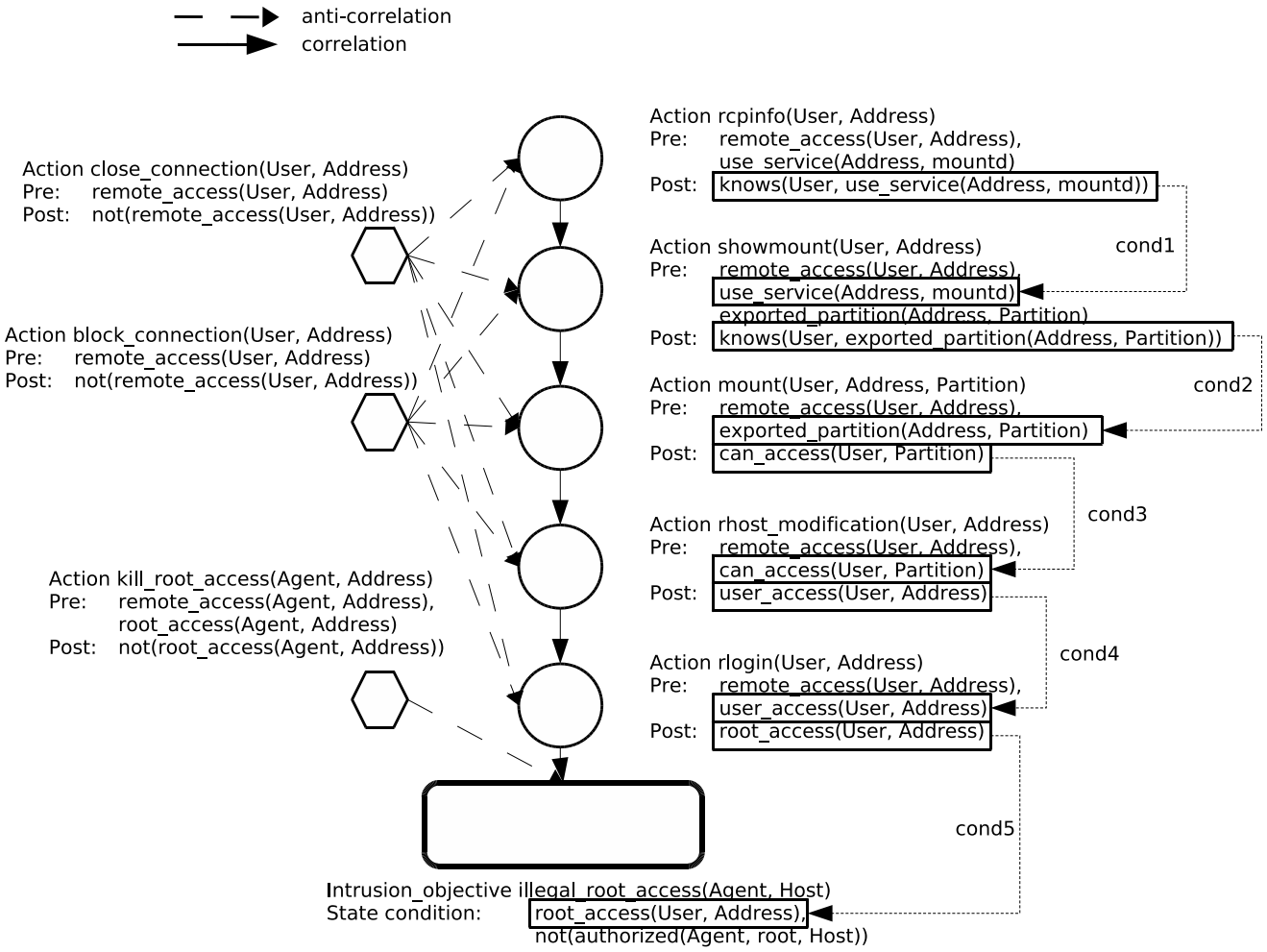

FIG. 2 – Graphe de corrélation du scénario *illegal nfs mount*
Correlation graph for the *illegal nfs mount* scenario

– cond4 : $Address_{rhost\_modification} = Address_{rlogin} \wedge$
$User_{rhost\_modification} = User_{rlogin}$
– cond5 : $Address_{rlogin} = Address_{illegal\_root\_access} \wedge$
$User_{rlogin} = User_{illegal\_root\_access}$

Ces conditions sont utilisées pour corréler les 4 alertes globales générées par notre fonction de fusion. La principale difficulté dans la reconnaissance de l'attaque complexe « illegal nfs mount » vient du fait qu'aussi bien Snort que e-Trust ne détectent pas l'étape 4, $rhost\_modification$. Pour résoudre ce problème nous procédons de la façon suivante. La fonction de corrélation reçoit une alerte correspondant à l'action *mount* et une autre correspondant à l'action *rlogin*. Comme la fonction de corrélation sait qu'il est possible de corréler *mount* avec $rhost\_modification$ et ensuite $rhost\_modification$ avec *rlogin*, elle fait l'hypothèse qu'il est possible de corréler indirectement *mount* avec *rlogin*. Dans ce cas, il s'agit d'effectuer une génération d'hypothèse. En d'autres termes, on procède à la création d'une nouvelle alerte (appelée alerte virtuelle) correspondant à $rhost\_modification$. Il est bien entendu que cette génération d'hypothèse ne peut réussir que si les conditions *cond*3, *cond*4 et *cond*5 sont satisfaites. Plus précisément, la génération d'hypothèse va échouer si $User_{mount} \neq User_{rlogin}$ ou bien $Address_{mount} \neq Address_{rlogin}$. En effet, dans ce cas les conditions *cond*3 et *cond*4 ne sont pas satisfaites. Lorsque la génération d'une hypothèse correspondant à une occurrence de l'action $rhost\_modification$ est concluante, la fonction de corrélation relie toutes les alertes générées par l'action *illegal nfs mount* dans un même scénario en faisant l'hypothèse que l'étape 4 a été réalisée mais n'a pas pu être détectée.

Nous avons testé la fonction de corrélation sur d'autres scénarios complexes. Dans [5] nous présentons un scénario permettant à une personne non autorisée d'imprimer un fichier. Dans [18, 19] un scénario basé sur le principe de l'attaque mise au point par Kevin Mitnick est présentée, ainsi que des contre-mesures adaptées.

V.3. Reconnaissance d'intentions et réaction

La réalisation du scénario « illegal nfs mount » permet à un attaquant d'atteindre un objectif d'intrusion « $illegal\_root\_access$ » qui correspond à l'ouverture d'une session administrateur en contournant les procédures d'authentification. La fonction de reconnaissance d'intentions est utilisée pour détecter que cet objectif d'intrusion est atteint.

Comme nous l'avons expliqué dans la section IV.5, la fonction de réaction peut être utilisée de deux façons différentes. Une première possibilité consiste à appliquer une contre-mesure pour neutraliser l'objectif d'intrusion. Par exemple, la contre-mesure $kill\_root\_access$ qui ferme la session de l'utilisateur est une contre-mesure candidate. Une autre possibilité est de réagir avant que l'objectif d'intrusion ne soit atteint. Dans ce cas, la contre-mesure consiste à contrecarrer l'une des étapes du scénario d'intrusion. Par exemple, la contre-mesure $close\_connection$, qui ferme la connexion entre la machine attaquée et l'attaquant en envoyant un TCP reset, est une contre-mesure candidate.

VI. Conclusion

Dans cet article, nous avons présenté les principales fonctions nécessaires pour développer un module de coopération destiné à améliorer les résultats fournis par les MDI actuels. La fonction de gestion des alertes permet de collecter et enregistrer les alertes générées par ces MDI dans une base de données relationnelle de manière à pouvoir facilement les analyser et les comparer. La fonction de regroupement d'alertes génère des paquets d'alertes répondant à des critères de similarité. La fonction de fusion synthétise les alertes contenues dans chaque paquet en générant une alerte globale plus précise et plus informative. Ces alertes globales sont fournies à la fonction de corrélation qui recherche des liens logiques permettant de mettre en évidence des scénarios d'attaques plus complexes nécessitant plusieurs étapes pour être réalisés. Ces scénarios sont pondérés afin de les ordonner suivant leur degré de plausibilité, ceci permettant de fournir à l'administrateur un diagnostique plus pertinent. La fonction de reconnaissance d'intention analyse et extrapole ces scénarios pour déterminer les objectifs de l'attaquant. Enfin, la fonction de réaction aide l'administrateur de sécurité dans le choix des contre-mesures qui permettent de neutraliser l'attaque.

Une première version de CRIM, implantant ces diverses fonctions, a été réalisée en Gnu-Prolog puis Swi-Prolog. Une deuxième version a été développée, le module CRIM est maintenant intégré dans la plate-forme de supervision de la sécurité Platinum (Platform of Intrusion Management) développée à l'ENST Bretagne Campus de Rennes. Cette plate-forme fournit plusieurs fonctions, en particulier la collecte des alertes dans une base de données gérée par PostgreSQL. Platinum permet également d'exécuter la contre-mesure choisie par l'administrateur de sécurité en utilisant le module de réaction de CRIM. CRIM a été recodé en C++, ce qui a permis une conception complètement modulaire des différentes fonctions présentées dans cet article d'une part, et d'augmenter la vitesse du moteur de corrélation d'autre part. La fonction de regroupement d'alertes a été redéfinie [2]. Au lieu de retourner une valeur vrai ou faux comme dans [13], la nouvelle fonction de similarité retourne un résultat appartenant à l'intervalle [0,1]. La définition de la similarité entre les différents attributs d'une alerte a été affinée. En particulier, le quantum de temps utilisé pour comparer les attributs temporels a été remplacé par un ensemble flou ce qui permet une transition moins brutale lorsque le quantum de temps est écoulé.

Nous continuons les tests sur la fonction d'agrégation et de fusion, en particulier nous voudrions utiliser des modules intégrant cette fonction suivant une architecture hiérarchique. En effet il parait intéressant de tenter de regrouper des ensembles d'alertes similaires générés par exemple périodiquement et reliés à un problème de configuration du système [21]. Pour ce faire nous pensons utiliser un module d'agrégation/fusion prenant en entrée les alertes globales générées par plusieurs autres modules d'agrégation/fusion, mais avec une configuration différente des fonctions de similarité, notamment la fonction de similarité sur le temps.

Les tests se poursuivent également sur des alertes générées par différents MDI. Jusqu'à présent, tous les MDI testés étaient basés sur l'approche par

scénario mais nous projetons d'intégrer également des MDI basés sur l'approche comportementale, en particulier l'approche proposée dans [7]. Pour pouvoir traiter les alertes générées par une telle approche, il nous sera nécessaire de les représenter avec le format IDMEF et de spécifier les classifications à associer à ces alertes. Il sera ainsi possible des les traiter avec le module d'agrégation/fusion et des les corréler si les modèles Lambda correspondant sont définis.

Il existe de nombreuses perspectives à ce travail. Nous en présentons seulement quelques-unes. Comme nous l'avons mentionné dans l'introduction, un objectif important de la corrélation est d'éliminer le plus grand nombre de faux positifs possible. Plusieurs approches complémentaires peuvent être intégrées à CRIM pour contribuer à cet objectif.

Tout d'abord, il est possible de considérer que les alertes globales générées par la fonction de fusion ayant un niveau de crédibilité peu élevé correspondent à des faux positifs. Cette approche peut, par exemple, être utilisée pour reconnaître des MDI utilisant des signatures insuffisamment précises. On peut également envisager « d'apprendre » les faux positifs, en s'inspirant de l'approche suggérée dans [21]. Cet article propose une approche pour identifier, dans un système, des causes générant régulièrement des volumes d'alertes important, par exemple un composant en panne ou un router mal configuré. Cette approche pourrait être intégrée à CRIM pour identifier des paquets d'alertes générées par la fonction de fusion et correspondant à des faux positifs.

Un autre raisonnement est de considérer que les alertes générées par les MDI correspondent en fait à deux types « d'attaques » :

- Les actions malveillantes qui correspondent à des actions qui permettent directement de violer la politique de sécurité.
- Les actions suspicieuses qui ne violent pas directement la politique de sécurité mais qui sont directement ou indirectement corrélées à des actions malveillantes.

Les MDI génèrent, en fait, beaucoup d'alertes qui correspondent à des actions qui sont seulement suspicieuses. Si cette alerte n'est pas ensuite corrélée à d'autres actions suspicieuses ou malveillantes, il est alors possible de considérer que cette alerte pourrait en fait correspondre à un faux positif. Cette démarche nécessite naturellement une analyse adéquate des aspects temporels pour être effective.

Un autre travail en cours, ayant un objectif voisin, consiste à intégrer des informations concernant la topologie et l'état du système surveillé par CRIM. En effet, la plupart des MDI actuels ne sont pas capables de distinguer une attaque réussie d'une tentative d'attaque qui échoue car le système n'est pas dans un état vulnérable à cette attaque. Pour cela, nous envisageons d'intégrer dans CRIM l'approche proposée par [32] concernant la conception du module M2D2. M2D2 propose un modèle qui permet de représenter plusieurs types d'information utiles pour la détection d'intrusion, en particulier des données concernant la topologie du système. L'intégration de M2D2 permettrait donc d'affiner le diagnostic produit par CRIM.

Remerciements

CRIM a été développé d'abord dans le cadre du projet MIRADOR financé par la DGA puis du projet RNTL DICO. Outre les auteurs de cet articles, les personnes suivantes ont participé à la conception et au développement de CRIM : Salem Benferhat, Alexandre Miège et Thierry Sans. Les auteurs souhaitent également remercier Nora Cuppens pour son aide dans l'écriture de cet article et ses nombreuses suggestions d'amélioration.

Références

[1] Autrel (F.), Benferhat (S.), Cuppens (F.), Utilisation de la corrélation pondérée dans un processus de détection d'intrusion. *Annales des Télécommunications*, 2004.

[2] Autrel (F.), Cuppens (F.), Using an Intrusion Detection Alert Similarity Operator to Aggregate and Fuse Alerts. *In the 4th Conference on Security and Network Architectures*, 6-10 June 2005, Batz sur Mer (France).

[3] Cuppens (F.), Autrel (F.), Bouzida (Y.), García (J.), Gombault (S.), Sans (T.), Anti-correlation as a criterion to select appropriate counter-measures in an intrusion detection framework. *Annales des Telecommunications*, n°61, pp 197-217, March 2006.

[4] Bace (R.), Intrusion Detection. *McMillan Technical Publishing*, 2000.

[5] Benferhat (S.), Autrel (F.), Cuppens (F.), Enhanced Correlation in an Intrusion Detection Process. *Second International Workshop Mathematical Methods, Models and Architectures for Computer Networks Security*, St. Petersburg, Russia, September 2003.

[6] Ben Amor (N.), Benferhat (S.), Elouedi (Z.), Naive Bayesian Networks in Intrusion Detection Systems. *Workshop on Probabilistic Graphical Models for Classification*, Cavtat-Dubrovnik, Croatia, September 2003.

[7] Bouzida (Y.), Gombault (S.), Profils propres pour la détection d'intrusion. *Symposium sur la Sécurité des Technologies de l'Information et de la Communication*, Rennes, France, June 2003.

[8] Cuppens (F.), Autrel (F.), Miège (A.), Benferhat (S.), Recognizing Malicious Intention in an Intrusion Detection Process. *Second International Conference on Hybrid Intelligent Systems*, Santiago, Chile, December 2002.

[9] Curry (D.), Debar (H.), *Intrusion Detection Message Exchange Format Data Model and Extensible Markup Language (XML) Document Type Definition* , draft-itetf-idwg-idmef-xml-14.txt, Janvier 2005.

[10] Cuppens (F.), Miège (A.), Alert correlation in a cooperative intrusion detection framework. *IEEE Symposium on Research in Security and Privacy*, Oakland, May 2002.

[11] Cuppens (F.), Ortalo (R.), LAMBDA : A Language to Model a Database for Detection of Attacks. *Proceedings of the Third International Workshop on the Recent Advances in Intrusion Detection (RAID'2000)*, Toulouse, France, October 2000.

[12] Cuppens (F.), Cooperative intrusion detection. *International Symposium on Information Superiority : tools for crisis and conflict-management*, Paris, France, September 2001.

[13] Cuppens (F.), Managing alerts in a multi-intrusion detection environment. *17th ACSAC conference*, New Orleans, December 2001.

[14] Site web de CRIM. *http ://crim-platinum.org/crim/.*

[15] Debar (H.), Wespi (A.), Aggregation and Correlation of Intrusion Detection Alerts. *Workshop on the Recent Advances in Intrusion Detection (RAID'2001)*, Davis, USA, October 2001.

[16] Geib (C.) and Goldman (R.), Plan Recognition in Intrusion Detection Systems, *DARPA Information Survivability Conference and Exposition (DISCEX)*, pp. 46-55, Anaheim, USA, June 2001.

[17] Huang (M.-Y.), A Large-scale Distributed Intrusion Detection Framework Based on Attack Strategy Analysis. *Proceedings of the First International Workshop on the Recent Advances in Intrusion Detection (RAID'98)*, Louvain-La-Neuve, Belgium, 1998.

[18] Garcia (J.), Autrel (F.), Borrell (J.), Castillo (S.), Cuppens(F.), Navarro (G.), Decentralized publish-subscribe system to prevent coordinated attacks via alert correlation. *Proceedings of the Sixth International Conference on Information and Communications Security (ICICS'2004)*, number 3269 in LNCS, October 2004.

[19] Garcia (J.), Autrel (F.), Borrell (J.), Castillo (S.), Cuppens(F.), Navarro (G.), Preventing coordinated attacks via alert correlation. *Proceedings of the Ninth Nordic Workshop on Secure IT Systems Encouraging Cooperation (NORDSEC'2004)*, November 2004.

[20] Gombault (S.), Diop (M.), Response function. *First NATO Symposium on Real Time Intrusion Detection*, Lisbonne, Portugal. May 2002.

[21] Julisch (K.), Clustering Intrusion Detection Alarms to Support Root Cause Analysis. *ACM Transactions on Information and System Security*, Novembre 2003.

[22] Ilgun (K.), USTAT : A real-time intrusion detection system for Unix. *IEEE Symposium on Security and Privacy*, 1993.

[23] Jackson (K.), DuBois (D.), Stalling (C.), An Expert System Application for Network Intrusion Detection. *14th National Computer Security Conference*, October 1991.

[24] Kleinwaechter (J.), The Limitations of Intrusion Detection on High Speed Networks. *Proceedings of the First International Workshop on*

the Recent Advances in Intrusion Detection (RAID'98), Louvain-La-Neuve, Belgium, 1998.

[25] Lee (W.), Combining Knowledge Discovery and Knowledge Engineering to Build IDSs. *Proceedings of the Second International Workshop on the Recent Advances in Intrusion Detection (RAID'99)*, Purdue, USA, October 1999.

[26] Lippmann (R.), Using Key String and Neural Networks to Reduce False Alarms and Detect New Attacks with Sniffer-Based Intrusion Detection Systems. *Proceedings of the Second International Workshop on the Recent Advances in Intrusion Detection (RAID'99)*, Purdue, USA, October 1999.

[27] Lunt (T.), IDES : An Intelligent System for Detecting Intruders. *Computer Security, Threats and Countermeasures*, November 1990.

[28] Mounji (A.), Le Charlier (B.), Continuous Assessment of a Unix Configuration : Integrating Intrusion Detection and Configuration Analysis. *Proceedings of the ISOC'97 Symposium on Network and Distributed System Security*, San Diego, USA, February 1997.

[29] Mann (D.), Christey (S.), Towards a Common Enumeration of Vulnerabilities. *2nd Workshop on Research with Security Vulnerability Databases*, Purdue University, West Lafayette, Indiana, January 1999.

[30] Morin (B.), Debar (H.), Correlation of Intrusion Symptoms : an Application of Chronicles. *Proceedings of the Sixth International Symposium on the Recent Advances in Intrusion Detection (RAID'02)*, Pittsburg, USA, September 2003.

[31] Mé (L.), Marakchi (Z.), Michel (C.), Debar (H.), Cuppens (F.), La détection d'intrusions : les outils doivent coopérer. *Revue de la REE*, 2001.

[32] Morin (B.), Mé (L.), Debar (H.), Ducassé (M.) M2D2 : A Formal Data Model for IDS Alert Correlation. *Fifth International Conference on Recent Advances in Intrusion Detection (RAID'02)*, Zurich, Switzerland, October 2002.

[33] Ning (P.), Cui (V.), Reeves (R. S.), Constructing attack scenarios through correlation of intrusion alerts. *Ninth ACM Conference on Computer and Communications Security*, Washington, D.C. November 2002.

[34] Fyodor, Nmap (Network Mapper) free open source security scanner. *http ://www.insecure.org/nmap/*

[35] Karg (D.), Gil (D.), Casal (J.), Ospitia (F.), Román (A.), Fournier (S.), Lorenzo (J.M.), Open Source Security Information Management. *http ://www.ossim.net*

[36] Porras (P.), Neumann (P.), Emerald : Event Monitoring Enabling Responses to Anomalous Live Disturbances. *National Security Conference*, 1997.

[37] Roesch (M.), Snort - Lightweight Intrusion Detection for Networks. *Proceedings of USENIX LISA'99*, November 1999.

[38] Valdes (A.), Anderson (D.), Statistical Methods for Computer Usage Anomaly Detection. *Third International Workshop on Rough Sets and Soft Computing*, San Jose, USA, 1995.

[39] Valdes (A.), Skinner (K.), Adaptive, Model-Based Monitoring for Cyber Attack Detection. *Proceedings of the Third International Workshop on the Recent Advances in Intrusion Detection (RAID'2000)*, Toulouse, France, October 2000.

[40] Valdes (A.), Skinner (K.), Probabilistic Alert Correlation. *Proceedings of the Fourth International Workshop on the Recent Advances in Intrusion Detection (RAID'2001)*, Davis, USA, October 2001.

[41] Zerkle (D.), A Data-Mining Analysis of RTID. *Proceedings of the Second International Workshop on the Recent Advances in Intrusion Detection (RAID'99)*, Purdue, USA, October 1999.