République Algérienne Démocratique et Populaire
Ministère de l'enseignement supérieur et de la recherche scientifique
Université Larbi Tébessi - Tébessa
Faculté des Sciences Exactes et des Sciences de la Nature et de la Vie
Département : Mathématiques et Informatique

Mémoire de fin d'étude
Pour l'obtention du diplôme de MASTER
Domaine : Mathématiques et Informatique
Filière : Informatique
Option : Réseaux et Securité Inforamatique

Thème

La détection d'intrusion sur le réseau et la sélection des contremesures dans le les systèmes de réseau virtuel

Présenté Par :

Khémaissia Rofaida

Devant le jury :

Mr.Ahmed Ahmim	*MCA*	*Univérsité Larbi Tébessa*	*Président*
Mr Taher Mkhaznia	*MCB*	*Univérsité Larbi Tébessa*	*Examinateur*
Mr Makhlouf Derdour	*MCA*	*Univérsité Larbi Tébessa*	*Encadreur*

Date de soutenance : 23/06/2019

Résumé

Dans le Cloud et les environnements virtuels, la sécurité est considérée comme le facteur le plus important, qui nécessite un système de défense performant en respectant la complexité du Cloud ses dépendances et les compétences des attaquants. Le système de réponse aux intrusions (IRS) est permet de répondre automatiquement aux incidents de la sécurité, par la prise en compte des coûts des contremesures et les dommages d'attaques qui sont considérés comme des limites pour l'IRS. Dans ce mémoire nous avons proposé une architecture pour sélectionner la contremesure optimale en prise en compte plusieurs critères, en estimant le risque à l'aide de la modélisation d'un graphe d'attaque et le graphe de dépendance des composants du Cloud Computing. Afin de sélectionner la contremesure optimale, nous avons proposé la métrique *CRORI* qui prise en compte plusieurs critères : les dépendances entre les contremesures, les dépendances entre les composants dans le Cloud, l'impact négatif d'application de la contremesure, etc. Notre méthode de sélection est implémentée en utilisant JAVA avec plusieurs API Java.

Mots clé : Contremesure optimale, *CRORI*, Graphe d'attaque, Vulnérabilité, Risque Evaluation, CVSS.

Abstract

In the Cloud and virtual environments, security is considered the most important factor, which requires a powerful defense system that respects the complexity of the Cloud's dependencies and the skills of the attackers. The Intrusion Response System (IRS) is used to automatically respond to security incidents, by taking into account the costs of countermeasures and attack damage that are considered limits for the IRS. In this work we proposed an architecture to select the optimal countermeasures taking into account several criteria, estimating the risk using the modeling of an attack graph and the dependency graph of the components of Cloud Computing. In order to select the optimal countermeasures we have proposed the CRORI metric that takes into account several criteria: the dependencies between the countermeasures, the dependencies between the components in the Cloud, the negative impact of the application of the countermeasures, etc. Our selection method is implemented using JAVA with several Java APIs.

Keywords: Cloud Computing, Countermeseaure, Attack Graph, Asset Graph, IRS, CRORI

ملخص

في السحابة والبيئات الافتراضية ، يعتبر الأمن العامل الأكثر أهمية ، والذي يتطلب نظام دفاع قوي يحترم تعقيدات تبعيات السحابة ومهارات المهاجمين. يستخدم نظام الاستجابة التسلل (IRS) للرد تلقائيًا على الحوادث الأمنية ، من خلال مراعاة تكاليف التدابير المضادة وأضرار. في هذه الأطروحة ، اقترحنا بنية لاختيار التدابير المضادة المثلى مع مراعاة العديد من المعايير ، وتقدير المخاطر باستخدام نمذجة الرسم البياني للهجوم والرسم البياني التبعي لمكونات الحوسبة السحابية. لتحديد التدابير المضادة المثلى ، اقترحنا قياس CRORI الذي يأخذ في الاعتبار عدة معايير: التبعيات بين التدابير المضادة ، والتبعيات بين المكونات في السحابة ، والتأثير السلبي لتطبيق التدابير المضادة ، وما إلى ذلك. يتم تطبيق طريقة الاختيار الخاصة بنا باستخدام JAVA مع العديد من واجهات برمجة تطبيقات Java.

الكلمات المفتاحية: السحابة والبيئات الافتراضية ,CRORI

Remerciements

Je souhaiterais tout d'abord remercier à dieu qui m'a donné patience, persévérance et santé pour mener à bien ce modeste travail

Je remercie beaucoup mon chéri mari Ala et ma chérie maman Fatiha qui m'ont toujours soutenu et encouragé ainsi que mon amour et mon fils Arslane, mes frères Malek et Nasro et ma sœur Imen et ses filles.

Je remercie beaucoup également mes belles amies Amira, Rania,Randa,Azhar.

Je remercie Mr Derdour Makhlof pour ses précieux conseils, sa disponibilité, Son encouragement et sa compréhension qui m'a permis de mener à bien ce travail.

Je remercie également les membres de Jury de nous avoir fait l'honneur d'y participer.

Table des matières

Liste des figures

Liste des tableaux

Introduction générale

Au cours des dernières décennies, diverses attaques sur réseau sont émergées. Ce qui a nécessité une réflexion sérieuse pour faire face à ses nombreuses conséquences, ainsi avec le Cloud et les environnements virtuels là où la sécurité est considérée comme le facteur le plus important, la détection d'intrusion ne suffit pas, mais plutôt des techniques permettant de contrer ces attaques qui sont devenues une préoccupation primordiale.

Dans les centres de données (Datacenter) traditionnels, les administrateurs système ont un contrôle total sur les ordinateurs, où les vulnérabilités peuvent être détectées et corrigées par l'administrateur système de manière centralisée. Cependant, dans les centres de données du Cloud, il est très difficile de corriger (patch) des vulnérabilités détectées, car les utilisateurs ont l'accès aux machines virtuelles et le contrôle des logiciels installés, en d'autres termes la violation du contrat de niveau de service (SLA).

Afin de répondre à la problématique de sécurité dans le Cloud, les systèmes actuels de gestion des informations et des évènements de sécurité (SIEM) constituent une plateforme centrale des centres opérationnels de sécurité modernes, en rassemblant les évènements provenant de divers capteurs (systèmes de détection d'intrusion (IDS), pare-feu, antivirus, etc.) et en corrélant ces évènements et en fournissant des vues synthétiques pour la gestion des menaces et la production des rapports de sécurité.

Malgré la performance et la précision des IDSs, plusieurs fausses alertes sont générées tout le temps, en considérant aussi la complexité d'attaque et les compétences des attaquants, en conséquence les IDSs ne représentent pas la solution efficace pour protéger nos systèmes, mais plutôt un composant d'un système d'alarme associé avec d'autres composants.

Pour surmonter les effets de ces attaques en Cloud Computing, un système de réponse à l'intrusion (IRS) en temps réel est nécessaire, qui est destiné à répondre

automatiquement aux incidents par le déploiement des contremesures appropries et adéquates.

La sélection d'une contremesure nécessite plusieurs critères tels que : le coût de déploiement et l'impact négatif sur le contrat de niveau du service qui sont considérés comme des limites qui peuvent conduire l'IRS à choisir des contremesures inappropriées (comme conséquence la réduction des performances du réseau et la déconnexion des utilisateurs légitimes).

Notre projet de fin d'étude consiste à proposer une architecture pour sélectionner la contremesure optimale qui répond à une évaluation précise de l'attaque par une estimation de risque basée sur plusieurs critères qui sont reliés à la contremesure tels que : l'impact négatif, le cout de déploiement, le pourcentage de mitigation de risque, etc.

Après l'introduction générale, ce mémoire comprend quatre chapitres présentés comme suit :

Le premier chapitre décrit les définitions des concepts clés de notre sujet et la présentation des relations entre eux.

Dans le deuxième chapitre, nous avons cité des travaux connexes en indiquant ses limites et ses avantages, et nous récapitulons par une catégorisation et une comparaison entre les différentes propositions.

Pour le troisième chapitre, nous abordons notre démarche proposée afin de répondre à la problématique du sujet qui est la sélection de la contremesure optimale. Une description des différentes de l'architecture proposée est donnée dans ce chapitre. Enfin, notre idée est expliquée à l'aide d'un cas d'étude.

Le dernier chapitre inclut des présentations sur les outils et les environnements utilisés pour le développement, et le déploiement de notre proposition en présentant les fonctionnalités de notre application munie par des interfaces.

Chapitre 1 : Etat de l'art

1.1 Introduction:

La technologie de l'information et la communication se développent ces dernières années pour améliorer les services dans ce domaine qui permet l'apparition d'un nouveau concept il s'agit du Cloud Computing, tel que la virtualisation est un élément fondamental du Cloud Computing, en particulier pour la fourniture d'infrastructure en tant que service (IaaS).

Le Cloud Computing est un nouveau paradigme qui fournit aux utilisateurs un utilitaire informatique via Internet à l'aide d'un navigateur Web. Ce dernier est un nouveau concept informatique qui consiste à proposer des services informatiques sous forme de services à la demande à un bassin partagé de ressources informatiques configurables et accessibles. Cette technologie permet aux entreprises de traiter une quantité d'information énorme, la sécurité et la confidentialité des services dans le Cloud sont des questions essentielles à prendre en compte.

Dans ce chapitre nous présentons le concept de la virtualisation, Cloud Computing, ses vulnérabilités, la détection des intrusions dans le Cloud et enfin les systèmes de réponses aux intrusions par des contre-mesures appliquées pour une meilleure sécurisation.

1.2 La virtualisation

La virtualisation est un mécanisme informatique qui présente l'ensemble des techniques matérielles et/ou logiciels qui permettent de faire fonctionner sur une seule machine plusieurs systèmes d'exploitation, plusieurs instances différentes et cloisonnées d'un même système ou plusieurs applications, séparément les uns des autres, comme s'ils fonctionnaient sur des machines physiques distinctes [18].

1.2.1 La virtualisation des services

La virtualisation des services consiste à créer des répliques des systèmes dont dépendent les nouvelles applications, afin de tester le bon fonctionnement de l'intégration entre ces applications et ces systèmes. On l'utilise principalement pour

intégrer des applications qui dépendent des architectures du Cloud et services orientées (SOA, service-oriented architecture) ou qui communiquent avec les données et les interfaces de programmation d'applications (API) de tiers. Les systèmes en question sont, par exemple, les services de gestion de la relation client (CRM, Customer relationship management) tels que Salesforce Service Cloud, les services de planification des ressources (ERP, entreprise ressource planning) tels que SAP ECC, et des systèmes internes qui sont en cours de développement [46].

1.2.2 La virtualisation des machines

Une machine virtuelle, ou VM (Virtual Machine), est un environnement d'application ou de système d'exploitation (OS, Operating System) installé sur un logiciel qui imite un matériel dédié. Côté utilisateur final, l'interaction avec une machine virtuelle est la même qu'avec un matériel dédié. Un logiciel spécialisé, appelé hyperviseur, émule intégralement les différentes ressources matérielles d'un serveur ou d'un PC client, telles que l'unité centrale, la mémoire, le disque dur ou le réseau, et permet à des machines virtuelles de les partager. L'hyperviseur peut émuler plusieurs plateformes matérielles virtuelles isolées les unes des autres. Il permet ainsi à des machines virtuelles d'exécuter les systèmes d'exploitation serveur Linux et Windows sur le même hôte physique sous-jacent [26].

1.2.3 La virtualisation des réseaux:

La virtualisation du réseau est la reproduction logicielle complète d'un réseau physique, la virtualisation de réseau est le processus consistant à combiner des ressources réseau matérielles et des ressources logicielles en une seule unité administrative. La virtualisation de réseau a pour objectif de fournir aux systèmes et aux utilisateurs un partage efficace, contrôlé et sécurisé des ressources réseau. Le produit final de la virtualisation du réseau est le réseau virtuel [11].

La virtualisation est le concept clé du modèle de Cloud Computing, Il permet aux programmes d'être portables sur plusieurs plates-formes et de gérer leur évolutivité, leur surveillance et leur sécurité. En profondeur, les réseaux virtuels augmentent l'inter connectivité des machines virtuelles, qui représente un défi important de sécurisation dans le Cloud Computing.

1.3 Le Cloud Computing

Le Cloud en français «le nuage» selon NIST (National Institute of Standards and Technology) le Cloud Computing: " est l'ensemble des disciplines, pratiques, technologies et modèles commerciaux utilisés pour délivrer comme un service à la demande et par le réseau des capacités informatiques (logiciels, plateformes, matériels) " [34].

Le groupe de travail de CIGREF [20] est intéressé aux fondamentaux du Cloud Computing par définir ce concept en quatre points :

1. Un Cloud est toujours un espace virtuel.
2. Contenants des informations qui sont fragmentées
3. Les fragments sont toujours dupliqués et répartis dans cet espace virtuel, lequel peut être sur un ou plusieurs supports physiques.
4. Et qui possède "une console (programme) de restitution" permettant de reconstituer l'information.

1.3.1 Caractéristiques du Cloud Computing

Le Cloud Computing se divise en cinq caractéristiques qui suivent:

1. **Accès réseau universel** : Un environnement de type Cloud Computing s'appuie obligatoirement sur le réseau (internet) et est accessible via le réseau, quel que soit le périphérique (PC, Mac, TV, Tablette, Smartphone). [3]
2. **Mise en commun de ressources :** En anglais, le pooling. Dans un environnement de type Cloud on ne pense pas en nombre de serveurs, taille de disques ou nombre de processeurs, mais en puissance de calcul, capacité totale de stockage, bande passante disponible. Les ressources de calcul sont mises à disposition des clients sur un modèle multilocataire, avec une attribution dynamique des ressources physiques et virtuelles en fonction de la demande. Le client n'a généralement aucun contrôle ou connaissance sur l'emplacement exact des ressources fournies. Toutefois, le client peut imposer de spécifier l'emplacement à un niveau plus haut d'abstraction (par exemple le pays, l'état ou le centre de données) [35].
3. **Élasticité** : Grâce au Cloud, il est possible de disposer de plus de ressources très rapidement pour soutenir une forte demande (par exemple : pour garantir

une bonne expérience d'achat aux clients sur une plateforme Web de e-commerce durant les fêtes de fin d'année). Inversement, au-delà d'approvisionner des ressources, il est possible avec le Cloud de diminuer les ressources utilisées (ex: en cas de baisse d'activité sur cette même plateforme Web de e-commerce) si celles-ci sont supérieures à ce qui est réellement nécessaire [3].

4. **Libre-Service :** Dans un environnement de type Cloud Computing, il est possible à un utilisateur de consommer les services ou ressources sans pour autant devoir faire une demande d'intervention auprès de son fournisseur : équipe IT ou fournisseur externe (ex : un développeur qui souhaite tester son application sur une machine virtuelle représentative d'un poste standardisé de son entreprise peut, seul et au travers d'un portail Web, provisionner et utiliser une machine sans devoir solliciter l'équipe IT). C'est la notion de self-service [3].
5. **Service mesurable et facturable :** Dans un environnement de type Cloud le fournisseur de la solution de Cloud est capable de mesurer de façon précise la consommation des différentes ressources (CPU, Stockage, bande passante…); cette mesure lui permet ensuite de facturer le client selon l'usage [3].

1.3.2 Les modèles de services de Cloud computing

Le NIST dénombre 3 modèles comme de service [4]:

- **Cloud Software as a Service (SaaS)** : l'utilisateur a la possibilité d'utiliser les applications du fournisseur de services via le réseau. Ces applications sont accessibles via différentes interfaces, clients légers, navigateur Web, terminaux mobiles… Le client ne gère et ne contrôle pas l'infrastructure Cloud sous-jacente, incluant le réseau, les serveurs, les systèmes d'exploitation, les bases de données, le stockage, mais il peut éventuellement bénéficier d'accès à des configurations restreintes, spécifiques à des catégories d'utilisateurs.
- **Cloud Platform as a Service (PaaS)** : le consommateur peut déployer sur l'infrastructure Cloud ses propres applications, dans la mesure où le fournisseur supporte le langage de programmation. L'utilisateur gère et ne contrôle pas l'infrastructure Cloud sous-jacente (réseau, serveurs, systèmes d'exploitation, bases de données, stockage), mais a le contrôle sur les

applications déployées et la possibilité de configurer l'environnement d'hébergement applicatif.

- **Cloud Infrastructure as a Service (IaaS)** : le client peut louer des capacités de traitement, de stockage, de réseau et autres ressources de calcul. L'utilisateur ne gère et ne contrôle pas l'infrastructure Cloud sous-jacente, mais a le contrôle sur les systèmes d'exploitation, les bases de données et les applications déployées.

- **Les avantages et les inconvénients de services du Cloud :**

Types	Avantages	Inconvénients
Saas	▪ Pas d'installation. ▪ Plus de licence. ▪ Migration.	▪ Logiciel limite. ▪ Besoin de sécurité. ▪ Dépendance du prestataire. ▪ Services dédies.
Paas	▪ Ne nécessite pas d'infrastructure. ▪ Pas d'installation. ▪ Environnement hétérogène.	▪ Limitation des langages. ▪ Pas de personnalisation dans la configuration des machines virtuelles
Iaas	▪ Administration. ▪ Personnalisation. ▪ Flexibilité d'utilisation	▪ Besoin de sécurité. ▪ Besoin d'un administrateur Système.

Tableau. 1.1 Avantages et Inconvénients des services de Cloud computing [17].

1.4 Les intrusions dans le système du Cloud:

Cette section illustre des intrusions courantes, qui entraînent des problèmes de disponibilité, de confidentialité et d'intégrité des ressources et des services dans le Cloud computing :

1.4.1 La vulnérabilité

Dans la sécurité informatique, une vulnérabilité est une faille ou une faiblesse dans un réseau qui peut être exploitée par une ou plusieurs menaces afin de violer la

politique de sécurité du système. Elle est aussi définie comme une erreur logicielle qui peut être exploitée par l'intrus afin de gagner l'accès au système ou au réseau [22].

Chaque vulnérabilité possède un score qui représente le CVSS (Common Vulnerability Scoring System) qui permet de fournir un moyen de capturer les principales caractéristiques de la vulnérabilité, afin de produire un score numérique en reflétant sa gravité. Le score numérique peut être ensuite traduit en une représentation qualitative (telle que faible, moyenne, élevée et critique) pour but d'aider les organisations à évaluer et à hiérarchiser correctement leurs processus de gestion des vulnérabilités .Ce score est construit sur trois groupe de métrique [8] :

- Groupe de base : représente les caractéristiques intrinsèques d'une vulnérabilité qui sont constantes dans le temps et dans tous les environnements utilisateur
- Groupe temporel : qui reflète les caractéristiques d'une vulnérabilité qui peut être évolué dans le temps, mais pas dans les environnements des utilisateurs.
- Groupe environnement : représente les caractéristiques d'une vulnérabilité qui sont pertinentes et uniques pour l'environnement d'un utilisateur particulier.

Donc on va concentrer sur le premier groupe qui est le groupe de base, car le CVSS a dépendu principalement sur ce groupe, et qui est considéré comme un score total de la vulnérabilité. Cette métrique est calculée par la dépendance des certains sous-groupes, on va les définir :

- Vecteur d'accès : le vecteur d'accès spécifie la méthode d'accès à la vulnérabilité, ces méthodes peuvent être locales, distantes et adjacentes. Si l'exploitation de la vulnérabilité nécessite un accès local à la machine cible, la valeur du vecteur d'accès devient locale, lorsqu'il est obligatoire que l'attaquant soit présent dans le réseau local, la valeur du réseau adjacent est utilisée. La valeur distante concerne la vulnérabilité exploitable à distance.
- Complexité d'accès: Cette caractéristique spécifie la complexité de l'attaque requise pour exploiter la vulnérabilité sur la machine cible. Si une vulnérabilité nécessite que la victime interagisse avec le mécanisme d'attaque, la complexité de l'accès est définie sur le niveau d'interaction requis; il peut être faible, moyen ou élevé.

- Authentification: pour lancer une attaque, le niveau d'authentification requis est représenté par cette mesure. Les valeurs pouvant être définies pour cette métrique sont multiples, uniques ou nulles. Si l'authentification est requise autant de fois que l'attaquant lance l'attaque, une valeur «multiple» est définie pour la métrique. De manière similaire, d'autres valeurs sont également définies en fonction de différents paramètres.
- Impact de confidentialité (C): l'impact de confidentialité est l'impact sur la confidentialité des ressources de la victime. Sa valeur peut être complet, partiel ou aucun. Si l'attaquant obtient l'accès à tous les fichiers système, alors la valeur est complète. S'il parvient à accéder à certains fichiers, il est partiel, sinon, il n'en reste aucun.
- Impact de l'intégrité (I): l'impact de l'intégrité mesure le degré d'intégrité du système affecté par une attaque. L'intégrité est la fiabilité de l'information. Cette valeur de métrique peut être complète, partielle ou aucune.
- Impact sur la disponibilité (D): l'impact sur la disponibilité est l'impact sur la disponibilité des fichiers et des services, etc. Cette valeur de métrique peut être complète, partielle ou nulle. Si une attaque arrête un service, la valeur de l'impact sur la disponibilité devient complète. Si elle réduit les performances du service, elle est partielle et, en l'absence d'effet, la valeur est none.

1.4.2 Les attaques dans le Cloud Computing

1.4.2.1 Les attaques internes :

On peut considérer plusieurs attaques internes du Cloud par des utilisateurs autorisés, pour corrompre ou bien tenter à avoir des privilèges non autorisés. On a par exemple le service d'Amazone Elastic Compute Cloud(EC2), c'est un service web, qui permet aux utilisateurs de profiter de ressource avec une capacité de calcule énorme, on peut citer quelque attaque interne selon [24] . On va commencer par AMIBomb : les utilisateurs ont le droit de créer leur propre AMI (Amazone Machine Instance) .

Les utilisateurs malveillants profitent de cette opportunité pour créer un DDoS interne dans le service EC2 par l'exécution de plusieurs machines AMIES en même temps. On ajoute aussi AMI course d'inscription, en général c'est la création des AMIs malveillants et les ajouter aux listes publiques pour les télécharges et exécutés leur code malicieux.

1.4.2.2 Les attaques par inondation:

L'attaquant tente d'inonder la victime en envoyant un très grand nombre de paquets d'un hôte innocent (zombies) sur le réseau. Les paquets peuvent être de type TCP, UDP, ICMP ou bien un mélange. En cas de Cloud, les demandes de VM sont accessibles à tous via Internet, ce qui peut provoquer une attaque par déni de service distribué (DDoS) via des zombies. L'attaque par inondation affecte la disponibilité du service pour les utilisateurs autorisés.

1.4.2.3 Attaque par l'utilisation des racines :

Un attaquant obtient un accès au compte d'un utilisateur légitime, en reniflant (sniffing) son mot de passe. Cela le rend capable d'exploiter les vulnérabilités pour obtenir un accès à la racine du système. Les débordements de mémoire sont utilisés pour générer des shells de racine à partir d'un processus exécuté en tant que racine.

1.4.2.4 Attaques sur les machines virtuelles ou un hyperviseur :

En compromettant l'hyperviseur de couche inférieure, l'attaquant peut prendre le contrôle des ordinateurs virtuels installés. Par exemple. BLUEPILL [19], SubVir [37] et DKSM [39] sont des attaques bien connues sur la couche virtuelle. Grâce à ces attaques, les pirates peuvent compromettre un hyperviseur installé pour prendre le contrôle de l'hôte. Des nouvelles vulnérabilités, telles que la vulnérabilité Zero Day se trouvent dans les ordinateurs virtuels qui incitent un attaquant à accéder à l'hyperviseur ou à d'autres ordinateurs virtuels installés.

1.4.2.5 Attaques par canaux des portes dérobées (backdoors) :

Il s'agit d'une attaque passive permettant aux pirates d'obtenir un accès distant au nœud infecté afin de compromettre la confidentialité de l'utilisateur. En utilisant des canaux de porte dérobée, les pirates peuvent contrôler les ressources de la victime et les compromettre en tant que zombie pour lancer une attaque par DDoS. Il peut également être utilisé pour divulguer les données confidentielles de la victime. De ce fait, le système compromis rencontre des difficultés pour effectuer ses tâches habituelles.

1.5 Les systèmes de détection d'intrusion (IDS) dans le Cloud

Dans le Cloud computing les techniques traditionnel des systèmes de détection et même de prévention d'intrusion sont aussi appliqués, en général se sont deux approches : l'approche de détection basée sur la signature, l'approche de détection comportementale. En plus il y a d'autres nouvelles techniques de détections qui sont appliquées dans le Cloud comme l'approche de détection qui base sur l'intelligence artificielle.

1.5.1 L'approche de détection basée sur la signature

La détection d'intrusion basée sur la signature permet de définir un ensemble de règles ou un ensemble de signatures, ou une base de connaissances prédéfinie pouvant être utilisée pour décider qu'un motif donné est un intrus ou non. En conséquence, les systèmes basés sur les signatures sont capables d'atteindre des niveaux de précision élevés, et un nombre minimal de faux positifs lors de la détection d'intrusions, même très subtiles. Une faible variation des attaques connues peut également affecter l'analyse si un système de détection n'est pas configuré correctement [9].

L'une des raisons motivant l'utilisation de la détection par signature est la facilité de maintenance et la mise à jour des règles préconfigurées, par exemple, dans le système SNORT les parties d'une signature sont l'en-tête (adresse source, adresse de destination, ports) et ses options (charge utile (payload), métadonnées), qui permettent de déterminer si le trafic réseau correspond à une signature connue.

Dans le Cloud, la technique de détection d'intrusion basée sur la signature peut être utilisée pour détecter une attaque connue. Il peut être utilisé soit à l'extrémité avant du Cloud pour détecter les intrusions externes, soit au back end du Cloud. Les nouvelles approches utilisent un système de détection d'intrusion basé sur la signature pour détecter les intrusions au niveau des machines virtuelles (VM).

1.5.2 L'approche de détection basée sur le comportement (anomalie) :

Cette approche concerne l'identification des évènements qui semblent anormaux par rapport au comportement habituel du système qui implique la collecte des données

relatives au comportement des utilisateurs légitimes dans une période donnée, puis l'application des tests statistiques au comportement observé, qui détermine si ce comportement est légitime ou non.

Cette technique peut être utilisée dans le Cloud pour détecter des attaques inconnues aux différents niveaux, car dans le Cloud, un grand nombre d'évènements se produisent (au niveau réseau ou système), ce qui rend leur surveillance ou leur contrôle difficile en basant sur la technique de détection d'anomalie (comportementale). [44], [21], [49] ont proposé des approches pour la détection d'anomalie au niveau des différentes couches du Cloud.

1.5.3 L'approche de détection basée sur le réseau de neurones artificiels (RNA)

L'objectif de l'utilisation des RNAs pour la détection des intrusions pour pouvoir généraliser des données à partir de données incomplètes et classer les données comme normales ou intrusives, les types de RNAs utilisés dans l'IDS sont les suivants selon Ibrahim [23]: réseaux de neurones à couches multiples (MLFF), perceptron à couches multiples (MLP) et rétro-propagation (BP).

1.6 Les systèmes de réponse aux intrusions

Les mécanismes de sécurité, tels que les pare-feu, l'authentification, la cryptographie et les contrôles d'accès dans le Cloud computing sont utilisés comme première ligne de défense contre les problèmes de sécurité, qui les rendre incapables de détecter les intrusions internes et fournir des contremesures de sécurité inadéquates. Comme solution les IDS et les IPS sont intégrés pour détecter et prévenir les intrusions potentielles en temps réel. Cependant, les IPS nécessitent des systèmes à hautes performances et sont difficiles à analyser et à prévenir les intrusions en particulier dans un environnement difficile. Ainsi, les contremesures de sécurité surveillant en permanence les performances du système est elles sont nécessaire pour identifier et gérer efficacement les incidents potentiels. Ces contremesures sont utilisées par des systèmes appelées les systèmes de réponse aux intrusions (IRS) figure 1.1.

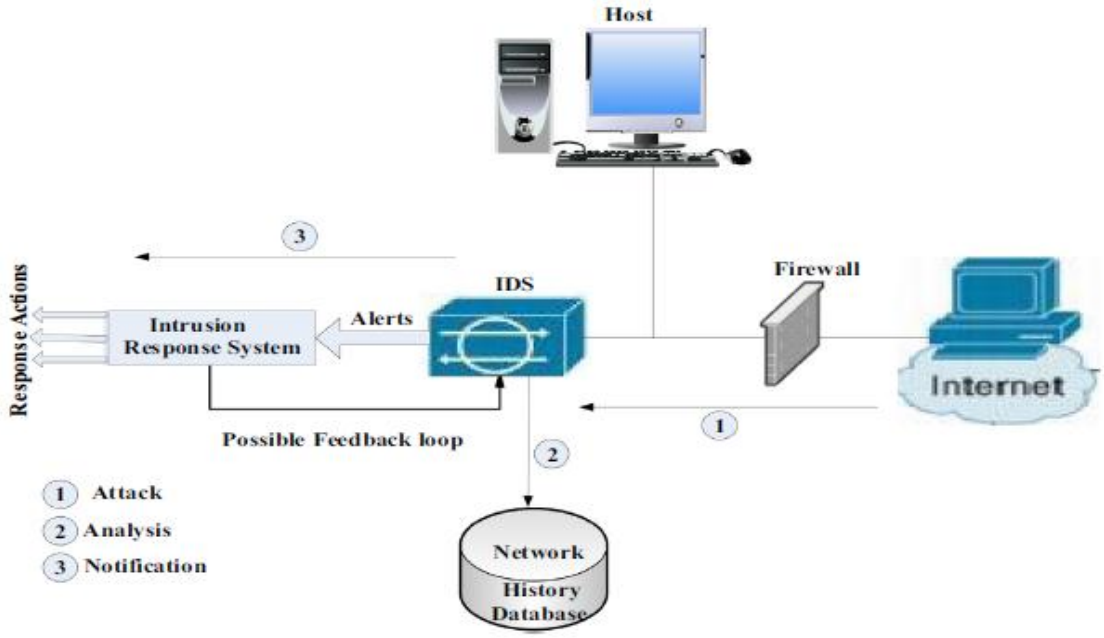


Figure 1.1 : le système de réponse aux intrusions [52]

En fonction des alertes des IDSs les IRSs surveillent en permanence la santé du système, afin que les activités malveillantes ou non autorisées soient gérées efficacement en appliquant des contremesures appropriées, pour empêcher l'aggravation des problèmes et redonner au système un mode de fonctionnement sain. Les IRSs peuvent être classés en trois catégories :

1.6.1 Les systèmes de notification

Ces systèmes génèrent principalement des alertes lorsqu'une attaque est détectée. Une alerte peut contenir des informations sur l'attaque, tel que sa description, l'heure de l'attaque, l'adresse IP source, le compte d'utilisateur, etc. Les alertes sont ensuite utilisées par l'administrateur pour sélectionner les mesures réactives à appliquer, le cas échéant. Les systèmes de notification fournissent principalement des informations sur l'intrusion qui sont ensuite utilisées par l'administrateur système pour sélectionner une réponse à une intrusion. La majorité des systèmes IDS existants fournissent un mécanisme de réponse aux notifications [40]. Le défi majeur de cette approche est le délai entre l'intrusion et la réponse humaine.

1.6.2 Les systèmes de réponse manuelle

Le système de notification est inadéquat, car il génère des alertes lors de la détection d'attaques. Dans ce type de système, l'administrateur applique un ensemble prédéterminé (préconfiguré) de réponses sur les symptômes des attaques pour générer une réponse manuelle. Cette approche est hautement automatisée par rapport à l'approche de notification [52]. Des études montrent que de nombreuses IRS existantes utilisent un système manuel (passif) comme approche de réponse.

1.6.3 Les systèmes de réponse automatique

Ces systèmes sont entièrement automatisés, de sorte qu'aucune intervention humaine ne soit requise, contrairement aux deux méthodes précédentes, qui laissent un intervalle de temps comme une vulnérabilité entre la première réponse et l'intrusion détectée. L'un des problèmes majeurs de cette approche est la possibilité qu'une réponse inappropriée soit apportée et exécutée en conséquence. Un autre défi lié à l'exécution d'une réponse automatisée consiste à s'assurer que la réponse est adéquate pour neutraliser l'attaque.

1.7 Les techniques et les méthodes de réponse aux intrusions

À cause de l'augmentation de la complexité et la propagation rapide de l'attaque aux dernières années qui montrent la nécessité d'un système très performant et résistant pour répondre aux intrusions, dans cette partie on va présenter les techniques utilisées dans les IRSs et les approches associées qui sont montrées dans la figure 1.2.

1.7.1 Le niveau d'automatisation

On a discuté précédemment les trois types des IRSs, qui sont : le système de notification, le système manuel et le système automatique. Ce dernier nous permet de catégoriser les IRSs selon leur coût de réponse, le temps de réponse, capacité de réglage, la sélection des réponses, ou par Lieu d'application, et enfin par Capacité de désactivation.

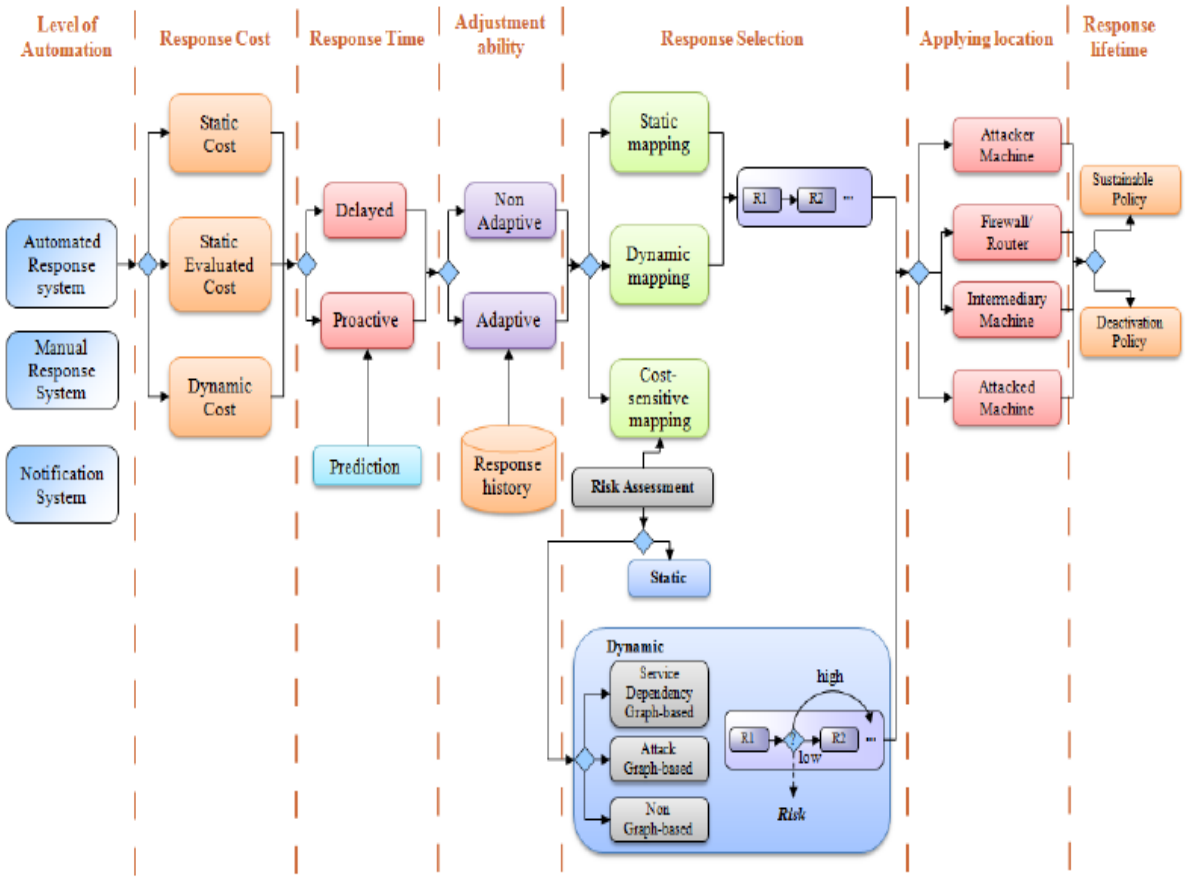


Figure 1.2 : taxonomie du système de réponse automatique [42]

1.7.1.1 Modèle de coût de réponse

Le coût de la réponse est l'impact de l'application de la réponse dans notre réseau. En termes de services réseau continus et de besoins des utilisateurs. Bien que la réponse forte comme la désactivation démon ait une forte capacité à atténuer les attaques et à protéger notre réseau, a un impact très important sur la continuité de service réseau et les utilisateurs en ligne [42].

L'évaluation des coûts de réponse est une partie importante de l'IRS. IRS peut être évalue sa réponse en trois modèles : coût statique, coût statique évalué ou bien un coût dynamique. La plupart des systèmes IRSs automatiques utilisent l'évaluation statique de la réponse, évitant ainsi le recours à une évaluation dynamique. Le modèle statique est un défi, car des paramètres précis sont nécessaires pour évaluer la réponse (contremesure). Les réponses ne peuvent pas être évaluées sans prendre en compte les attaques elles-mêmes dans le sens de la confidentialité, l'intégrité des données et la disponibilité du service, on va présenter les trois modelés :

1.7.1.1.1 Modèle du coût statique

Le coût de réponse statique est obtenu en assignant une valeur statique en basant sur l'opinion d'un expert. Donc, dans cette approche, pour chaque réponse une valeur statique est considérée telle que : ($CR_s = Constante$).

1.7.1.1.2 Modèle de coût statique évalué

Un coût statique évalué, obtenu par un mécanisme d'évaluation, est associé à chaque réponse telle que ($CR_{se} = f(x)$) . Le modèle de coût statique évalué permet de classer les IRSs selon le temps de réponse en deux catégories : les IRSs différés (delayed) et les IRSs proactifs (proactive).

1.7.1.1.3 Modèle de coût dynamique

Le coût dynamique (CR_{dy}) est basé sur la situation du réseau. Nous pouvons évaluer le coût de la réponse en ligne en fonction des dépendances entre ressources et utilisateurs en ligne.

1.7.1.2 Le temps de réponse

Ce système peut être divisé en deux catégories, L'action de réponse de la première catégorie est retardée jusqu'à la confirmation de l'attaque. En assurant par des mesures de confiance de l'IDS. La majorité des IRSs existants utilisent cette approche, parmi eux ils existent les approches de [25] et [6]. Bien qu'il soit inefficace pour une sécurisation maximale. Comme exemple un attaquant tente d'accède au base de donnes (confidentialité) avant que l'IDS puisse détecte-le, dans ce cas la réponse n'est plus utile, et incapable de corriges l'état du système. Par contre au deuxième système la réponse proactive permet de prévoir les réponses (contre-mesures) avant que l'attaque ça arrive. La prévision est généralement difficile, elle repose souvent sur des mesures de probabilité et l'analyse du comportement actuel des utilisateurs ou bien les systèmes. Les chercheurs dans [31] ont proposés un IRS proactif.

1.7.1.3 Capacité de réglage (*Adjustment ability*)

Il existe deux modelés pour la capacité de réglage : adaptatif et non-adaptatif. On va commencer par le modèle adaptatif, le système peut ajuster automatiquement et de manière appropriée l'ordre des réponses en fonction de leur historique. Par contre le

modèle non-adaptatif, l'ordre des réponses reste le même pendant la vie de l'IRS. En fait, il n'existe aucun mécanisme permettant de suivre les comportements des réponses déployées. En fait selon [41] chaque réponse est identifié par une métrique de sucées est un paramètre dynamique qui représente l'historique des succès (S) et des échecs (E) de chaque réponse dans un hôte spécifique, ce paramètre garantit que le modèle sera adaptatif et aidera l'IRS à préparer les meilleures réponses.

1.7.1.4 La sélection des réponses (*response selection*)

Il existe trois modèles de sélection de réponses, on va les présenter :

1.7.1.4.1 Modèle du mappage statique

Ces systèmes sont faciles à construire. Une alerte est mappée sur une réponse prédéfinie, donc le principal inconvénient est que la réponse peut être prévisible par l'attaquant.

1.7.1.4.2 Modèle du mappage dynamique

Les auteurs de [1] ont défini que les réponses du mappage dynamique sont basées sur plusieurs facteurs, tels que l'état du système, les métriques d'attaque (fréquence, gravité, confiance, etc.) et la stratégie de réseau. En d'autres termes, les réponses à une attaque peuvent différer, en fonction de l'hôte ciblé. La contremesure optimale est choisie dynamiquement parmi un ensemble de réponses en fonction de caractéristiques de l'attaque. L'un des inconvénients de ce modèle est qu'il n'apprend rien d'une attaque à l'autre, de sorte que le niveau de l'intelligence reste le même jusqu'à la prochaine mise à niveau (upgrade).

1.7.1.4.3 Modèle du mappage sensible aux coûts

Cette technique tente de s'accorder entre le coût de dommage d'intrusion et le coût de réponse. Certaines approches comme [4], [27] et [32] utilisent une évaluation des risques hors ligne (statique), en attribuant aux ressources des valeurs statiques qui sont évaluées par l'avance. Par contre l'évaluation des risques en ligne (dynamique) permet de mieux préciser le coût de dommage d'intrusion. Comme contre-exemple si l'IRS effectue une évaluation inexacte comme conséquence l'impact de la réponse peut être très élevé si l'attaque est faible ou vice versa.

Le travail de [42] a catégorisé les approches d'évaluation dynamique des IRSs en trois : le graphe d'attaque, le graphe de dépendance des services, et le non-graphe.

a. Le graphe d'attaque

Un graphe d'attaque est un outil de modélisation permettant d'illustrer tous les chemins d'attaque multi-étapes et multi-hôtes possibles, qui sont essentiels pour comprendre les menaces, puis pour décider des contremesures appropriées [48].

b. Le graphe de dépendance des services

Trois propriétés sont définies pour chaque service: C (S), I (S) et D (S), qui indiquent respectivement la confidentialité, l'intégrité et la disponibilité du service (S). L'impact de l'attaque sur un service est propagé à d'autres services en fonction du type de dépendance [42].

c. Le non-graphe

L'évaluation des risques est réalisée indépendamment de l'attaque détectée par l'IDS. Cela signifie que l'IDS détecte une attaque et envoie une alerte au composant d'évaluation des risques, qui effectue une analyse de risque sur la base des statistiques d'alerte et d'autres informations fournies dans l'alerte [5].

1.7.1.5 Lieu d'application (*Applying location*)

La majorité des IRS appliquent des réponses au niveau de la machine attaquée (la cible) ou sur la machine des intrus (attaquant) si elle est accessible, par l'extraction du "chemin d'attaque", qui se compose [42] de :

1. Le point de départ : la machine de l'intrus.
2. Le point pare-feu : qui comprend les pare-feu et les routeurs.
3. Le point médian, qui inclut toutes les machines intermédiaires exploitées par l'intrus pour compromettre l'hôte cible.
4. Le point final : la machine cible des intrus

Parmi les travaux par le chemin d'attaque existant [50], [51].

1.7.1.6 Capacité de désactivation (*Deactivation ability*)

Les auteurs de [47] ont proposé cette technique qui permet de désactiver la réponse, car elle est une action temporaire peut être influencé sur l'état global du système de contrôle, la question qui se pose quand et où on peut désactiver cette réponse.

1.8 Conclusion

Au cours de ce chapitre, nous avons essayé de faire un survol sur les principales clés concepts : le Cloud computing, les IDSs et les IRSs. Les IDSs ne sont pas capables de mitiger la propagation des attaques au niveau des couches du Cloud, l'IRS devient un composant important pour la détection des intrusions en choisissant des contremesures appropriées pour que le système récupère sa santé.

Chapitre 2 : Travaux Connexes

2.1 Introduction

Lors de la conception d'un système IRS sensible aux coûts, plusieurs modèles ont été proposés pour harmoniser et combiner entre les dommages d'attaque et les contremesures, mais ils n'utilisent pas la même unité de mesure. Dans ce chapitre on va présenter quelques travaux connexes, qui ont été basé sur la sélection des contremesures par l'évaluation d'une métrique.

2.2 Présentation des travaux connexes

Dans cette section nous présentons un ensemble de travaux antérieurs sur la sélection des contremesures :

2.2.1 La sélection des contremesures basant sur l'analyse du graphe d'attaque

Les auteurs de [7] ont proposé NICE" *Network Intrusion Detection and Countermeasure selection*" pour établir une défense en profondeur dans le cadre de détection d'intrusion. Ce système permet d'analyser les graphes d'attaque afin de détecter les intrusions. NICE comporte 3 modes figure 2.1 : la modélisation de menace, le modèle du graph d'attaque et le modèle de protection de la machine virtuelle (VM). Cette solution peut être déployée dans un système de Cloud computing est précisément dans le modèle de service infrastructure en tant que service (*IaaS*).

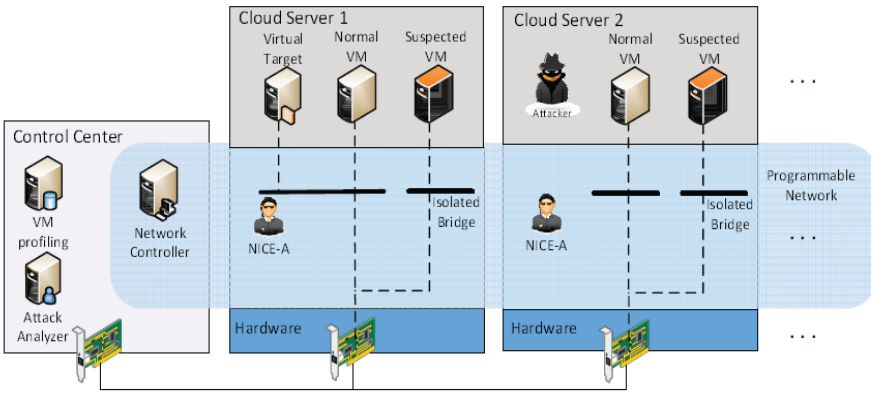


Figure 2.1 : représente l'architecture de NICE [7]

La figure 2.1 montre NICE dans un cluster des serveurs en Cloud. Les principaux composants de cette infrastructure sont : un agent NICE-A sur chaque serveur physique de Cloud, un contrôleur de réseau, un serveur de profils de machine virtuelle et un analyseur d'attaque.

NICE-A : un agent logiciel implémenté dans chaque serveur Cloud connecté au centre de contrôle via un canal sécurisé dédié et isolé. Il analyse le trafic circulant entre les machines virtuelles et les entrées / sorties des serveurs physique de Cloud.

Contrôleur : Le contrôleur de réseau est responsable du déploiement des contre-mesures en fonction des décisions prises par l'analyseur d'attaques et la fonctionnalité de reconfiguration de réseau virtuel basée sur le protocole Open-Flow. Le contrôleur de réseau est chargé de collecter les informations sur le réseau actuel et fournit des données à l'analyseur d'attaque pour construire le graphe d'attaque.

Serveur de profils du VM: permet d'obtenir des informations précises sur leur état, les services en cours, et les ports ouverts. La connaissance des services fonctionnant sur une VM permet de vérifier l'authenticité des alertes relatives à cette VM. Les informations sur les ports ouverts d'une machine virtuelle et leur historique ouvert jouent un rôle important pour déterminer ses vulnérabilités.

Analyseur d'attaque : l'analyseur d'attaque évalue la gravité de l'alerte en fonction du graphe d'attaque, et prendre des stratégies de contremesures, qui représente les fonctions principales du système NICE. L'analyseur d'attaque comprend des procédures telles que la construction et la mise à jour des graphes, la corrélation des alertes et la sélection des contremesures optimales.

Dans cette approche le *ROI* (Return On Investement) est la métrique utilisée pour choisir la contremesure optimale, on peut la présenter avec l'équation (2.1) :

$$ROI = \frac{benifit[t, cm]}{cost.cm + intrusiveness.cm} \qquad (2.1)$$

$benifit$: représente l'avantage de déployer une contremesure cm en temps t.

$cost$: est l'unité qui décrit les dépenses nécessaires à l'application de la contremesure en termes de ressources et de complexité opérationnelle. Elle est définie dans une plage allant de 1 à 5, le numéro 5 signifie un coût plus élevé.

intrusiveness : est l'effet négatif qu'une contremesure apporte à l'accord de niveau de service (SLA).

Cette approche souffre par plusieurs problèmes, premièrement le problème posé est se considère comme une optimisation multi-objectifs, par contre ils le réduit par une seule optimisation objective qui donne une solution unique. Deuxièmes, les chercheurs ont définis l'efficacité par le pourcentage de changement de probabilité d'un nœud, pour lequel la contremesure est appliquée, ce qui signifie qu'il est dynamique. Bien qu'ils sont utilisés ses expérimentations par des valeurs statiques. Troisièmes, ils ne considèrent pas l'effet d'une contremesure sur une autre contremesure.

2.2.2 La sélection des contremesures basant sur l'analyse du graphe de dépendance des services

Les auteurs de [43] ont proposés une plateforme de défense dynamique qui sélectionne les contremesures optimales. Ils ont proposé un nouveau modèle de défense centrale qui est basé sur un graphe de dépendance des services pour sélectionner dynamiquement la contremesure optimale. Ils formulent le problème à l'aide d'un concept d'optimisation objective multiple qui optimise les avantages en matière de sécurité, qui minimise l'impact négatif sur les utilisateurs et les services et minimise les coûts de déploiement de contremesure optimale.

Le modèle IRS proposé comprend quatre composants principaux :

1. Cout du dommage d'attaque : représente le montant des dommages causés à une cible d'attaque lorsque les contrôles existants de sécurité sont indisponibles, ou inefficaces.
2. L'effet positif de contremesure : représente l'effet positif de déploiement d'une contremesure sur la protection des ressources.

3. Impact négatif des contremesures : se réfère à l'impact d'une contremesure sur la disponibilité des services et l'accès des utilisateurs.

4. Le coût de déploiement de la contremesure : coût d'administration ou complexité de l'ajout d'une politique de sécurité dans les dispositifs de sécurité.

La figure 2.2 illustre l'architecture proposée du système de réponse d'intrusion automatisé, il y a 2 structures de données : arbres de défense d'attaque (ADT) et le graphe de dépendance de service (SDG). Dans un modèle de réseau étendu, différents ADT peuvent être définis en fonction du SDG de manière chaque ADT protège un service. Lorsqu'une alerte est déclenchée par IDS, l'infrastructure la mappe sur l'ADT sélectionné, à partir duquel tous les chemins d'attaque possibles vers la cible sont extraits. Ensuite, tous les points de défense sont identifiés pour chaque chemin d'attaque. Sur la base du type de point de défense, des contremesures appropriées sont sélectionnées. Ensuite, pour chacun d'eux trois fonctions sont évaluées de manière indépendante : *security benefit*, *security impact*, et *security cost*. Un ensemble optimal de Pareto est généré pour voir quelles contremesures satisfont le compromis entre ces trois fonctions. Enfin, la contremesure optimale est choisie parmi l'ensemble optimal de Pareto en ce qui concerne la gravité de l'attaque. Ce dernier est mesuré par le composant Cout du dommage d'attaque qui est basé sur le SDG.

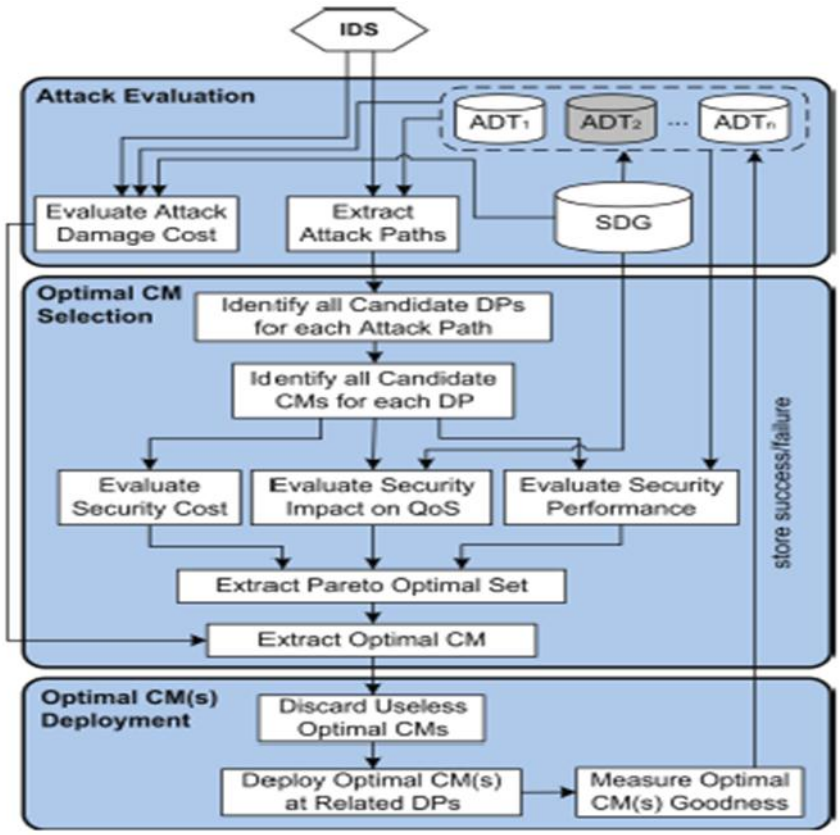


Figure 2.2 : Architecture du système IRS automatisé [43]

La métrique utilisée pour évaluer et sélectionner la meilleure contremesure comme montre l'équation (2.2)

$$\vartheta(cm_k) = \omega_P.S_P(cm_k) + \omega_Q.S_Q(cm_k) + \omega_C.S_C(cm_k) \qquad (2.2)$$

Ou $\vartheta(cm_k)$ se représente comme le score final, et S_P, S_Q, S_C ce sont des valeurs normalises qui s'exprime respectivement la performance de la sécurité, l'impact de sécurité sur la qualité du service, et le cout de sécurité. ω_P (*attack damage*), ω_Q, ω_C s'exprime respectivement les poids utilisés pour favoriser un objectif par rapport aux autres afin de sélectionner la contremesure de sécurité optimale parmi l'ensemble optimal de Pareto. Ces poids devraient être satisfaits cette condition :

$$\omega_P + \omega_Q + \omega_C = 1 \qquad (2.3)$$

$$\omega_P > 0, \omega_Q > 0, \omega_C > 0$$

2.2.3 La sélection des contremesures optimales en évaluant une métrique basant sur l'analyse du graphe d'attaque

2.2.3.1 Sélection individuelle des contremesures en basant sur le retour en investissement de la réponse (RORI)

Les auteurs de [15] ont proposé une métrique qui il s'agit d'un indice de dépendance de service pour les réponses sensibles aux coûts, basé sur une comparaison financière des alternatives des réponses. Leur solution considère l'approche proposée par [29] en améliorant la métrique proposée *RORI (return on response investement*), en prendre en compte le cout de la contremesure, et la mitigation du risque associé, mais aussi la valeur et la surface de l'infrastructure, et les pertes annuelles attendues en cas d'attaque ou une intrusion. Le *RORI* amélioré permet de fournir une contremesure qui est relative avec à la taille l'infrastructure. Le *RORI* se représente avec l'équation (2.4)

$$RORI = \frac{(ALE \times RM) - ARC}{ARC + AIV} * 100 \quad (2.4)$$

ALE : *Annual loss expectancy*, est l'espérance de perte annuelle, qui fait référence au coût d'impact obtenu dans le cas d'absence de mesures de sécurité. ALE est exprimé en devise par an ($/an) et dépendra directement à la gravité et la probabilité de l'attaque.

RM : *Risk Mitigation*, est le niveau d'atténuation des risques, qui est associé à une solution particulier ($0 \leq RM \leq 100$), $RM = 0\%$ en absence de contremesure.

ARC : *Annual Response Cost*, est le coût de réponse annuel, qui est engagé par la mise en œuvre d'une nouvelle action de sécurité, ARC est toujours supérieur ou égal à zéro($ARC \geq 0$), ARC est exprimé en devise par an ($/an).

AIV : *Annual Infrastructure Value*, est la valeur annuelle de l'infrastructure, comme le cout des équipements, les services pour les opérations régulières…etc. AIV est supérieur strictement à zéro($ARC > 0$), AIV est exprimé en devise par an ($/an).

ALE et AIV sont des paramètres fixes, par contre ARC et RM se sont des valeurs variables.

Cette approche peut être connue par plusieurs limites, on va commencer par la couverture de la surface de la contremesure qui est définie comme le pourcentage de la surface d'attaque, qu'elle recouvre. Cependant, dans la pratique, il n'est pas simple d'établir une correspondance entre les contremesures et les attaques. Dans ce modèle, il est possible d'appliquer plus d'une contremesure simultanément, dans ce cas, la surface d'une contremesure, peut être partiellement ou totalement recouverte par d'autres. Les auteurs ne considèrent pas l'impact négatif d'une contremesure sur les services.

2.2.3.2 Sélection des contremesures en basant sur le retour en investissement de la réponse par état (StRORI)

Les auteurs de [12] ont proposé une amélioration de la métrique *RORI (Return On Response Investment)* proposée dans [13,14]. Cette métrique est le retour avec état sur l'investissement de réponse (en anglais *Stateful Return On Response Investment StRORI)* qui considère à quel état le *RORI* est effectué. Ce formalisme permet de connecter les actions d'attaques en basant sur des pré- condition et post-condition. Le graphe d'attaque est généré automatiquement en utilisant la configuration du réseau,

les indices publique disponibles sur les vulnérabilités. Dans cette approche les auteurs ont concentré sur deux modes pour sélectionner et imposer les contremesures, le mode préventif et le mode réactif. Pour le premier mode la sélection des contremesures se base sur l'évaluation du niveau local et global du risque dans le système grâce à l'extraction du toute les actions d'attaque, afin d'appliquer un ensemble préventif des contremesures, en basant sur le *StRORI* pour choisir les contremesures optimales. Pour le mode réactif, un nouvel ensemble des contremesures sont sélectionnées et appliquer pour mitiger la propagation des attaques. La différence principale entre les deux modes se résume que certaines contremesures peuvent être choisies dans le mode préventif, mais s'appliquent et s'imposent seulement dans le mode réactif.

L'approche permet de classer et sélectionner la contremesure ou le groupe les plus appropriés contre une attaque donnée dans un état particulier du système. Le *StRORI* se représente comme une extension vers un nouveau retour dynamique d'investissement sur la réponse. Le *StRORI* se présente par l'équation (2.5) :

$$StRORI = \frac{(ALE \times RM) - ARC}{ARC + AIV} * 100 \qquad (2.5)$$

ALE : est calculé par le produit $ALE = Cost * ARO$ ou $Cost$ fait la référence à la gravité de l'attaque et son occurrence.

AIV : est calculé par $AIV = \sum_{i=0}^{n} AEC_i$,ou AEC se réfère au cout annuel des équipements de tous les points d'application de la politique, qui apparaît dans l'instantané (*snapshot*) du système.

ARC : *Annual response cost*, $ARC = C_i + C_m + C_d + Od_c + I_c$ ou C_i se refere ou cout d'implémentation de la contremesure, C_m est le cout de la maintenance, C_d est le coût de suppression de la contremesure, Od_c et I_c sont respectivement le cout direct et indirect.

RM : *Risk mitigation*, $RM = EF \times COV$, où EF=se réfère à l'efficacité de la contremesure, et COV se réfère à la couverture de la contremesure.

Le *StRORI* prend en compte l'état de la contremesure durant le calcul de ses paramètres ARC et RM comme représente le tableau 2.1 :

Etat de la contremesure	Le calcul du ARC	Le calcul du RM
Ajout	$ARC = C_i + C_m + Od_c + I_c$	$RM = EF \times COV$
Suppression	$ARC = C_d$	$RM = 0$
Garder	ARC=0	$RM = rien\ a\ changer$

Tableau 2.1 le calcul des variables ARC et RM en fonction de la contremesure [12].

Les approches au-dessus montrent un manque dans leur métrique, principalement ils ne prennent pas en compte l'impact négatif d'une contremesure sur la topologie, ou son impact sur une autre contremesure. En plus les métriques RORI et StRORI ne considèrent pas les dépendances entre les contremesures, pour bien réduire les couts.

2.2.4 Synthèse

Après l'étude et l'analyse des travaux connexes nous avons remarqué qu'ils se souffrent par plusieurs manques, nous avons souligné ces faiblesses au-dessus, et nous avons concentré sur eux pour établir notre proposition qui représente la sélection de la contremesure optimale en basant sur l'analyse du graphe d'attaque et l'évaluation d'une métrique. Le tableau 2.2 présente la synthèse du ces travaux :

Approche	IRS	Estimation des risques	Métrique de sélection
[7]	cout sensible	Graphe d'attaque	*ROI* (équation 2.1)
[43]	cout sensible	Graphe de dépendance de service	$\vartheta(cm_k)$ (équation 2.2)
[15]	cout sensible	Graphe d'attaque	*RORI* (équation 2.4)
[12]	cout sensible	Graphe d'attaque	*StRORI* (équation 2.5)

Tableau 2.2 la synthèse des travaux connexes

2.4 Conclusion

Dans ce chapitre nous avons présenté quelques travaux connexes, qui nous aident à comprendre comment évalué le risque en basant sur le graphe d'attaque qui se considère comme une solution efficace en permettant la modélisation des menaces

dans le Cloud dans le mode hors ligne, ainsi la sélection de la contremesure optimale en utilisant une métrique.

Dans le chapitre prochain, on va exposer notre architecture proposée pour sélectionner la contremesure optimale.

Chapitre 3 Contribution

3.1 Introduction

Les IDS ne sont pas capables de présenter les relations entre les vulnérabilités, et leur impact dans le Cloud computing, donc le graphe d'attaque représente la meilleure technique pour analyser et modéliser l'exploitation de ses vulnérabilités, et qui nous permet d'offrir une protection complète des ressources infectés par une estimation précise des risques, qui peut nous aide à choisir une solution optimale des contremesures, afin de reprendre la santé du système Cloud.

La conception d'une stratégie de défense efficace, c'est un défi en ce qui concerne la complexité du réseau Cloud, les attaques qui sont très répandues et sophistiquées, les compétences des attaquants, et enfin la diversité des dispositifs de la sécurité. Le problème posé et comment sélectionner les contremesures optimales automatiquement en prenant en compte plusieurs critères par exemple leur effet sur le système du Cloud computing, concernant sa performance et la qualité des services fournit (SLA), qui mène un système de réponse aux intrusions (IRS) très précis et puissant.

3.2 Contribution

Après les études et la discussion des travaux connexes dans le chapitre précèdent, notre travail consiste à définir la démarche permettant la sélection de la meilleure contremesure en prend en compte plusieurs critères (ex : la dépendance entre les composants, le niveau de risque, la dépendance entre les contremesures, les impacts négatifs des contremesures, etc.). Ces critères sont prises en considération dans la métrique proposée CRORI qui évalue les contremesures afin de trouver la mesure optimale. Le processus de sélection se passe par plusieurs étapes, qui se résument dans l'architecture proposée dans la figure 3.1.

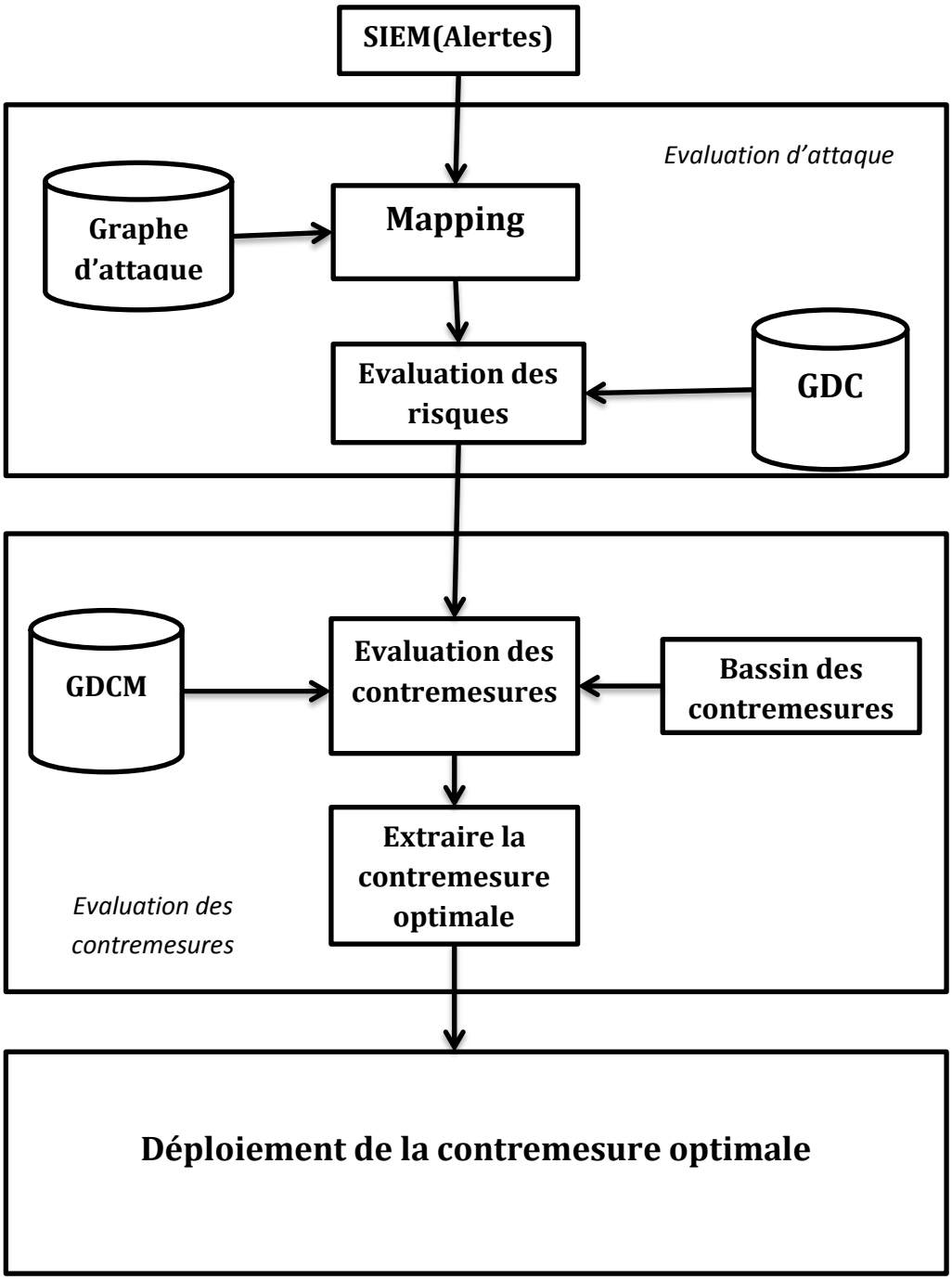


Figure 3.1 : l'architecture de sélectionner la contremesure optimale en Cloud

La figure 3.1 représente notre architecture proposée qui est divisée en trois parties : l'évaluation d'attaque, l'évaluation de la contremesure et le déploiement de la contremesure optimale. On va commencer par l'explication de chaque section.

3.2.1 L'évaluation d'attaque

3.2.1.1 SIEM :

Security Information Management System, ou Système de Gestion de la Sécurité du Système d'Information. Un SIEM permet de centraliser les informations relatives à la sécurité du SI, notamment en collectant les logs de déférentes sources (services, système, IDS, etc...), puis en offrant la possibilité de les corréler entre eux afin de lever des alertes de sécurité plus ciblées [36].

3.2.1.2 Graphe d'attaque :

C'est un graphe $G = (S, L)$, ou S contient l'ensemble des nœuds du graphe, $S = E \cup C$, ou E représente l'ensemble des exploits définit par le tuple (V, H_s, H_d, P_r, P_c), où V représente CVE de la vulnérabilité exploitée par l'attaquant, H_s se considère comme la position de l'attaquant , H_d se considère comme l'hôte cible qui contient la vulnérabilité à exploiter, P_r représente la probabilité locale du nœud qui correspond à la complexité d'accès, P_c représente la probabilité conditionnelle (la probabilité du nœud sachant ses parents). C représente l'ensemble des conditions, $C = (PR, H_d, P_c)$ où PR représente le privilège, L représente l'ensemble des liens entre les nœuds (pré-condition, post-condition) ou $L = L_{pre} \cup L_{post}$ où $L_{pre} \subseteq C * E$ et $L_{post} \subseteq E * C$.

a. Calcule de la probabilité conditionnelle

On peut associer une probabilité conditionnelle avec chaque nœud n dans G. Cette probabilité représente la probabilité d'exploiter la vulnérabilité dans n avec la prise en compte des probabilités des pré-conditions (les parents de n).

Si $n \in E$ alors est donnée par les équations 3.1 et 3.2 suivantes :

$$PcD(n|p_n) = P_r(n) * (1 - \prod_{c \in p_n}(1 - PcD(c|p_c)) \qquad (3.1)$$

Sinon si $n \in C$ alors :

$$PcD(n|p_n) = 1 - \prod_{c \in p_n}(1 - PcD(c|p_c)) \qquad (3.2)$$

Où $p_n$est l'ensemble des parents de n, $P_r(n)$ est la probabilité locale de n. En d'autre terme $PcD(n|p_n)$ est la probabilité globale de n dans G.

3.2.1.3 Alerte :

structure de données, qui représente par (src, dis, CVE, T), où *src* représente adresse IP de l'hôte source, *dis* représente adresse IP de l'hôte de destination, et *CVE* représente l'indice de la vulnérabilité à exploiter, *T* représente le privilège gagné.

3.2.1.4 Mapping :

Cette opération nous permet la cartographie des alertes générées par les SIEMs sur le graphe d'attaque. Le modèle que nous avons suivre se mappe sur le graphe d'attaque par adresse IP de l'hôte de destination comme représente l'algorithme suivant.

Algorithme Mapping

Entrées : $Alerte\ A = (src, dis, CVE, T), Graphe\ GA = (S, L)$

Sortie : $map \subseteq l_{post}$

1. ***Début***
2. Soit $map = 0$;
3. ***Pour tout*** $(e_i, c_i) \in L_{post}$:
4. Si $(A.src = e_i.H_s) \wedge (A.dis = c_i.H_d) \wedge (A.CVE = e_i.V) \wedge (A.T = c_i.PR)$ $alors\ map = map \cup \{(e_i, c_i)\}$;
5. ***Fin pour.***
6. ***Fin***

3.2.1.5 Graphe de dépendance des composants (GDC)

Nous avons inspiré de l'idée de [38] pour proposer notre graphe *GDC* qui permet la présentation des valeurs des composants dans le réseau du Cloud computing et la relation entre eux. Telle que $GDC = (A, Dp)$, où A est l'ensemble des nœuds. Chaque nœud est représenté par le tuple $< NC, C, I, D, VC >$. NC c'est le nom du composant. C, I, D représentent la confidentialité, l'intégrité et la disponibilité respectivement de chaque composant par rapport à l'infrastructure. Le VC est la valeur du composant, on peut le calculer par l'équation (3.3):

$$VC = (C + I + D)/3 \quad (3.3)$$

Où C, I, D sont des entiers dans [0,5], et Dp représente la relation de la dépendance entre les nœuds du graphe. L'ensemble $Dp = Dp_{inc} \cup Dp_{con} \cup Dp_{lec} \cup Dp_{ecr}$ est l'ensemble d'arcs. $Dp_{inc}, Dp_{con}, Dp_{lec}$ et Dp_{ecr} représentent respectivement une dépendance physique-inclusion, une dépendance physique-connexion, une dépendance logique-lecture/exécution, et une dépendance logique-écriture/modification/suppression. Le tableau 3.1 explique en détail ces dépendances :

La Dépendance	Explication
Physique ($Dp_{inc} \cup Dp_{con}$)	***Physique-Inclusion (Dp_{inc}):*** a_i inclut dans a_j ($a_i < a_j$) se signifie que a_i est contenu dans a_j, où a_i est une partie de a_j. Alors $(a_i, a_j) \in Dp_{inc}$ ***Physique-Connexion (Dp_{con}):*** a_i est connecté à a_j. Alors $(a_i, a_j) \in Dp_{con}$
Logique ($Dp_{lec} \cup Dp_{ecr}$)	Une dépendance logique entre deux composants implique la dépendance d'accès entre eux : ***lecture/exécution (Dp_{lec}) :*** a_j un composant qui permet à a_i de lire ou d'exécuter une tâche. Alors $(a_i, a_j) \in D_{lec}$ ***écrire/modifier/supprimer (D_{ecr}) :*** a_j un composant qui permet a a_i d'écrire, modifier ou supprimer une telle tâche. Alors $(a_i, a_j) \in D_{ecr}$

Tableau 3.1 les dépendances entre les composants dans un réseau

Dans le Cloud computing chaque composant possède sa propre valeur. Quand en prise en compte ces dépendances, les valeurs des composants va être modifié. En cas d'attaque et précisément en cas d'exploitation d'une vulnérabilité d'un composant, il peut être infecté d'autre composant, par exemple si un attaquant exploite une vulnérabilité dans un serveur http, il peut aussi accéder à la base de données, donc la valeur de la base de données est influencée par la valeur du serveur http. Nous avons proposé l'algorithme suivant qui nous permettent de la mise à jour les valeurs des composants en prise en compte les dépendances entre les composants :

Algorithme Mis à jour GDC

Entrées : Graphe $GDC = (A, Dp)$

Sortie : Graphe $GDC = (A, Dp)$ où chaque VC_i pour chaque a_i est modifié

7. ***Début***
8. ***Soit*** $finale = \{a_i | \forall (a_j, a_k) \in D/D_{inc}, a_j \neq a_i \wedge \forall (a_j, a_k) \in D_{inc}, a_k \neq a_i\}$;
9. ***Soit*** $T = A$;
10. ***Tant que*** $T \neq \emptyset$ ***faire*** :
11. ***Pour chaque*** $a_i \in \{a_j | a_j \in T \wedge \forall (a_j, a_k) \in Dp,\ a_k \in finale\}$ ***faire*** :
12. $C(a_i) = Max\{C(a_i)\} \cup \{C(a_j) | (a_i, a_j) \in Dp_{lec}\}$;
13. $I(a_i) = Max\{I(a_i)\} \cup \{I(a_j) | (a_i, a_j) \in Dp_{ecr}\}$;
14. $D(a_i) = Max\{D(a_i)\} \cup \{D(a_j) | (a_i, a_j) \in Dp_{ecr}\}$;
15. $VC(a_i) = (C(a_i) + I(a_i) + D(a_i))/3$;
16. $VC_{inc}(a_i) = Max\{VC(a_i)\} \cup \{VC(a_j) | (a_i, a_j) \in Dp_{inc}\}$
17. ***Soit*** $VC_{con}(a_i) = 0$;
18. ***Pour chaque*** $a_j \in \{a_j | (a_j, a_i) \in Dp_{con}\}$ ***faire***
19. ***si*** $VC(a_i) \geq VC(a_j)\ VC_{con}(a_i)_{a_j} = VC(a_j)$
20. ***sinon*** $VC_{con}(a_i)_{a_j} = ceil(VC(a_i) + 1/2(VC(a_i) - VC(a_j)))$;
21. $VC_{con}(a_i) = Max(VC_{con}(a_i)_{a_j}, VC_{con}(a_i))$;
22. ***Fin pour*** ;
23. $T = T/\{a_i\}$;
24. $final = final \cup \{a_i\}$;
25. ***Fin pour*** ;
26. ***Fin tant que*** ;
27. ***Fin***

6. L'évaluation du risque

Chaque nœud dans le graphe d'attaque doit posséder un risque qui permet d'identifier le risque local et global par apport l'environnement du Cloud. Dans cette section on va définir comment calculé le risque local de chaque nœud, ainsi le risque global en basant sur des donnés extraite depuis le graphe d'attaque et le GDC.

a) Calcule les valeurs de risque associées aux éléments dans le graphe d'attaque

La valeur $PcD(n|p_n)$ ne mesure pas exactement le risque de n puisque elle considère seulement la vulnérabilité et non pas le composant a qui comporte cette dernière. Alors il faut combiner $PcD(n|p_n)$ et l'impact du composant cible a dans n (la valeur VC_a associée à a) et la valeur d'impact de la vulnérabilité v utilisée dans n (VI_v).

Si $n \in E$ alors le niveau de risque est calculé par l'équation (3.4):

$$Rs(n) = VC_a * VI_v * PcD(n|p_n) \quad (3.4)$$

Où :

- VC_a est la valeur de a calculé par l'algorithme Mis à jour GDC.
- $VI_v = 10{,}41 * (1 - (1 - C_v) * (1 - I_v) * (1 - D_v))$ (3.5)

  Où C_v, I_v et A_v est respectivement la confidentialité, l'intégrité et la disponibilité de la vulnérabilité $v \in n$ selon CVSS [8] indices.

- $PcD(n|p_n)$ représente la probabilité conditionnelle associée au nœud du graphe d'attaque. Elle est calculée par l'équation (3.1, 3.2).

Sinon si $n \in C$ alors le niveau de risque sera calculé par l'équation (3.6):

$Rs(n) = VC_a * PcD(n|p_n)$ (3.6)

b) Calcule de risque associé aux chemins dans le graphe d'attaque:

Le risque du chemin est calculé par l'algorithme suivant, donc comme résultat on a plusieurs chemins d'attaque, on va déterminer le chemin le plus probable ça veut dire le chemin le plus dangereux ChD_i :

Algorithme Chemin dangereux

Entrées : $Chemins$=Toutes les chemins qui traversent les nœuds dans map

Sortie : ChD l'ensemble des chemins les plus dangereux.

1. ***Début***
2. ***Soit*** $ChD = \emptyset$, $Rs = \emptyset$;
3. ***Pour tout*** $ch_i \in chemins$ ***faire :***
4. Rs_{ch_i}=*1 ;*
5. ***Pour tout*** $n_i \in ch_i$ ***faire :***
6. $Rs_{ch_i} = Rs_{ch_i} * Rs(n_i)$;
7. ***fin pour ;***
8. Ajouter Rs_{ch_i} à Rs.
9. ***Fin pour ;***
10. ***Pour tout*** $Rs_{ch_i} \in Rs$ ***faire :***
11. ***Si*** $Rs_{ch_i} = \max(Rs)$ ***alors***
12. *ajouter* ch_i *à* ChD.

3.2.2 L'évaluation de la contremesure

Dans cette partie on va évaluer l'ensemble des contremesures (Bassin des contremesures) approprié aux vulnérabilités dans le chemin le plus dangereux afin de réduire la propagation d'attaque. On va utiliser la métrique proposée $CRORI$ pour choisir la contremesure optimale selon plusieurs critères. Avant de présenter cette dernière métrique, on va expliquer le graphe de dépendance entre les contremesures. On va expliquer en détail comment cette partie se consiste.

3.2.2.1 Graphe de dépendance des contremesures (GDCM)

Nous avons proposé un graphe de dépendance entre les contremesures *GDCM*, c'est un graphe $G' = (CM, R)$, où CM représente l'ensemble des nœuds cm_i de graphe, où chaque nœud se représente par le tuple <$Ncm_i, Vcm_i, INcm_i$>, où Ncm_i représente le

nom de la contremesure, Vcm_i représentent les CVE des vulnérabilités qui sont couvris par Ncm_i , $INcm_i$ représente l' impact négatif quand en appliquant cm_i. R L'ensemble des liens entre les contremesures, qui représente le nombre des vulnérabilités partagées entre deux contremesures.

3.2.2.2 L'évaluation des contremesures avec CRORI

CRORI est une métrique qui permet le retour en investissement de la réponse en Cloud (en anglais *Cloud Return On Response Ivestment*), c'est une métrique améliorée de la métrique *StRORI* [12] et adaptée avec le Cloud computing. Elle est présentée dans le chapitre 2. *StRORI* [12] ne prise pas en compte ni l'impact négatif de la contremesure après l'implémentation ni sa dépendance d'autre contremesure (la couverture).

L'évaluation des contremesures sur le chemin le plus dangereux permet de sélectionner la contremesure optimale. La métrique proposée utilise des paramètres spécifiques à ce chemin et d'autres associés à la contremesure.

a- Les attributs associés à chaque contremesure

Le tableau 3.2 définit les trois attributs essentiels associés à chaque contremesure : Diminution de risque (DR), Cout de réponse (CR) et l'impact négatif (IN).

b- Les attributs associés au chemin le plus dangereux

Le tableau 3.2 montre aussi comment calculer les attributs associés au chemin le plus dangereux : Perte Estimée (PE) et la Valeur d'Infrastructure (VI).

c- Ajuster les valeurs des attributs associés aux contremesures

Notre métrique prend en considération la dépendance entre les contremesures, c.-à-d. si une contremesure était déjà installée, et la nouvelle possède une dépendance avec elle, qui représente le nombre des vulnérabilités partagées entre eux, dans ce cas les paramètres de la métrique de la nouvelle (optimale) sont ajustés. On va montrer comment ajuster ces valeurs des attributs depuis le graphe $GDCM$ présenté précédemment. Lors d'évaluation d'une contremesure, si une ou plusieurs sont installées et si ces dernières ont des relations de dépendances avec la première alors les valeurs des attributs DR et IN sont ajustées :

$$DR_{ChD_i,cm_i} = DR_{ChD_i,cm_i} - (DR_{ChD_i,cm_i} * nbvc_{cm_i,dep}/nbv_{cm_i}) \quad (3.7)$$

Où $nbvc$ représente le nombre des vulnérabilités partagée entre les contremesures et l'ensemble dep est définit comme l'ensemble des contremesures déjà installée et dépendent de cm_i , nbv_{cm_i} le nombre des vulnérabilités couvris par cm_i.

$$IN_{ChD_i,cm_i} = IN_{ChD_i,cm_i} - (IN_{cm_i} * nbvc_{cm_i,dep}/nbv_{cm_i}) \quad (3.8)$$

Paramètres	Description	Calcul
PE	Perte Estimée dans le chemin le plus dangereux	$PE = \sum_i GR_i$, on a inspiré de l'approche proposée par [2], ou GR_i c'est une valeur monétaire (par DZ) qui définit comme la gravité dans le cas d'exploitation de la vulnérabilité dans le nœud i. Ou GR_i est mesurée selon six niveaux de «insignifiant » allant de « grave ».
DR	Diminution du Risque dans le chemin le plus dangereux	$DR = \frac{nb_v}{nb_{vp}}$ ou nb_v représente le nombre des vulnérabilités qui sont couvris par la contremesure, et nb_{vp} le nombre total des vulnérabilités exploité dans le chemin d'attaque le plus probable.
CR	Coût de réponse	$CR = C_i + C_m + C_d + Od_c + I_c$ [12] où C_i est le coût de l'implémentation de la contremesure, C_m est coût de maintenance, C_m est le coût de la suppression d'une contremesure, Od_c est autre coût directe, I_c est les coûts indirecte.
IN	Impact Négatif sur les composants dans le chemin le plus dangereux	Représente une valeur monétaire, qui mesure l'effet de l'implémentation d'une contremesure sur les utilisateurs du Cloud par rapport au contrat SLA. Où IN est définit sur sept niveaux: «négligeable, très faible, faible, moyen, haut, très haut, et extrême », où chaque niveau correspond une valeur monétaire qui est précisé par le fournisseur du Cloud.
VI	Cout de l'infrastructure dans le chemin le plus dangereux	$VI = \sum_{ai \in chD}^{an} CE_{ai}$ où CE_{ai} est définit comme le coût d'équipement de toutes les points d'application de la politique de sécurité dans le composant a_i par rapport au chemin chD.

Tableau 3.2 les paramètres de CRORI

Donc le *CRORI* est calculé par l'équation (3.9) :

$$CRORI(ChD_i, cm_i) = \frac{\left(PE_{ChD_i} \times DR_{ChD_i,cm_i}\right) - \left(CR + IN_{ChD_i,cm_i}\right)}{VI_{ChD_i} + CR + IN_{ChD_i,cm_i}} \quad (3.9)$$

L'évaluation précédente est appliquée sur l'ensemble de contremesures appropriées avec les vulnérabilités du chemin $\boldsymbol{ChD_i}$. La contremesure optimale est la contremesure avec la valeur $\boldsymbol{CRORI}$ maximale.

3.2.2.3 Déploiement de la contremesure optimale

Dans les sections précédentes, on a sélectionné la meilleur contremesure, qui est appropriée avec les vulnérabilités du chemin le plus dangereux, donc le déploiement de la contremesure consiste à l'installé sur la machine cible qui peut être une machine virtuelle, un par feu, un hyperviseur, etc.

3.3 Exemple illustratif

Pour illustrer notre proposition, nous avons adopté l'exemple des auteurs dans [28] qui ont travaillé sur la modélisation des menaces dans un centre de donnée (*Datacenter*) du Cloud. Nous avons adopté cet exemple comme un cas d'étude pour expliquer notre architecture qui se base sur une méthode claire et efficace pour évaluer le risque en Cloud et choisir les solutions optimales adéquates. La figure 3.2 montre l'infrastructure de Cloud pour le cas d'étude.

3.3.1 Établir le graphe d'attaque

Pour construire un scénario d'attaque réel on a adopté l'exemple du graphe d'attaque utilisé par [28] pour présenter un scénario d'attaque qui nous aide à exploiter des vulnérabilités réelles, qui sont liées aux composants hardwares et softwares de l'infrastructure. Ces vulnérabilités sont indiquées dans la base de données nationale des vulnérabilités (NVD) [33]. La probabilité P_r est récupéré à partir la NVD [33] qui représente le CVSS [8].

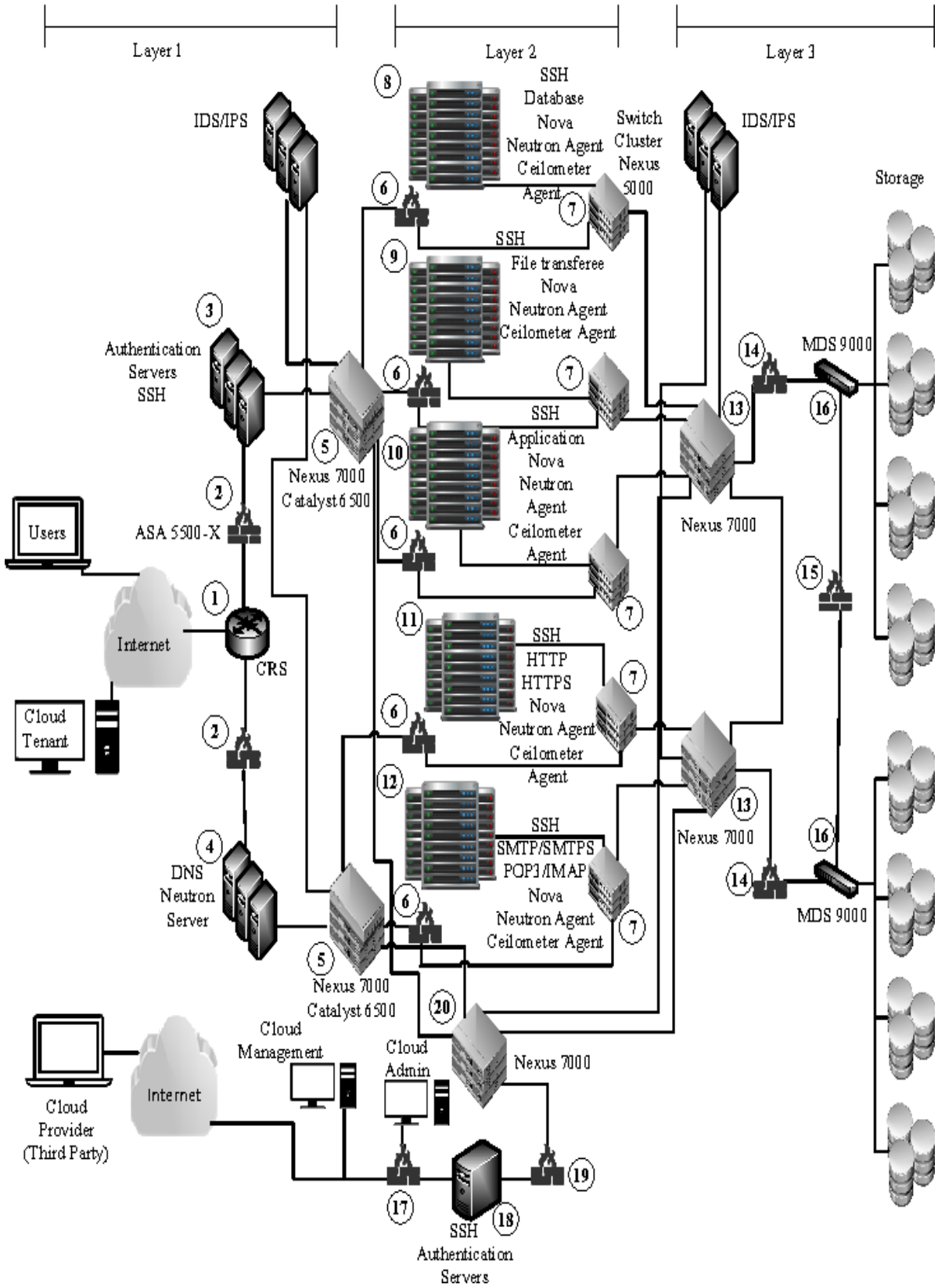


Figure 3.2: l'infrastructure du Datacenter en Cloud [28]

Il est supposé que l'attaquant est accédé aux services du Cloud Tenant (locataire du Cloud) comme utilisateur légitime(UL) le hôte 0, afin de voler les données stockées, l'utilisateur doit être accédé au VM du http ainsi VM du serveur d'application et VM du serveur de la base de données pour atteindre son but, en supposant que les services utilisés dans le Datacenter sont les suivant :

- Tectia Server version 5.2.3, pour ssh exécuté dans toutes les VMs.
- Serveur http Apache exécuté dans VM http.
- Oracle version 10.1.0.2, était installé sur VM d'application.
- Oracle version 10.2.1, était installé sur VM de la base de données.
- Xen version 4.3.0, est exécuté comme un hyperviseur pour contrôler les VMs sur les machines physiques.

Le tableau 3.3 montre les vulnérabilités exploitées qui sont liées aux services précédents :

Service	CVE de la vulnérabilité
V_{http1}	2007-5156
V_{http2}	2007-1741
V_{ssh}	2007-5616
V_{app1}	2006-0586
V_{app2}	2004-1774
V_{bdd1}	2005-0297
V_{bdd2}	2007-1442
V_{Xen}	2013-4344

Tableau 3.3 Vulnérabilités et expositions communes associées aux services du centre de données [28]

Le scénario d'attaque se déroule en quatre étapes :

- ***Etape 1 :*** Pour que l'attaquant gagner l'accès comme un utilisateur légitime, il a employé la vulnérabilité (CVE-2007-5156) du VM du serveur http (hôte 11), qui lui permet d'injecter et exécuter le code arbitraire qui contient .php. dans l'extension du fichier. Dans la même VM, la vulnérabilité (CVE-2007-1741) est utilisée pour gagner les privilèges de la racine en renommant le répertoire ou en effectuant un lien symbolique. La vulnérabilité (CVE-2007-5156) est liée au VM du serveur ssh (hôte 11) qui peut aussi être utilisée pour gagner les privilèges de la racine dans la même VM.

- ***Etape 2 :*** Maintenant l'attaquant peut connecter au serveur d'application (hôte 10), en utilisant la vulnérabilité (CVE-2006-0586) qui est associée avec eux. Dans ce cas, l'attaquant est autorisé de gagner les privilèges des utilisateurs légitimes, par l'exécution des commandes arbitraires SQL à travers plusieurs paramètres. Pour gagner les privilèges de la racine dans cette VM, il peut employer l'un des deux possibilités, soit il applique la vulnérabilité (CVE-2004-1774) ou bien la (CVE-2007-5616) qui est associée au serveur ssh (hôte 10), à ce stade l'attaquant peut démarrer une connexion au VM du serveur de la base de données.

- ***Etape 3 :***l'attaquant utilise une vulnérabilité (CVE-2005-0297) associée au VM du serveur de la base de données (hôte 8) afin de gagner un accès comme un utilisateur légitime, puis dans cette VM, il peut gagner l'accès à la racine par l'application de la vulnérabilité (CVE-2007-1442).

- ***Etape 4 :*** Dans cette étape l'attaquant peut obtenir des données en relation avec cette VM (hôte 8), il peut obtenir encore plus d'information reliée au VM qui sont exécuté dans la même machine physique en gagnant l'accès aux hyperviseurs par l'exploitation de la vulnérabilité (CVE-2013-4344).

La figure 3.3 montre le graphe d'attaque qui modélise le scénario précèdent en calculant la probabilité conditionnelle associe à chaque nœud du graphe en utilisant les équations (3.1 et 3.2).

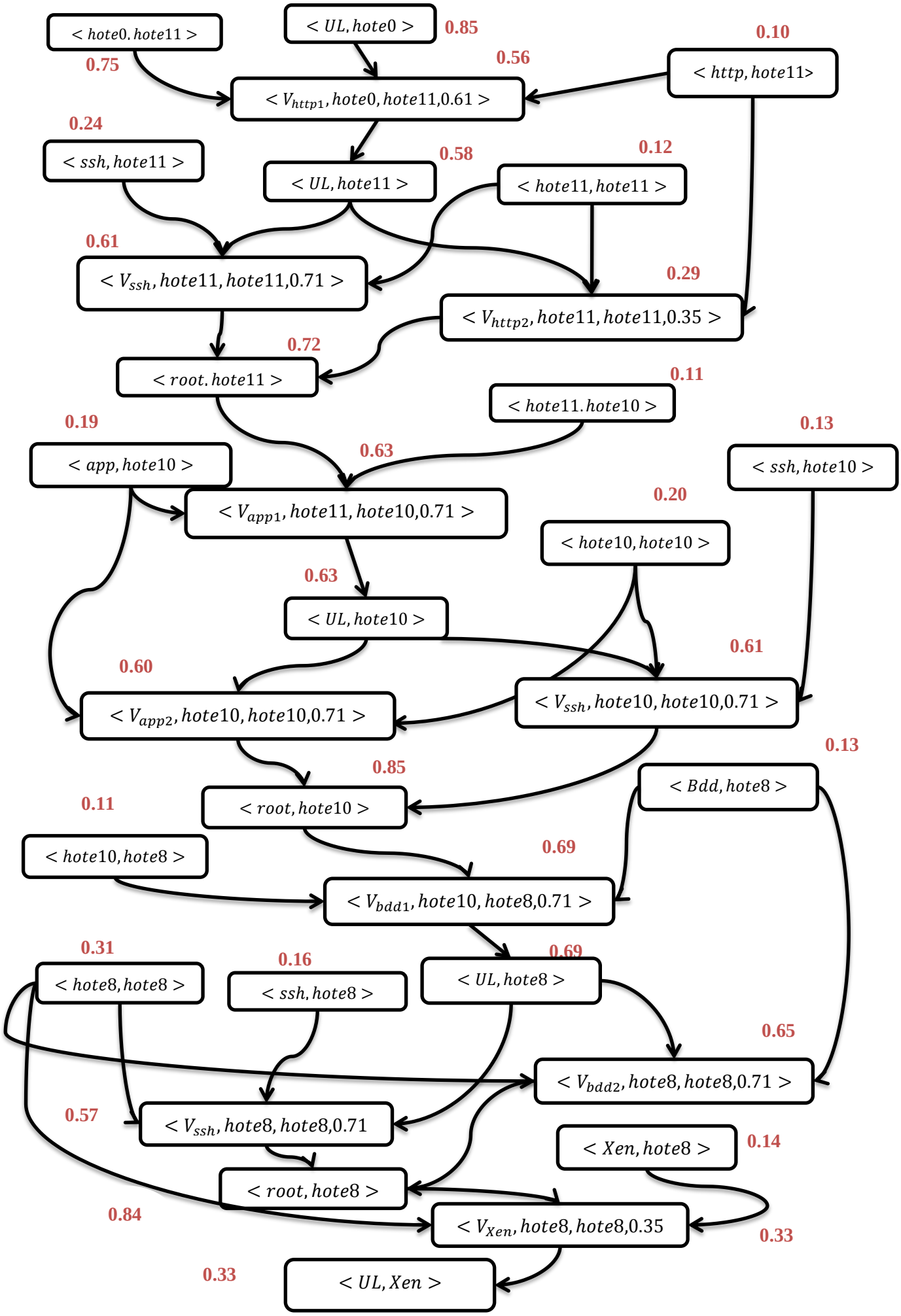


Figure 3.3 : Graphe d'attaque avec la probabilité conditionnelle associée

3.3.2 Établir le graphe de dépendance des composants

a. La création du graphe GDC

Dans cette partie on va construire le graphe de dépendance des composants à l'aide de l'infrastructure présentée précédemment. La figure 3.4 présente une partie du graphe dépendance où la ligne continue représente la dépendance Dp_{inc}, par contre la ligne discontinue représente la dépendance Dp_{con} :

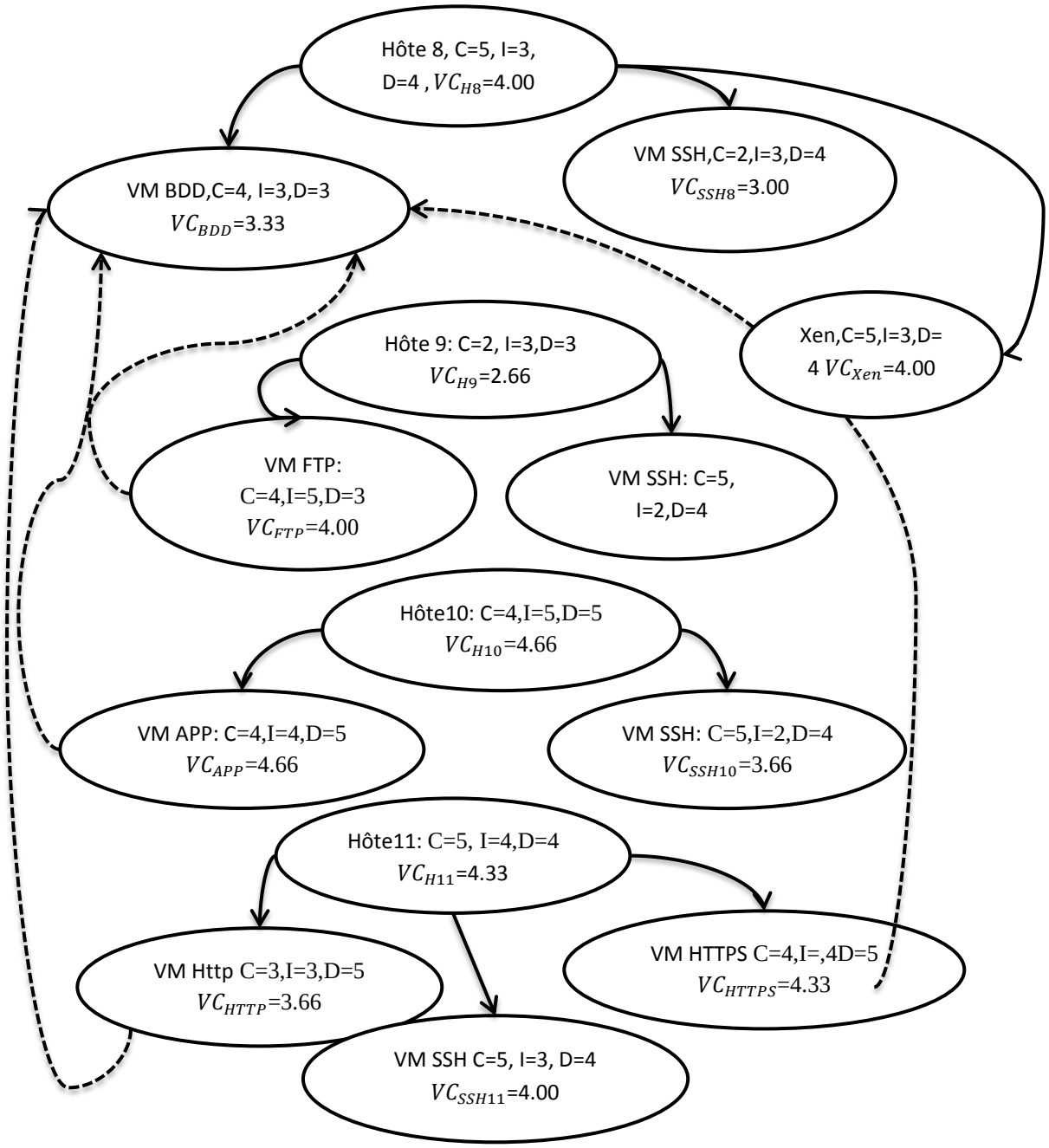


Figure 3.4 : représente le graphe de dépendance des composants

b. La mise à jour des valeurs des composants du GDC

Dans cette partie on va ajuster les valeurs des composants à l'aide de l'algorithme, le calcul se fait par étape.

- ***Etape 1 :*** On va calculer les valeurs VC des composants qui possèdent une dépendance physique-inclusion comme $\{Hote8, Hote9, Hote10, Hote11\}$ avec les composants du graphe GDC dont leurs valeurs ne seront pas changées comme :

$$\begin{Bmatrix} VM\ SSH_{H8}, Xen, VM\ FTP, VM\ \ SSH_{H9}, VM\ APP, VM\ SSH_{H10}, VM\ HTTP, \\ VM\ HTTPS, , VM\ SSH_{H11} \end{Bmatrix}$$

Où : $VC_{H9} = \max(VC_{H9}, VC_{SSH9}, VC_{FTP}) = \max(2.66,4.00,3.66) = 4.00$

$VC_{H10} = \max(VC_{H10}, VC_{SSH10}, VC_{APP}) = \max(4.66,3.66,4.66) = 4{,}66$

$VC_{H11} = \max(VC_{H11}, VC_{HTTP}, VC_{HTTPS}, VC_{SSH11}) = \max(4.33,3.66,4.33,4.00) = 4{,}33$

- ***Etape 2 :*** Dans cette étape on va calculer les valeurs VC des composants qui possèdent une dépendance physique-connexion comme $\{VM\ BDD\}$ avec les composants dont les valeurs ne seront pas changées comme :$\{VM\ FTP, VM\ APP, VM\ HTTP, VM\ HTTPS\}$.

On a $VC_{BDD} \geq VC_{HTTP}$ alors $VC_{con}(BDD)_{HTTP} = VC_{BDD} = 3.33$

On a $VC_{BDD} < VC_{APP}$ alors $VC_{con}(BDD)_{APP} = VC(BDD) + 1/2(VC(APP) - VC(BDD) = 3.33 + 0{,}67 = 4.00$

On a $VC_{BDD} < VC_{HTTP}$ alors : $VC_{con}(BDD)_{HTTP} = VC(BDD) + 1/2(VC(HTTP) - VC(BDD) = 3.33 + 0.17 = 3.50.$

Le même calcul est appliqué entre $VM\ BDD$ et $VM\ FTP$.

Enfin :

$VC_{BDD} =$ $\max(VC_{con}(BDD)_{HTTPS}, VC_{con}(BDD)_{HTTP}, VC_{con}(BDD)_{APP}, VC_{con}(BDD)_{FTP}) = 4.00.$

- ***Etape 3 :*** on va calculer la valeur de $H8$ puisque toutes ces relations sont avec des composants dont les valeurs ne seront pas changées $\{VM\ BDD, VM\ SSH8\}$

$VC_{H8} = \max(VC_{H8}, VC_{BDD}, VC_{SSH8}) = \max(4.00,4.00,3.00) = 4.00.$

3.3.3 Evaluation de risque

a. Les valeurs des composants associés au graphe d'attaque

Le tableau 3.4 suivant montre les valeurs des composants après leur mise à jour par l'algorithme $Mis\ a\ jour\ GDC$.

Le composant	**VC** met à jour	Normalisation (division sur 5)
Hôte 8	VC_{H8} =4.00	0.80
VM BDD	VC_{BDD} =4.00	0.80
VM $\mathbf{SSH_{H8}}$	VC_{SSH8} =3.00	0.60
Xen	VC_{Xen} = 4.00	0.80
Hôte 9	VC_{H9} =4.00	0.80
VM FTP	VC_{FTP} =4.00	0.80
VM $\mathbf{SSH_{H9}}$	VC_{SSH9} =3.66	0.73
Hôte 10	VC_{H10} =4.66	0.93
VM APP	VC_{APP} =4.66	0.93
VM $\mathbf{SSH_{H10}}$	VC_{SSH10} =3.66	0.73
Hôte 11	VC_{H11} =4.33	0.86
VM http	VC_{HTTP} =3.66	0.73
VM HTTPS	VC_{HTTPs} =4.33	0.86
VM $\mathbf{SSH_{H11}}$	VC_{SSH11} =4.00	0.80

Tableau 3.4 les valeurs finales des composants et ses normalisations

b. Le calcul de la valeur d'impact de chaque vulnérabilité associé au graphe d'attaque

A l'aide de la base nationale des vulnérabilités NVD [33], on a associé à chaque vulnérabilité les trois valeurs C, I et D, qui nous permet de calculer sa valeur d'impact VI par l'équation (3.5).

Service	CVE de la vulnérabilité	C	I	A	VI	Normalisation (division sur 10)
V_{http1}	2007-5156	0.275	0.275	0.275	6.44	0.64
V_{http2}	2007-1741	0.66	0.66	0.66	10.00	1.00
V_{ssh}	2007-5616	0.66	0.66	0.66	10.00	1.00
V_{app1}	2006-0586	0.275	0.275	0.275	6.44	0.64
V_{app2}	2004-1774	0.66	0.66	0.66	10.00	1.00
V_{bdd1}	2005-0297	0.275	0.275	0.275	6.44	0.64
V_{bdd2}	2007-1442	0.66	0.66	0.66	10.00	1.00
V_{Xen}	2013-4344	0.66	0.66	0.66	10.00	1.00

Tableau 3.5 les valeurs d'impact des vulnérabilités

c. Le calcul de niveau de risque pour chaque nœud appartient le graphe d'attaque

On va calculer le niveau de risque de chaque nœud par l'application de l'équation (3.6), où les valeurs sont présentées sur le graphe d'attaque par la figure 3.5.

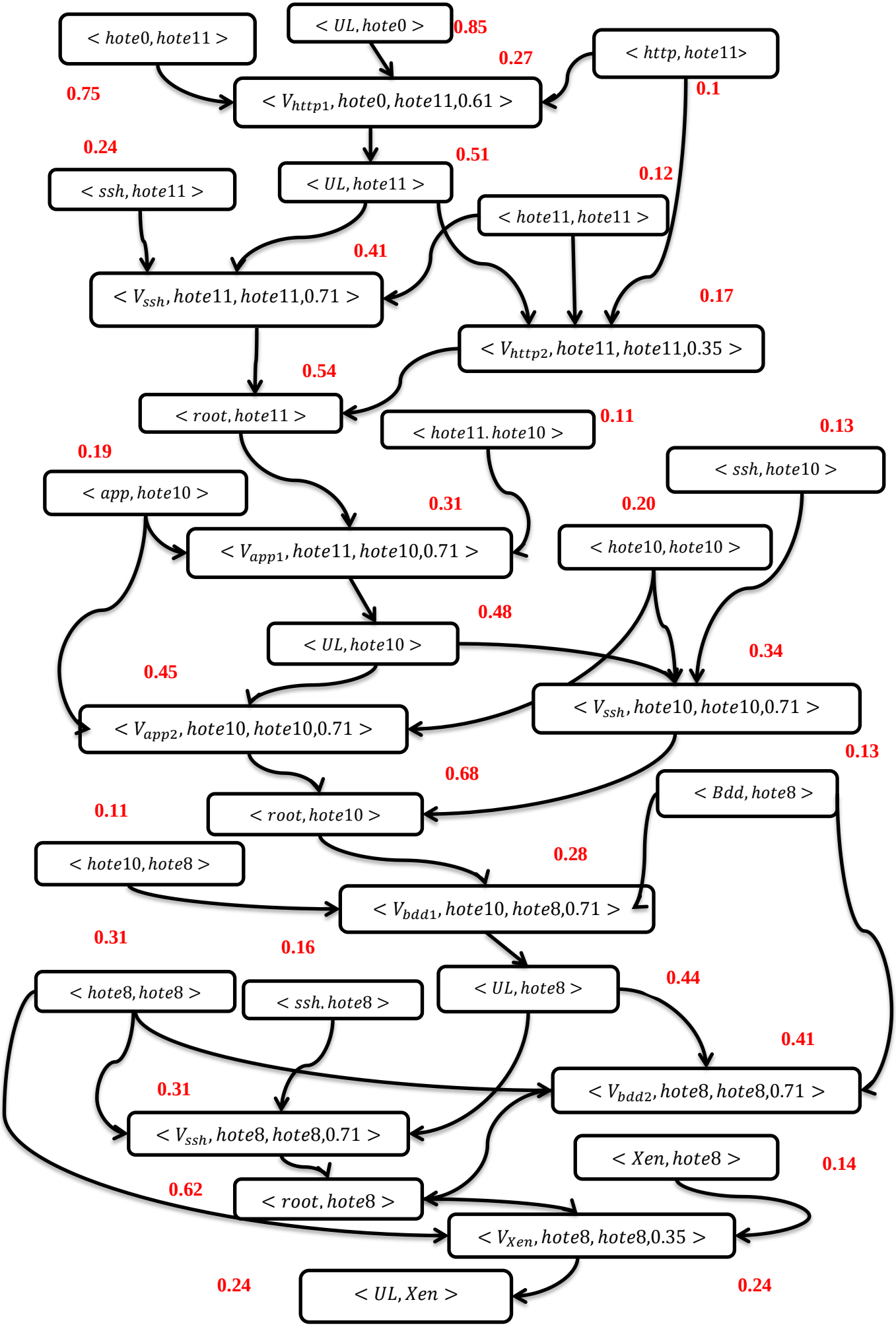


Figure 3.5 : Le niveau de risque de chaque nœud sur le graphe d'attaque

3.3.4 Mapping de l'alerte

On suppose que l'alerte envoyée par l'IDS comporte les informations suivantes :

- Adresse source : hôte 11.
- Adresse destination : hôte 10.
- CVE de la vulnérabilité exploitée : 2006-0586.
- Type de l'alerte : accès non autorisé.

Après l'application de l'algorithme mapping sur le graphe d'attaque, on a trouvé que l'alerte dans la figure 3.6 est correspond au couple : $< V_{app1}, hote11, hote10,0.71 >$ et $< UL, hote10 >$ (nœud avec la couleur rouge). Alors l'attaquant a réussi a exploité la vulnérabilité V_{app1} pour gagner l'accès utilisateur à l'hôte 10.

- **Extraire le chemin le plus dangereux**

En appliquant l'algorithme Chemin-dangereux, nous avons obtenu dix-neuf chemins qui traversent le nœud qui déclenche l'alerte, le tableau 3.6 les montre:

Chemin	Risque
Ch1=[<hote11,hote11>,<Vhttp2,hote11,hote11>,<root,hote11>,<Vapp1,hote11,hote10,0.71>, <UL,hote10>,<Vssh,hote10,hote10,0.71>,<root,hote10>,<Vbdd1,hote10,hote8,0.71>, <UL,hote8>, <Vssh,hote8,hote8,0.71>,<root,hote8>,<VXen,hote8,hote8,0.35>,<UL,Xen>]	6.1 $* 10^{-7}$
Ch2=[<hote11,hote11>,<Vhttp2,hote11,hote11,0.35>,<root,hote11>,<Vapp1,hote11,hote10,0.71>,<UL,hote10>,<Vssh,hote10,hote10,0.71>,<root,hote10>,<Vbdd1,hote10,hote8,0.71>, <UL,hote8>,<Vbdd2,hote8,hote8,0.71>,<root,hote8>,<Xen,hote8,hote8,0.35>,<UL,Xen>]	8.13 $* 10^{-7}$
Ch3=[<hote11,hote11>,<Vhttp2,hote11,hote11,0.35>,<root,hote11>,<Vapp1,hote11,hote10,0.71>,<UL,hote10>,<Vapp2,hote10,hote10,0.71>,<root,hote10,<Vbdd1,hote10,hote8,0.71>, <UL,hote8>,<Vssh,hote8,hote8,0.71>,<root,hote8>, <VXen,hote8,hote8,0.35>, <UL,Xen>]	8.13 $* 10^{-7}$
Ch4=[<hote11,hote11>,<Vhttp2,hote11,hote11,0.35>,<root,hote11>,<Vapp1,hote11,hote10,0.71>,<UL,hote10>,<Vapp2,hote10,hote10,0.71>,<root,hote10>,<Vbdd1,hote10,hote8,0.71>, <UL,hote8>,<Vbdd2,hote8,hote8,0.71>,<root,hote8>, <VXen,hote8,hote8,0.35>, <UL,Xen>]	1.071 $* 10^{-6}$
Ch4=[<http,hote11>,<Vhttp2,hote11,hote11,0.35>,<root,hote11>,<Vapp1,hote11,hote10,0.71>,<UL,hote10>,<Vssh,hote10,hote10,0.71>,<root,hote10>,<Vbdd1,hote10,hote8,0.71>,<UL,hote8>,<Vssh,hote8,hote8,0.71>, <root,hote8>, <VXen,hote8,hote8,0.35>, <UL,Xen>]	5.14 $* 10^{-7}$
Ch5=[<http,hote11>,<Vhttp2,hote11,hote11,0.35>,<root,hote11>,<Vapp1,hote11,hote10,0.71>,<UL,hote10>,<Vssh,hote10,hote10,0.71>,<root,hote10>, <Vbdd1,hote10,hote8,0.71>, <UL,hote8>,<Vbdd2,hote8,hote8,0.71>,<root,hote8>, <VXen,hote8,hote8,0.35>, <UL,Xen>]	6.78 $* 10^{-7}$

Ch6=[<http,hote11>,<Vhttp2,hote11,hote11,0.35>,<root,hote11>,<Vapp1,hote11,hote10,0.71> ,<UL,hote10>,<Vapp2,hote10,hote10,0.71>,<root,hote10>,<Vbdd1,hote10,hote8,0.71>, <UL,hote8>,<Vssh,hote8,hote8,0.71>,<root,hote8>, <VXen,hote8,hote8,0.35>,<UL,Xen>]	$6.78 * 10^{-7}$
Ch7=[<http,hote11>,<http2,hote11,hote11,0.35>,<root,hote11>,<Vapp1,hote11,hote10,0.71>, <UL,hote10>,<Vapp2,hote10,hote10,0.71>, <root,hote10>,<Vbdd1,hote10,hote8,0.71>, <UL,hote8>,<Vbdd2,hote8,hote8,0.71>, <root,hote8>,<VXen,hote8,hote8,0.35>,<UL,Xen>]	$8.93 * 10^{-7}$
Ch8=[<UL,hote0>,<Vhttp1,hote0,hote11,0.61>,<UL,hote11>,<Vhttp2,hote11,hote11,0.35>, <root,hote11>,<Vapp1,hote11,hote10,0.71>, <UL,hote10>,<Vssh,hote10,hote10,0.71>, <root,hote10>, <Vbdd1,hote10 hote8,0.71>, <UL,hote8>,<ssh,hote8,hote8,0.71> ,<root,hote8>, <VXen,hote8,hote8,0.35>, <UL,Xen>]	$6.22 * 10^{-7}$
Ch9=[<UL,hote0>,<Vhttp1,hote0,hote11,0.61>,<UL,hote11>,<Vhttp2,hote11,hote11,0.35>,<root,hote11>,<Vapp1,hote11,hote10,0.71>,<UL,hote10>,<Vssh,hote10,hote10,0.71>,<root,hote10>,<Vbdd1,hote10,hote8,0.71>,<UL,hote8>,<Vbdd2,hote8,hote8,0.71>,<root,hote8>, <VXen,hote8,hote8,0.35>,<UL,Xen>]	$8.20 * 10^{-7}$
Ch10=[<UL,hote0>,<Vhttp1,hote0,hote11,0.61>,<UL,hote11>,<Vhttp2,hote11,hote11,0.35>,<root,hote11>,<Vapp1,hote11,hote10,0.71>,<UL,hote10>, <Vapp2,hote10,hote10,0.71>, <root,hote10>,<Vbdd1,hote10, hote8,0.71>, <UL,hote8>,<Vssh,hote8,hote8,0.71>, <root,hote8>,<VXen,hote8,hote8,0.35>,<UL,Xen>]	$8.19 * 10^{-7}$
Ch11=[<UL,hote0>,<Vhttp1,hote0,hote11,0.61>,<UL,hote11>,<Vhttp2,hote11,hote11,0.35>,<root,hote11>,<Vapp1,hote11,hote10,0.71>,<UL,hote10>,<Vapp2,hote10,hote10,0.71>, <root,hote10>,<Vbdd1,hote10,hote8,0.71>,<UL,hote8>,<Vbdd2,hote8,hote8,0.71>, <root,hote8>,<VXen,hote8,hote8,0.35>, <UL,Xen>]	$1.079 * 10^{-6}$
Ch12=[<http,hote11>,<Vhttp1,hote0,hote11,0.61>,<UL,hote11>,<Vhttp2,hote11,hote11,0.35>, <root,hote11>,<Vapp1,hote11,hote10,0.71>,<UL,hote10>,<Vssh,hote10,hote10,0.71>, <root,hote10>,<Vbdd1,hote10,hote8,0.71>,<UL,hote8>,<Vssh,hote8,hote8,0.71>,<root,hote8>, <VXen,hote8,hote8,0.35>, <UL,Xen>]	$7.32 * 10^{-8}$
Ch13=[<http,hote11>,<Vhttp1,hote0,hote11,0.61>,<UL,hote11>,<Vhttp2,hote11,hote11,0.35>, <root,hote11>,<Vapp1,hote11,hote10,0.71>,<UL,hote10>,<Vssh,hote10,hote10,0.71>, <root,hote10>,<Vbdd1,hote10,hote8,0.71>, <UL,hote8>, <Vbdd2,hote8,hote8,0.71>, <root,hote8>, <VXen,hote8,hote8,0.35>, <UL,Xen>]	$9.64 * 10^{-8}$
Ch14=[<http,hote11>,<Vhttp1,hote0,hote11,0.61>,<UL,hote11>,<Vhttp2,hote11,hote11,0.35>, <root,hote11>,<Vapp1,hote11,hote10,0.71>,<UL,hote10>,<Vapp2,hote10,hote10,0.71>, <root,hote10>,<Vbdd1,hote10,hote8,0.71>,<UL,hote8>,<Vssh,hote8,hote8,0.71>, <root,hote8> , <VXen,hote8,hote8,0.35>, <UL,Xen>]	$9.64 * 10^{-8}$
Ch15= [<http,hote11>,<Vhttp1,hote0,hote11,0.61>,<UL,hote11>, <Vhttp2,hote11, hote11,0.35>,<root,hote11>,<Vapp1,hote11, hote10,0.71>, <UL,hote10>, <Vapp2,hote10,hote10,0.71>,<root,hote10>,<Vbdd1,hote10,hote8,0.71>,<UL,hote8>, <Vbdd2,hote8,hote8,0.71>,<root,hote8>,<VXen,hote8,hote8,0.35>, <UL,Xen>]	$1.27 * 10^{-7}$

Ch16=[<hote0,hote1>,<Vhttp1,hote0,hote11,0.61>,<UL,hote11>, <Vhttp2,hote11,hote11,0.35> ,<root,hote11>,<Vapp1,hote11,hote10,0.71>, <UL,hote10>,<Vssh,hote10,hote10,0.71>, <root,hote10>, <Vbdd1,hote10,hote8,0.71>, <UL,hote8>,<Vssh,hote8,hote8,0.71>, <root,hote8>, <VXen,hote8,hote8,0.35>, <UL,Xen>]	$5.49 * 10^{-7}$
Ch17= [<hote0,hote11>,<Vhttp1,hote0,hote11,0.61>,<UL,hote11>, <Vhttp2,hote11, hote11,0.35>,<root,hote11>,<Vapp1,hote11, hote10,0.71>, <UL,hote10>, <ssh,hote10,hote10,0.71>,<root,hote10>,<Vbdd1,hote10,hote8,0.71>,<UL,hote8>,<Vbdd2,hote 8,hote8,0.71>,<root,hote8>,<VXen,hote8,hote8,0.35>,<UL,Xen>]	$7.23 * 10^{-7}$
Ch18= [<hote0,hote11>,<Vhttp1,hote0,hote11,0.61>,<UL,hote11>, <Vhttp2,hote11, hote11,0.35>,<root,hote11>,<Vapp1,hote11,hote10,0.71>, <UL,hote10>, <Vapp2,hote10,hote10,0.71>,<root,hote10>,<Vbdd1,hote10,hote8,0.71>,<UL,hote8>, <Vssh,hote8,hote8,0.71>,<root,hote8>, <VXen,hote8,hote8,0.35>, <UL,Xen>]	$7.23 * 10^{-7}$
Ch19=[<hote0,hote11>,<Vhttp1,hote0,hote11,0.61>,<UL,hote11>,<Vhttp2,hote11,hote11,0.35 >,<root,hote11>,<Vapp1,hote11,hote10,0.71>,<UL,hote10>,<Vapp2,hote10,hote10,0.71>, <root,hote10>,<Vbdd1,hote10,hote8,0.71>, <UL,hote8>,<Vbdd2,hote8,hote8,0.71>, <root,hote8> ,<VXen,hote8,hote8,0.35>, <UL,Xen>]	$9.52 * 10^{-7}$

Tableau 3.6 les chemins d'attaques possibles et leur risques

Donc le chemin le plus dangereux est : Ch11 avec risque $= 1.079 * 10^{-6}$ comme il est représenté dans la figure 3.7.

On va calculer les valeurs associées au chemin Ch_{11} (tableau 3.7) qui sont nécessaires pour l'évaluation des contremesures.

La vulnérabilité exploitée et l'hôte cible	La gravité et sa valeur monétaire	Le coût d'équipement
<Vhttp1,hote0,hote11,0.61>	Significative=10^5 DA	10^4 DA
<Vhttp2,hote11,hote11,0.35>	Mineur=10^4 DA	10^4 DA
<Vapp1,hote11,hote10,0.71>	Significative=10^5 DA	10^5 DA
<Vapp2,hote10,hote10,0.71>	Grave=10^8 DA	10^5 DA
<Vbdd1,hote10,hote8,0.71>	Significative=10^5 DA	10^6 DA
<Vbdd2,hote8,hote8,0.71>	Sérieux=10^6 DA	10^6 DA
<VXen,hote8,hote8,0.35>	Grave=10^8 DA	10^6 DA

Tableau 3.7 le calcul des attributs PE et VI pour chaque contremesure.

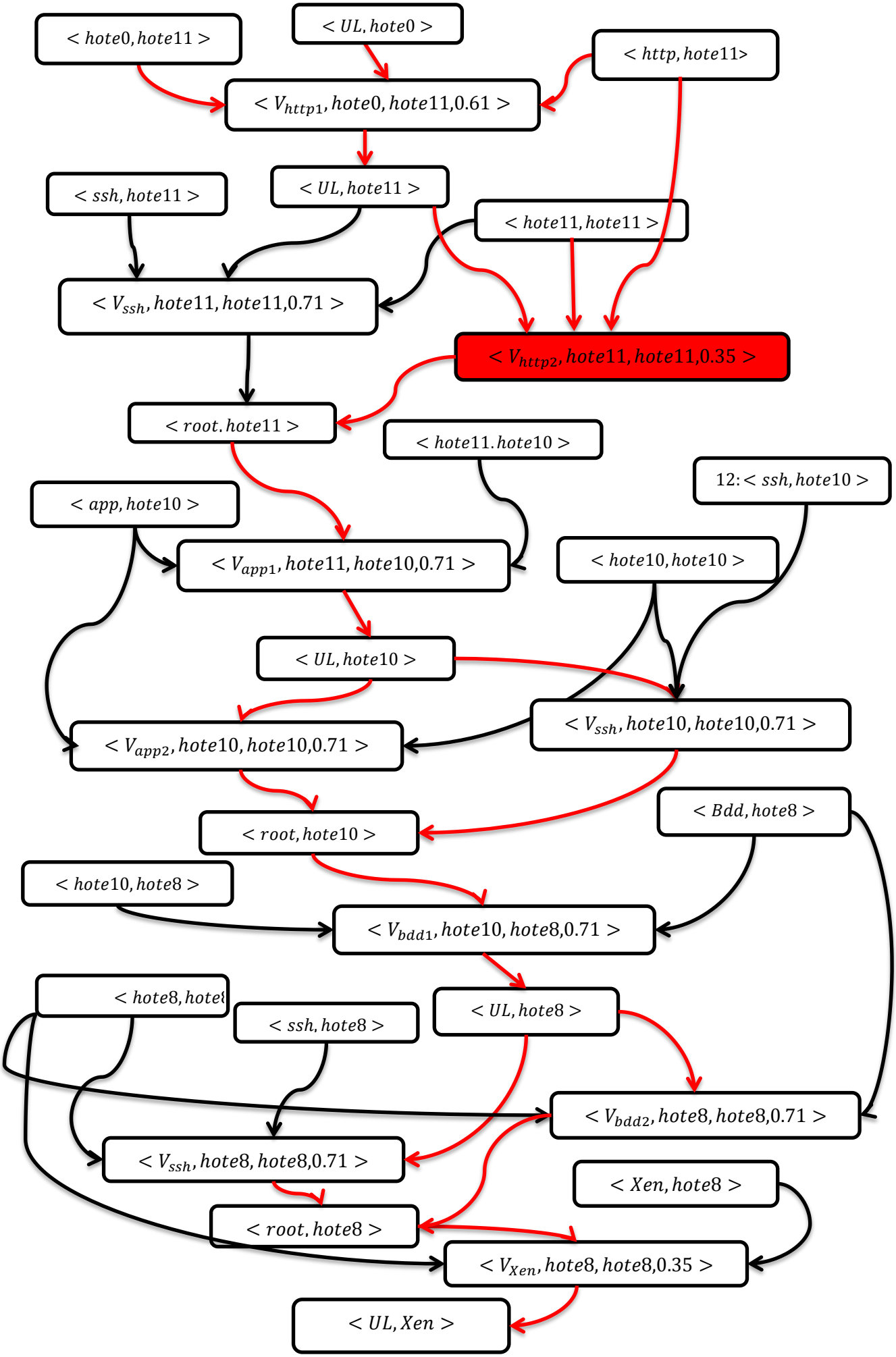


Figure 3.6 : Les chemins d'attaque qui traversent l'alerte choisie

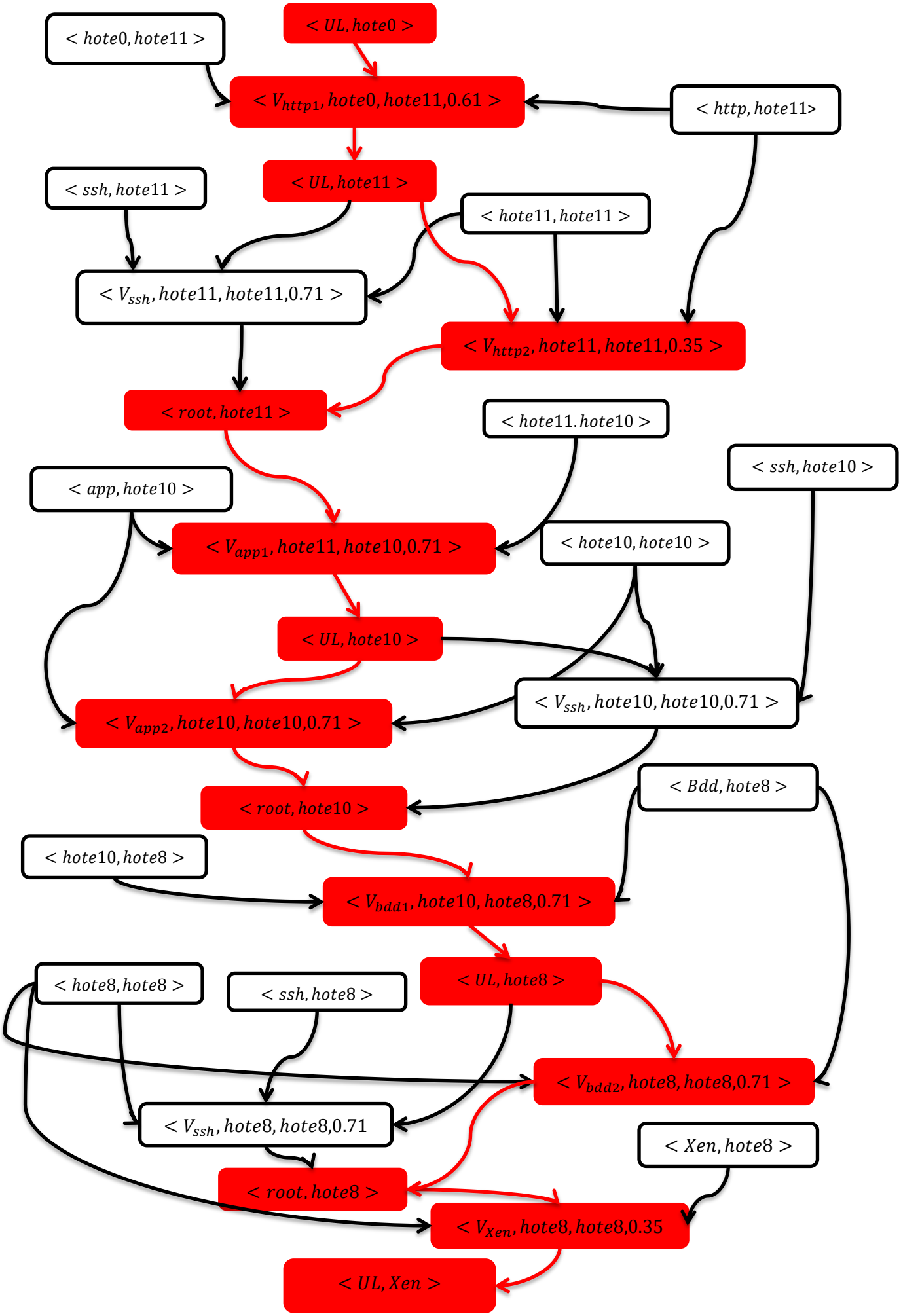


Figure 3.7 : Le chemin d'attaque le plus dangereux

- La perte totale *associée au chemin* Ch_{11} *est:*

$$PE_{ChD_{11}} = \sum_i GR_i = 20131 * 10^4\ DZ$$

- La valeur totale d'infrastructure *associée au chemin* Ch_{11} *est:*

$$VI_{ChD_{11}} = \sum_i CE_i = 111 * 10^4\ DZ$$

3.3.5 La sélection des contremesures

a) **Bassin des contremesures possibles :** Les contremesures sont considérées comme des règles à appliquer au niveau du pare-feu afin d'arrêter la propagation d'attaque, donc on va citer les contremesures pour chaque serveur du Datacenter, où les pare-feu PF2, PF6 représentent respectivement noeud2 et 6 sur la figure 3.2 :

CMs pour le pare-feu (noeud2-couche1)	CMs pour le pare-feu (noeud6-hote8)
$\boldsymbol{cm_1 = BlocIP(PF2, s, hote0)}$	$cm_8 =$ BlocIP(PF6 − H8, s, VM APP)
$\boldsymbol{cm_2 = BlocIP(PF2, r, VM\ HTTP)}$	$cm_9 =$ BlocPort(× PF6 − H8, r, mysql)
$\boldsymbol{cm_3 = BlocPort(PF2, r, http)}$	$cm_{10} =$ BlocIPPort(PF6 − H8, VM BDD, mysql)
$\boldsymbol{cm_4 = BlocIPPort(PF2, s, hote0, http)}$	$cm_{11} =$ BlocTouteTraffic(PF6 − H8)
$\boldsymbol{cm_5 = BlocTouteTraffic(PF2)}$	$cm_{12} =$ FermerConnexion(mysql)
$\boldsymbol{cm_6 = Redemarer(PF2)}$	$cm_{13} =$ Déconnecter(VM BDD)
$\boldsymbol{cm_7 = FermerConnexion(http)}$	$cm_{14} =$ eteindre(Switch7 − H8)
	$cm_{15} =$ Redemarer(VM BDD)
	$cm_{16} =$ eteindre(VM BDD)
	$cm_{17} =$ Tuer(mysql)
	$cm_{18} =$ Redémarrer(mysql)
	$cm_{19} =$ Désactiver(mysql)
CMs pour le pare-feu (noeud6-hote11)	**CMs pour le pare-feu (noeud6-hote10)**
$\boldsymbol{cm_{20} = BlockIP(PF6 - H11, s, hote0)}$	$cm_{27} =$ BlocIP(PF6 − H10, r, VM APP)
$\boldsymbol{cm_{21} = BlockIP(PF6 - H11, r, VM\ HTTP)}$	$cm_{28} =$ BlocIPPort(PF6 − H10, VM APP, app)
$\boldsymbol{cm_{22} = BlockIPPort(PF6 - H11, s, hote0, http)}$	$cm_{30} =$ FermerConnexion(VM APP)
$\boldsymbol{cm_{23} = BlocIPPort(PF6 - H11, VM\ HTTP, http)}$	$cm_{31} =$ BlockIPPort(PF6 − H10, s, VM HTTP, app)
$\boldsymbol{cm_{24} = FermerConnexion(http)}$	$cm_{32} =$ Déconnectre(VM APP)
$\boldsymbol{cm_{25} = Déconnecter(VM\ HTTP)}$	$cm_{33} =$ eteindre(VM APP)
$\boldsymbol{cm_{26} = eteindre(VM\ HTTP)}$	$cm_{34} =$ Redemarer(VM APP)

Tableau 3.8 Le bassin des contremesures appropriées avec le Ch_{11}

b) L'évaluation des contremesures avec CRORI

Dans Le tableau 3.9 on a choisi quelques contremesures à évaluer présentant les vulnérabilités couvris, et les valeurs associes à chaque contremesure :

CMs	V_{http1}	V_{http2}	V_{app1}	V_{app2}	V_{bdd1}	V_{bdd2}	V_{Xen}	*IN*	*DR*	*CR*
cm_1	×	×	×	×	×	×	×	Négligeable= 0	1	10^3
cm_3	×	×	×	×	×	×	×	Faible= 10^3	1	10^3
cm_5	×	×	×	×	×	×	×	Extrême=10^6	1	10^3
cm_{11}					×	×	×	Très haut=10^5	0. 43	10^3
cm_{12}					×	×	×	Très haut=10^5	0.43	10^3
cm_{16}					×	×	×	Très haut=10^5	0.43	10^4
cm_{21}	×	×	×	×	×	×	×	Faible=10^3	1	10^3
cm_{24}	×	×	×	×	×	×	×	Faible=10^3	1	10^3
cm_{26}	×	×	×	×	×	×	×	Faible=10^3	1	10^4
cm_{27}			×	×	×	×	×	Très haut=10^5	0.71	10^3
cm_{31}			×	×	×	×	×	Très haut=10^5	0.71	10^3
cm_{33}			×	×	×	×	×	Très haut=10^5	0.71	10^4

Tableau 3.9 Calcul des attributs associés à chaque contremesure

c) Établir le graphe de dépendance des contremesures (GDCM)

Pour établir le *GDCM* on va adopter le tableau 3.9 comme il est présenté dans le figure 3.8 :

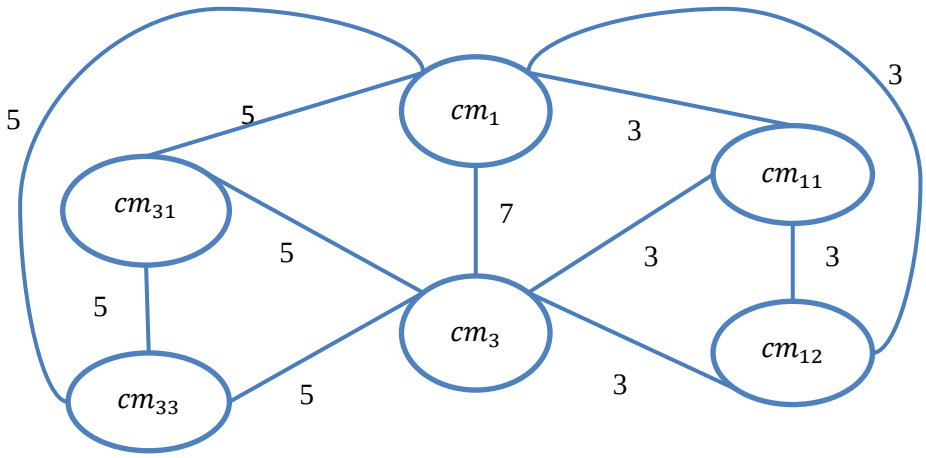


Figure 3.8 : Une partie du graphe de dépendance entre les contremesures

d) Evaluation avec CRORI

Après l'extraction des contremesures et leurs attributs, maintenant on va évaluer les contremesures avec CRORI (équation 9) et extraire l'optimale parmi eux. Le tableau 3.10 nous montre les résultats du calcul :

Contremesures	*CRORI*
$\boldsymbol{cm_1}$	$CRORI_{cm_1} = \frac{(20131 * 10^4 * 1) - (10^3 + 0)}{111 * 10^4 + 10^3 + 0} = 181.19$
$\boldsymbol{cm_3}$	$CRORI_{cm_3} = \frac{(20131 * 10^4 * 1) - (10^3 + 10^3)}{111 * 10^4 + 10^3 + 10^3} = 181.03$
$\boldsymbol{cm_5}$	$CRORI_{cm_5} = \frac{(20131 * 10^4 * 1) - (10^3 + 10^6)}{111 * 10^4 + 10^3 + 10^6} = 94.88$
$\boldsymbol{cm_{11}}$	$CRORI_{cm_{11}} = \frac{(20131 * 10^4 * 0.43) - (10^3 + 10^5)}{111 * 10^4 + 10^3 + 10^5} = 71.39$
$\boldsymbol{cm_{12}}$	$CRORI_{cm_{12}} = \frac{(20131 * 10^4 * 0.43) - (10^3 + 10^5)}{111 * 10^4 + 10^3 + 10^5} = 71.39$
$\boldsymbol{cm_{16}}$	$CRORI_{cm_{16}} = \frac{(20131 * 10^4 * 0.43) - (10^4 + 10^5)}{111 * 10^4 + 10^4 + 10^5} = 65.91$
$\boldsymbol{cm_{21}}$	$CRORI_{cm_{21}} = \frac{(20131 * 10^4 * 1) - (10^3 + 10^3)}{111 * 10^4 + 10^3 + 10^3} = 181.03$
$\boldsymbol{cm_{24}}$	$CRORI_{cm_{24}} = \frac{(20131 * 10^4 * 1) - (10^3 + 10^3)}{111 * 10^4 + 10^3 + 10^3} = 181.03$
$\boldsymbol{cm_{26}}$	$CRORI_{cm_{26}} = \frac{(20131 * 10^4 * 1) - (10^4 + 10^3)}{111 * 10^4 + 10^4 + 10^3} = 181.18$
$\boldsymbol{cm_{27}}$	$CRORI_{cm_{27}} = \frac{(20131 * 10^4 * 0.71) - (10^3 + 10^5)}{111 * 10^4 + 10^3 + 10^5} = 117.94$
$\boldsymbol{cm_{31}}$	$CRORI_{cm_{31}} = \frac{(20131 * 10^4 * 0.71) - (10^3 + 10^5)}{111 * 10^4 + 10^3 + 10^5} = 117.94$
$\boldsymbol{cm_{33}}$	$CRORI_{cm_{31}} = \frac{(20131 * 10^4 * 0.71) - (10^4 + 10^5)}{111 * 10^4 + 10^4 + 10^5} = 117.06$

Tableau 3.10 l'évaluation des contremesures avec CRORI

Après le calcul du *CRORI* de chaque contremesure on a remarqué que ($cm_1 =$ BlocIP(PF2, s, hote0)) est la contremesure optimale avec *CRORI* le plus grand.

Avant d'installer cm_1, on doit ajuster leurs valeurs associées avec la contremesure qui était déjà déployée. Dans notre exemple on ne va pas ajuster les valeurs, car on va considérer cm_1 la première déployée.

3.3 Conclusion

Dans ce chapitre nous avons présenté notre architecture proposée munie d'un exemple illustratif, qui explique notre démarche. Dans le Chapitre 4 on va présenter notre application en détaillant son fonctionnement en utilisant l'exemple au-dessus comme donnée.

Chapitre 4 : Implémentation

4.1 Introduction

Dans ce chapitre nous présentons notre application, ses objectifs et les outils étaient utilisés pour le développement, en présentant le déroulement des étapes par des interfaces comme un exemple.

4.2 Présentation de l'application

4.2.1 Contexte

Notre application permet à partir d'un graphe d'attaque et des alertes reçues d'extraire tous les chemins d'attaque en calculant leurs risques (évaluation de risque) afin de choisir le chemin le plus dangereux, après nous avons choisir les contremesures appropries avec lui, ces dernières sont évaluées avec notre métrique proposée *CRORI* afin de choisir parmi eux la contremesure optimale l'optimale.

4.2.2 Objectifs

Notre application permet à un administrateur système de :

- Importer et un graphe d'attaque sous forme d'un fichier XML[1].
- Visualiser le graphe d'attaque et ses probabilités conditionnelles, et le risque de chaque nœud.
- Importer et visualiser un graphe de dépendance des composants sous forme XML et visualiser les informations de chaque composant.
- Visualiser tous les chemins probables avec ses risques.
- Visualiser le chemin le plus dangereux.
- Importer le graphe de dépendance des contremesures sous forme XML.
- Visualiser le graphe de dépendance des contremesures et les attribues de chaque contremesure.

[1] https://www.w3.org/TR/xml

- Visualiser la contremesure optimale avec le chemin le plus dangereux.

4.3 Outils et Environnement de développement

Pour réaliser notre architecture proposée, nous avons utilisé le langage de programmation Java (JDK 8 et JRE 8) dans l'environnement Eclipse, avec l'interface de programmation(API) JGraphX, JDOM.

4.3.1 Le langage Java[2]

Le langage Java est un langage généraliste de programmation synthétisant les principaux langages existants lors de sa création en 1995 par *Sun Microsystems*. Il permet une programmation orientée-objet (à l'instar de SmallTalk et, dans une moindre mesure, C++), modulaire (langage ADA) et reprend une syntaxe très proche de celle du langage C.Outre son orientation objet, le langage Java a l'avantage d'être modulaire (on peut écrire desportions de code génériques, c.-à-d. utilisable par plusieurs applications), rigoureux (la plupart des erreurs se produisent à la compilation et non à l'exécution) et portable (un même programme compilé peut s'exécuter sur différents environnements) [16].

4.3.2Eclipse[3]

Eclipse est une plateforme de développement Java gratuite, connue pour ses plug-ins permettant aux développeurs de développer et de tester du code écrit dans d'autres langages de programmation. Eclipse est publié sous les termes de la licence publique Eclipse [45].

4.3.3 JGrahT[4]

JGraphT est une bibliothèque JAVA permet de créer et manipuler des graphes. Elle permet aussi d'utiliser plusieurs techniques reliées avec les graphes comme l'algorithme de Dijikstra qui trouve le plus court chemin. On a utilisé cette API JAVA pour crées nos graphes dans l'application.

[2] https://www.java.com/
[3] https://www.eclipse.org/
[4] https://jgrapht.org/

4.3.4 JGraphX[5]

JGraphx est une bibliothèque Java qui permet de dessiner des graphes en 2D dans une application, JGraphX vous permet de créer des applications Java Swing comportant des fonctionnalités de création de diagrammes interactifs. La fonctionnalité client principale de JGraphX est une bibliothèque compatible avec Java 5 qui décrit, affiche et interagit avec les diagrammes dans le cadre de votre plus grande application Java Swing. JGraphX est principalement conçu pour être utilisé dans un environnement de bureau [45]. On a utilisé cette API JAVA pour la visualisation des graphes dans l'application.

4.3.5 JDOM[6]

JDOM est une API open source Java dont le but est de représenter et manipuler un document XML de manière intuitive pour un développeur Java sans requérir une connaissance pointue de XML. Par exemple, JDOM utilise des classes plutôt que des interfaces. Ainsi pour créer un nouvel élément, il faut simplement instancier une classe.

Le but de JDOM n'est pas de définir un nouveau type de parseur, mais de faciliter la manipulation au sens large de document XML : lecture d'un document, représentation sous forme d'arborescence, manipulation de cet arbre, définition d'un nouveau document, exportation vers plusieurs formats cibles [10]. Cette API JAVA est utilisée dans notre application pour importer les informations depuis les fichiers XML.

4.4 Déroulement de l'application

4.4.1 L'interface principale de l'application

La figure 4.1 montre le code source Java dans l'environnement Eclipse. La figure 4.2 montre l'interface principale de notre application. Les données utilisées pour tester l'application sont tirées de l'exemple présenté dans le chapitre précédent. On a assumé que les alertes concernent les nœuds <http2,hote11,VM HTTP hote11,0.35> et <bdd1,hote10,VM BDD hote8,0.71> dans le graphe d'attaque.

[5] https://github.com/jgraph/jgraphx
[6] http://www.jdom.org/

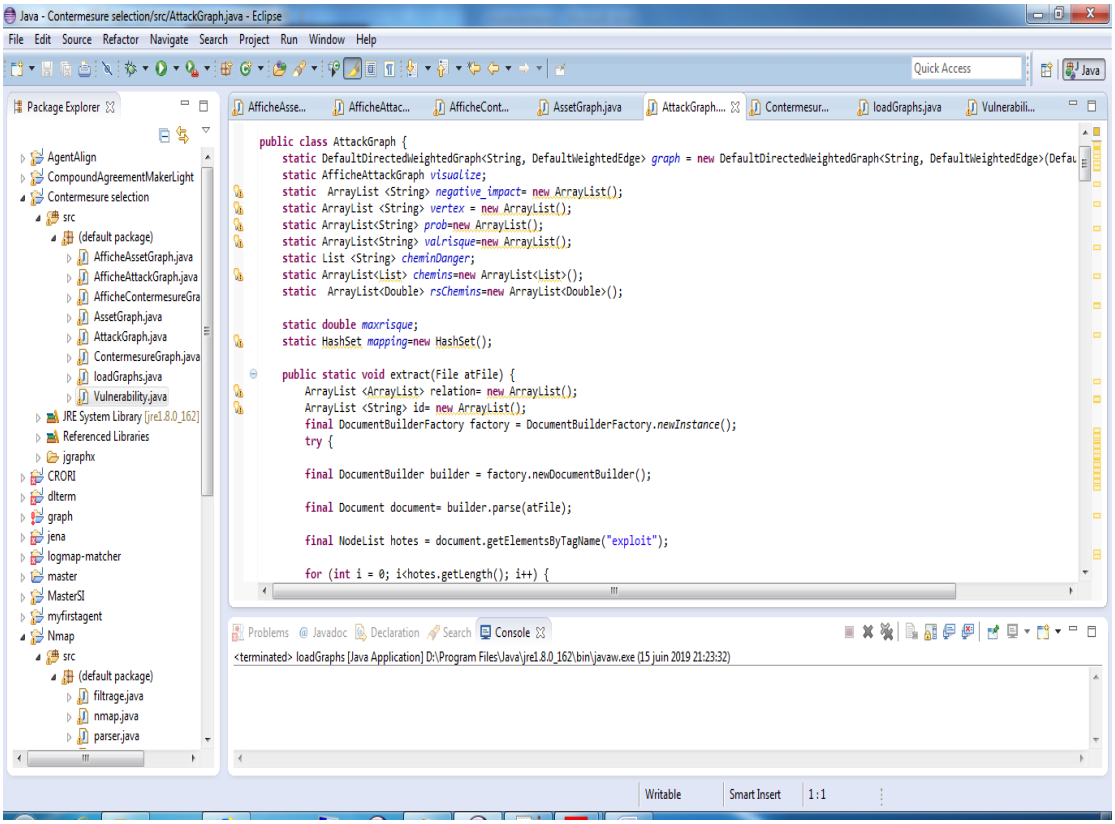
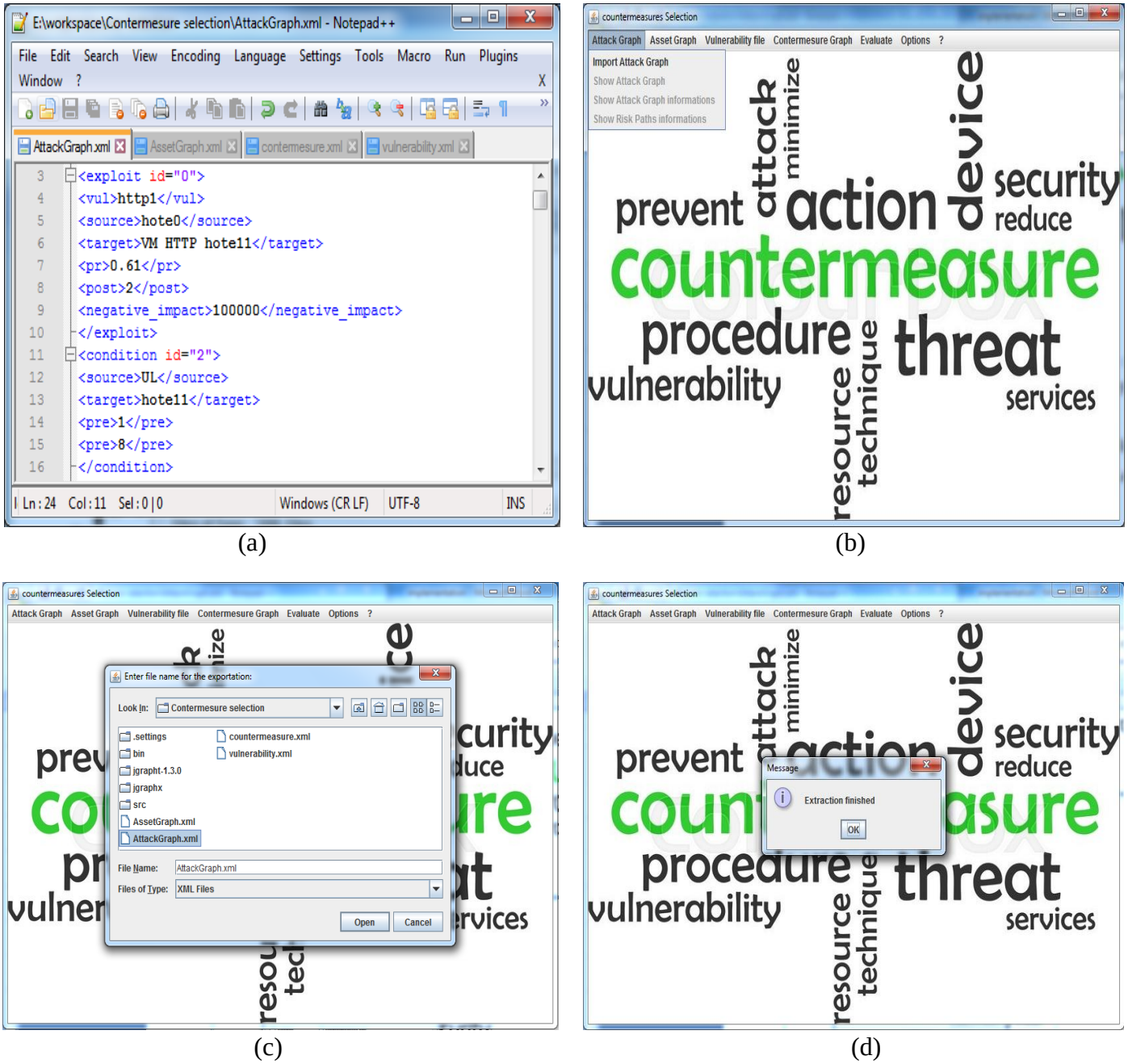


Figure 4.1 : Le code source Java dans l'environnement Eclipse

Figure 4.2 : interface principale de l'application

4.4.2 L'importation des fichiers XML

Il faut aller vers le menu « Attack Graph » en ensuite cliquer sur « import Attack Graph » et sélectionner le fichier XML correspondant. La figure 4.3(a) présente le contenu et la structure du fichier XML du graphe d'attaque. Les figures 4.3 (b) (c) et (d) montrent le déroulement de ces étapes sur l'importation du graphe d'attaque. Il faut suivre ces étapes pour Les autres importations.



(a) (b)

(c) (d)

Figure 4.3 : Importation du graph d'attaque

4.4.3 Visualiser les graphes importés

Afin de visualiser le graphe d'attaque importé, il faut aller vers le menu correspondant. Ensuite cliquer sur « *show Attack Graph* », la figure 4.4 va être affichée. Les mêmes étapes doivent être suivre pour afficher les autres graphes. Les figures 4.5 et 4.6 montrent respectivement la visualisation des graphes de dépendances entre les composants et le graphe de dépendances entre les contremesures.

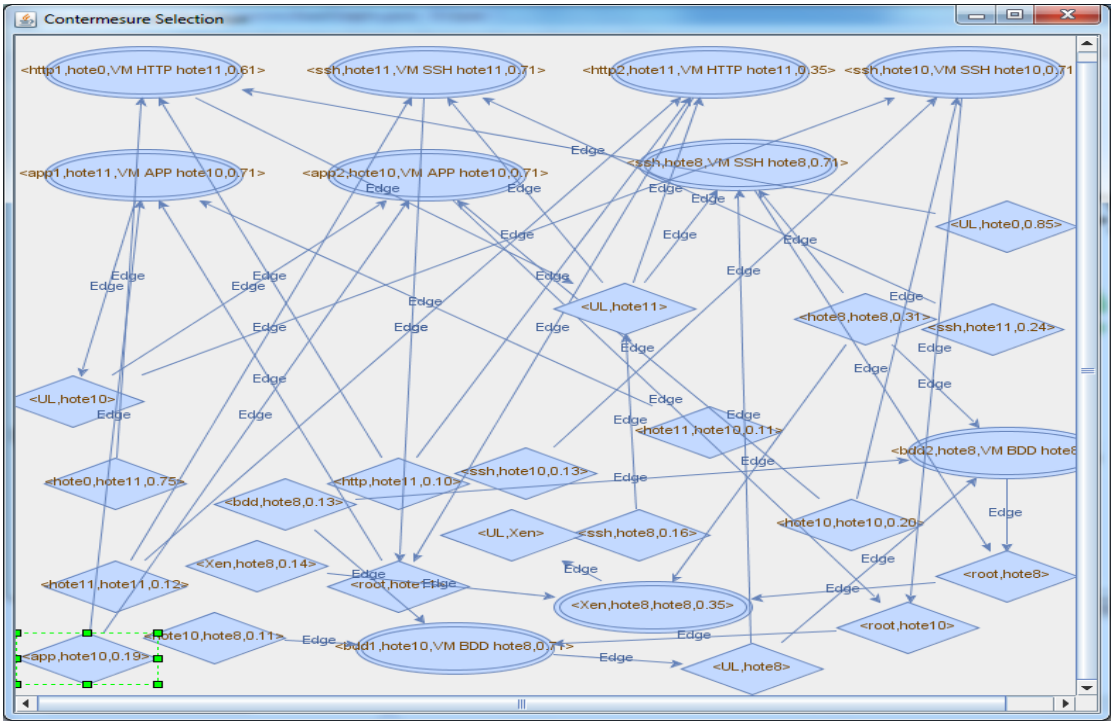


Figure 4.4 : Visualisation du chemin le plus dangereux

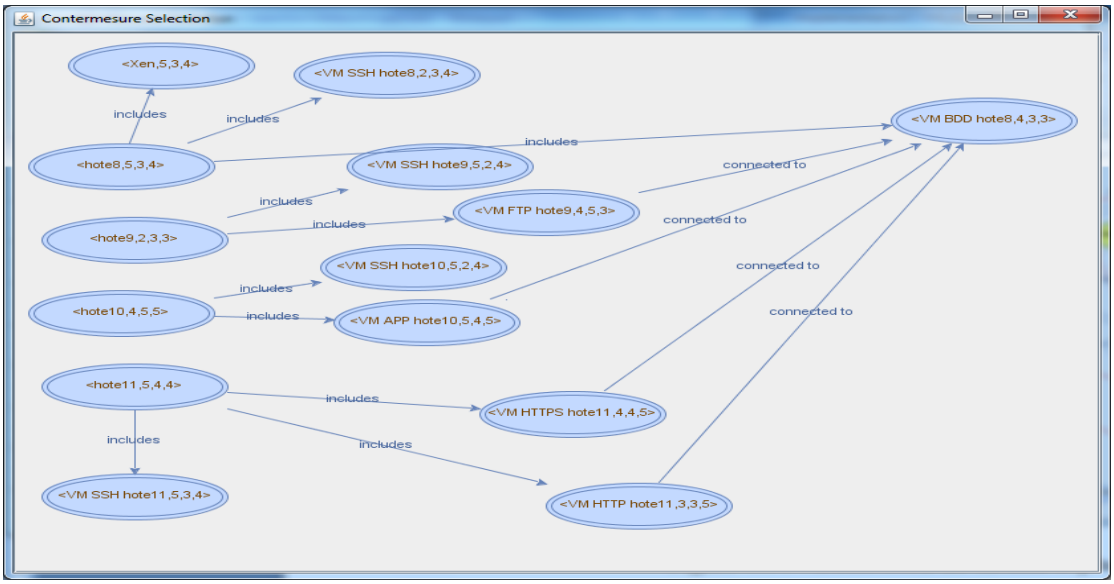


Figure 4.5 : Visualisation du graphe de dépendances entre les composants

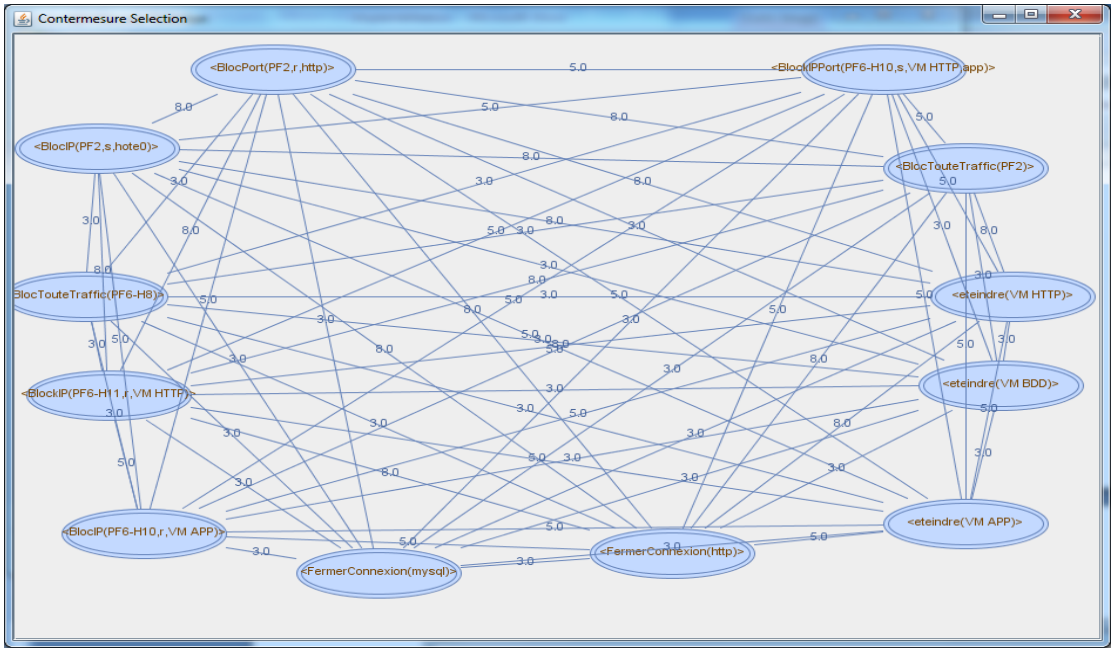


Figure 4.6 : Visualisation du graphe de dépendances entre les contremesures

4.4.4 Visualiser les données des graphes importés

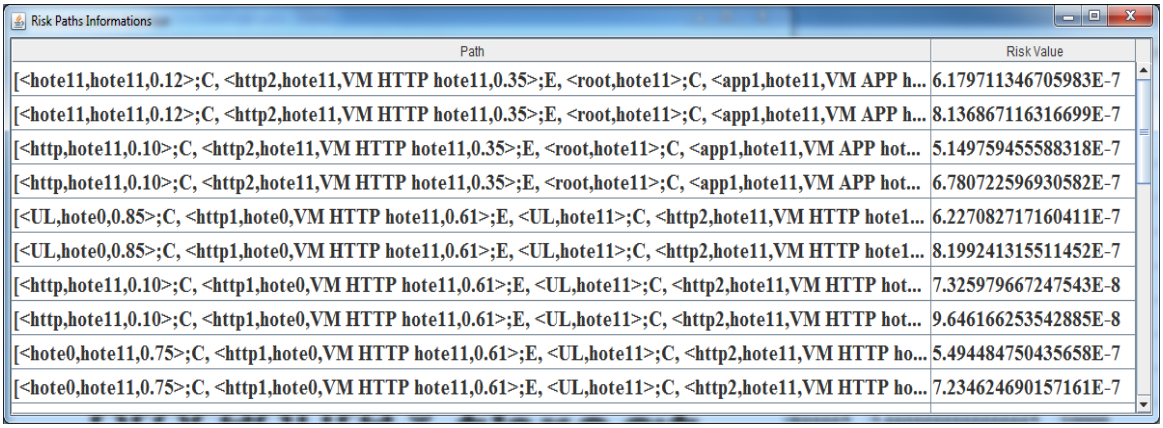
Après l'importation des fichiers XML, l'application va calculer d'autres données additionnelles comme : les probabilités conditionnelles, les risques des nœuds, les risques des chemins, les valeurs des composants…etc. Pour afficher ces informations associées aux nœuds du graphe d'attaque, il faut aller vers le menu correspondant et cliquer sur « show Attack Graph Information » et alors la figure 4.7 va être affichée.

Attack Graph Informations

IdVul	Source	Destination	Acces Prob	Cond Prob	Risk value	Negative Impact
http1	hote0	VM HTTP hote11	0.61	0.5894125	0.2784885411176172	100000
ssh	hote11	VM SSH hote11	0.71	0.5150333467999999	0.41206150852720397	100000
http2	hote11	VM HTTP hote11	0.35	0.23618514499999999	0.1732170815081063	10000
app1	hote11	VM APP hote10	0.71	0.5204021746285347	0.31294098224839534	100000
app2	hote10	VM APP hote10	0.71	0.4893466325030962	0.45676213324935333	100000000
ssh	hote10	VM SSH hote10	0.71	0.47300193861443657	0.3468974111579674	100000
bdd1	hote10	VM BDD hote8	0.71	0.5620541369026907	0.28970413749929297	100000
bdd2	hote8	VM BDD hote8	0.71	0.5233417798517065	0.4187088168579264	1000000
ssh	hote8	VM SSH hote8	0.71	0.5297782702016477	0.31789383332249843	120000
Xen	hote8	hote8	0.35	0.30344939088381545	0.24278003462521855	100000000
	UL	hote11		0.5894125	0.5108241666666666	0
	UL	hote0	0.85	0.85	0.85	0
	hote0	hote11	0.75	0.75	0.75	0
	http	hote11	0.10	0.10	0.1	0
	ssh	hote11	0.24	0.24	0.24	0
	hote11	hote11	0.12	0.12	0.12	0
	root	hote11		0.6295752661062066	0.5456318972920456	0
	hote11	hote10	0.11	0.11	0.11	0
	app	hote10	0.19	0.19	0.19	0
	ssh	hote10	0.13	0.13	0.13	0
	hote10	hote10	0.20	0.20	0.2	0
	UL	hote10		0.5204021746285347	0.4857086963199657	0
	hote10	hote8	0.11	0.11	0.11	0
	root	hote10		0.730886665289122	0.6821608876031806	0
	bdd	hote8	0.13	0.13	0.13	0
	UL	hote8		0.5620541369026907	0.4496433095221526	0
	ssh	hote8	0.16	0.16	0.16	0
	hote8	hote8	0.31	0.31	0.31	0
	Xen	hote8	0.14	0.14	0.14	0
	root	hote8		0.7758649471992656	0.6206919577594125	0
	UL	Xen		0.3034493908838154	0.2427595127070523 2	0

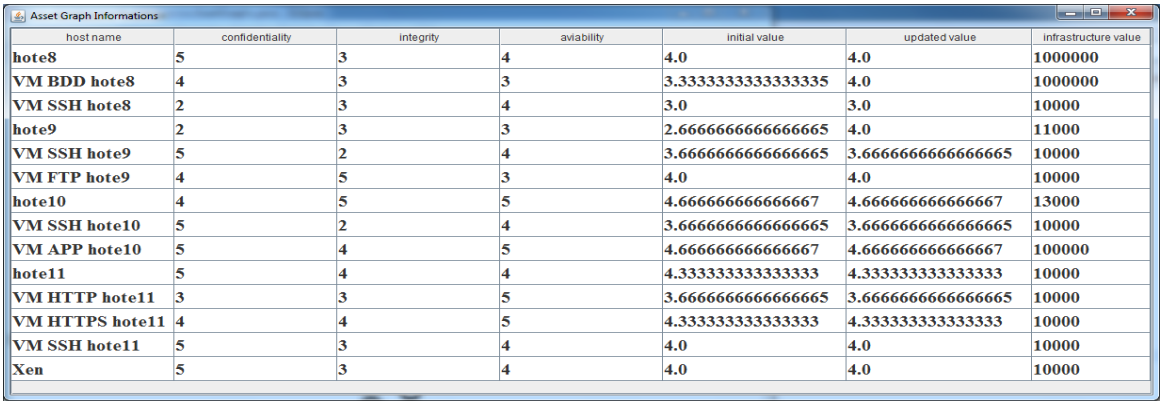
Figure 4.7 : Visualisation des données sur les nœuds du graph d'attaque

Il faut suivre les mêmes étapes pour afficher les informations souhaitées. La figure 4.8 montre les valeurs de risques relatives aux chemins probables. La figure 4.9 présente les données importées et calculées des composants. La figure 4.10 donne la visualisation des attributs importés et calculés des contremesures.


Risk Paths Informations

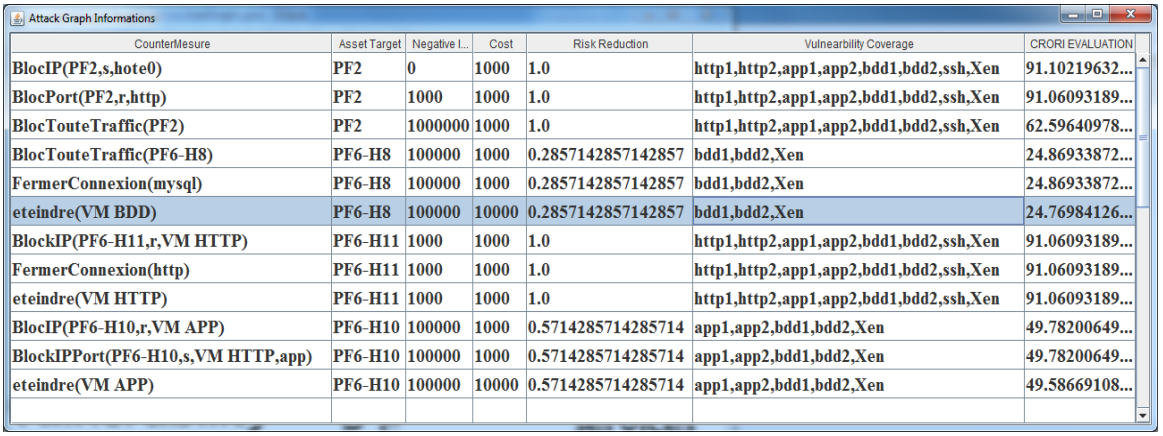
Path	Risk Value
[<hote11,hote11,0.12>;C, <http2,hote11,VM HTTP hote11,0.35>;E, <root,hote11>;C, <app1,hote11,VM APP h...	6.179711346705983E-7
[<hote11,hote11,0.12>;C, <http2,hote11,VM HTTP hote11,0.35>;E, <root,hote11>;C, <app1,hote11,VM APP h...	8.136867116316699E-7
[<http,hote11,0.10>;C, <http2,hote11,VM HTTP hote11,0.35>;E, <root,hote11>;C, <app1,hote11,VM APP hot...	5.149759455588318E-7
[<http,hote11,0.10>;C, <http2,hote11,VM HTTP hote11,0.35>;E, <root,hote11>;C, <app1,hote11,VM APP hot...	6.780722596930582E-7
[<UL,hote0,0.85>;C, <http1,hote0,VM HTTP hote11,0.61>;E, <UL,hote11>;C, <http2,hote11,VM HTTP hote1...	6.227082717160411E-7
[<UL,hote0,0.85>;C, <http1,hote0,VM HTTP hote11,0.61>;E, <UL,hote11>;C, <http2,hote11,VM HTTP hote1...	8.199241315511452E-7
[<http,hote11,0.10>;C, <http1,hote0,VM HTTP hote11,0.61>;E, <UL,hote11>;C, <http2,hote11,VM HTTP hot...	7.325979667247543E-8
[<http,hote11,0.10>;C, <http1,hote0,VM HTTP hote11,0.61>;E, <UL,hote11>;C, <http2,hote11,VM HTTP hot...	9.646166253542885E-8
[<hote0,hote11,0.75>;C, <http1,hote0,VM HTTP hote11,0.61>;E, <UL,hote11>;C, <http2,hote11,VM HTTP ho...	5.494484750435658E-7
[<hote0,hote11,0.75>;C, <http1,hote0,VM HTTP hote11,0.61>;E, <UL,hote11>;C, <http2,hote11,VM HTTP ho...	7.234624690157161E-7

Figure 4.8 : Visualisation des données sur les risques des chemins probables


Asset Graph Informations

host name	confidentiality	integrity	aviability	initial value	updated value	infrastructure value
hote8	5	3	4	4.0	4.0	1000000
VM BDD hote8	4	3	3	3.3333333333333335	4.0	1000000
VM SSH hote8	2	3	4	3.0	3.0	10000
hote9	2	3	3	2.6666666666666665	4.0	11000
VM SSH hote9	5	2	4	3.6666666666666665	3.6666666666666665	10000
VM FTP hote9	4	5	3	4.0	4.0	10000
hote10	4	5	5	4.666666666666667	4.666666666666667	13000
VM SSH hote10	5	2	4	3.6666666666666665	3.6666666666666665	10000
VM APP hote10	5	4	5	4.666666666666667	4.666666666666667	100000
hote11	5	4	4	4.333333333333333	4.333333333333333	10000
VM HTTP hote11	3	3	5	3.6666666666666665	3.6666666666666665	10000
VM HTTPS hote11	4	4	5	4.333333333333333	4.333333333333333	10000
VM SSH hote11	5	3	4	4.0	4.0	10000
Xen	5	3	4	4.0	4.0	10000

Figure 4.9 : Visualisation des données sur les hôtes dans le graphe de composants


Attack Graph Informations

CounterMesure	Asset Target	Negative I...	Cost	Risk Reduction	Vulnearbility Coverage	CRORI EVALUATION
BlocIP(PF2,s,hote0)	PF2	0	1000	1.0	http1,http2,app1,app2,bdd1,bdd2,ssh,Xen	91.10219632...
BlocPort(PF2,r,http)	PF2	1000	1000	1.0	http1,http2,app1,app2,bdd1,bdd2,ssh,Xen	91.06093189...
BlocTouteTraffic(PF2)	PF2	1000000	1000	1.0	http1,http2,app1,app2,bdd1,bdd2,ssh,Xen	62.59640978...
BlocTouteTraffic(PF6-H8)	PF6-H8	100000	1000	0.2857142857142857	bdd1,bdd2,Xen	24.86933872...
FermerConnexion(mysql)	PF6-H8	100000	1000	0.2857142857142857	bdd1,bdd2,Xen	24.86933872...
eteindre(VM BDD)	PF6-H8	100000	10000	0.2857142857142857	bdd1,bdd2,Xen	24.76984126...
BlockIP(PF6-H11,r,VM HTTP)	PF6-H11	1000	1000	1.0	http1,http2,app1,app2,bdd1,bdd2,ssh,Xen	91.06093189...
FermerConnexion(http)	PF6-H11	1000	1000	1.0	http1,http2,app1,app2,bdd1,bdd2,ssh,Xen	91.06093189...
eteindre(VM HTTP)	PF6-H11	1000	1000	1.0	http1,http2,app1,app2,bdd1,bdd2,ssh,Xen	91.06093189...
BlocIP(PF6-H10,r,VM APP)	PF6-H10	100000	1000	0.5714285714285714	app1,app2,bdd1,bdd2,Xen	49.78200649...
BlockIPPort(PF6-H10,s,VM HTTP,app)	PF6-H10	100000	1000	0.5714285714285714	app1,app2,bdd1,bdd2,Xen	49.78200649...
eteindre(VM APP)	PF6-H10	100000	10000	0.5714285714285714	app1,app2,bdd1,bdd2,Xen	49.58669108...

Figure 4.10 : Visualisation des données sur les contremesures

4.4.5 Le chemin le plus dangereux et la contremesure optimale

Pour afficher des informations sur la contremesure optimale et le chemin le plus dangereux, aller vers le menu « Evalute » et cliquer sur « The most danger path and Optimal countermeasure ». La figure 4.11 montre la décision finale sur la contremesure optimale et le chemin le plus dangereux.

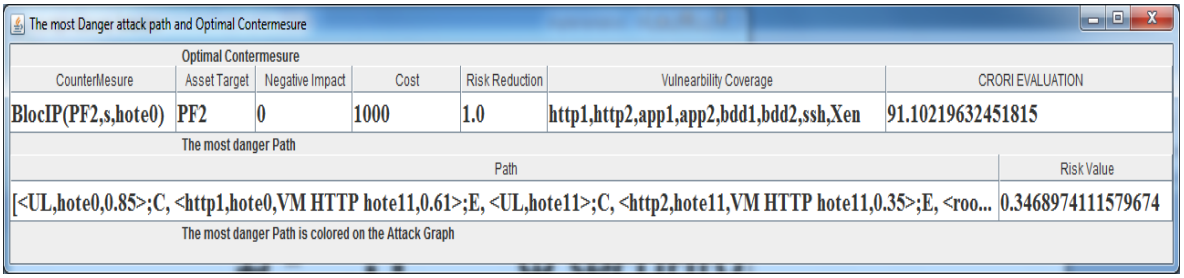


Figure 4.11 : Visualisation des données sur la contremesure optimale et le chemin le plus dangereux

4.4.6 Visualisation du chemin le plus dangereux

Il faut retourner à l'affichage du graphe d'attaque après l'évaluation pour voir le chemin le plus dangereux. Les nœuds présentés en jaune et l'arc en rouge. La figure 4.12 montre le chemin le plus dangereux pour l'évaluation précédente.

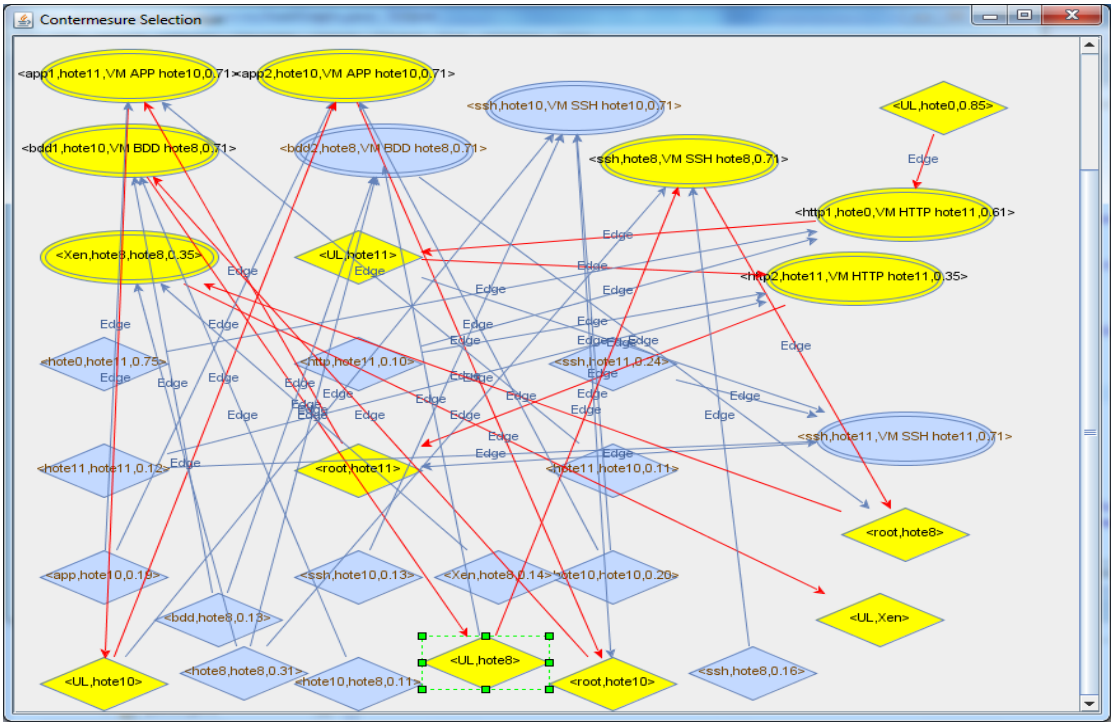


Figure 4.12 : Visualisation du chemin le plus dangereux

4.5 Conclusion

Dans ce chapitre, on a présenté la réalisation de notre travail en utilisant l'environnement Eclipse et plusieurs API Java. Notre application permet à l'administrateur réseaux de trouver la contremesure optimale pour l'attaque détectée. L'utilisateur capable de voir plusieurs données concernant les graphes utilisés. Il permet aussi de visualiser les résultats d'évaluations afin de comprendre bien la décision finale.

Conclusion Générale

À cause du nombre d'attaques, le nombre des alertes généré par l'IDS augmente, donc les systèmes d'information et principalement le Cloud computing ont obligé d'implémenter des stratégies de sécurité efficaces, pour assurer la confidentialité, l'intégrité et la disponibilité de leurs données.

En conséquence la détection d'intrusion sera inutile, sans le besoin d'un système de réponse aux intrusions(IRS) qui garantira la bonne santé de notre système.

Après l'analyse de quelques travaux de recherche, nous avons conclure que la sélection des réponses aux intrusions nécessite un traitement précis, en prend en considération la modélisation d'attaque à travers le graphe d'attaque, afin d'identifier les chemins empruntés par un attaquant dans l'exploitation d'une série de vulnérabilités, qui permet de sélectionner des meilleures réponses.

Dans notre approche, nous avons évalué l'attaque en estimant ses risques associés, pour extraire le chemin le plus probable. A l'aide de ce dernier, nous avons choisir les contremesures qui peuvent arrêter la propagation d'attaque, ensuite nous avons les évalué avec notre métrique $CRORI$ et le graphe de dépendance des contremesures, afin d'extraire la plus optimale et qui respecte plusieurs critères.

Enfin nous implémentons notre proposition par une application de Java, qui nous permet de visualiser les graphes (Graphe d'attaque, Graphe de dépendance des composants, et Graphe de dépendance des contremesures) avec ses évaluations, en exportant les changements pour une future évaluation.

Notre approche est améliorable par la sélection et le déploiement du plusieurs contremesures optimales avec l'apprentissage en profondeur des paramètres associés à chaque contremesure.

Références

[1] A. Curtis and J. Carver,"Adaptive agent-based intrusion response," Ph.D. thesis, Texas A&M University, USA, 2001.

[2] A Guide for Government Agencies Calculating Return on Security Investment, [En ligne] .Disponible sur< http://lockstep.com.au/library/return_on_investment, (2004) > (Consulté le 4/06/2019).

[3] Alma Whitten 2014 : Cloud Computing Définitions et notions de base ;Alma Whitten Directrice de la confidentialité chez Google démissionne [En ligne].Disponible sur <: http://communication.sysdis.fr/tag/cloud/> (Consulté le 01 04 2014).

[4] B. Foo, Y. S. Wu, Y. C. Mao, S. Bagchi, and E. Spafford, "ADEPTS: adaptive intrusion response using attack graphs in an e-commerce environment," International Conference on Dependable Systems and Networks, pp. 508-517, 2005.

[5] C. P. Mu, X. J. Li, H. K. Huang and S. F. Tian, "Online risk assessment of intrusion scenarios using D-S evidence theory," 13th European Symposium on Research in Computer Security, pp. 35-48, Malaga, Spain, 2008.

[6] C. Strasburg, N. Stakhanova, S. Basu, and J. S. Wong, "A Framework for Cost Sensitive Assessment of Intrusion Response Selection," Proceedings of IEEE Computer Software and Applications Conference, pp. 355-360, 2009.

[7] C. J.Chung , P.Khatkar, T. Xing, J. Lee, & D. Huang, (2013). NICE: Network intrusion detection and countermeasure selection in virtual network systems. IEEE transactions on dependable and secure computing, 10(4), 198-211.

[8] CVSS [En ligne].Disponible sur https://www.first.org/cvss. (Consulté le 1/06/2019).

[9] D. J. Brown, B. Suckow, and T. Wang. A Survey of Intrusion Detection

Systems. Department of Computer Science, University of California,San Diego.

[10] Développons en Java [en ligne].Disponible sur < https://www.jmdoudoux.fr/java/dej/chap-jdom.htm > (Consulté le 11/6/2019).

[11] Guide d'administration système: interfaces réseau et virtualisation de réseau [En ligne]. Disponible sur <https://docs.oracle.com/cd/E19120-01/open.solaris/819-6990/gfkbw/index.html >. (Consulté le 30/05/2019).

[12] G.Gonzalez-Granadillo, E.Doynikova, and I.Kotenko, & Garcia-Alfaro, J. (2017, October). Attack graph-based countermeasure selection using a stateful return on investment metric. In International Symposium on Foundations and Practice of Security (pp. 293-302). Springer, Cham.

[13]G. Gonzalez-Granadillo, M. Belhaouane, H. Debar, and G. Jacob. RORI-based countermeasure selection using the OrBAC formalism. International journal of information security, 13(1):63{79, 2014}.

[14] G. Gonzalez-Granadillo, J. Garcia-Alfaro, E. Alvarez, M. El-Barbori, and H. Debar.Selecting optimal countermeasures for attacks against critical systems using the attack volume model and the RORI index. Computers & Electrical Engineering, 47:13{34, 2015.

[15] G.Granadillo , H. Débar, G.Jacob , C.Gaber,, & M. Achemlal. (2012, October). Individual countermeasure selection based on the return on response investment index. In International Conference on Mathematical Methods, Models, and Architectures for Computer Network Security (pp. 156-170). Springer, Berlin, Heidelberg.

[16] Gauthier Picard : Initiation à la programmation orientée-objet [en ligne].Ecole National Supérieure des Mines, 2013-2014,69p. Disponible sur< https://www.emse.fr/~picard/cours/1A/java/livretJava.pdf> (Consulté le 10/06/2019).

[17] H. Saouli. Découverte de services web via le Cloud computing à base d'agents mobiles. Thèse de doctorat. Université Mohamed Khider de Biskra, 2015.

[18] Jean Patrick Gelas : " introduction à la virtualisation ".Lyon1.Disponible sur <http://perso.univ-lyon1.fr/fabien.rico/site/_media/cloud:2016:virtualisation-intro.pdf> (Consulté le 12/04/2019).

[19] J. Rutkowska, "Subverting VistaTM Kernel for Fun and Profit," Black Hat Conference, 2006.

[20] J-F. Pépin, S. Bouteiller, A-S. Boissard, J. Watrinel, Fondamentaux du Cloud computing- le point de vue des grandes entreprises. Réseau de Grandes Entreprises (CIGREF). Mars 2013. http ://www.eurocloud.fr/doc/cigref2.pdf

[21] K. A. B. A. V. Dastjerdi, and S. G. H. Tabatabaei, "Distributed intrusion detection in clouds using mobile agents," in Third International Conference on Advanced Engineering Computing and Applications in Sciences, 2009. ADVCOMP '09, 2009, pp. 175 – 180.

[22] L. H. Aslanyan, D. Alipour, and M. Heidari. Comparative analysis of attack graphs. In *Mathematical Problems of Computer Science*, volume 40, pages 85– 93, 2013.

[23] L. M. Ibrahim, "Anomaly Network Intrusion Detection System Based On Distributed Time-Delay Neural Network," Journal of Engineering Science and Technology, vol. 5, no. 4, pp. 457 – 471, 2010.

[24] M. Slaviero, "BlackHat presentation demo vids: Amazon." [En ligne].Disponible sur< http://www.sensepost.com/blog/3797.html/ > (Consulté le 27/02/2019).

[25] M. Papadaki and S. M. Furnell, "Achieving automated intrusion response: a prototype implementation," Information Management and Computer Security, vol. 14, no. 3, 2006, pp. 235-251

[26] Machine Virtuelle (VM) [En ligne] https://www.lemagit.fr/definition/Machine-Virtuelle-VM / [Accès le 12/03/2019].

[27] M. Papadaki and S. M. Furnell, "Achieving automated intrusion response: a prototype implementation," Information Management and Computer Security, vol. 14, no. 3, 2006, pp. 235-251.

[28] N.Alhebaishi, L.Wang, S.Jajodia, & A.Singhal.(2016, October). Threat modeling for cloud data center infrastructures. In International Symposium on Foundations and Practice of Security (pp. 302-319). Springer, Cham.

[29] N. Kheir, N.Cuppens-Boulahia , F.Cuppens, H.Debar: A Service Dependency

Model for Cost-Sensitive Intrusion Response, In Proceedings of the 15th European

Symposium on Research in Computer Security (ESORICS), pp. 626{642 (2010).

[30] N. Kheir, N.Cuppens-Boulahia, F.Cuppens, and H.Debar:"A service dependency model for cost sensitive intrusion response," Proceedings of the 15th European Conference on Research in Computer Security, pp. 626-642, 2010.

[31] N. Stakhanova, S. Basu and J. Wong, "A cost-sensitive model for preemptive intrusion response systems," Proceedings of the 21st International Conference on Advanced Networking and Applications, IEEE Computer Society, Washington,DC, USA, pp. 428-435, 2007

[32] N. Stakhanova, S. Basu and J. Wong,"A cost-sensitive model for preemptive intrusion response systems," Proceedings of the 21st International Conference on Advanced Networking and Applications, IEEE Computer Society, Washington,DC, USA, pp. 428-435, 2007.

[33] National Vulnerability Data base [en ligne] .Disponible sur https://nvd.nist.gov/ (Consulté le 23/05/2019).

[34] P. Mell, T. Grance, The NIST Definition of Cloud Computing,Recommendation of NIST. Special Publication 800-145, 2011.http //csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf.

[35] Quelle définition pour le « Cloud Computing » [En ligne] http://blog.econocom.com/blog/quelle-definition-pour-le-cloud-computing/ [Accès le 25/02/2019].

[36] Rapport de Stage de fin d'Etudes Master 2 : «Scénarios d'Attaques et Détection d'Intrusions » par Julien Iguchi-Cartigny.

[37] S. King, P. Chen, and Y-M. Wang, "SubVirt: Implementing malware with virtual machines," 2006 IEEE Symposium on Security and Privacy, 2006, pp.314-327.

[38] S.Basu, A.Sengupta, & C. Mazumdar(2017). A Quantitative Methodology for Cloud Security Risk Assessment. CLOSER.

[39] S. Bahram, X. Jiang, Z. Wang, and M. Grace, "DKSM: Subverting Virtual Machine Introspection for Fun and Profit," Proceedings of the

29th IEEE International Symposium on Reliable Distributed Systems,2010.

[40] Stakhanova, N., Basu, S., & Wong, J. S. (2006). A Taxonomy of intrusion response systems.

[41] Shameli-Sendi, A., Ezzati-Jivan, N., Jabbarifar, M., & Dagenais, M. (2012). Intrusion response systems: survey and taxonomy. Int. J. Comput. Sci. Netw. Secur, 12(1), 1-14.

[42] Shameli-Sendi A, CherietM,Hamou-LhadjA.Taxonomy of intrusion risk assessment and response system.ComputSecur 2014;45:1–16.

[43] Shameli-Sendi, A., Louafi, H., He, W., & Cheriet, M. (2018). Dynamic optimal countermeasure selection for intrusion response system. IEEE Transactions on Dependable and Secure Computing, 15(5), 755-770.

[44] T. Garfinkel, and M. Rosenblum, "A Virtual Machine Introspection Based Architecture for Intrusion Detection," In Proc. Network andDistributed Systems Security Symposium, pp. 191-206, 2003.

[45] TechTarget. Eclipse (Fondation Eclipse) [en ligne], Disponible sur : < https://searchmicroservices.techtarget.com/definition/Eclipse.(Consulté le 10/6/2019).

[46] Virtualisation des services [En ligne] https://whatis.techtarget.com/fr/definition/Virtualisation-des-services / [Accès le 12/03/2019].

[47] W. Kanoun, N. Cuppens-Boulahia, F. Cuppens, and S. Dubus, "Risk-Aware Framework for Activating and Deactivating Policy-Based Response," Proceedings of the Fourth International Conference on Network and System Security, pp. 207-215, 2010**.**

[48] X. Ou, W. F. Boyer, and M. A. McQueen, "A scalable approach to attack graph generation," Proc. of the 13th ACM conf. on Computer and communications security (CCS '06), pp. 336–345. 2006.

[49] Y. Guan, and J. Bao, "A CP Intrusion Detection Strategy on Cloud Computing," In International Symposium on Web Information Systems and Applications (WISA), pp. 84–87, 2009.

[50] Y. Chen, B. Boehm, and L. Sheppard, ”Value Driven Security Threat Modeling Based on Attack Path Analysis,” In 40th Hawaii International Conference on System Sciences, Big Island, Hawaii, January 2007.

[51] Y. Zhang, X. Fan, Y. Wang, and Z. Xue, ”Attack grammar: A new approach to modeling and analyzing network attack sequences,” Proceedings of the Annual Computer Security Applications Conference (ACSAC 2008), pp. 215-224, 2008.

[52] Z., Inayat, A. Gani, A., N. B Anuar, M. K .Khan, & S.Anwar. (2016). Intrusion response systems: Foundations, design, and challenges. Journal of Network and Computer Applications, 62, 53-74.

.