Sécurité de la gestion dynamique des ressources basée sur la prise en compte des profils de consommation en ressources des machines virtuelles, dans un cloud IaaS

Kahina Lazri

Université Paris 13
U.F.R. de Sciences
École doctorale Galilée

Thèse

Présentée par

Kahina Lazri

en vue de l'obtention du grade de

DOCTEUR DE L'UNIVERSITÉ PARIS 13

Discipline : Informatique

Titre : Sécurité de la gestion dynamique des ressources basée sur la prise en compte des profils de consommation en ressources des machines virtuelles, dans un cloud IaaS

MEMBRES du JURY

Hervé Debar	Professeur des universités	Télécom SudParis	Rapporteur
Noël De Palma	Professeur des universités	Université Grenoble 1	Rapporteur
Jalel Ben-Othman	Professeur des universités	Université Paris 13	Directeur de thèse
Sylvie Laniepce	Expert Recherche	Orange Labs	Encadrante de thèse
Christophe Cérin	Professeur des universités	Université Paris 13	Examinateur
Thierry Coupaye	Directeur de Recherche cloud	Orange Labs	Examinateur

A ma mère, mon père

Remerciements

Je souhaite tout d'abord remercier Sylvie Laniepce de m'avoir encadrée pendant ces trois années de thèse. Nos discussions, parfois très animées ont toujours été intéressantes et passionnées. Je la remercie d'avoir été exigeante, parfois difficile à convaincre mais toujours à l'écoute. Merci Sylvie d'avoir fait de cette expérience de thèse une aventure professionnelle et humaine inoubliable.

Merci ensuite à Jalel Ben-othman, d'avoir accepté la direction de cette thèse et d'avoir réussi à encadrer ces travaux malgré la distance géographique imposée par les contraintes d'une thèse CIFRE. Merci à lui pour son suivi, ses conseils et surtout sa confiance qui m'a beaucoup aidée à croire en moi.

Je remercie Mohammed Kassi-Lahlou, pour ses conseils et son attention constamment manifestée concernant l'avancement des travaux de cette thèse. Merci Kassi d'avoir été à l'écoute et de bon conseil tout au long de cette thèse.

Je ne peux que remercier Haiming Zheng que j'ai co-encadré pendant son stage et qui a participé aux développements réalisés dans cette thèse. Thank you Haiming for your strong involvement in this work.

Merci aux membres de l'équipe Ecocenter : Laurent Di Pietro, Bassam Ben Hamouda et Rachid Saharaoui, pour leur précieuse collaboration et assistance pour la collecte de traces de consommation de cloud réel qui ont servi à la validation des travaux de cette thèse.

Je remercie l'ensemble des membres permanents de l'équipe NPS d'Orange Labs Caen (Network and Product Security) au sein de laquelle j'ai effectué cette thèse, pour leur bienveillance, soutien et support aux doctorants. Je me suis toujours sentie protégée au sein de cette équipe.

Je remercie les doctorants et post-doctorants de l'équipe NPS : Chrystel, Hachem, Romain, Mohammed, Marie, Olivier, Ghada, Muhannad. Merci pour les nombreuses discussions interminables que nous avons partagées, les pauses cafés, nos fous rires et parfois nos peines partagées durant nos thèses.

Enfin, mes derniers remerciements vont à toute ma famille. Merci à eux pour leur soutien indéfectible, si important dans les moments difficiles.

Résumé

La virtualisation matérielle telle que mise en oeuvre dans le cloud computing, permet le partage de ressources matérielles entre plusieurs machines virtuelles pouvant appartenir à différents utilisateurs. Ce partage des ressources constitue l'atout majeur de ces infrastructures, qui permet aux fournisseurs d'exploiter plus efficacement les ressources des centres de données, notamment à travers l'allocation dynamique des ressources. Cependant, le partage des ressources introduit de nouvelles contraintes de sécurité. Plusieurs travaux de l'état de l'art ont démontré l'apparition de nouvelles stratégies d'attaques propres aux infrastructures cloud computing, exploitant le partage des ressources. Néanmoins, il a aussi été démontré qu'il est possible de tirer avantage de la position privilégiée de la couche de virtualisation pour offrir une meilleure sécurité que celle assurée dans les plate-formes traditionnelles d'hébergement en silo.

Cette thèse poursuit deux axes de recherche complémentaires. Le premier axe traite des nouvelles vulnérabilités liées aux infrastructures cloud computing. Nous avons démontré une attaque que nous appelons attaque par 'migrations intempestives de machines virtuelles', dans laquelle un attaquant parvient à amener le système de gestion dynamique de ressources à migrer de façon abusive des machines virtuelles, par simple manipulation des quantités de ressources consommées par des machines virtuelles qui sont sous son contrôle. Nous avons démontré cette attaque sur une plate-forme constituée de cinq serveurs et analysé les conditions nécessaires à son succès ainsi que l'exposition des clusters vis-à-vis de la vulnérabilité qu'elle exploite.

Le second axe propose de tirer avantage de la position privilégiée de l'opérateur qui dispose à la fois d'une vue multi-couches plus riche de l'utilisation des ressources et d'une vue plus globale des contextes d'exécution des machines virtuelles, comparativement à la vue limitée de l'utilisateur, pour offrir une meilleure sécurité. Nous avons proposé AMAD (Abusive VM Migration Attack Detection), un système de supervision, chargé de détecter l'occurrence des attaques par migrations intempestives de machines virtuelles et d'identifier de façon automatique celles à l'origine de l'attaque. AMAD est implémenté sur notre plate-forme d'expérimentation et évalué à l'aide de traces de consommation de machines virtuelles collectées sur des clouds réels. Les résultats d'évaluation montrent qu'AMAD opère avec une bonne précision de détection.

Abstract

Hardware virtualisation is the core technology which enables resource sharing among multiple virtual machines possibly belonging to different tenants within cloud infrastructures. Resource sharing is the main feature that enables cost effectiveness of cloud platforms, achieved through dynamic resource management. However, resource sharing brings several new security concerns. Several proofs of concepts have demontrated new attack strategies brought by the resource sharing paradigm, known as cross-virtual machine attacks. Even so, it is also showed that the priviligied position of the virtualisation layer can be leveraged to offer better security protection mecanisms than the ones offered in non virtualized platfoms.

This thesis follows two main objectives. The first one is related to the domain of cloud-specific vulnerabilities. We have demonstrated a new attack, called the abusive virtual machine migration attack, in which an attacker can leverage the sharing of resources, through the manipulation of the amounts of resources consumed by virtual machines under his control, to abusively enforce the dynamic resource management system to trigger virtual machine migrations. We have demonstrated this attack on a virtualized platform composed of five physical machines, the necessary conditions for the attack to succeed and the vulnerability exposure of clusters against this kind of attack is also analyzed.

The second main contribution of this thesis aims at leveraging the privilged position of the cloud provider who has both a more reliable view of the ressource utilisation and a more complete view of the virtual machine execution contexts compared to the limited view of cloud users, to provide better security. We propose AMAD (Abusive Virtual Machine Migration Attack Detection), a system designed for detecting an abusive use of the dynamic virtual machine migration, in the case of the abusive virtual machine migration attack. AMAD identifies the virtual machines possibly at the origin of the attack by analyzing their resource consumption profiles which show fluctuation and correlation in the usage of resources. We have implemented AMAD on top of our laboratory platform and evaluated it with the help of virtual machine resource consumption traces collected from real cloud. Our evaluation results show that AMAD identifes the attacking virtual machines with high detection accuracy.

Table des matières

Introduction générale

Le cloud computing constitue un nouveau paradigme informatique conçu pour offrir aux utilisateurs un mode d'utilisation des ressources informatiques, dans lequel ceux-ci peuvent externaliser à la demande leurs données et services au sein d'infrastructures tierces déployées et gérées par des fournisseurs cloud, avec un principe de payement à l'usage.

Le cloud computing est construit sur le principe du partage des ressources matérielles par des services appartenant à différents utilisateurs. Le partage des ressources est géré par une couche logicielle dite de virtualisation, qui agit comme une couche d'indirection entre les ressources matérielles de l'infrastructure et les ressources virtuelles présentées aux différents utilisateurs. Cette couche de virtualisation permet de partitionner les ressources de l'infrastructure et de les conteneuriser sous forme de machines virtuelles, qui sont autant d'environnements logiciels utilisateurs. L'élasticité dans l'allocation des ressources, qui est la capacité d'ajuster dynamiquement les quantités de ressources allouées aux machines virtuelles en fonction de leurs besoins variables dans le temps, est l'une des perspectives les plus attractives du cloud puisque c'est cette propriété en particulier qui permet d'offrir un payement à l'usage aux utilisateurs.

La convergence de toutes ces technologies au sein d'une même infrastructure soulève plusieurs nouvelles problématiques de sécurité. Le partage des ressources entre des machines virtuelles appartenant potentiellement à différents utilisateurs, représente un nouveau vecteur de vulnérabilités que des attaquants peuvent chercher à exploiter pour mettre en oeuvre de nouvelles stratégies d'attaques. Il a été démontré que dans ces infrastructures, un attaquant peut prendre le contrôle sur une machine virtuelle pour attaquer des machines virtuelles co-résidentes s'exécutant dans le même espace de ressources. Plusieurs travaux de l'état de l'art ont démontré la possibilité de mettre en oeuvre de telles attaques sur des plate-formes virtualisées en laboratoire, voire sur des plate-formes de cloud réels, qui ont pour effets possibles : le vol de ressources, le vol d'information, la dégradation de performances, etc...

Néanmoins, les environnements virtuels offrent de nouvelles voies pour la sécurité. La virtualisation peut permettre d'isoler davantage les mécanismes de sécurité des environnements à protéger. L'élasticité dans l'allocation des ressources offre des opportunités de rendre élastiques

les systèmes de supervision pour leur permettre de s'adapter à la charge du système à superviser. De plus, la co-résidence des machines virtuelles au sein d'une même machine physique peut procurer à la couche de virtualisation une vue riche sur les contextes d'exécution grâce à la vue multi-utilisateurs permettant d'effectuer des corrélations à plusieurs niveaux (entre les machines virtuelles, les hôtes, les différentes couches de la pile logicielle) pour identifier des attaques que la supervision des ressources d'un seul utilisateur ne pourra pas révéler.

Dans le cadre des travaux de cette thèse, nous cherchons à analyser de nouvelles stratégies d'attaques sur les plate-formes cloud en exploitant les systèmes de gestion dynamique des ressources. Dans l'état de l'art actuel, ces systèmes de gestion dynamique de ressources prennent leurs décisions en analysant les quantités de ressources consommées par les machines virtuelles. Or, ces quantités sont par nature aisément manipulables par les utilisateurs des machines virtuelles. Cette dépendance des décisions des systèmes de gestion dynamique des ressources aux quantités de ressources consommées par les machines virtuelles associée à l'inter-dépendance des exécutions des machines virtuelles, qui résulte précisément du partage des ressources, peut constituer une vulnérabilité pour ces systèmes.

Au delà du cas d'attaque particulier que nous démontrons et analysons dans cette thèse, l'objectif de la réflexion menée est de montrer que la sécurité des plate-formes cloud ne se limite pas aux considérations liées aux attaques connues jusqu'à présent dans les systèmes conventionnels et doit intégrer la gestion élastique des ressources. Il convient de considérer les spécificités du cloud, en particulier le partage des ressources et l'élasticité dans l'allocation des ressources en prenant en compte la sécurité des mécanismes qui assurent ces propriétés d'élasticité dès leur conception.

Dans les travaux de cette thèse, nos contributions principales sont les suivantes :

— Nous avons identifié comment exploiter l'algorithme de gestion dynamique des ressources VMware, i.e. Distributed Resource Scheduler (DRS), basée sur la migration de machines virtuelles pour mener une attaque contre l'infrastructure. Nous avons démontré cette attaque sur une plate-forme virtualisée en agissant par simple manipulation des quantités de ressources consommées par des machines virtuelles sous le contrôle d'attaquants. Nous avons ainsi pu démontrer qu'un attaquant peut manipuler les quantités de ressources consommées par des machines virtuelles sous son contrôle pour amener le système de gestion dynamique des ressources à migrer de manière abusive des machines virtuelles. Les contextes d'exécution de l'attaque ont été analysés et le niveau d'exposition des clusters vis-à-vis de cette vulnérabilité a également été mesuré,

— Nous avons proposé le système AMAD (de l'anglais, Abusive VM Migration Detection) : un système de supervision des décisions de migration des systèmes de gestion dynamique

des ressources, qui adopte une approche de détection réactive pour la détection de l'attaque par migrations intempestives de machines virtuelles d'une part, et une approche d'analyse en boite noire pour la modélisation des profils de consommation en ressources des machines virtuelles dans l'objectif d'identifier celles à l'origine du nombre anormal de migrations exécuté par le système de gestion dynamique des ressources d'autre part,

— La dernière contribution de la thèse est une analyse de traces de consommation en ressources de machines virtuelles s'exécutant sur cloud privé d'Orange permettant d'étudier le critère de fluctuation de la consommation en ressources.

Le mémoire est organisé de la manière suivante :

Ce mémoire est organisé en cinq parties. Le chapitre 1 présente les concepts fondamentaux des plate-formes cloud et introduit les deux axes principaux dans lesquels s'inscrivent les travaux de cette thèse. La premier axe est lié aux nouvelles vulnérabilités spécifiques aux plate-formes cloud. Le second axe présente une introduction aux systèmes de détection d'intrusions, en particulier aux approches basées sur la couche de virtualisation. Le chapitre 2 livre d'abord une analyse des systèmes de gestion dynamique des ressources de l'état de l'art, puis introduit notre première contribution principale : l'attaque par migrations intempestives de machines virtuelles. Les quantités minimum de ressources nécessaires à l'exécution de l'attaque sont également mesurées et analysées dans ce chapitre. Le chapitre 3 présente la deuxième contribution principale de la thèse autour du système AMAD de supervision des décisions de migration d'un système de gestion dynamique de ressources, en détaillant sa conception, son implémentation et son évaluation. Le chapitre 4 expose les résultats d'une analyse de traces de consommation en ressources colletées sur cloud privé d'Orange, qui constitue la troisième contribution principale de cette thèse. Enfin, la dernière partie livre les conclusions de ces travaux et en présente les perspectives.

Chapitre 1

Sécurité du cloud et de la virtualisation

Le partage de ressources entre plusieurs machines virtuelles permet l'optimisation de l'exploitation des ressources dans les plate-formes cloud. Assurer une 'isolation forte' entre les machines virtuelles qui s'exécutent sur une même machine physique demeure considérée comme la propriété principale à garantir pour une exécution sûre de ces VMs. Ce chapitre introduit les concepts fondamentaux liés aux technologies fondatrices des plate-formes cloud (virtualisation, élasticité) et analyse les nouvelles stratégies d'attaques rendues possibles par le partage de ressources qui caractérise ces plate-formes. L'analyse présentée nous amène à discuter des impacts de ce type d'attaque sur la gestion de la sécurité dans le cloud. En dernière partie, ce chapitre livre une introduction aux systèmes de détection d'intrusions conventionnels d'une part, et aux nouveaux systèmes de détection qui exploitent la technologie de virtualisation pour tirer avantage de la position privilégiée de l'hyperviseur pour implémenter une détection plus fiable que celle implémentée au niveau utilisateur d'autre part.

Sommaire

1.1 Introduction

La sécurité du cloud computing est considérée comme un des freins majeurs à l'adoption massive de ce modèle de service par les utilisateurs [1] ; l'adoption du cloud signifie pour un utilisateur d'externaliser ses données et traitements vers une infrastructure opérée par une entité tierce. La multiplicité des technologies et des acteurs (utilisateur, opérateur, fournisseur de l'infrastructure, entité tierce) co-existant au sein d'un même environnement informatique, rend la gestion de la sécurité dans le cloud complexe et fragmentée. D'un point de vue opérateur cloud, les principes techniques de ces infrastructures introduisent de nouveaux défis de sécurité liés en particulier à la mutualisation des ressources qui offre au cloud la capacité d'adapter les quantités de ressources allouées aux VMs en fonction de la variation de la charge de celles-ci dans le temps, permettant ainsi une facturation à l'usage des ressources consommées.

Les deux principales caractéristiques techniques nouvelles du cloud sont la virtualisation des ressources matérielles et l'élasticité dans l'allocation des ressources. Ces propriétés introduisent de nouvelles vulnérabilités qui peuvent être exploitées par des attaquants cherchant à tirer avantage et à exploiter le partage des ressources entre les VMs pour mettre en oeuvre de nouveaux procédés d'attaques. L'*isolation forte* entre les VMs est l'une des propriétés principales à garantir pour empêcher une exploitation malveillante de la propriété de partage de ressources entre les VMs : non seulement une VM ne doit pas pouvoir accéder aux ressources d'une autre VM mais aussi, l'exécution d'une VM ne doit pas impacter celles des VMs co-résidentes sous la forme d'effets observables sur ces autres VMs (dégradations de performances ou disponibilité de services par exemple).

Les systèmes de détection d'intrusions et de supervision de la sécurité doivent également être adaptés pour gérer au mieux la sécurité dans ces nouveaux environnements. La sécurité des VMs dans le cloud est opérable par l'utilisateur propriétaire de la VM et/ou par le fournisseur de l'infrastructure. Chaque entité bénéficie de points avantageux mais se heurte aussi à des verrous à lever, ayant trait à la visibilité sur l'utilisation des ressources d'une part et à l'isolation vis-à-vis des attaques d'autre part. La couche de virtualisation opérée par l'opérateur présente un potentiel important pour ces systèmes, offrant des propriétés d'isolation, de passage à l'échelle, de visibilité fiable sur l'utilisation des ressources et d'une vue riche des contextes d'exécution des VMs.

Dans ce chapitre, nous présentons d'abord les concepts fondamentaux des technologies fondatrices des plate-formes cloud. Nous introduisons également les deux principaux domaines dans lesquels s'inscrivent les travaux de cette thèse, celui des nouvelles vulnérabilités introduites par les caractéristiques techniques spécifiques aux plate-formes cloud d'une part et celui de la supervision de la sécurité pour lequel nous discutons du potentiel et des limites qu'offre la virtualisation d'autre part.

Organisation du chapitre

Ce chapitre est organisé comme suit. La section 1.2 livre des considérations générales sur le cloud. La section 1.3 introduit les différentes approches de virtualisation matérielle et discute les avantages et les limites de chacune d'entre elles. La section 1.4 introduit les approches de la gestion élastique des ressources dans le cloud. La section 1.5 donne un panorama des attaques spécifiques aux plate-fomes cloud qui tirent avantage du partage des ressources entre plusieurs VMs pour produire des effets inter-VMs. La section 1.6 donne une introduction aux différentes approches de détection d'intrusions et discute du potentiel et des limites des nouveaux systèmes de détection qui tirent avantage de la virtualisation. Enfin, la section 1.7 livre une discussion sur les nouvelles stratégies d'attaques et les opportunités liées aux nouveaux mécanismes possibles de supervision de la sécurité dans le cloud.

1.2 Cloud Computing

Dans cette section, nous définissons plus en détails ce qu'est le cloud computing, ses caractéristiques et modèles de services.

1.2.1 Définition

Le cloud computing, littéralement 'informatique en nuage', représente un nouveau mode de consommation des ressources informatiques qui exploite plusieurs technologies existantes et d'autres émergentes. Le cloud computing vise à fournir des ressources informatiques à la demande sous la forme de services et à facturer aux utilisateurs uniquement les ressources réellement consommées. Du point de vue des fournisseurs des infrastructures d'hébergement, le cloud représente un potentiel d'optimisation de l'utilisation des serveurs qui composent les infrastructures grâce à la virtualisation matérielle et à la mutualisation des ressources. D'un point de vue des utilisateurs, le cloud offre un moyen d'externaliser la gestion des ressources informatiques en ayant la garantie d'avoir un accès permanent aux ressources et une facturation à l'usage.

La définition du cloud a fait l'objet de plusieurs propositions au sein de la communauté scientifique [2, 3], toutefois la définition établie par le National Institute of Standards and Technology (NIST) est aujourd'hui celle qui est communément adoptée [4]. Le NIST définit le cloud comme un modèle qui permet d'avoir un accès omniprésent, à la demande, à un pool de ressources partagées (réseau, serveurs, stockage, applications), qui peuvent être mises à disposition des utilisateurs rapidement, avec un minimum d'interaction avec le fournisseur de service cloud.

1.2.2 Caractéristiques

Le NIST définit cinq propriétés essentielles pour caractériser le modèle cloud [4].

1. **Un libre service à la demande :** l'utilisateur garde la possibilité de mettre en place et de configurer un service lorsqu'il le souhaite et ce, sans avoir recours à une intervention humaine d'un administrateur du côté du fournisseur cloud,
2. **Accès ubiquitaire au réseau d'accès au cloud :** le service cloud auquel souscrit l'utilisateur doit être accessible de manière permanente via le réseau,
3. **Mise en commun des ressources :** aussi qualifiée de mutualisation des ressources, elle consiste à agréger des ressources informatiques parfois hétérogènes en des pools de ressources mis à disposition des utilisateurs pour qui, cette agrégation est complètement transparente. La mutualisation des ressources est la propriété qui permet une attribution dynamique des ressources en fonction de la demande des utilisateurs et c'est aussi la propriété qui rend techniquement possible la rentabilité économique des offres cloud,
4. **Accès rapide et flexible :** ajustement des quantités de ressources allouées aux utilisateurs en fonction de leurs besoins en ressources, ces besoins pouvant être variables dans le temps,
5. **Facturation à l'usage :** le fournisseur met en oeuvre des mécanismes pour mesurer précisément les quantités de ressources consommées par les services auxquels souscrivent les utilisateurs.

1.2.3 Modèles de services

Le NIST propose une architecture en couches pour catégoriser les types de services cloud, qui se déclinent en trois offres possibles. Cette architecture est connue sous l'appellation SPI (software/ Plate-forme/ Infrastructure) as a Service [2].

— **Infrastructure (Infrastructure as a Service, IaaS) :** le service offert dans ce modèle est une quantité de ressources informatiques (CPU, mémoire, réseau, disque) pouvant prendre la forme de VMs, considérées comme la forme la plus élémentaire pour la mise à disposition de ressources aux utilisateurs. Dans ce modèle de services, l'utilisateur est responsable de l'installation et de la gestion de la totalité de la pile logicielle nécessaire à ses traitements, incluant le système d'exploitation. Les exemples d'offres IaaS les plus répandues sont Amazon Elastic Compute Cloud (EC2) [5], Windows Azure Cloud Services [6] et Google Compute Engine [7],

— **Plate-forme (Platform as a Service, PaaS) :** le service dans ce modèle est une plate-forme de développement équipée d'une plate-forme de programmation, des APIs et des

outils permettant de simplifier l'interaction avec d'autres éléments tels que des bases de données. Le PaaS permet de simplifier la conception et le déploiement d'applications en minimisant l'interaction avec l'infrastructure. Des exemples d'offres PaaS sont Google App Engine [8] et IBM Smart Cloud Application Services [9],

— **Application (Software as a Service, SaaS) :** le service offert dans ce modèle est un logiciel. Le fournisseur reste propriétaire du logiciel c'est à dire qu'il n'y a pas de code binaire à installer par l'utilisateur. L'accès au service s'effectue via internet souvent à l'aide d'une simple interface web. Ce modèle d'abstraction permet aux utilisateurs de s'affranchir de la totalité de la couche applicative faisant abstraction des conditions d'exécution des logiciels qu'ils utilisent. Des exemples d'offres SaaS sont Google Drive [10] pour l'édition de documents, Microsoft Office 356 [11] et Gmail pour la gestion du courrier électronique [12].

1.2.4 Modèles de déploiement

Il existe trois modèles de déploiement du cloud :

— **Cloud privé :** les clouds privés correspondent à des clouds dans lesquels l'infrastructure est réservée à une seule organisation, généralement une entreprise ou une agence gouvernementale. Cette infrastructure peut être déployée au sein même de l'organisation. Elle peut être déployée par l'organisation elle-même ou par une tierce partie mais la gestion post-déploiement de l'infrastructure relève de la responsabilité de l'organisation. Une variante de cloud privé, nommée *cloud privé virtuel*, consiste en une zone d'infrastructure au sein d'un cloud public dont les ressources et les accès seront réservés à une seule organisation. L'avantage principal du cloud privé est la sécurité des données car ces dernières restent hébergées dans une infrastructure dédiée à la seule entreprise qui les détient,

— **Cloud communautaire :** l'infrastructure est partagée par une communauté, pouvant être une organisation, une association ou tout groupe d'individus ayant des intérêts communs,

— **Cloud public :** l'infrastructure est mise à disposition d'un large public (entreprise, individus, organisation gouvernementale), déployée et gérée par un fournisseur cloud. Les ressources virtuelles allouées à des clients différents, peuvent être fournies par un pool commun de ressources physiques,

— **Cloud hybride :** l'infrastructure est une composition (et non pas une agrégation) de deux ou plusieurs infrastructures ayant chacune son propre modèle de déploiement

(privé, communautaire ou public). Dans ce modèle, les ressources de plusieurs clouds sont interconnectées par des technologies permettant la portabilité des données. Le cloud hybride peut représenter un bon compromis pour les entreprises, leur permettant d'interconnecter les différents modèles du cloud en fonction des contraintes des services hébergés par ces infrastructures. Une entreprise peut par exemple héberger en propre un certain nombre de données ou d'applications sur des clouds privés pour des besoins de sécurité et externaliser le reste des traitements sur un cloud public. Une autre application intéressante est d'avoir recours au cloud public pour absorber des pics d'activités à des périodes particulières de l'activité de l'entreprise [13].

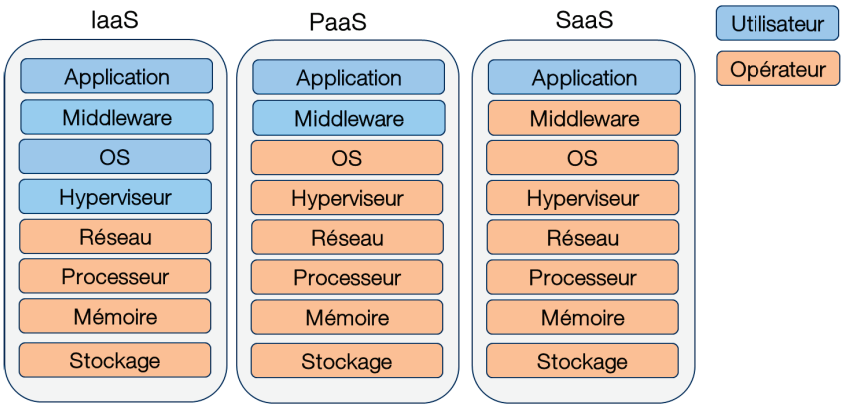


FIGURE 1.1 – Les modèles de services cloud

La figure 1.1 reprend les couches de services cloud auxquelles peut accéder un utilisateur selon les différents modèles de déploiement. On observe que le service IaaS représente la forme la plus basique du cloud dans laquelle le seul service auquel accède le client est l'infrastructure, alors que le SaaS représente la forme la plus aboutie du cloud dans laquelle le client est un simple consommateur de la ressource informatique. Dans le reste du mémoire, le mot cloud fera implicitement référence à un cloud IaaS.

Les caractéristiques clés du cloud, à savoir l'élasticité et la facturation à l'usage sont rendues possibles grâce à la virtualisation des ressources physiques. La section suivante livre des éléments sur les approches possibles de virtualisation matérielle.

1.3 Virtualisation matérielle

Dans cette section, sont présentés les principes et approches de virtualisation matérielle.

1.3.1 Définition

La virtualisation a été introduite dans les systèmes informatiques durant les années 70 avec les premiers modèles formels sur la virtualisation établis par *Golberg* et *Poepk* [14, 15]. Le succès de la virtualisation a été d'une courte durée du fait des faibles coûts des composants matériels ainsi que de l'émergence des systèmes d'exploitation multi-tâches. Cependant, durant les années 90, il a été constaté que les ressources de calcul (mémoire, processeur, réseau, disque) étaient fortement sous-utilisées, la virtualisation est alors considérée comme une technique qui permettrait d'optimiser l'exploitation des ressources matérielles notamment en permettant le partage d'un pool de ressources entre plusieurs systèmes hétérogènes [16].

La virtualisation est la technologie coeur des plate-formes cloud. Elle permet d'exécuter plusieurs composants logiciels en parallèle sur la même machine physique grâce à une couche logicielle appelée hyperviseur ou Moniteur de Machines Virtuelles (VMM, de l'anglais Virtual Machine Monitor).

L'hyperviseur peut être vu comme une fine couche logicielle chargée de créer une abstraction des ressources matérielles, produisant une forme virtuelle de ces ressources matérielles pour les présenter à une couche logicielle supérieure telle qu'un système d'exploitation [17, 18]. Lorsque cette forme virtuelle est différente de la forme matérielle des ressources, on parle alors d'émulation. L'hyperviseur s'interpose entre les VMs et les ressources matérielles (mémoire, processeur et les périphériques d'entrée sorties) et son rôle principal est d'exécuter, ordonnancer et contenir les VMs dans leur espace de ressources matérielles virtuelles, à la manière de la gestion des processus par un système d'exploitation.

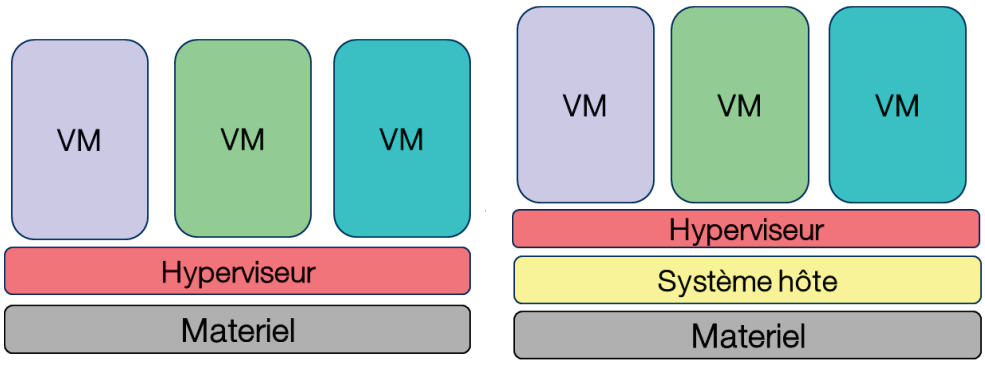
1.3.2 Types d'hyperviseurs

Dans le cadre de la virtualisation matérielle, il s'agit de virtualiser la totalité des ressources matérielles sous la forme de VMs pour permettre d'accueillir un système d'exploitation complet. Il existe deux principales approches de virtualisation matérielle, catégorisées selon que l'hyperviseur s'exécute directement sur la couche matérielle de la machine physique (type I) ou que celui-ci s'exécute au sein d'un système d'exploitation (type II).

— **Virtualisation de type I :** les hyperviseurs qui implémentent cette approche sont dénommés 'hyperviseurs bare-metal'. L'hyperviseur s'exécute directement sur les ressources matérielles et doit par conséquent implémenter la majorité des fonctionnalités que fournissent les noyaux des systèmes d'exploitation. Pour alléger ces fonctionnalités, certains hyperviseurs de type I déploient une VM privilégiée qui exécute un système d'exploitation pour bénéficier de ses fonctionnalités pour exécuter les pilotes des périphériques et à qui sont délégués tous les traitements des activités liées aux entrées/sorties [19]. *VMware*

ESX [20] et *Xen* [21] sont deux exemples d'hyperviseurs répandus, de type I,

— **Virtualisation de type II :** l'hyperviseur est réduit à un logiciel s'exécutant au sein d'un système d'exploitation appelé système hôte. Cette approche est antérieure à la virtualisation de type I et était justifiée par le fait que les architectures matérielles x86 ne permettent pas la virtualisation native des processeurs. L'hyperviseur dans cette approche peut être vu comme une application qui s'exécute sur un système d'exploitation bénéficiant de l'ensemble de ses fonctionnalités, la VM est alors considérée comme un simple processus. D'un point de vue de la sécurité, cette approche de virtualisation est considérée comme étant moins sécurisée que la virtualisation de type I du fait de la présence du système d'exploitation hôte, héritant ainsi de sa complexité et de ses vulnérabilités. Un exemple d'hyperviseur de type II est *VMware Workstation* [22].



(a) Type I (b) Type II

FIGURE 1.2 – Hyperviseurs de type I et de type II

Nous allons maintenant introduire les différentes techniques de virtualisation qu'implémentent les hyperviseurs.

1.3.3 Techniques de virtualisation

Les VMs utilisent pour leur exécution, la technique dite de *trap and emulate*, qui consiste à permettre à une VM de s'exécuter directement sur le processeur avec moins de privilèges que l'hyperviseur, et de faire en sorte que lorsqu'une VM tente d'accéder à des instructions ou des registres possédant un privilège plus élevé que celui qui leur est attribué, le CPU génère une Trap, renvoyant automatiquement le contrôle de l'exécution à l'hyperviseur qui exécutera l'instruction de la VM et lui retournera le résultat obtenu dans une nouvelle instruction. La Trap générée doit alors contenir une information qui identifie l'instruction ou le registre auquel la VM a tenté

d'accéder ; ce type de Trap est appelé Trap sémantique. Les architectures matérielles x86 ne permettent pas de générer des Trap sémantiques ; de ce fait, elles ont longtemps été considérées comme étant non virtualisables [23].

Il existe principalement trois différentes catégories de techniques de virtualisation utilisées pour la création de la couche d'abstraction des ressources matérielles.

1.3.3.1 Virtualisation complète

La virtualisation native, également qualifiée de virtualisation complète, permet à un système d'exploitation conçu pour une architecture matérielle particulière – la plus répandue étant l'architecture x86 – de s'exécuter au sein d'un environnement virtuel (VM) sans besoin de le modifier. En l'absence de support matériel à la virtualisation (jusqu'en 2005 avec l'introduction des extensions matérielles de support à la virtualisation, que nous détaillons plus bas), la technique la plus utilisée pour implémenter la virtualisation native est la technique de traduction binaire (BT, de l'anglais Binary Translation) [23], qui déploie un interpréteur responsable d'intercepter à la volée les instructions du système d'exploitation de la VM dans un traducteur, exécute les instructions de niveau hôte et renvoie le résultat obtenu à la VM [24].

Cette technique présente l'avantage de permettre une transparence élevée de la virtualisation vis-à-vis des systèmes d'exploitation des VMs, mais cette transparence se fait au détriment des performances puisque l'hyperviseur analyse l'ensemble des instructions exécutées par les VMs, générant ainsi des traitements supplémentaires systématiques. Tout comme les hyperviseurs *Workstation* et *ESX*, le projet libre QEMU [25] implémente cette technique.

1.3.3.2 Paravirtualisation

La paravirtualisation implémente une forme de collaboration entre les systèmes d'exploitation des VMs et l'hyperviseur. A la différence de la virtualisation native qui cherche à rendre l'exécution sur environnement virtuel complètement transparente aux VMs, la paravirtualisation nécessite la modification des noyaux des systèmes d'exploitation des VMs pour leur permettre d'interagir avec l'hyperviseur via des appels systèmes, tel que les *hypercalls* dans le cas de l'hyperviseur Xen. L'hyperviseur conserve le contrôle et la maîtrise de l'ordonnancement des accès des VMs aux ressources. Les hyperviseurs qui implémentent la paravirtualisation sont nécessairement de type I.

Bien que la technique de paravirtualisation présente une meilleure performance que la virtualisation native, du fait de l'exécution des instructions utilisateurs via des hypercalls, cette approche possède pour inconvénient majeur les limites de portabilité puisque la modification des noyaux des systèmes d'exploitation est requise et il est difficile, s'agissant des systèmes d'exploitation fermés dont le code source n'est pas disponible publiquement, d'assurer ces

modifications. L'hyperviseur le plus répandu implémentant la paravirtualisation est l'hyperviseur Xen [21].

1.3.3.3 Virtualisation assistée par le matériel

Les technologies dites de virtualisation assistée par le matériel fournissent un support pour faciliter la virtualisation mais ne se substituent pas à l'hyperviseur.

Afin de rendre les architectures x86 nativement virtualisables, Intel et AMD ont doté, en 2006, leurs processeurs respectivement des technologies VT/Vanderpool [26, 27] et Pacifica [28]. La virtualisation assistées par le processeur consiste en un ensemble d'opérations appelées VMX qui permettent de contrôler au niveau matériel les niveaux de privilèges accordés aux instructions des VMs, et ce de manière complètement transparente au système d'exploitation des VMs. L'activation de l'extension VMX déclenche la création de deux états : l'état *vmx-root*, correspondant à l'état d'exécution des instructions de niveau hyperviseur, procurant un contrôle complet sur le processeur, et l'état *vmx-non-root* correspondant à l'état d'exécution des instructions de niveau VM.

Avec cette approche, les systèmes d'exploitation continuent de s'exécuter comme sur les architectures non dotées d'extensions de virtualisation, hormis que certaines instructions provoquent un basculement automatique du processeur de l'état *vmx-non-root* vers l'état *vmx-root*, engendrant des transitions dites *vm-exits* qui redonnent le contrôle du processeur à l'hyperviseur.

Les instructions qui provoquent les transitions VMX sont contrôlées par une structure appelée structure de contrôle de machines virtuelles (VMCS, de l'anglais Virtual Machine Control Structure) qui sert à stocker les données d'état les plus importantes du système d'exploitation. Cette structure est lue à chaque activation de l'environnement virtuel et mise à jour à chaque sortie de celui-ci.

Les extensions de virtualisation simplifient considérablement le développement des hyperviseurs, participant à l'amélioration des performances d'exécution des environnements virtuels ainsi qu'à la réduction de leur surface d'attaque.

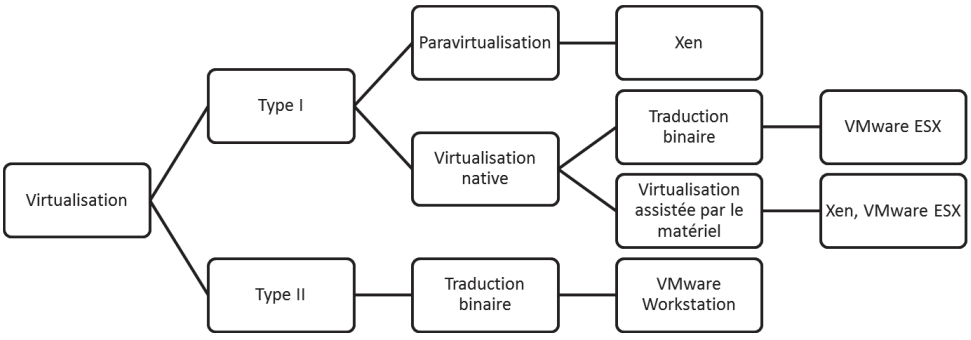


FIGURE 1.3 – Classification des approches de virtualisation matérielle

La figure 1.3 résume les différentes approches de virtualisation détaillées ci-dessus avec des exemples d'hyperviseurs pour chacune d'entre elle.

1.3.4 Propriétés des hyperviseurs

De par leur position privilégiée au sein de la pile logicielle d'un environnement virtuel, les hyperviseurs offrent trois propriétés principales [29] :

— **Encapsulation :** l'hyperviseur encapsule entièrement l'état des VMs, ce qui lui procure la capacité d'accéder à tout instant au contenu de leurs pages mémoire, registres CPU, connections réseaux, etc... L'encapsulation permet à l'hyperviseur de réagir en cas de faille matérielle, besoin de migration, instanciation d'une nouvelle VM, avortement de l'exécution d'une VM ou tentative de violation d'une politique de sécurité. Elle permet aussi de faciliter la mobilité logicielle, puisqu'un utilisateur peut facilement faire une copie d'une VM à travers le réseau,

— **Isolation :** l'hyperviseur assure l'isolation en garantissant qu'une VM ne puisse pas accéder ou modifier l'état d'une autre VM. Cette propriété est indispensable pour une exécution sécurisée dans un environnement mutualisé. Par ailleurs, la propriété d'isolation permet aussi d'assurer l'indépendance entre les cycles de vie des VMs, faisant en sorte qu'une faille ou tout dysfonctionnement dans une VM ne peut avoir d'impacts sur l'exécution d'une autre VM partageant le même pool de ressources matérielles,

— **Interposition :** le trafic échangé entre les VMs ainsi que l'ensemble des appels systèmes transitent par l'hyperviseur. Cela permet de garantir un contrôle sur l'ensemble des exécutions et des flux provenant ou à destination des différentes VMs. L'interposition au niveau de l'hyperviseur permet d'introduire plusieurs nouvelles fonctionnalités telles que l'inspection du trafic réseau, la gestion de la sécurité depuis une VM centralisée, ainsi que la flexibilité à travers une allocation dynamique et ajustable des ressources.

La section suivante donne une introduction à la notion d'élasticité dans l'allocation des ressources, rendue possible par la virtualisation, qui sera détaillée dans le chapitre 2.

1.4 Gestion des ressources dans le cloud

L'optimisation de l'utilisation des ressources est assurée par les systèmes de gestion de ressources dans le cloud, par consolidation des VMs sur les serveurs hôtes qui les hébergent,

grâce à la couche de virtualisation qui constitue une couche d'indirection permettant de mettre en oeuvre une élasticité dans l'allocation des ressources.

La figure 1.4 récapitule les étapes d'un processus de gestion de ressources. La première étape consiste à approvisionner les quantités de ressources configurées aux VMs ; l'étape de dimensionnement des quantités de ressources à allouer aux VMs est souvent réalisée par l'utilisateur de la VM lui-même. Le système de gestion dynamique des ressources est ensuite principalement responsable de deux opérations : la première consiste à déterminer le placement initial des VMs au sein d'un cluster d'hôtes et la seconde consiste à réviser de manière dynamique la répartition des VMs au sein de ce cluster de sorte à maximiser l'utilisation des ressources d'un centre de données tout en veillant à optimiser les performances d'exécution des VMs.

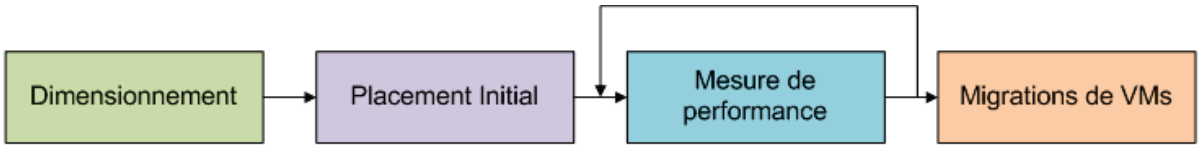


FIGURE 1.4 – Les étapes d'un processus de gestion dynamique de ressources

L'élasticité peut être définie comme étant la capacité d'un système à ajuster en relatif entre les différentes VMs hébergées sur un hôte, les quantités de ressources configurées pour chacune d'entre elles par réajustement automatique des quantités de ressources allouées aux VMs en fonction de leurs besoins variables dans le temps [30].

Afin d'assurer un partage équitable des ressources entre les VMs qui s'exécutent sur un même pool de ressources, les systèmes de gestion de ressources définissent pour chaque VM des valeurs plafonds de quantités de ressources qui permettent d'éviter qu'en situation de contention de ressources sur un hôte, une VM puisse bénéficier d'un traitement plus avantageux qu'une VM co-résidente de même niveau de priorité. Ces valeurs plafonds fixent deux seuils. Un premier seuil qui définit la borne inférieure (dénommée *réservation* dans les hyperviseurs, souvent exprimée comme une valeur absolue en MB ou en Mhz) représentant la quantité minimum de ressources que garantit le système de gestion de ressources à la VM, quel que soit son contexte d'exécution. Un deuxième seuil qui définit la borne supérieure (connue par la dénomination *limite*, aussi souvent exprimée comme une valeur absolue) représentant la quantité maximale de ressources que peut consommer la VM conformément aux quantités qui lui sont configurées.

La gestion élastique des ressources consiste en un réajustement automatique de la borne supérieure, prenant en compte le besoin réel et variable dans le temps en ressources de la VM. Une sous-estimation de cette valeur plafond est susceptible de provoquer des violations des engagements de qualité de services (SLAs, de l'anglais Service Level Agreement) passés entre l'opérateur de l'infrastructure cloud et ses clients si jamais la VM venait à ne pas disposer des ressources nécessaires pour atteindre le niveau de performances requis. Une surestimation de cette

valeur plafond n'est pas souhaitable non plus, puisqu'elle peut conduire à une sous-utilisation des ressources de l'infrastructure.

Pour le réajustement automatique de la valeur plafond, deux approches ont principalement été retenues. La première approche implémente une élasticité dite *'élasticité verticale'* (le terme anglais est le 'scaling-up'), qui consiste à réajuster les quantités de ressources allouées à une VM en la maintenant sur le même hôte. Cette première approche est très peu pratiquée dans les systèmes actuels, du fait des limites des systèmes d'exploitation en termes de possible réajustement dynamique de la mémoire allouée à une VM alors que le système d'exploitation est en cours d'exécution [30, 31]. La deuxième forme d'élasticité est une *'élasticité horizontale'* (le terme anglais est le 'scaling-out'), dans laquelle les quantités de ressources allouées aux VMs sont réajustées grâce à la duplication (ou suppression) de VMs sur différents hôtes permettant ainsi d'agréger de manière ajustable l'espace de ressources alloué à un utilisateur et qui repose sur un mécanisme d'équilibrage de charge pour distribuer la charge entre les instances de VMs.

Cette dernière approche est la plus utilisée dans les offres cloud actuelles. Elle reste néanmoins pénalisante pour les performances du fait de deux facteurs. Le premier est qu'elle tend à augmenter le nombre de VMs exécutant les mêmes applications ce qui augmente les quantités de ressources réservées au niveau de l'hyperviseur pour la gestion de ces VMs. Le second facteur est lié à la nécessité d'introduire une couche de gestion supplémentaire qui est celle de l'équilibrage de charge entre les instances de VMs.

Dans les offres cloud actuelles, l'utilisateur continue d'acheter des quantités statiques de ressources mises à disposition sous la forme de VMs à taille configurée statique et l'utilisateur révise ces quantités statiques en cas d'évolution du besoin. L'utilisateur reste responsable de définir les quantités de ressources nécessaires aux besoins de ses applications.

L'élasticité peut être aussi considérée du côté de l'utilisateur. Elle consiste dans ce cas, pour l'utilisateur, à mesurer les performances et besoins en ressources de ses VMs et à demander une allocation ou dé-allocation de ressources à l'opérateur si nécessaire. De nouvelles approches proposent des mécanismes de gestion de ressources coopératifs entre les utilisateurs et les fournisseurs cloud [32].

Dans le cadre de cette thèse nous nous intéressons à l'élasticité de la gestion des ressources offerte par l'infrastructure, sous le contrôle de l'opérateur. La forme actuelle de l'élasticité dynamique offerte par les infrastructures repose sur le sur-engagement des ressources. Le sur-engagement des ressources consiste à configurer toutes VMs confondues davantage de ressources que la capacité totale de l'hôte les exécutant, en escomptant que l'ensemble des VMs n'aient pas à consommer au même moment la totalité des ressources qui leur sont configurées, faute de quoi des contentions de ressources apparaissent. Si une situation de contention apparaît, l'hyperviseur active un certain nombre de mécanismes automatiques de gestion de la contention, tels que le ballooning (récupération de mémoire depuis une VM en sous-utilisation de ressources, pour

l'attribuer à une VM en contention) [20] ou la migration à chaud de VM, qui consiste en le transfert de l'état courant d'une VM d'un hôte source vers un hôte de destination pour libérer des ressources sur l'hôte en contention. C'est à la migration de VMs et à sa mise en oeuvre par les systèmes de gestion dynamique de ressources que nous nous intéressons dans cette thèse. Nous en détaillerons l'état de l'art dans le chapitre 2.

Dans les sections précédentes, nous avons abordé les concepts fondamentaux des plate-formes cloud computing et avons présenté les technologies de virtualisation et la gestion élastique de ressources dans le cloud. Nous allons maintenons nous intéresser à la sécurité du partage des ressources dans les plate-formes cloud virtualisées.

Dans la section suivante, nous présentons une classification des attaques démontrées dans l'état de l'art qui exploitent le partage des ressources, en fonction des effets de ces attaques sur les infrastructures et sur les VMs co-résidentes avec les VMs de l'attaquant.

1.5 Vulnérabilités liées aux plate-formes cloud

Nous livrons dans cette section un panorama d'attaques qui exploitent la propriété de partage des ressources matérielles entre plusieurs VMs pouvant appartenir à différents utilisateurs. Nous considérons les attaques qui mettent en cause la propriété d'isolation forte qui doit être garantie pour une exécution sécurisée des VMs sur une plate-forme virtualisée. Les différentes attaques que nous présentons dans cette section sont catégorisées en fonction des effets produits par l'attaque.

1.5.1 Attaque sur l'hyperviseur

Les hyperviseurs ont longtemps été considérés comme des composants logiciels sûrs du fait de leur faible quantité de lignes de codes rendant leur sécurité plus facile à vérifier que celle des systèmes d'exploitation conventionnels [33, 16].

Les hyperviseurs restent toutefois des cibles privilégiées pour les attaquants du fait du haut niveau de privilège dont ils disposent : l'attaquant qui parvient à prendre le contrôle sur l'hyperviseur parvient à prendre le contrôle sur les VMs qui s'exécutent sur cet hyperviseur. De plus, l'introduction de plus en plus de fonctionnalités au niveau des hyperviseurs (gestion de la sécurité par exemple), tend à augmenter leur complexité et à élargir leur surface d'attaque. Deux vecteurs d'attaques sont à distinguer : le premier concerne l'exploitation de failles logicielles au niveau même de l'implémentation des hyperviseurs (le plus souvent exploitées via les pilotes des périphériques) et le second vecteur concerne des failles de niveau matériel.

Plusieurs preuves de concepts ont été publiées pour mettre en évidence la vulnérabilité des hyperviseurs vis-à-vis de failles logicielles. Ormandy de Google [34], a réalisé un audit des

principales plate-formes de virtualisation incluant *VMware Workstation*, *VMware ESX*, *Xen* et *Qemu*, en utilisant un test à données aléatoires (connue sous la dénomination anglaise *fuzzing*). Ces outils génèrent une séquence aléatoire d'octets et d'activités d'entrées/sorties dans les VMs jusqu'à ce qu'une erreur ou un crash de l'hyperviseur se produise. Les résultats de cette étude montrent qu'aucune des versions d'hyperviseurs auditées n'a complètement réussi le test. De multiple failles ont pu être exploitées pour mener des attaques, notamment des attaques par débordement de pile ou encore une faille au niveau des pilotes qui permet à un attaquant qui contrôle une VM d'effectuer une escalade de privilège de son espace de ressources vers l'espace de ressources de l'hyperviseur. L'escalade de privilège est la forme la plus aboutie de mise en péril de l'isolation, grâce à laquelle un attaquant peut réussir à prendre le contrôle sur l'hyperviseur.

S'agissant des attaques exploitant des failles matérielles, l'une des premières attaques démontrées exploite la virtualisation assistée par le matériel et en particulier l'extension AMD *Pacifica*. Cette attaque est réalisée avec le *rootkit BluePill* (nommé BluePill en référence au film Matrix [35] dans lequel une pilule bleue permet de vivre dans un monde virtuel en ayant le sentiment de vivre dans un monde réel) qui permet de virtualiser à la volée un système d'exploitation d'une VM, créant ainsi une couche logicielle intermédiaire sous la forme d'un hyperviseur malicieux qui se positionne entre la VM et l'hyperviseur et qui peut intercepter ainsi la totalité des informations échangées entre la VM et l'hyperviseur. En 2008, l'attaque BluePill a été réalisée contre l'hyperviseur Xen qui utilise la virtualisation imbriquée (nested virtualisation) assistée par le matériel, pour prendre le contrôle sur l'hyperviseur depuis une VM. Dans cette attaque, une VM acquiert le privilège de niveau hyperviseur et peut faire basculer l'hyperviseur vers un niveau de privilège de niveau VM et prendre ainsi le contrôle sur les VMs de l'hôte.

Subvirt [36] est un autre rootkit de ce type, qui utilise la virtualisation complète. Subvirt est spécifique à VMware et contrairement à BluePill, il nécessite de modifier la séquence de démarrage du système d'exploitation de la VM et de procéder au redémarrage du système d'exploitation de la VM pour pouvoir en prendre le contrôle.

Dans [37], L. Duflot et al. montrent comment créer une porte dérobée sur une carte réseau grâce à une commande à distance d'un accès DMA (Direct Memory Access) pour lire et écrire dans la mémoire de la machine hôte.

Plusieurs exploits permettant la compromission d'hyperviseurs ont donc été démontrés sur la majorité des plate-formes de virtualisation [36, 38], y compris les plus répandues : CloudBurst [39] et Subvirt sur VMware, BluePill sur Xen [40]. Cela dit, il est important de noter que mener ces attaques requiert des conditions très sévères telles que disposer de privilèges de niveau super utilisateur (root) ou noyau au niveau VM, qui sont difficiles à obtenir pour une attaque à mener sur cloud réel.

1.5.2 Vol d'information

Le partage de ressources entre plusieurs VMs rend les plate-formes cloud vulnérables aux attaques inter-VMs par canaux cachés (covert channel attacks) et par canaux auxiliaires (side channel attacks). Ce type d'attaques permet à une VM malicieuse d'acquérir la capacité d'outrepasser les politiques de sécurité de l'hyperviseur pour accéder à une information qui lui est interdite par les canaux de communication habituels (réseau ou système). Ces attaques reposent sur l'existence d'une corrélation entre les traitements effectués par un programme et les quantités de ressources utilisées par ce programme et qui sont mesurables.

Dans les attaques par canaux auxiliaires, un processus mesure l'utilisation que fait un autre processus d'une ressource partagée, dans le but de déduire une information, sans que ce dernier ne soit conscient de cette mesure [41]. Une telle attaque est démontrée dans [42], dans laquelle les auteurs utilisent le bilan de la consommation énergétique d'un hôte pour déterminer la combinaison des VMs s'exécutant sur cet hôte. Cette attaque peut être utilisée pour vérifier la présence d'un service donné sur une machine, dans le but de l'attaquer par la suite.

Dans une attaque par canaux cachés, deux ou plusieurs processus s'exécutent en collaboration afin de communiquer à travers la manipulation d'une ressource partagée qu'ils sont en mesure de manipuler et de mesurer [43]. Il existe plusieurs types d'attaque par canaux cachés. Les plus répandues sont les attaques par analyse temporelle, dans lesquelles deux processus ou plus coordonnent leur exécution de façon intentionnelle en manipulant et en mesurant les temps et le séquencement de l'utilisation d'une ressource partagée, de sorte que chacun des deux processus impliqués dans l'attaque puisse interpréter la manipulation effectuée par l'autre processus.

Concernant l'attaque par canaux cachés avec analyse temporelle, la ressource la plus vulnérable à cette attaque est le processeur. Cela est dû à la facilité de manipulation de la charge CPU depuis les VMs et aussi du fait que les coeurs CPU sont souvent partagés par différentes VMs. Dans [44], les auteurs réalisent une fuite d'information grâce à la manipulation de la charge CPU sur un hyperviseur Xen. L'attaque requiert de prendre le contrôle sur deux VMs s'exécutant sur un même hôte et partageant un même coeur CPU. La première est celle appartenant à l'utilisateur cible de l'attaque, dans laquelle l'attaquant installe un virus de type logiciel espion par exemple, responsable de la lecture d'une information sensible telle qu'un numéro de carte de crédit. La seconde VM est sous le contrôle de l'attaquant et doit communiquer avec le logiciel espion installé dans la première VM qui devra manipuler ses temps d'accès au CPU de sorte à ce que la deuxième VM puisse mesurer et interpréter la manipulation et déduire l'information transmise par le logiciel espion. Cette attaque n'est pas détectable par un pare-feu ou par un système de détection d'intrusions chargé de la supervision des activités réseau, puisqu'il n'y a aucun échange d'information via le réseau. Les résultats de ces travaux montrent que les deux processus peuvent communiquer avec une vitesse de transmission de 0.49 bits par seconde avec 91% à 100% de précision dans un environnement avantageux pour l'attaque, dans lequel les VMs co-résidentes

génèrent très peu de bruit.

L'attaque exploitant les canaux cachés la plus connue dans le domaine de la sécurité du cloud est démontrée dans [45]. Dans cette attaque, T. Ristenpart et al. démontrent la possibilité de réaliser une attaque par canaux cachés sur le cloud EC2 d'Amazon, pour mettre en évidence une co-résidence de VMs. L'attaque comprend deux étapes. La première étape nécessite de placer une VM malicieuse sur le même hôte que la VM cible de l'attaque. Cela est obtenu en combinant une analyse de la distribution d'adresses IP en fonction des régions du cloud EC2, une mesure d'échanges réseau effectués entre la VM cible et une source externe d'émission de paquets (machine de l'attaquant) et enfin une mesure des durées d'échange entre la VM cible et celle co-résidente sous le contrôle de l'attaquant. La seconde étape consiste à vérifier le succès de la co-résidence via une attaque par canaux cachés exploitant le disque comme ressource partagée. Les auteurs ont également démontré une attaque par canaux cachés exploitant le processeur pour réaliser une fuite d'information entre deux VMs se partageant un même coeur CPU sur une plate-forme Xen.

Dans [46], les auteurs évaluent la portée des attaques par canaux cachés exploitant la ressource CPU, sur une plate-forme de laboratoire et sur la plate-forme EC2, en faisant varier les caractéristiques matérielles des VMs, les profils de consommation en charge CPU de ces VMs ainsi que les configurations de l'hyperviseur. Ils ont ainsi pu observer que, bien que sous certaines conditions ils réussissent à mener l'attaque avec des performances meilleures que celles obtenues dans [45], la quantité d'information qui peut être récupérée grâce à ce type d'attaque reste très limitée.

1.5.3 Vol de ressources

L'objectif des attaques par 'vol de ressources', auxquelles il est souvent fait référence par attaques par 'vol de services', est de bénéficier indûment de ressources aux dépens de l'opérateur ou des VMs co-résidentes alors possiblement privées de pouvoir utiliser la quantité totale de ressources à laquelle elles ont souscrit. Ce type de mise en péril de la propriété d'isolation entre les VMs peut avoir pour résultats des problèmes de facturation, des dégradations de performances cross-VMs, une famine de ressources pour les VMs victimes, voire un déni de service dans un scenario extrême de l'attaque.

Dans [47], les auteurs exploitent une vulnérabilité dans l'ordonnanceur des crédits CPU de l'hyperviseur Xen pour démontrer une attaque par vol de ressources CPU. Le principe de base de cette attaque consiste à exploiter la périodicité et donc le caractère prédictif de l'ordonnanceur des crédits CPU de l'hyperviseur Xen, dont le fonctionnement est le suivant : pour garantir un partage équitable des ressources entre les VMs, l'ordonnanceur de crédits CPU attribue et débite les crédits vCPU alloués à chaque VM en évaluant périodiquement, à une fréquence fixe, les durées CPU consommées [48]. De ce fait, une VM malicieuse peut fausser la comptabilisation

de ses ressources consommées qui a lieu à échéance prévisible, en interrompant son exécution – entrant dans une phase de sommeil – juste avant chaque point de comptabilisation de l'utilisation de ressources. Cela permet à la VM malicieuse de maintenir le crédit CPU qui lui est alloué non débité, ce qui lui procure une priorité plus élevée que celles des autres VMs pour l'accès au CPU. Ce niveau de priorité supérieur acquis par la VM attaquante peut conduire à l'interruption de l'exécution des autres VMs avec lesquelles elle partage le coeur CPU (ce mécanisme est en particulier implémenté dans le mode Boost Priority [49]).

Les auteurs ont démontré sur une plate-forme de laboratoire qu'une VM malicieuse peut de cette façon, consommer près de 98% du coeur CPU qu'elle partage avec des VMs co-résidentes (configurées avec la même priorité et le même niveau de partage que la VM attaquante). Ils ont également démontré cette attaque sur le cloud EC2 d'Amazon qui utilise des hyperviseurs Xen. Dans ce dernier contexte, une VM malicieuse parvient à consommer près de 85% d'un coeur CPU (correspondant à 40% de ressources supplémentaires à la quantité de CPU allouée à la VM), bien que sur cette plate-forme les VMs s'exécutent en mode *non conservatif* qui est un mode ne permettant pas de consommer davantage de ressources que la quantité allouée (y compris en cas de disponibilité de vCPU). Sur la plate-forme EC2, les auteurs ont constaté que l'attaque ne réussit pas à priver les VMs co-résidentes des ressources qui leur sont allouées, ce qui fait penser aux auteurs que l'hyperviseur déployé dans EC2 est corrigé contre des attaques inter-VMs par vol de ressources.

1.5.4 Dégradation de performances

L'isolation de performance consiste à garantir qu'une VM n'est pas pénalisée par la consommation intensive de ressources d'une VM co-résidente. Alors que les mécanismes de virtualisation des ressources mémoire et CPU permettent un partitionnement stricte de ces ressources, les implémentations des hyperviseurs peuvent être déficientes pour ce qui est du partage des accès entrées/sorties. D'un point de vue de la sécurité, ce manque d'isolation inter-VM dans le traitement des entrées/sorties peut poser deux problèmes : d'une part, une VM peut souffrir d'une attaque par déni de service qui vise une VM co-résidente, d'autre part une VM a la possibilité de générer un trafic réseau important au niveau de ses propres ressources (i.e. auto-flooding), afin d'amener les VMs co-résidentes à observer des dégradations de performances.

Afin de quantifier la dégradation de performances qu'observe une VM lorsqu'elle s'exécute sur des ressources partagées avec une VM ayant une activité intensive, Jeanna et al. [50] ont mené une série d'expérimentations sur plusieurs hyperviseurs, avec plusieurs profils de consommation en ressources de VMs. Les tests ont considéré les ressources mémoire, processeur, disque et réseau. Bien que les résultats ne soient pas alarmants en termes de dégradation de performances (temps de réponse), ils révèlent toutefois que sur l'hyperviseur Xen, une VM peut souffrir du manque d'isolation lors de la réception et de la transmission de paquets réseau.

Dans [51], Barker et al. évaluent les impacts sur les temps de réponse applicatifs, d'une co-résidence d'une VM ayant une activité intensive avec une VM qui exécute une application sensible à la latence. Ils démontrent que les performances de la VM sont impactées par les pics de consommation des VMs co-résidentes. Les résultats montrent que le réseau et le disque sont les ressources qui souffrent le plus du manque d'isolation (dégradation de 75% de performance des tâches liées au disque [51]).

1.5.5 Déni de service économique

Le payement à l'usage est l'une des caractéristiques clés du modèle cloud permettant aux utilisateurs de payer en fonction des quantités de ressources consommées par leur services.

Les attaques par déni de service économique, aussi qualifiées d'attaques par consommation frauduleuse de ressources [52], cherchent à exploiter la propriété d'élasticité de l'allocation des ressources implémentée par les systèmes de gestion élastique de ressources adaptant les quantités de ressources allouées aux VMs en fonction de leurs besoins variables dans le temps. Ces systèmes peuvent être vulnérables à toute attaque agissant sur les quantités de ressources consommées par les VMs : déni de service ou processus malicieux s'exécutant au sein d'une VM pour consommer délibérément des ressources dans le but de provoquer une sur-facturation des ressources au client. Une attaque par déni de service économique peut présenter les mêmes caractéristiques qu'une augmentation légitime de la consommation en ressources d'une VM, ce qui rend sa détection peu aisée.

Dans [53], les auteurs estiment le coût que peut provoquer une attaque par déni de service économique à 280.8$ / heure dans un contexte où l'attaque est réalisée sur un service qui facture la bande passante réseau à 0.12$ / GB avec une attaque par déni de service distribué menée pendant une heure et générant une activité réseau moyenne de 5.2 Gbps.

Contrairement à une attaque par déni de service qui cherche à porter atteinte au service sur une courte durée en provoquant une indisponibilité du service, les attaques par déni de service économique cherchent à mettre en cause la durabilité du service fourni par l'opérateur en menant des attaques dont les effets sont plus subtiles mais observables sur une plus longue durée [54]. La première conséquence posée par ce type d'attaque est liée à la facturation des ressources consommées par l'attaque puisqu'un client pourrait se voir contraint de payer pour des ressources qui lui sont attribuées pour répondre à une attaque.

La section suivante engage une discussion autour des catégories d'attaques que nous venons de présenter, en cherchant à pointer les particularités de ces attaques qui exploitent les systèmes cloud.

1.5.6 Discussion

Le tableau 1.5 récapitule l'ensemble des catégories d'attaques détaillées dans les sections précédentes. Ces attaques sont classées en fonction de l'entité que doit manipuler l'attaquant pour mener son attaque. Les étiquettes représentent le type d'effets de l'attaque, pouvant être : vol d'information, vol de ressources, dégradation de disponibilité ou facturation.

— La première colonne du tableau regroupe des attaques dans lesquelles l'attaquant manipule uniquement l'espace de ressources de la VM sous son contrôle, pour produire des effets sur les VMs co-résidentes avec sa VM qu'il manipule. Les effets de ce type d'attaque sont obtenus par effets de bord, précisément grâce au partage des ressources entre les VMs,

— La deuxième colonne du tableau regroupe les attaques dans lesquelles l'attaquant manipule les ressources de la VM attaquée, lui portant ainsi préjudice. Ce type d'attaque n'exploite pas le partage de ressources et assurer la sécurité des VMs contre de telles attaques est de la responsabilité de l'utilisateur propriétaire de la VM,

— La dernière colonne du tableau représente les attaques menées contre l'hyperviseur, pour lesquelles il y a une intrusion directe dans l'espace de ressources de l'hyperviseur. Une attaque qui réussit à impacter l'hyperviseur, peut potentiellement provoquer les quatre résultats possibles d'une attaque ci-dessus mentionnés.

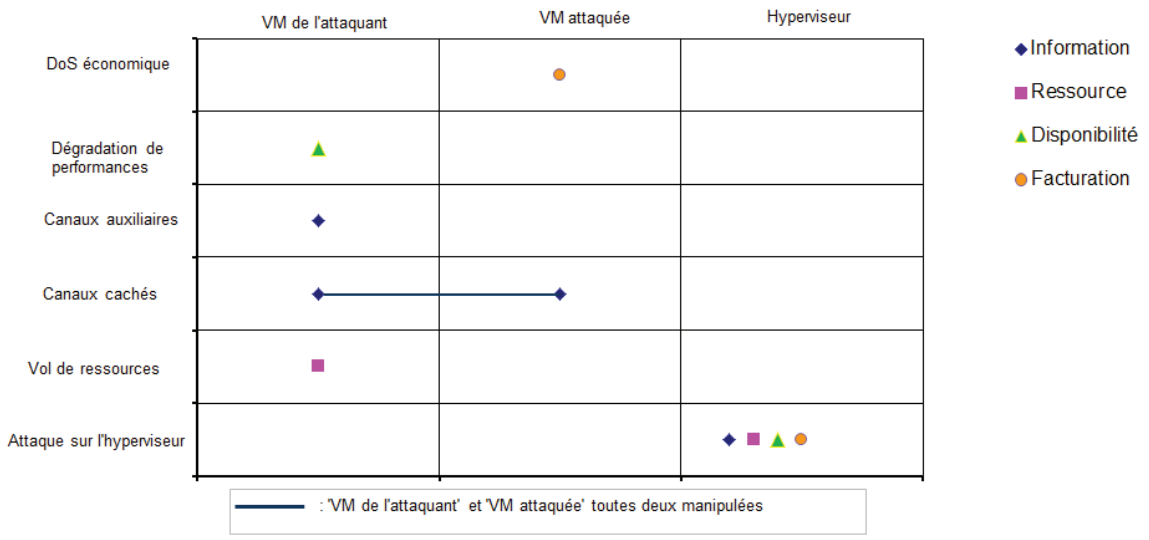


FIGURE 1.5 – Classification des attaques en fonction de l'entité manipulée

Les attaques regroupées dans la colonne 1 sont des attaques propres aux plate-formes virtuelles, car elles exploitent le partage des ressources entre des VMs appartenant à différents utilisateurs, ce qui est une propriété exclusive de ces plate-formes.

Une VM qui tente de se protéger au sein d'une plate-forme virtuelle peut superviser, grâce à des moyens de sécurité qui lui sont propres, son propre espace de ressources uniquement. Par conséquent, une VM cible d'attaques telles que celles regroupées dans la colonne 1 du tableau n'est pas en mesure de détecter la source de l'attaque. De plus, ces attaques ne visent pas l'hyperviseur (les effets de l'attaque ne sont pas visibles au niveau de l'hyperviseur). Par conséquent, si l'hyperviseur se limite à assurer sa propre protection uniquement, celui-ci ne sera pas non plus en mesure de détecter ces attaques.

Pour ces raisons, nous pensons que dans une plate-forme virtuelle, une sécurité centrée sur l'utilisateur – dans ce cas les moyens de sécurité inspectent uniquement l'espace de ressources de la VM à protéger – n'est pas suffisante pour assurer la sécurité des VMs et qu'une coopération forte entre les VMs et l'hyperviseur est nécessaire pour la détection de ce type d'attaque.

Dans la dernière partie de ce premier chapitre, nous introduisons les principales approches de détection d'intrusions et discutons des opportunités qu'offre la virtualisation pour ces systèmes.

1.6 Approches de détection d'intrusions

Un système de détection d'intrusions (IDS, de l'anglais Intrusion Detection System) a pour rôle d'observer et d'analyser l'exécution d'un système pour détecter toute forme d'activité anormale provoquée par une action possiblement malveillante au niveau de la cible analysée [55]. Les IDS offrent ainsi une sécurité complémentaire à celle assurée par les fonctions de sécurité défensives implémentées dans un système informatique.

Les systèmes de détection d'intrusions procèdent généralement en trois phases. Une première phase de détection dans laquelle l'IDS collecte les données nécessaires à l'analyse du système surveillé, analyse ces données et lève une alerte dès lors que les analyses effectuées révèlent une activité anormale du système. Une seconde phase de décision dans laquelle les alertes sont analysées et évaluées vis-à-vis des politiques de sécurité du système surveillé, à la suite de quoi l'IDS peut décider le déclenchement de contre-mesures (détection active) pour bloquer l'attaque. Enfin, une dernière phase de réaction qui correspond à l'activation de contre-mesures. Un système qui intègre cette dernière phase de réaction avant le succès de l'attaque est qualifié de système de prévention d'intrusions (IPS, de l'anglais Intrusion Prevention System).

Dans [56], Debar et *al.* proposent une taxonomie des IDS selon plusieurs paramètres : méthode de détection, source de données, détection active ou passive et périodicité de détection. Nous nous intéressons ci-après à deux critères particuliers de cette classification, à savoir les

méthodes de détection et la source de données.

1.6.1 Méthodes de détection

Pour l'analyse des événements collectés par un IDS, il existe principalement deux approches.

La première est une approche d'analyse comportementale ('anomaly-based detection'). Elle est basée sur l'hypothèse selon laquelle il est possible de modéliser le fonctionnement normal d'un système caractérisant un état d'exécution hors attaque du système puis considère toute déviation vis-à-vis de ce fonctionnement comme étant suspecte [57]. Ce type d'approche requiert une première étape d'apprentissage et de modélisation du système préliminaire à la phase d'observation pour la détection. L'avantage principal de cette approche est sa capacité à détecter des attaques non connues dès lors que celles-ci produisent des effets qui dévient d'une activité considérée comme normale par l'IDS. Cependant, cette approche a pour limite principale le nombre élevé de faux positifs correspondant à des situations d'activité normale du système, vues à tort par l'IDS comme anormales du fait qu'elles dévient du comportement normal appris par l'IDS.

La seconde approche, à base de signatures ('misuse-based detection'), considère qu'il est possible de décrire les profils d'exécution des attaques. Cette approche repose sur l'idée que la majorité des attaques exploitent des vulnérabilités qui sont connues et qu'il suffit par conséquent de caractériser les tentatives d'exploitation de ces vulnérabilités pour être en mesure de les identifier. Dans cette approche, l'IDS confronte l'exécution du système à une base de signatures d'attaques et lève une alerte dès lors que cette exécution correspond à l'une des signatures de la base [58]. L'inconvénient principal de cette approche est son incapacité à détecter des attaques non référencées dans la base de signatures.

Ces méthodes d'analyse sont applicables sur des données que l'on peut collecter à différents niveaux du système cible de l'analyse. Nous présentons ci-après les principales catégories d'IDS en fonction de la source des données analysées.

1.6.2 Sources de données

On distingue habituellement deux familles d'IDS : détection basée sur des données réseau et détection basée sur des données relatives à l'hôte. Pour un système virtualisé, une troisième source de données possible donne lieu à l'émergence d'une troisième catégorie d'IDS, les données relatives à l'hyperviseur.

1.6.2.1 Détection d'intrusions réseau

La détection d'intrusions basée sur le réseau (NIDS, de l'anglais Network Intrusion Detection) analyse les paquets réseaux échangés sur le réseau. Les sondes de collecte des données sont réparties soit à travers le réseau, soit en périphérie pour une analyse du trafic à l'entrée du réseau offrant ainsi une sécurité périmétrique.

Ces approches sont efficaces pour la détection d'attaques réseau telles que les dénis de service distribués, les exploits de failles de la pile réseau, les attaques de type 'homme au milieu', les attaques au niveau des protocoles, etc... Ces approches présentent une bonne résistance aux attaques puisque l'IDS s'exécute dans un espace de ressources isolé et différent de l'espace de ressources du système à surveiller [55]. Néanmoins, l'information limitée au trafic réseau, qui est prise en considération par cette approche, peut se révéler insuffisante pour la détection des attaques.

Dans cette approche, l'IDS supervise essentiellement des attaques en provenance du réseau et il est difficile pour ce type de système de détecter des attaques internes telles qu'une violation de politique de sécurité, une exécution d'un code malicieux ou toute attaque s'exécutant en local sur un hôte.

Dans un contexte cloud, il est difficile pour un utilisateur de reconstituer l'architecture physique de son infrastructure puisqu'il ne dispose pas d'information concernant la localisation de ses VMs, celles-ci pouvant également migrer entre les hôtes et changer d'emplacement dans le temps. Dans ces conditions, il devient difficile de déployer des NIDS pour assurer une sécurité périmétrique.

1.6.2.2 Détection d'intrusions hôte

La détection d'intrusions basée sur l'hôte (HIDS, de l'anglais Host Intrusion Detection) collecte et analyse l'activité d'un seul hôte en surveillant l'usage qu'il fait des ressources réseau, mémoire, processeur et stockage. Le HIDS s'exécute au sein même de l'hôte surveillé, ce qui le rend vulnérable aux attaques puisqu'une prise de contrôle sur l'hôte peut permettre la prise de contrôle du système de détection également. Cette approche bénéficie d'une vue riche de l'activité du système, ce qui lui procure la capacité de détecter un large spectre d'attaques que ne peut pas couvrir une détection NIDS, telles que des rootkits, des processus cachés ou un cheval de troie. Les sources de données analysées par cette catégorie d'IDS, sont souvent les journaux systèmes, les appels systèmes et les systèmes de fichiers.

Dans un système virtualisé, le HIDS peut s'exécuter à l'intérieur de la VM sous la forme d'un programme à installer dans la VM. Le HIDS inspecte la forme virtuelle des ressources allouées à la VM. Dans cette forme d'utilisation, l'IDS est exécuté par l'utilisateur propriétaire de la VM, reste sous son contrôle et hérite des avantages et limites des HIDS traditionnels.

1.6.2.3 Détection d'intrusions hyperviseur

L'observation possible de l'activité des VMs depuis la couche hyperviseur s'interposant entre les ressources matérielles et la VM, ouvre une troisième approche IDS possible où la protection des VMs vis-à-vis des intrusions est déléguée au fournisseur cloud. En 2003, Garfinkel et Rosenblum [29] introduisent l'approche de détection d'intrusion dite d'*introspection de VMs*, dans laquelle l'IDS tire avantage de la position privilégiée de l'hyperviseur pour analyser les ressources manipulées par les VMs, en s'exécutant au niveau de l'hyperviseur ou depuis une VM privilégiée dénommée VM de sécurité. Cette nouvelle catégorie d'IDS présente des propriétés intéressantes combinant l'isolation dont bénéficient les systèmes de détection basés sur le réseau et la vue riche de l'utilisation des ressources dont bénéficient les systèmes de détection basés sur l'hôte.

Plus précisément et toujours en référence aux travaux de Garfinkel, une exécution IDS au niveau de l'hyperviseur offre trois propriétés principales :

— **Isolation** : le système de détection ne partage pas le même espace de ressources que le système surveillé, il devient alors plus difficile pour un attaquant qui contrôle une VM de compromettre le système de détection,

— **Inspection** : en se positionnant au niveau de l'hyperviseur, le système de détection bénéficie de la propriété d'encapsulation de l'hyperviseur pouvant lui procurer une vue complète de l'état de la VM (registres CPU, mémoire, réseau et stockage),

— **Interposition** : l'hyperviseur en tant que couche s'interposant entre les ressources matérielles et les ressources virtuelles présentées aux VMs, procure au système de détection la capacité de s'interposer dans les flux d'exécution de la VM surveillée. Cette propriété permet une surveillance active dotée d'une capacité d'interruption de l'exécution d'une instruction voire de blocage d'une attaque, grâce à des programmes (hook) installés préalablement dans la VM [59] ou à l'extérieur de la VM au niveau de l'hyperviseur [60].

L'introspection est cependant confrontée à un certain nombre de défis, le plus important étant certainement le manque d'information sémantique sur l'activité de la VM (connue par le terme anglais 'Semantic Gap') [33] : depuis l'hyperviseur, l'état d'une VM est vu comme un simple flux de données binaires qu'il est difficile d'interpréter sans aucune connaissance de l'architecture du système d'exploitation de la VM.

Durant ces dernières années, un effort actif de recherche a été consenti pour résoudre cette problématique de reconstruction de l'écart sémantique entre la vue obtenue depuis l'hyperviseur et celle fournie par un système d'exploitation. Dans [61], les auteurs proposent une classification

en trois catégories, des différentes approches possibles de détection d'intrusions par introspection, en fonction de la méthode utilisée pour reconstruire l'écart sémantique :

— **Dans la bande :** le système de détection utilise un agent de collecte installé à l'intérieur de la VM, chargé de fournir l'information sémantique nécessaire à l'interprétation des données binaires que collecte le module principal qui s'exécute au niveau de l'hyperviseur. L'intérêt de cette approche est qu'elle permet d'obtenir aisément l'information sémantique de la VM permettant de réaliser des corrélations entre la vue de l'hyperviseur et celle de la VM pour détecter une activité que la seule vue des ressources depuis la VM ne permet pas de détecter. **Antfarm** [62] est un système qui implémente une analyse croisée de vues intra-VM et hyperviseur pour la détection de processus cachés. Cette approche, de par la présence de l'agent intra-VM, perd l'avantage de l'isolation de l'IDS vis-à-vis du système surveillé, recherché avec l'introspection et peut aussi être vue comme trop intrusive par les utilisateurs,

— **Hors bande** : le système de détection n'installe aucun agent à l'intérieur de la VM et découvre l'architecture du système d'exploitation de la VM grâce aux tables de symboles (copie du fichier System.map des VMs vers le Dom 0 ou l'hyperviseur pour des systèmes d'exploitation linux), ce qui permet à l'IDS d'interpréter l'activité binaire de la VM. Cette approche permet d'avoir une meilleure isolation que dans l'approche 'dans la bande', mais elle demeure dépendante d'une information délivrée par le système d'exploitation surveillé. Dans [63], les auteurs démontrent que cette approche est mise à mal si la structure du noyau du système d'exploitation de la VM est modifiée, l'architecture du système d'exploitation de la VM apprise par l'IDS devenant obsolète à la suite de cette modification,

— **Dérivation** : dans cette approche, seules les données bas niveau de la VM sont utilisées par l'IDS qui utilise uniquement sa connaissance de l'architecture matérielle pour interpréter les informations binaires. Si cette approche possède le grand avantage d'être complètement isolée du système à superviser et, de ce fait, robuste aux attaques, les informations que peut déduire un IDS de ce type restent très limitées. Cette méthode consiste par exemple à superviser les accès au registre CR3 qui contient les tables de processus de chaque VM. **Nitro** [64] est un système qui implémente cette approche pour l'analyse d'appels systèmes, révélant suffisamment d'information pour la détection de certains maliciels. Un autre intérêt de ce type d'approche est qu'elle est agnostique en ce qui concerne le système d'exploitation de la VM, ce qui lui procure une grande portabilité.

Compte tenu des défis posés par l'introspection, les propositions d'introspection actuelles sont souvent spécifiques à un système d'exploitation particulier et à des catégories d'attaques bien particulières également. Nous pouvons citer par exemple : vérification d'intégrité [65], analyse post-mortem [66], détection de rootkits [67], analyse de maliciels [68, 69]. Plusieurs de ces propositions utilisent la libraire d'introspection LibVirt [70], qui supporte l'introspection de systèmes d'exploitation linux, principalement avec les hyperviseurs Xen et KVM.

1.6.3 Discussion

La figure 1.6 récapitule les avantages et verrous de chacune des catégories d'IDS que nous avons détaillées dans cette section (réseau, hôte, hyperviseur). Pour chacune des catégories, nous considérons l'entité pouvant mettre en oeuvre ce type d'IDS dans le cloud. Nous retenons les points suivants :

— Protection basée sur le réseau : elle se heurte à la difficulté d'assurer une protection périmétrique des VMs d'un utilisateur à des points d'entrées spécifiques de leur réseau, à cause de la co-résidence des VMs qui implique le partage du réseau virtuel entre les VMs et leur dynamicité qui peut provoquer la reconfiguration du placement des VMs sur les hôtes de l'infrastructure,
— Protection basée sur l'hôte : exécutée par l'utilisateur de la VM, c'est l'option la plus pertinente pour une sécurité opérée par l'utilisateur, car celle-ci rend l'IDS complètement indépendant de la topologie réseau des VMs,
— Protection basée sur l'hyperviseur : opérable uniquement par l'opérateur de l'infrastructure, elle présente des propriétés avantageuses pour un IDS pouvant permettre d'offrir une valeur ajoutée réelle avec une vue multi-utilisateurs et multi-niveaux de l'exécution des VMs.

Nous pouvons observer à partir du tableau illustré sur cette figure 1.6, que l'opérateur possède plusieurs atouts pour offrir une meilleure sécurité aux utilisateurs, bénéficiant de la position avantageuse de l'hyperviseur lui permettant de gérer les ressources matérielles et virtuelles. L'approche hyperviseur permet d'effectuer des corrélations à travers les couches de la plate-forme virtuelle et aussi de répondre aux intrusions de façon optimisée, en mettant par exemple la VM infectée en quarantaine ou en bloquant le trafic réseau avant qu'il n'atteigne la VM cible de l'attaque [71].

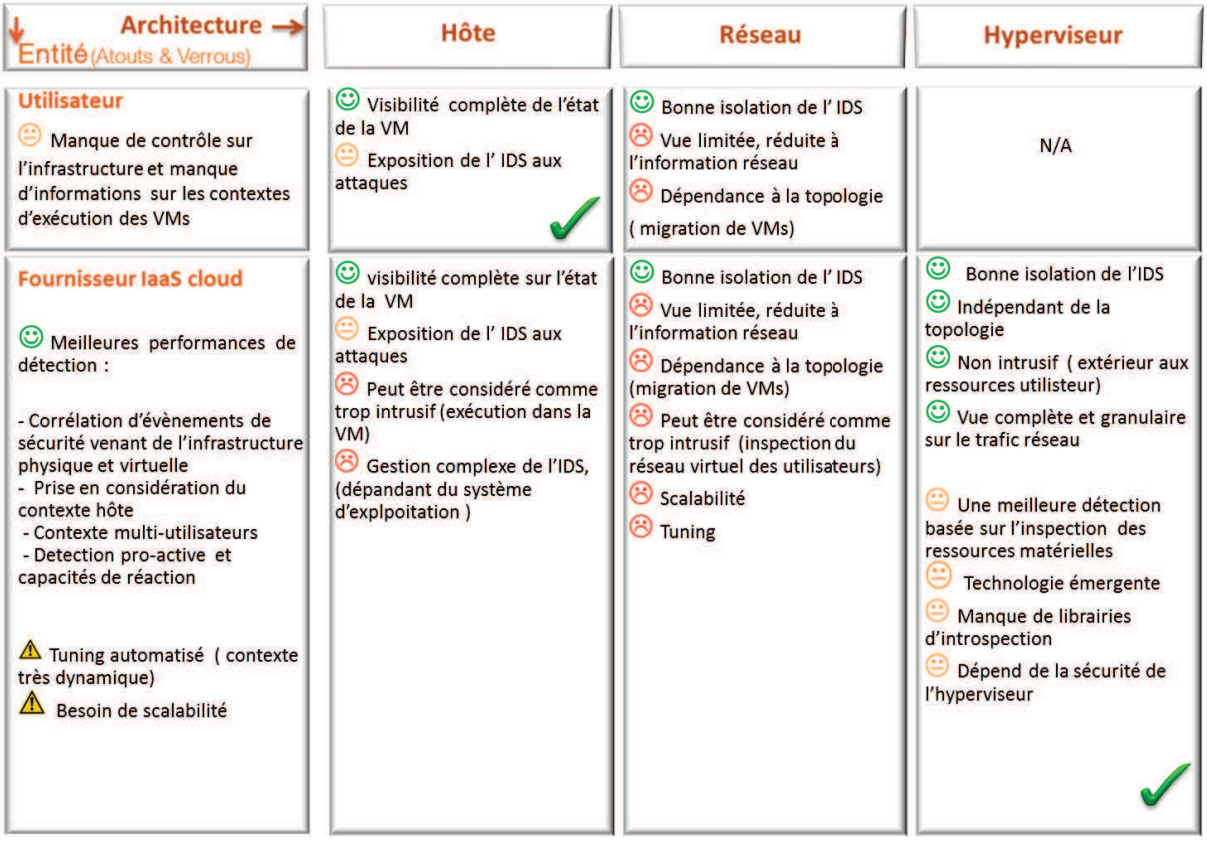


FIGURE 1.6 – Avantages et Inconvénients des IDS en fonction de l'entité qui les exécute

1.7 Conclusion

Dans ce chapitre nous avons introduit les concepts fondamentaux relatifs aux deux thématiques dans lesquelles s'inscrivent les travaux de cette thèse. Après avoir défini le cloud computing ainsi que les technologies qui lui sont inhérentes, à savoir la virtualisation et la gestion élastique des ressources, nous avons abordé les nouvelles vulnérabilités qu'introduisent ces caractéristiques dans les plate-formes cloud. Nous avons également considéré les systèmes de détection d'intrusions et le potentiel qu'offre la virtualisation à ces systèmes avec le recours aux techniques d'introspection.

Les premières conclusions de notre étude concernant les vulnérabilités et les solutions de détection d'intrusions pour les plate-formes cloud, montrent les limites d'une sécurité centrée sur l'utilisateur. Cela provient d'une part de l'apparition d'attaques par effets de bord inter-VM et d'autre part, de l'émergence de nouvelles solutions avantageuses de détection d'intrusions basées sur l'hyperviseur, qui permettent d'implémenter une sécurité plus sûre que celle implémentée par un IDS exécuté au sein même de la VM analysée.

Dans la suite de cette thèse, nous nous intéressons plus particulièrement à la sécurité de la gestion dynamique des ressources basée sur la migration de VMs, en considérant deux dimensions :

nous nous attachons tout d'abord à démontrer l'existence d'un nouveau domaine de vulnérabilités apparu avec la gestion élastique des ressources en environnement virtuel partagé, en démontrant et analysant l'attaque par migrations intempestives de VMs (chapitre 2). Nous proposons ensuite un système de supervision de l'utilisation du mécanisme de migration de VMs, analysant les profils de consommation de ressources des VMs pour identifier celles possiblement à l'origine d'un excès de migrations dû à l'attaque par migrations intempestives de VMs (chapitre 3).

Travaux de l'auteur sur cette thématique

— Reconsidering Intrusion Monitoring Requirements in Shared Cloud Platforms. **Kahina Lazri**, Sylvie Laniepce, Jalel Ben-Othman, RaSiem Workshop, 8th IEEE International Conference in, Availability, Reliability and Security (**ARES'13**), september 2013, Regensburg, Allemagne.

— Engineering Intrusion Prevention Services for IaaS Clouds : The Way of the Hypervisor. Sylvie Laniepce, Marc Lacoste, Mohammed Kassi-Lahlou, Fabien Bignon, **Kahina Lazri**, Aurelien Wailly, 7th IEEE International Symposium in, Service Oriented System Engineering **(SOSE'13)**, mars 2013, San Francisco Bay, États-Unis.

Chapitre 2

Attaque par migrations intempestives de VMs

Dans ce chapitre, nous examinons comment les systèmes de gestion dynamique de ressources peuvent être exploités par simple manipulation des quantités de ressources consommées par les VMs, pour amener ces systèmes à déclencher de façon abusive des migrations de VMs. Ces migrations consomment des ressources de l'infrastructure et impactent les performances des VMs migrées. Pour démontrer cette attaque inter-VMs, nous utilisons une plate-forme VMware qui met en oeuvre un algorithme de gestion automatique des migrations de VMs, i.e. Distributed Resource Scheduler (DRS). Nous analysons le détail du déroulement des expérimentations en supervisant le fonctionnement de cet algorithme pendant l'attaque. Nous examinons dans divers contextes la quantité minimum de ressources nécessaire au succès de l'attaque. Dans nos expérimentations réalisées sur des petits clusters, nous observons que la vulnérabilité d'un cluster vis-à-vis de cette attaque augmente avec la taille du cluster et avec le niveau d'agressivité de DRS. Enfin, nous montrons que le schéma de base de l'attaque peut être reproduit successivement plusieurs fois pour provoquer une série de migrations de VMs.

Sommaire

2.1 Introduction

DANS le chapitre 1, nous avons vu que dans les plate-formes cloud, les ressources sont gérées dynamiquement par logiciel grâce à la couche de virtualisation qui s'interpose entre les ressources physiques des serveurs hôtes et les environnements d'exécution des utilisateurs, i.e. les VMs consommant les ressources virtualisées qui leur sont présentées par l'hyperviseur. Les utilisateurs de VMs surestiment leurs besoins en ressources [72] et l'optimisation des taux d'utilisation des ressources par les opérateurs d'infrastructure cloud repose notamment sur le sur-engagement de ressources [20]. La virtualisation permet un certain nombre de mécanismes pour l'optimisation de l'exploitation des ressources, en particulier la migration automatique de VMs, qui est un mécanisme pivot des systèmes de gestion dynamique des ressources.

Nous avons analysé plusieurs catégories d'attaques dans lesquelles, il a été démontré que la consolidation sur un même hôte de VMs appartenant à des utilisateurs différents pose un certain nombre de défis en termes de sécurité pour les plate-formes du cloud [30, 73]. Il a été démontré que, sous certaines conditions, le partage des ressources permet à un attaquant ayant le contrôle d'une VM d'attaquer une VM co-résidente avec la sienne [37, 74]. S'agissant des mécanismes de gestion dynamique des ressources, l'existence de telles attaques exploitant la co-résidence de VMs n'a jamais été démontrée dans l'état de l'art antérieur.

Dans ce chapitre, nous examinons comment un attaquant peut influencer le système de gestion dynamique des ressources pour l'amener de manière abusive à déclencher des migrations de VMs. Plus précisément, nous étudions comment un attaquant peut manipuler les quantités de ressources consommées par ses propres VMs (ou des VMs sous son contrôle) durant leur exécution, pour impacter les décisions de migration du système de gestion des ressources. Les migrations consomment des ressources de l'infrastructure tant au niveau de l'hôte cédant la VM qu'au niveau de l'hôte destinataire de la VM [31] qu'au niveau de la bande passante réseau pour transmettre l'état courant des VMs migrées ; ces migrations sont donc coûteuses pour l'infrastructure. De plus, les VMs migrées – pas nécessairement celles manipulées pour provoquer des migrations – peuvent subir des dégradations de performances dues au processus de migration [75].

De nombreux travaux se sont intéressés à l'optimisation, l'évaluation ou encore la modélisation de la migration dynamique de VMs mais à notre connaissance celle-ci n'a jamais été considérée sous l'angle de la sécurité. Dans les travaux menés dans la thèse, nous considérons la migration automatique en tant que vecteur d'attaque exploité pour consommer de manière malveillante des ressources des infrastructures et dégrader les performances des VMs hébergées.

Au delà de cette attaque particulière que nous dénommons 'attaque par migrations intempestives de VMs' démontrée dans nos travaux, l'objectif est de mettre en évidence la vulnérabilité des systèmes de gestion dynamique des ressources vis-à-vis de profils de consommation en

ressources malveillants. Notre raisonnement s'appuie sur deux constats. D'une part, ces systèmes réagissent par nature à la consommation en ressources des VMs qui sont possiblement sous le contrôle d'attaquants. D'autre part, la multi-location crée une interdépendance entre les VMs [76] qui, dans le cas particulier de la gestion dynamique des ressources, fait subir aux VMs co-résidentes avec celles attaquantes des dégradations de performances en particulier lorsqu'elles sont migrées.

Contributions du chapitre

Dans ce chapitre, nous détaillons les contributions suivantes :

— Démonstration par l'expérimentation que les systèmes de gestion dynamique en ressources peuvent être vulnérables à la manipulation malveillante de la consommation de ressources des VMs,

— Analyse détaillée de l'algorithme VMware Distributed Resource Scheduler (DRS) que nous avons utilisé pour démontrer l'attaque par migrations intempestives de VMs,

— Évaluation par la mesure de l'exposition des clusters - considérés en tant qu'ensembles d'hôtes - vis-à-vis de l'attaque, dans le cas de DRS.

Organisation du chapitre

Ce chapitre est organisé comme suit. La section 2.2 livre des considérations générales sur les systèmes de gestion dynamique des ressources. La section 2.3 donne une description de l'algorithme DRS nécessaire à la compréhension de l'attaque. La section 2.4 présente l'attaque. La section 2.5 décrit en détail l'exécution de l'attaque par migrations intempestives de VMs et évalue les conditions de réussite de l'attaque, au travers d'une pluralité d'expérimentations. Enfin, la section 2.6 livre une discussion quant aux limites actuelles de la gestion dynamique des ressources dans le cloud en termes de sécurité et conclut ce chapitre.

2.2 Gestion dynamique de ressources et migration de VMs

Les systèmes de gestion dynamique de ressources sont principalement responsables de deux fonctions, i) placement initial des VMs sur les hôtes, ii) mesure durant toute la durée d'exécution de l'utilisation en ressources ou des performances d'exécution de ces VMs afin de garantir que les VMs disposent bien des ressources nécessaires pour atteindre une bonne performance d'exécution. Si une VM observe une contention de ressources, la migration de VMs peut être utilisée comme mécanisme permettant l'ajustement des quantités de ressources allouées à la VM.

Ci-dessous, nous détaillons le mécanisme de migration de VMs, d'une part pour mieux com-

prendre en quoi ce mécanisme est nécessaire à la gestion dynamique des ressources dans le cloud et d'autre part pour expliciter le coût que peut avoir la migration de VMs pour l'infrastructure et pour les VMs hébergées.

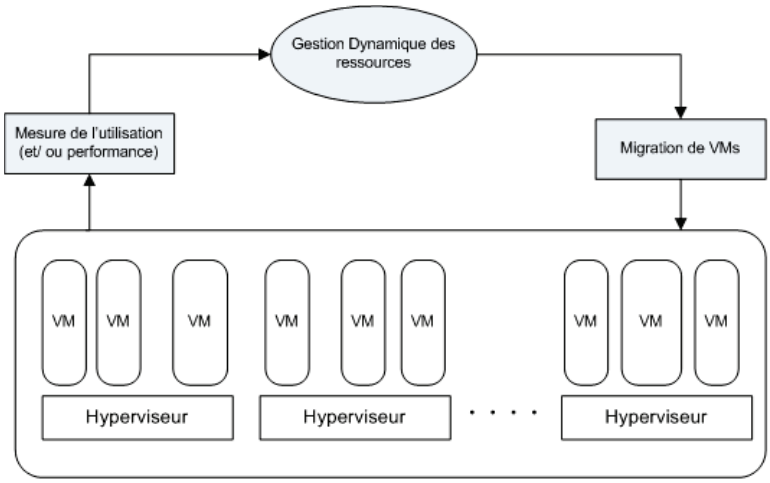


FIGURE 2.1 – Gestion dynamique de ressources

2.2.1 Migration de VMs

La migration d'une VM consiste à transférer son état courant, représenté par le contenu des registres CPU, le contenu des pages mémoire et les connexions entrées / sorties (réseau et disques), d'un hôte source vers un hôte de destination. Il existe différentes approches pour le transfert de l'état courant d'une VM d'un hôte source vers un hôte de destination. Ces approches se regroupent en trois catégories :

1. Arrêt-et-copie : plus connue sous l'appellation de migration à froid, dans cette approche il y a d'abord interruption de l'exécution de la VM sur l'hôte source, ensuite l'ensemble de l'état courant de la VM est transféré sur l'hôte destination, et enfin la VM est redémarrée sur l'hôte de destination. La migration à froid permet de déplacer le contenu des disques associés à une VM d'un espace de stockage vers un autre,

2. Pré-copie : cette approche est plus connue sous l'appellation de migration à chaud de VMs. Elle consiste à copier sur l'hôte destination le contenu des pages mémoire de la VM à migrer alors que celle-ci continue son exécution sur l'hôte source, à la suite de quoi la VM est redémarrée sur l'hôte de destination. Les pages mémoire modifiées au cours de cette phase par la VM à migrer sont par la suite itérativement transférées vers l'hôte de destination alors que la VM continue son exécution sur l'hôte source [77]. Enfin, pour

éviter de recopier indéfiniment les pages modifiées, l'exécution de la VM est suspendue sur l'hôte source pour empêcher toute nouvelle modification de page et une copie finale des pages modifiées est réalisée, cet arrêt de la VM intervient si la portion restante de mémoire modifiée est très faible ou si le seuil prédéfini d'itérations de copie est atteint. La VM peut alors reprendre son exécution sur l'hôte de destination. Le disque virtuel associé à la VM reste dans le même espace de stockage, ce qui implique que l'espace de stockage de la VM migrée doit être partagé entre l'hôte source et l'hôte destination,

3. Post-copie : cette approche est aussi une migration à chaud de VMs mais elle est moins répandue que la migration en pré-copie. Dans cette approche, seul l'état des vCPU et celui des connexions des entrées/sorties sont transférés dans une première phase [78], les pages mémoire de la VM sont transférées a posteriori à la demande alors que la VM s'exécute sur l'hôte de destination.

Nous examinons ci-après le coût du mécanisme de migration de machine virtuelle pour l'infrastructure et pour les VMs migrées.

2.2.2 Coût de la migration de VMs

Dans les deux approches de migration à chaud, le mécanisme de migration comprend deux phases, une phase de copie (pré-copie ou de post-copie) et une phase d'interruption de l'exécution de la VM. Durant la migration de VMs, celles-ci peuvent observer des dégradations de performances qui se quantifient essentiellement par deux métriques, i) le temps d'arrêt d'exécution de la VM (plus connue sous le terme anglais 'down time', ii) la durée totale de la migration, durant cette phase les performances de la VM migrée peuvent être affectées (le terme anglais 'migration time' désigne cette métrique, applicables aux étapes de copie et d'interruption).

Dans l'approche de migration en arrêt-et-copie, les durées du *'down time'* et du *'migration time'* sont proportionnelles à la taille de la mémoire physique utilisée par les VMs [79]. Les approches de migration en pré-copie et post-copie privilégient la minimisation du *'down time'* au risque d'une augmentation significative de la durée totale de la migration *'migration time'*.

Dans l'article [80] consacré à la migration à chaud en mode pré-copie, les auteurs rapportent que l'exécution des VMs peut être suspendue pendant une durée variant de 60 ms à 3 secondes selon la nature des applications exécutées par la VM, et l'exécution ralentie de 20% pendant toute la durée du processus pour une VM de 800 MB de mémoire. La durée totale du processus dépend de la nature des applications exécutées par la VM, de la capacité de la VM et du niveau de contention des hôtes source et destination impliqués dans la migration de la VM..

La migration est donc coûteuse i) pour l'infrastructure du fait des ressources consommées pour l'exécution de la migration des VMs, ii) pour la VM migrée du fait de la dégradation de performances engendrée par ce processus et, iii) possiblement pour les VMs co-résidentes si l'activité due à la migration au niveau des hyperviseurs s'ajoute à des états de contention des hôtes source et destination. Enfin, dans l'article [81], les auteurs démontrent que les migrations de VMs provoquent un surplus de consommation de puissance électrique qui peut être exploité par des attaquants, conjointement à d'autres moyens, pour réaliser une attaque par pics de consommation de puissance électrique pouvant aboutir au déclenchement des disjoncteurs des installations électriques.

Dans la suite du mémoire, nous utiliserons le terme migration de VMs pour désigner la migration à chaud en pré-copie. Dans la section suivante, nous détaillons les différentes approches de gestion dynamique de ressources dans le cloud qui s'appuient sur la migration de VM comme moyen utilisé pour assurer une allocation des ressources au plus près des besoins en ressources des VMs.

2.2.3 Gestion dynamique de ressources basée sur la migration de VMs

Le problème du placement des VMs sur les hôtes dans le cloud est souvent considéré dans l'état de l'art comme équivalent au problème du sac à dos multidimensionnel (MKP - Multidimensional Knapsack Problem) [82], qui est un problème NP-difficile. Nous ne donnons pas ici la formulation mathématique de ce problème puisque l'objectif n'est pas de proposer une solution à ce problème mais simplement de mettre en évidence la complexité du problème de placement de VMs sur des hôtes. En termes simples, le problème du sac à dos modélise une situation dans laquelle on cherche à trouver un remplissage d'un sac à dos ne pouvant supporter plus d'un certain poids, par un nombre donné d'objets ayant chacun une valeur et un poids. L'objectif étant de maximiser la valeur totale des objets sans dépasser le poids maximum supporté par le sac à dos.

Si l'on considère les VMs comme étant des objets multi-dimensionnels (chaque ressource étant une dimension : CPU, mémoire, ressources E/S) et l'hôte physique d'une capacité donnée comme étant le sac à dos, l'objectif sera de trouver un placement optimal des VMs sur les hôtes de sorte à maximiser l'occupation des hôtes. Alors que le placement des objets dans le problème du sac à dos est statique, il s'ajoute à la complexité de ce problème lorsqu'il s'agit du placement de VMs sur des hôtes, la variation dans le temps de la consommation en ressources des VMs ainsi que le coût de la migration de VMs qui est une contrainte importante puisqu'elle impose de prendre en considération ce coût si toutefois la distribution des VMs sur les hôtes est revue. Cette dernière contrainte impose de minimiser le nombre de migrations à opérer au sein d'un cluster. Les algorithmes de placement de VMs existant dans l'état de l'art s'appuient sur des heuristiques, les solutions proposées ne sont donc pas nécessairement optimales.

Nous livrons maintenant une présentation générale des différentes approches de gestion dynamique de ressources dans le cloud qui s'appuient sur la migration de VMs comme moyen d'optimisation de l'exploitation de l'utilisation des ressources des centres de données virtualisés. Nous examinons ci-après divers objectifs poursuivis par les algorithmes de gestion dynamique de ressources mettant en oeuvre des mécanismes de migration dynamique de VMs [83].

2.2.3.1 Migration pour la gestion des contentions

Les contentions de ressources imputables à l'opérateur de l'infrastructure surviennent lorsqu'une VM dispose d'une quantité de ressources insuffisante sur un hôte donné vis-à-vis des engagements en qualité de service contractualisés entre l'opérateur et ses clients. Si l'hôte hébergeant la VM ne dispose pas de suffisamment de ressources pour honorer les engagements en qualité de service, la VM est migrée sur un autre hôte pouvant répondre au besoin en ressources de la VM.

La détection de l'occurrence d'une contention en ressources peut se faire soit de façon réactive soit de façon proactive.

La détection réactive de contention repose sur la supervision de métriques qui quantifient les performances des VMs et la consommation en ressources des VMs (latence, débit, nombre d'instructions exécutées par seconde, etc...). Le système considère qu'il y a contention si la dégradation de la performance et/ou l'utilisation en ressources des VMs dépasse un niveau toléré.

La détection proactive de contentions se base sur des techniques d'apprentissage pour l'identification anticipée de l'occurrence d'une contention grâce à l'identification de symptômes qui caractérisent une contention de ressources. Une augmentation de la demande en ressources d'une VM peut être révélatrice d'une possible augmentation du besoin en ressources de la VM. Lorsqu'une contention est détectée et que l'hôte ne dispose pas d'une réserve de ressources pour palier à la contention, la migration de VMs est alors utilisée comme un moyen permettant de mettre à disposition de la VM migrée davantage de ressources sur un autre hôte disposant de ressources libres ou bien de migrer une autre VM de l'hôte actuel vers un autre hôte pour libérer des ressources sur l'hôte actuel.

Sandpipper [84] est un système de gestion dynamique de ressources implémenté avec des hyperviseurs Xen [21]. Il détecte l'occurrence d'une contention de ressources et considère cette contention comme critère de déclenchement de la migration d'une VM lorsqu'il n'est pas possible de mettre davantage de ressources à la disposition de la VM sur l'hôte actuel. La détection de la contention par *Sandpipper* se fait de façon réactive soit en adoptant une approche en boite noire dans laquelle seule l'utilisation en ressources CPU et réseau des VMs vue de l'hyperviseur est prise en compte pour l'évaluation de la contention des VMs, soit sur une approche en semi-boite noire (gray-boxe) dans laquelle des agents sont installés à l'intérieur de la VM et sont chargés de

mesurer des métriques de niveau système d'exploitation afin d'estimer en particulier l'utilisation mémoire des VMs. Cette approche en semi-boite noire est moins souhaitable car plus intrusive que l'approche boite noire, mais dans la version de Xen utilisée, il n'est pas possible de collecter la consommation mémoire des VMs depuis l'hyperviseur.

Lorsque la contention est identifiée, *Sandpipper* doit sélectionner les VMs à migrer ainsi que les hôtes en mesure de recevoir ces VMs. Pour cela *Sandpipper* utilise deux métriques, appelées 'volume', intégrant les trois ressources mémoire, CPU, réseau pour restituer la charge des VMs et celle des hôtes. Pour sélectionner l'hôte pouvant recevoir une VM, *Sandpipper* trie les VMs et les hôtes par ordre décroissant de volume et cherche à déterminer si la VM du plus fort volume s'exécutant sur l'hôte de plus fort volume pourrait s'exécuter sur l'hôte de plus faible volume. L'algorithme traite ensuite l'hôte sous contention ayant la charge la plus élevée suivant, et reproduit le même processus.

Une difficulté supplémentaire pour ce type d'approche est de définir les seuils de charge et de contention des hôtes à l'origine du déclenchement de la migration de VMs. Pour *Sandpipper*, les auteurs proposent un seuil autorisé d'utilisation CPU et réseau de 75% sur un serveur.

Au total, la décision de la migration de VMs dans *Sandpipper* est basée sur des métriques liées à la charge des VMs et à la charge du hôte exécutant ces VMs, ce qui fait de *Sandpipper* une approche centrée sur l'hôte (Host-Centric).

Scatter [85] est un système de gestion de ressources qui prend en considération le sur-engagement des ressources des hôtes du cluster. La particularité de *Scatter* est de considérer le cas où aucun hôte ne dispose de suffisamment de ressources pour accueillir une VM et de traiter ce cas par un échange de VMs entre les hôtes. Lors de la migration d'une VM, Scatter prend en considération le niveau de corrélation de la consommation en ressources entre les VMs d'un même hôte ainsi que le coût en bande passante réseau des échanges entre VMs. Si deux VMs échangent beaucoup de trafic réseau, *Scatter* peut alors migrer les deux VMs vers un même hôte ou alors garder ces VMs sur l'hôte actuel et privilégier la migration d'une VM tierce qui n'a pas de dépendance de communication réseau avec des VMs de l'hôte. *Scatter* cherche à minimiser les risques de surcharge des hôtes, le nombre de migrations exécutées ainsi que minimiser les coûts des échanges réseau entre les VMs.

Comme pour *Sandpipper*, *Scatter* prend ses décisions de migration de VM sur des métriques de mesure de la contention d'un hôte, *Scatter* poursuit aussi une approche centrée sur l'hôte.

2.2.3.2 Migration pour l'optimisation énergétique

La migration peut également être utilisée dans un objectif de minimiser les quantités de ressources non utilisées sur les hôtes d'un cluster, i.e. de maximiser la consolisation des hôtes. Dans ce cas, la migration est utilisée pour optimiser le placement des VMs sur les différents hôtes composant un cluster de sorte à maximiser les taux d'utilisation des ressources de chaque

hôte pour minimiser le nombre d'hôtes constituant un cluster. La figure 2.2 illustre le résultat de l'application de cette approche sur un cluster d'hôtes.

Une étude de Microsoft révèle que le coût des ressources physiques compte pour 45% du coût total des centres de données et que le coût de l'énergie compte pour 15% du coût total [86]. Le coût énergétique est par conséquent non négligeable, ce qui renforce l'intérêt de la consolidation des hôtes.

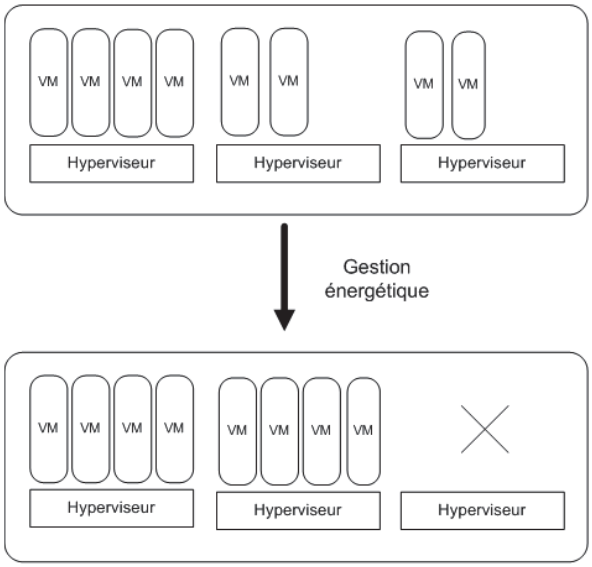


FIGURE 2.2 – Gestion de ressources suivant l'approche par consolidation des serveurs

Dans [87], Ferreto *et al*. réduisent le problème du placement des VMs sur les hôtes au problème du sac à dos multi-dimensionnel avec la contrainte supplémentaire de minimiser le nombre d'hôtes en fonctionnement dans le cluster, plusieurs heuristiques permettant de résoudre ce problème ont été proposées. Le besoin de redistribuer les VMs sur les hôtes est ré-évalué de façon périodique, tout en veillant à minimiser le nombre de migrations de VMs pour limiter les dégradations de performances induites par ces migrations.

Verma *et al.* proposent ***pMapper*** [88], un système de placement de VMs prenant en compte l'optimisation de la consommation énergétique, les contraintes de performances d'exécution des VMs ou encore les contraintes liées à l'intégration du coût de la migration de VMs dans le processus de décision de migration. *pMapper* collecte les performances d'exécution de toutes les VMs en présence dans un cluster ainsi que leurs consommations énergétiques. L'intelligence de pMapper réside dans l'agrégation des recommandations dictées par des gestionnaires d'énergie, de performances et de migrations pour trouver un compromis et proposer une allocation en ressources au plus près des besoins des VMs tout en cherchant à garantir une bonne performance d'exécution des VMs et à minimiser la consommation en énergie des clusters.

Entropy [89] est un système de placement dynamique de VMs sur un cluster d'hôtes, proposé par l'école des Mines de Nantes. Entropy a pour objectif de minimiser le nombre d'hôtes en utilisation dans un cluster et utilise la programmation par contraintes en suivant deux phases principales. La première basée sur les contraintes, prend en considération la capacité des hôtes, la capacité des VMs (CPU et mémoire), calcule un placement qui implique un nombre minimum d'hôtes et élabore un plan de reconfiguration de la répartition des VMs qui permet de satisfaire ces contraintes. La seconde phase prend en considération le coût et la faisabilité des migrations proposées en première phase et cherche à réduire le nombre de migrations. Entropy est construit en deux couches, un serveur Entropy central et des instances Entropy s'exécutant dans les VMs, ce qui permet à Entropy d'adapter le nombre d'instances en fonction du nombre d'hôtes et de VMs du cluster.

Xu *et al.* [90], proposent un algorithme chargé de déterminer une distribution des workloads sur différentes VMs puis de distribuer ces VMs sur les hôtes en utilisant la migration de VMs avec trois objectifs, i) minimisation des quantités de ressources non utilisées sur les hôtes, ii) minimisation de la consommation énergétique et, iii) minimisation de la consommation thermique c'est à dire les pics de température engendrés par l'activité des serveurs. Ils utilisent l'algorithme GGA [91] ('Grouping Genetic Algorithm') qui permet de regrouper des problèmes ayant des contraintes différentes en plusieurs groupes. L'algorithme proposé traite uniquement du placement initial de VMs et ne considère pas la révision du placement en fonction de la consommation en ressources des VMs.

Dong *et al* [92] réduisent le problème de placement de VMs sur les hôtes d'un cluster à deux problèmes connus, qui sont le problème du sac à dos et le problème de l'affectation quadratique [93]. Les auteurs proposent un algorithme qui considère non seulement la contrainte de minimisation du nombre d'hôtes constituant un cluster pour minimiser la consommation énergétique mais aussi la minimisation de la bande passante réseau consommée au sein d'un centre de données. Cet algorithme traite uniquement du placement initial de VMs et ne réagit pas à la variation de la charge des VMs pour changer dynamiquement la répartition des VMs sur les hôtes.

2.2.3.3 Migration pour l'équilibrage des charges

L'équilibrage de la charge entre les hôtes composant un cluster vise à optimiser le placement des VMs sur les différents hôtes pour minimiser les risques de contention sur chacun des hôtes tout en maximisant les taux d'utilisation des ressources, en évitant de surcharger certains hôtes en présence d'hôtes très peu chargés. Cette approche qui vise de minimiser la dispersion de la charge entre les hôtes, suppose une connaissance au niveau cluster de la répartition des charges entre les hôtes pour juger des migrations de VMs à opérer. La figure 2.3 illustre le résultat de l'application de cette approche par équilibrage des charges au sein d'un cluster.

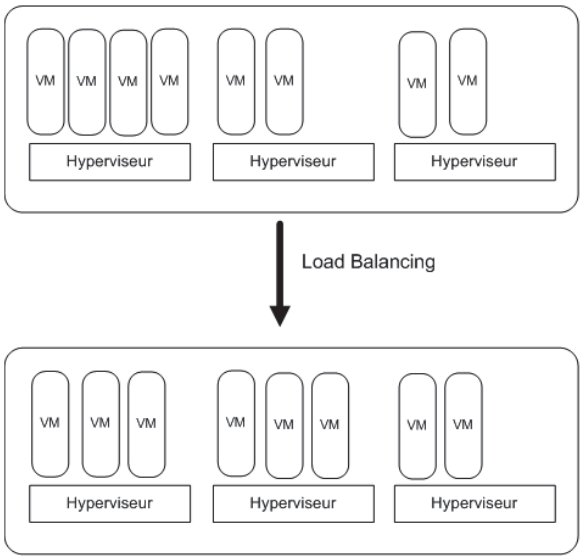


FIGURE 2.3 – Gestion de ressources suivant l'approche par équilibrage de charge

Harmony [94] définit des seuils de charge autorisés pour chaque hôte et serveur de stockage au delà desquels les VMs doivent être migrées pour alléger la charge en excès. En maintenant l'utilisation en ressources des hôtes au plus près de ces seuils, *Harmony* assure une répartition équilibrée de la charge des hôtes composant un cluster.

Arzuaga *et al.* [95] proposent le calcul d'une nouvelle métrique appelée *Virtualized Server Load* (VSL), cette métrique représente l'utilisation en ressources des hôtes et est utilisée pour le calcul de l'équilibrage de charge au sein d'un cluster d'hôtes comme suit. Le ratio de la moyenne sur l'écart-type de ces métriques relatives aux différents hôtes est considéré comme un indicateur de l'équilibrage de la charge au sein d'un cluster. Lorsqu'il y a déséquilibrage de la charge des hôtes, des VMs sont migrées de sorte à minimiser l'écart type des *VSL* des hôtes.

En conclusion de cette présentation de quelques systèmes de gestion dynamique des ressources et selon notre connaissance de l'état de l'art, les algorithmes de gestion dynamique de ressources prennent leurs décisions uniquement en fonction des quantités de ressources consommées par les VMs sans aucune considération de sécurité. Or, les quantités de ressources consommées par les VMs durant leur exécution sont aisément manipulables à des fins malveillantes par des attaquants pour forger des profils de consommation pouvant fausser les décisions de ces algorithmes.

Dans la suite de ce chapitre, nous nous intéressons à démontrer que cette dépendance des décisions des systèmes de gestion dynamique des ressources aux quantités de ressources consommées par les VMs constitue par nature une vulnérabilité de sécurité de ces systèmes, qu'il convient de considérer.

Pour démontrer l'attaque par migrations intempestives de VMs, nous utilisons l'algorithme commercial VMware DRS [96] qui poursuit prioritairement un objectif d'équilibrage des charges au sein du cluster. Notre choix s'est porté sur DRS pour deux raisons principales. D'abord, DRS est l'algorithme de gestion dynamique des ressources souvent utilisé en référence dans l'état de l'art pour l'évaluation de propositions de nouveaux algorithmes. De plus, l'utilisation d'un algorithme commercialisé et disponible facilite la reproduction par la communauté des résultats que nous obtenons dans nos expérimentations. Selon notre connaissance de l'état de l'art, il n'existe pas d'algorithmes de gestion dynamique de ressources ouverts et les versions d'algorithmes publiées dans les articles tels que ceux décrits dans la section précédente [84, 87, 94, 95, 90] sont souvent partielles.

Dans la section suivante, nous détaillons des éléments de conception de l'algorithme DRS.

2.3 L'algorithme Distributed Resource Scheduler (DRS)

L'algorithme DRS [96, 97] adopte une approche globale de niveau cluster pour le placement initial des VMs au sein du cluster et pour leurs éventuelles migrations automatiques ultérieures garantissant une distribution de charge équilibrée entre les hôtes composant le cluster.

Les sections suivantes donnent les éléments de compréhension des différents principes algorithmiques de DRS dont une version simplifiée publiée par VMware est présentée dans l'algorithme 1.

Algorithm 1: Version Simplifiée de l'algorithme DRS [96]

```
Entrées: Vue globale du cluster (hôtes et VMs)
chlsd ← σ(N_h)
NumMigrations ← 0
while (chlsd < thlsd) and (NumMigrations < MaxMigrations) do
    BestMigration ← NULL
    Max_o ← 0
    foreach VM v dans le cluster do
        foreach hôte h de destination compatible do
            o←amélioration du chlsd si v est migrée vers h
            if le gain de la migration > coût then
                if o>Max_o then
                    BestMigration←migrer v vers h
                    Max_o←o
    if BestMigration is null then
        break
    Appliquer BestMigration dans l'algorithme et mettre à jour chlsd
    NumMigrations++
```

Ci-dessous, nous analysons l'algorithme DRS en détail, nous nous intéressons en particulier aux métriques et aux paramètres que DRS prend en considération pour évaluer le besoin de migrer une VM au sein d'un cluster.

2.3.1 Métriques de décision

Les décisions de migration de l'algorithme DRS s'appuient sur trois niveaux de métriques - niveau VM, niveau hôte et niveau cluster - établis pour les ressources mémoire et CPU dans la version de DRS étudiée (version 4.1 vSphere) :

— **Niveau VM** : DRS calcule pour chaque VM, à chaque invocation de l'algorithme, la métrique d' 'Engagement Dynamique' (le terme utilisé habituellement est le terme anglais *Dynamic Entitlement* ici employé) qui correspond à la quantité de ressources à allouer à une VM pour la mémoire d'une part et le CPU d'autre part. Le calcul de l'engagement dynamique prend en considération sa demande courante de ressources, ses contraintes de ressources (réservation, limite et partage), la consommation des autres VM co-localisées et la capacité du cluster. DRS calcule aussi la métrique de 'Ressources Demandées' (*Static Entitlement*) qui est une estimation en absolu du besoin en ressources d'une VM prenant en considération les contraintes que définit le client pour sa VM (limite, réserva-

tion, partage) et faisant abstraction du contexte d'exécution des VMs, c'est à dire de la consommation des VMs co-résidentes. Par conséquent, une VM qui s'exécute sur un hôte en état de contention peut se voir traitée de façon moins avantageuse en comparaison avec une VM s'exécutant sur un hôte disposant de suffisamment de ressources. En état de contention, l'engagement dynamique estimé par DRS peut être inférieur à la quantité de ressources demandée par une telle VM tel qu'estimé par DRS,

— **Niveau hôte** : DRS calcule la métrique d' 'Engagement Normalisé' (le terme utilisé habituellement est le terme anglais *Normalized Entitlement* ici employé) pour chaque hôte, qui est la somme des *Dynamic Entitlement* des VMs hébergées par l'hôte rapportée à la capacité du hôte. Si l'Engagement Normalisé' d'un hôte est inférieur à 1, cela signifie que l'hôte n'est pas en état de contention et que les VMs devraient recevoir la totalité des ressources qu'elles demandent. L'ensemble des métriques d'Engagement Normalisé rapportées à chacun des hôtes donne une vision de l'utilisation des ressources du cluster,

— **Niveau cluster** : DRS prend en compte deux métriques principales pour évaluer le besoin de migration : la métrique 'Ecart type du chargement des hôtes courant' (le terme utilisé habituellement est le terme anglais 'Current Host Load Standard Deviation' (*chlsd*) ici employé) et la métrique 'Ecart type du chargement des hôtes cible' (le terme utilisé habituellement est le terme anglais 'Target Host Load Standard Deviation' (*thlsd*) ici employé). *chlsd* est l'écart-type courant de l'ensemble des métriques *Normalized Entitlement*. *thlsd* est la valeur seuil de *chlsd*, au delà de laquelle DRS entre dans un processus de décision pour évaluer le besoin et les possibilités de migrations. Si *chlsd* est inférieur au seuil *thlsd*, aucune migration de VM n'a lieu. Si *chlsd* excède *thlsd*, DRS procède à une série d'évaluations de certaines conditions - décrites ci-dessous - pour définir quelle VM doit être migrée, le cas échéant. Au total, *thlsd* définit le degré de déséquilibre de charge toléré au sein du cluster. Les valeurs prises par *thlsd* sont davantage détaillées en section 2.3.4.

La condition de valeur de *chlsd* supérieure à *thlsd* n'est cependant pas la seule prise en compte par DRS. DRS adopte également un principe de minimum de gain.

2.3.2 Minimum de gain

Pour décider la migration d'une VM, DRS évalue le gain qu'elle permet en termes d'amélioration de l'état d'équilibrage de charge au sein du cluster.

Le seuil définissant le minimum de gain nécessaire pour procéder à la migration d'une VM est recalculé dynamiquement et dépend notamment du nombre de VMs et d'hôtes en présence

dans un cluster et sa valeur diminue lorsque le déséquilibrage de charge dans le cluster est très élevé. Les valeurs que prend le seuil du minimum de gain nécessaire pour la migration d'une VM paraissent importantes au vu des observations de nos expérimentations et peuvent considérablement impacter les décisions de migration de VMs édictées par DRS, cependant à notre connaissance la formule précise de calcul de ce seuil n'est pas documentée.

Nos observations relevées lors de nos expérimentations et rapportées en section 2.5 nous indiquent que les valeurs prises par ce seuil sont déterminantes et impactent les décisions de migrations de VMs dictées par DRS.

Enfin, DRS prend en considération une évaluation du coût et du bénéfice d'une migration pour juger de sa pertinence.

2.3.3 Coût-Bénéfice

Dans cette phase d'évaluation, DRS établit une estimation du risque et du bénéfice de la migration de la VM sélectionnée en évaluant la capacité de cette migration à résorber la contention. Les migrations de VMs jugées trop coûteuses par rapport au gain et à la pérennité de ce gain, en termes d'amélioration d'équilibrage de charge dans le cluster ainsi que de la durée estimée de cette amélioration sont rejetées. C'est pourquoi DRS cherche à éviter de migrer des VMs dont le profil de consommation est considéré comme étant instable puisque le risque, dans ce cas, de voir par la suite la VM demander davantage de ressources ou réduire sa consommation en ressources après la migration est considéré comme étant élevé. Nous reviendrons sur ce principe que nous avons pu vérifier grâce à nos observations à l'occasion de nos expérimentations (section 2.5.3) et qui s'avère avantageux pour l'attaque.

Dans cette évaluation des coûts et des bénéfices d'une migration donnée, est considéré comme coût, le coût de la migration en termes de ressources consommées et en termes de dégradation de performances d'exécution de la VM migrée (tel que présenté en section 2.2.2), est considéré comme risque, le critère d'instabilité de la consommation de la VM à migrer, et enfin est considéré comme bénéfice, l'apport en équilibrage de charge pour le cluster, les quantités de ressources qui seront rendues disponibles pour les VMs de l'hôte source co-résidentes avec la VM migrée et aussi le gain en disponibilité de ressources pour la VM migrée sur l'hôte de destination.

2.3.4 Seuil de déséquilibrage de charge toléré

Trouver le bon compromis entre le degré de tolérance de déséquilibre de charge au sein du cluster et le nombre résultant de migrations nécessaires pour maintenir l'équilibre visé est essentiel. Une tolérance au déséquilibre de charge trop contraignante (*thlsd* faible) génère un nombre possiblement trop élevé de migrations ; par opposition, une large tolérance au déséquilibre

de charge (*thlsd* élevé) ne permet pas d'optimiser les taux d'utilisation des ressources. Pour permettre le paramétrage de cette tolérance, DRS offre cinq niveaux possibles d'agressivité déclinés selon autant de valeurs de *thlsd* : 'conservative', 'moderately conservative', 'moderate' (valeur par défaut), 'moderately aggressive' et 'aggressive'.

A notre connaissance, peu d'attention a été portée aux valeurs de *thlsd* dans la littérature. Nous nous sommes intéressés à ces valeurs car nous pensons qu'un examen attentif de ces valeurs est indispensable pour maîtriser pleinement le fonctionnement des plate-formes de virtualisation VMware.

Grâce à notre plate-forme d'expérimentation (décrite plus loin dans ce chapitre, à la section 2.5), nous avons pu collecter certaines valeurs de *thlsd*. Il apparaît que dans la version de DRS étudiée (version 4.1) *thlsd* varie non seulement avec le niveau d'agressivité de DRS mentionné ci-dessus mais aussi avec le nombre d'hôtes composant le cluster, à l'exclusion de tout autre paramètre. Les valeurs collectées nous ont permis de reconstituer la formule de calcul de *thlsd* :

$$thlsd = \frac{Agr}{\sqrt{N}} \tag{2.1}$$

où N est le nombre d'hôtes dans le cluster et Agr une constante dépendant du niveau d'agressivité de DRS : 0.424, 0.282, 0.141, 0.07 pour respectivement les niveaux 'moderately conservative', 'moderate', 'moderately aggressive' et 'agressive' (la migration automatique est désactivée pour le niveau 'conservative').

La figure 2.4 illustre la variation des valeurs de *thlsd* de la formule de calcul (2.1) pour chacun des quatre niveaux d'agressivité de DRS, en fonction du nombre d'hôtes du cluster (limité à 32 hôtes avec DRS). Nous observons que pour les quatre niveaux d'agressivité de DRS, la valeur de *thlsd* décroît quand la taille du cluster augmente ; nous pensons que ce choix de conception de VMware permet d'exploiter le nombre accru d'opportunités de migrations dans les grands clusters, dans l'objectif d'un meilleur équilibrage des charges. Nous observons également sur cette figure 2.4 que les valeurs de *thlsd* sont de moins en moins différenciées en fonction du niveau d'agressivité de DRS, quand la taille du cluster augmente. Cela signifie que les petits clusters sont plus sensibles au niveau d'agressivité de DRS que les grands, en matière de valeurs de *thlsd*.

De cette analyse de DRS, nous retenons que *thlsd* est le paramètre essentiel impactant le plus l'occurrence de migrations dans le cluster.

Dans la section suivante, nous démontrons comment un attaquant peut modifier la dispersion de charges entre les hôtes d'un cluster (*chlsd*), par une simple manipulation des quantités de ressources consommées par les VMs sous son contrôle, pour amener DRS à exécuter des migrations de VMs.

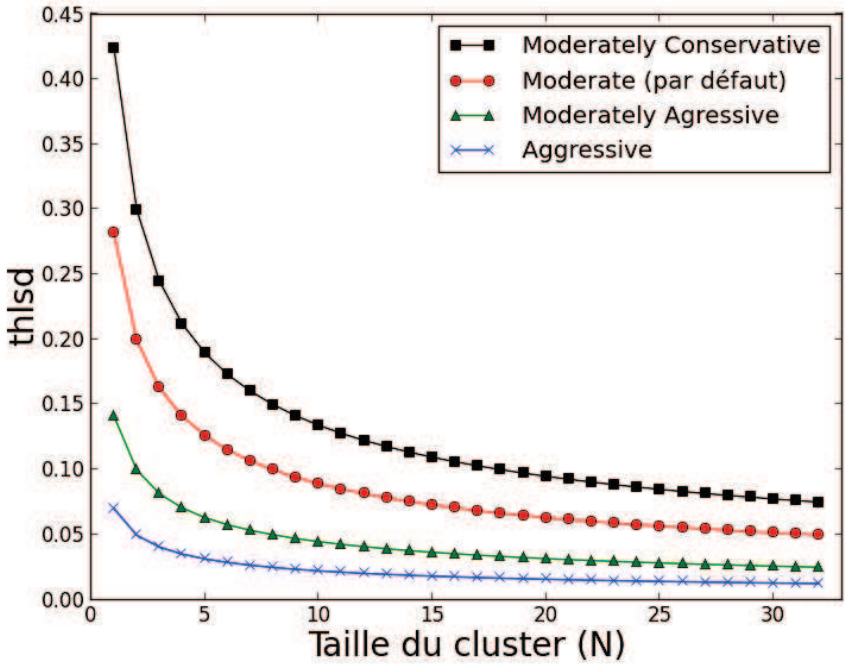


FIGURE 2.4 – Valeurs de *thlsd* pour les quatre niveaux d'aggressivité de DRS

2.4 Description de l'attaque

Pour réaliser l'attaque par migrations intempestives de VMs, l'attaquant fait délibérément fluctuer les quantités de ressources consommées par les VMs sous son contrôle, pour créer suffisamment de déséquilibre de charge entre les hôtes du cluster afin d'amener DRS à déclencher des migrations pour rétablir l'équilibre perdu.

Plus précisément, l'attaquant alterne les quantités de ressources consommées par ses VMs, entre des valeurs très hautes et des valeurs très basses. L'élément clef de cette stratégie d'attaque est simple. Quand les VMs malveillantes consomment de faibles quantités de ressources, l'hôte les hébergeant est vu par DRS comme étant en sous-charge et DRS tend à migrer des VMs vers cet hôte : les VMs sont en quelque sorte "attirées" par cet hôte en sous-charge. Par un raisonnement similaire, on comprend que lorsque les VMs malveillantes consomment des quantités importantes de ressources, l'état de l'hôte les hébergeant est considéré par DRS comme étant surchargé et DRS tend à déclencher des migrations de VMs pour alléger cette charge : les VMs sont en quelque sorte "repoussées" hors de l'hôte.

L'effet peut être cumulatif si le profil de consommation de ressources malveillant à effet de repoussement sur les VMs est combiné à un profil à effet d'attraction sur les VMs car, dans ce cas, l'attaquant peut exécuter le profil à effet de repoussement sur les VMs alors que l'hôte exécutant les VMs de l'attaquant est dans un état favorable à l'attaque - i.e. déjà chargé de par

une charge préalablement créée grâce à la phase d'attraction - ce qui permet d'atteindre plus aisément le niveau de charge provoquant un déséquilibre suffisant pour amener DRS à déclencher des migrations.

Chaque migration de VM - attraction ou repoussement - est obtenue en créant un déséquilibre de charge entre hôtes plus grand que celui toléré, ce qui nécessite pour l'attaquant de manipuler la quantité de ressources consommées par ses VMs de telle sorte à obtenir un *chlsd* supérieur au seuil *thlsd*. La démonstration de la faisabilité de cette attaque et son évaluation sont présentées dans la section suivante.

2.5 Implémentation et Evaluation

Dans cette section, nous décrivons dans un premier temps le dispositif d'expérimentation que nous avons mis en place pour la démonstration de l'attaque par migrations intempestives de VMs, ensuite nous présentons de façon détaillée le déroulement d'un schéma d'exécution de base de cette attaque, plus loin nous livrons une évaluation du niveau d'exposition des clusters face à ce type de vulnérabilité et enfin les résultats de l'exécution du schéma d'attaque combinant 'repoussement' et 'attraction' introduit dans la section 2.4 seront présentés.

2.5.1 Dispositif d'expérimentation

Pour démontrer la faisabilité de l'attaque par migrations intempestives de VMs, nous utilisons une plate-forme de virtualisation VMware composée d'un cluster de 5 hôtes IBM 6550 M3 de 8 coeurs Xeon 2.133 GHs et 16 GB de mémoire chacun. Les versions logicielles suivantes sont utilisées : hyperviseur VMware ESXi 4.1 et logiciel de management de cloud vCenter 4.1 avec DRS activé en mode migration automatique, le processus de migration de la VM est exécuté sur plate-forme VMware par le module vMotion [77].

Les VMs sont déployées uniformément sur les hôtes du cluster pour garantir une égalité de contexte d'exécution pour les VMs, 10 VMs par hôte : 8 VMs 2 Gb 1 vCPU, 1 VM 1GB 1 vCPU et 1 VM 512 Mo 1vCPU. Les ressources de chaque hôte sont sur-engagées comme suit : 13% pour la mémoire et 25% pour le CPU.

Pour mener nos expérimentations, nous utilisons deux types d'outils développés en interne. Le premier est dédié à la génération de charge au sein des VMs, pour la mémoire d'une part et pour le CPU d'autre part, selon un profil paramétrable et possiblement fluctuant dans le temps obtenu par déploiement d'agents de consommation s'exécutant dans les VMs. Afin de créer un état initial de parfait équilibre, la charge est fixée à 75% pour toutes les VMs (mémoire et CPU), exceptées celles sous le contrôle de l'attaquant dont on fait varier la consommation en mémoire et en CPU lors de nos expérimentations.

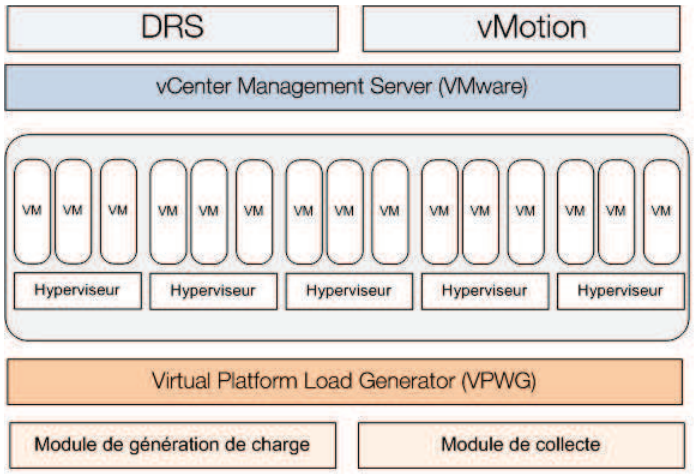


FIGURE 2.5 – Plate-forme d'expérimentation

Le second outil permet le suivi de la consommation de ressources des VMs et des hôtes toutes les 20 secondes, il permet aussi de suivre les statistiques de DRS calculées à chaque invocation de l'algorithme (*Dynamic Entitlement*, *chlsd* et *thlsd*) grâce aux interfaces de programmation fournies par VMware.

La figure 2.5 illustre les composants principaux de cette plate-forme d'expérimentation.

Au total, le point de fonctionnement de notre cluster d'expérimentation avec les valeurs de sur-engagement des ressources et de charge retenues est tel que DRS prend ses décisions de migration dans des délais courts, ce qui nous permet d'observer les effets de l'attaque immédiatement après son exécution évitant ainsi toute erreur d'interprétation de nos observations.

La section suivante détaille l'exécution de l'attaque de base, en déroulant dans le temps les effets résultant des profils de consommation mémoire et CPU injectés dans les VMs.

2.5.2 Attaque de base

Dans cette section, nous détaillons l'attaque par attraction de VMs décrite dans la section 2.4, telle que nous l'avons réalisée sur notre plate-forme d'expérimentation : 5 hôtes et DRS paramétré à un niveau d'aggressivité par défaut ('moderate'), ce qui fixe *thlsd* à une valeur de 0.126 comme indiqué précédemment sur la figure 2.4.

Nous illustrons ci-dessous le déroulement de l'attaque avec deux figures. Figure 2.6 donne la variation de la valeur de *chlsd* durant toute la durée de l'attaque ainsi que la valeur seuil *thlsd*. Figure 2.7 montre les effets de la manipulation par l'attaquant des quantités de ressources consommées par ses VMs, sur cette même durée. Plus précisément, figure 2.7(a) et figure 2.7(b) indiquent respectivement pour le CPU et pour la mémoire, la valeur de la métrique *Dynamic Entitlement* de la VM de l'attaquant telle que calculée par DRS à chaque invocation (toutes les 5

minutes, valeur par défaut) du fait de la consommation en ressources de la VM résultant de la charge produite avec notre outil. Figure 2.7(c) et figure 2.7(d) illustrent l'utilisation résultante des ressources CPU et mémoire de l'hôte sur lequel s'exécute la VM sous le contrôle de l'attaquant.

Pour éviter tout état initial qui serait favorable au déroulement de l'attaque, l'attaque est lancée dans un état initial de parfait équilibre du cluster (*chlsd* = 0) ; cela maximise le déséquilibre de charge qui doit être créé par l'attaque (applicable pour toutes les expérimentations présentées dans ce chapitre). Cet état de parfait équilibre est visible sur la figure 2.6 au niveau des 30 premières minutes de l'expérimentation : la valeur de chlsd est égale à zéro et résulte d'un niveau de charge fixé à 75% pour toutes les VMs, à la fois pour la mémoire et pour le CPU.

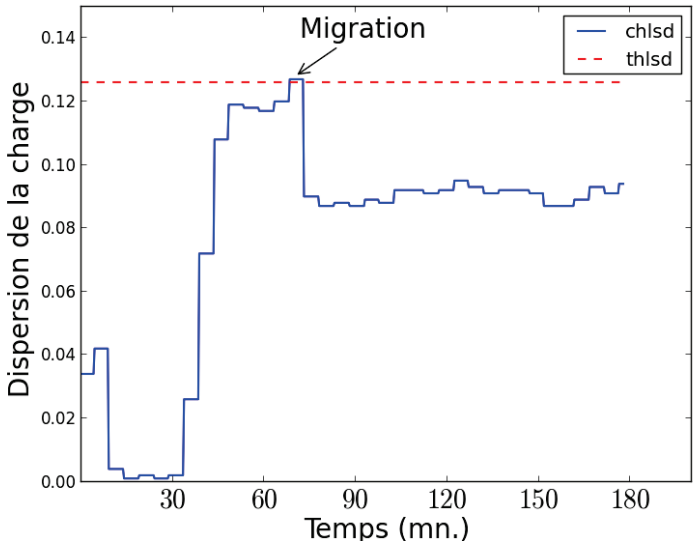
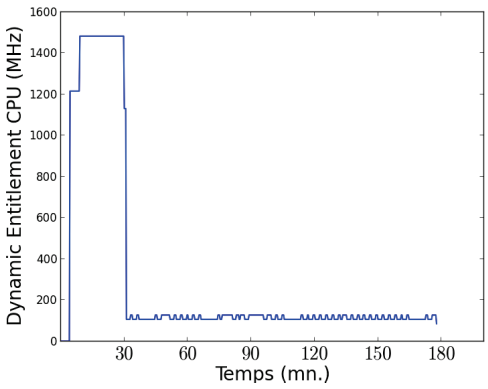


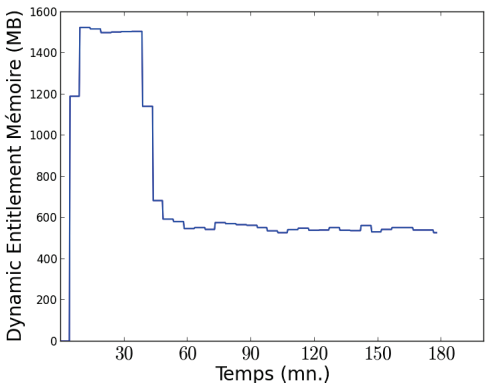
FIGURE 2.6 – Variation de *chlsd* au cours de l'attaque

L'attaque commence à T = 30 mn quand l'attaquant diminue brutalement la consommation de sa VM passant de 75% jusqu'à 5% de sa mémoire et de son CPU. Sur les figure 2.7(a) et figure 2.7(b), nous pouvons observer que cette diminution se répercute immédiatement sur les métriques *Dynamic Entitlement* mémoire et *Dynamic Entitlement* CPU de la VM de l'attaquant. L'effet du déséquilibre de charge résultant entre les hôtes est observable sous la forme d'une augmentation brutale de la valeur du *chlsd* jusqu'à atteindre *thlsd* aux environs de T = 70 mn sur la figure 2.6. A ce moment précis, une migration a lieu et nous pouvons observer sur la figure 2.7(c) et la figure 2.7(d) l'augmentation de la consommation de ressources au niveau de l'hôte qui a attiré une VM du fait de sa sous-charge occasionnée par la diminution de la consommation de la VM attaquante qui s'exécute sur cet hôte. On peut voir sur la figure 2.6 que la migration de VM opérée provoque le retour du cluster dans un état de charge équilibré ($chlsd \leq thlsd$).

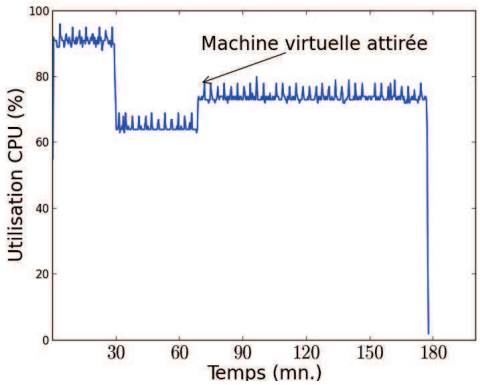
Avec cette expérimentation, nous démontrons et analysons qu'un attaquant peut par simple manipulation des quantités de ressources consommées par ses propres VMs (ou des VMs dont il



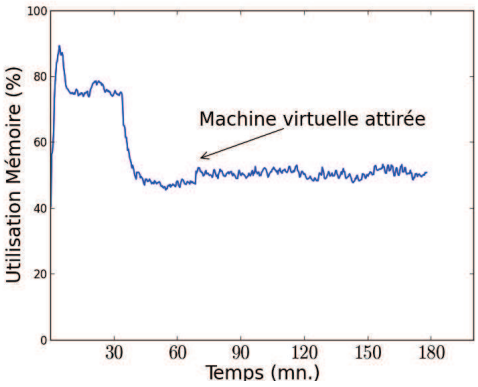
(a) *Dynamic Entitlement* mémoire machine virtuelle



(b) *Dynamic Entitlement* CPU machine virtuelle



(c) Utilisation CPU hôte



(d) Utilisation mémoire hôte

FIGURE 2.7 – Suivi de l'attaque de base au niveau machine virtuelle et hôte

a le contrôle) amener le cluster dans un état de déséquilibre de charge et amener le système de gestion dynamique de ressources à déclencher une migration de VM. Dans la section suivante, nous proposons une métrique de mesure du niveau d'exposition des clusters vis-à-vis de ce type de vulnérabilités.

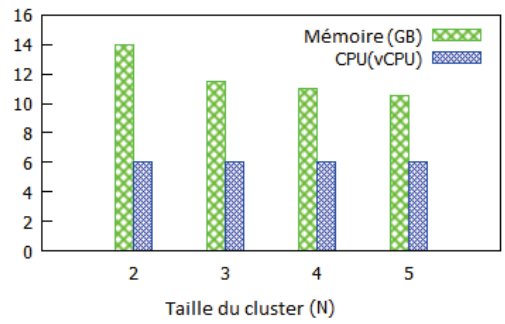
2.5.3 Exposition des clusters vis-à-vis de la vulnérabilité

La faisabilité de l'attaque décrite ci-dessus dépend de la quantité de ressources sous le contrôle de l'attaquant, considérée pour sa capacité à impacter la valeur de la métrique *Normalized Entitlement* de l'hôte hébergeant ses VMs, elle-même considérée pour sa capacité à impacter la dispersion des métriques *Normalized Entitlement* de l'ensemble des hôtes composant le cluster. En résumé, la capacité de l'attaquant à réaliser l'attaque dépend de sa capacité à faire augmenter la valeur du *chlsd* en relatif par rapport à *thlsd* ; cette dernière valeur dépend du nombre d'hôtes (N) composant le cluster et du niveau d'agressivité (Agr) de DRS comme déjà vu précédemment

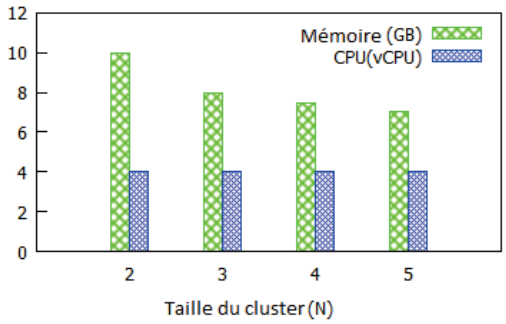
avec la formule de calcul (2.1) de *thlsd*.

Nous défendons l'idée que cette quantité de ressources nécessaire à la réalisation de l'attaque, constitue une mesure du niveau d'exposition des clusters contre ce type d'attaque. Plus cette quantité requise de ressources est faible, plus le cluster est exposé à la vulnérabilité considérée.

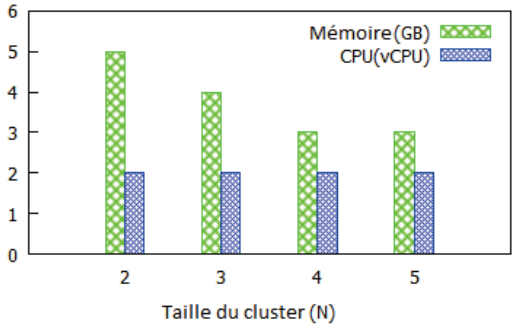
Pour quantifier cette vulnérabilité, nous avons mené une pluralité d'expérimentations pour mesurer - avec une précision de 512 Mo et 1 vCPU qui est la plus petite capacité de VM dont nous disposons dans nos expérimentations - la quantité minimum de ressources nécessaire à l'attaquant pour différentes tailles de cluster exprimées en nombre de hôtes (N) et pour différents niveaux dagressivité de DRS (Agr), qui sont des paramètres essentiels pour définir le point de fonctionnement des clusters. Nous rappelons que N et Agr, à l'exclusion de tout autre paramètre, impactent la valeur de *thlsd*. N impacte également *chlsd* ; les autres paramètres impactant le calcul de *chlsd* sont maintenus constants pour toutes nos expérimentations en configurant les générateurs de charge de toutes les VMs, exceptées ceux des VMs attaquantes, à une valeur unique et constante de sorte que ces autres paramètres n'ont pas d'influence sur l'interprétation de nos résultats.



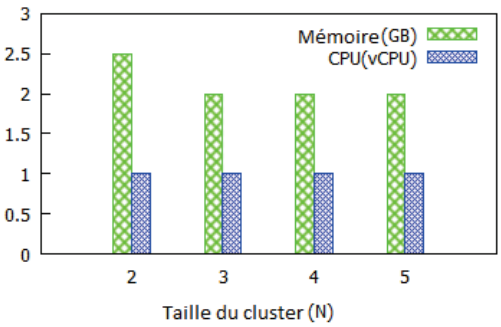
(a) Moderately Conservative



(b) Moderate



(c) Moderately Aggressive



(d) Aggressive

FIGURE 2.8 – Quantité de ressources minimale nécessaire à l'attaque, pour les différents niveaux d'agressivité de DRS

Figure 2.8(a), figure 2.8(b), figure 2.8(c) et figure 2.8(d) fournissent les quantités minimum de ressources mémoire et CPU nécessaires pour réaliser l'attaque, en fonction de la taille du cluster

et pour les quatre niveaux d'agressivité de DRS. Pour chaque expérimentation, nous mesurons ces quantités de mémoire et de vCPU. Pour une comparaison plus aisée des résultats, le nombre de vCPU est maintenu constant d'une expérimentation à l'autre pour un niveau d'agressivité de DRS donné.

Tout d'abord, nous observons que, quelque soit le niveau d'agressivité de DRS, la quantité minimum de ressources requise pour réaliser l'attaque, diminue quand la taille du cluster augmente.

Ce résultat peut paraître surprenant car on pourrait intuitivement penser que la quantité minimum de ressources requise pour réaliser l'attaque doit augmenter quand la taille du cluster augmente du fait que la capacité d'influence de l'attaquant, en relatif par rapport à la capacité du cluster, sur la valeur de dispersion des charges des hôtes du cluster (*chlsd*) diminue. La raison pour laquelle l'attaquant peut disposer de moins de ressources pour réaliser l'attaque quand la taille du cluster augmente est liée aux valeurs du seuil de déclenchement des migrations *thlsd* ; la valeur de *thlsd* diminue également quand la taille du cluster augmente et ce suffisamment rapidement - en particulier dans le cas des clusters de petite taille comme dans nos expérimentations (2 à 5 hôtes) - pour que le *chlsd* créé par l'attaquant continue de dépasser la valeur du *thlsd* alors que la capacité de l'attaquant à faire augmenter la valeur de *chlsd* diminue effectivement quand la taille du cluster augmente.

Une deuxième représentation des quantités minimum de ressources requises pour l'attaque est illustrée sur la figure 2.9. Sur cette figure, nous représentons la variation de la quantité minimum de mémoire requise pour l'attaque en fonction du niveau d'agressivité de DRS (le CPU, non représenté, est constant pour une même taille de cluster et décroit d'un niveau d'agressivité au niveau d'agressivité supérieur et n'a pas d'impact sur l'interprétation des résultats), pour les quatre tailles de clusters dont nous disposons (variant de 2 à 5 hôtes).

Nous pouvons, grâce à cette figure, faire l'observation suivante : pour une taille de cluster donnée, la quantité minimum de ressources requise pour réaliser l'attaque, décroît quand le niveau d'agressivité augmente. A titre d'exemple, pour un cluster de 2 hôtes, la quantité de mémoire requise décroît de 14 GB à 2.5 GB et le nombre de vCPU de 6 vCPU à 2 vCPU, quand le niveau d'agressivité passe de 'moderately conservative' (ModCons) à 'agressive' (Ag).

Là encore, ce résultat s'explique par les valeurs de *thlsd* ; la valeur de *thlsd* est d'autant plus faible que le niveau d'agressivité de DRS est élevé, pour une taille de cluster donnée.

Nous pouvons également observer que les petits clusters sont plus sensibles au niveau d'agressivité de DRS en comparaison avec les grands clusters. Sur la figure 2.9, il est mis en évidence que la différence entre la quantité minimum de mémoire nécessaire au succès de l'attaque varie plus rapidement avec le niveau d'agressivité de DRS pour les clusters composés de 2 hôtes que pour les clusters composés de 5 hôtes.

Dans cette section, nous avons analysé l'algorithme DRS avec une attention particulière pour

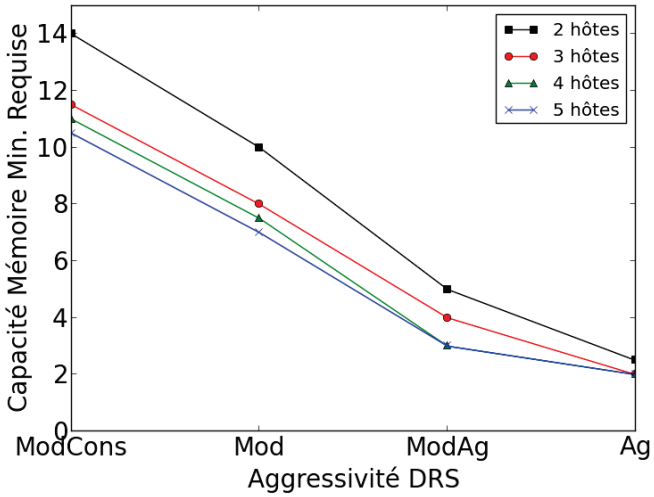


FIGURE 2.9 – Variation de la quantité de ressources minimale nécessaire en fonction de l'agressivité de DRS

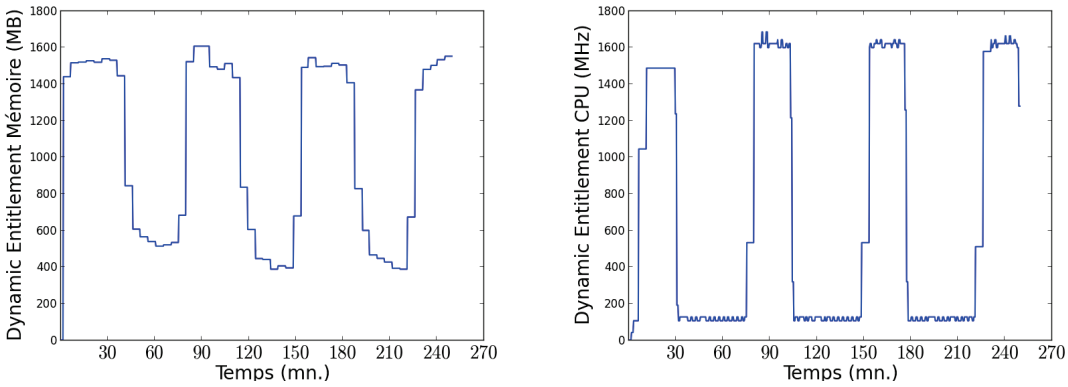
les valeurs de *thlsd*, pour interpréter nos mesures du niveau d'exposition des clusters vis-à-vis de la vulnérabilité exploitée par l'attaque par migrations intempestives de VMs. Dans la section suivante, nous montrons qu'un attaquant peut amener DRS à déclencher une série de migrations successives sous certaines conditions, en reproduisant successivement le schéma de base de l'attaque.

2.5.4 Attaque coordonnée

Dans cette section, nous démontrons qu'un attaquant peut amener DRS à déclencher non seulement une migration mais une série de migrations successives, en coordonnant l'exécution de profils de consommation à effet d'attraction de VMs et celle de profils à effet de repoussement de VMs, tous les deux introduits dans la section 2.4.

La capacité d'un attaquant à influencer le déséquilibre de charge au sein d'un cluster est plus importante s'il peut contrôler les quantités de ressources consommées par des VMs hébergées sur deux hôtes différents. Il s'agit de faire se coordonner les VMs pour que leurs consommations de ressources fluctuent de manière cyclique et en opposition de phase entre le premier et le deuxième hôte : quand les VMs s'exécutant sur le premier hôte consomment de faibles quantités de ressources, celles s'exécutant sur le second hôte consomment des quantités de ressources élevées et vice versa. De cette manière, l'attaquant forge des écarts de charges significatifs et à répétition et, de ce fait, crée des opportunités successives de migration de VMs amenant DRS à rétablir l'équilibre de charge entre les hôtes successivement en sous-charge (attraction de VMs) et en sur-charge (repoussement de VMs).

Pour réaliser cette attaque, les générateurs de charge sont configurés de telle sorte que les consommations des VMs de l'attaquant fluctuent toutes les 40 minutes entre deux valeurs extrêmes, 5 % et 75% de leur CPU et mémoire configurés. Dans cette expérimentation, le cluster est composé de 4 hôtes et DRS reste paramétré selon le niveau d'agressivité par défaut, ce qui fait prendre à *thlsd* la valeur de 0.141. Les autres conditions d'expérimentation restent inchangées par rapport aux cas précédents.



(a) *Dynamic Entitlement* mémoire machine virtuelle (b) *Dynamic Entitlement* CPU machine virtuelle

FIGURE 2.10 – Suivi de l'attaque coordonnée au niveau machine virtuelle

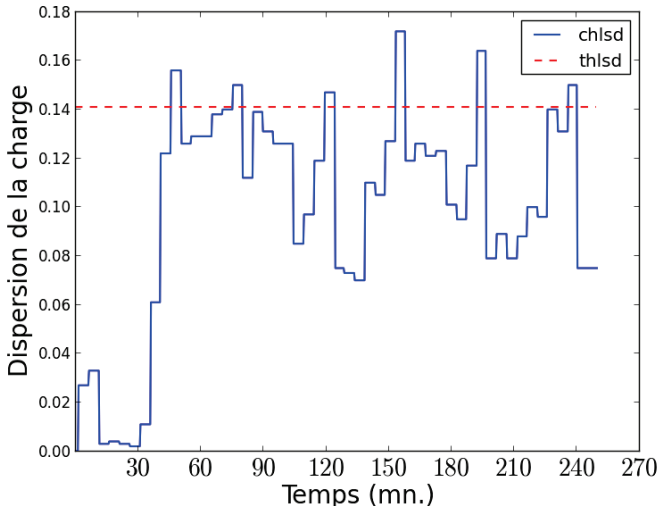


FIGURE 2.11 – Variation de *chlsd* au cours de l'attaque

Figure 2.10(a) et figure 2.10(b) illustrent la variation de *Dynamic Entitlement* mémoire et CPU d'une des VMs de l'attaquant, pendant toute la durée de l'attaque. La variation résultante de *chlsd* sur cette même période est donnée figure 2.11.

Le scenario s'exécute de manière successive avec la même logique que celle décrite précédemment dans la section 2.5.2. Pendant cette expérimentation, nous avons observé six migrations de VMs. Sur la figure 2.11, on peut voir que ces migrations obtenues aux points de dépassement du *thlsd* par le *chlsd* sont effectivement synchronisées dans le temps avec les fluctuations des profils de consommation des VMs de l'attaquant, ce qui ne laisse aucun doute sur l'interprétation que nous faisons de nos observations pour conclure à un rapport de cause à effet entre la fluctuation des profils de consommation des VMs attaquantes et l'occurrence de migrations nécessairement induites par cette fluctuation.

Nous avons exécuté plusieurs fois l'attaque et avons observé que la plupart du temps, DRS migre des VMs autres que celles de l'attaquant. Comme abordé précédemment dans l'analyse de l'algorithme DRS en section 2.3, cela est dû au fait que DRS considère que les VMs fluctuantes sont moins prédictibles quant à leur niveau futur de consommation et risquent donc d'être moins aptes à solutionner de manière durable, de par leur migration, le déséquilibre du cluster. Cela donne deux avantages à l'attaquant. D'une part, les VMs de l'attaquant restent sur le même hôte et conservent ainsi leur capacité d'attaque et d'autre part, les VMs qui pâtissent des effets négatifs sur les performances du fait de la migration de VMs, sont celles co-localisées avec les VMs de l'attaquant.

2.6 Conclusion

Dans ce chapitre, nous avons présenté en détails l'attaque par migrations intempestives de VM qui est, à notre connaissance, la première attaque rapportée contre les systèmes de gestion dynamique des ressources. Après avoir étudié les principes de plusieurs systèmes de gestion dynamique de ressources existant dans l'état de l'art et avoir analysé qu'ils prennent leur décisions en fonction des quantités de ressources consommées par les VMs, nous avons examiné plus particulièrement l'algorithme DRS pour démontrer la faisabilité de l'attaque par migrations intempestives de VMs et nous permettre de comprendre les raisons pour lesquelles cette attaque est réalisable. Enfin, nous avons proposé un moyen de mesurer le niveau d'exposition des clusters vis-à-vis de ce type de vulnérabilités.

Cette étude démontre que les systèmes de gestion dynamique des ressources qui omettent de prendre en compte les aspects sécurité peuvent être vulnérables à la manipulation malveillante des quantités de ressources consommées par des VMs. Nous pensons que la sécurité des systèmes de gestion dynamique des ressources est un nouveau domaine de recherche insuffisamment exploré, qui mérite davantage d'attention de la part de la communauté scientifique.

Les conclusions issues de nos résultats concernant la sécurité des systèmes de gestion dynamique des ressources viennent appuyer la nécessité de concevoir des systèmes plus robustes. La robustesse pour un système de gestion dynamique des ressources consiste à développer une

'résistance' aux variations de consommation en ressources des VMs tout en cherchant à minimiser l'occurrence de violations des engagements en qualité de service passés entre l'opérateur et ses clients.

Pour répondre à ce besoin de robustesse pour ce type de système, il apparaît naturel d'allouer aux VMs des quantités de ressources proches de leurs besoins maximums (incluant les pics de consommation), ce qui malheureusement tend à réduire les taux de consolidation des hôtes. Trouver le bon compromis entre la robustesse et la réactivité est par conséquent crucial pour une gestion de ressources efficace, sachant que plus un système est réactif, plus il optimise l'exploitation des ressources d'un centre de données, mais plus il devient vulnérable. Par opposition, plus un système est robuste, plus il est résistant face à la vulnérabilité présentée dans ce chapitre mais cela se fait au dépend de la consolidation des hôtes d'un centre de données.

Dans le chapitre suivant de cette thèse, nous étudions une voie complémentaire à une forme de robustesse intrinsèque des algorithmes de gestion dynamique des ressources, celle de la mitigation réactive de ce type de vulnérabilités.

Nous adoptons une approche basée sur une supervision du fonctionnement du système de gestion dynamique de ressources, dont la finalité est de permettre le déclenchement d'une suite de traitements, agissant grâce à une boucle de rétroaction sur les décisions du système de gestion des ressources, dès lors que les décisions de gestion des ressources observées révèlent une anomalie de fonctionnement du système.

Travaux de l'auteur sur cette thématique

— When Dynamic VM Migration Falls Under the Control of VM Users. **Kahina Lazri**, Sylvie Laniepce, Jalel Ben-Othman, 5th IEEE International Conference on Cloud Computing Technology and Science **CloudCom'13**. Décembre 2013, Bristol, Royaume-Uni.

— Sécurité de la gestion dynamique des ressources dans le cloud : prise de contrôle sur le déclenchement de migrations automatiques de machines virtuelles. **Kahina Lazri**, Sylvie Lanièpce, Haiming Zheng, Jalel Ben-Othman, Symposium sur la Sécurité des Technologies de l'Information et des Communications **SSTIC'14**. Juin 2014, Rennes, France.

— Procédé de détection d'attaques. **Kahina Lazri**, Sylvie Lanièpce, N° dépôt : 13 51866, déposé le 01/03/13 (N° référence extension PCT : FR2014/050458).

Chapitre 3

Supervision de la sécurité d'un système de gestion dynamique de ressources

Dans ce chapitre, nous présentons AMAD – Abusive VM Migration Attack Detection – un système de supervision des décisions de migration de VMs d'un système de gestion dynamique des ressources, qui permet de détecter l'occurrence de l'attaque par migrations intempestives de VMs. AMAD analyse les profils de consommation en ressources des VMs du cluster afin d'identifier les VMs fluctuantes et coordonnées qui sont considérées comme étant celles à l'origine de l'attaque. Nous avons implémenté AMAD sur une plate-forme d'expérimentation et nous avons évalué ses performances de détection sur notre plate-forme d'une part et hors ligne sous des conditions d'exécution représentatives de contextes clouds réels d'autre part. Nos résultats d'évaluation montrent qu'AMAD s'exécute avec une bonne performance de détection pour identifier les VMs attaquantes que nous avons intentionnellement injectées dans nos expérimentations.

Sommaire

3.1 Introduction

DANS le chapitre 2, nous avons démontré l'attaque par migrations intempestives de VMs dans laquelle un attaquant influence le système de gestion dynamique des ressources pour l'amener de manière abusive à déclencher des migrations de VMs. Nous avons démontré que cette attaque est réalisée grâce à une simple manipulation des quantités de ressources consommées par des VMs appartenant à un attaquant (ou des VMs tierces sous son contrôle).

La faisabilité de ce type d'attaque impose d'améliorer la robustesse des systèmes de gestion dynamique des ressources vis-à-vis de la variation des quantités de ressources consommées par les VMs. La robustesse est une propriété nécessaire pour un système de gestion dynamique des ressources mais celle-ci est par définition antinomique avec l'optimisation de la consolidation des serveurs : trouver le bon compromis entre la robustesse et la réactivité reste un verrou important pour la conception des systèmes de gestion dynamique des ressources.

Dans cette thèse, nous proposons une approche réactive pour assurer la protection des infrastructures cloud de l'attaque par migrations intempestives de VMs, qui consiste à superviser les décisions de migrations de VMs prises par le système de gestion dynamique des ressources. L'avantage principal de cette approche est d'intégrer dans la boucle décisionnelle de l'algorithme de gestion dynamique des ressources la complexité de contraintes additionnelles liées à la sécurité, uniquement si un événement de sécurité telle que la détection d'une attaque venait à se produire. Mettre en place un tel mécanisme de supervision permet de surveiller le comportement du système pour identifier de potentielles anomalies d'une part et d'être en mesure de mettre en place des moyens correctifs pour réagir aux situations où le système dévie de son fonctionnement attendu d'autre part.

Partant de cette motivation, nous proposons AMAD – Abusive VM Migration Attack Detection – un système qui permet de détecter l'occurrence de l'attaque par migrations intempestives de VMs – caractérisée par l'exécution d'un nombre élevé de migrations de VMs – et d'identifier de manière automatique l'ensemble des VMs pouvant être à l'origine de l'attaque. Pour identifier les VMs attaquantes, AMAD se base sur l'analyse des profils de consommation en ressources des VMs s'exécutant au sein du cluster supervisé en adoptant une approche d'analyse en boite noire, c'est à dire basée uniquement sur des métriques de consommation en ressources des VMs disponibles au niveau de l'hyperviseur écartant ainsi toute dépendance aux VMs.

Pour que l'attaque par migrations intempestives de VMs produise des effets suffisamment préjudiciables au cluster – relevant effectivement d'effets pouvant être assimilés à une attaque – l'attaquant doit manipuler plusieurs VMs pour disposer d'une capacité en ressources suffisante pour influencer significativement, grâce à la fluctuation de la consommation de ces VMs, la répartition de la charge au sein du cluster de manière à déclencher des migrations (4 GB de mémoire et 4vCPU dans le contexte de l'attaque démontrée dans la section 2.5.2). Partant de ce

constat, AMAD met en oeuvre des mécanismes d'analyse de la consommation en ressources des VMs permettant de caractériser et d'identifier les VMs à profils de consommation fluctuants et temporellement coordonnées, qui sont considérées comme étant celles à l'origine de l'attaque.

Contributions du chapitre

Dans ce chapitre, nous détaillons les contributions suivantes :

— Proposition d'AMAD, un système qui permet de détecter l'occurrence d'une attaque par migrations intempestives de VMs et d'identifier automatiquement l'ensemble des VMs fluctuantes et coordonnées considérées comme étant à l'origine de l'attaque,

— Proposition d'un algorithme d'analyse de la consommation en ressources de VMs qui peut être utilisé indépendamment d'AMAD pour une identification automatique des VMs qui présentent un profil de consommation en ressources fluctuant,

— Implémentation d'AMAD et évaluation intensive de sa performance de détection à travers des expérimentations menées sur notre plate-forme de laboratoire,

— Évaluation des algorithmes d'AMAD sur le plan de leur performance et de leur robustesse de détection, au travers d'expérimentations menées sur plate-forme de laboratoire d'une part et hors ligne dans des conditions d'exécution issues de contextes clouds réels d'autre part.

Organisation du chapitre

Ce chapitre est organisé comme suit. La section 3.2 livre une analyse de l'état de l'art associé à la détection des anomalies d'exécution dans le cloud. La section 3.3 introduit l'architecture globale d'AMAD et détaille chacun de ses modules. La section 3.4 livre des éléments sur l'implémentation d'AMAD et évalue sa performance de détection avec des expérimentations menées sur notre plate-forme de laboratoire. La section 3.5 évalue la performance de détection de l'algorithme d'identification des VMs fluctuantes dans des conditions d'exécution de clouds réels. La section 3.6 évalue la robustesse de l'algorithme d'identification des VMs fluctuantes à l'aide de variantes de profils de consommation de VMs attaquantes puis lorsque l'algorithme s'exécute dans un contexte à niveau de fluctuation élevé. Enfin, la section 3.7 conclut ce chapitre.

3.2 État de l'art

Toute VM démontrant un profil de consommation en ressources extrêmement fluctuant n'est pas nécessairement une VM malicieuse. Or, il est difficile de distinguer, en particulier depuis l'infrastructure, une fluctuation due à une activité normale de la VM d'une fluctuation malicieuse

qui a pour objectif d'affecter négativement l'infrastructure et les performances d'exécution des VMs comme c'est le cas pour l'attaque par migrations intempestives de VMs. C'est la raison pour laquelle il s'agit d'identifier en première ligne de défense ce type de VMs à profil fluctuant pour chercher ensuite à étudier leur capacité à générer des impacts négatifs, sans les considérer a priori comme étant des VMs nécessairement malicieuses. Cela nous amène à examiner l'état de l'art de la détection d'anomalie plutôt que de se limiter à la détection d'activité malicieuse.

L'effet d'une anomalie d'exécution des VMs dans le cloud se traduit souvent par une dégradation de performances quantifiable grâce à des métriques de mesure de la performance des applications qui s'exécutent dans les VMs (telles que la latence, le débit, le nombre d'instructions exécutées par seconde). Ces dégradations de performances peuvent avoir pour cause, i) une faute interne à la VM, telle qu'une montée en charge, une panne logicielle, une attaque, etc... ou, ii) une faute de niveau infrastructure induite par la co-résidence de VMs, telle que l'interférence de ressources ou la contention de ressources qui active des mécanismes de gestion de la contention (compression de pages mémoire, ballooning ou encore double pagination de pages mémoire [20]).

Pour traiter de ces anomalies, deux catégories principales de système de détection et d'analyse des anomalies co-existent. La première catégorie d'approche considère uniquement l'espace de ressources des VMs ayant observé l'anomalie, pour identifier la cause de l'anomalie. Ces approches centrées sur la VM et détaillées en section 3.2.1, traitent principalement de quatre types de faute : fuite mémoire qui consiste en une occupation croissante et non maîtrisée de la mémoire, épuisement de la ressource processeur, saturation de la bande passante réseau et saturation des entrées/sorties disque.

La seconde catégorie comprend les approches qui prennent en considération la propriété de partage des ressources entre les VMs et qui considèrent que les effets d'une anomalie observés au niveau d'une VM peuvent résulter d'une faute externe à cette VM. Ce deuxième type d'approche détaillé en section 3.2.2, analyse une pluralité de VMs qui partagent un espace commun de ressources sans se limiter aux VMs ayant subi les effets de la faute et prend en compte les services de l'infrastructure, ce qui rend ces approches aptes à traiter non seulement les fautes internes aux VMs mais aussi celles liées au contexte d'exécution.

Le tableau 3.1 donné en fin de section récapitule l'ensemble des systèmes présentés ci-dessous, en décrivant pour chacun d'eux le niveau de mesure des métriques considérées (application, VM, hyperviseur) et le type de faute détectable par le système.

3.2.1 Traitement centrée VM

PAL (Propagation-aware Anomaly Localization) [98] traite de la détection d'anomalies au sein de VMs impliquées dans une exécution en mode distribué d'applications et cherche à identifier toute forme de dégradation de performances à l'origine d'une violation de SLAs.

Pour identifier une dégradation de performances qui se produit au sein d'une VM, PAL se base sur un outil intra-VM configuré pour lever une alerte vers l'infrastructure lorsque les performances de la VM diminuent au-dessous d'un seuil prédéfini. PAL prend pour hypothèse que les effets d'une anomalie sont à l'origine de points de changement observables au niveau des métriques de mesure de la consommation en ressources des VMs et que ces effets se propagent sur les VMs qui s'exécutent en mode distribué. Suite à la réception de l'alerte, PAL exécute un algorithme d'analyse pour détecter tout point de changement anormal dans l'utilisation de ressources par les VMs s'exécutant en mode distribué, puis trie ces points de changement selon leur ordre chronologique d'apparition et considère la VM ayant observé le point de changement en premier sur l'échelle de temps comme étant celle à l'origine de l'anomalie.

PAL est en mesure de traiter les anomalies suivantes : fuite mémoire, consommation intensive du processeur, accès intensifs au disque et changement significatif de la charge de VMs (augmentation du nombre de requêtes des utilisateurs d'un serveur web par exemple). PAL analyse des métriques de mesure de la consommation en ressources des VMs sans prendre en considération l'utilisation des ressources des VMs co-résidentes.

PeerWatch [99] cherche à détecter des anomalies se produisant au sein de VMs qui s'exécutent en mode distribué, en modélisant le niveau de corrélation entre les consommations en ressources des différentes VMs, mesurées grâce à des outils intra-VM, et considère qu'il y a anomalie d'exécution dès lors que cette corrélation diminue en dessous d'un seuil prédéterminé.

PeerWatch procède en deux phases. La première phase de modélisation et d'apprentissage a pour objectif de définir le modèle de corrélation entre les instances d'applications qui s'exécutent dans les différentes VMs, pour établir les seuils de corrélation entre les VMs. Une seconde phase de supervision, lors de laquelle toute baisse de corrélation entre les VMs au dessous des seuils définis durant la première phase est considérée comme une anomalie.

PeerWatch analyse les valeurs de consommation en ressources collectées depuis la VM et tire avantage de la vue multi-VMs pour déterminer la source d'une anomalie qui se propage au sein du groupe de VMs. Contrairement à PAL qui adopte une approche temporelle, PeerWatch identifie les couples de VMs démontrant les plus fortes baisses de valeur de corrélation et considère la VM impliquée dans le plus de couples comme étant celle à l'origine de l'anomalie.

Fchain [100] traite également de la détection et de la localisation d'anomalies à l'origine de dégradations de performances au sein de VMs s'exécutant en mode distribué. Fchain adopte une approche décentralisée. Des modules esclaves installés dans chacune des VMs privilégiées (dom 0) des différents hôtes, mesurent et construisent par apprentissage le modèle de fluctuation de la consommation en ressources des VMs. Des modules maîtres s'exécutant sur des serveurs dédiés, sont chargés de l'identification des causes des anomalies. A la réception d'une alerte signalant l'occurrence d'une violation de SLA pour une VM, Fchain contacte les modules esclaves pour vérifier si les VMs s'exécutant en mode distribué avec la VM ayant émis l'alerte,

ont observé des points de changement suspicieux et récupérer leur temps d'occurrence. Fchain identifie la VM à l'origine de l'anomalie en caractérisant d'abord la relation de dépendance des flux d'exécution des VMs grâce à une analyse des échanges réseaux entre les VMs qui s'exécutent en distribué [101] pour reconstruire la chaîne d'inter-dépendance d'exécution entre les composants. Après reconstitution de chaque chaîne, Fchain trie les points de changement identifiés selon leur ordre d'apparition dans la chaîne d'inter-dépendance et considère la VM ayant observé le point de changement en premier comme celle à l'origine de l'anomalie.

Fchain utilise des métriques de consommation des VMs collectées au niveau de l'hyperviseur sans aucune considération pour la consommation des VMs co-résidentes avec le groupe de VMs auquel appartient la VM ayant levé l'alerte.

PREPARE [102] implémente une approche de détection proactive de la dégradation de performances de VMs. Pour cela, il utilise une modélisation à l'aide de chaînes de Markov [103] pour prédire les valeurs à venir des métriques de mesure des performances à partir des valeurs courantes de ces métriques. PREPARE utilise une classification des anomalies à l'aide de réseaux Bayésiens [104] pour filtrer les valeurs représentatives d'un état normal de celles qui révèlent l'existence d'une anomalie. PREPARE collecte la consommation en ressources des VMs au niveau de l'hyperviseur, à l'exception de l'utilisation mémoire qu'il récupère depuis la VMs grâce aux outils du système d'exploitation de la VM. PREPARE ne prend pas en considération les consommations en ressources des VMs co-résidentes avec les VMs ayant observé l'anomalie, pour identifier la source de l'anomalie.

Les systèmes de détection d'anomalies de performances des VMs sont souvent dépendants de métriques disponibles au niveau de la VM, en particulier les métriques de mesure des performances des applications qui s'exécutent dans les VMs, d'où la difficulté pour ces systèmes d'adopter des approches de détection et d'analyse purement en boite noire et donc d'opérer au niveau infrastructure seul sans vue intra-VM.

Les systèmes que nous venons de décrire cherchent à tirer avantage de la vue multi-VMs dont bénéficie l'opérateur ou un utilisateur qui dispose d'une interface de management sur des VMs s'exécutant en mode distribué, pour retrouver la VM à l'origine de l'anomalie. Ces systèmes ne sont pas en mesure d'identifier les causes d'anomalies qui relèvent de l'infrastructure.

En conclusion, ce type d'approche n'est pas en mesure de détecter des attaques de type migrations intempestives de VMs puisqu'il ne considère pas la migration de VMs comme une potentielle cause aux dégradations de performances dont peuvent souffrir les VMs.

3.2.2 Traitement hybride : VM / infrastructure

L'hétérogénéité et la dynamicité qui caractérisent l'exécution des VMs dans le cloud augmentent la surface d'exposition de ces infrastructures vis-à-vis de failles logicielles, matérielles

et attaques. Aux anomalies internes aux VMs, s'ajoutent des anomalies d'exécution possibles au niveau des services offerts par l'opérateur, qui peuvent impacter les VMs.

On distingue deux sources principales d'anomalies ayant pour origine l'infrastructure : i) la première est introduite par le partage des ressources matérielles et résulte en une problématique de possibles interférences et/ou contentions des ressources : mémoire, cpu, réseau et disque, ii) la seconde est introduite par les services fournis par l'opérateur tels que les services de gestion de ressources (élasticité verticale et horizontale, migration de VMs).

La détection des problèmes de performances dus aux interférences de ressources et à leurs contentions de niveau hôte, se fait en modélisant la relation entre les quantités de ressources allouées aux VMs et les performances applicatives de ces VMs [105]. Dans ce type de modélisation, le système de détection prend en considération le contexte d'exécution des VMs, ce qui permet d'identifier des causes d'anomalie de niveau infrastructure. Ces approches sont souvent composées de deux étapes. Une première étape de construction du modèle de performances de référence, qui peut se faire en environnement non virtualisé pour garantir l'absence de bruit. Une seconde phase de supervision durant laquelle les performances des VMs sont confrontées au modèle de référence et tout écart vis-à-vis de celui-ci est considéré comme anormal.

DAPA [106] est un système d'analyse qui cherche à identifier des causes d'anomalies qui affectent les performances de VMs s'exécutant selon une architecture trois tiers. DAPA est constitué de trois modules : collecte, modélisation et analyse. Pour la mesure des performances des applications s'exécutant dans les VMs, DAPA utilise un outil intra-VM configuré pour lever deux type d'alerte : la première se déclenche suite à la détection de symptômes qui prédisent l'occurrence de violation de SLAs et la seconde se déclenche suite à l'occurrence effective de la violation de SLAs. La première alerte agit comme un déclencheur de collecte de la consommation en ressources des VMs et de modélisation. Suite à la réception de la première alerte, le module de modélisation de DAPA construit pour chacune des VMs, les modèles de corrélation entre les performances des applications, différentes mesures de l'utilisation en ressources des VMs collectées depuis l'hyperviseur et des mesures de niveau infrastructure telles que des valeurs de *ballooning*. Un module d'analyse applique ensuite l'algorithme de partitionnement k-moyennes [107] sur ces modèles pour identifier les valeurs atypiques de métriques de mesure de la consommation en ressources, qui sont considérées à l'origine de la dégradation de performances. La migration de VMs n'est pas considérée par DAPA comme étant une cause potentielle de dégradation de performances des VMs.

vPerfGuard [108] cherche à identifier automatiquement les valeurs des métriques à l'origine d'une violation de SLAs provoquée par une contention de ressources au sein de l'hôte exécutant la VM affectée. vPerfGuard collecte des métriques de mesure de la performance des applications s'exécutant dans les VMs de l'hôte, des métriques de mesure de la consommation en ressources des VMs qu'il collecte depuis les VMs et depuis l'hyperviseur. vPerfGuard s'exécute selon deux

étapes. Une première étape de réduction du nombre de métriques nécessaires à la construction du modèle de performances des VMs dans laquelle les métriques redondantes sont supprimées (par exemple, utilisation CPU collectée depuis l'hyperviseur et celle collectée depuis la VM sont similaires) d'une part et de sélection des combinaisons de métriques (performance, système) les plus pertinentes pour la construction du modèle de performance des VMs d'autre part. Une seconde étape qui a pour objectif de détecter le point de changement à partir duquel le modèle de performance construit en première phase ne parvient plus à modéliser la relation entre les performances des applications et les consommations en ressources des VMs, et qui de ce fait renseigne sur l'occurrence d'une anomalie. vPerfGuard considère les contentions de ressources provoquées par le contexte d'exécution des VMs mais ne prend pas en considération l'occurrence de migrations de VMs dans son processus d'analyse.

Q-clouds [105] est un système qui traite des dégradations de performances provoquées par les interférences entre les exécutions respectives des VMs de l'hôte. Durant l'exécution des VMs, Q-clouds construit les modèles de performances des VMs en temps réel avec la modélisation 'entrées multiples, sorties multiples' (MIMO, de l'anglais Multi-Input Multi-Output), les entrées étant les quantités de ressources allouées aux VMs, collectées au niveau de l'hyperviseur, et les sorties étant les performances applicatives des VMs mesurées périodiquement en intra-VM. Q-clouds considère que les VMs souffrent d'interférences dès que les performances courantes des VMs ne sont pas conformes au modèle construit. Pour pouvoir réagir aux dégradations de performances provoquées par les interférences, Q-clouds réserve au niveau de chaque hôte une quantité de ressources, dite de 'secours', qui peut être allouée aux VMs observant une dégradation de performances. L'intérêt de cette approche est de n'allouer des ressources supplémentaires aux VMs que si elles permettent d'améliorer leurs performances.

Dans [109], les auteurs proposent un système de détection d'anomalies au niveau infrastructure cloud. Les anomalies sont classifiées en fonction de leurs effets sur les ressources matérielles : mémoire, processeur, réseau et disque. Le système collecte plus de 800 métriques de mesure d'utilisation et de performance au niveau infrastructure et au niveau VM, PCA (Principal Component Analysis) [110] est ensuite appliqué sur ces mesures pour réduire leurs dimensions. Les *filtres de Kalman* [111] sont utilisés pour prédire les mesures à venir. Si la différence entre les mesures réelles et celles prédites dépasse un seuil préalablement défini, une alerte est alors générée.

MELA [112] est un système de supervision du fonctionnement des systèmes de gestion élastique des ressources dans le cloud qui s'intéresse à détecter des corrélations entre les décisions prises par le système de gestion élastique de ressources et des violations des propriétés SLAs pré-définies par les utilisateurs. MELA mesure la capacité et la réactivité du système à satisfaire les besoins des utilisateurs et identifie de façon automatique l'ensemble des valeurs de métriques pouvant être à l'origine de l'anomalie détectée.

Cloud PD [113] est un système qui implémente une approche de bout-en-bout incluant la détection, l'analyse, la classification et la réaction à des anomalies d'exécution observées par des VMs dans le cloud. Après avoir mis en évidence la difficulté d'appliquer les approches basées sur les seuils dans les environnements cloud du fait de la forte dynamicité des VMs, les auteurs proposent cloud PD qui implémente différents algorithmes d'apprentissage et de corrélation pour la détection des anomalies et pour leur classification. Cloud PD traite des deux catégories d'anomalie discutées ci-dessus i) celles spécifiques aux infrastructures cloud : contention due au partage de ressources, erreur d'estimation des quantités de ressources à allouer à une VM, défaut de migration de VM, ii) les anomalies non spécifiques au cloud : erreur de configuration d'application, variation conséquente de la charge d'une VM, faille logicielle. Pour la détection des anomalies, Cloud PD implémente trois techniques de modélisation comportementale : modèle de markov caché (HMM, de l'anglais Hidden Markov Models), modèle des k plus proches voisins (kNN, de l'anglais nearest-neighbor) et le modèle des k-moyennes. Pour la phase d'identification des causes des anomalies, Cloud PD implémente une approche par signatures.

Cloud PD est à notre connaissance le seul système de l'état de l'art qui intègre la migration de VMs comme cause possible des dégradations de performances des VMs. Cloud PD considère que la migration de VM constitue une faute dès lors que celle-ci est à l'origine d'une dégradation de performances significative pour la VM (corrélation entre augmentation de la latence et temps d'occurrence de migration de VM par exemple). Cependant, Cloud PD considère chaque migration de VM comme un événement isolé et ne considère pas l'existence de causes communes à des occurrences multiples de migrations de VMs, telles que l'attaque par migrations intempestives de VMs.

AMAD est un système qui permet de superviser le service cloud de migration de VM opéré par le système de gestion dynamique des ressources, pour détecter l'occurrence d'anomalies. Le mécanisme de migration automatique de VMs s'exécute sous la responsabilité du fournisseur de l'infrastructure uniquement, ce qui rend l'opérateur responsable d'assurer le non détournement de ce système par des attaquants à des fins malveillantes pouvant impacter négativement les performances d'exécution des VMs hébergées. Selon notre connaissance de l'état de l'art, AMAD est le premier système à superviser l'usage de la migration des VMs par les systèmes de gestion dynamique des ressources pour en garantir le bon fonctionnement.

La section suivante livre les détails de conception du système AMAD.

Système	Niveau de mesure	Types de faute
Pal	VM, hyperviseur	Fuite mémoire Épuisement cpu Saturation bande passante réseau Variation significative de la charge
PeerWatch	VM	fuite mémoire Épuisement cpu Saturation bande passante réseau Saturation des accès disque
Fchain	application, hyperviseur	Fuite mémoire Contention CPU Saturation bande passante réseau Saturation des accès disque
PREPARE	VM, hyperviseur	Fuite mémoire Épuisement cpu Augmentation significative de la charge
DAPA	application, hyperviseur	Anomalies internes à la VM Ballooning Contention CPU niveau hôte Saturation disque niveau hôte
vPerfGuard	application, VM, hyperviseur	Contention mémoire Saturation CPU Contention disque Augmentation de la charge
Q-clouds	application, hyperviseur	interférence
[109]	VM, hyperviseur	Mise à jour automatique de la base de signatures des fautes
MELA	VM, hyperviseur	Violation de propriétés définies par les utilisateurs
Cloud PD	application, VM, hyperviseur	Anomalies logicielles Variation significative de la charge Migration non valide Défaut d'estimation de ressources Contention au niveau hôte
AMAD	**hyperviseur, cluster**	**Migration abusive de VMs**

TABLE 3.1 – Récapitulatif des systèmes de détection d'anomalies

3.3 Conception

Le système de supervision AMAD détecte l'occurrence d'une attaque par migrations intempestives de VMs et identifie les VMs à l'origine de l'attaque.

AMAD est composé de trois modules pour assurer deux opérations principales : i) générer une alerte lorsque le nombre de migrations de VMs exécutées au sein d'un cluster révèle que le cluster est possiblement sous attaque, cette opération est assurée par *un module de génération d'alerte*, ii) analyser les profils de consommation en ressources des VMs s'exécutant au sein du cluster afin de retrouver l'ensemble des VMs à profils de consommation fluctuants et coordonnés considérées comme étant celles à l'origine de l'attaque, cette opération est assurée par *le module d'analyse*. Comme illustré sur la figure 3.1, ces deux modules s'exécutent avec un troisième *module de monitoring* chargé de communiquer avec la couche de management du cloud pour récupérer les valeurs des métriques nécessaires au module de génération d'alerte et au module d'analyse de la consommation en ressources des VMs.

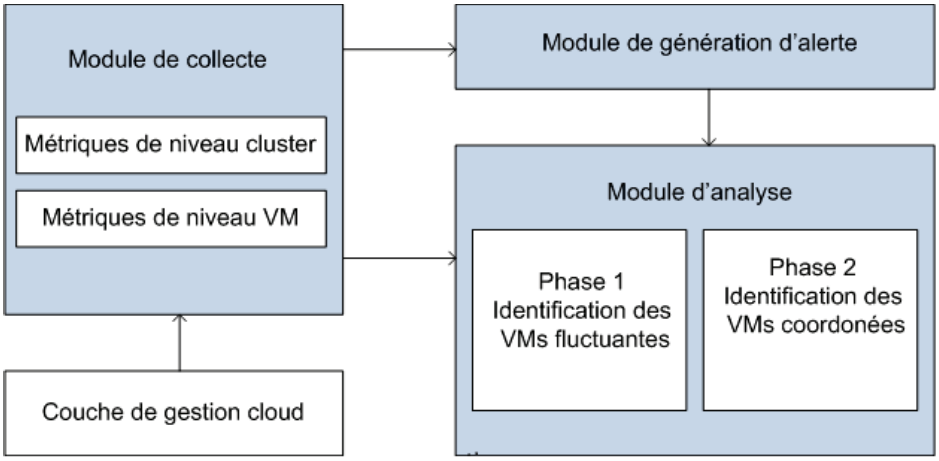


FIGURE 3.1 – Architecture globale d'AMAD

3.3.1 Module de monitoring

Le module de monitoring est chargé de collecter les métriques nécessaires au module de génération d'alerte pour évaluer si le cluster est possiblement sous attaque ainsi que les métriques nécessaires au module d'analyse pour caractériser les profils de consommation en ressources des VMs, pour cela :

— Le module de monitoring vérifie de façon périodique, pour chaque cluster, l'occurrence de migrations de VM. Le cas échéant, il rapporte quelle VM a été migrée ainsi que le

temps d'occurrence de la migration, au module de *génération d'alerte*. Ces informations sont les seules requises pour le module de génération d'alerte pour évaluer si le cluster est sous attaque,

— Le module de monitoring collecte aussi en temps réel les valeurs instantanées d'utilisation mémoire et d'utilisation CPU des VMs telles que calculées par l'hyperviseur. Ces métriques représentent l'utilisation courante en ressources des VMs. Elles représentent la quantité courante de mémoire (resp. CPU) utilisée de façon active par les VMs, exprimée en pourcentage de la totalité de la mémoire (resp. CPU) configurée à la VM. Ces métriques permettent de restituer la variation de la consommation des VMs en temps réel, ce qui permet au module d'analyse de construire les profils de consommation des VMs,

— Le module de monitoring collecte aussi en temps réel la métrique d'*engagement dynamique* calculée par DRS, pour la mémoire d'une part et pour le CPU d'autre part. Ces métriques sont collectées pour une variante du module d'analyse personnalisée pour une application au système de gestion dynamique VMware DRS et ne sont pas requises pour le fonctionnement du module dans sa version de base.

3.3.2 Module de génération d'alerte

Le module de génération d'alerte est chargé de lever une alerte dès que le nombre (n) de migrations de VMs exécutées au sein du cluster sur une durée de temps donnée dépasse un seuil (N) préalablement défini. Ce seuil (N) est configurable et représente la moyenne du nombre de migrations de VMs exécutées sur une période de temps qui couvre un état d'exécution normal du cluster.

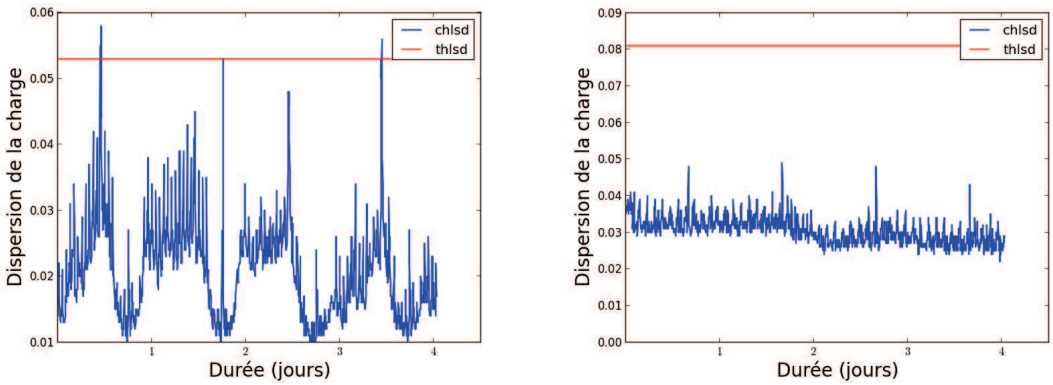
Les occurrences de migrations restent en nombre limité et surtout varient assez peu dans le temps ce qui rend la prédiction de leur nombre possible. C'est la raison pour laquelle nous pensons qu'il est possible de caractériser l'état d'exécution normal (du point de vue du cluster) en fonction du nombre de migrations de VMs pour une durée donnée, comme nous le préconisons pour le module de génération d'alerte.

Il existe très peu de données publiques concernant les pratiques des fournisseurs cloud pour ce qui est du recours à la migration de VMs dans le cloud. A notre connaissance, la seule étude qui traite de cet aspect à un niveau large échelle – mesure sur 19 jours de 90 k VMs exécutées sur 8 k serveurs – a été réalisée et publiée par IBM en 2013 [114]. Cet article révèle que la migration de VMs reste utilisée de façon occasionnelle et suit un schéma répétitif et donc prédictible. Les auteurs expliquent notamment que sur les 19 jours de mesure, la grande majorité des VMs (78%) n'a jamais été migrée. De plus, seules 3% des VMs qui ont déjà été migrées sont migrées une seconde fois, au plus tôt deux jours après la première migration. L'étude relève aussi que les

migrations se produisent à des moments similaires de la journée, ce qui permet de prédire les occurrences de migrations.

Nous livrons avec la figure 3.2 des observations d'une analyse de clouds privés d'Orange sur plate-forme de virtualisation VMware portant sur la migration de VMs, par relevés des métriques VMware *chlsd* de mesure du déséquilibre de charge et *thlsd* de définition du déséquilibre maximum de charge toléré. On rappelle, comme expliqué en section 2.3.4, que ce sont ces valeurs *chlsd* dépassant le seuil *thlsd* qui déclenchent au niveau de DRS un processus de décision pour évaluer le besoin et les possibilités de migrations.

La figure 3.2 illustre deux exemples représentatifs de la variation des valeurs de *chlsd* sur une durée de 4 jours associés à deux clusters appartenant à des clouds privés d'Orange. Sur la figure 3.2(a), nous pouvons observer une variation régulière des valeurs de *chlsd* qui dépassent le *thlsd* au même moment de chaque journée susceptibles de produire à ces moments une migration de VM. Sur la figure 3.2(b), nous pouvons observer que le cluster affiche des valeurs de *chlsd* très faibles ne dépassant jamais le *thlsd* et leurs variations dans le temps restent très faibles également.



(a) Variation de chlsd au sein d'un cluster durant 4 jours - fluctuation régulière

(b) Variation de chlsd au sein d'un cluster durant 4 jours - valeurs de chlsd faibles

FIGURE 3.2 – Variation du chlsd d'un cluster du cloud réel sur une durée de 4 jours

Les éléments en notre possession concernant la migration de VMs montrent qu'il est envisageable de définir des seuils pour le nombre de migrations de VMs qui caractérisent un état d'exécution normal d'un cluster.

3.3.3 Module d'analyse

Le module d'analyse est chargé de caractériser les profils de consommation en ressources des VMs pour identifier les VMs qui sont à l'origine de l'attaque par migrations intempestives de VMs. Plus précisément, ce module identifie l'ensemble des VMs fluctuantes et coordonnées car

de telles VMs présentent des profils de consommation qui peuvent être à l'origine d'une création fréquente d'un déséquilibrage de charge dépassant le seuil de déséquilibrage de charge autorisé amenant ainsi le système de gestion dynamique de ressources à exécuter un nombre important de migrations de VMs.

L'analyse de la consommation en ressources des VMs par AMAD est composée de deux phases principales. La première phase identifie l'ensemble des VMs à profil de consommation en ressources fluctuant, en présence dans le cluster. La seconde phase analyse les VMs fluctuantes identifiées lors de la première phase et identifie dans cet ensemble le sous ensemble de VMs à profils de consommation fortement corrélés. Cette deuxième phase écarte les VMs dont la fluctuation est due à des pics de consommation générés par une activité normale puisque le niveau de corrélation temporelle entre ces VMs non attaquantes est vraisemblablement faible, comme cela sera détaillée dans la section 3.5. L'analyse des profils de consommation en ressources des VMs s'effectue sur un historique d'analyse *auto-adaptatif* que l'on dénomme ***fenêtre de rétro-analyse*** dans le reste du mémoire. Nous commençons par définir la fenêtre de rétro-analyse avant de présenter la phase d'identification des VMs fluctuantes et la phase d'identification des VMs fortement coordonnées.

3.3.3.1 Définition de la fenêtre de rétro-analyse

Pour couvrir la totalité de la durée sur laquelle le cluster subit l'attaque, l'analyse des profils de consommation en ressources des VMs s'effectue sur une fenêtre de rétro-analyse dont la borne inférieure doit repérer au plus près le début de l'attaque. Cette fenêtre auto-adaptative est définie comme suit : [T_{mig1} - δ,T_{alert}], avec T_{alert} représente le temps d'occurrence de l'alerte, qui fait également référence au temps d'occurrence de la N^{ieme} migration de VM de la série de migrations à l'origine de l'alerte et (T_{mig1} - δ) représente le temps d'occurrence moins δ de la première migration de VM de la série de migrations à l'origine de l'alerte. δ est un paramètre d'ajustement de la borne inférieure de l'intervalle d'analyse pour que celle-ci couvre le temps d'occurrence du début de l'attaque qui se produit très vraisemblablement avant l'occurrence de la première migration de VM de la série de migrations à l'origine de l'alerte. De cette façon, la fenêtre de rétro-analyse maximise la signification statistique des données puisqu'elle permet de centrer l'analyse sur l'exécution du cluster sur la durée de l'attaque uniquement, éliminant ainsi les données associées à un état d'exécution du cluster antérieur à l'attaque.

C'est sur cette fenêtre de rétro-analyse ainsi définie qu'AMAD identifie l'ensemble des VMs qui présentent des profils de consommation en ressources fluctuants, lors de la phase 1 du module d'analyse.

3.3.3.2 Identification des VMs fluctuantes

L'objectif de cette phase 1 est d'identifier les VMs ayant des profils de consommation en ressources fluctuants. Nous définissons *la fluctuation* comme une succession cyclique de consommation de faibles et de fortes quantités de ressources.

Pour identifier les VMs ayant de tels profils de consommation, nous tirons avantage de la puissance de l'algorithme des k-moyennes [107] qui a pour objectif de partitionner des données selon des critères de discrimination. Étant donnés deux ensembles de données associées à n observations, l'algorithme des k -moyennes partitionne ces données en k partitions de telle sorte que les données appartenant à une même partition soient le plus possiblement similaires, ce qui signifie que la distance entre les points associés aux données d'une même partition est petite en comparaison avec la distance entre des points associés aux données appartenant à deux partitions distinctes.

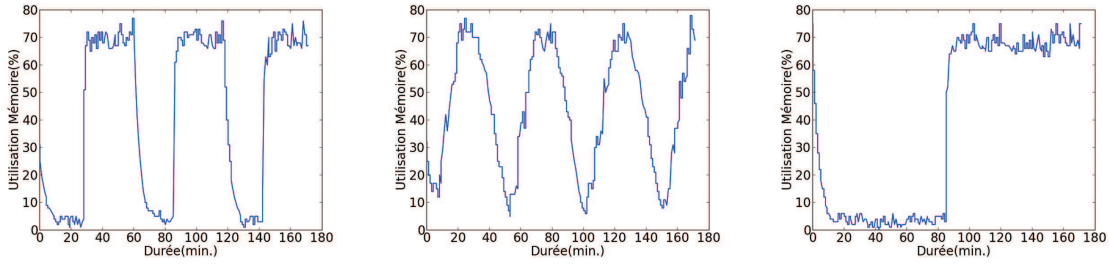
Pour être capable de discriminer les VMs en fonction du critère de fluctuation de la consommation en ressources telle que nous l'avons définie, nous proposons une méthode qui permet de transformer les données de la série temporelle de la consommation des VMs en des vecteurs de valeurs qui restituent le niveau de fluctuation de la consommation en ressources des VMs. Cette transformation permet d'obtenir des données sur lesquelles l'application de l'algorithme des k -moyennes avec k = 2 produit un partitionnement des VMs en deux groupes, un groupe contenant les VMs à profil de consommation fluctuant et un deuxième groupe contenant les VMs à profil de consommation en ressources relativement stable. Notre choix de l'algorithme de partitionnement des k-moyennes est motivé d'une part par le fait que nous connaissons a priori le nombre de clusters (deux clusters qui correspondent aux VMs fluctuantes et aux VMs stables) et d'autre part par le fait que l'algorithme implémenté doit être en mesure de traiter un nombre important de données du fait que les clusters sont composés de dizaine d'hôtes contenant chacun des dizaines de VMs dont la consommation en ressources est mesurée sur toute la durée de l'historique d'analyse.

Pour introduire la métrique - nommée 'indice de fluctuation' - que nous proposons pour partitionner les VMs en fonction du critère de fluctuation par application de l'algorithme des k-moyennes, nous présentons maintenant quelques profils de consommation en ressources particuliers afin de discuter des représentations possibles avec différents types de données.

La figure 3.3 illustre trois profils de consommation en ressources associés à différentes VMs. Les figures 3.3(a), 3.3(b) et 3.3(c) représentent les séries temporelles, c'est à dire la variation dans le temps de l'utilisation mémoire de ces trois VMs. Les figures 3.3(d), 3.3(e) et 3.3(f) représentent les distributions respectives des séries temporelles de ces trois VMs.

Le premier profil de consommation, représenté en figure 3.3(a), correspond au type de profil des VMs attaquantes introduit dans le chapitre 2. La figure 3.3(d) montre que les données de la série temporelle de ce profil sont distribuées principalement autour de deux valeurs (5%

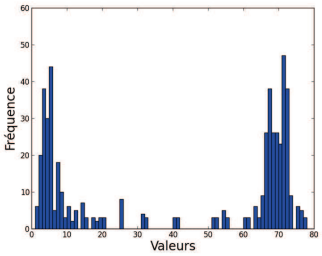
et 70%). Le second profil de consommation, figure 3.3(b), est une variante du premier profil, permettant d'exécuter pareillement l'attaque par migrations intempestives de VMs. Cependant, nous pouvons observer sur la figure 3.3(e) que les données de la série temporelle de ce profil sont distribuées de façon uniforme sur l'ensemble des valeurs comprises entre 5% et 75%. Cela produit une distribution différente de celle obtenue avec le premier profil de consommation. Cela est dû au fait que les données de la série temporelle du second profil présentent des dispersions plus faibles que celles du premier profil puisqu'elles prennent des valeurs intermédiaires entre 5% et 75%. Bien que le troisième profil, figure 3.3(c), présente une distribution, illustrée en figure 3.3(f), similaire à celle du premier profil, il est évident que ce troisième profil de consommation ne peut pas correspondre à un profil de consommation d'une VM attaquante puisque celui-ci ne peut pas être à l'origine d'une exécution multiple de migrations de VMs et doit par conséquent être considéré par AMAD comme étant un profil de consommation stable.



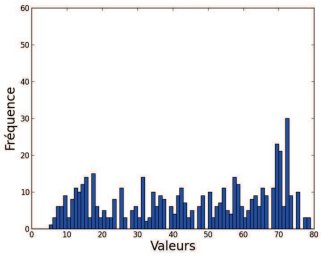
(a) Profil de consommation mémoire fluctuant (1)

(b) Profil de consommation mémoire fluctuant (2)

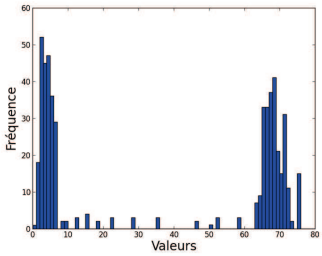
(c) Profil de consommation mémoire stable



(d) Profil de consommation mémoire fluctuant (1) Distribution



(e) Profil de consommation mémoire fluctuant (2) Distribution



(f) Profil de consommation mémoire stable Distribution

FIGURE 3.3 – Profils de consommation en ressources de trois VMs et distributions correspondantes

Ni l'application de l'algorithme des k-moyennes sur les données des séries temporelles de ces trois VMs, ni l'application de l'algorithme des k-moyennes sur les distributions associées aux données des séries temporelles ne permet de partitionner les VMs en fonction de leur niveau de fluctuation. En effet, un partitionnement des VMs basé sur les données des séries temporelles regroupe les trois VMs – les deux premières fluctuantes et la dernière stable – dans un même groupe, avec toute autre VM qui présente une moyenne de consommation en ressources similaire.

Un partitionnement des VMs basé sur les distributions est tout aussi insatisfaisant puisqu'il regroupe la première VM qui est une VM fluctuante avec la troisième VM qui est quant à elle stable, dans un même groupe. Ces considérations nous amènent à proposer une nouvelle métrique que nous appelons *indice de fluctuation*, pour restituer le caractère plus ou moins fluctuant des profils de consommation des VMs qui est le critère prépondérant pour le renouvellement de l'état de déséquilibre de charge créé dans l'attaque par migrations intempestives de VMs.

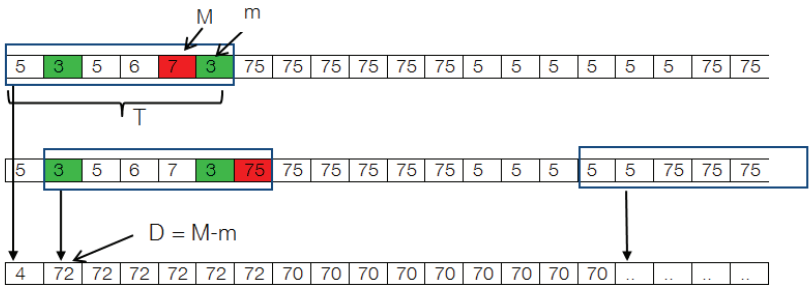
L'indice de fluctuation est défini par la différence (D) entre la valeur maximum (M) et la valeur minimum (m) des données des séries temporelles de consommation d'une VM, calculée sur une fenêtre temporelle glissante. La taille (T) de cette fenêtre temporelle glissante est égale à la moyenne des durées qui séparent deux migrations de VMs consécutives exécutées durant la durée de l'attaque (la durée de l'attaque se définissant pour AMAD par la taille de la fenêtre de rétro-analyse). Partant du constat que la migration de VMs occasionnée par l'attaque est provoquée par le basculement de la consommation en ressources des VMs, définir la taille de cette fenêtre par la moyenne des durées qui séparent les migrations permet d'obtenir une taille de fenêtre temporelle glissante qui soit de l'ordre de grandeur de la durée qui sépare deux basculements de consommation des VMs attaquantes maximisant ainsi les valeurs de la différence (D) qui forment le vecteur d'indices de fluctuation.

La figure 3.4 donne deux exemples de transformation des données de la série temporelle de consommation en un vecteur d'indices de fluctuation, pour une VM fluctuante (figure 3.4(a)) et pour une VM stable (figure 3.4(b)). Sur ces deux figures, les deux premiers vecteurs sont identiques et illustrent les mêmes données de série temporelle de consommation d'une VM avec la fenêtre glissante positionnée sur deux positions consécutives (qui correspondent à deux relevés consécutifs de consommation en ressources) et le troisième vecteur représente le vecteur d'indices de fluctuation résultant.

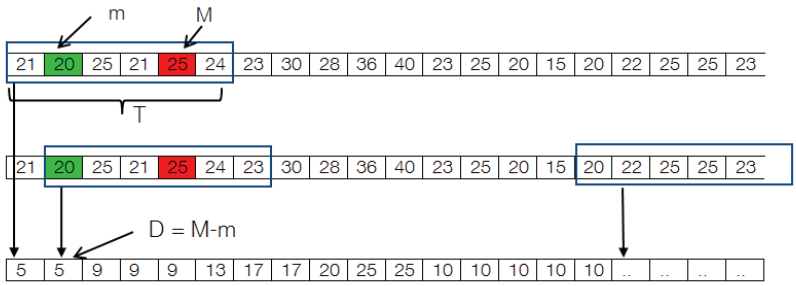
Nous pouvons observer que le vecteur d'indices de fluctuation associé à la VM fluctuante contient des valeurs qui sont nettement plus élevées que celles du vecteur associé à la VM stable.

Grâce à cette transformation en indices de fluctuation des données de la série temporelle de consommation, les données brutes sont transformées en des données qui restituent le degré de fluctuation de la consommation d'une VM. L'application de l'algorithme des k -moyennes avec k = 2 sur l'ensemble des vecteurs d'indices de fluctuation associés aux VMs – la longueur de ces vecteurs d'indices de fluctuation est définie par la durée de la fenêtre de rétro-analyse – permet de partitionner les VMs en deux groupes : un premier groupe contenant les VMs avec les vecteurs ayant des valeurs d'indices de fluctuation élevées, qui correspondent aux VMs fluctuantes et un second groupe contenant les VMs avec les vecteurs ayant des valeurs d'indices de fluctuation faibles, qui correspondent aux VMs stables.

L'application de l'algorithme des k-moyennes sur les vecteurs d'indices de fluctuation associés aux exemples de profils de consommation illustrés sur la figure 3.3, permet effectivement



(a) Calcul du vecteur d'indices de fluctuation pour une VM fluctuante



(b) Calcul du vecteur d'indices de fluctuation pour une VM stable

FIGURE 3.4 – Exemples de calcul d'indices de fluctuation

de regrouper les deux premières VMs dans un même groupe de VMs fluctuantes, alors que la troisième VM est regroupée avec les autres VMs stables du cluster dans un autre groupe.

Nous allons maintenant décrire ci-dessous la deuxième phase d'analyse d'AMAD qui consiste à analyser le groupe de VMs fluctuantes identifiées lors de cette première phase, pour identifier parmi cet ensemble le sous ensemble de VMs dont les profils de consommation sont temporellement fortement corrélés.

3.3.3.3 Identification des VMs corrélées

Pour produire des effets significatifs, l'attaque par migrations intempestives de VMs requiert une capacité d'influence suffisante sur le déséquilibre de charge, que l'attaquant acquiert en coordonnant l'exécution de plusieurs VMs cumulant ainsi les effets de déséquilibre de charge que produit chacune des VMs afin d'amener le système de gestion dynamique des ressources à exécuter de multiples migrations de VMs [115]. Cette phase 2 d'identification des VMs coordonnées, a pour objectif d'identifier au sein de l'ensemble des VMs fluctuantes identifiées lors de la première phase d'analyse d'AMAD, le sous ensemble de VMs dont les profils de consommation sont temporellement fortement corrélés. Pour cela, nous utilisons la méthode d'analyse canonique des corrélations (CCA), une technique statistique qui permet la modélisation de la relation de corrélation [116].

Proposée par H. Hoteling en 1936, CCA est une méthode statistique qui a pour objectif de caractériser et de quantifier la dépendance linéaire entre deux ensembles de variables mesurées sur les mêmes individus, CCA cherche à répondre à la question suivante : *'quels sont les profils qui se produisent en même temps dans les champs 'A' et 'B' associés à un même groupe d'individus et quel est leur degré de corrélation*' [117].

Étant donnés deux ensembles de mesures de consommation en ressources de deux VMs, CCA mesure la magnitude de leur relation. Nous appliquons CCA sur les données des séries temporelles de consommation de chaque paire de VMs fluctuantes identifiées lors de la phase précédente, sur une durée d'analyse définie par la fenêtre de rétro-analyse. Si la valeur de corrélation calculée pour un couple de VMs dépasse un seuil préalablement défini, alors nous considérons que les deux VMs sont fortement et anormalement corrélées. Ces VMs reconnues comme fluctuantes en phase 1 d'analyse et comme fortement corrélées en phase 2 d'analyse par AMAD, sont considérées comme étant à l'origine du nombre anormal de migrations de VMs exécuté au sein du cluster.

Dans la section suivante, nous évaluons AMAD en mesurant l'exactitude de la détection dans une pluralité de contextes.

3.4 Evaluation sur Plateforme

Nous avons implémenté le système AMAD sur notre plate-forme d'expérimentation de laboratoire et nous avons conduit des expérimentations approfondies pour l'évaluation de sa performance de détection. Dans cette section, nous allons d'abord décrire la méthodologie d'évaluation définie, puis nous présenterons les mesures de la performance de détection d'AMAD obtenues et enfin nous analyserons l'impact de la taille de la fenêtre de rétro-analyse sur cette performance de détection.

3.4.1 Méthodologie d'évaluation

Pour l'évaluation d'AMAD, nous avons exécuté une série d'attaques par migrations intempestives de VMs sur notre plate-forme d'expérimentation et pour chacune des attaques réalisées, nous mesurons l'exactitude avec laquelle AMAD détecte l'occurrence de l'attaque et identifie les VMs qui sont à l'origine de l'attaque.

La plate-forme d'expérimentation est constituée de 5 serveurs IBM 3550 M3 avec 8 coeurs Xeon 2.133 GHz et 16 GB RAM chacun. Les serveurs exécutent des hyperviseurs VMware ESXi 4.1 et sont gérés par la couche de management VMware vCenter 4.1. DRS est activé en mode automatique pour la gestion de la migration des VMs.

Les VMs sont réparties de façon uniforme sur les hôtes afin d'assurer une équité des contextes d'exécution des VMs. Nous exécutons 9 VMs de 2 GB mémoire et 1 vCPU sur chaque hôte (i.e.

45 VMs pour l'ensemble de la plate-forme d'expérimentation). Chacune des VMs exécute le système d'exploitation Ubuntu 64 bits serveur. Avec une telle configuration, le sur-engagement des hôtes est de 16% pour la mémoire et de 25% pour le CPU.

Pour la génération de la charge, nous utilisons les outils de génération de charge configurables présentés en section 2.5.1. Pour mener nos expérimentations, ces outils sont paramétrés avec des profils de consommation de VM attaquante tels que ceux illustrés en figure 3.3(a) d'une part, et avec des valeurs de consommation mémoire ou de CPU issues de la consommation de VMs qui s'exécutent sur des clouds privés d'Orange pour les autres VMs (non attaquantes) d'autre part.

Nous avons exécuté 50 expérimentations d'une durée de 3 heures chacune avec :

— 10 VMs attaquantes distribuées de façon égale sur deux hôtes avec, sur toute la durée de l'expérimentation, des profils de consommation coordonnées et en opposition de phase entre les VMs s'exécutant sur un premier hôte et celles s'exécutant sur un deuxième hôte, respectant une périodicité dans le profil de consommation établie à 1 heure tels qu'illustrés sur la figure 2.5.1,

— 35 VMs normales (non attaquantes) distribuées sur les cinq hôtes. Les agents de consommation au sein de chaque VM sont configurés avec des traces de consommation de VM collectées sur clouds réels, les traces à injecter dans les VMs sont sélectionnées de façon aléatoire pour chaque expérimentation à partir d'une base contenant 200 traces de consommation de VMs.

Pour quantifier la performance de détection d'AMAD, nous utilisons les mesures standards de *précision* et de *rappel*. Soient VP, FN, FP, VN, les métriques de mesure qui représentent respectivement, 1) Vrais Positifs : les VMs attaquantes correctement identifiées en tant que telles, 2) Faux Négatifs : les VMs attaquantes qui n'ont pas été identifiées, 3) Faux Positifs : les VMs normales considérées à tord comme étant des VMs attaquantes, 4) Vrais Négatifs : les VMs qui sont correctement identifiées comme étant des VMs normales.

La *précision* et le *rappel* sont définis par les formules 3.1 et 3.2.

$$Precision = \frac{VP}{VP + FP} \tag{3.1}$$

$$Rappel = \frac{VP}{TP + FN} \tag{3.2}$$

La *précision* mesure l'exactitude de détection d'un système et le rappel mesure sa complétude. Un système parfait est un système qui atteint 1 pour la précision et pour le rappel.

3.4.2 Évaluation temps réel

La figure 3.5 livre une image globale des alertes générées par AMAD au cours des 50 expérimentations menées. Ces alertes sont le résultat des migrations de VMs déclenchées par DRS pour remédier au déséquilibrage de charge créé par les profils de consommation en ressource des VMs attaquantes lorsque AMAD est configuré comme suit. Le nombre de migrations de VMs (N) pour lever une alerte est configuré à 3 migrations (étant donné la durée relativement courte de nos expérimentations qui est de trois heures). La borne inférieure de la fenêtre de rétro-analyse [T_{mig1} - δ,T_{alert}] est définie avec un δ qui est fixé à une heure (un retour en arrière de l'ordre d'une période du profil de consommation d'une VM attaquante).

Sur cette figure 3.5, l'axe des abscisses X représente le temps d'occurrence T_{alert} des différentes alertes, l'axe des ordonnées Y de gauche indique la taille des fenêtres de rétro-analyse représentées par des barres et associées aux différentes alertes dont le temps d'occurrence est représenté en X et enfin l'axe des ordonnées Y de droite représente les temps d'occurrence de la première et de la troisième migration - respectivement représentées par des symboles carré et triangle - de la série de migrations à l'origine de ces mêmes alertes. Étant donnée une alerte, celle-ci est représentée par une barre dont la hauteur indique la taille de la fenêtre de rétro-analyse associée à l'alerte telle que calculée par AMAD, le symbole carré inférieur, resp. le symbole triangle supérieur, porté sur cette barre représente la première migration de VM, resp. la troisième migration de VM (N=3), de la série des migrations à l'origine de l'alerte dont le temps d'occurrence est porté sur l'axe des abscisses.

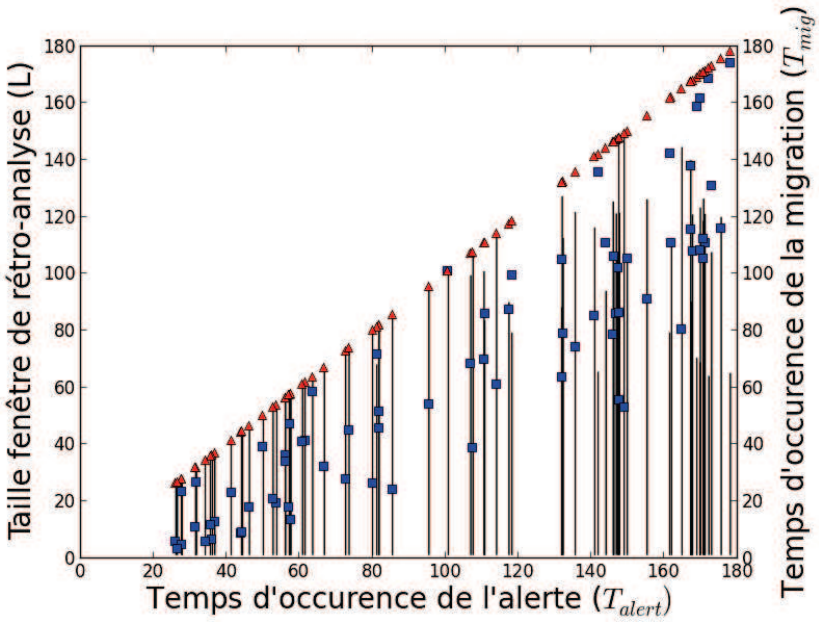


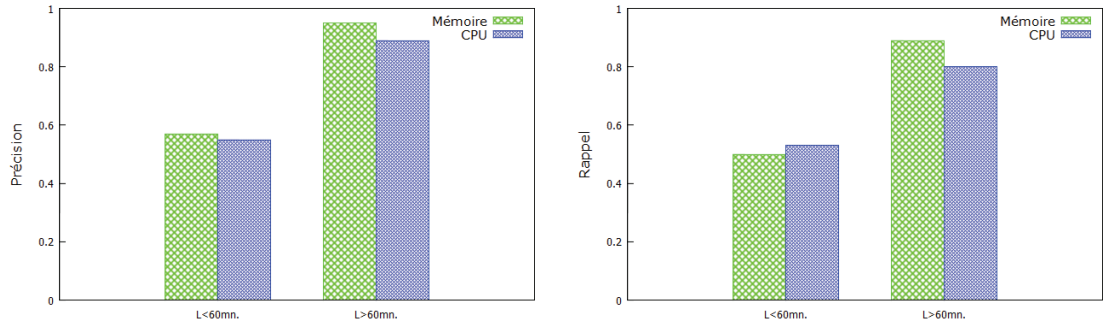
FIGURE 3.5 – Image globale des alertes générées par AMAD

Nous pouvons observer sur la figure 3.5 que les effets de l'attaque par migrations intempestives de VMs dépendent fortement du contexte d'exécution – i.e. des profils de consommation en ressources des VMs co-résidentes avec les VMs attaquantes – puisque l'intervalle de temps entre la première migration et la troisième migration varie significativement d'une alerte à une autre : 1 minute d'intervalle de temps pour T_{alert} = 27mn (migrations multiples simultanées de VMs) à 131 minutes pour T_{alert} = 147 mn.

Nous reviendrons dans la section suivante sur le cas des fenêtres de rétro-analyse de petite taille qui correspondent à des cas particuliers impactant négativement les performances de détection d'AMAD.

La figure 3.6 illustre les moyennes des valeurs de *précision* et de *rappel* (mémoire et CPU) calculées pour l'ensemble des alertes générées au cours de nos expérimentations. Nous représentons sur la figure 3.6(a) séparément les moyennes de précision (mémoire et CPU) obtenues pour des fenêtres de rétro-analyse de taille inférieure à 60 minutes et pour celles de taille supérieure à 60 minutes. La figure 3.6(b) donne une représentation similaire pour le *rappel*.

Nous pouvons observer qu'avec des fenêtres de rétro-analyse de taille inférieure à 60 minutes, la précision et le rappel sont de l'ordre de 0.6 pour la mémoire et pour le CPU alors que les valeurs moyennes de précision et de rappel sont de l'ordre de 0.9 lorsque les fenêtres de rétro-analyse ont des tailles supérieures à 60 minutes.



(a) Précision en fonction de différentes tailles de la fenêtre de rétro-analyse

(b) Rappel en fonction de différentes tailles de la fenêtre d'analyse

FIGURE 3.6 – Précision et rappel pour l'ensemble des alertes, basés sur les métriques d'utilisation mémoire et CPU

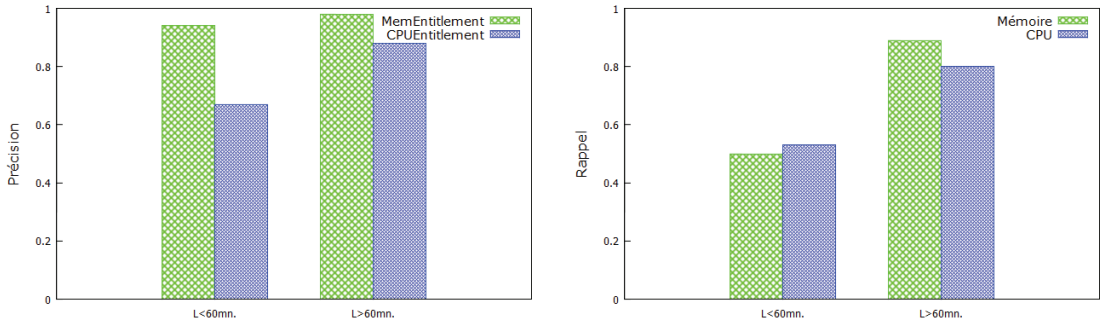
Pour expliquer les moins bonnes performances obtenues pour les cas particuliers d'alertes correspondant à des tailles de fenêtre de rétro-analyse trop courtes, on rappelle que le module d'analyse d'AMAD est basé sur un algorithme d'analyse qui exploite à travers le calcul de l'indice de fluctuation, l'occurrence de basculements de consommation dans les profils des VMs attaquantes. Par conséquent, la performance de détection d'AMAD est négativement impactée lorsque le nombre d'occurrences de basculements de consommation dans les profils des VMs attaquantes n'est pas suffisant (soit aucun basculement soit un seul) sur la durée de la fenêtre de

rétro-analyse. Cela se produit pour les alertes générées en début d'expérimentation (δ tronqué) lorsque la fenêtre de rétro-analyse couvre une période de temps qui inclut un seul état de déséquilibrage de charge à l'origine de trois migrations très rapprochées dans le temps. Ce cas particulier est comparable au contexte d'exécution réel où les effets de l'attaque déclenchent une alerte très rapidement après le début de l'attaque du fait de migrations simultanées ou très rapprochées dans le temps dès le premier déséquilibrage de charge. Dans ce cas, la fenêtre de rétro-analyse couvre une période de temps antérieure à l'attaque (δ couvrant essentiellement cette période antérieure à l'attaque), ce qui n'est pas favorable à l'identification des VMs attaquantes.

Enfin, nous illustrons sur la figure 3.7 le même type de mesures, *précision* et *rappel*, obtenues avec une variante de l'algorithme, modifié pour prendre en considération les métriques spécifiques VMware d''*engagement dynamique*' en lieu et place des métriques standard d'utilisation des ressources mémoire et CPU comme c'est le cas dans la version par défaut de l'algorithme d'AMAD. Pour cette variante d'AMAD, les vecteurs d'indices de fluctuation sont calculées en transformant non pas les données des séries temporelles de consommation en ressources des VMs, mais en transformant les données des métriques VMware d''*engagement dynamique*'. L'interprétation des observations donnée ci-dessous permet d'en comprendre l'intérêt. Nous pouvons observer que cette fois-ci, AMAD montre une moyenne de précision de détection de 0.9 pour la mémoire lorsque la taille de la fenêtre de rétro-analyse est inférieure à 60 minutes. Cette performance est significativement meilleure que le résultat obtenu précédemment avec la version par défaut d'AMAD calculant les vecteurs d'indices sur les données des séries temporelles de consommation des VMs. Cette amélioration de la précision mémoire est due au nombre de *faux positifs* qui est moindre dans le cas de cette variante d'AMAD. En effet, pour cette variante d'AMAD, les VMs à profil de consommation non attaquant qui présentent des pics de consommation mémoire instantanés ne sont plus considérées à tord comme des VMs attaquantes car la métrique VMware d' 'engagement dynamique' mémoire lisse les pics de consommation instantanée et atténue donc la fluctuation du profil. Les autres résultats de *précision* et de *rappel* obtenus avec la métrique d'*engagement dynamique* restent aussi performants que ceux obtenus dans la version par défaut d'AMAD basée sur les données des séries temporelles de consommation des VMs.

Dans cette section, nous avons démontré qu'AMAD est performant dans la majorité des cas de figure pour identifier les VMs attaquantes. Par principe de conception, AMAD requiert la présence de basculements dans les profils de consommation des VMs attaquantes pour être en mesure de les détecter, d'où ses moindres performances quand l'attaque débute très peu de temps avant la génération de l'alerte.

Nous allons maintenant analyser plus en détails la performance de détection de l'algorithme d'analyse d'AMAD en fonction de la taille de la fenêtre de rétro-analyse.



(a) Précision en fonction de différentes tailles de la fenêtre de rétro-analyse

(b) Rappel en fonction de différentes tailles de la fenêtre de rétro-analyse

FIGURE 3.7 – Précision et rappel pour l'ensemble des alertes, basés sur les métriques VMware d'*engagement dynamique*

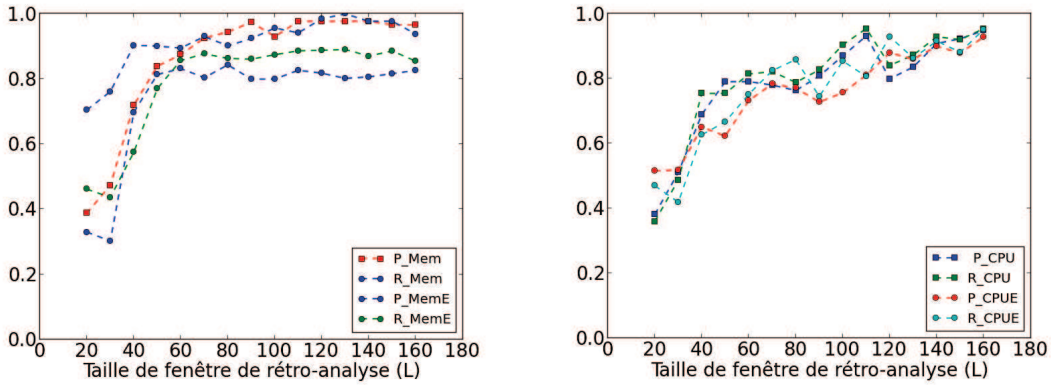
3.4.3 Impact de la taille de la fenêtre de rétro-analyse

Pour mieux analyser l'impact de la taille (L) de la fenêtre de rétro-analyse sur la performance de détection du module d'analyse d'AMAD, nous appliquons l'algorithme d'analyse utilisé dans ce module, hors ligne, sur chacune des 50 traces collectées durant les expérimentations que nous avons menées pour l'évaluation temps réel d'AMAD et nous faisons varier la taille de la fenêtre de rétro-analyse d'une durée de 20 minutes à une durée de 180 minutes par pas d'incrémentation de 10 minutes entre deux analyses consécutives.

La figure 3.8(a) illustre les valeurs moyennes de *précision* et de *rappel* obtenues lorsque l'algorithme d'analyse est appliqué sur les données des séries temporelles de consommation mémoire des VMs, ainsi que celles obtenues pour la variante d'AMAD basée sur les métriques d' 'engagement dynamique' mémoire. Ces moyennes sont calculées sur les 50 traces pour chaque valeur de taille de fenêtre de rétro-analyse. Une représentation similaire est procurée pour le CPU sur la figure 3.8(b).

Nous pouvons observer, à la fois pour la mémoire et pour le CPU, que l'algorithme démontre constamment des valeurs moyennes de *précision* et de *rappel* élevées dès qu'il est appliqué sur une fenêtre de rétro-analyse de taille suffisante : supérieures à 0.7 à partir d'une taille L = 40 mn. Cette taille correspond dans nos expérimentations à la première analyse effectuée sur une fenêtre de rétro-analyse incluant un basculement de consommation (la demi-période du profil de la VM attaquante est de 30mn), en particulier le premier basculement qui apparaît dans le profil de la VM attaquante à T = 30 mn. La fenêtre de rétro-analyse devient optimale pour la performance de détection de l'algorithme d'analyse d'AMAD, dès qu'elle couvre au moins deux basculements de consommation (L =60).

Ces résultats confirment l'importance de la taille de la fenêtre de rétro-analyse pour la performance de détection d'AMAD. C'est le mécanisme de l'alerte qui permet d'agir sur la



(a) Impact de la taille de la fenêtre de rétro-analyse sur la précision pour la mémoire

(b) Impact de la taille de la fenêtre de rétro-analyse sur la précision pour le CPU

FIGURE 3.8 – Impact de la taille de la fenêtre de rétro-analyse sur l'exactitude d'AMAD pour la mémoire et le CPU

définition de la taille de la fenêtre de rétro-analyse (L) via l'ajustement du nombre (N) de migrations de VMs qui agit comme seuil pour le déclenchement de l'alerte. Lorsque N augmente, la durée entre la première migration et la migration déclenchant l'alerte augmente aussi la plupart du temps, ce qui augmente également la taille de la fenêtre de rétro-analyse qui couvre l'attaque à l'origine de cette alerte. Avec les traces de consommation de nos expérimentations, nous obtenons une moyenne de tailles des fenêtres de rétro-analyse de 81 minutes lorsque N=3 et 27% des alertes ont des tailles de fenêtre de rétro-analyse inférieures à 60 minutes. Lorsque N prend la valeur 4, cette moyenne augmente à 97 minutes et 17% seulement des alertes ont des tailles de fenêtres de rétro-analyse inférieures à 60 minutes (nous avons paramétré N à 3 dans nos expérimentations pour une complétude de nos évaluations).

Dans la section suivante, nous évaluons l'algorithme d'analyse d'AMAD lorsque celui-ci est appliqué dans des conditions d'exécution des VMs issues de contextes d'exécution de clouds réels.

3.5 Évaluation dans un contexte cloud

Dans cette section, nous évaluons l'exactitude (précision et rappel) de la capacité d'AMAD à identifier des VMs attaquantes parmi les traces de consommations de VMs collectées au sein d'un cloud réel. En construisant ainsi un ensemble de traces mêlant des traces de consommation de VMs attaquantes à celles de VMs d'un cloud réel, on simule une exécution de l'attaque par migrations intempestives de VMs dans un cloud réel. L'objectif de cette évaluation est de mesurer les performances du module d'analyse d'AMAD vis-à-vis de l'hétérogénéité qui caractérise les contextes d'exécution cloud : un nombre de VMs fortement variable d'un cluster à un autre,

une hétérogénéité des configurations en ressources des VMs (mémoire et CPU) et une forte dynamicité des profils de consommation des applications exécutées par certaines VMs.

Pour mener cette évaluation, nous utilisons des traces réelles de consommation de 6 clusters, collectées sur des clouds privés d'Orange. Le nombre de VMs varie entre 35 VMs et 242 VMs d'un cluster à un autre.

Nous appliquons une analyse par cluster après avoir injecté dans chacun de ces clusters des traces de consommation de 8 VMs attaquantes collectées lors de nos expérimentations (profils de consommation en opposition de phase sur deux hôtes, répartis à égalité entre les profils à effets d'*'attraction'* et ceux à effets de *'repoussement'*). Pour chaque cluster, nous appliquons l'algorithme d'analyse d'AMAD sur une durée d'historique de 180 minutes, la précision et le rappel obtenus sont mesurés séparément pour chaque cluster.

Les détails de la composition de ces clusters ainsi que les résultats de l'évaluation sont présentés dans le tableau 3.2.

Nous pouvons observer sur ce tableau que l'algorithme d'analyse d'AMAD est robuste aux variations des configurations en ressources des VMs. Les valeurs de rappel et de précision (respectivement données dans les colonnes 5 et 6, à la fois pour la mémoire et le CPU) sont supérieures à 0.8, ce qui signifie que l'algorithme d'identification des VMs attaquantes n'a pas de dépendance particulière au nombre de VMs par cluster ni à la variation des configurations en ressources des VMs.

Nous pouvons enfin trouver un dernier résultat dans la dernière colonne du tableau 3.2 qui donne pour chacun des clusters, le résultat de la phase d'identification des VMs fluctuantes (phase 1 décrite dans la section 3.3). Nous pouvons observer que le nombre de VMs fluctuantes identifiées par la phase 1 de l'algorithme d'analyse est plus élevé que le nombre de VMs attaquantes (fluctuantes) injectées dans chacune de nos expérimentations. Cela est dû à la présence dans le cluster, de VMs fluctuantes autres que les VMs fluctuantes attaquantes. Ces VMs fluctuantes autres sont filtrées par la suite par l'algorithme d'identification des VMs attaquantes, grâce à la phase 2 d'analyse de l'algorithme, car les VMs attaquantes autres ne répondent pas au critère de forte corrélation temporelle recherché par la phase 2 de l'algorithme.

Nous pensons que la capacité de la phase 1 de l'algorithme d'analyse d'AMAD, à identifier de façon automatique les VMs fluctuantes est un résultat intéressant en lui-même qui pourrait être exploité dans un autre domaine que celui de la sécurité. Il pourrait être intéressant de considérer son application dans le but d'appliquer des politiques de gestion des ressources particulières aux VMs à profil de consommation fluctuant. Cette phase d'identification des VMs fluctuantes sera plus approfondie dans le chapitre 4.

Dans la section suivante, nous évaluons la robustesse d'AMAD à détecter des variantes de profils de VMs attaquantes puis sa robustesse vis-à-vis de contextes à forte fluctuation de consommation en ressources résultant de la fluctuation de VMs co-résidentes avec les VMs

Cluster	Nb.VMs	Mem (MB)	CPU (Hz)	Précision (cpu, mém)	Rappel (cpu, mém)	VMs Fluc.
Cluster 1	35 + 8	896 (1) 1024 (2) 2048 (20) 4096 (7) 8192 (5)	2300 (16) 4600 (12) 9200 (7)	(1, 1)	(0.8, 0.8)	13
Cluster 2	63 + 8	896 (14) 1024 (5) 2048 (18) 4096 (20) 8192 (6)	2000 (42) 4000 (16) 8000 (5)	(1, 1)	(0.87, 1)	11
Cluster 3	85 + 8	896 (5) 1024 (1) 2048 (32) 4096 (24) 8192 (12) 12288 (7) 16384 (4)	2133 (42) 4266 (24) 8532 (19)	(1, 0.7)	(0.87, 1)	24
Cluster 4	95 + 8	4096 (9) 8192 (36) 12288 (1) 16384 (48) 65536 (1)	3998 (10) 7996 (85)	(1, 1)	(0.8, 1)	11
Cluster 5	100 + 8	2048 (22) 4096 (41) 8192 (15) 12288 (13) 14336 (2) 16384 (7)	1999 (25) 3998 (42) 7996 (29) 15992 (4)	(1, 0.72)	(0.87, 1)	12
Cluster 6	242 + 8	896 (1) 1024 (3) 2048 (59) 4096 (74) 8192 (57) 12288 (42) 16384 (6)	1999 (66) 3998 (76) 7996 (100)	(1, 0.8)	(0.87, 1)	35

TABLE 3.2 – Évaluation d'AMAD dans des configurations de clouds réels

attaquantes à identifier.

3.6 Évaluation de la robustesse de la phase 1

L'objectif des évaluations que nous présentons dans cette section est de mesurer l'exactitude de la phase d'identification des VMs fluctuantes, c'est à dire celle de la phase 1 du module d'analyse d'AMAD, dans deux cas de figure :

— Le premier cas traite d'une situation dans laquelle l'attaquant parvient à mener l'attaque par migrations intempestives de VMs en exécutant des variantes de profil de consommation en ressources de VM attaquante. Le principe de cette évaluation est de déformer les profils fluctuants réguliers que nous avons utilisés jusqu'à présent dans nos expérimentations, afin de mesurer la capacité d'un attaquant à contourner la phase d'identification des VMs fluctuantes,

— Le deuxième cas de figure traite d'une situation dans laquelle le contexte d'exécution des VMs sous le contrôle de l'attaquant est extrêmement fluctuant. L'objectif, dans ce cas, est d'évaluer si les VMs contrôlées par l'attaquant demeurent, dans ces conditions de contexte fluctuant, considérées comme fluctuantes par la phase 1 de l'algorithme d'analyse d'AMAD, faute de quoi AMAD échoue à identifier les VMs attaquantes.

3.6.1 Variantes de profil de consommation de VMs attaquantes

Nous avons identifié trois variantes possibles de profil de consommation de VM attaquante aptes à produire les effets de l'attaque par migrations intempestives de VMs. Nous obtenons ces trois variantes en manipulant respectivement trois paramètres définissant les profils de VM attaquante, i) la pente de la variation de consommation lors des phases d'augmentation et de diminution de la consommation, ii) la période du profil de consommation iii) le rapport cyclique du profil de consommation, c'est à dire le rapport de la durée de faible consommation sur la période du profil.

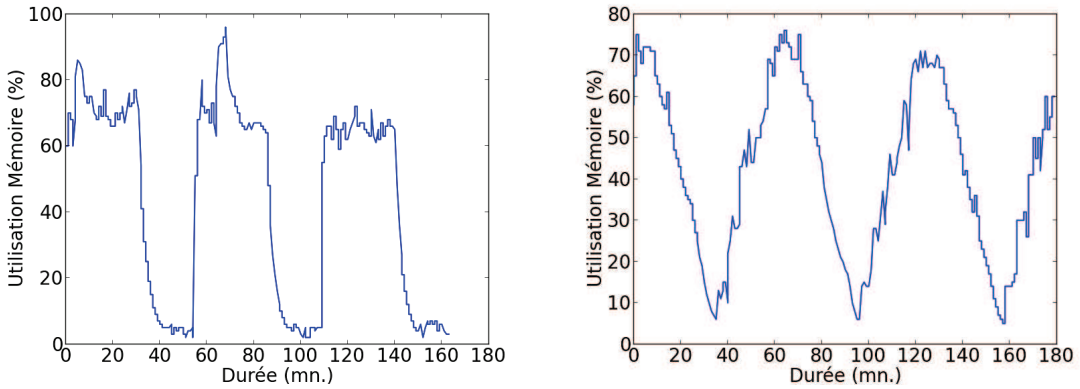
Pour chacune des évaluations hors-ligne détaillées dans les trois sections suivantes, nous reprenons les 50 traces de nos expérimentations dans lesquelles nous remplaçons successivement les profils de consommation des VMs attaquantes par leurs différentes variantes présentées ci-dessus. Pour cette série d'évaluations, nous maintenons le nombre de VMs attaquantes ainsi que tous les autres paramètres de l'algorithme inchangés par rapport aux évaluations présentées en section 3.4. La taille de la fenêtre rétro-analyse est fixée à 180 minutes qui correspond à la durée de nos expérimentations.

3.6.1.1 Variation de la pente de variation de la consommation

Lors de l'exécution de l'attaque par migrations intempestives de VMs, nous avons utilisé des profils de consommation en ressources dans lesquels la variation de la consommation bascule cycliquement de 5% d'utilisation à 75% d'utilisation puis de 75% d'utilisation à 5% d'utilisation.

Dans les variantes de profil de consommation en ressources considérées ici, les variations de 5% à 75% et de 75% à 5% se font de manière incrémentale en prenant plusieurs valeurs intermédiaires. Les profils sont forgés à partir du profil de consommation initial, pour définir quatre profils pour lesquels le passage de 5% à 75% d'utilisation (resp. de 75% à 5%) se fait en prenant respectivement 0 (profil de base), 1, 2, 3 puis 4 valeurs de consommation intermédiaires avec une demi-période de consommation inchangée, égale à 30 minutes.

Les figures 3.9(a) et 3.9(b) illustrent respectivement la variation de l'utilisation mémoire du profil de base d'une VM attaquante (avec 0 valeur de consommation intermédiaire) et l'utilisation mémoire de la variante extrême de profil de consommation présentant 4 valeurs intermédiaires.



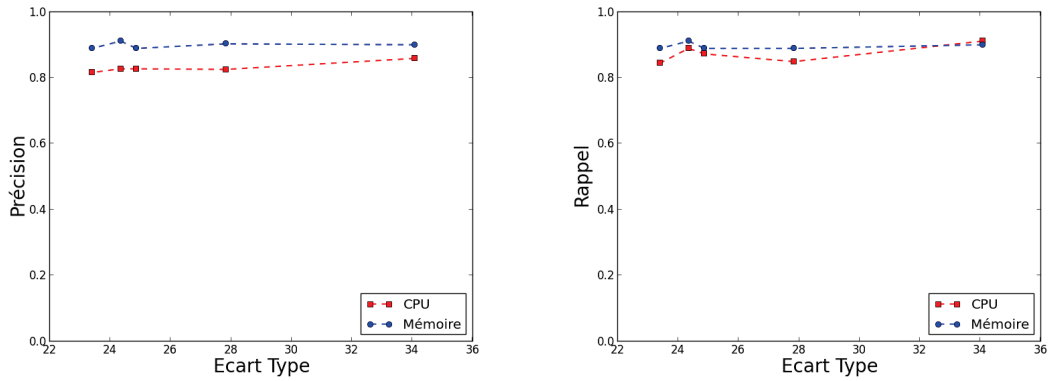
(a) Profil de consommation de base d'une VM attaquante

(b) Variante de profil de consommation d'une VM attaquante

FIGURE 3.9 – Profils de consommation mémoire de VM attaquante

Les figures 3.10(a) et 3.10(b) illustrent respectivement la variation des moyennes des valeurs de *précision* et ce celles du *rappel* obtenues lors de l'application de l'algorithme d'identification des VMs fluctuantes pour la mémoire et le CPU, en fonction de l'écart-type des valeurs de consommation des profils attaquants. Cet écart-type quantifie l'allure de la variation de consommation configurées en entrée des outils de génération de charge ; les profils dont la variation de consommation passe par le moins de valeurs intermédiaires (profil représenté figure 3.9(a)) sont ceux de plus forte valeur d'écart-type.

Nous pouvons observer à partir de cette figure 3.10 que l'algorithme d'identification des VMs fluctuantes démontre de très bonnes performances de détection en termes de précision et de rappel (mémoire et CPU), uniformes pour les quatre variantes de profil de consommation. Ce



(a) Robustesse de la précision vis-à-vis de variantes de profil attaquant

(b) Robustesse du rappel vis-à-vis de variantes de profil attaquant

FIGURE 3.10 – Précision et rappel obtenues avec des variantes de profil de VM attaquante

résultat est directement lié au mode de calcul du vecteur d'indices de fluctuation des VMs, utilisé par AMAD pour le partitionnement : la différence entre maximum et minimum de consommation calculée sur une fenêtre glissante, qui est le principe de calcul des indices de fluctuation, est peu voire pas impactée par la présence de valeurs intermédiaires de consommation, pour la taille de fenêtre ici considérée.

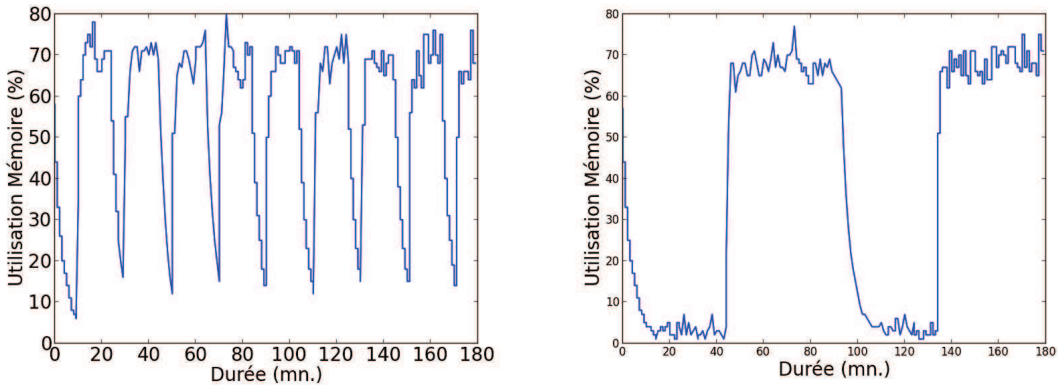
En conclusion, ces résultats démontrent qu'AMAD a la capacité d'identifier les variantes de profil de consommation en ressources de VMs attaquantes présentant divers pentes de variation de consommation.

3.6.1.2 Variation de la durée de la période de fluctuation

Dans cette section, nous évaluons la robustesse de l'algorithme d'identification des VMs fluctuantes vis-à-vis d'une variation de la période de fluctuation des VMs attaquantes. La manipulation de la période de fluctuation des VMs attaquantes impacte la fréquence d'occurrence des migrations de VMs occasionnées par l'attaque. La période de fluctuation du profil de base des VMs attaquantes était fixée à 60 minutes jusqu'à présent dans nos expérimentations. Pour cette évaluation, nous la faisons maintenant varier de 20 minutes à 90 minutes par pas de 10 minutes, avec pour objectif d'identifier d'éventuels points de rupture au delà desquels les VMs attaquantes ne sont plus identifiées comme telles par la phase 1 d'analyse d'AMAD.

Les figures 3.11(a) et 3.11(b) illustrent la variation de l'utilisation mémoire, respectivement pour le profil de plus courte période (20 minutes) et celui de plus longue période (90 minutes).

Les figures 3.12(a) et 3.12(b) illustrent respectivement la variation des moyennes des valeurs de précision et celle des valeurs de rappel, obtenues lors de l'application de l'algorithme d'identification des VMs fluctuantes pour la mémoire et le CPU, en fonction de la période de fluctuation

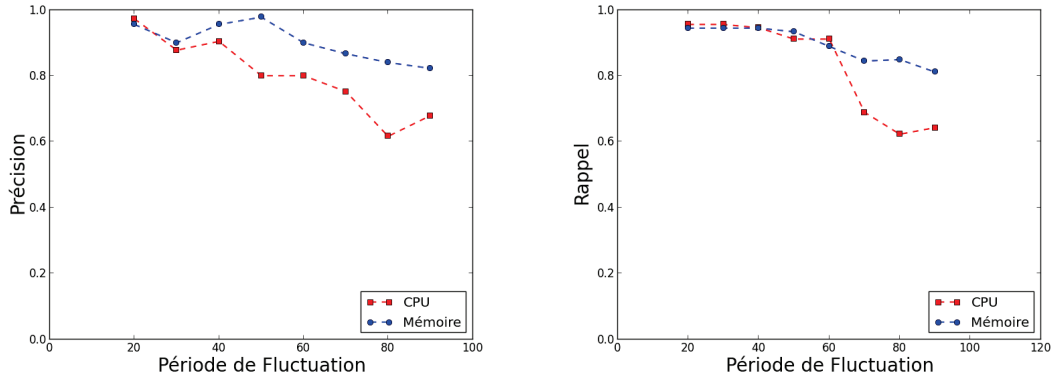


(a) Profil de VM attaquante de plus courte période (20 minutes)

(b) Profil de VM attaquante de plus longue période (90 minutes)

FIGURE 3.11 – consommation mémoire de VM attaquante - Variation de la période

du profil de consommation des VMs attaquantes.



(a) Impact de la variation de la période de fluctuation sur la précision

(b) Impact de la variation de la période de fluctuation sur le rappel

FIGURE 3.12 – Impact de la variation de la période de fluctuation sur la précision et le rappel

Nous pouvons observer sur ces deux figures que l'algorithme d'identification des VMs fluctuantes montre de bonnes performances de détection en termes de *précision* et de *rappel*, à la fois pour la mémoire et pour le CPU jusqu'à ce que la période de fluctuation atteigne 60 minutes. A partir de 70 minutes, nous observons, pour les valeurs de CPU essentiellement, que les performances de détection sont négativement impactées atteignant des valeurs inférieures à 0.7. Cela est due au fait que le nombre de cycles de fluctuation sur la durée de l'expérimentation devient trop faible lorsque la période devient élevée, rendant les profils de consommation des VMs attaquantes trop semblables à ceux des VMs normales ; les pics de consommation présents dans les profils des VMs normales sont aussi vus comme une forme de fluctuation par

AMAD compte tenu du mode de calcul de l'indice de fluctuation. S'agissant de la précision, la dégradation de la performance de détection n'est pas très préjudiciable puisque les VMs normales fluctuantes retenues à tort en phase 1, sont vraisemblablement rejetées en phase 2 d'analyse d'AMAD faute de corrélation suffisante. La dégradation du rappel avec l'augmentation de la période est plus critique puisqu'elle est due à un défaut de détection de certaines VMs fluctuantes attaquantes du fait du contexte fluctuant : les VMs fluctuantes de périodicité de 70 minutes ne présentent que deux basculements de consommation sur la durée d'expérimentation de 180 minutes.

En conclusion, les performances d'identification des VMs attaquantes sont impactées négativement, dès lors que la période de fluctuation du profil de consommation des VMs attaquantes, en relatif par rapport à la longueur de la fenêtre de rétro-analyse, devient grande ne laissant apparaître que peu de basculements de consommation dans le profil. Néanmoins, si le nombre de basculements de consommation est faible sur la durée de la fenêtre d'analyse (définie de manière auto-adaptative pour couvrir la durée de l'attaque), cela signifie que l'attaque provoque un nombre limité de migrations et n'est donc pas très préjudiciable.

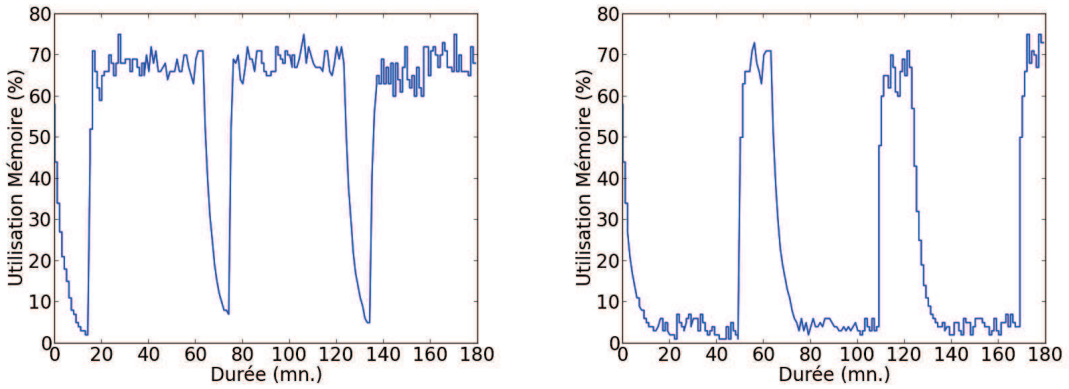
3.6.1.3 Variation du rapport cyclique

La dernière variante de profil de consommation de VM attaquante que nous considérons consiste à faire varier le rapport cyclique du profil de consommation : on introduit une asymétrie dans le profil, la durée pendant laquelle la VM attaquante consomme une faible quantité de ressources n'est plus égale à la durée pendant laquelle elle consomme une forte quantité de ressources. Maintenant la période de 60 minutes inchangée, nous faisons varier la durée de faible consommation de 10 minutes à 50 minutes par pas de 10 minutes, pour définir cinq profils de consommation de VM attaquante.

Les figures 3.13(a) et 3.13(b) illustrent la variation de l'utilisation mémoire respectivement pour le profil de plus petit rapport cyclique (10 minutes de faible consommation sur une période de 60 minutes) et celui de plus fort rapport cyclique (50 minutes de faible consommation sur une période de 60 minutes).

Les figures 3.14(a) et 3.14(b) illustrent respectivement la variation des moyennes des valeurs de *précision* et celles de *rappel* obtenues lorsque l'algorithme d'identification des VMs fluctuantes d'AMAD est appliqué sur la mémoire et sur le CPU, en fonction du rapport cyclique du profil de consommation des VMs attaquantes.

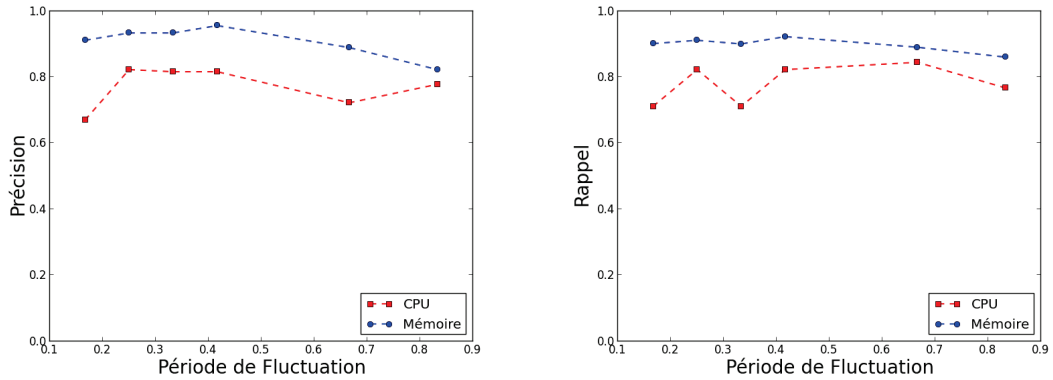
Nous pouvons observer à partir de ces deux figures que les moyennes de *précision* et de *rappel* restent relativement stables avec la variation du rapport cyclique du profil de consommation des VMs attaquantes. Cela est dû au fait que le nombre de basculements d'utilisation en ressources présents dans les variantes de profil des VMs attaquantes reste inchangé lorsque le rapport cyclique est modifié.



(a) Rapport cyclique de fluctuation avec 10 minutes de consommation faible

(b) Rapport cyclique de fluctuation avec 50 minutes de consommation faible

FIGURE 3.13 – Profils de consommation de VM attaquante - Variation du rapport cyclique pour une période constante de 60 minutes



(a) Impact de la variation du rapport cyclique sur la précision

(b) Impact de la variation du rapport cyclique sur le rappel

FIGURE 3.14 – Impact de la variation du rapport cyclique sur la précision et le rappel

En conclusion, la variation du rapport cyclique de fluctuation n'impacte pas la performance de détection de l'algorithme d'identification des VMs fluctuantes d'AMAD (mémoire et CPU).

Les résultats de ces évaluations nous permettent de conclure que l'algorithme AMAD d'identification des VMs fluctuantes montre de bonnes propriétés de robustesse pour l'identification de divers profils de consommation en ressources susceptibles de réaliser l'attaque par migrations intempestives de VMs.

Dans la section suivante, nous cherchons à évaluer la robustesse de la phase 1 d'analyse d'AMAD lorsque le niveau de fluctuation du contexte d'exécution des VMs contrôlées par l'attaquant, augmente. On rappelle que le contexte d'exécution des VMs attaquantes est constituée,

par définition, de l'ensemble des VMs du cluster co-résidentes avec les VMs attaquantes.

3.6.2 Robustesse vis-à-vis de la fluctuation du contexte

L'algorithme des k-moyennes appliqué lors de la phase 1 réalise le partitionnement des VMs en se basant sur le calcul de distances euclidiennes entre les vecteurs d'indices de fluctuation des VMs. Ces distances sont naturellement influencées par le niveau de fluctuation de l'ensemble des VMs à partitionner et pas seulement par celui des VMs attaquantes fluctuant de manière coordonnée.

Nous souhaitons mesurer l'influence de la présence de VMs fluctuantes dans le contexte d'exécution et vérifier que les VMs attaquantes restent, dans ces conditions, identifiables par AMAD ; le risque étant que la phase 1 de l'algorithme AMAD d'identification des VMs fluctuantes, ne cesse alors de traiter les VMs attaquantes comme fluctuantes.

Nous manipulons le niveau de fluctuation du contexte d'exécution des VMs attaquantes, au travers de la manipulation de trois paramètres, i) la fréquence d'apparition de pics de consommation dans les profils de consommation de VMs du contexte ii) l'amplitude de pics de consommation introduits dans les profils de consommation de VMs du contexte d'exécution, iii) la part de VMs fluctuantes dans le contexte d'exécution.

Dans nos expérimentations, nous manipulons la fluctuation de 10 profils de consommation de VMs non attaquantes, appelées VMs fluctuantes normales ; ces VMs contribuant à la fluctuation du contexte d'exécution des 10 VMs attaquantes.

Nous présentons d'abord ci-dessous les résultats d'évaluation obtenus avec la manipulation de la fréquence d'apparition des pics de consommation, suivis des résultats d'évaluation obtenus avec la manipulation de l'amplitude de ces pics.

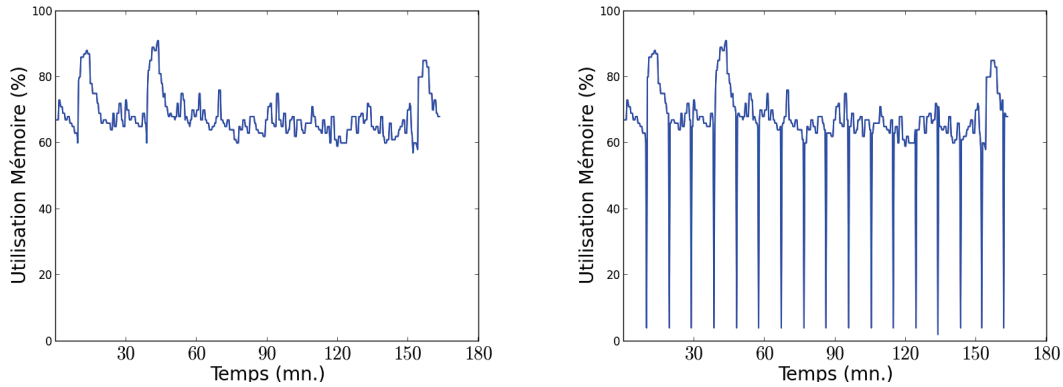
3.6.2.1 Fréquence d'apparition des pics de consommation

Dans cette section, nous évaluons la robustesse de la première phase d'analyse d'AMAD pour l'identification des VMs fluctuantes *attaquantes* en présence de VMs *normales* à profil de consommation fluctuant pouvant résulter d'une activité non malveillante générée par exemple par l'exécution d'une application sporadique.

Pour effectuer cette évaluation, nous utilisons les 50 traces de consommation collectées lors de nos précédentes expérimentations – décrites dans la section 3.4.3. Pour chaque expérimentation, 10 profils de consommation de VMs sont sélectionnés de façon aléatoire et leur niveau de fluctuation est manipulé par l'introduction de pics de consommation dont l'amplitude est maintenue constante d'une expérimentation à l'autre avec une valeur d'utilisation de la ressource de 70% et dont la fréquence d'apparition varie d'une expérimentation à l'autre : la durée séparant deux pics de consommation successifs varie de 10 minutes à 120 minutes par pas de 10 minutes

d'une expérimentation à l'autre. Nous mesurons alors les moyennes des valeurs de *précision* et de celles de *rappel* d'identification des VMs fluctuantes attaquantes, à l'issue de la phase 1 d'AMAD.

Les figures 3.15(a) et 3.15(b) illustrent respectivement les variations de l'utilisation mémoire d'une VM normale de référence et de cette même VM après bruitage par l'introduction de pics de consommation d'amplitude 70% toutes les 10 minutes.



(a) Profil de consommation mémoire d'une VM normale - avant bruitage

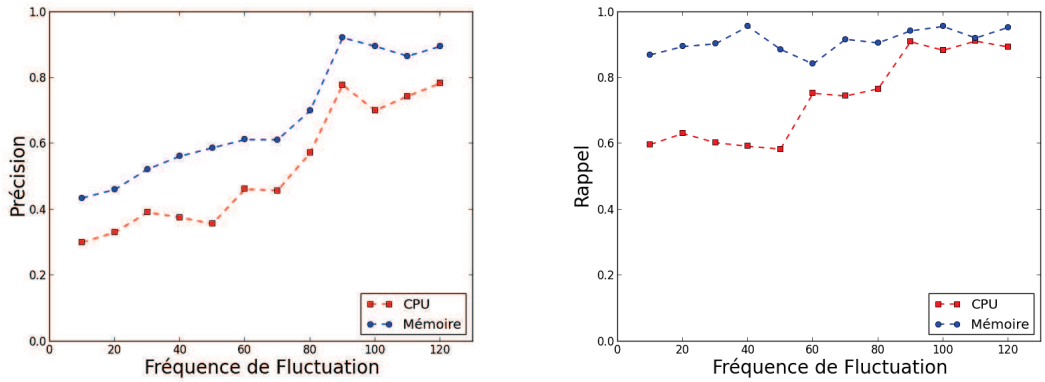
(b) Introduction de pics de consommation toutes les 10 minutes dans le profil d'une VM normale

FIGURE 3.15 – Manipulation de la fluctuation du contexte d'exécution des VMs attaquantes - Manipulation de la fréquence

Les figures 3.16(a) et 3.16(b) illustrent, à la fois pour la mémoire et le CPU, respectivement la variation des moyennes des valeurs de précision et rappel obtenues suite à la phase 1 d'AMAD, en fonction de la fréquence d'apparition des pics de consommation dans les profils des VMs normales du contexte.

Nous pouvons observer sur la figure 3.16(a) que la précision de détection de l'algorithme d'identification des VMs fluctuantes est impactée par le niveau de fluctuation du contexte d'exécution des VMs : lorsque la fréquence d'apparition des pics de consommation correspond à des périodes inférieures à 90 minutes, la précision de l'algorithme est faible, de l'ordre de 0.4 pour le CPU et de 0.6 pour la mémoire. Cela est dû au fait que les VMs normales fluctuantes deviennent aussi fluctuantes que les VMs attaquantes fluctuantes qui doivent être détectées par la phase 1 d'AMAD, ce qui provoque leur assimilation à des VMs attaquantes lors de la première phase d'analyse d'AMAD. Le nombre de faux positifs s'en trouve alors augmenté et la précision dégradée.

Sur la figure 3.16(b), nous pouvons observer que les valeurs de rappel pour la mémoire restent aussi satisfaisantes que celles obtenues lors de nos précédentes expérimentations. Cela démontre que les VMs attaquantes sont effectivement détectées par le module d'identification des VMs fluctuantes, malgré le niveau de fluctuation élevé du contexte : les nombres de vrais



(a) Impact de la fréquence de fluctuation du contexte sur la précision

(b) Impact de la fréquence de fluctuation du contexte sur le rappel

FIGURE 3.16 – Impact de la fréquence d'apparition de pics de consommation sur la précision et le rappel

positifs et de faux négatifs ne sont pas impactés et donc le rappel non plus. Sur la même figure, nous pouvons observer que la détection pour le CPU est impactée lorsque le niveau de fluctuation du contexte est élevé mais reste tout de même satisfaisante (de l'ordre de 0.7).

En conclusion, le niveau de fluctuation du contexte est un paramètre impactant l'exactitude de détection de la phase 1 d'AMAD chargée d'identifier les VMs fluctuantes. La précision de détection des VMs fluctuantes attaquantes est très impactée en phase 1, ce qui justifie d'appliquer la phase 2 d'AMAD chargée d'identifier les VMs coordonnées parmi les VMs fluctuantes identifiées en phase 1, pour pouvoir filtrer les VMs fluctuantes normales. Les différents résultats obtenus pour la mesure du rappel démontrent qu'AMAD est un système robuste puisqu'il identifie les VMs attaquantes comme telles, malgré le bruit introduit dans le contexte.

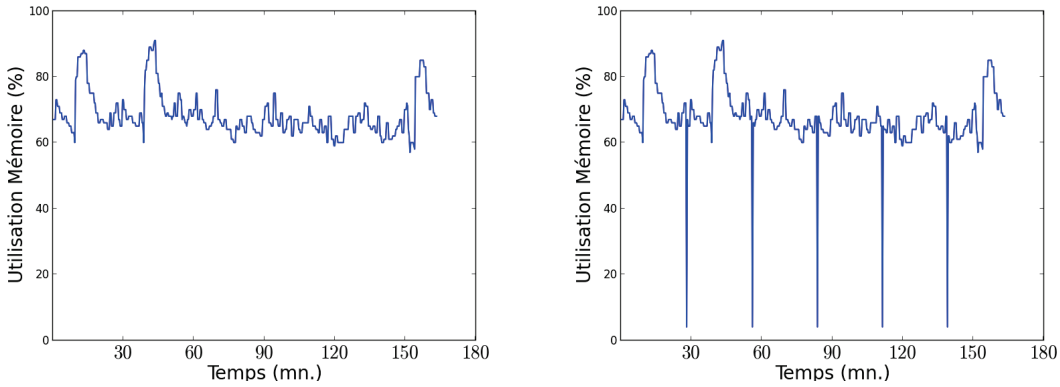
3.6.2.2 Variation de l'amplitude des pics de consommation

Dans cette section, nous évaluons la robustesse de l'algorithme d'identification des VMs fluctuantes vis-à-vis d'une variation de l'amplitude des pics de consommation en ressources introduits dans les VMs normales du contexte. L'objectif de cette évaluation est double : il consiste d'une part à déterminer à partir de quel niveau d'amplitude de fluctuation du contexte, les VMs normales sont considérées par la première phase d'analyse d'AMAD comme étant des VMs fluctuantes et d'autre part à évaluer si les VMs fluctuantes attaquantes sont toujours vues comme étant fluctuantes en présence de VMs normales fluctuantes dans le contexte.

Les conditions d'expérimentation sont les mêmes que précédemment, avec cette fois une variation du niveau de fluctuation du contexte obtenue par l'introduction de pics de consommation avec une fréquence d'apparition constante à 30 minutes et une amplitude de consommation des

pics variant entre 10% et 70% par pas de 10% d'une expérimentation à une autre.

Les figures 3.17(a) et 3.17(b) illustrent respectivement les variations de l'utilisation mémoire d'une VM normale de référence et de cette même VM après bruitage par l'introduction de pics de consommation toutes les 30 minutes avec une amplitude de 70%.



(a) Profil de consommation mémoire de la VM normale de référence

(b) Introduction de pics de consommation d'amplitude 70% dans le profil d'une VM normale

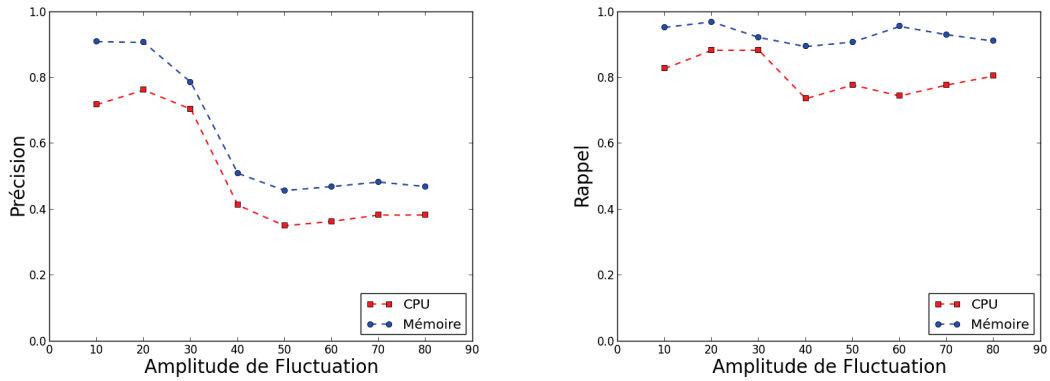
FIGURE 3.17 – Manipulation de la fluctuation du contexte d'exécution des VMs attaquantes - Manipulation de l'amplitude

Les figures 3.18(a) et 3.18(b) illustrent, à la fois pour la mémoire et le CPU, respectivement la variation des moyennes des valeurs de précision et rappel obtenues lors de l'application de l'algorithme d'identification des VMs attaquantes fluctuantes, en fonction de l'amplitude des pics de consommation dans les profils des VMs normales du contexte.

Nous pouvons observer sur la figure 3.18(a) que la fluctuation du contexte impacte la précision de l'algorithme d'identification des VMs fluctuantes, à la fois pour la mémoire et pour le CPU, lorsque l'amplitude des pics de consommation introduits dans le contexte atteint une valeur supérieure à 40%. Cela signifie que l'algorithme d'identification des VMs fluctuantes considère les VMs ayant des pics de consommation supérieurs à 40% de variation comme étant des VMs fluctuantes.

Sur la figure 3.18(b), nous pouvons observer que les valeurs de rappel pour la mémoire et pour le CPU restent très stables, ce qui démontre que l'algorithme d'identification des VMs fluctuantes identifie correctement les VMs attaquantes comme étant fluctuantes, malgré la fluctuation du contexte. Nous pouvons également observer sur cette figure que l'application de l'algorithme d'analyse sur la mémoire est plus robuste en comparaison avec son application sur le CPU ; cela est dû au fait que la consommation mémoire varie moins instantanément que la consommation CPU.

En conclusion, l'impact de l'amplitude des pics de consommation introduits pour augmenter le niveau de fluctuation du contexte est similaire à celui évalué dans la section ci-dessus consacrée



(a) Impact de l'amplitude de fluctuation sur la précision de AMAD

(b) Impact de l'amplitude de fluctuation sur le rappel de AMAD

FIGURE 3.18 – Impact de l'amplitude des pics de consommation des VMs normales du contexte sur la précision et le rappel

à la fréquence : d'une part, la précision de détection de la phase 1 de l'algorithme d'identification des VMs attaquantes d'AMAD est impactée, mais la phase 2 d'AMAD permet de neutraliser ce défaut en filtrant les VMs fluctuantes normales non coordonnées du contexte et d'autre part, AMAD est un système robuste avec une valeur de rappel peu impactée par la variation de la fluctuation du contexte.

3.7 Conclusion

Dans ce chapitre, nous avons présenté AMAD, un système qui permet de superviser les décisions de migration de VMs au sein d'un cluster. AMAD est à notre connaissance le premier système de détection d'anomalies d'exécution dans le cloud dédié à la supervision du fonctionnement d'un système de gestion dynamique des ressources. AMAD utilise un algorithme robuste d'identification des VMs à profils de consommation en ressources fluctuants et corrélés dans le temps, qui sont considérées par AMAD comme étant les VMs à l'origine du nombre anormal de migrations provoqué par l'attaque par migrations intempestives de VMs. L'algorithme d'identification des VMs fluctuantes implémenté par AMAD, peut être utilisé en lui-même dans un domaine plus large que celui de la sécurité, tel que celui de la gestion dynamique des ressources dans le cloud.

Nous avons implémenté AMAD sur une plate-forme d'expérimentation en laboratoire et avons utilisé pour l'évaluation d'AMAD des traces de consommation collectées sur des clouds réels. Les expérimentations menées pour l'évaluation de la performance de détection d'AMAD montrent qu'AMAD identifie correctement les VMs attaquantes que nous avons intentionnellement injectées dans nos expérimentations.

L'approche de supervision réactive d'AMAD protège les infrastructures cloud de l'attaque par migrations intempestives, sans avoir à impacter la conception du système de gestion dynamique des ressources. AMAD est parfaitement indépendant du système de gestion dynamique des ressources puisqu'il agit comme une couche de supervision indépendante qui exploite des informations relevant du niveau de l'infrastructure virtualisée uniquement.

L' intérêt d'identifier les VMs fluctuantes est de permettre de leur appliquer des politiques de placement particulières, dans le but de réagir à une attaque par migrations intempestives de VMs, par exemple en limitant la capacité des VMs à influencer le déséquilibrage de charge du cluster grâce à l'application de règles d'anti-affinités de VMs venant contraindre le système de gestion des ressources à intégrer des considérations pour la sécurité.

L'intérêt principal de l'approche adoptée est de permettre l'amélioration de la robustesse d'un système de gestion dynamique des ressources tout en préservant sa réactivité, ce qui permet l'optimisation de la consolidation des ressources au sein de l'infrastructure et, en conséquence, l'amélioration de la performance économique des offres cloud. En effet, le système de supervision AMAD détecte les événements de migrations en excès résultant d'une trop grande réactivité aux profils de consommation de la part du système de gestion dynamique des ressources. Grâce à cela, il peut réagir à ces événements en contraignant, uniquement à l'occasion de ces événements, le système de gestion dynamique à réviser sa réactivité, garantissant ainsi sa robustesse d'une part et un fonctionnement optimisant les taux d'utilisation des ressources en l'absence d'anomalie d'autre part.

Dans le chapitre suivant, nous revenons sur les traces que nous avons collectées dans des clouds privés d'Orange pour les analyser plus en détails et pour définir des métriques de quantification de la fluctuation de la consommation en ressources des VMs.

Travaux de l'auteur sur cette thématique

— Méthode de détection des attaques par migrations intempestives de machines virtuelles . **Kahina Lazri**, Sylvie Laniepce, N° dépôt : 14 50763, déposé le 30/01/14.

— AMAD : Resource Consumption Profile-Aware Attack Detection in IaaS Cloud. **Kahina Lazri**, Sylvie Laniepce, Haiming Zheng, Jalel Ben-Othman. 7th IEEE/ACM International Conference on Utility and Cloud Computing, **UCC'14**. Décembre 2014, Londres, Royaume-Uni.

Chapitre 4

Caractérisation de la consommation de VMs d'un cloud privé

Dans ce chapitre, nous présentons une analyse de traces de consommation en ressources collectées sur un cloud privé d'Orange. Nous présentons d'abord une étude statistique qui met en évidence les pratiques des utilisateurs sur des clouds réels en termes de configurations de ressources des VMs, puis nous considérons l'utilisation en ressources de ces VMs. Les premiers résultats de cette étude révèlent que les utilisateurs surestiment fortement les quantités de ressources nécessaires à l'exécution de leurs VMs. La deuxième partie de l'analyse de ces traces traite de l'étude de la fluctuation de la consommation en ressources des VMs, l'application de l'algorithme AMAD d'identification des VMs fluctuantes sur ces traces permet d'estimer que 20% des VMs ont une consommation fluctuante en mémoire et 17% ont une consommation fluctuante en CPU.

Sommaire

4.1 Introduction

DANS le chapitre 3, nous avons utilisé des traces de consommation en ressources de VMs s'exécutant sur cloud privé d'Orange pour l'évaluation de la performance de détection d'AMAD. A défaut de pouvoir exécuter AMAD in situ dans un cloud réel, ces traces de consommation de VMs nous ont permis d'évaluer la robustesse du système AMAD dans un contexte d'exécution issu d'un cloud réel.

L'émergence des plate-formes cloud donne lieu à de multiples interrogations concernant l'évolution de l'exploitation opérationnelle de ce type d'infrastructure d'hébergement. Les caractéristiques attribuées aux plate-formes cloud telles que la complexité, l'hétérogénéité ou encore la dynamicité sont autant de propriétés qui nécessitent de mettre en place de nouveaux outils adaptés à ces infrastructures. Disposer de traces de consommation issues du cloud permet de reconstituer le fonctionnement de ces plate-formes pour mieux analyser leur évolution et comprendre les profils de consommation en ressources des VMs hébergées, ce qui permettra de mieux définir des politiques de dimensionnement autonomiques des VMs et des centres de données.

L'optimisation de la consolidation des ressources physiques grâce au partage des ressources entre plusieurs VMs est la vocation première de la virtualisation. La maximisation de cette consolidation des ressources physiques dans le respect des engagements en qualité de services passés entre l'opérateur et les clients, est un objectif majeur poursuivi par les fournisseurs cloud pour optimiser la rentabilité économique de leurs infrastructures. Définir des taux de remplissage optimaux des serveurs au regard de la dualité consolidation/performance représente le défi à relever pour être différenciant et concurrentiel.

Plusieurs études révèlent que le bénéfice réel du cloud pour les utilisateurs réside principalement dans l'application d'un payement à l'usage [118, 119]. Dans ce contexte, la question posée est de savoir si, au delà de l'effet de mode que le cloud a généré, ce modèle peut définitivement être durable, si les clients n'y trouvent pas une attractivité économique réelle. Il est d'importance majeure pour la recherche qui poursuit la mise en place de systèmes de plus en plus autonomiques de s'appuyer sur l'analyse de l'état d'exploitation des infrastructures cloud afin de confronter la vision de la recherche à celle du terrain. Ce n'est qu'avec une compréhension fine de l'état d'exploitation des environnements actuels que l'on peut élaborer des propositions technologiques levant les verrous auxquels se confronte le besoin d'optimisation de l'exploitation des centres de données.

Dans ce chapitre, nous nous intéressons à établir et à analyser des statistiques qui qualifient les traces de consommation en ressources de VMs collectées sur cloud privé d'Orange. Nos résultats montrent que les clients surestiment fortement les quantités de ressources nécessaires à l'exécution de leur VMs, résultant dans de faibles moyennes de taux d'utilisation en ressources

des VMs. Notre étude de la fluctuation de la consommation en ressources des VMs montre qu'il est possible d'identifier automatiquement des VMs fluctuantes pour une meilleure exploitation des ressources au sein des clouds virtualisés actuels.

Contributions du chapitre

Dans ce chapitre, nous détaillons les contributions suivantes :

— Analyse statistique des configurations et consommations mémoire et CPU de VMs s'exécutant sur cloud privé d'Orange,

— Expérimentations pour analyser des métriques VMware de quantification de la consommation mémoire par les VMs, telle que vue par l'hyperviseur,

— Analyse de la fluctuation des profils de consommation en ressources des VMs.

Organisation du chapitre

Ce chapitre est organisé comme suit. La section 4.2 présente l'état de l'art relatif à l'analyse des traces de consommation de VMs dans le cloud. La section 4.3 livre une image globale des VMs analysées dans le cadre de cette étude. La section 4.4 analyse la consommation en ressources des VMs sur une durée de 28 jours. La section 4.5 livre une analyse de la fluctuation de la consommation en ressources des VMs et propose une première approche de quantification de la fluctuation de la consommation en ressources. Enfin la section 4.6 conclut ce chapitre.

4.2 État de l'art analyse de traces

L'analyse de traces de consommation de VMs s'exécutant sur clouds réels a connu un intérêt croissant lors de ces dernières années car disposer de telles traces de consommation est crucial pour la communauté scientifique pour permettre la validation de travaux de recherche dans des contextes d'exécution représentatifs de la réalité. L'exploitation des traces peut se faire soit en rejouant les traces collectées (comme effectué pour la validation d'AMAD dans le chapitre 3) ou bien au travers de la construction de modèles mathématiques caractérisant les consommations des VMs observées pour générer des workloads synthétiques.

A ce jour, très peu de traces de consommation en ressources ont été rendues disponibles par les fournisseurs d'infrastructures clouds.

Google a publié deux traces de consommation cloud. La première a été rendue publique en décembre 2009, elle consiste en des relevés de consommation de clusters de calcul "*Google Compute Engine*" couvrant une durée de 7 heures avec des relevés de consommation effectués toutes les 5 minutes [120]. La seconde trace a été rendue publique en novembre 2011, elle

consiste en des relevés de consommation de plus de 1200 serveurs couvrant une durée de 30 jours avec des relevés effectués toutes les 5 minutes [121].

Dans [122], Reiss et al. décrivent la sémantique, le format des structures de données ainsi que les anomalies et les informations manquantes pour cette deuxième version de trace de Google. La trace contient 55 272 *jobs*, un job étant une unité de travail soumis pour exécution par un utilisateur. Chaque job est subdivisé en un nombre variable de tâches allant d'une seule tâche jusqu'à des centaines de tâches (la trace contient 1 405 572 tâches). Ces tâches sont ordonnancées de façon parallèle et chacune d'entre elles peut être décrite par plusieurs paramètres tels que la quantité maximum de ressources autorisée pour cette tâche, contraintes de placement telles que les règles d'affinité ou encore de règles de compatibilité matérielle et logicielle. En moyenne, près de 20 paramètres sont associés aux tâches d'un seul job [123].

Yahoo ! a de son côté publié des traces de consommation [124] de son cluster Hadoop M45, cependant les traces de Yahoo ! sont peu documentées, ce qui rend leur exploitation difficile. Par conséquent, très peu de travaux sont publiés autour de ces traces.

D'autres travaux analysent des traces de consommation privées puis livrent les résultats de leurs études, sans pour autant publier les données analysées. Bien qu'il soit difficile dans ce cas de reproduire l'analyse fournie ou de l'utiliser pour la validation d'autres travaux, il demeure utile de disposer de retours d'expérience concernant les pratiques ainsi que les limites d'exploitation rencontrées au sein des infrastructures clouds. Ce type d'analyse est souvent réalisé par les fournisseurs de clouds qui disposent des données mais qui, pour des raisons de confidentialité vis-à-vis de leur client et de la concurrence [125], ne souhaitent pas les rendre publiques en tant que telles.

Nous catégorisons dans ce qui suit les travaux de l'état de l'art selon le type de traces qu'ils analysent, en deux catégories : les analyses effectuées sur des traces publiques et celles effectuées sur des traces privées.

4.2.1 Analyse de traces publiques

De très nombreux travaux de recherche ont exploité la trace publiée par Google en 2011, car celle-ci couvre une durée et un nombre de serveurs conséquents et est parfaitement documentée.

Un des objectifs poursuivi par l'analyse de traces est d'identifier l'existence de modèles d'exécutions communs [126][127] tels que : des relations entre des jobs soumis par un même utilisateur, des relations entre les profils de consommation de plusieurs VMs exécutant un même type d'application ou une périodicité dans les profils de consommation en ressources des VMs. Certains travaux [128] proposent des modèles pour la caractérisation de la consommation en ressources des VMs dans le but de proposer des politiques d'optimisation de l'exploitation des

infrastructures d'une part et pour motiver des propositions relatives à la gestion autonomique de ressources d'autre part.

Dans [129], Mishra et al. cherchent à caractériser les tâches s'exécutant sur quatre clusters de calcul de Google. L'objectif de leur analyse est d'identifier, selon une approche de partitionnement, les tâches ayant des caractéristiques communes pour optimiser l'exploitation des ressources et produire des informations d'entrées pour l'ordonnanceur des tâches (placement des tâches sur les hôtes). La classification effectuée révèle l'existence de deux types de tâches principalement : celles dont la durée d'exécution est inférieure à 30 minutes, qui représentent la majorité des tâches, et celles ayant une durée d'exécution supérieure à 18 heures. Ces deux classes de tâches sont ensuite partitionnées à leur tour en fonction des quantités de ressources qu'elles consomment ; il est observé que les tâches de durées les plus longues sont celles qui consomment les plus fortes quantités de ressources par unité de temps.

Dans [72], les auteurs définissent la notion de workload ('charge de travail') comme étant un ensemble de tâches appartenant à un même utilisateur considérant ainsi l'utilisateur comme paramètre dans l'étude du profil de consommation global du workload. L'étude montre par exemple qu'une très faible proportion des utilisateurs est à l'origine d'une majorité des tâches présentes dans la trace et que la majorité des utilisateurs est individuellement à l'origine d'une très faible proportion (0,1%) des tâches de la trace. Ces travaux montrent également qu'il existe une corrélation très forte entre les profils de consommation en ressources des différentes tâches exécutées par un même utilisateur et que 90% des utilisateurs pour lesquels la consommation en ressources des VMs a été observée surestiment largement les besoins en ressources de leurs VMs. Les travaux proposent aussi un simulateur de workloads réels qui est construit sur la base de paramètres extraits de cette trace.

Le caractère hétérogène des workloads présents dans la trace Google 2011 a été relevé par plusieurs travaux [130] [72]. Cette hétérogénéité provient en partie de l'hétérogénéité des caractéristiques des tâches telles que définies par les utilisateurs : priorité associée à chaque tâche (entre 0 et 11), temps d'exécution maximal des tâches ou encore types de ressources sollicitées par chaque tâche. La majorité des travaux relèvent toutefois que les quantités de ressources consommées par les workloads sont stables sur la durée de la collecte.

Google permet aux utilisateurs d'associer des contraintes aux calculs qu'ils soumettent, soient des contraintes d'incompatibilité hardware qui se traduisent par l'impossibilité d'exécuter une tâche sur une machine physique, soient des contraintes de performance. Dans [131], les auteurs s'intéressent à l'impact des contraintes de placement appliquées aux VMs, sur leurs performances d'exécution. Cette étude montre que ces contraintes de placement engendrent des effets négatifs sur l'exécution des tâches, tels que l'augmentation du temps d'attente des tâches pour accéder aux ressources qui varie d'un facteur allant de 2 à 6, ce qui représente une dizaine de minutes d'attente supplémentaire pour l'exécution de ces tâches. Ces travaux proposent une

méthodologie pour générer des workloads synthétiques intégrant les impacts de contrainte de placement découverts grâce à l'analyse de traces effectuée.

Enfin, une comparaison entre les workloads de la trace 2011 de Google et ceux issus d'une infrastructure de calcul HPC/Grid a été réalisée dans [132]. Les comparaisons sont effectuées sur des critères liés à la charge des hôtes, et sur des critères liés aux jobs et tâches soumis tels que la durée d'exécution des tâches ou la fréquence et la quantité de ressources consommées. Les résultats de cette comparaison entre l'infrastructure Google et celle HPC/Grid étudiée permettent de démontrer une gestion des ressources plus efficace (durées d'exécution des tâches plus courtes, temps d'attente plus courts, etc...) avec de meilleurs taux d'utilisation des ressources des hôtes côté Google.

4.2.2 Analyse de traces privés

Les analyses effectuées sur traces privées sont, la plupart du temps, menées à des fins de validation d'hypothèses ou de propositions de solutions. La grande majorité des travaux existants dans le domaine de l'analyse des consommations en ressources au sein des clouds est réalisé par IBM.

Dans [133], les auteurs présentent une analyse de la consommation en ressources de VMs et de serveurs physiques s'exécutant sur cloud privé virtuel d'IBM. Les serveurs et VMs dont la consommation a été mesurée appartiennent à 6 entreprises différentes et s'exécutent sur deux centres de données répartis sur différents sites eux mêmes localisés dans différents pays. Le premier centre de données est constitué de 393 serveurs et le deuxième de 3681 serveurs. Cette analyse s'intéresse à l'évolution temporelle de la consommation en ressources des VMs et des serveurs sur une durée qui couvre 12 mois. L'étude réalisée considère exclusivement l'utilisation CPU. Elle se focalise sur l'étude de la périodicité des profils de consommation de VMs appartenant à une même entreprise, identifie les événements pouvant déclencher la ré-allocation des quantités de ressources allouées aux VMs et discute la nécessité de mettre en oeuvre des politiques de management autonomique pour l'approvisionnement dynamique des ressources aux VMs.

Dans d'autres travaux [134], IBM utilise une trace de consommation d'un cloud privé IBM, pour étudier la surestimation des ressources par les utilisateurs cloud et son impact en termes de sous-utilisation des ressources des serveurs physiques. Après avoir rappelé l'importance du sur-engagement des ressources pour l'optimisation économique des infrastructures cloud, les auteurs considèrent les workloads à profil de consommation fluctuant de cette trace, pour définir des seuils de surengagement de ressources, établis en fonction du type de workload. L'approche retenue propose de surengager les ressources des hôtes exécutant des VMs à profil de consommation fluctuant car, statistiquement, les maximums de consommation de ces VMs

se compensent étant donné qu'elles ne consomment pas continûment leurs ressources, mais seulement de façon occasionnelle.

C'est l'analyse rapportée dans [114] qui couvre le plus large nombre de VMs et de serveurs. Elle est effectuée sur 90k VMs s'exécutant sur 8k serveurs physiques, couvrant une durée d'analyse de 19 jours avec des collectes effectuées toutes les 15 minutes. Cette étude se focalise sur les politiques de placement des VMs au sein des centres de données, en analysant la distribution de la mémoire et du CPU configurés aux VMs, leurs répartitions sur les hôtes et les cycles de vie des VMs (mise sous tension, mise hors tension, migrations des VMs). Les résultats de cette étude pointent encore une fois le problème de la sur-estimation des besoins en ressources des VMs par les utilisateurs. L'étude conclut que le potentiel d'optimisation des ressources, rendu possible avec la virtualisation, reste peu exploité : consolidation des serveurs, sur-engagement des ressources physiques, élasticité dans l'allocation des ressources ou encore migration de VMs.

Dans [135], les auteurs tentent de modéliser la distribution de l'utilisation CPU des VMs s'exécutant sur cloud privé d'IBM, en tenant compte de la performance applicative observée par les VMs. Une description des distributions de la consommation CPU des workloads est d'abord établie, une modélisation avec plusieurs modèles de files d'attente est ensuite réalisée. L'analyse montre des résultats prometteurs quant à l'utilisation d'une file d'attente de type M/M/1/k [136] quant à son utilisation pour la prédiction de l'utilisation future en CPU des VMs.

La section suivante présente la trace Orange que nous analysons par la suite pour étudier les taux d'utilisation CPU et mémoire des VMs ainsi que leur fluctuation.

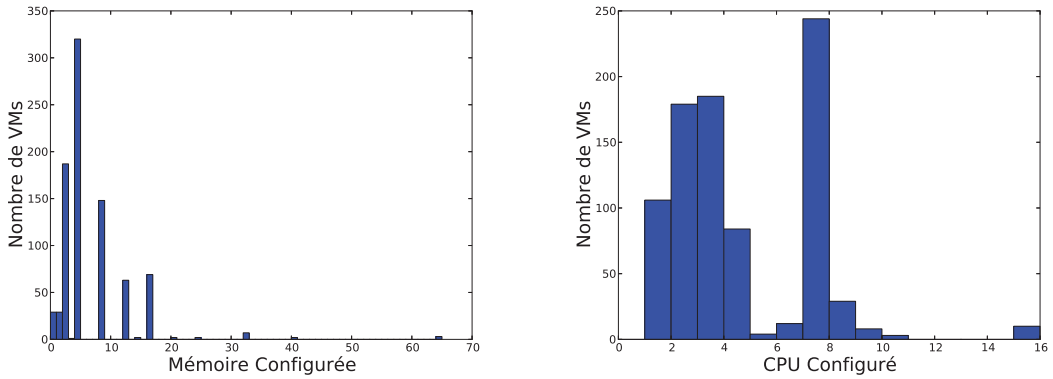
4.3 Description des données de la trace Orange

La trace Orange que nous avons collectée contient des historiques de consommation en ressources mémoire et CPU de VMs d'un centre de données en production chez Orange, constitué de 863 VMs réparties sur 105 machines physiques, réparties à leur tours, sur 22 clusters. L'infrastructure de ces centres de données est virtualisée avec la suite VMware vSphere (hyperviseur ESXi et couche de management vCenter), les serveurs physiques sont regroupés en clusters qui constituent des pools de ressources indépendants les uns des autres. Comme nous nous intéressons aux profils de consommation de VMs uniquement, qui sont indépendants de la hiérarchisation faite de la gestion des ressources, l'analyse livrée dans ce chapitre considère l'ensemble des VMs du centre de données comme appartenant à un pool de ressources unique.

Les figures 4.1(a) et 4.1(b) fournissent une vue d'ensemble des configurations mémoire et CPU des VMs analysées dans les sections suivantes.

Ces figures font apparaître que la configuration des VMs varie entre 812 MB et 65 GB pour la mémoire et entre 2 Ghz et 15 Ghz pour le CPU. La configuration la plus utilisée est 4 GB pour

la mémoire et 8 Ghz pour le CPU et 4% des VMs seulement ont vu leur configuration mémoire ou CPU varier sur une durée de 2 mois. Seules les VMs dont les configurations en mémoire et en CPU restent inchangées sur la période de la collecte, sont représentées sur ces deux figures.



(a) Répartition des VMs mémoire configurée (b) Répartition des VM CPU configuré

FIGURE 4.1 – Vue globale de la configuration des VMs en mémoire et en CPU

Dans le reste du chapitre, nous nous intéresserons à l'analyse des quantités de ressources mémoire et CPU consommées par ces VMs.

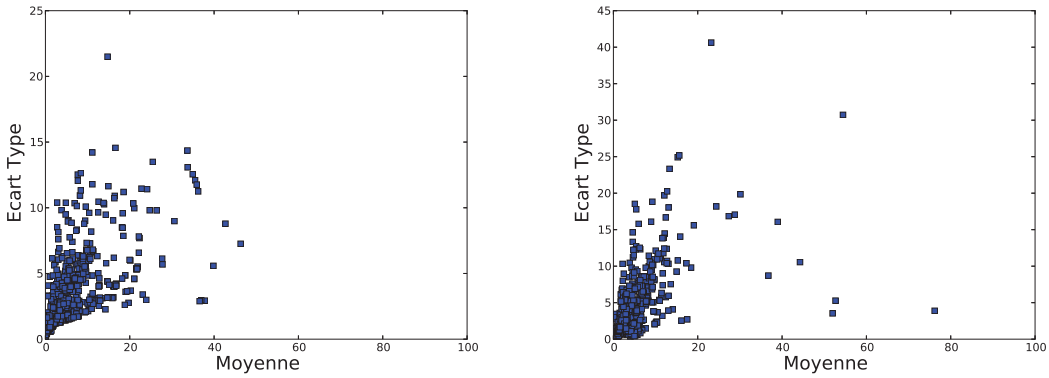
4.4 Analyse de la consommation en ressources des VMs

Dans cette section, nous analysons les quantités de ressources mémoire et CPU consommées par les VMs, sur une durée de 28 jours.

Les figures 4.2(a) et 4.2(b) donnent le nuage de points représentant les VMs respectivement par l'écart-type de leurs valeurs d'utilisation mémoire et CPU sur la période de 28 jours, en fonction de la moyenne de ces mêmes valeurs. Nous pouvons observer que sur cette durée, 95% des VMs ont une moyenne d'utilisation mémoire qui ne dépasse pas 20% de la mémoire configurée et que 98% des VMs considérées ont qu'une moyenne d'utilisation CPU qui ne dépasse pas 20% de la quantité de CPU configurée.

Nous pouvons donc conclure que pour plus de 95% des VMs, l'utilisation moyenne des ressources est très en-deçà des quantités de ressources qui leur sont allouées par configuration statique avec les systèmes de gestion dynamique des ressources actuellement déployés dans les clouds et que l'ajustement des quantités de ressources allouées aux VMs n'est que très peu effectif.

VMware définit plusieurs métriques de quantification de la mémoire utilisée par les VMs. Nous détaillons ceux que nous avons retenus pour servir nos objectifs de mesure et d'analyse.



(a) Moyenne d'utilisation mémoire (b) Moyenne d'utilisation CPU

FIGURE 4.2 – Nuages des VMs représentées par l'écart-type et la moyenne de leur utilisation en ressources sur une période de 28 jours

Pour quantifier l'utilisation mémoire d'une VM, l'hyperviseur doit effectuer une estimation de cette utilisation car il ne dispose pas de la vue sémantique de l'utilisation des ressources, offerte par le système d'exploitation de la VM.

En effet, l'hyperviseur manque de visibilité sur l'utilisation des pages mémoire allouées, une fois l'allocation de la mémoire à une VM effectuée : une page mémoire précédemment utilisée par la VM et qui ne l'est plus, n'est pas vue par l'hyperviseur comme ayant été libérée. Pour estimer l'utilisation courante de la mémoire, l'hyperviseur ESX procède par calcul du **Working Set Size** (WSS) selon une méthode d'échantillonnage de la mémoire comme suit. De façon périodique – toutes les 20 secondes par défaut – l'hyperviseur sélectionne des régions de la mémoire physique – 100 pages mémoire par défaut – pour lesquelles il supprime les liens entre les pages de la mémoire virtuelle présentées à la VM et les pages de la mémoire physique du serveur hôte l'exécutant, ce qui génère un défaut de page au niveau de l'hyperviseur lorsqu'une VM tente d'accéder à une page de cette région et permet ainsi de détecter que la page mémoire est utilisée par la VM, à la suite de quoi l'hyperviseur rétablit le lien supprimé pour permettre à la VM d'accéder à la page mémoire requise. La métrique VMware d'utilisation mémoire vue par l'hyperviseur calculée à partir du WSS, est la *mémoire active*. Celle-ci estime la quantité de mémoire activement utilisée par la VM.

Une autre métrique VMware de quantification de la mémoire utilisée est la *mémoire consommée*, qui est définie par la différence entre la quantité de la mémoire allouée et la quantité de la mémoire partagée entre les VMs. En cas de sur-estimation des quantités de ressources configurées aux VMs de la part des clients, cette métrique produit des valeurs largement supérieures à l'utilisation mémoire réelle des VMs, comme nous le démontre nos mesures présentées ci-dessus.

La figure 4.3 illustre par VM et sur la période de 28 jours considérée, l'écart-type des valeurs de *mémoire active* (symboles bleus) et de *mémoire consommée* (symboles rouges) en fonction

de la moyenne de ces mêmes valeurs, sur un nuage de points représentant l'ensemble des VMs. L'objectif de cette illustration est de mettre en évidence la différence entre les valeurs que prennent ces deux métriques. On peut observer que les moyennes de mémoire actives associées aux différentes VMs sont nettement inférieures à celles de mémoire consommée.

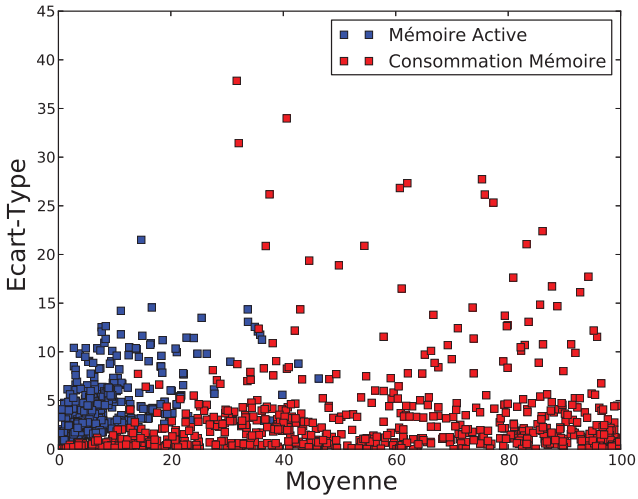
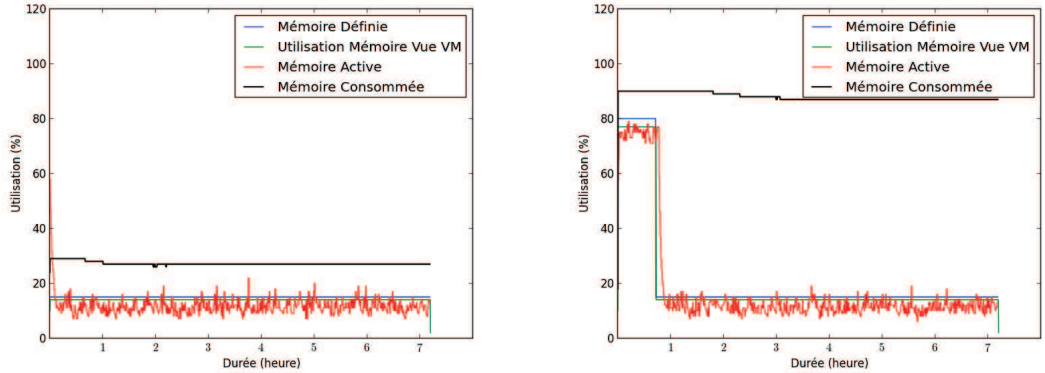


FIGURE 4.3 – Moyenne de la mémoire active et de la mémoire consommée pour les différentes VMs

Pour mieux illustrer la non adéquation de la mémoire consommée comme métrique d'appréciation de l'utilisation mémoire par les VMs, nous rapportons les observations d'une expérimentation faite sur notre plate-forme VMware avec notre outil de génération de charge configurable introduit dans la section 2.5.1 du chapitre 2, qui nous permet d'obtenir les valeurs de métriques de mémoire active et de mémoire consommée en fonction de valeurs paramétrables de consommation de mémoire, telle que vue par l'hyperviseur et par le système d'exploitation de la VM. Nous considérons que les métriques les plus pertinentes pour estimer et restituer, depuis l'hyperviseur, l'utilisation mémoire de la VM sont celles ayant les valeurs les plus proches des valeurs d'utilisation mémoire collectées depuis le système d'exploitation de la VM.

Nos agents de génération de charge mémoire sont déployés dans deux VMs de capacité mémoire égale (2GB), pour des durées d'expérimentation de 7 heures. Dans la première VM, le paramétrage de l'agent de consommation mémoire est ajusté pour consommer 15% de la mémoire configurée sur une durée de 7 heures. Dans la deuxième VM, l'outil est paramétré pour consommer 80% de la mémoire configurée pendant 30 minutes, puis réduire cette consommation à 15% pour le reste de la durée de l'expérimentation. Notre module de collecte nous permet de corréler les vues de consommation mémoire mesurées par le système d'exploitation (utilisation mémoire telle qu'obtenue avec la commande *top* sous linux) et par l'hyperviseur (mémoire active, mémoire consommée).

Les figures 4.4(a) et 4.4(b) illustrent les résultats, respectivement de la première et de la deuxième expérimentation, en indiquant la variation de l'utilisation mémoire vue par le système d'exploitation depuis la VM, ainsi que les variations de la mémoire active et de la mémoire consommée vues par l'hyperviseur.



(a) Comparaison des vues VM et hyperviseur de quantification de la mémoire - expérimentation 1

(b) Comparaison des vues VM et hyperviseur de quantification de la mémoire - expérimentation 2

FIGURE 4.4 – Comparaison des vues VM et hyperviseur de quantification de la mémoire

Pour la première expérimentation (figure 4.4(a)), les valeurs de mémoire active vue hyperviseur et d'utilisation mémoire vue VM sont cohérentes avec la valeur de mémoire définie à l'aide de notre agent de consommation configurable : 15%. La valeur de mémoire consommée, qui est de l'ordre de 30%, représente la quantité de mémoire consommée par l'outil de génération de charge à laquelle s'ajoute celle utilisée par le système d'exploitation au démarrage de la VM.

Pour la deuxième expérimentation (figure 4.4(b)), on peut observer que pendant les 30 premières minutes de l'expérimentation, l'utilisation mémoire vue de la VM ainsi que la mémoire active vue hyperviseur indiquent un taux d'utilisation de près de 80% qui correspond au taux défini par l'outil de génération de charge mémoire, la mémoire consommée indique quant à elle un taux de l'ordre de 90%. Durant la deuxième période de l'expérimentation 2, lorsque l'outil réduit la consommation mémoire de 80% de consommation à 15% de consommation, la mémoire active ainsi que l'utilisation mémoire vue VM diminuent de 80% à 15% alors que la mémoire consommée reste quant à elle aux environs de 90% de consommation pour le reste de la durée de l'expérimentation. Cela est dû au fait que les pages mémoire utilisées par l'outil de génération de charge mémoire lorsqu'il consommait 80% de mémoire n'ont pas été libérées par le système d'exploitation et sont toujours considérés par l'hyperviseur comme étant de la mémoire consommée.

Ces observations nous amènent à émettre l'hypothèse suivante : quant aux valeurs de mémoire consommées relevées sur les centres de données Orange, qui sont pour certaines très élevées (autour de 100% selon la figure 4.3) : les VMs avec des valeurs moyennes de mémoire consommée

élevées et des moyennes de mémoire active faibles ont observé au moins un pic d'utilisation mémoire qui a fait augmenter la mémoire consommée sans qu'elle ne soit libérée lorsque la mémoire active a diminué, ces pics de mémoire pouvant être antérieurs à notre période de collecte.

En conclusion, la métrique mémoire consommée vue hyperviseur restitue mal les variations d'utilisation de la mémoire par les VMs et majore les utilisations mémoire véritables. L'ensemble de ces observations, nous amènent à analyser la fluctuation de la consommation en ressources des VMs à l'aide de la métrique de mémoire active car c'est celle qu'il convient de considérer pour caractériser les variations de consommation.

4.5 Analyse de la fluctuation des VMs

Dans cette section, nous livrons nos résultats de l'analyse de la fluctuation de la consommation en ressources de VMs s'exécutant sur cloud privé d'Orange.

4.5.1 Motivation

Dans la section précédente, nous avons pu noter qu'une grande partie des VMs a des moyennes de taux d'utilisation mémoire et CPU très faibles, qui sont la résultante de la sur-estimation des besoins en ressources des VMs par les utilisateurs. Un moyen pour l'opérateur de compenser cette sous-utilisation des ressources est de sur-engager les ressources de l'infrastructure, en configurant aux VMs s'exécutant sur un hôte davantage de ressources que la capacité totale du hôte. Cependant, sur-engager les ressources augmente le risque de violation des engagements en qualité de service passés entre l'opérateur et ses clients, si l'ensemble des VMs venaient à utiliser au même moment la totalité des ressources à laquelle elles ont souscrit. Considérer la variation des profils de consommation des VMs est de nature à minimiser le risque lié au sur-engagement des ressources : l'identification automatique des VMs à profil de consommation fluctuant peut par exemple orienter le placement dynamique des VMs au sein d'un centre de données.

Plusieurs travaux de l'état de l'art considèrent que le sur-engagement des ressources est plus tolérable en présence de VMs à profil de consommation en ressources fluctuant. Cette position est motivée par le fait que ce type de VMs ne consomment les ressources qui leur sont allouées que de façon occasionnelle (durant les pics de consommation uniquement). Néanmoins, le raisonnement poursuivi par ces travaux prend pour hypothèse que les VMs stables ont des moyennes de taux d'utilisation élevées, ce qui n'est ni confirmé par les traces de consommation des VMs dont nous disposons ni par les traces de consommation publiées dans les travaux antérieurs.

Pour notre part, nous pensons en première analyse que le sur-engagement de ressources est moins préférable pour les VMs fluctuantes car celles-ci ont des profils de consommation moins prédictibles que les profils des VMs à consommation stable dans le temps. On peut penser que les VMs fluctuantes présentent davantage de risques d'observer des violations de SLAs lors de l'occurrence de pics de consommation en ressources, si les ressources sont sur-engagées. C'est la raison pour laquelle nous défendons l'idée que le sur-engagement de ressources est mieux maîtrisé en présence de VMs à profil de consommation en ressources stables avec de faibles moyennes de taux d'utilisation.

Dans les deux cas de raisonnement, on constate qu'il serait avantageux d'appliquer des politiques de gestion de ressources particulières (placement, élasticité autonomique, sécurité) aux VMs à profil de consommation fluctuant, ce qui motive fortement le besoin d'identification automatique de ce type de VMs.

4.5.2 Partitionnement des VMs selon le critère de fluctuation

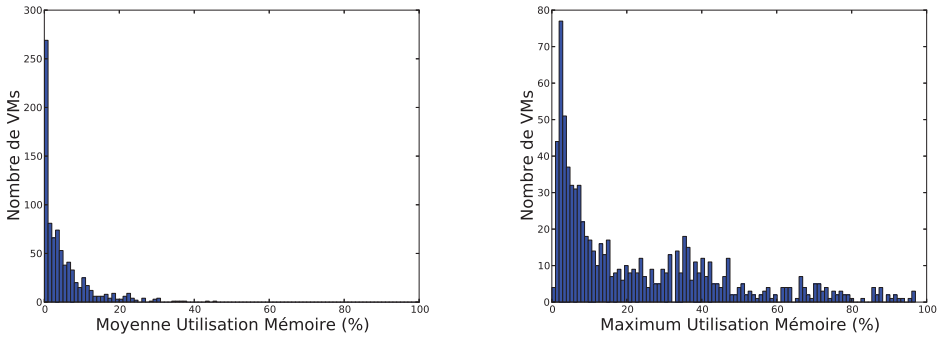
Dans cette section, nous nous intéressons à l'analyse de la fluctuation des profils de consommation en ressources mémoire et CPU des VMs de notre campagne de collecte. Étant donné que nous nous intéressons aux comportements de fluctuation à temps court (de l'ordre d'une heure), nous appliquons l'algorithme d'identification des VMs fluctuantes d'AMAD (détaillé en section 3.3) sur un historique de consommation (dénommé fenêtre de rétro-analyse dans le chapitre 3 de présentation d'AMAD) d'une durée de 24 heures avec une taille de fenêtre temporelle glissante fixée à 30 minutes.

L'algorithme AMAD d'identification des VMs à profil de consommation fluctuant produit les résultats suivant : 20% des VMs sont fluctuantes lorsque l'algorithme est appliqué sur l'utilisation mémoire (mémoire active) et près de 17% des VMs sont fluctuantes lorsque l'algorithme est appliqué sur l'utilisation CPU.

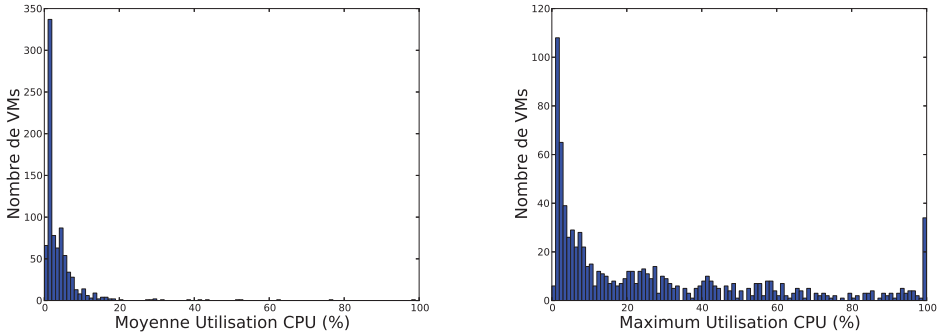
Pour mieux caractériser les profils de consommation en ressources des VMs analysées, nous représentons sur la figure 4.5 les distributions de l'utilisation mémoire et CPU de ces VMs.

Les figures 4.5(a) et 4.5(c) représentent respectivement la distribution des moyennes d'utilisation mémoire et CPU des VMs, sur la durée de 24 heures considérée. Les figures 4.5(b) et 4.5(d) représentent respectivement les distributions des valeurs maximums d'utilisation mémoire et CPU de chaque VM sur cette même période.

Sur les figures 4.5(a) et 4.5(c), on peut observer que seules 2% environ des VMs dépassent 30% de moyenne d'utilisation mémoire et CPU sur la durée de 24 heures. Sur la figure 4.5(b), on peut observer que seules 2% des VMs atteignent une valeur maximum d'utilisation mémoire supérieure à 80% de la quantité de mémoire qui leur est configurée. S'agissant de la valeur maximum d'utilisation CPU illustrée figure 4.5(d), seules 8% des VMs présentent une valeur maximum d'utilisation CPU supérieure à 80% de la quantité de CPU qui leur est configurée.



(a) Moyenne d'utilisation mémoire (b) Maximum d'utilisation mémoire

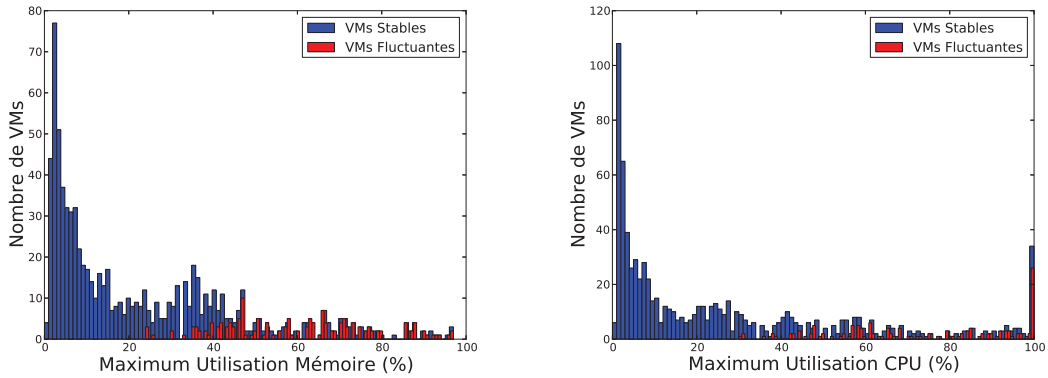


(c) Moyenne d'utilisation CPU (d) Maximum d'utilisation CPU

FIGURE 4.5 – Histogrammes d'utilisation mémoire et CPU sur une durée de 24 heures

L'interprétation de ces distributions nous amène à la conclusion suivante : les VMs observées qui présentent des valeurs maximum d'utilisation mémoire élevées démontrent nécessairement une forme de fluctuation puisque leurs moyennes d'utilisation restent faibles. En effet, ces VMs n'utilisent pas continûment des valeurs élevées de quantité de ressources (puisque leur moyenne est faible) et présentent néanmoins des maximums (correspondant donc à des pics de fluctuation), elles sont donc fluctuantes.

Sur la figure 4.6, nous reprenons les distributions de l'ensemble des valeurs maximums d'utilisation mémoire et CPU (respectivement déjà représentées en figures 4.5(b) et 4.5(d)), auxquelles nous superposons, en rouge, celles des VMs identifiées comme fluctuantes par l'algorithme de la phase 1 d'AMAD. Nous pouvons observer respectivement pour la mémoire et pour le CPU, sur les figures 4.6(a) et 4.6(b), que les VMs identifiées par AMAD se superposent majoritairement aux VMs qui présentent des valeurs élevées de maximum d'utilisation en ressource (maximum utilisation mémoire ou CPU > 40%) et dont on a déjà démontré qu'elles manifestent nécessairement une forme de fluctuation (par référence aux moyennes qui restent faibles), ce qui tend à démontrer le caractère effectivement fluctuant des VMs identifiées comme fluctuantes par l'algorithme AMAD.



(a) Maximum d'utilisation mémoire

(b) Maximum d'utilisation CPU

FIGURE 4.6 – Distribution des maximum d'utilisation mémoire et CPU pour les VMs stables et fluctuantes

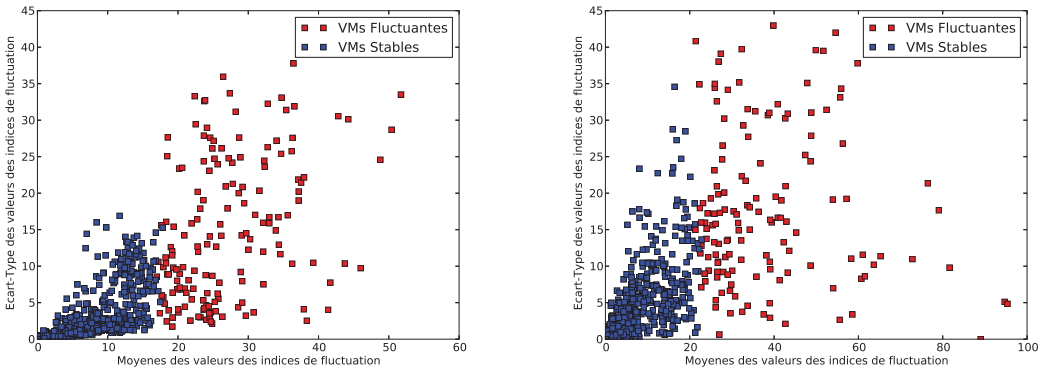
Il est difficile d'évaluer la performance d'identification des VMs fluctuantes de l'algorithme AMAD, avec des VMs à consommation réelle, puisque nous ne disposons pas de la vérité terrain permettant de qualifier le niveau de fluctuation de ces VMs. Néanmoins, la concordance des résultats d'identification des VMs fluctuantes par AMAD avec l'interprétation du caractère nécessairement fluctuant des VMs présentant des valeurs d'utilisation moyenne faible et maximum forte, nous donne quelques assurances quant aux principes de conception d'AMAD.

Selon notre connaissance de l'état de l'art actuel, il n'existe pas de méthode, ni de métrique qui permettent de quantifier la fluctuation des VMs. Dans la section suivante, nous tentons d'établir une première approche de quantification de la fluctuation des VMs, basée sur notre proposition de vecteurs d'indice de fluctuation.

4.5.3 Quantification de la fluctuation

Le calcul des vecteurs d'indices de fluctuation défini dans le chapitre 3 et utilisé par AMAD, est conçu de sorte à ce que les vecteurs associés aux VMs les plus fluctuantes contiennent des valeurs d'indices de fluctuation plus élevées que celles des vecteurs associés aux VMs relativement stables.

La figure 4.7 illustre, avec un nuage comptant autant de points que de VMs, l'écart-type des valeurs composant les vecteurs d'indices de fluctuation de chaque VM en fonction de la moyenne de ces mêmes valeurs. Les VMs identifiées par l'algorithme AMAD comme étant stables sont représentées avec un symbole bleu et celles identifiées comme étant fluctuantes avec un symbole rouge. La figure 4.7(a) donne cette représentation lorsque l'algorithme d'identification des VMs fluctuantes est appliqué sur la mémoire, et la figure 4.7(b) livre la même représentation pour un partitionnement effectué sur le CPU.



(a) Moyenne et écart-type des indices de fluctuation des VMs pour la mémoire

(b) Moyenne et écarts-types des indices de fluctuation des VMs pour le CPU

FIGURE 4.7 – Nuage de VMs représentant leur moyenne et écarts-type de valeurs d'indices de fluctuation

On peut observer sur les deux figures que les moyennes des valeurs des vecteurs d'indices de fluctuation associées aux différentes VMs fluctuantes (rouge) sont supérieures à celles associées aux différentes VMs stables (bleu). La deuxième observation est que les écarts-types des valeurs des vecteurs d'indices de fluctuation sont beaucoup plus hétérogènes et ne permettent pas de dégager une tendance claire quant à la relation entre l'écart-type des valeurs d'indices de fluctuation d'une VM et son niveau de fluctuation.

Intuitivement, on peut penser que l'écart-type des valeurs d'indices de fluctuation contenues dans un vecteur d'une VM restitue une information quant au niveau de fluctuation de celle-ci : à moyenne de valeurs d'indices de fluctuation égale, la VM présentant l'écart-type le plus faible devrait correspondre à la VM ayant le niveau de fluctuation le plus élevé. Cette notion n'a pas été approfondie et fera l'objet de travaux futurs.

Dans la dernière partie de cette section, nous cherchons à comprendre comment la moyenne des valeurs d'indices de fluctuation varie en fonction du niveau de fluctuation d'une VM.

Pour mieux cerner la notion de fluctuation, nous avons besoin d'une valeur de fluctuation de référence que nous pouvons manipuler pour caractériser comment le niveau de fluctuation d'une VM est restitué par la moyenne des valeurs d'indices de fluctuation. Pour cela, nous avons sélectionné une trace de consommation d'une VM, obtenue lors des expérimentations menées dans le chapitre 3, et dont le profil d'utilisation mémoire est illustré sur la figure 4.8 dans laquelle nous injectons des pics de consommation mémoire et CPU.

Comme pour les évaluations menées dans le chapitre 3, nous considérons que le niveau de fluctuation d'une VM peut être impacté par deux paramètres : la **fréquence** d'apparition des pics de consommation mémoire ou CPU dans le profil d'une VM et l'**amplitude** de variation de ces pics.

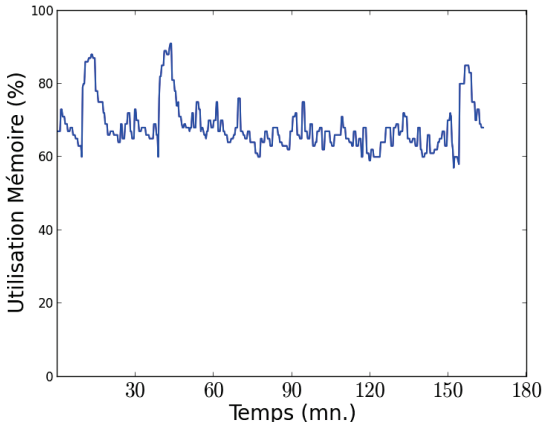


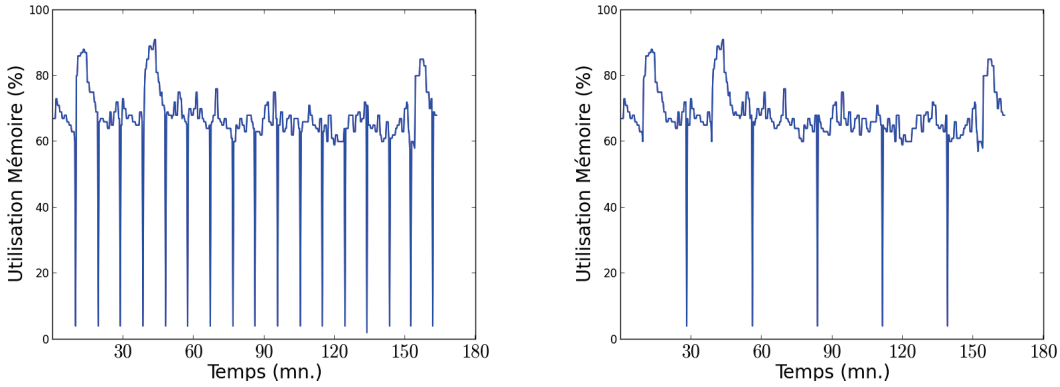
FIGURE 4.8 – Profil d'utilisation mémoire de la VM référence

Pour manipuler le niveau de fluctuation de la VM de référence par une variation de la fréquence d'apparition de pics de consommation, nous maintenons l'amplitude des pics de consommation constante à 70% d'utilisation mémoire ou CPU et nous faisons varier la fréquence d'apparition des pics de consommation en les injectant toutes les X minutes, X variant de 10 minutes jusqu'à 70 minutes d'une expérimentation à une autre. La figure 4.9(a) illustre le profil d'utilisation mémoire résultant de l'injection de pics de consommation avec une fréquence de 10 minutes.

Pour manipuler le niveau de fluctuation de la VM de référence par une variation de la l'amplitude des pics de consommation, nous maintenons la fréquence d'apparition des pics de consommation constante à 30 minutes et faisons varier l'amplitude des pics de l'utilisation mémoire et CPU de 10% d'utilisation à 70% d'utilisation. La figure 4.9(b) illustre le profil d'utilisation mémoire résultant de l'injection de pics de consommation avec une amplitude de 70% d'utilisation mémoire.

Pour chaque fréquence d'injection de pics de consommation, nous calculons la moyenne des valeurs d'indices de fluctuation du vecteur associé au profil de consommation résultant. Les figures 4.10(a) et 4.10(b) illustrent, respectivement pour le CPU et pour la mémoire, la variation de la moyenne des indices de fluctuation en fonction de la fréquence d'apparition des pics de consommation dans le profil de référence.

Nous pouvons observer, à la fois pour le CPU et pour la mémoire, que les moyennes des indices de fluctuation augmentent avec l'augmentation de la fréquence d'apparition des pics de consommation, ce qui confirme que la moyenne des indices de fluctuation de la VM de référence ainsi bruitée, est une métrique qui permet de restituer son niveau de fluctuation. Partant de ce constat, nous pouvons ordonner les profils de consommation en ressources des VMs s'exécutant sur cloud réel en fonction de leur niveau de fluctuation avec les VMs ayant les moyennes d'indices de fluctuation les plus élevés qui seront considérées comme celles ayant le

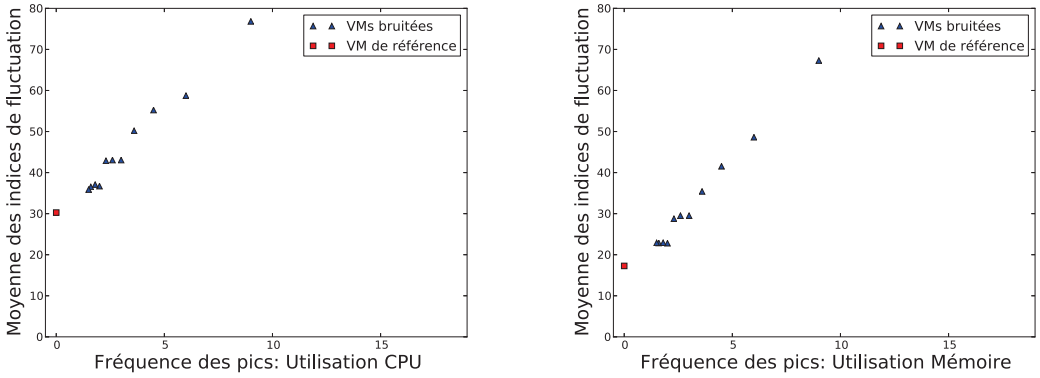


(a) VM référence avec des pics de consommation introduits toutes les 10 minutes avec une amplitude d'amplitude 70% de 70%

(b) VM référence avec des pics de consommation introduits toutes les 30 minutes avec une amplitude d'amplitude 70% de 70%

FIGURE 4.9 – Manipulation de la fluctuation du profil de la VM référence

niveau de fluctuation le plus élevé.



(a) fluctuation en fonction de la fréquence d'apparition des pics de consommation, pour le CPU

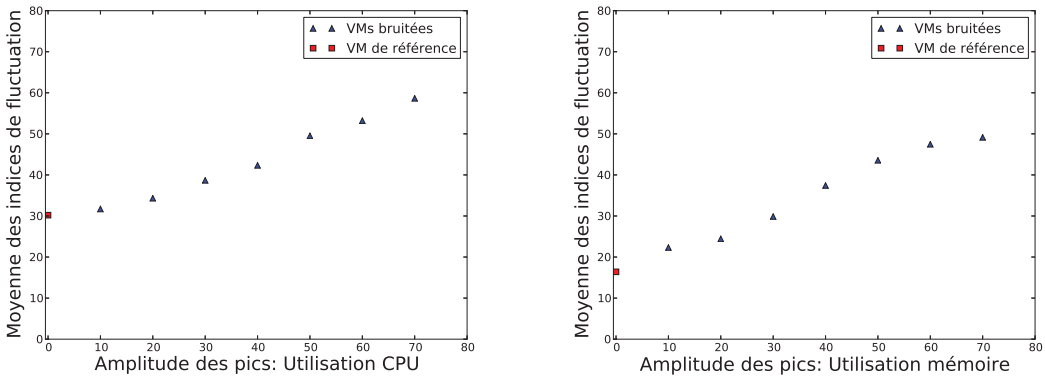
(b) fluctuation en fonction de la fréquence d'apparition des pics de consommation, pour la mémoire

FIGURE 4.10 – fluctuation en fonction de la fréquence d'apparition des pics de consommation, pour le CPU

Enfin, les figures 4.11(a) et 4.11(b) illustrent respectivement pour le CPU et pour la mémoire la variation des moyennes des indices de fluctuation en fonction de l'amplitude des pics de consommation injectés dans les profils de consommation de la VM de référence.

Les observations que l'on peut faire sont similaires à celles précédentes, avec cette fois une augmentation de la moyenne des indices de fluctuation avec l'augmentation de l'amplitude des pics de consommation introduits dans le profil de la VM référence.

Nous pouvons conclure que lorsque la fluctuation est régulière comme c'est le cas dans nos



(a) fluctuation en fonction de l'amplitude des pics de consommation, pour le CPU

(b) fluctuation en fonction de l'amplitude des pics de consommation, pour la mémoire

FIGURE 4.11 – fluctuation en fonction de l'amplitude des pics de consommation

expérimentations avec des pics de consommation introduits de manière périodique, le niveau de fluctuation des VMs est restitué par la moyenne des indices de fluctuation associé à chaque VM. Néanmoins, nous n'avons pas d'éléments pour affirmer la validité de ces résultats dans le cas où la fluctuation devient intermittente, avec la difficulté de définir correctement, dans ce cas, la taille de la fenêtre temporelle glissante. L'investigation de ce point nécessite de plus amples travaux.

4.6 Conclusion

Dans ce chapitre, nous avons analysé les profils de consommation en ressources de VMs s'exécutant sur cloud privé. Nos résultats d'analyse rejoignent les conclusions des travaux de l'état de l'art, à savoir que les utilisateurs de VMs surestiment largement les quantités de ressources nécessaires à l'exécution de leurs VMs et que l'enjeu du potentiel de la virtualisation en matière de gestion dynamique des ressources, considérée pour sa capacité à allouer des ressources au plus près des besoins des VMs, est donc majeur. Au delà de ce constat partagé, notre analyse est la première à s'intéresser à la fluctuation de la consommation en ressources des VMs. L'application de l'algorithme d'identification des VMs fluctuantes sur des traces d'un historique de 24 heures, a permis d'estimer à près de 20% la part de VMs présentant un profil de consommation fluctuant.

Nous avons aussi proposé une première approche de quantification de la fluctuation de la consommation en ressources des VMs, basée sur les valeurs des indices de fluctuation associées aux VMs. Bien que nos résultats démontrent des perspectives encourageantes en matière de quantification de la fluctuation, le sujet reste à investiguer davantage pour traiter de la fluctuation par intermittence, de l'impact de la taille de fenêtre temporelle glissante (fixée jusqu'à présent à

T = 30 minutes) sur le type de fluctuation que l'on cherche à identifier ou encore de l'interprétation des valeurs d'écart-type d'indices de fluctuation obtenues dans nos expérimentations.

Enfin, l'étude et l'apport de politiques de gestion des ressources tirant partie de l'identification automatique des VMs fluctuantes, restent autant de voies de recherche qui devront faire l'objet de travaux futurs.

Chapitre 5

Conclusion

5.1 Bilan

Les travaux de cette thèse avaient pour vocation de traiter de nouvelles problématiques de sécurité apparues avec certaines propriétés intrinsèques aux plate-formes cloud, à savoir le partage des ressources entre des VMs appartenant à différents utilisateurs d'une part et l'élasticité dans l'allocation des ressources d'autre part.

Après une analyse fine de l'état de l'art, notre attention s'est portée sur les systèmes de gestion dynamique des ressources qui sont des nouveaux systèmes dont l'objectif est d'optimiser la consolidation des ressources des centres de données tout en veillant à assurer un partage équitable des contextes d'exécution entre les VMs. La dépendance de ces systèmes de gestion dynamique des ressources aux quantités de ressources consommées par les VMs, associée à l'inter-dépendance entre les performances d'exécution des VMs que crée le partage des ressources, nous a amenés à nous questionner sur une vulnérabilité possible de ces systèmes vis-à-vis d'utilisateurs qui manipulent à des fins malveillantes les quantités de ressources consommées par des VMs sous leur contrôle.

Les systèmes de gestion dynamique de ressources de l'état de l'art actuel reposent essentiellement sur le mécanisme de migration de VMs pour l'optimisation de la consolidation des serveurs. Ce mécanisme qui consiste en le transfert de l'état courant de la VM d'un hôte source vers un hôte de destination, est coûteux pour l'infrastructure en termes de ressources consommées pour gérer la migration d'une part et pour la VM migrée en termes de dégradation de performances qu'elle peut observer, en particulier si elle est déjà sous contention de ressources, d'autre part.

La migration de VM a été considérée sous plusieurs aspects dans l'état de l'art antérieur mais jamais sous l'angle de la sécurité comme nous l'avons fait dans cette thèse. Le coût de la migration pour une infrastructure cloud et la potentielle vulnérabilité des systèmes de gestion dynamique des ressources à une manipulation malicieuse des quantités de ressources consommées par les VMs, motivent nos travaux consistant à identifier comment un attaquant

pourrait amener le système de gestion dynamique des ressources à migrer de façon abusive des VMs par une simple manipulation des quantités de ressources consommées par des VMs sous son contrôle.

Pour la démonstration de cette attaque, nous avons utilisé l'algorithme VMware DRS qui adopte une approche par équilibrage des charges. Notre choix s'est porté sur DRS, parce que c'est un algorithme validé, commercialisé et largement utilisé par les fournisseurs cloud. Il n'existe pas à notre connaissance d'algorithme équivalent à DRS largement adopté par la communauté industrielle et académique dans l'état de l'art actuel.

Nous avons démontré la faisabilité de l'attaque par migrations intempestives de VMs sur une plate-forme d'expérimentation en laboratoire constituée de cinq serveurs, grâce à l'exécution de profils de consommation en ressources fluctuants et coordonnés. Comme DRS adopte une approche centrée sur le cluster, les conditions de succès de l'attaque dépendent fortement de la répartition de la charge des hôtes qui constituent le cluster. Nous avons analysé les conditions de succès de l'attaque ainsi que les niveaux d'exposition des clusters vis-à-vis de cette vulnérabilité et nos résultats montrent, qu'avec l'algorithme VMware DRS, la quantité minimum de ressources qui doit être sous le contrôle de l'attaquant pour réussir l'attaque, diminue lorsque la taille du cluster augmente ou lorsque le niveau d'agressivité de DRS augmente.

Cette preuve de concept met en évidence les limites actuelles de la sécurité de la gestion dynamique des ressources qui ne prend pas en considération les contraintes de sécurité dans le processus de décision. Au delà de ce constat applicable aux systèmes actuels, les principes plus prospectifs d'une gestion élastique des ressources autonomique risquent également d'être exposés à ce type de vulnérabilité.

La mitigation de ce type d'attaque peut se faire de façon proactive en intégrant les éléments de sécurité nécessaires dans le processus de décision des systèmes de gestion dynamique des ressources ou, de façon réactive avec une approche de supervision pour détecter l'occurrence d'un nombre anormal de migrations de VMs comme nous l'avons fait dans cette thèse. Nous avons proposé un système qui adopte une approche réactive de supervision des décisions de migration de VMs et qui agit comme une couche supplémentaire à celle du système de gestion dynamique de ressources et qui permet de détecter une exécution d'un nombre anormal de migrations de VMs et de déclencher une phase d'analyse des profils de consommation en ressources des VMs pour identifier celles ayant des profils de consommation en ressources fluctuants et temporellement coordonnées considérées comme celles à l'origine de l'attaque. L'approche de supervision réactive d'AMAD protège l'infrastructure de l'attaque par migrations intempestives de VMs, sans que la conception du système de gestion dynamique de ressources soit impactée.

Pour l'identification des VMs fluctuantes par AMAD, nous avons d'abord mis en évidence les limites des métriques actuelles pour la caractérisation de ce type de VMs puis avons proposé

une transformation des séries temporelles de mesure de la consommation en ressources de VMs en indices de fluctuation qui permettent la prise en compte du critère de fluctuation de la consommation en ressources des VMs. Nous avons évalué les performances d'AMAD pour l'identification des VMs attaquantes dans de multiples contextes : i) sur plate-forme de laboratoire à l'aide de traces de consommation en ressources de VMs s'exécutant sur clouds réels, ii) par injection de profils de consommation des VMs attaquantes dans un contexte issu de cloud réel, iii) par déformation de profils de VMs attaquantes et enfin, iv) par manipulation de la fluctuation du contexte d'exécution des VMs attaquantes.

Enfin, notre dernière étude porte sur l'analyse de traces de consommation en ressources de VMs issues de clouds privés d'Orange. Nos résultats viennent d'une part confirmer les conclusions des études disponibles dans l'état de l'art concernant la sur-estimation par les utilisateurs des quantités de ressources configurées aux VMs, et d'autre part, montrer le potentiel d'utilisation de l'algorithme AMAD d'identification des VMs fluctuantes.

Les travaux menés durant cette thèse ouvrent des perspectives dans le domaine de la sécurité du partage de ressources au service de l'optimisation de la consolidation en ressources des centres de données.

5.2 Perspectives

Nous présentons dans cette partie les perspectives de la thèse, tout d'abord dans le domaine de la sécurité, puis dans le domaine de la gestion dynamique de ressources dans le cloud.

5.2.1 Sécurité

Réaction

L'attaque par migrations intempestives de VMs est exécutée à l'aide de profils de consommation en ressources fluctuants qui permettent de créer et de renouveler fréquemment un déséquilibrage de charge au sein d'un cluster, pour amener le système de gestion dynamique de ressources à migrer fréquemment des VMs. Toutefois, de tels profils de consommation en ressources fluctuants (voire sporadiques) peuvent résulter d'une activité applicative non malveillante exécutée par les VMs.

Un mode de réaction possible à la suite de la détection d'une attaque par migrations intempestives de VMs, serait de mitiger les effets de l'exécution de profils de consommation fluctuants, de préférence sans porter préjudice à l'exécution des VMs car la fluctuation de profil de consommation de ressources peut aussi résulter d'une activité applicative non malveillante. Cette mitigation pourrait être réalisée en édictant des règles d'anti-affinité de VMs qui viendraient contraindre

le système de gestion de ressources dans ses décisions de placement des VMs, de telle sorte à séparer les VMs ayant des profils cumulatifs (sur le même hôte) ou en opposition de phase (sur deux hôtes) pour contrecarrer la capacité de nuisance de ces VMs.

Attaque par migrations intempestives de VMs sur cloud réel

Dans nos expérimentations, nous avons mené l'attaque par migrations intempestives de VMs sur une plate-forme de laboratoire, pour laquelle nous avons la maîtrise de l'ensemble des paramètres.

La faisabilité de l'attaque sur cloud réel reste à démontrer et nécessiterait des moyens complémentaires de supervision de l'état courant du contexte d'exécution des VMs de l'attaquant, pour permettre à l'attaquant d'avoir un retour sur le déroulement de l'attaque et sur l'occurrence de migrations de VMs. La mesure des performances de la VM attaquante ou encore la détection de contentions de ressources au niveau de la VM attaquante, sont autant d'indications pour l'attaquant pour lui permettre d'apprécier la mobilité des VMs co-résidentes sur l'hôte exécutant ses VMs.

Il serait alors pertinent de mesurer les conditions de succès de l'attaque dans un contexte de cloud réel ainsi que l'exposition des clusters à cette vulnérabilité, en fonction de la capacité des hôtes, de la capacité des VMs sur un hôte donné, du placement des VMs ou encore du taux de charge des hôtes.

Vulnérabilités de la gestion élastique des ressources

Une des perspectives des systèmes de gestion dynamique des ressources dans le cloud est d'implémenter des systèmes autonomiques qui ajustent automatiquement les quantités de ressources allouées aux VMs en fonction de leurs besoins.

Dans cette thèse, nous avons démontré qu'un système de gestion dynamique des ressources poursuivant une stratégie d'équilibrage de la charge au sein des clusters dans le but de limiter les contentions de ressources des hôtes, peut être vulnérable à une simple manipulation de quantités de ressources consommées par des VMs malicieuses. Il serait judicieux de poursuivre l'étude de la sécurité des systèmes de gestion dynamique de ressources, en examinant des systèmes autres, qui adoptent des principes différents, pour étudier si la dépendance des décisions de ces systèmes aux quantités de ressources consommées, peut présenter de plus amples risques en termes de sécurité.

Introspection

Les solutions que nous avons proposées dans cette thèse pour l'identification des VMs malicieuses sont basées essentiellement sur une caractérisation *quantitative* des profils de consomma-

tion en ressources des VMs.

Pour une analyse plus fine de la fluctuation des profils, il conviendrait de caractériser d'un point de vue sémantique, les profils de consommations en ressources suspects afin d'être en mesure de distinguer une activité fluctuante normale d'une activité fluctuante malicieuse. L'apport de l'introspection pour laquelle il a été démontré que par reconstruction de l'abstraction offerte par le système d'exploitation des VMs et/ou par connaissance de l'architecture hardware, l'hyperviseur est apte à découvrir certaines informations sémantiques à partir d'une visibilité de niveau binaire des ressources consommées par les VMs, serait à considérer pour déterminer s'il existe une information sémantique accessible depuis l'hyperviseur qui permet de juger du caractère malicieux d'un profil fluctuant.

Des voies d'investigation prometteuses ont été identifiées pour permettre par exemple une surveillance des quantités de ressources mémoire et processeur consommées par processus. Ces pistes pourraient permettre de mener une analyse plus fine des profils de consommation en ressources des VMs pour effectuer des corrélations entre les résultantes quantitatives des profils, le nombre de processus produisant ce profils et le type de service exécuté par la VM.

5.2.2 Gestion dynamique des ressources

Quantification de la fluctuation

La définition des taux de sur-engagement des hôtes, le placement initial des VMs ou la facturation des ressources aux utilisateurs se font aujourd'hui de manière statique sans prise en considération des profils de consommation en ressources des VMs. Disposer d'outils d'automatisation de la caractérisation des profils de consommation en ressources des VMs est un besoin établi dans l'état de l'art actuel.

Nous avons amorcé dans le chapitre 4, une première méthodologie de quantification de la fluctuation de la consommation des VMs. Plusieurs notions méritent d'être approfondies : la fluctuation par intermittence, les impacts de la taille de la fenêtre temporelle glissante sur les indices de fluctuation des VMs, la restitution possible de la fluctuation en fonction des amplitudes et fréquences des pics de consommation.

En particulier, nous proposons d'adapter la taille de la fenêtre glissante en fonction du type de profil fluctuant que l'on cherche à identifier. Cela pourrait permettre de définir en paramètre d'entrée de notre algorithme d'identification de VMs fluctuantes, le type de profil fluctuant que l'on souhaite identifier. Identifier des profils ayant une période de fluctuation donnée peut en particulier être pertinent pour les systèmes de placement des VMs.

Prise en compte proactive de la sécurité

Une approche alternative possible à la supervision réactive telle que celle d'AMAD, consisterait à prévenir les attaques par manipulation malicieuse des quantités de ressources consommées par les VMs de façon proactive en mettant en oeuvre des systèmes de gestion de ressources robustes aux variations de consommation des VMs. La question est de savoir alors, s'il est possible et efficace de chercher à améliorer la robustesse intrinsèque des systèmes de gestion dynamique des ressources en basant leurs décisions non seulement sur des événements instantanés de consommation de ressources mais aussi en intégrant dans l'analyse, possiblement en relatif, les historiques de comportement de consommation en ressources des VMs.

Publications de l'auteur

∗ **Brevets**

— Procédé de détection d'attaques. **Kahina Lazri**, Sylvie Laniepce, N° dépôt : 13 51866, déposé le 01/03/13 (N° référence extension PCT : FR2014/050458).

— Méthode de détection des attaques par migrations intempestives de machines virtuelles. **Kahina Lazri**, Sylvie Laniepce, N° dépôt : 14 50763, déposé le 30/01/14.

∗ **Conférences internationales avec actes et comité de lecture**

— AMAD : Resource Consumption Profile-Aware Attack Detection in IaaS Cloud. **Kahina Lazri**, Sylvie Laniepce, Haiming Zheng, Jalel Ben-Othman. 7th IEEE/ACM International Conference on Utility and Cloud Computing, (**UCC'14**). Décembre 2014, Londre, Royaume-Uni.

— When Dynamic VM Migration Falls Under the Control of VM Users.**Kahina Lazri**, Sylvie Laniepce, Jalel Ben-Othman, 5th IEEE International Conference on Cloud Computing Technology and Science **(CloudCom'13)**. Décembre 2013, Bristol, Royaume-Uni.

— Reconsidering Intrusion Monitoring Requirements in Shared Cloud Platforms. **Kahina Lazri**, Sylvie Laniepce, Jalel Ben-Othman, RaSiem Workshop, 8th IEEE International Conference. Availability, Reliability and Security (**ARES'13**), Septembre 2013, Regensburg, Allemagne.

— Engineering Intrusion Prevention Services for IaaS Clouds : The Way of the Hypervisor. Sylvie Laniepce, Marc Lacoste, Mohammed Kassi-Lahlou, Fabien Bignon, **Kahina**

Lazri, Aurelien Wailly, 7th IEEE International Symposium Service Oriented System Engineering **(SOSE'13)**, March 2013, San Francisco Bay, États-Unis.

∗ **Conférences nationales avec actes et comité de lecture**

— Sécurité de la gestion dynamique des ressources dans le cloud : prise de contrôle sur le déclenchement de migrations automatiques de machines virtuelles. **Kahina Lazri**, Sylvie Laniepce, Haiming Zheng, Jalel Ben-Othman, Symposium sur la Sécurité des Technologies de l'Information et des Communications (**SSTIC'14**), juin 2014, Rennes, France

— Reconsidering Isolation in IaaS Clouds : a Security Perspective. **K. Lazri**, S. Laniepce, J. Ben-othman, 19th conference on Computer Electronics Security Applications Rendez vous (**CESAR'12**), november 2012, Rennes, France

Bibliographie

[1] Aurélien WAILLY, Marc LACOSTE et Hervé DEBAR : Vespa : Multi-layered self-protection for cloud resources. *In Proceedings of the 9th International Conference on Autonomic Computing*, ICAC '12, pages 155–160, New York, NY, USA, 2012. ACM. [cité p. 6]

[2] Michael ARMBRUST, Armando FOX, Rean GRIFFITH, Anthony D. JOSEPH, Randy KATZ, Andy KONWINSKI, Gunho LEE, David PATTERSON, Ariel RABKIN, Ion STOICA et Matei ZAHARIA : A view of cloud computing. *Commun. ACM*, 53(4):50–58, avril 2010. [cité p. 7, 8]

[3] Yanpei CHEN, Vern PAXSON et Randy H. KATZ : Whats new about cloud computing security ? Rapport technique UCB/EECS-2010-5, EECS Department, University of California, Berkeley, Jan 2010. [cité p. 7]

[4] Timothy Grance PETER MELL : The nist definition of cloud computing. recommendations of the national institute of standards and technology, September 2011. [cité p. 7, 8]

[5] http ://aws.amazon.com/fr/ec2/ (last accessed on july 2014). [cité p. 8]

[6] http ://azure.microsoft.com/fr-f2014 (last accessed on july 2014). [cité p. 8]

[7] https ://cloud.google.com/products/compute-engine/ (last accessed on july 2014). [cité p. 8]

[8] https ://cloud.google.com/products/app-engine/ (last accessed on july 2014). [cité p. 9]

[9] http ://www.ibm.com/cloud-computing/us/en/paas.html (last accessed on july 2014). [cité p. 9]

[10] https ://drive.google.com/ (last accessed on july 2014). [cité p. 9]

[11] http ://office.microsoft.com/fr-fr/ (last accessed on july 2014. [cité p. 9]

[12] https ://accounts.google.com/ (last accessed on august 2014). [cité p. 9]

[13] Tian GUO, Upendra SHARMA, Prashant SHENOY, Timothy WOOD et Sambit SAHU : Cost-aware cloud bursting for enterprise applications. *ACM Trans. Internet Technol.*, 13(3):10 :1–10 :24, mai 2014. [cité p. 10]

[14] R. P. GOLDBERG : Architecture of virtual machines. *In Proceedings of the workshop on virtual computer systems*, pages 74–112, New York, NY, USA, 1973. ACM. [cité p. 11]

[15] Gerald J. POPEK et Robert P. GOLDBERG : Formal requirements for virtualizable third generation architectures. *Commun. ACM*, 17(7):412–421, juillet 1974. [cité p. 11]

[16] Rosenblum MENDEL : The reincarnation of virtual machines. *Queue*, 2(5):34–40, juillet 2004. [cité p. 11, 18]

[17] Mendel ROSENBLUM et Tal GARFINKEL : Virtual machine monitors : Current technology and future trends. *Computer*, 38(5):39–47, mai 2005. [cité p. 11]

[18] Mendel ROSENBLUM et Carl WALDSPURGER : I/o virtualization. *Queue*, 9(11):30 :30–30 :39, novembre 2011. [cité p. 11]

[19] Darren ABRAMSON, Jeff JACKSON, Sridhar MUTHRASANALLUR, Gil NEIGER, Greg REGNIER, Rajesh SANKARAN, Ioannis SCHOINAS, Rich UHLIG, Balaji VEMBU et John WEIGERT : Intel Virtualization Technology for directed I/O. *INTEL-TECH*, 10(3):179–192, août 2006. [cité p. 11]

[20] Carl A. WALDSPURGER : Memory resource management in vmware esx server. *SIGOPS Oper. Syst. Rev.*, 36(SI):181–194, décembre 2002. [cité p. 12, 18, 34, 64]

[21] Paul BARHAM, Boris DRAGOVIC, Keir FRASER, Steven HAND, Tim HARRIS, Alex HO, Rolf NEUGEBAUER, Ian PRATT et Andrew WARFIELD : Xen and the art of virtualization. *In Proceedings of the nineteenth ACM symposium on Operating systems principles*, SOSP '03, pages 164–177, New York, NY, USA, 2003. ACM. [cité p. 12, 14, 39]

[22] Jeremy SUGERMAN, Ganesh VENKITACHALAM et Beng-Hong LIM : Virtualizing i/o devices on vmware workstation's hosted virtual machine monitor. *In Proceedings of the General Track : 2002 USENIX Annual Technical Conference*, pages 1–14, Berkeley, CA, USA, 2001. USENIX Association. [cité p. 12]

[23] Keith ADAMS et Ole AGESEN : A comparison of software and hardware techniques for x86 virtualization. *In Proceedings of the 12th international conference on Architectural support for programming languages and operating systems*, ASPLOS-XII, pages 2–13, New York, NY, USA, 2006. ACM. [cité p. 13]

[24] Edouard BUGNION, Scott DEVINE, Mendel ROSENBLUM, Jeremy SUGERMAN et Edward Y. WANG : Bringing virtualization to the x86 architecture with the original vmware workstation. *ACM Trans. Comput. Syst.*, 30(4):12 :1–12 :51, novembre 2012. [cité p. 13]

[25] http ://wiki.qemu.org/mainpage (last accessed on 21 july 2014). [cité p. 13]

[26] Rich UHLIG, Gil NEIGER, Dion RODGERS, Amy L. SANTONI, Fernando C.M. MARTINS, Andrew V. ANDERSON, Steven M. BENNETT, ALAIN, Felix H. LEUNG et Larry SMITH : Intel virtualization technology. *Computer*, 38(5):48–56, 2005. [cité p. 14]

[27] INTEL. : Pci-sig single root i/o virtualization (sr-iov) support in intel virtualization technology for connectivity. Rapport technique. [cité p. 14]

[28] AMD : *AMD64 Virtualization Codenamed "Pacifica" Technology : Secure Virtual Machine Architecture Reference Manual*, May 2005. [cité p. 14]

[29] Tal GARFINKEL et Mendel ROSENBLUM : A virtual machine introspection based architecture for intrusion detection. *In Proceedings of the Network and Distributed Systems Security Symposium*, pages 191–206, 2003. [cité p. 15, 28]

[30] Luis M. VAQUERO, Luis RODERO-MERINO et Rajkumar BUYYA : Dynamically scaling applications in the cloud. *SIGCOMM Comput. Commun. Rev.*, 41(1):45–52, janvier 2011. [cité p. 16, 17, 34]

[31] Yangyang WU et Ming ZHAO : Performance modeling of virtual machine live migration. *In proceedings of the 4th International Conference on cloud computing*, pages 492–499, 2011. [cité p. 17, 34]

[32] Alain TCHANA, Giang Son TRAN, Laurent BROTO, Noel De PALMA et Daniel HAGIMONT : Two levels autonomic resource management in virtualized iaas. *Future Generation Comp. Syst.*, 29(6):1319–1332, 2013. [cité p. 17]

[33] Peter M. CHEN et Brian D. NOBLE : When virtual is better than real. *In Proceedings of the Eighth Workshop on Hot Topics in Operating Systems*, HOTOS '01, pages 133–, Washington, DC, USA, 2001. IEEE Computer Society. [cité p. 18, 28]

[34] ORMANDY : An empirical study into the security exposure to hosts of hostile virtualized environments. *In Proceedings of the CanSecWest Conference*, Vancouver. Bc, April 2007. [cité p. 18]

[35] Lana WACHOWSKI et Andrew Paul WACHOWSKI : The matrix, 1999. [cité p. 19]

[36] Samuel T. KING, Peter M. CHEN, Yi-Min WANG, Chad VERBOWSKI, Helen J. WANG et Jacob R. LORCH : Subvirt : Implementing malware with virtual machines. *In Proceedings of the 2006 IEEE Symposium on Security and Privacy*, SP '06, pages 314–327, Washington, DC, USA, 2006. IEEE Computer Society. [cité p. 19]

[37] Loic DUFLOT, Olivier GRUMELARD, Olivier LEVILLAIN et Benjamin MORRIN : On the limits of hypervisor- and virtual machine monitor-based isolation. *Information Security and Cryptography, pages 349366. Springer Berlin Heidelber*, 2010. [cité p. 19, 34]

[38] Martin MULAZZANI, Sebastian SCHRITTWIESER, Manuel LEITHNER, Markus HUBER et Edgar WEIPPL : Dark clouds on the horizon : using cloud storage as attack vector and online slack space. *In Proceedings of the 20th USENIX conference on Security*, SEC'11, pages 5–5, Berkeley, CA, USA, 2011. USENIX Association. [cité p. 19]

[39] Kostya KORTCHINSKY. : Cloudburst : A vmware guest to host escape story. *In Black Hat USA*, 2009. [cité p. 19]

[40] Nelson ELHAGE. : Virtunoid : A kvm guest ! host privilege escalation exploit. In Black Hat USA, 2011. [cité p. 19]

[41] Zhenghong WANG et Ruby B. LEE : Covert and side channels due to processor architecture. *In Proceedings of the 22Nd Annual Computer Security Applications Conference*, ACSAC '06, pages 473–482, Washington, DC, USA, 2006. IEEE Computer Society. [cité p. 20]

[42] Helmut HLAVACS, Thomas TREUTNER, Jean-Patrick GELAS, Laurent LEFEVRE et Anne-Cecile ORGERIE : Energy consumption side-channel attack at virtual machines in a cloud. *In proceedings of theNinth International Conference on Dependable, Autonomic and Secure Computing*, DASC '11, pages 605–612, Washington, DC, USA, 2011. IEEE Computer Society. [cité p. 20]

[43] Nate LAWSON : Side-channel attacks on cryptographic software. *IEEE Security & Privacy*, 7(6):65–68, 2009. [cité p. 20]

[44] Keisuke OKAMURA et Yoshihiro OYAMA : Load-based covert channels between xen virtual machines. *In Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, pages 173–180, New York, NY, USA, 2010. ACM. [cité p. 20]

[45] Thomas RISTENPART, Eran TROMER, Hovav SHACHAM et Stefan SAVAGE : Hey, you, get off of my cloud : exploring information leakage in third-party compute clouds. *In Proceedings of the 16th ACM conference on Computer and communications security*, CCS '09, pages 199–212, New York, NY, USA, 2009. ACM. [cité p. 21]

[46] Yunjing XU, Michael BAILEY, Farnam JAHANIAN, Kaustubh JOSHI, Matti HILTUNEN et Richard SCHLICHTING : An exploration of l2 cache covert channels in virtualized environments. *In Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, CCSW '11, pages 29–40, New York, NY, USA, 2011. ACM. [cité p. 21]

[47] Fangfei ZHOU, M. GOEL, P. DESNOYERS et R. SUNDARAM : Scheduler vulnerabilities and coordinated attacks in cloud computing. *In proceeding of the 10th IEEE International Symposium on Network Computing and Applications (NCA)*, pages 123 –130, aug. 2011. [cité p. 21]

[48] Ludmila CHERKASOVA, Diwaker GUPTA et Amin VAHDAT : Comparison of the three cpu schedulers in xen. *SIGMETRICS Perform. Eval. Rev.*, 35(2):42–51, septembre 2007. [cité p. 21]

[49] Seehwan YOO, Kuen-Hwan KWAK, Jae-Hyun JO et Chuck YOO : Toward under-millisecond i/o latency in xen-arm. *In Proceedings of the Second Asia-Pacific Workshop on Systems*, APSys '11, pages 14 :1–14 :5, New York, NY, USA, 2011. ACM. [cité p. 22]

[50] Jeanna Neefe MATTHEWS, Wenjin HU, Madhujith HAPUARACHCHI, Todd DESHANE, Demetrios DIMATOS, Gary HAMILTON, Michael MCCABE et James OWENS : Quantifying the performance isolation properties of virtualization systems. *In Proceedings of workshop on Experimental computer science*, ExpCS '07, New York, NY, USA, 2007. ACM. [cité p. 22]

[51] Sean Kenneth BARKER et Prashant SHENOY : Empirical evaluation of latency-sensitive application performance in the cloud. *In Proceedings of the first annual ACM SIGMM conference on Multimedia systems*, MMSys '10, pages 35–46, New York, NY, USA, 2010. ACM. [cité p. 23]

[52] Joseph IDZIOREK, Mark F. TANNIAN et Doug JACOBSON : The insecurity of cloud utility models. *IT Professional*, 15(2):22–27, March 2013. [cité p. 23]

[53] J. IDZIOREK et M. TANNIAN : Exploiting cloud utility models for profit and ruin. *In proceedings of the 4th International Conference on cloud computing*, pages 33–40, July 2011. [cité p. 23]

[54] J. IDZIOREK, M. TANNIAN et D. JACOBSON : Attribution of fraudulent resource consumption in the cloud. *In proceeding of the IEEE 5th International Conference onCloud Computing (CLOUD"12).*, pages 99–106, June 2012. [cité p. 23]

[55] Karen SCARFONE, Karen SCARFONE, Scarfone CYBERSECURITY, Peter MELL, Rebecca M. BLANK et Acting SECRETARY : Guide to intrusion detection and prevention systems (idps, 2007. [cité p. 25, 27]

[56] Hervé DEBAR, Marc DACIER et Andreas WESPI : Towards a taxonomy of intrusion-detection systems. *Comput. Netw.*, 31(9):805–822, avril 1999. [cité p. 25]

[57] Varun CHANDOLA, Arindam BANERJEE et Vipin KUMAR : Anomaly detection : A survey. *ACM Comput. Surv.*, 41(3):15 :1–15 :58, juillet 2009. [cité p. 26]

[58] Sandeep KUMAR et Eugene H. SPAFFORD : A pattern matching model for misuse intrusion detection. *In Proceedings of the 17th National Computer Security Conference*, pages 11–21, 1994. [cité p. 26]

[59] B.D. PAYNE, M. CARBONE, M. SHARIF et Wenke LEE : Lares : An architecture for secure active monitoring using virtualization. *In proceeding of the IEEE Symposium onSecurity and Privacy*, pages 233–247, May 2008. [cité p. 28]

[60] A.S. IBRAHIM, J. HAMLYN-HARRIS, J. GRUNDY et M. ALMORSY : Cloudsec : A security monitoring appliance for virtual machines in the iaas cloud model. *In proceeding of the 5th International Conference on Network and System Security (NSS)*, pages 113 –120, sept. 2011. [cité p. 28]

[61] Jonas PFOH, Christian SCHNEIDER et Claudia ECKERT : A formal model for virtual machine introspection. *In Proceedings of the 2nd Workshop on Virtual Machine Security (VMSec '09)*, pages 1–10, Chicago, Illinois, USA, novembre 2009. ACM Press. [cité p. 28]

[62] Stephen T. JONES, Andrea C. ARPACI-DUSSEAU et Remzi H. ARPACI-DUSSEAU : Antfarm : Tracking processes in a virtual machine environment. *In In proceeding of the USENIX Annual Technical Conference*, 2006. [cité p. 29]

[63] S. BAHRAM, Xuxian JIANG, Zhi WANG, M. GRACE, Jinku LI, D. SRINIVASAN, Junghwan RHEE et Dongyan XU : Dksm : Subverting virtual machine introspection for fun and profit. *In proceeding of the 29th IEEE Symposium on Reliable Distributed Systems*, pages 82–91, Oct 2010. [cité p. 29]

[64] Jonas PFOH, Christian SCHNEIDER et Claudia ECKERT : Nitro : Hardware-based system call tracing for virtual machines. *In Advances in Information and Computer Security*, volume 7038 de *Lecture Notes in Computer Science*, pages 96–112. Springer, novembre 2011. [cité p. 29]

[65] A.M. AZAB, Peng NING, E.C. SEZER et Xiaolan ZHANG : Hima : A hypervisor-based integrity measurement agent. *In proceeding of the Annual Computer Security Applications Conference, 2009. ACSAC '09*, pages 461 –470, dec. 2009. [cité p. 30]

[66] Brian HAY et Kara NANCE : Forensics examination of volatile system data using virtual introspection. *SIGOPS Oper. Syst. Rev.*, 42(3):74–82, avril 2008. [cité p. 30]

[67] Ryan RILEY, Xuxian JIANG et Dongyan XU : Guest-transparent prevention of kernel rootkits with vmm-based memory shadowing. *In proceeding of the International Symposium on Recent Advances in Intrusion Detection, RAID'08*, 2008. [cité p. 30]

[68] Ashlesha JOSHI, Samuel T. KING, George W. DUNLAP et Peter M. CHEN : Detecting past and present intrusions through vulnerability-specific predicates. *SIGOPS Oper. Syst. Rev.*, 39(5):91–104, octobre 2005. [cité p. 30]

[69] Artem DINABURG, Paul ROYAL, Monirul SHARIF et Wenke LEE : Ether : Malware analysis via hardware virtualization extensions. *In Proceedings of the 15th ACM Conference on Computer and Communications Security*, CCS '08, pages 51–62, New York, NY, USA, 2008. ACM. [cité p. 30]

[70] http ://libvirt.org.(last accessed on april 2014). [cité p. 30]

[71] Abhinav SRIVASTAVA et Jonathon GIFFIN : Tamper-resistant, application-aware blocking of malicious network connections. *In Proceedings of the 11th international symposium on Recent Advances in Intrusion Detection*, RAID '08, pages 39–58, Berlin, Heidelberg, 2008. Springer-Verlag. [cité p. 30]

[72] I.S. MORENO, P. GARRAGHAN, P. TOWNEND et Jie XU : An approach for characterizing workloads in google cloud to derive realistic resource utilization models. *In Service Oriented System Engineering (SOSE), 2013 IEEE 7th International Symposium on*, pages 49–60, 2013. [cité p. 34, 105]

[73] K. LAZRI, S. LANIEPCE et J. BEN-OTHMAN : Reconsidering intrusion monitoring requirements in shared cloud platforms. *In proceeding of theEighth International Conference on Availability, Reliability and Security (ARES)*, pages 630–637, Sept 2013. [cité p. 34]

[74] Nelson ELHAGE : Virtunoid : A kvm guest ! host privilege escalation exploit. In Black Hat USA, 2011. [cité p. 34]

[75] Akshat VERMA, Gautam KUMAR, Ricardo KOLLER et Aritra SEN : Cosmig : Modeling the impact of reconfiguration in a cloud. *In proceedings 19th Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems*, MASCOTS '11, pages 3–11, Washington, DC, USA, 2011. IEEE Computer Society. [cité p. 34]

[76] Yanpei CHEN, Vern PAXSON et Randy H. KATZ : Whats new about cloud computing security. Rapport technique UCB/EECS-2010-5, EECS Department, University of California, Berkeley, Jan 2010. [cité p. 35]

[77] Michael NELSON, Beng-Hong LIM et Greg HUTCHINS : Fast transparent migration for virtual machines. *In Proceedings of the annual conference on USENIX Annual Technical Conference*, ATEC '05, pages 25–25, Berkeley, CA, USA, 2005. USENIX Association. [cité p. 36, 50]

[78] Michael R. HINES, Umesh DESHPANDE et Kartik GOPALAN : Post-copy live migration of virtual machines. *SIGOPS Oper. Syst. Rev.*, 43(3):14–26, juillet 2009. [cité p. 37]

[79] Wenjin HU, Andrew HICKS, Long ZHANG, Eli M. DOW, Vinay SONI, Hao JIANG, Ronny BULL et Jeanna N. MATTHEWS : A quantitative study of virtual machine live migration. *In Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference*, CAC '13, pages 11 :1–11 :10, New York, NY, USA, 2013. ACM. [cité p. 37]

[80] William VOORSLUYS, James BROBERG, Srikumar VENUGOPAL et Rajkumar BUYYA : Cost of virtual machine live migration in clouds : A performance evaluation. *In Proceedings of the 1st International Conference on Cloud Computing*, CloudCom '09, pages 254–265, Berlin, Heidelberg, 2009. Springer-Verlag. [cité p. 37]

[81] Zhang XU, Haining WANG, Zichen XU et Xiaorui WANG : Power attack : An increasing threat to data centers. *In proceedings of the 2014 Network and Distributed System Security Symposium, NDSS'14*, February 2014. [cité p. 38]

[82] M. MISHRA et A. SAHOO : On theory of vm placement : Anomalies in existing methodologies and their mitigation using a novel vector based approach. *In Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 275–282, July 2011. [cité p. 38]

[83] M. MISHRA, A. DAS, P. KULKARNI et A. SAHOO : Dynamic resource management using virtual machine migrations. *Communications Magazine, IEEE*, 50(9):34–40, 2012. [cité p. 39]

[84] Timothy WOOD, Prashant SHENOY, Arun VENKATARAMANI et Mazin YOUSIF : Black-box and gray-box strategies for virtual machine migration. *In Proceedings of the 4th USENIX conference on Networked systems design and implementation*, NSDI'07, pages 17–17, Berkeley, CA, USA, 2007. USENIX Association. [cité p. 39, 44]

[85] Xiangliang ZHANG, Zon-Yin SHAE, Shuai ZHENG et Hani JAMJOOM : Virtual machine migration in an over-committed cloud. *In NOMS'12*, pages 196–203, 2012. [cité p. 40]

[86] Albert GREENBERG, James HAMILTON, David A. MALTZ et Parveen PATEL : The cost of a cloud : Research problems in data center networks. *SIGCOMM Comput. Commun. Rev.*, 39(1):68–73, décembre 2008. [cité p. 41]

[87] Tiago C. FERRETO, Marco A. S. NETTO, Rodrigo N. CALHEIROS et César A. F. DE ROSE : Server consolidation with migration control for virtualized data centers. *Future Gener. Comput. Syst.*, 27(8):1027–1034, octobre 2011. [cité p. 41, 44]

[88] Akshat VERMA, Puneet AHUJA et Anindya NEOGI : pmapper : Power and migration cost aware application placement in virtualized systems. *In proceedings of the 9th ACM/IFIP/USENIX*

International Conference on Middleware, Middleware '08, pages 243–264, New York, NY, USA, 2008. Springer-Verlag New York, Inc. [cité p. 41]

[89] Fabien HERMENIER, Xavier LORCA, Jean-Marc MENAUD, Gilles MULLER et Julia LAWALL : Entropy : A consolidation manager for clusters. *In Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, VEE '09, pages 41–50, New York, NY, USA, 2009. ACM. [cité p. 42]

[90] Jing XU et Jose A. B. FORTES : Multi-objective virtual machine placement in virtualized data center environments. *In Proceedings of the 2010 IEEE/ACM Conference on Green Computing and Communication Conference on Cyber, Physical and Social Computing*, GREENCOM-CPSCOM '10, pages 179–188, Washington, DC, USA, 2010. IEEE Computer Society. [cité p. 42, 44]

[91] Emanuel FALKENAUER : A hybrid grouping genetic algorithm for bin packing, 1996. [cité p. 42]

[92] Jiankang DONG, Xing JIN, Hongbo WANG, Yangyang LI, Peng ZHANG et Shiduan CHENG : Energy-saving virtual machine placement in cloud data centers. *In Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*, pages 618–624, May 2013. [cité p. 42]

[93] Rainer E. Burkard ER et Leonidas S. PITSOULIS : The quadratic assignment problem ? [cité p. 42]

[94] A. SINGH, M. KORUPOLU et D. MOHAPATRA : Server-storage virtualization : Integration and load balancing in data centers. *In High Performance Computing, Networking, Storage and Analysis, 2008. SC 2008. International Conference for*, pages 1–12, 2008. [cité p. 43, 44]

[95] Emmanuel ARZUAGA et David R. KAELI : Quantifying load imbalance on virtualized enterprise servers. *In Proceedings of the first joint WOSP/SIPEW international conference on Performance engineering*, WOSP/SIPEW '10, pages 235–242, New York, NY, USA, 2010. ACM. [cité p. 43, 44]

[96] Ajay GULATI, Anne HOLLER, Minwen JI, Ganesha SHANMUGANATHAN, Carl WALDSPURGER et Xiaoyun ZHU : Vmware distributed resource management : Design, implementation, and lessons learned, 2012. [cité p. 44, 45]

[97] Epping DUNCAN et Frank DENNEMAN : *VMware vSphre 4.1 HA and DRS technical deepdive*. 2010. [cité p. 44]

[98] Hiep NGUYEN, Yongmin TAN et Xiaohui GU : Pal : Propagation-aware anomaly localization for cloud hosted distributed applications. *In proceeding of the Managing Systems via Log Analysis and Machine Learning Techniques*, SLAML '11, pages 1 :1–1 :8, New York, NY, USA, 2011. ACM. [cité p. 64]

[99] Hui KANG, Haifeng CHEN et Guofei JIANG : Peerwatch : A fault detection and diagnosis tool for virtualized consolidation systems. *In Proceedings of the 7th International Conference on Autonomic Computing*, ICAC '10, pages 119–128, New York, NY, USA, 2010. ACM. [cité p. 65]

[100] Hiep NGUYEN, Zhiming SHEN, Yongmin TAN, Xiaohui GU et Mathworks INC : Fchain : Toward black-box online fault localization for cloud systems. *In proceeding of the IEEE 33rd International Conference on Distributed Computing Systems ICDCS'133*, 2013. [cité p. 65]

[101] Paramvir BAHL, Ranveer CHANDRA, Albert GREENBERG, Srikanth KANDULA, David MALTZ et Ming ZHANG : Towards highly reliable enterprise network services via inference of multi-level dependencies. *In SIGCOMM*. Association for Computing Machinery, Inc., August 2007. [cité p. 66]

[102] Yongmin TAN, Hiep NGUYEN, Zhiming SHEN, Xiaohui GU, C. VENKATRAMANI et D. RAJAN : Prepare : Predictive performance anomaly prevention for virtualized cloud systems. *In In proceeding of the 32nd International Conference on Distributed Computing Systems ICDCS'12*, pages 285–294, June 2012. [cité p. 66]

[103] Pierre BREMAUD : *Markov Chains : Gibbs Fields, Monte Carlo Simulation, and Queues*. Texts in Applied Mathematics. Springer, 1999. [cité p. 66]

[104] Ira COHEN, Moises GOLDSZMIDT, Terence KELLY, Julie SYMONS et Jeffrey S. CHASE : Correlating instrumentation data to system states : A building block for automated diagnosis and control. *In Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation - Volume 6*, OSDI'04, pages 16–16, Berkeley, CA, USA, 2004. USENIX Association. [cité p. 66]

[105] Ripal NATHUJI, Aman KANSAL et Alireza GHAFFARKHAH : Q-clouds : managing performance interference effects for qos-aware clouds. *In Proceedings of the 5th European conference on Computer systems*, EuroSys '10, pages 237–250, New York, NY, USA, 2010. ACM. [cité p. 67, 68]

[106] Hui KANG, Xiaoyun ZHU et Jennifer L. WONG : Dapa : diagnosing application performance anomalies for virtualized infrastructures. *In Proceedings of the 2nd USENIX conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services*, Hot-ICE'12, pages 8–8, Berkeley, CA, USA, 2012. USENIX Association. [cité p. 67]

[107] R.M. ESTEVES, T. HACKER et Chunming RONG : Competitive k-means, a new accurate and distributed k-means algorithm for large datasets. *In proceeding of the 2013 IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom'13),*, volume 1, pages 17–24, Dec 2013. [cité p. 67, 75]

[108] Pengcheng XIONG, Calton PU, Xiaoyun ZHU et Rean GRIFFITH : vperfguard : An automated model-driven framework for application performance diagnosis in consolidated cloud environments. *In Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering*, ICPE '13, pages 271–282, New York, NY, USA, 2013. ACM. [cité p. 67]

[109] Qiang GUAN et Song FU : Adaptive anomaly identification by exploring metric subspace in cloud computing infrastructures. *In proceedings of the IEEE Symposium on Reliable Distributed Systems*, 0:205–214, 2013. [cité p. 68, 70]

[110] Richard O. DUDA, Peter E. HART et David G. STORK : *Pattern Classification (2Nd Edition)*. Wiley-Interscience, 2000. [cité p. 68]

[111] R. E. KALMAN : A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME Journal of Basic Engineering*, (82 (Series D)):35–45, 1960. [cité p. 68]

[112] Moldovan DANIEL, Copil GEORGIANA, Truong HONG-LINH et Schahram DUSTD : Mela : Monitoring and analyzing elasticity of cloud services. *In proceeding of the 5th IEE International Conference on Cloud Computing Technology and Science*, December 2013. [cité p. 68]

[113] B. SHARMA, P. JAYACHANDRAN, A. VERMA et C.R. DAS : Cloudpd : Problem determination and diagnosis in shared dynamic clouds. *In proceeding of the 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'13)*, pages 1–12, June 2013. [cité p. 69]

[114] R. BIRKE, A. PODZIMEK, L.Y. CHEN et E. SMIRNI : State-of-the-practice in data center virtualization : Toward a better understanding of vm usage. *In Dependable Systems and Networks (DSN), 2013 43rd Annual IEEE/IFIP International Conference on*, pages 1–12, June 2013. [cité p. 72, 107]

[115] K. LAZRI, S. LANIEPCE et J. BEN-OTHMAN : When dynamic vm migration falls under the control of vm users. *In proceeding of the IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom'13)*, volume 1, pages 395–402, Dec 2013. [cité p. 78]

[116] Galen ANDREW, Raman ARORA, Jeff BILMES et Karen LIVESCU : Deep canonical correlation analysis. *In* Sanjoy DASGUPTA et David MCALLESTER, éditeurs : *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 28, pages 1247–1255. JMLR Workshop and Conference Proceedings, mai 2013. [cité p. 78]

[117] Harold HOTELLING : Relations Between Two Sets of Variates. *Biometrika*, 28(3/4):321–377, décembre 1936. [cité p. 79]

[118] Vyas SEKAR et Petros MANIATIS : Verifiable resource accounting for cloud computing services. *In Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, CCSW '11, pages 21–26, New York, NY, USA, 2011. ACM. [cité p. 102]

[119] Jeffrey C. MOGUL : Operating systems should support business change. *In Proceedings of the 10th Conference on Hot Topics in Operating Systems - Volume 10*, HOTOS'05, pages 8–8, Berkeley, CA, USA, 2005. USENIX Association. [cité p. 102]

[120] http ://code.google.com/p/googleclusterdata/wiki/clusterdata2011. [cité p. 103]

[121] http ://code.google.com/p/googleclusterdata/wiki/traceversion2. [cité p. 104]

[122] Charles REISS, John WILKES et Joseph L. HELLERSTEIN : Google cluster-usage traces : format + schema. Technical report, Google Inc., Mountain View, CA, USA, novembre 2011. [cité p. 104]

[123] Zitao LIU et Sangyeun CHO : Characterizing machines and workloads on a google cluster. *In proceeding of the 41st International Conference on Parallel Processing Workshops (ICPPW' 12)*, pages 397–403, Sept 2012. [cité p. 104]

[124] S. KAVULYA, J. TAN, R. GANDHI et P. NARASIMHAN : An analysis of traces from a production mapreduce cluster. *In In proceeding of the 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid'10)*, pages 94–103, May 2010. [cité p. 104]

[125] C. REISS, J. WILKES et J.L. HELLERSTEIN : Obfuscatory obscanturism : Making workload traces of commercially-sensitive systems safe to release. *In proceeding of the IEEE Network Operations and Management Symposium (NOMS'12)*, pages 1279–1286, April 2012. [cité p. 104]

[126] Qi ZHANG, Joseph HELLERSTEIN et Raouf BOUTABA : Characterizing task usage shapes in google compute clusters. *In Proceedings of the 5th International Workshop on Large Scale Distributed Systems and Middleware*, 2011. [cité p. 104]

[127] Guanying WANG, Ali R. BUTT, Henry MONTI et Karan GUPTA : Towards synthesizing realistic workload traces for studying the Hadoop ecosystem. *In proceeding of the 19th IEEE Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS'11)*, pages 400–408, Raffles Hotel, Singapore, juillet 2011. IEEE Computer Society. [cité p. 104]

[128] Sheng DI, Derrick KONDO et Walfredo CIRNE : Host load prediction in a Google compute cloud with a Bayesian model. *In proceeding of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC'12)*, pages 21 :1–21 :11, Salt Lake City, UT, USA, novembre 2012. IEEE Computer Society Press. [cité p. 104]

[129] Asit K. MISHRA, Joseph L. HELLERSTEIN, Walfredo CIRNE et Chita R. DAS : Towards characterizing cloud backend workloads : insights from google compute clusters. *SIGMETRICS Perform. Eval. Rev.*, 37(4):34–41, mars 2010. [cité p. 105]

[130] Charles REISS, Alexey TUMANOV, Gregory R. GANGER, Randy H. KATZ et Michael A. KOZUCH : Heterogeneity and dynamicity of clouds at scale : Google trace analysis. *In Proceedings of the Third ACM Symposium on Cloud Computing*, SoCC '12, pages 7 :1–7 :13, New York, NY, USA, 2012. ACM. [cité p. 105]

[131] Bikash SHARMA, Victor CHUDNOVSKY, Joseph L. HELLERSTEIN, Rasekh RIFAAT et Chita R. DAS : Modeling and synthesizing task placement constraints in Google compute clusters. *In proceeding of the 2nd ACM Symposium on Cloud Computing (SoCC'11)*, pages 3 :1–3 :14, Cascais, Portugal, octobre 2011. ACM. [cité p. 105]

[132] Sheng DI, Derrick KONDO et Walfredo CIRNE : Characterization and comparison of cloud versus Grid workloads. *In porceeding of the IEEE International Conference on Cluster Computing (CLUSTER'12)*, pages 230–238, Beijing, China, septembre 2012. [cité p. 106]

[133] Robert BIRKE, Lydia Y. CHEN et Evgenia SMIRNI : Usage patterns in multi-tenant data centers : A temporal perspective. *In Proceedings of the 9th International Conference on Autonomic Computing*, ICAC '12, pages 161–166, New York, NY, USA, 2012. ACM. [cité p. 106]

[134] R. GHOSH et V.K. NAIK : Biting off safely more than you can chew : Predictive analytics for resource over-commit in iaas cloud. *In In proceeding of the 2012 5th IEEE International Conference onCloud Computing (CLOUD'12).*, pages 25–32, June 2012. [cité p. 106]

[135] R. BIRKE, L.Y. CHEN, M. GRIBAUDO et P. PIAZZOLLA : Characterization analysis of resource utilization distribution. *In proceeding of the 21st IEEE International Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS'13)*, pages 370–374, Aug 2013. [cité p. 107]

[136] Leonard KLEINROCK : *Queueing Systems*, volume I : Theory. Wiley Interscience, 1975. (Published in Russian, 1979. Published in Japanese, 1979. Published in Hungarian, 1979. Published in Italian 1992.). [cité p. 107]

Liste des abréviations

AMAD	Abusive VM Migration Attack Detection
API	Application Programming Interface
CCA	Canonical Correlation Analysis
*CR*3	Control Register 3
DRS	Distriuted Resource Scheduler
*EC*2	Elastic Computing
HPC	High Performance Computing
DMA	Direct Memory Access
HIPS	Host based Intrusion Detection System
HMM	Hidden Markov Models
IaaS	Infrastructure as a Service
IDS	Intrusion Detection System
IP	Internet Protocol
kNN	Nearest Neighbor
IPS	Intrusion Prevention System
MIMO	Multi Input Multi Output
NIST	National Institute of Standards and Technology
NIDS	Network based Intrusion Detection System
PaaS	Platform as a Service
PCA	Principal Component Analysis
SaaS	Software as a Service
SLA	Service Level Agreement
VMM	Vitual Machine Monitor
VMCS	Virtual Machine Control Structure

Table des figures

Liste des tableaux

La virtualisation matérielle telle que mise en oeuvre dans le cloud computing, permet le partage de ressources matérielles entre plusieurs machines virtuelles pouvant appartenir à différents utilisateurs. Ce partage des ressources constitue l'atout majeur de ces infrastructures, qui permet aux fournisseurs d'exploiter plus efficacement les ressources des centres de données, notamment à travers l'allocation dynamique des ressources. Cependant, le partage des ressources introduit de nouvelles contraintes de sécurité. Plusieurs travaux de l'état de l'art ont démontré l'apparition de nouvelles stratégies d'attaques propres aux infrastructures cloud computing, exploitant le partage des ressources. Néanmoins, il a aussi été démontré qu'il est possible de tirer avantage de la position privilégiée de la couche de virtualisation pour offrir une meilleure sécurité que celle assurée dans les plate-formes traditionnelles d'hébergement en silo.

Cette thèse poursuit deux axes de recherche complémentaires. Le premier axe traite des nouvelles vulnérabilités liées aux infrastructures cloud computing. Nous avons démontré une attaque que nous appelons attaque par 'migrations intempestives de machines virtuelles', dans laquelle un attaquant parvient à amener le système de gestion dynamique de ressources à migrer de façon abusive des machines virtuelles, par simple manipulation des quantités de ressources consommées par des machines virtuelles qui sont sous son contrôle. Nous avons démontré cette attaque sur une plate-forme constituée de cinq serveurs et analysé les conditions nécessaires à son succès ainsi que l'exposition des clusters vis-à-vis de la vulnérabilité qu'elle exploite.

Le deuxième axe propose de tirer avantage de la position privilégiée de l'opérateur qui dispose à la fois d'une vue multi-couches plus riche de l'utilisation des ressources et d'une vue plus globale des contextes d'exécution des machines virtuelles, comparativement à la vue limitée de l'utilisateur, pour offrir une meilleure sécurité. Nous avons proposé AMAD (Abusive VM Migration Attack Detection), un système de supervision, chargé de détecter l'occurrence des attaques par migrations intempestives de machines virtuelles et d'identifier de façon automatique celles à l'origine de l'attaque. AMAD est implémenté sur notre plate-forme d'expérimentation et évalué à l'aide de traces de consommation de machines virtuelles collectées sur des clouds réels. Les résultats d'évaluation montrent qu'AMAD opère avec une bonne précision de détection.

Title : Resource consumption profile-based attack detection in IaaS clouds

Hardware virtualisation is the core technology which enables resource sharing among multiple virtual machines possibly belonging to different tenants within cloud infrastructures. Resource sharing is the main feature that enables cost effectiveness of cloud platforms, achieved through dynamic resource management. However, resource sharing brings several new security concerns. Several proofs of concepts have demontrated new attack strategies brought by the resource sharing paradigm, known as cross-virtual machine attacks. Even so, it is also demonstated that the priviligied position of the virtualisation layer can be leveraged to offer better security protection mecanisms than the ones offered in non virtualized platfoms.

This thesis follows two main objectives. The first one is related to the domain of cloud-specific vulnerabilities. We have demonstrated a new attack, called the abusive virtual machine migration attack, in which an attacker can leverage the sharing of resources, through the manipulation of the amounts of resources consumed by virtual machines under his control, to abusively enforce the dynamic resource management system to trigger virtual machine migrations. We have demonstrated this attack on a virtualized platform composed of five physical machines, the conditions necessary for the attack to succeed and the vulnerability exposure of clusters against this kind of attack is also analyzed.

The second main contribution of this thesis aims at leveraging the privilged position of the cloud provider who has both a more reliable view of the ressource utilisation and a more complete view of the virtual machine execution contexts compared to the limited view of cloud users, to provide better security. We propose AMAD (Abusive Virtual Machine Migration Attack Detection), a system designed for detecting an abusive use of the dynamic virtual machine migration, in the case of the abusive virtual machine migration attack. AMAD identifies the virtual machines possibly at the origin of the attack by analyzing their resource consumption profiles which show fluctuation and correlation in the usage of resources. We have implemented AMAD on top of our laboratory platform and evaluated it with the help of virtual machine resource consumption traces collected from real cloud. Our evaluation results show that AMAD identifes the attacking virtual machines with high detection accuracy.