Sécurité de la gestion dynamique des ressources dans le cloud : prise de contrôle sur le déclenchement de migrations automatiques de machines virtuelles

Kahina Lazri, Haiming Zheng, Sylvie Laniepce et Jalel Ben-Othman
kahina.lazri@orange.com
hzheng.ext@orange.com
s.laniepce@orange.com
jalel.ben-othman@univ-paris13.fr

Orange Labs, Laboratoire L2TI de l'université Paris Nord

Résumé De nouvelles vulnérabilités liées au partage des ressources apparaissent avec les infrastructures cloud multi-locataires. Dans cet article, nous examinons comment le partage des ressources peut être exploité par simple manipulation des quantités de ressources consommées par les machines virtuelles, pour amener le système de gestion dynamique des ressources à déclencher des migrations de machines virtuelles. De telles migrations intempestives consomment inutilement des ressources de l'infrastructure et impactent les performances des machines virtuelles migrées.

Pour démontrer cette attaque cross-machines virtuelles, nous utilisons une plate-forme VMware qui met en oeuvre un algorithme de gestion des migrations automatiques de machines virtuelles, i.e. Distributed Resource Scheduling (DRS). Nous analysons dans le détail le déroulement de nos expérimentations en surveillant le fonctionnement de DRS pendant toute la durée de 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 engendrer une série de migrations.

1 Introduction

Dans les plate-formes cloud, les ressources sont gérées dynamiquement par logiciel grâce à la couche de virtualisation (hyperviseur) qui s'interpose entre les ressources physiques des serveurs hôtes et les environnements d'exécution des utilisateurs, i.e. les machines virtuelles consommant les ressources virtualisées qui leur sont présentées par l'hyperviseur. Les utilisateurs de machines virtuelles surestiment leurs besoins en ressources

[15] et l'optimisation des taux d'utilisation des ressources par les opérateurs d'infrastructure cloud repose notamment sur le surengagement de ressources [22]. Le surengagement de ressources consiste à configurer pour l'ensemble des machines virtuelles hébergées sur un même hôte une quantité totale de ressources supérieure à la quantité de ressources physiques réellement disponible sur cet hôte, en escomptant que l'ensemble des machines virtuelles n'aient pas à consommer au même moment la totalité de leur ressources configurées faute de quoi des contentions de ressources apparaissent. La virtualisation permet un certain nombre de mécanismes pour faire face aux contentions, le cas échéant, en particulier la migration automatique de machines virtuelles pour augmenter la disponibilité en ressources d'un serveur hôte qui se voit libérée de la machine virtuelle migrée [16].

La consolidation sur un même hôte de machines virtuelles appartenant à des locataires différents pose un certain nombre de défis en termes de sécurité pour les plate-formes du cloud [20] [12]. Il a été démontré que sous certaines conditions, le partage des ressources permet à un attaquant ayant le contrôle d'une machine virtuelle d'attaquer une machine virtuelle co-localisée avec la sienne [6] [8]. Une isolation forte entre les machines virtuelles est une nécessité pour garantir leur sécurité.

Dans cet article, nous examinons comment un attaquant peut influencer le système de gestion des ressources pour l'amener de manière abusive à déclencher des migrations de machines virtuelles. Plus précisément, nous étudions comment un attaquant peut manipuler les quantités de ressources consommées par ses propres machines virtuelles (ou des machines virtuelles sous son contrôle) durant leur exécution, pour impacter les décisions de migration du système de gestion des resources. Les migrations consomment des resources de l'infrastructure tant au niveau de l'hôte cédant la machine virtuelle qu'au niveau de l'hôte destinataire de la machine virtuelle [24] ou encore en termes de bande passante réseau pour transmettre l'état courant des machines virtuelles migrées ; ces migrations sont donc coûteuses pour l'infrastructure. De plus, les machines virtuelles migrées – pas nécessairement celles manipulées par l'attaquant – subissent des dégradations de performances dues au processus de migration.

De nombreux travaux se sont intéressés à l'optimisation, l'évaluation ou encore la modélisation de la migration dynamique de machines virtuelles mais celle-ci n'a jamais été considérée sous l'angle de la sécurité à notre connaissance. Dans cet article, 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 machines virtuelles hébergées.

Au delà de l'attaque particulière par migration intempestive de machines virtuelles ici présentée, ces travaux mettent en évidence la vulnérabilité des sytèmes de gestion dynamique des ressources vis-à-vis de profils de consommation de ressources malveillants. Notre raisonnement s'appuie sur deux constats. Ces systèmes réagissent par nature à la consommation de ressources des machines virtuelles qui sont possiblement sous le contrôle d'attaquants. La multi-location crée une interdépendance entre les machines virtuelles [5] qui, dans le cas particulier du gestion dynamique des ressources, fait subire aux machines virtuelles co-localisées avec celles attaquantes des dégradations de performances.

Dans cet article, nous détaillons les contributions suivantes : 1) démonstration par l'expérimentation que les systèmes de gestion dynamique de resources peuvent être vulnérables à la manipulation malveillante de la consommation de ressources des machines virtuelles, 2) anlayse détaillée de l'algorithme Distributed Resource Scheduler (DRS) que nous avons utilisé pour démontrer l'attaque par migration intempestive de machines virtuelles et, 3) évaluation par la mesure de la vulnérabilité de clusters - considérés en tant qu'ensembles d'hôtes - vis-à-vis de l'attaque dans le cas de DRS.

L'article est structurée comme suit. La section II livre des considérations générales sur les systèmes de gestion dynamique des ressources. La section III donne une description de l'algorithme DRS nécessaire à la compréhension de l'attaque. La section IV présente l'attaque. La section V décrit en détail l'exécution de l'attaque par migration intempestive de machines virtuelles et évalue les conditions de réussite de l'attaque, au travers d'une pluralité d'expérimentations. La section VI livre une analyse de l'état de l'art et la section VII conclut l'article.

2 Généralités sur les systèmes de gestion dynamique des ressources

L'optimisation de l'utilisation des ressources est un objectif majeur des systèmes de gestion des ressources du cloud, obtenue par consolidation des machines virtuelles sur les serveurs hôtes qui les hébergent grâce à une couche d'indirection - la couche de virtualisation - entre les ressources physiques et celles virtualisées présentées aux machines virtuelles. Cette indirection permet de dimensionner en relatif entre les différentes machines virtuelles hébergées sur un hôte, les quantités de ressources configurées

pour chacune d'entre elles soit par ajustement des quantités configurées en fonction des besoins variables dans le temps des machines virtuelles soit par migration de machines virtuelles.

La migration d'une machine virtuelle consiste à transférer de manière itérative son état courant d'un hôte source vers un hôte destination alors que la machine virtuelle est en cours d'execution [16]. Le méchanisme comprend deux phases, 1) une phase de pré-copie qui consiste à copier l'état courant de la machine virtuelle alors qu'elle continue de s'éxecuter sur l'hôte source, chaque page copiée modifiée par la machine virtuelle au cours de cette phase est copiée de nouveau, 2) une phase de 'stop et copie' pour éviter de recopier indéfiniment les pages modifiées, l'exécution de la machine virtuelle est suspendue sur l'hôte source pour empêcher toute nouvelle modification de page et une copie finale des pages modifées est réalisée. La machine virtuelle peut alors reprendre son exécution sur l'hôte de destination.

Pendant la phase de migration, les performances de la machine virtuelle migrée peuvent être affectées. Dans l'article [21], les auteurs rapportent que l'exécution des machines virtuelles peut être supsendue (le terme anglais est le 'down time', correspondant à la phase 2 décrite ci-dessus) pendant une durée variant de 60 ms à 3 secondes selon la nature des applications exécutées par la machine virtuelle, et l'exécution ralentie de 20% pendant toute la durée du processus (le terme anglais est le 'migration time', correspondant aux phases 1 et 2) pour une machine virtuelle de 800 MB de mémoire. La durée totale du processus dépend de la nature des applications exécutées par la machine virtuelle, de la capacité de la machine virtuelle et du niveau de contention des hôtes source et destination de la machine virtuelle migrée. La migration est donc coûteuse 1) pour l'hyperviseur du fait des ressources consommées pour l'éxécution de la migration, 2) pour la machine virtuelle migrée et, 3) possiblement pour les machines virtuelles co-localisées si l'activité due migrations au niveau des hyperviseurs s'ajoute à des états contentionnés. Enfin, dans l'article [25], les auteurs démontrent que les migrations de machines virtuelles 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.

Nous examinons ci-après divers objectifs poursuivis par les algorithmes de gestion dynamique de ressources mettant en oeuvre des méchanismes de migration dynamique de machines virtuelles [14] :

- Migration pour la gestion des contentions [23][26][4][19] : les contentions de ressources imputables à l'opérateur de l'infrastructure surviennent lorsqu'une machine virtuelle dispose d'une quantité de ressources insuffisante sur un hôte donné vis-à-vis des engagements en qualité de service contractualisés. Si l'hôte hébergeant la machine virtuelle ne dispose pas de suffisamment de ressources pour honorer les engagements en qualité de service, la machine virtuelle est migrée sur un autre hôte pouvant répondre au besoin en ressources,
- Migration pour la consolidation des hôtes : la migration peut également être mise en oeuvre dans un ojectif de rentabilité des plates-formes cloud par l'optimisation des taux d'utilisation des ressources. Dans ce cas, la migration est utilisée pour optimiser le placement des machines virtuelles sur les différents hôtes composant les clusters à des fins de minimisation du nombre d'hôtes associée à une optimisation énergétique [9] et de maximisation des taux d'utilisation des ressources de chacun des hôtes,
- Migration pour l'équilibrage des charges au sein du cluster : l'équilibrage de la charge entre les hôtes composant le cluster vise à optimiser le placement des machines virtuelles 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 [1][10]. Cette approche suppose une connaissance au niveau cluster de la répartition des charges entre les hôtes, pour juger des migrations de machines virtuelles à opérer.

Selon notre connaissance de l'état de l'art, les alogithmes de gestion dynamique de ressources prennent leurs décisions uniquement en fonction des quantités de ressources consommées par les machines virtuelles sans aucune considération de sécurité. Or, les quantités de ressources consommées par les machines virtuelles durant leur exécution sont aisément manipulables et peuvent donc potentiellement fausser les décisions de ces algorithmes si elles sont manipulées à des fins malveillantes. Dans la suite de cet article, nous nous intéressons à démontrer que cette dépendance des décisions des systèmes de gestion dynamique des ressources aux quantités de ressources consommées par les machines virtuelles constitue par nature une vulnérabilité de sécurité de ces systèmes, qu'il convient de considérer.

Pour démontrer l'attaque par migration intempestive de machines virtuelles, nous utilisons l'algorithme commercial VMware DRS qui poursuit prioritairement un objectif d'équilibrage des charges au sein du cluster.

Nous détaillons des éléments de conception de cet algorithme dans la section suivante.

3 Analyse de DRS

L'algorithme DRS [10] [7] adopte une approche globale de niveau cluster pour le placement initial des machines virtuelles au sein du cluster et pour leurs éventuelles migrations automatiques ultérieures garantissant une distribution de charge équilibrée au sein du cluster.

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

- Niveau machine virtuelle : DRS calcule à chaque invocation de l'algorithme la métrique d' 'Engagement Dynamique' (le terme utilisé habituellement est le terme anglais *Dynamic Entitlement* ici employé) pour chaque machine virtuelle en prenant en considération sa demande courante de ressources, ses contraintes de ressources, la consommation des autres machines virtuelles co-localisées et la capacité du cluster ; cette métrique constitue la quantité de ressources à allouer à la machine virtuelle pour la mémoire d'une part et le CPU d'autre part,
- 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 machines virtuelles hébergées par l'hôte rapportée à la capacité du hôte. L'ensemble de ces métriques rapportés 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 n'a lieu. Si *chlsd* excède *thlsd*, DRS procède à une série d'évaluations

de certaines conditions - notamment une estimation du risque et du bénéfice de la capacité des migrations à résorber la contention - pour définir quelle machine virtuelle doit être migrée, le cas échéant. Au total, *thlsd* définit le degré de déséquilibre de charge toléré au sein du cluster.

Trouver le bon compromis entre le degré de tolérance de déséquilibre de charge au sein du cluster et le nombre résultant de migrations nécessaire pour le maintenir, est essentiel. Une tolérance au déséquilibre de charge trop contraignante (*thlsd* faible) génère un nombre possiblement trop élévé 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 paramètrage 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 alors que 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 cet article), nous avons pu collecter certaines valeurs de *thlsd*. Il apparaît que *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* :

$$thlsd = \frac{Agr}{\sqrt{N}} \tag{1}$$

où N est le nombre de hôtes dans le cluster et Agr une constante dépendant du niveau d'agressivité de DRS : respectivement 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').

Figure 1 trace les valeurs de *thlsd* de la formule de calcul (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 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 1 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'aggressivité de DRS que les grands, en matière de valeurs de thlsd.

De cette analyse de DRS, nous retenons que *thlsd* est un paramètre essentiel impactant l'occurence 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 machines virtuelles sous son contrôle, pour amener DRS à éxécuter des migrations de machines virtuelles.

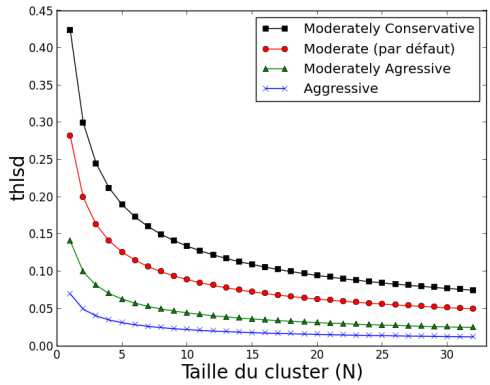


Figure 1. Valeurs de *thlsd* pour les quatre niveaux d'aggressivité de DRS

4 Description de l'attaque

Pour réaliser l'attaque par migration intempestive de machines virtuelles, l'attaquant fait délibérément fluctuer les quantités de ressources consommées par les machines virtuelles 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 la quantité de ressources consommées par ses machines virtuelles, entre des valeurs très hautes et des valeurs très basses. L'élément clef de cette stratégie d'attaque est simple. Quand les machines virtuelles 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 machines virtuelles vers cet hôte : les machines virtuelles sont en quelque sorte "attirées" par cet hôte en sous-charge. Par un raisonnement similaire, on comprend que lorsque les machines malveillantes consomment des quantités importantes de ressources, l'état du hôte les hébergeant est considéré par DRS comme étant surchargé et DRS tend à déclencher des migrations de machines virtuelles pour alléger cette charge : les machines virtuelles 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 machines virtuelles est combiné à un profil à effet d'attraction sur les machines virtuelles car, dans ce cas, l'attaquant peut exécuter le profil à effet de repoussement sur les machines virtuelles alors que l'hôte exécutant les machines virtuelles de l'attaquant est dans un état favorable à l'attaque - i.e. déjà chargé de par une charge préablement 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 machine virtuelle - 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 machines virtuelles 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.

5 Implantation et évaluation

5.1 Dispositif d'expérimentation

Pour démontrer la faisabilité de l'attaque par migration intempestive de machines virtuelles, 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.

Les machines virtuelles sont déployées uniformément sur les hôtes du cluster pour garantir une égalité de contexte d'exécution pour les machines virtuelles, 10 machines vituelles par hôte : 8 machines virtuelles 2 Gb 1 vCPU, 1 machine virtuelle 1GB 1 vCPU et 1 machine virtuelle 512 Mo 1vCPU. Les ressources de chaque hôte sont surengagées 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 intra machine virtuelle, pour la mémoire d'une part et pour le CPU d'autre part, selon un profil paramétrable et possiblement fluctuant dans le temps. La charge est fixée à 75% pour toutes les machines virtuelles (mémoire et CPU), exceptées celles sous le contrôle de l'attaquant que l'on fait varier lors de nos expérimentations. Le second outil permet le suivi de la consommation de ressources des machines virtuelles et des hôtes toutes les 20 secondes, il permet aussi de suivre les statistiques de DRS calculées à chaque invocation de l'algortihme (*Dynamic Entitlement*, *chlsd* et *thlsd*) grâce aux interfaces de programmation fournies par VMware.

Au total, le point de fonctionnement de notre cluster d'expérimentation avec les valeurs de surengagement 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 ressources injectés dans les machines virtuelles.

5.2 Déroulement de l'attaque de base

Dans cette section, nous détaillons l'attaque par attraction de machine virtuelle décrite dans la section 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 sur la figure 1.

Nous illustrons ci-dessous le déroulement de l'attaque avec deux figures. Figure 2 donne la variation de la valeur de *chlsd* durant toute la durée de l'attaque ainsi que la valeur seuil *thlsd*. Figure 3 montre les effets de la manipulation par l'attaquant des quantité de ressources consommées par ses machines virtuelles, sur cette même durée. Plus précisément, figure 3(a)et figure 3(b) indiquent respectivement pour le CPU et pour la mémoire, la valeur de la métrique *Dynamic Entitlement* de la machine virtuelle de l'attaquant telle que calculée par DRS à chaque invocation (toutes les 5 minutes, valeur par défaut). figure 3(c) et figure 3(d) illustrent l'utilisation résultante des ressources CPU et mémoire de l'hôte hébergeant la machine virtuelle attaquante.

Pour éviter tout état initial qui serait favorable au déroulement de l'attaque, l'attaque est lancée dans un état initial de parfait équilibre

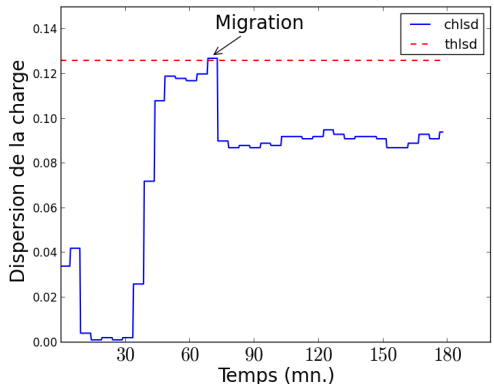
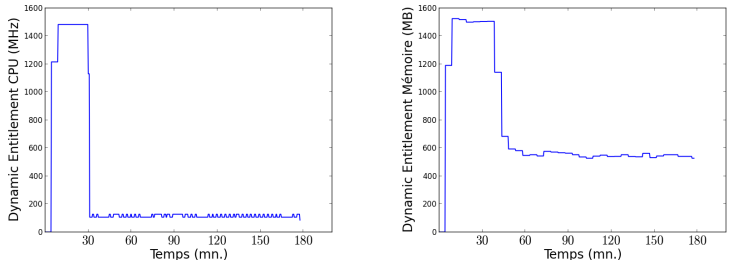


FIGURE 2. Variation de *chlsd* au cours de l'attaque

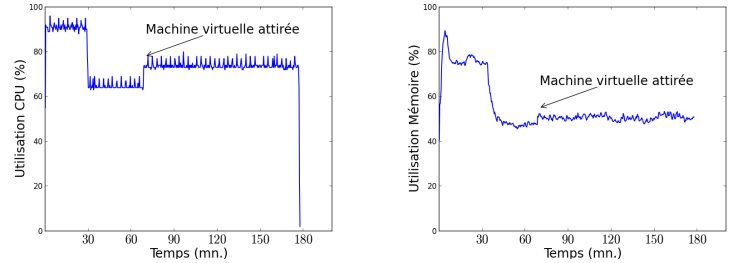
du cluster ($chlsd = 0$) ; cela maximise le déséquilibre de charge qui doit être créé par l'attaque (applicable pour tout l'article). Cet état de parfait équilibre est visible sur la figure 2 au niveau des 30 premières minutes de l'expérimentation : la valeur de chlsd est égale à zéro, obtenue en configurant les niveaux de charge de toutes les machines virtuelles à 75% pour la mémoire et pour le CPU.

L'attaque commence à T = 30 mn quand l'attaquant diminue brutalement la consommation de sa machine virtuelle jusqu'à 5% de sa mémoire et de son CPU. Sur les figure 3(a) et figure 3(b) , nous pouvons observer que cette diminution se répercute immédiatement sur les métriques *Dynamic Entitlement* mémoire et *Dynamic Entitlement* CPU de la machine virtuelle 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. A ce moment, une migration a lieu et nous pouvons observer sur les figures figure 3(c) et figure 3(d) l'augmentation de la consommation de ressources au niveau de l'hôte qui a attiré une machine virtuelle du fait de sa sous charge occasionnée par la diminution de la consommation de la machine attaquante qu'il héberge. On peut voir sur la figure 2 que la migration de machine virtuelle opérée provoque le retour du cluster dans un état 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 machines virtuelles amener le cluster dans un état de déséquilibre de charge et lui faire déclencher une migration de ma-



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



(c) Utilisation CPU hôte (d) Utilisation mémoire hôte

Figure 3. Suivi de l'attaque de base au niveau machine virtuelle et hôte

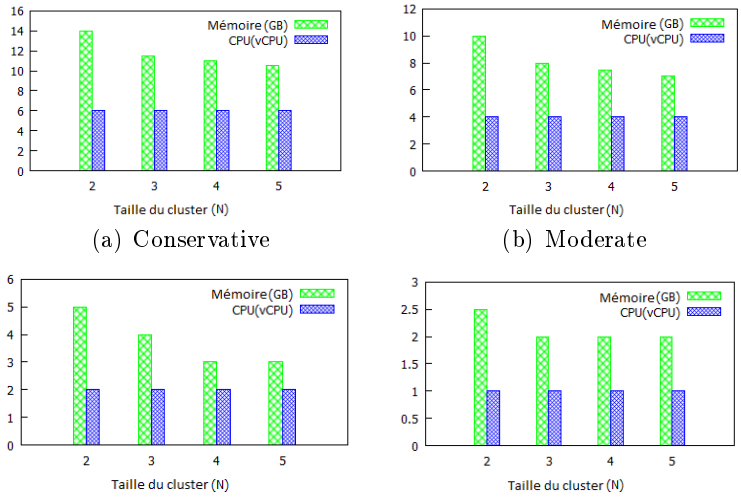
chine virtuelle. Dans la section suivante, nous introduisons une métrique de mesure de la vulnérabillité des clusters vis-à-vis de l'attaque.

5.3 Evaluation de la vulnérabilité du cluster

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 du métrique *Normalized Entitlement* de l'hôte hébergeant ses machines virtuelles, 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 de hôtes (N) composant le cluster et du niveau d'agressivité (Agr) de DRS comme déjà vu avec la formule de calcul (1) de *thlsd*.

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

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 machine virtuelle 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 d'aggressivité 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 machines virtuelles, exceptées ceux des machines virtuelles attaquantes, à une valeur unique et constante de sorte que ces autres paramètres n'aient pas d'influence sur l'interprétation de nos résultats.



(a) Conservative (b) Moderate

(c) Moderately Aggressive (d) Aggressive

Figure 4. Quantité de ressources minimale nécessaire à l'attaque

Figure 4(a), figure 4(b), figure 4(c) et figure 4(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 4 niveaux d'aggressivité de DRS. Pour chaque expérimentation, nous mesurons les quantités de mémoire et de vCPU toutefois, pour une comparaison plus aisée des résultats, le nombre de vCPU est maintenu constant d'une expérimentation à l'autre pour un niveau d'aggressivité 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 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 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 dimension 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 bien quand la taille du cluster augmente.

La deuxième observation que nous pouvons faire grâce à la figure 4 est la 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 'conservative' à 'agressive' (figure 4(a) et figure 4(d)).

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.

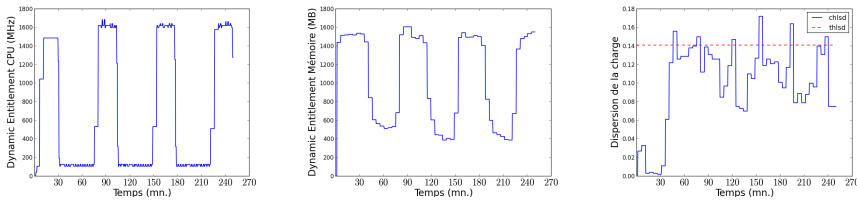
Dans cette section, nous avons analysé l'alogorithme DRS avec une attention particulière pour les valeurs de *thlsd*, pour interpréter nos mesures du niveau de vulnérabilité des clusters vis-à-vis des attaques par migrations intempestives. Dans la section suivante, nous montrons qu'un attaquant peut amener DRS à déclencher une série de migrations successives sous certaines conditions.

5.4 Attaque coordonnée

Dans cette section, nous démontrons qu'un attaquant peut mener 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 machines virtuelles et celle de profils à effet de repoussement de machines virtuelles, tous les deux introduits dans la section 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 machines virtuelles hébergées sur deux hôtes différents. Il s'agit de faire coordonner les machines virtuelles pour que leurs consommations de ressources fluctuent de manière cyclique et en opposition de phase entre le premier et le deuxième hôte : quand les machines virtuelles 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 et versa. De cette manière, l'attaquant forge des opportunités successives de migration amenant DRS à rétablir l'équilibre de charge entre les hôtes successivement en sous-charge (attraction de machines virtuelles) et en sur-charge (repoussement de machines virtuelles).

Pour réaliser cette attaque, nous configurons les générateurs de charge afin que les consommations des machines virtuelles 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 est paramétré selon le niveau d'aggressivité 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* CPU machine virtuelle (b) *Dynamic Entitlement* mémoire machine virtuelle (c) Variation de *chlsd* au cours de l'attaque

Figure 5. Suivi de l'attaque coordonnée au niveau machine virtuelle et cluster

Figure 5(a) et figure 5(b) illustrent la variation de *Dynamic Entitlement* mémoire et CPU d'une des machines virtuelles de l'attaquant, pendant toute la durée de l'attaque. La variation de *chlsd* sur cette même période est donnée figure 5(c).

Le scenario s'exécute avec la même logique que celle décrite précedemment dans la section 5.2, plusieurs fois. Pendant cette expérimentation, nous avons observé six migrations de machine virtuelle. Sur la figure 5, on

peut voir que ces migrations sont bien synchronisées dans le temps avec les fluctuations des profils de consommation des machines virtuelles de l'attaquant.

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

6 Etat de l'art

Les travaux de recherche antérieurs sur la sécurité du partage des ressources en environnement virtualisé, ont démontré que les attaquants peuvent exploiter le manque d'isolation des ressources de l'infrastructure, pour cibler des machines virtuelles co-localisées avec les leurs (ou celles sous leur contrôle) par simple manipulation de leurs propres machines virtuelles ; ces attaques sont connues sous le nom d'attaque cross-machine virtuelle. Selon notre connaissance de l'état de l'art, aucun de ces travaux n'a jamais démontré d'attaque cross-machine virtuelle exploitant les systèmes de gestion dynamique des ressources.

Dans cette section, nous examinons plusieurs types d'attaques cross-machine virtuelle classifiées selon le résultat produit par l'attaque.

- Dans les plate-formes partagées du cloud, les attaques par canaux cachés permettent de provoquer des fuites d'information. Il a par exemple été démontré qu'on peut 1) réaliser des fuites de données grâce à la manipulation de la charge CPU [17], 2) détecter le type d'application s'exécutant dans les machines virtuelles par suivi de leur consommation énergétique [11] ou, 3) vérifier la co-résidence de machines virtuelles attaquantes sur un même hôte par la manipulation de leurs ressources réseau, CPU ou disque [18] [3]. Ce type d'attaque cross-machine virtuelle cible les machines virtuelles co-localisées avec celles de l'attaquant (effet intra-hôte de l'attaque).
- L'attaque par vol de ressources consiste pour une machine virtuelle attaquante à obtenir indûment des ressources aux dépends de ma-

chines virtuelles co-localisées empêchant ainsi ces dernières de bénéfier pleinement de leur propre service de souscription (ou aux dépends du fournisseur de l'infrastructure). Dans l'article [27], la machine virtuelle attaquante contourne la phase de débit de l'ordonanceur de crédit CPU de Xen en s'interrompant délibérément pour entrer en mode sommeil juste avant le passage périodique de l'ordonanceur. Ce type d'attaque cross-machine virtuelle peut avoir pour conséquences de 1) dégrader les performances de machines virtuelles co-localisées, 2) provoquer des famines de ressources ou encore, 3) générer des facturations indues. Les effets cross-machine virtuelle produits par ce type d'attaque visent les seules machines virtuelles partageant un coeur CPU avec les machines virtuelles de l'attaquant (effet intra-hôte de l'attaque),

- Du fait de possibles interférences entre les ressources partagées au sein d'un environnement virtualisé [13], une machine virtuelle peut observer une dégradation de ses performances du fait de l'activité intensive d'une machine virtuelle co-localisée qui serait sous le contrôle d'un attaquant. Les résultats de l'article [2] montrent que les ressources souffrant le plus d'un manque d'isolation sont le réseau et le disque, avec une dégradation de 75% du temps de réponse des applications sensibles à la latence due aux accès disque. Les effets cross-machine virtuelle produits par ce type d'attaque sont également limités aux machines virtuelles partageant la même resource physique que celle de l'attaquant (effet intra-hôte de l'attaque).

L'attaque par migration intempestive de machines virtuelles qui vise à prendre le contrôle sur les décisions du système de gestion dynamique des ressources, diffère de ces attaques préalablement décrites qui exploitent le fait que des machines virtuelles sont en concurrence pour une même ressource. De plus, les attaques par migration intempestive de machines virtuelles peuvent permettre de dégrader les performances des machines virtuelles hébergées sur les hôtes source et destination impliqués dans la migration intempestive. Ces attaques ont donc non seulement un effet cross-machine virtuelle intra-hôte mais aussi un effet cross-machine virtuelle cross-hôte, ce qui augmente la capacité de nuisance de ce type d'attaque.

7 Conclusion

Dans cet article, nous avons présenté en détails l'attaque par migrations intempestives qui est, à notre connaissance, la première attaque rap-

portée contre les systèmes de gestion dynamique des ressources. Nous avons examiné l'algorithme DRS pour nous permettre de comprendre les raisons pour lesquelles cette attaque est réalisable. Enfin, nous avons proposé un moyen de mesurer la vulnérabilité des clusters vis-à-vis de cette attaque.

Nous investiguons actuellement des moyens de mitigations de ce type de menace, soit de manière proactive en agissant sur les politiques de placement des machines virtuelles pour réduire la capacité d'action d'un attaquant en limitant les quantités de ressources de ses machines virtuelles sur un même hôte, soit de manière réactive par détection des profils de consommation de ressources suspects.

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 machines virtuelles. 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.

8 Remerciements

Les auteurs remercient leur collègue d'Orange Labs, Fabien Bignon, pour l'assistance technique fournie pour le développement des outils de génération de charge et de monitoring utilisés dans le cadre des expérimentations rapportées dans cet article.

Références

1. Emmanuel Arzuaga and 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.
2. Sean Kenneth Barker and 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.
3. Adam Bates, Benjamin Mood, Joe Pletcher, Hannah Pruse, Masoud Valafar, and Kevin Butler. Detecting co-residency with active traffic analysis techniques. In *Proceedings of the 2012 ACM Workshop on Cloud computing security workshop*, CCSW '12, pages 1–12, New York, NY, USA, 2012. ACM.

4. Nicolò Maria Calcavecchia, Ofer Biran, Erez Hadad, and Yosef Moatti. Vm placement strategies for cloud scenarios. In *Proceedings of the 2012 IEEE Fifth International Conference on Cloud Computing*, CLOUD '12, pages 852–859, Washington, DC, USA, 2012. IEEE Computer Society.
5. Yanpei Chen, Vern Paxson, and Randy H. Katz. Whats new about cloud computing security. Technical Report UCB/EECS-2010-5, EECS Department, University of California, Berkeley, Jan 2010.
6. Loic Duflot, Olivier Grumelard, Olivier Levillain, and Benjamin Morrin. On the limits of hypervisor- and virtual machine monitor-based isolation. *In Ahmad-Reza Sadeghi and David Naccache, editors, Towards Hardware-Intrinsic Security, Information Security and Cryptography, pages 349366. Springer Berlin Heidelber*, pages 10.1007/978–3–642–14452–316., 2010.
7. Epping Duncan and Frank Denneman. *VMware vSphre 4.1 HA and DRS technical deepdive*. 2010.
8. Nelson Elhage. Virtunoid : A kvm guest ! host privilege escalation exploit. In Black Hat USA, 2011.
9. Tiago C. Ferreto, Marco A. S. Netto, Rodrigo N. Calheiros, and César A. F. De Rose. Server consolidation with migration control for virtualized data centers. *Future Gener. Comput. Syst.*, 27(8) :1027–1034, October 2011.
10. Ajay Gulati, Anne Holler, Minwen Ji, Ganesha Shanmuganathan, Carl Waldspurger, and Xiaoyun Zhu. Vmware distributed resource management : Design, implementation, and lessons learned, 2012.
11. Helmut Hlavacs, Thomas Treutner, Jean-Patrick Gelas, Laurent Lefevre, and Anne-Cecile Orgerie. Energy consumption side-channel attack at virtual machines in a cloud. In *Proceedings of the 2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*, DASC '11, pages 605–612, Washington, DC, USA, 2011. IEEE Computer Society.
12. Kahina Lazri, Sylvie Laniepce, and Jalel Ben-Othman. Reconsidering intrusion monitoring requirements in shared cloud platforms. In *The 2nd International Workshop on Recent Advances in Security Information and Event Management (RaSIEM 2013)*, Sept. 2013.
13. Jeanna Neefe Matthews, Wenjin Hu, Madhujith Hapuarachchi, Todd Deshane, Demetrios Dimatos, Gary Hamilton, Michael McCabe, and James Owens. Quantifying the performance isolation properties of virtualization systems. In *Proceedings of the 2007 workshop on Experimental computer science*, ExpCS '07, New York, NY, USA, 2007. ACM.
14. M. Mishra, A. Das, P. Kulkarni, and A. Sahoo. Dynamic resource management using virtual machine migrations. *Communications Magazine, IEEE*, 50(9) :34–40, 2012.
15. I.S. Moreno, P. Garraghan, P. Townend, and 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.
16. Michael Nelson, Beng-Hong Lim, and 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.

17. Keisuke Okamura and 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.
18. Thomas Ristenpart, Eran Tromer, Hovav Shacham, and 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.
19. A. Singh, M. Korupolu, and 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.
20. Luis M. Vaquero, Luis Rodero-Merino, and Daniel Moreno. Locking the sky : a survey on iaas cloud security. *Computing*, 91(1) :93–118, January 2011.
21. William Voorsluys, James Broberg, Srikumar Venugopal, and 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.
22. Carl A. Waldspurger. Memory resource management in vmware esx server. *SIGOPS Oper. Syst. Rev.*, 36(SI) :181–194, December 2002.
23. Timothy Wood, Prashant Shenoy, Arun Venkataramani, and Mazin Yousif. Black-box and gray-box strategies for virtual machine migration. In *Proceedings of the 4th USENIX conference on Networked systems design ; implementation*, NSDI'07, pages 17–17, Berkeley, CA, USA, 2007. USENIX Association.
24. Yangyang Wu and Ming Zhao. Performance modeling of virtual machine live migration. In *IEEE CLOUD'11*, pages 492–499, 2011.
25. Zhang Xu, Haining Wang, Zichen Xu, and Xiaorui Wang. Power attack : An increasing threat to data centers. In *The Proceedings of the 2014 Network and Distributed System Security Symposium, NDSS'14*, February 2014.
26. Xiangliang Zhang, Zon-Yin Shae, Shuai Zheng, and Hani Jamjoom. Virtual machine migration in an over-committed cloud. In *NOMS'12*, pages 196–203, 2012.
27. Fangfei Zhou, M. Goel, P. Desnoyers, and R. Sundaram. Scheduler vulnerabilities and coordinated attacks in cloud computing. In *Network Computing and Applications (NCA), 2011 10th IEEE International Symposium on*, pages 123 –130, aug. 2011.