**MajecSTIC 2012**
**Lille, France, du 29 au 31 octobre 2012**

# Étude de l'impact des attaques distribuées et multi-chemins sur les solutions de sécurité réseaux

Damien Riquet, Gilles Grimaud et Michaël Hauspie

Université Lille 1, LIFL, CNRS UMR 8022, Cité Scientifique Bâtiment M3, 59655 Villeneuve d'Ascq Cédex, France

Contact : {damien.riquet,gilles.grimaud,michael.hauspie}@lifl.fr

---

**Résumé**
Le Cloud Computing est un modèle récent qui tend à résoudre les situations qui nécessitent notamment beaucoup de ressources. Bien que ce modèle soit populaire, l'aspect sécuritaire du Cloud Computing peut retarder son adoption massive. En effet, ce type d'architecture se doit de proposer un service fiable en ce qui concerne la disponibilité et la confidentialité des données. Ce type de structure est vraisemblablement sujet à des problèmes de sécurité s'il est confronté à des attaques distribuées.
Dans ce papier, nous nous intéressons aux attaques distribuées dirigées vers ou depuis ce type d'architecture. Pour cela, nous avons étudié les solutions de sécurité qui sont mises en place dans ce type de structure et nous les avons confrontées à ces attaques. Plus particulièrement, nos travaux se sont tournés vers le balayage de ports distribué, une partie préliminaire des attaques à proprement parler. Nos travaux se sont focalisés sur l'aspect multi-chemins du Cloud Computing : plusieurs chemins existent vers une machine et les attaquants exploitent cet aspect pour ne pas être détectés. Nos résultats montrent que les solutions de sécurité testées ne sont pas adaptées pour le Cloud Computing. En effet, dans un tel environnement, plus le nombre de chemins possibles est grand, plus l'attaque est efficace.

**Mots-clés :** Sécurité, Cloud Computing, Attaques distribuées, Multi-chemins, Scan de ports.

## 1. Introduction

Le Cloud Computing est actuellement un modèle populaire pour traiter de larges volumes de données. Ce modèle fournit différentes couches de services selon les besoins des utilisateurs. La plupart d'entre eux utilise le Cloud Computing par exemple pour stocker des données confidentielles. C'est pourquoi il est nécessaire que le Cloud Computing propose un système de sécurité fiable.
La sécurité d'un réseau est généralement assurée par des composants tels que les pare-feux ou les Systèmes de Détection d'Intrusions (*Intrusion Detection Systems* ou *IDS* en anglais). Ces composants complémentaires peuvent détecter les intrusions et bloquer les attaques depuis ou ciblant le Cloud Computing. Malheureusement, les éléments de sécurité les plus utilisés ne sont pas conçus pour détecter les attaques distribuées. Ces attaques sont menées en subdivisant une attaque, de manière à ce que les éléments de sécurité ne soient pas capables de détecter les sous-parties de l'attaque. De plus, l'attaquant peut utiliser la structure du Cloud Computing pour être davantage discret. En effet, si un attaquant peut emprunter plusieurs chemins pour s'adresser à une machine, sa discretion est accrue puisque chaque solution de sécurité ne voit qu'une partie du trafic.
Dans cet article, nous nous intéressons aux attaques dirigées vers ou depuis le Cloud Computing. Plus particulièrement, notre étude se concentre sur l'aspect multi-chemins de ce modèle.
Pour cela, nous avons mené des attaques distribuées (balayage de ports) sur des architectures de type Cloud Computing. Notre étude s'est focalisée sur l'IDS Snort, une solution de sécurité Open source populaire. Ces expérimentations, réalisées dans des environnements différents (différents nombres d'attaquants, de cibles, d'élements de sécurité, de type de balayage de ports, etc.) indiquent

que les solutions de sécurité réseaux ne sont plus adaptées aux nouveaux types d'attaques. En effet, dans certains cas, seuls 32 attaquants sont suffisants pour mener une attaque en toute discretion. De nos jours, de telles ressources sont accessibles facilement, grâce à des botnets, après une infection virale de type ver informatique, ou tout simplement en louant des ressources auprès des fournisseurs de Cloud Computing. C'est pourquoi nous nous intéressons à ces attaques, qui peuvent retarder l'adoption massive du Cloud Computing.
L'article est structuré de la manière suivante. Dans un premier temps, la Section 2 introduit le domaine de la sécurité du Cloud Computing. Ensuite, la Section 3 présente le balayage de ports distribué. Les Sections 4 et 5 présentent respectivement le protocole expérimental et les résultats des expérimentations. Enfin, la Section 6 conclut cet article et présente les prochains axes de travail.

## 2. La sécurité dans le Cloud Computing : Motivation et enjeux

Dernièrement, industriels et chercheurs ont rencontré une difficulté commune : le traitement de larges volumes de données. Le Cloud Computing semble être la solution à ce problème.
Rimal *et al.* présentent le Cloud Computing dans [1] comme étant le concept de la prochaine évolution des systèmes distribués. Le Cloud Computing peut être caractérisé selon un certain nombre de paramètres, parmi lesquels la gestion de la virtualisation, la tolérance aux fautes, la répartition de la charge et également la sécurité. D'autre part, Rimal *et al.* indiquent dans leur article que les utilisateurs stockent des données confidentielles dans ces structures. Dans de mauvaises mains, ces données peuvent engendrer de graves situations.
Les éléments principaux de la sécurité d'une telle architecture sont les pare-feux et les IDS. Bellovin *et al.* décrivent dans [2] les pare-feux comme étant des composants placés entre deux réseaux. Tout le trafic entre ces deux réseaux doit passer par le pare-feu qui pourra ainsi filtrer ce qui est interdit, défini par une politique de sécurité locale. Dans [3], Debar *et al.* soulignent que la tache principale d'un IDS est de surveiller l'utilisation des systèmes et de détecter les états insécurisés. Ces derniers détectent les tentatives d'intrusion et abus volontaires d'utilisateurs légitimes qui tentent d'exploiter des failles ou de passer outre leurs privilèges.
Bien que la sécurité d'un réseau se base sur ces éléments, il ne suffit pas de les placer plus ou moins aléatoirement pour que le réseau soit sécurisé. La sécurité du Cloud Computing doit être adaptable de part le fait que la structure est dynamique (les utilisateurs peuvent changer durant une attaque). En outre, les utilisateurs du Cloud peuvent être malicieux et vouloir en compromettre la structure. C'est pourquoi le Cloud Computing doit garantir la disponibilité et l'intégrité des données en son sein. Pour finir, il est nécessaire d'empêcher les attaques distribuées telles que les DDOS ou les vers informatique, même à l'intérieur du Cloud.

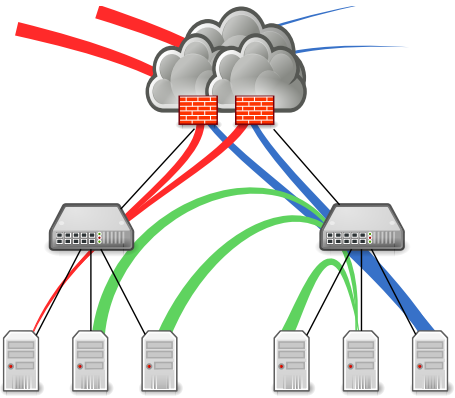


FIGURE 1 – Attaques distribuées dirigées vers ou depuis le Cloud Computing

De part leur position (en bordure du réseau), les pare-feux ne seraient pas capables de détecter ces dernières attaques, puisque le trafic ne transiterait pas par eux. Toutefois, des IDS bien placés au

sein du réseau semblent pouvoir détecter ce type d'attaque, c'est pourquoi nous nous intéressons à ces solutions de sécurité. La Figure 1 représente les attaques possibles ciblant le Cloud ou l'utilisant. Dans un premier temps, une attaque distribuée (en rouge) peut survenir depuis l'extérieur et cibler une machine au sein du Cloud. Inversement, un attaquant (en bleu) peut utiliser plusieurs machines du Cloud pour attaquer une ou plusieurs machines à l'extérieur. Enfin, une attaque interne, qui ne transite pas par les pare-feux est représentée en vert.
Les prochaines sous-sections présentent la mécanique interne d'un IDS, ce qu'est une attaque distribuée et les travaux récents menés dans ce domaine.

### 2.1. Systèmes de Détection d'Intrusions

Dans le domaine de la sécurité d'un réseau, on peut différencier différentes étapes : la prévention, la détection et la correction. Les Systèmes de Détection d'Intrusions (*Intrusion Detection System* ou IDS en Anglais) ont pour principale tâche de détecter les intrusions.
Dans [4], Peddisetty résume les définitions proposées dans [5] et [6]. Il décrit alors la détection d'intrusion comme étant la surveillance d'événements et leur analyse pour trouver des signes d'intrusions ou de tentatives de compromission de la confidentialité, de l'intégrité ou de la disponibilité d'un système. Peddisetty classifie les IDS selon un certain nombre de caractéristiques, comme par exemple leur emplacement, leur source d'événements, leurs méthodes de détection, etc. Nous décrivons à la suite de cette sous-partie plusieurs de ces éléments.

#### Source des événements

Les événements traités par les IDS peuvent provenir de différentes sources. Communément, les IDS font partie d'un réseau et analysent le trafic qui transite par celui-ci. On les catégorise alors de Système de Détection d'Intrusions Réseau (*Network-based IDS* ou NIDS). Il existe également les IDS qui surveillent les informations collectées au sein d'une machine. Ceux-ci analysent les journaux systèmes et surveillent les fichiers critiques pour détecter les intrusions. Ces IDS sont alors catégorisés comme IDS Machine (*Host-based IDS* ou HIDS). D'autre part, il existe des IDS hybrides, qui utilisent ces deux sources d'événements, ou même d'autres IDS.

#### Architecture

Dernièrement, les systèmes distribués se répandent et cela s'applique également aux IDS. Des IDS distribués (aussi appelés DIDS pour *Distributed IDS*) ont été proposés dans [5] [6]. Ces DIDS traitent les événements de l'ensemble des éléments de sécurité, placés à différents points du système étudié. Les solutions adoptent des architectures différentes en ce qui concerne l'analyse des événements collectés à plusieurs endroits. Les premières solutions se satisfaisaient d'une analyse centralisée, c'est à dire qu'un seul IDS analysait les évènements. Cela a pour principal désavantage que si le noeud d'analyse s'arrête, l'ensemble du système est paralysé. C'est pourquoi les architectures décentralisées ont été proposées : chaque noeud recueille et analyse les événements locaux et les alertes levées sont partagées dans l'architecture. L'ensemble des noeuds du système forme alors un Système de Détection d'Intrusions Collaboratif (*Collaborative-IDS* ou CIDS).

#### Méthodes de détection

Pour détecter les intrusions, les IDS utilisent des méthodes de détection. Ces dernières analysent les événements et lèvent des alarmes une fois qu'elles détectent une intrusion. Dans [4], Peddisetty présente les stratégies de détection les plus utilisées : la détection d'anomalie et la détection d'abus. La détection d'anomalie est basée sur la recherche de comportement anormal. Tout ce qui dévie de la normalité lève une alarme. Ce type de détection est efficace sur des attaques inconnues mais peut provoquer un grand nombre de faux-positifs. La détection d'abus se base sur une connaissance a priori des intrusions possibles sur le système. Cette méthode consiste à comparer chaque événement à cette connaissance et si une similarité est trouvée, une alarme est levée. Ce type de détection a pour principal désavantage d'être limité à la connaissance du système. Les nouvelles attaques ne seraient donc pas détectées car elles ne seraient pas présentes dans la base de données des motifs connus comme étant malveillants.
Il faut ajouter à ces deux méthodes la corrélation d'événements présentée dans [7], utilisée par les CIDS. Ces systèmes utilisent l'ensemble d'un réseau pour corréler des événements et en déduire

qu'une attaque se produit à plusieurs endroits du système. Ce type de détection d'intrusions s'adresse particulièrement aux attaques distribuées.

### 2.2. Attaques distribuées

Dans cet article, nous nous sommes focalisés sur les attaques distribuées, telles que le balayage de ports distribué ou les dénis de services distribués. Ces attaques sont difficiles à détecter, selon Zhou *et al.* [7], parce que les IDS ne surveillent que des petites portions du système et non sa globalité. Ils présentent alors les solutions collaboratives qui visent à résoudre ce problème.
Notre travail consiste à vérifier que les solutions de sécurité communes ne sont effectivement pas adaptées face à ce nouveau type d'attaque. Pour cela, nous avons dans un premier temps étudié les attaques distribuées. Dans l'article [8], nous avons expérimenté le balayage de ports distribué sur une structure assimilable au Cloud Computing avec une seule route. Les résultats montraient qu'un attaquant souhaitant mener une attaque avait besoin de seulement 32 machines pour ne pas être détecté, pour la plupart des cas de figure.
Dans cet article, nous nous intéressons aux cas où l'attaque était détectée et nous montrons qu'ils peuvent être améliorés. Particulièrement, l'aspect multi-chemins du Cloud Computing peut permettre à l'attaquant d'utiliser moins de machines pour mener son attaque sans être détecté. La prochaine section présente le balayage de ports distribué, utilisé lors de nos expérimentations.

## 3. Scan de ports distribué

Nous avons choisi d'expérimenter les attaques distribuées en utilisant le balayage de ports. Fyodor, contributeur initial de l'outil d'audit de sécurité Nmap, considère dans [9] que le balayage de ports permet aux attaquants de découvrir des canaux de communication exploitables, en vue d'une attaque à proprement parler. Dans cette section, nous présentons tout d'abord ce qu'est un balayage de ports, aussi appelé scan de ports. Ensuite, nous expliquons comment détecter cette attaque et comment les attaquants peuvent éviter d'être détectés. Pour finir, nous présentons comment distribuer cette attaque.

### 3.1. Scan de ports

M. Roesch, auteur de l'IDS Open source Snort [10], décrit le scan de ports comme une phase de reconnaissance. Durant cette phase, un attaquant détermine quels types de protocoles réseaux ou services une machine propose. Le scan de ports consiste en un échange de paquets réseaux. En fonction de l'échange, l'attaquant peut conclure l'état du port : ouvert, fermé ou filtré.

### 3.2. Détection d'un scan de ports

Les solutions de sécurité ont plusieurs méthodes pour détecter les scans de ports. Il faut différencier les différentes techniques de scan de ports utilisées. Certaines imitent un trafic légitime (*i.e.* ce trafic peut se produire sur un réseau) alors que d'autres non. Ces techniques sont expliquées dans [9]. Dans notre article, nous nous intéressons aux techniques Connect et SYN, qui tentent d'établir une connexion TCP, ce qui est légitime. La technique SYN ne réalise pas toutes les étapes de connexion contrairement à la technique Connect.
Pour les techniques de scan de ports qui produisent un trafic illégitime, il faut appliquer une détection d'abus. Typiquement, on peut caractériser le trafic engendré dans une base de données d'attaques connues. Ceci implique que l'attaque est détectée à partir du premier paquet réseau.
Pour les autres techniques, il faut appliquer une heuristique, qui fait appel à des statistiques. Pour cela, les IDS utilisent communément un compteur, qui est incrémenté à chaque fois qu'un événement suspect se produit. Une fois que le compteur dépasse un seuil fixé, une alerte est levée. Afin de ne pas générer de faux-positifs, il faut mettre en place une politique d'amnistie pour décrémenter ou réinitialiser ces compteurs. Kang *et al.* proposent de telles politiques dans [11]. La première, *Positive-Reward-based method*, consiste à récompenser le bon comportement des utilisateurs. Par exemple, un bon comportement peut se traduire par une connexion réussie vers une autre machine. La seconde, *Timeout-based method*, attribue une durée de vie à chaque événement. Lorsqu'un événement expire, on décrémente le compteur.

### 3.3. Évasion des IDS

L'objectif principal d'un attaquant est de passer inaperçu lors du scan de ports, ainsi il peut enchaîner sur une autre attaque. Pour ne pas être détecté, un attaquant doit tout d'abord adopter une technique de scan de ports qui produit un trafic légitime. La plupart des IDS utilisent des compteurs, c'est pourquoi il doit chercher à les décrémenter. Pour cela, il peut se connecter à des machines dont il connait l'état des ports et ainsi cibler les ports ouverts. Ou bien, il peut tout simplement ralentir son attaque, de manière à ce que les événements expirent avant la détection de l'attaque.
Pour finir, nous avons montré dans [8] qu'il était simple de ne pas se faire détecter grâce à la distribution de l'attaque. En effet, même pour une attaque rapide, 32 machines sont suffisantes pour la plupart des cas de figure testés.

### 3.4. Distribution du scan de ports

La distribution d'une attaque consiste à subdiviser l'attaque en tâches, qui seront alors réalisées par différentes machines. De nombreuses manières de distribuer une attaque existent, nous présentons ici les deux manières que nous avons utilisées pour les expérimentations. Pour ces méthodes de distribution, l'attaquant possède un ensemble de machines, appelées scanners, qu'il peut utiliser pour mener son attaque. Les expérimentations menées pour cet article suppose qu'un attaquant sait quand un scanner a été détecté. Cela n'est bien entendu pas le cas pour de vraies attaques, mais cela nous permet de calculer des taux de réussite, dans la Section 5.
Kang *et al* présentent dans [11] une distribution naïve. Cette distribution utilise séquentiellement les scanners ; elle sélectionne un scanner et commence l'attaque avec celui-ci. Une fois qu'il est détecté, elle selectionne alors un autre scanner (encore non détecté) et poursuit ainsi jusqu'à la fin de l'attaque ou bien lorsqu'il n'y a plus de scanners à disposition.
La distribution en boucle se base sur la précédente. Toutefois, on change de scanner à chaque fois qu'une tâche est réalisée : on boucle sur l'ensemble des scanners disponibles. Cette méthode, pourtant naïve, permet de ne pas être détecté lorsque deux utilisations d'un même scanner sont assez espacées dans le temps.

## 4. Protocole expérimental

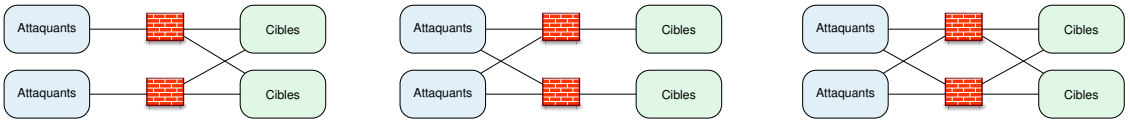
Cette section présente le protocole utilisé pour les expérimentations. Dans un premier temps, nous présentons Snort, la solution de sécurité testée. Ensuite, nous présentons les architectures réseaux et les différentes variables utilisées pour ces expérimentations.

### 4.1. Solution de sécurité

Nous avons choisi d'expérimenter la solution de sécurité Snort, un IDS Open source. Selon [10], Snort (version 2.9.1) utilise une base de données contenant les attaques connues. De cette manière, il est capable d'analyser le trafic en temps-réel et de le comparer à ces dernières. En ce qui concerne le scan de ports, Snort peut le détecter de deux manières. Le trafic illégitime est détecté très rapidement car (normalement) présent dans la base de données. D'autre part, Snort utilise le module sfPortscan, qui analyse statistiquement les événements du réseau. Ce module adopte une politique d'amnistie basée sur la durée de vie des événements.

### 4.2. Architecture réseau

Dans cet article, nous nous intéressons aux architectures multi-chemins. C'est à dire qu'il existe plusieurs chemins pour intéragir avec les acteurs de l'attaque, que ce soit un attaquant ou une cible. La Figure 2 représente les différents cas de figure que nous étudions, pour un environnement à deux routes. La topologie T1 (Figure 2a) correspond par exemple à une isolation géographique des attaquants ; ces derniers transitent par un seul point d'entrée pour s'adresser à l'ensemble des cibles. À l'inverse, la topologie T2 (Figure 2b) correspond à une isolation des cibles. Pour finir, la topologie T3 (Figure 2c) correspond à une interconnexion totale : un attaquant peut utiliser plusieurs chemins pour s'adresser à une cible.




(a) Topologie T1 (b) Topologie T2 (c) Topologie T3

FIGURE 2 – Différentes topologies étudiées pour un environnement à deux routes

### 4.3. Variables des configurations

Nous avons effectué les expérimentations selon un certain nombre de variables. Voici ces variables et leurs valeurs :

– Nombre de machines attaquantes : 4, 8, 16, 32 ;
– Nombre de machines ciblées : 64 ;
– Topologies : T1, T2 et T3 ;
– Distribution du scan de ports : naïve et en boucle ;
– Technique de scan de ports : Connect et SYN scanning ;
– Ports ciblés : 100 (parmi lesquels 10 sont ouverts) ;
– Temps entre chaque scan de ports : 50, 250 et 1000 ms ;
– Nombre de routes : 1, 2, 4 et 8.

Dans [8], nous avons déterminé qu'une attaque utilisant au moins 32 scanners (pour une seule route) et qui était assez lente avait peu de chance de se faire détecter. Dans cet article, nous cherchons à améliorer les cas de figure où la solution de sécurité détectait l'attaque. C'est pourquoi nous nous limitons à des attaques rapides qui génèrent un trafic légitime. En ce qui concerne les configurations avec plusieurs routes, le choix de la route se fait de manière aléatoire.

## 5. Résultats

### 5.1. Évaluation

Pour évaluer les différentes expérimentations, nous utilisons une métrique appelée Taux de Réussité de l'Attaquant (*Attacker Success Rate* ou ASR dans [8]). Les expérimentations sont alors évaluées de la manière suivante : $ASR = \frac{n}{T}$, avec $n$ correspondant au nombre de ports scannés avec succès avant détection et $T$ au nombre total de ports à scanner. On considère un scan de ports réalisé avec succès lorsque le scanner utilisé n'a pas été détecté, que l'état du port trouvé est correct et que le trafic généré parvient à la cible (certains pare-feux peuvent filtrer des paquets).

Nous nous intéressons dans cet article au bénéfice qu'aura un attaquant de mener une attaque distribuée sur une structure multi-chemins. L'évaluation du bénéfice (ou du gain) réalisé par un attaquant pour une structure $m$-chemins est calculé ainsi : $Gain_m = \frac{ASR_m}{ASR_1}$. Les résultats sont présentés dans la prochaine sous-section.

### 5.2. Résultats

Nous abordons dans cette section les résultat importants des expérimentations ; l'ensemble des sources et des résultats est disponible sur demande [1].

Le Tableau 1 présente les gains de manière décroissante. La première chose à remarquer est que le gain d'une attaque peut atteindre jusqu'à 6, ce qui n'est pas linéaire mais s'en rapproche. D'autre part, il faut aussi noter le fait que le gain le moins élevé correspond en fait à un cas de figure où l'attaque n'est pas détectée (ASR égal à 1), ce qui ne peut être amélioré.

Parmi les meilleurs gains, on peut constater la prépondérance de la distribution en boucle, qui logiquement réalise de meilleurs résultats. En effet, puisque Snort associe une durée de vie à chaque événement, l'attaquant doit chercher à faire expirer les événements qu'il a produit. Dans notre cas, la distribution en boucle est plus efficace car deux utilisations consécutives d'un même scanner est plus espacé.

On peut également remarquer que la technique de scan de ports n'a pas d'incidence majeure sur

1. Pour retrouver les sources et les résultats, rendez-vous sur la page http ://lifl.fr/~riquetd ou bien envoyez un mail à damien.riquet@lifl.fr.

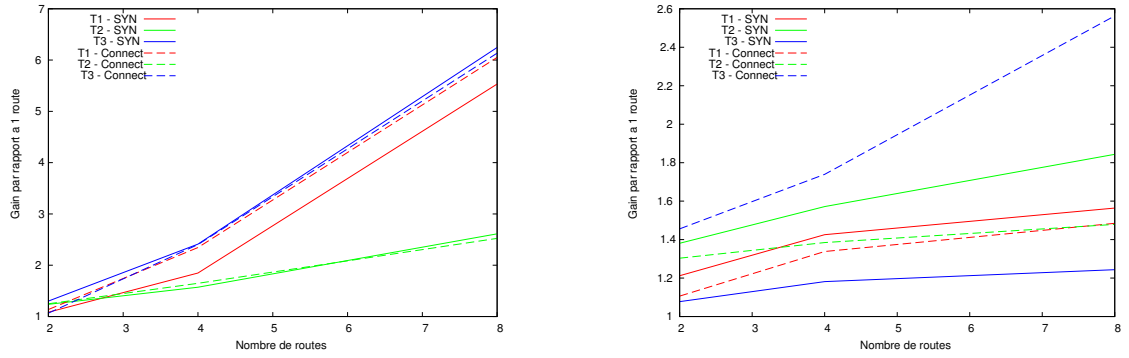
| Distribution | Topologie | Technique | Délai (ms) | Nb scanners | Nb routes (= p) | $ASR_1$ | $ASR_p$ | $Gain_p$ |
|---|---|---|---|---|---|---|---|---|
| Boucle | T3 | SYN | 250 | 32 | 8 | 0.05 | 0.33 | 6.25 |
| Boucle | T3 | Connect | 250 | 32 | 8 | 0.04 | 0.26 | 6.13 |
| Boucle | T1 | Connect | 250 | 32 | 8 | 0.04 | 0.26 | 6.05 |
| Boucle | T1 | SYN | 250 | 32 | 8 | 0.05 | 0.29 | 5.53 |
| Naïve | T2 | Connect | 1000 | 16 | 8 | 0.02 | 0.06 | 2.83 |
| Naïve | T3 | SYN | 1000 | 32 | 8 | 0.05 | 0.12 | 2.69 |
| Boucle | T2 | SYN | 250 | 32 | 8 | 0.03 | 0.09 | 2.61 |
| Naïve | T3 | Connect | 250 | 32 | 8 | 0.05 | 0.11 | 2.56 |
| Boucle | T2 | Connect | 250 | 32 | 8 | 0.03 | 0.09 | 2.52 |
| Naïve | T3 | Connect | 1000 | 32 | 8 | 0.05 | 0.11 | 2.43 |
| Boucle | T3 | SYN | 250 | 32 | 4 | 0.05 | 0.13 | 2.41 |
| Boucle | T3 | Connect | 250 | 32 | 4 | 0.04 | 0.10 | 2.40 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| Boucle | T3 | SYN | 1000 | 32 | 2 | 1.00 | 1.00 | 1.00 |

TABLE 1 – Gain des scans de port distribuées classés dans l'ordre décroissant

les résultats. Snort ne fait pas de distinction entre les deux, bien que la technique Connect scanning génère un peu plus d'événements (Connect scanning établie une connexion complète alors que la version SYN se satisfait de la moitié dans le cas d'un port ouvert).

En ce qui concerne la topologie du réseau, une interconnexion totale (T3) est favorable aux attaquants, quelque soit la méthode de distribution, comme le montre la Figure 3. Ces courbes représentent le gain d'un scan de ports distribué par rapport au nombre de routes selon la méthode de distribution, le nombre d'attaquants et la topologie.

Pour la majorité des cas, la topologie T2 est moins efficace car chaque IDS voit l'intégralité du trafic dirigé vers une cible donnée de son sous-réseau, ce qui n'est pas le cas pour les autres topologies. Effectivement, un scan de ports vers une cible particulière transitant par différents points d'entrées est propice à l'attaquant, car Snort associe un compteur à chaque couple (scanner, cible). Ainsi, maximiser le nombre de couples permet de ne pas incrémenter un seul compteur et donc échapper à la détection. D'autre part, dans le cas de la topologie T3 et de la méthode de distribution naïve, on remarque une nette amélioration. Dans ce cas de figure, la distribution naïve peut s'apparenter à une distribution en boucle, car deux utilisations consécutives d'un même scanner va emprunter plusieurs routes, ce qui génère donc moins d'événements sur chaque point d'entrée.




(a) Distribution en boucle (32 Attaquants, 250ms)

(b) Distribution naïve (32 Attaquants, 250ms)

FIGURE 3 – Amélioration de l'efficacité des attaques grâce aux multi-chemins

## 6. Conclusion

Le Cloud Computing est une architecture pleine de ressources. Elle propose plusieurs couches de services en fonction des besoins des utilisateurs. Actuellement, seuls les problèmes de sécurité retardent son adoption massive. Dans ce papier, nous avions étudié les attaques distribuées dirigées depuis ou vers ce type d'architecture. Plus particulièrement, nous avons analysé le déroulement d'une attaque distribuée lorsqu'il y a plusieurs chemins pour s'adresser à une machine.
Dans de précédents travaux, nous avons montré qu'il était possible de mener une attaque avec 32 machines sans se faire détecter, dans un environnement avec un seul chemin. Dans cet article, nous avons étudié l'impact qu'aurait une structure ayant plusieurs chemins, ce qui se rapproche des structures proposées sur le marché. Les résultats indiquent que plus il y a de chemins, plus l'attaque sera efficace. En effet, l'efficacité d'une attaque entre une structure ne proposant qu'un seul chemin et une autre en proposant huit peut être multipliée jusqu'à six. L'environnement d'expérimentations n'incluant pas de bruit réseau, on peut imaginer que les résultats pourraient être pires pour les solutions de sécurité dans le cas de vraies attaques.
C'est pourquoi il nous semble primordial de proposer des solutions de sécurité distribuées qui collaborent, pour parer à ce type d'attaque. Nos prochains travaux s'orienteront selon cet axe de recherche. Particulièrement, nous souhaitons intégrer des sondes physiques (utilisant la technologie FPGA) et virtuelles, qui intéragissent ensemble pour détecter les attaques.

## Remerciements

Les expérimentations présentées dans cet article ont été réalisées en utilisant la plateforme expérimentale Grid'5000, développée dans le cadre du projet INRIA ALADDIN avec pour support le CNRS, RENATER et plusieurs universités ainsi que d'autres entités (voir https ://www.grid5000.fr).

## Bibliographie

1. B. Rimal, E. Choi, and I. Lumb, "A taxonomy and survey of cloud computing systems," in *INC, IMS and IDC, 2009. NCM '09. Fifth International Joint Conference on*, pp. 44 –51, aug. 2009.
2. S. Bellovin and W. Cheswick, "Network firewalls," *Communications Magazine, IEEE*, vol. 32, pp. 50 –57, sept. 1994.
3. H. Debar, M. Dacier, and A. Wespi, "Towards a taxonomy of intrusion-detection systems," *Computer Networks*, vol. 31, no. 8, pp. 805 – 822, 1999.
4. Peddisetty, "State-of-the-art intrusion detection : Technology, challenges, and evaluation.," 2005.
5. S. R. Snapp, S. R. Snapp, J. Brentano, G. V. Dias, T. L. Goan, L. T. Heberlein, C.-l. Ho, K. N. Levitt, B. Mukherjee, S. E. Smaha, T. Grance, D. M. Teal, and D. Mansur, "Dids (distributed intrusion detection system) - motivation, architecture, and an early prototype," *In Proceedings of the 14th National Computer Security Conference*, pp. 167–176, 1991.
6. J. Balasubramaniyan, J. Garcia-Fernandez, D. Isacoff, E. Spafford, and D. Zamboni, "An architecture for intrusion detection using autonomous agents," in *Computer Security Applications Conference, 1998, Proceedings., 14th Annual*, pp. 13 –24, dec 1998.
7. C. V. Zhou, C. Leckie, and S. Karunasekera, "A survey of coordinated attacks and collaborative intrusion detection," *Computers and Security*, vol. 29, no. 1, pp. 124 – 140, 2010.
8. D. Riquet, G. Grimaud, and M. Hauspie, "Large-scale coordinated attacks : Impact on the cloud security," *Workshop MCNCS, International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS-2012)*, July 2012.
9. G. F. Lyon, *Nmap Network Scanning : The Official Nmap Project Guide to Network Discovery and Security Scanning*. USA : Insecure, 2009.
10. M. Roesch, "Snort - lightweight intrusion detection for networks," in *Proceedings of the 13th USENIX conference on System administration*, LISA '99, (Berkeley, CA, USA), pp. 229–238, USENIX Association, 1999.
11. M. G. Kang, J. Caballero, and D. Song, "Distributed evasive scan techniques and countermeasures," in *Proceedings of the 4th international conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, DIMVA '07, (Berlin, Heidelberg), pp. 157–174, Springer-Verlag, 2007.