Une architecture de honeypots distribués pour superviser le réseau P2P KAD

Thibault Cholez, Isabelle Chrisment, Olivier Festor

Une architecture de honeypots distribués pour superviser le réseau P2P KAD

Thibault Cholez
MADYNES - INRIA Nancy Grand Est, France
thibault.cholez@loria.fr

Isabelle Chrisment
MADYNES - Nancy Université, France
isabelle.chrisment@loria.fr

Olivier Festor
MADYNES - INRIA Nancy Grand Est, France
olivier.festor@loria.fr

ABSTRACT

Superviser efficacement un grand réseau pair-à-pair entièrement distribué est très complexe. Nous proposons, dans ce papier, une nouvelle architecture de honeypots distribués nous permettant d'étudier et de lutter efficacement contre des contenus malveillants dans le cadre du réseau pair-à-pair KAD. Après avoir évalué les nouveaux mécanismes de protection contre l'attaque Sybil insérés dans KAD, nous montrons comment notre architecture les contourne et nous donne ainsi les moyens de prendre localement le contrôle de la DHT par une attaque ciblée et distribuée, de superviser et d'agir sur des contenus spécifiques (mots clés et fichiers) au sein du réseau P2P. Les premiers déploiements réalisés sur le réseau KAD montrent la validité de notre approche avec peu de ressources.

Keywords

Réseaux P2P, DHT, Attaque Sybil, Honeypot, KAD

1. INTRODUCTION

Les réseaux pair-à-pair (P2P) sont devenus une application majeure de l'Internet en permettant à des millions d'utilisateurs de partager rapidement et sans coût d'infrastructure de grandes quantités de fichiers. Cependant, les réseaux P2P peuvent également être un support pour des activités malveillantes en propageant des données indésirables (virus, contenus liés à la cybercriminalité, ...). Les réseaux P2P étant entièrement dynamiques et distribués, il est très difficile de superviser et d'agir sur ce type de contenu au sein d'un grand réseau P2P tel que KAD. Nous proposons dans ce papier une architecture de honeypots distribués (Honeynet) pour le réseau KAD, capable de superviser et d'agir sur les contenus malveillants et d'observer le comportement des pairs, malgré les dernières protections visant à limiter les possibilités d'action sur la DHT.

Notre approche est plus performante que les solutions habituellement utilisées pour superviser les réseaux P2P. Les crawlers, tels que pratiqués sur Gnutella [8] ou KAD [9] [6], interrogent de manière active l'ensemble des pairs d'un réseau pour obtenir des informations et, dans le cas de KAD, ne permettent pas d'obtenir de connaissances sur les contenus indexés. Coupler un crawler à une attaque Sybil, qui consiste à créer un nombre important de faux pairs, permet de dépasser cette limitation [7] mais cette méthode est désormais fortement limitée par des mécanismes de protection [1]. Les honeypots [3] [4] sont une autre manière de collecter des informations sur les réseaux P2P. Ceux-ci attirent les pairs malveillants en annonçant de faux fichiers et enregistrent les demandes de téléchargement reçues, sans garantie, toutefois, de pouvoir attirer la totalité des pairs recherchant le contenu étudié.

L'architecture de Honeynet que nous proposons permet de superviser de manière passive toutes les requêtes destinées à un contenu spécifique (mot clé ou fichier), mais également d'agir sur ce dernier tout en étant très peu intrusif pour le réseau KAD. Nous pouvons ainsi éclipser ou modifier l'indexation des références malveillantes sur la DHT. Notre architecture contrôlant la totalité des requêtes émises pour le contenu étudié, nous pouvons annoncer de faux fichiers extrêmement attractifs tout en éclipsant les autres, garantissant que l'ensemble des requêtes de téléchargement aboutiront sur les honeypots. La stratégie que nous adoptons pour contrôler le réseau ne repose pas sur l'injection massive de Sybils, ce qui permet de contourner les mécanismes de protection récemment introduits dans KAD pour lutter contre l'attaque Sybil.

Cet article est organisé comme suit : la section 2 présente le contexte actuel de KAD, en particulier concernant l'attaque Sybil et les nouveaux mécanismes de protection insérés. Nous présentons ensuite dans la section 3 notre nouvelle approche contournant ces protections ainsi que les fonctionnalités de notre architecture rendues possibles par le contrôle de la DHT. La section 4 décrit l'implantation réalisée et les résultats obtenus lors de nos expériences sur le réseau KAD. Enfin, la section 5 conclut et présente nos prochains travaux.

2. LA SÉCURITÉ DANS KAD

2.1 Le réseau KAD

KAD est un réseau P2P structuré basé sur le protocole de routage Kademlia [5] et implanté par les clients libres eMule et aMule permettant le partage de fichiers entre utilisateurs. Ces clients étaient précédemment conçus pour utiliser le réseau P2P eDonkey composés de serveurs autonomes et ont progressivement intégré le réseau KAD entièrement décentralisé depuis 2004. Principalement utilisé en Europe et en Chine, le nombre d'utilisateurs simultanés est estimé à 3 millions, faisant de KAD l'un des plus importants réseaux P2P.

Chaque nœud de KAD possède un identifiant "KADID" de 128 bits définissant sa position dans la table de hachage distribuée (DHT). Le routage est basé sur la métrique XOR grâce à laquelle on mesure la distance entre deux pairs. La

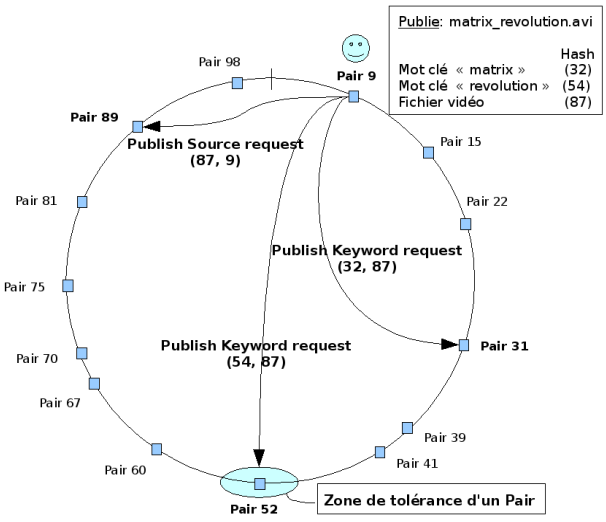

Figure 1: Publication d'un fichier sur KAD

table de routage de chaque pair est constituée de différents groupes de K contacts (appellés K-bucket) organisés en arbre de manière à ce que chaque niveau **i** détienne un groupe de pairs dont la distance est comprise entre 2^{128-i} et 2^{127-i} par rapport au KADID du pair courant. Par conséquent, ces groupes de K-contacts couvrent une zone d'autant plus petite qu'ils sont plus proches du pair. Cette organisation permet d'obtenir un routage en O(log n).

En tant que support de partage de fichiers, la fonction principale de la DHT de KAD est d'indéxer des mots clés et des fichiers. Lorsqu'un fichier est partagé, son contenu ainsi que chacun des mots clés du titre sont hachés par une fonction MD5. Les KADIDs ainsi générés sont publiés sur le réseau comme présenté par la figure 1. Les pairs potentiellement responsables de l'indexation d'un fichier ou d'un mot clé sont tous ceux dont le KADID est assez proche du hash publié. Cette distance, appelée "zone de tolérance", est définie par les 8 premiers bits du hash qui doivent être communs. Un mécanisme d'indexation à deux niveaux permet ensuite de retrouver un fichier particulier étant donné un ensemble de mots clés. Pour publier un fichier, deux types de requêtes sont nécessaires :

- les requêtes *KADEMLIA2_PUBLISH_KEY_REQ* sont envoyées vers le hash des mots clés et associent un mot clé avec un fichier ;
- les requêtes *KADEMLIA2_PUBLISH_SOURCE_REQ* sont envoyées vers le hash du fichier et associent un fichier avec une source (un pair le partageant).

Ainsi, la réalisation de services (publication ou recherche) se fait en deux étapes. La première consiste à chercher des pairs dans la zone de tolérance souhaitée. Ceci est réalisé de manière itérative par l'envoi de plusieurs requêtes *KADEMLIA2_REQ* en parallèle, chaque itération fournissant des contacts plus proches de l'identifiant recherché. Une fois la zone de tolérance atteinte, des requêtes spécifiques au service demandé sont envoyées, comme les requêtes de publication décrites ci-dessus.

2.2 L'attaque Sybil

L'attaque Sybil, telle que l'a décrite Douceur [2], consiste à créer un grand nombre de faux pairs appellés les "Sybils" et à les placer de manière précise au sein de la DHT afin d'en prendre le contrôle. L'attaque Sybil est d'autant plus efficace que le nombre de Sybils créé est important, celui-ci dépendant du rapport entre les ressources de l'attaquant et les mécanismes de défense éventuels du réseau P2P.

Des études récentes ont montré que KAD pouvait être gravement affecté par ce type d'attaque, créant d'importants dysfonctionnements dans le réseau avec peu de ressources. Steiner et al [7] furent les premiers à réaliser une attaque Sybil sur KAD permettant de contrôler une partie du réseau. L'attaque fut réalisée en deux étapes. La première consiste à parcourir le réseau P2P en envoyant de manière systématique de nombreuses requêtes de découverte de pairs, afin de connaître le plus de contacts possibles. Quand tous les pairs du réseau ou d'une zone particulière ont été découverts, la seconde étape consiste à polluer leur table de routage en annonçant les Sybils comme de nouveaux pairs. Steiner et al ont ainsi injecté 2^{16} Sybils à partir de la même machine physique dans une zone couvrant 1/256 ème de KAD, la contrôlant totalement. En effet, toutes les requêtes de publication et de recherche à l'intérieur de cette zone pouvant être interceptées, les Sybils peuvent manipuler profondément l'indexation des fichiers. Une application possible expérimentée dans [7] est une éclipse attaque faisant disparaître certains mots clés du réseau en feignant d'indexer toutes les requêtes de publication pour ces mots clés et en niant ensuite les requêtes de recherche.

Ces problèmes de sécurité n'ont été pris en considération que récemment dans le réseau P2P KAD. Les dernières versions des clients intègrent désormais des mécanismes de protection basés sur une détection locale des comportements suspects afin de limiter les effets d'une attaque Sybil.

2.3 Les mécanismes de protection

Nous avons identifié et évalué trois mécanismes de protection lors nos précédents travaux [1].

2.3.1 Filtrage des requêtes

Le premier mécanisme est l'ajout d'une mémoire à court terme (12 minutes) dans le client enregistrant les derniers paquets reçus afin de filtrer les comportements suspects. Cet historique permet par exemple de supprimer les réponses illégitimes envoyées par certains pairs alors qu'aucune requête ne leur a été adressée. Mais sa principale fonction est de permettre une protection contre le flooding : l'envoi rapide de nombreuses requêtes depuis une même source est maintenant détectée, limitant ainsi les possibilités d'un crawler. Un pair dépassant sa limite d'émission verra ses messages supprimés dans un premier temps avant d'être banni du client l'ayant détecté.

2.3.2 Limitation des adresses IP

La limitation des adresses IP est le cœur du mécanisme de protection pour combattre l'attaque Sybil. Pour se défendre, les nouveaux clients considèrent que le nombre de pairs normaux partageant une même adresse IP est très limité, contrairement aux Sybils. Ainsi, avant d'ajouter un contact à la table de routage, son adresse est désormais comparée aux contacts déjà présents dans la table et supprimée le cas échéant. La limitation tient également compte de la prox-

imité des adresses IP et interdit plus de 10 adresses venant d'un même sous réseau. Une dernière contrainte oblige ces 10 adresses à appartenir à des pairs éloignés dans la DHT pour éviter une attaque ciblée venant d'une seule entité. Ces limitations rendent l'attaque Sybil décrite précédemment extrêmement coûteuse, puisque chaque Sybil doit dorénavant afficher une adresse IP publique différente.

2.3.3 *Vérification des identités*

Enfin, l'identité d'un pair est désormais contrôlée avant que ce dernier puisse être ajouté ou modifié dans la table de routage. D'une part, son adresse IP est vérifiée par un échange de messages (three-way handshake) empêchant les Sybils d'utiliser une adresse IP usurpée. D'autre part, un pair pouvant mettre à jour son adresse IP dans la table d'un autre contact, une clé est maintenant associée à son KADID afin d'empêcher l'écrasement de l'adresse IP par un pair malveillant. Cette vulnérabilité, utilisée à grand échelle, permettait de partitionner le réseau P2P [10].

2.4 Evaluation des protections

2.4.1 *Propagation des Sybils*

Pour évaluer l'efficacité des nouvelles protections implantées, nous avons réalisé diverses attaques sur différentes versions des clients KAD [1]. Notre méthode consiste à injecter des Sybils dans la table de routage d'un client instrumenté permettant de mesurer son infection selon les mécanismes de protection activés. Notre attaque annonce les Sybils au pair ciblé en envoyant de nombreux messages *Hello_REQ* avec des KADIDs forgés. Héritée de Kademlia, bien que sensiblement différente, la table de routage d'un pair de KAD a une organisation bien précise centrée sur son KADID. Afin de maximiser la pollution de la table de routage, les KADIDs des Sybils sont calculés d'après le KADID du pair ciblé grâce à une série de masques de telle sorte que KADID(cible) XOR masque[i] = KAID(i^{eme} Sybil).

Une première attaque a été réalisée sans tenir compte des nouvelles protections : les Sybils s'annoncent rapidement et affichent tous la même adresse IP. La figure 2 montre comment se comportent deux clients différents devant cette attaque, une ancienne version vulnérable et une nouvelle incluant les mécanismes de protection.

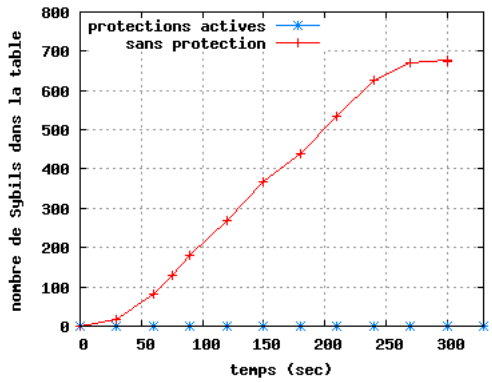

Figure 2: Propagation des Sybils dans la table de routage

Alors que l'ensemble des Sybils injectés infectent la version non protégée du client, remplissant totalement sa table de routage, aucun ne passe les filtres de protection lorsque ceux-ci sont activés.

Une seconde expérience a été réalisée sur un client protégé en adaptant le comportement de l'attaque aux nouvelles contraintes, les Sybils usurpant désormais leur adresse IP. Cette modification de l'attaque permet d'une part de dépasser les limitations d'adresses IP implantées mais également la protection contre le flooding, les messages semblant provenir légitimement de plusieurs machines alors qu'ils sont issus d'un seul hôte. La figure 3 montre que les Sybils injectés ne peuvent se maintenir dans la table de routage avec une adresse IP usurpée. Les Sybils sont dans un premier temps insérés avec un statut "non vérifié", les empêchant d'être utilisés pour router des messages, puis, comme le test vérifiant l'identité échoue, les Sybils sont finalement supprimés de la table de routage.

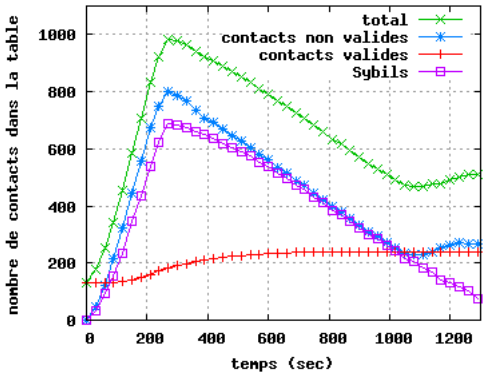

Figure 3: Evolution du statut des contacts lors d'une attaque Sybil

Ces résultats montrent que les mécanismes de protection empêchent aujourd'hui l'injection massive de Sybils provenant d'une même adresse IP, et de ce fait, augmentent considérablement le coût de l'attaque Sybil telle que pratiquée par [7]. S'il est devenu très couteux en adresses IP de réaliser une attaque à large échelle sur le réseau, nous allons montrer que des attaques très ciblées restent en revanche possibles avec peu de ressources.

2.4.2 *Vers une attaque distribuée*

En effet, la principale faiblesse autorisant des attaques ciblées sur la DHT est la possibilité offerte aux pairs de choisir librement leur KADID, ce qui reste possible. En choisissant un KADID extrêmement proche d'un hash particulier (beaucoup plus proche que n'importe quel pair choisissant son KADID aléatoirement sur 128bits) et étant donné l'algorithme de routage dans KAD "envoyer les X nœuds les plus proches de la cible", il est alors possible d'attirer avec une très forte probabilité les requêtes sur des nœuds que l'on contrôle sans injecter directement les Sybils dans la table de routage, et ainsi, dépasser les nouveaux filtres.

La section suivante décrit comment notre architecture expoite ces vunérabilités permettant de controler la DHT de KAD localement à un coût raisonnable.

3. CONCEPTION D'UNE ARCHITECTURE DE HONEYPOTS DISTRIBUÉS

3.1 Placement des Honeypeers

La première étape de notre architecture consiste à positionner précisément les nœuds contrôlés, que nous appellerons Honeypeers, par rapport au mot clé ou au fichier faisant l'objet de l'étude. Le premier paramètre à considérer est donc le KADID ciblé. Pour calculer les 128 bits qui le composent, il faut appliquer la fonction de hachage MD5 utilisée par KAD au mot clé ou au fichier binaire étudié. Une fois le KADID obtenu, celui-ci est passé en paramètre à chaque Honeypeer qui peut alors en dériver son propre KADID.

La fonction générant le KADID d'un Honeypeer copie simplement les 96 premiers bits de la cible et définit les 32 autres bits aléatoirement. Les 32 bits aléatoires permettent de générer les KADIDs des Honeypeers sans concertation avec un risque de collision très minime. D'autre part, 96bits est une valeur suffisante pour s'assurer qu'aucun autre pair n'est plus proche de la cible que nos Honeypeers. Soit N le nombre moyen de pairs étant au moins aussi proche de la cible ; N est défini par la probabilité qu'un pair choisisse aléatoirement les premiers 96bits composant son KADID similaires à la cible, multiplié par le nombre de pairs participant à la DHT. En considérant un nombre de pairs largement majoré de 5 Millions, le résultat (1) montre qu'avec un préfixe de 96bits, la probabilité qu'un pair légitime soit placé plus proche que les Honeypeers est extrêmement faible.

$$N = \frac{2^{32}}{2^{128}} \times 5 \times 10^6 = 6.31 \times 10^{-23} \quad (1)$$

3.2 Capture des requêtes pendant la procédure de recherche

Alors que les Honeypeers sont les nœuds les plus proches de la référence ciblée, le point critique de l'architecture est d'être capable d'attirer toutes les requêtes émises, étant donné les fonctions réalisant les services de KAD. Nous allons décrire dans cette section la procédure de recherche de KAD et montrer en quoi sa grande efficacité constitue une menace lorsque l'on considère des attaques locales.

Tous les services de KAD nécessitent des recherches sur la DHT réalisées par l'objet "Search". Ainsi, chaque mot clé ou fichier, publié ou recherché par un utilisateur va générer un objet "Search" autonome chargé de réaliser le service demandé. Plusieurs objets "Search" sont gérés en parallèle par l'application, mais tous sont distincts et indépendants.

La procédure de recherche est constituée de deux phases distinctes. La première partie consiste à trouver les pairs actifs dans la zone de tolérance de la référence recherchée afin de remplir le tableau des contacts *possibles* utilisé par l'objet "Search". Ce tableau contient tous les contacts compatibles et ordonnés par distance par rapport à la cible. Cette partie est réalisée en envoyant plusieurs *KADEMLIA_REQ* en parallèle avec le KADID ciblé pour paramètre. Le routage de KAD se fait de manière itérative avec plusieurs requêtes envoyées en parallèle comme présenté par la figure 4 : en premier, les 3 contacts les plus proches de la cible trouvés dans la table de routage sont interrogés pour fournir leurs connaissances encore plus proches, leurs réponses sont attendues et les meilleurs des 3 nouveaux contacts reçus sont à

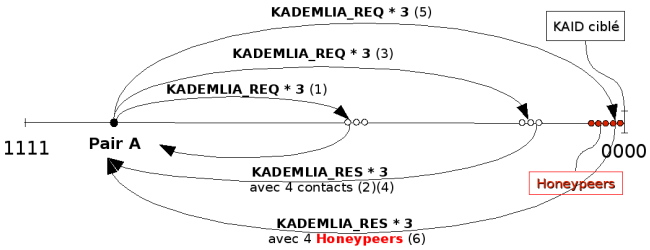

Figure 4: Echange de messages pendant la procédure de recherche

nouveau interrogés. Seuls les contacts actifs (ayant répondu) pourront être utilisés à l'étape suivante.

Tant que des contacts plus proches sont retournés à chaque itération, la recherche continue de progresser, sans commencer la réalisation du service. Il est intéressant de noter que les dernières versions des clients KAD (eMule 0.49c and aMule 2.2.4), vérifient les contacts retournés par les réponses *KADEMLIA_RES* comme s'ils allaient être insérés dans la table de routage, en particulier : un seul KADID est autorisé par adresse IP et seules deux adresses IP venant du même sous réseau /24. Ces nouvelles contraintes imposent d'utiliser une architecture distribuée.

La seconde étape débute lorsque la recherche de contacts *possibles* n'a pas retourné de contact plus proche pendant les 3 dernières secondes, signifiant que les derniers pairs interrogés ne sont pas en mesure de fournir de meilleurs pairs se rapprochant de la cible. La seconde étape consiste alors à envoyer aux meilleurs contacts *possibles* trouvés des requêtes spécifiques selon le service souhaité (par exemple *KADEMLIA2_PUBLISH_KEY_REQ* pour publier un mot clé). Les réponses retournées conditionnent la poursuite ou la fin de la recherche. L'objet "Search" se termine si le nombre nécessaire de réponses positives au service est atteint ou, en cas d'échec, par un timeout. Ainsi, une publication est considérée comme terminée lorqu'au moins 10 pairs l'ont acquittée.

Lorsqu'une requête *KADEMLIA_REQ* arrive sur un Honeypeer pour le KADID ciblé, celui-ci répond avec la liste des autres Honeypeers composant l'architecture et activés pour cette référence. Comme la liste des contacts *possibles* gérés par l'objet "Search" est toujours ordonnée par distance, tous les Honeypeers sont placés dans les meilleurs positions pour recevoir les requêtes de service. Ainsi, dès qu'un pair a découvert un des Honeypeers pendant la phase de recherche, toutes les requêtes émises pour terminer la recherche et réaliser le service seront capturées par l'architecture.

3.3 Fonctionnalités passives

L'interception de l'ensemble des requêtes par les Honeypeers permet plusieurs types d'applications pour le réseau KAD.

3.3.1 Supervision de contenus

Le comportement le plus passif possible consiste à superviser le réseau P2P en enregistrant dans une base de données toutes les requêtes destinées au contenu ciblé. Une fois enregistrées, les requêtes sont traitées normalement par les Honeypeers. Ce comportement est le moins intrusif pour le réseau car les utilisateurs ne sont impactés d'aucune manière par l'architecture dans le traite-

ment des requêtes interceptées. Ainsi, lorsqu'une requête *KADEMLIA2_PUBLISH_KEY_REQ* est capturée, un Honeypeer peut enregistrer les informations suivantes : l'adresse IP et le port de l'émetteur, le KADID du mot clé, la liste des KADID des fichiers publiés pour ce mot, et pour chacun d'eux, une liste de tags contenant les proriétés du fichier (nom complet, taille). Une requête *KADEMLIA2_PUBLISH_SOURCE_REQ* contient moins de champs : l'adresse IP et le port de l'émetteur, le KADID du fichier publié, le KADID et le port de la source. Cette fonctionnalité nous permet de connaître précisément l'activité d'un fichier étudié (quels pairs le partagent) ou d'un mot clé (quels sont les nouveaux fichiers publiés).

3.3.2 Eclipse de contenus

Un second comportement permet d'éclipser facilement des contenus du réseau. Pour éclipser un mot clé ou un fichier, le comportement des Honeypeers doit être modifié en deux points de la procédure de traitement des requêtes. Tout d'abord, il est nécessaire de contourner deux contraintes affectant la capacité d'indexation d'un client. La première limite définit le nombre maximal de références pouvant être indéxées par un client afin de ne pas donner trop de poids à un unique pair dans la DHT. La seconde limite définit le nombre maximum de références enregistrées pour un KADID particulier afin que les contenus populaires ne soient pas sur-référencés et n'affectent les autres. Si ces limites sont atteintes, un Honeypeer ne répond plus positivement aux requêtes qui sont alors retransmises à d'autres pairs et échappent à l'architecture. Pour éviter ce scénario qui est d'autant plus probable que la cible est un mot clé ou un fichier populaire, ces deux limites doivent être enlevées du client. Les Honeypeers sont alors en mesure d'acquitter toutes demandes de publication.

Pour éclipser le contenu du réseau, les Honeypeers affirment prendre en charge l'indexation du contenu ciblé en acquittant les requêtes de publication mais nient en parallèle les requêtes de recherche s'y rapportant. Les Honeypeers attirant également l'ensemble des recherches, le contenu, bien que publié, restera introuvable. Une application possible de cette fonctionnalité est de pouvoir faire disparaître les contenus malveillants du réseau, et ce faisant, d'en protéger les utilisateurs.

3.4 Déploiement de honeypots

3.4.1 Insertion de faux fichiers

Pour qu'un honeypot P2P soit attractif, les fichiers annoncés doivent présenter un grand nombre de sources. En effet, le nombre de sources pour un contenu est une information essentielle permettant à un utilisateur de trier les résulats d'une recherche. Les fichiers présentant un nombre de sources élevé sont privilégiés car ceux-ci sont populaires, plus fiables, et seront téléchargés plus rapidement. Si le nom d'un fichier ou sa taille constituent des informations pouvant être maîtrisées par une architecture classique, afficher un grand nombre de sources pour assurer une bonne visibilité à un honeypot nécessite habituellement énormément de ressources (une adresse IP par source). Notre solution ne se contente pas simplement d'annoncer des fichiers, elle contrôle la DHT, permet de déployer des honeypots très attractifs tout en nécessitant peu de ressources.

Afin d'attirer les pairs malveillants vers les faux fichiers proposés par les Honeypeers, notre architecture procède en deux étapes. La première étape consiste à prendre le contrôle du mot clé étudié et d'en polluer les résultats avec de faux fichiers. Cette étape est similaire à une éclipse de contenu mis à part que les requêtes de recherche ne sont pas niées par les Honeypeers mais renvoyées en incluant de fausses références dont les paramètres sont entièrement maîtrisés (nom, taille, nombre de sources). La pollution peut être partielle ou totale (couplée à une éclipse des autres références) selon la configuration.

3.4.2 Honeynet final

L'étape finale consiste à capturer la demande de téléchargement lorsqu'un faux fichier est sélectionné. Pour cela, notre architecture doit être capable d'attirer les recherches de sources pour les fichiers proposés et d'y répondre avec les références des Honeypeers, formant ainsi un ensemble de honeypots distribués ou Honeynet. Afin de capturer facilement les requêtes cherchant les sources des faux fichiers annoncés, les KADIDs des fichiers créés sont tous forgés pour rester extrêmement proches (96bits) du mot clé étudié. Grâce à cette optimisation, chaque recherche de sources est attirée de la même manière que la recherche de mot clé initiale, sans que les Honeypeers n'interviennent à une autre position de la DHT.

Les sources renvoyées dans les réponses étant les Honeypeers eux-mêmes (KADID,IP,port), l'architecture est assurée de capturer la requête de téléchargement. Cette organisation permet de s'assurer de l'intention d'un utilisateur malveillant. En capturant l'ensemble des requêtes émises, son comportement peut être vérifié depuis le début, par la recherche de mot clé, jusqu'à la demande de téléchargement finale au travers de l'annonce des faux fichiers. La partie suivante décrit l'implantation de notre architecture et les résultats obtenus lors de son déploiement sur le réseau KAD.

4. EXPÉRIMENTATIONS SUR KAD

4.1 Implantation de l'architecture

L'architecture est composée de 2 parties principales (figure 5) : les Honeypeers et la base de données.

4.1.1 Les Honeypeers

La principale difficulté d'implantation de notre architecture vient de la limitation des adresses IP dans les derniers clients KAD et obligeant la création d'une solution distribuée. Ainsi, tous les Honeypeers déployés sur le même KADID ciblé doivent présenter une adresse IP publique différente pour ne pas être filtrés. Par ailleurs, un maximum de 2 Honeypeers peuvent appartenir au même sous réseau. Pour répondre à ces contraintes, nous avons choisi d'exécuter notre Honeynet sur le réseau PlanetLab Europe, comme décrit par la figure 5.

Les Honeypeers exécutent une version modifiée du client aMule dans sa version démon qui présente les avantages d'être très légère (sans interface graphique) et déployable à distance. Le code du client a été profondément remanié pour implanter les différentes fonctionnalités décrites précédemment, en particulier, les fonctions traitant les paquets reçus et plusieurs paramètres de KAD ont été modifiés. Le comportement souhaité pour un Honeypeer (supervision, éclipse, honeypot), le KADID ciblé ainsi que d'autres paramètres

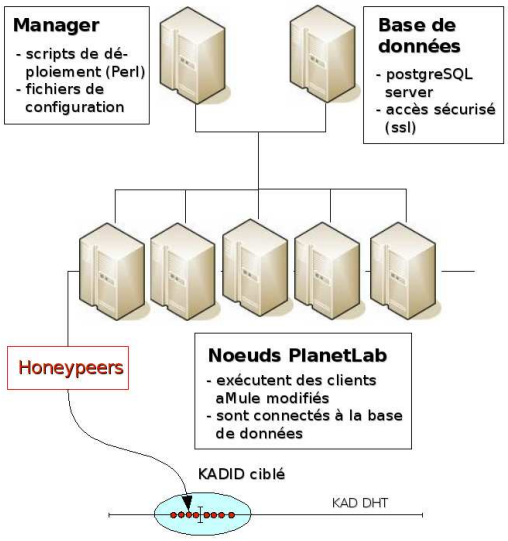

Figure 5: Architecture réseau du Honeynet

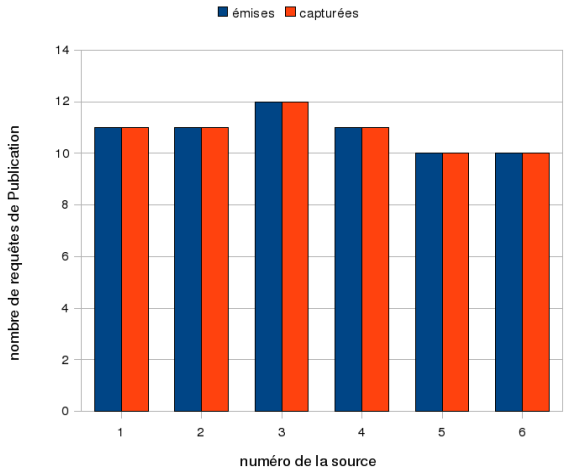

Figure 6: Nombre de requêtes de publication capturées pour différentes sources

sont définis dans un fichier de configuration spécifique lu par notre client modifié.

Des scripts Perl permettent d'automatiser le déploiement de l'architecture sur PlanetLab et le changement de comportement des Honeypeers. Ces scripts sont rassemblés et exécutés depuis un ordinateur "Manager" gérant le Honeynet. L'écriture des scripts a été facilitée par l'architecture homogène de PlanetLab où chaque Honeypeer exécute le même client dans le même environnement. La première version du Honeynet implantée ne peut cibler qu'un KADID à la fois, le passage à l'échelle sera assuré dans les prochains développements. Etant donné la très faible consommation en ressources d'aMuled, un nœud pourra facilement gérer des dizaines de KADID distincts sur des ports différents dans les prochaines versions.

4.1.2 La base de données

Le second composant principal du Honeynet est la base de données enregistrant les informations capturées par les Honeypeers. La base de données est implantée par un serveur PostgreSQL dont l'accès est sécurisé par un module SSL additionnel afin d'empêcher les connexions indésirables. La plupart des accès à la base de données se font en écriture mais les Honeypeers sont également amenés à la consulter pour utiliser certaines informations. Ainsi, la liste des Honeypeers(KADID, IP et port) activés pour un même KADID et celle décrivant les faux fichiers à annoncer sont lues dans la base.

Les différentes tables ont été conçues pour pouvoir anonymiser facilement les adresses IP récoltées sans toutefois perdre trop d'informations. Lorsqu'une nouvelle adresse IP est observée par le Honeynet, la base de données lui attribue un index unique pendant toute l'exécution.

4.2 Résultats

Nous avons déployé l'architecture composée de 20 Honeypeers sur le réseau KAD.

La première expérience réalisée a pour objectif de vérifier que l'architecture parvient bien à attirer toutes les requêtes pour le KADID étudié. Pour cela, nous avons publié un KADID ciblé par le Honeynet depuis 6 différents clients instrumentés. Pour éviter toute mesure biaisée par les conditions initiales, chacune des 6 sources publie depuis une position différente sur la DHT et rejoint le réseau depuis une liste différente de contacts. Etant donnée cette configuration, nous pouvons comparer le nombre de requêtes de publication émises par les différentes sources et celui constaté par notre architecture pour chacune de celles-ci. La figure 6 montre que l'ensemble des requêtes émises sont capturées par le Honeynet, comme le laissait supposer l'analyse de la procédure de recherche de KAD. Le nombre de requêtes émises (entre 10 et 12) correspond au facteur de réplication (10) nécessaire pour publier un fichier.

Les résultats d'une recherche effectuée par un client KAD normal pendant notre test sont présentés dans la figure 7.

Nous avons ensuite testé les fonctionnalités de notre architecture. L'expérience consiste à éclipser et à réorienter vers le Honeynet les recherches réalisées pour un mot clé populaire. Pour cela, nous avons choisi de cibler le mot clé "spiderman" pendant une journée, éclipsant les fichiers indexés et les remplaçant par 4 faux fichiers affichant un nombre différent de sources, 2 apparaissant populaires et 2 faiblement partagés.

Nous pouvons observer que le Honeynet réussit parfaitement à éclipser les véritables références et à les remplacer par les 4 faux fichiers proposés. La figure 8 présente pour chacun des faux fichiers, la proportion de recherches de sources émises par des pairs distincts. Nous pouvons constater que les 2 fichiers populaires sont choisis en priorité par 96% des utilisateurs, ce qui confirme l'importance de maîtriser le nombre de sources pour créer un honeypot attractif. Cette expérience illustre l'efficacité de notre approche et le contrôle qu'elle permet sur le réseau KAD.

5. CONCLUSION

Alors que les anciens clients KAD étaient largement affectés par une attaque Sybil ne nécessitant qu'un seul ordinateur, notre évaluation des nouveaux mécanismes de protection a montré qu'une détection des comportements sus-

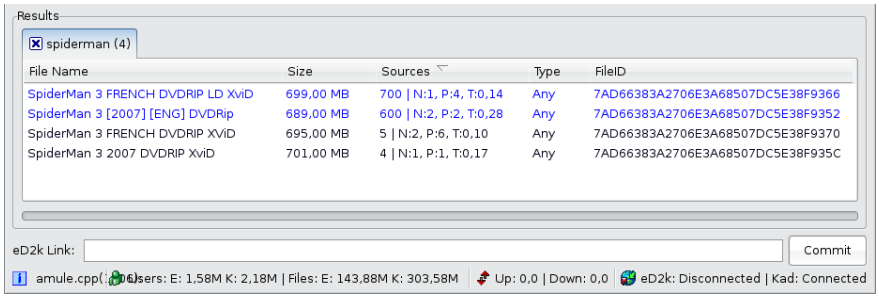

Figure 7: Résultat d'une recherche sur "spiderman" ciblé par le Honeynet avec 4 faux fichiers

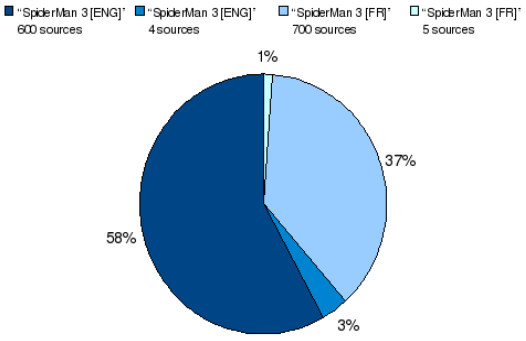

Figure 8: Proportion des recherches de sources reçues pour chaque faux fichier

pects par chaque client permettait de limiter efficacement cette menace, en augmentant considérablement son coût en ressources.

Néanmoins, nous avons montré que certaines caractéristiques de KAD, à savoir le fait de choisir librement son KADID associé à un algorithme de recherche très performant, le rendent vulnérable à des attaques ciblées et distribuées. Exploitant ces failles, notre architecture de Honeynet est capable de superviser discrètement et de contrôler efficacement le réseau KAD au travers des mots clés et fichiers indexés. Les tests de notre architecture réalisés sur le réseau réel ont montré l'efficacité de notre approche, celle-ci étant capable d'attirer toutes les requêtes pour un KADID donné. Tout en nécessitant peu de ressources (20 nœuds), le Honeynet peut totalement contrôler un mot clé très populaire et annoncer de faux fichiers très attractifs pour alimenter les honeypots.

Nos travaux futurs consistent à améliorer notre architecture afin de pouvoir cibler plusieurs KADID simultanément. Ensuite, le Honeynet sera déployé pour étudier et combattre les différents types de contenus malveillants propagés dans KAD (virus, malware, ...) ou, plus généralement, tout phénomène lié au contenu des réseaux P2P.

Remerciements : Ces travaux sont financés par le projet ANR MAPE (Measurement and Analysis of Peer-to-peer Exchanges for pedocriminality fighting and traffic profiling), sous le contrat ANR-07-TLCOM-24 et supportés par le réseau d'excellence européen EC IST-EMANICS(#26854).

6. REFERENCES

[1] T. Cholez, I. Chrisment, and O. Festor. Evaluation of sybil attacks protection schemes in kad. In *AIMS 2009: 3rd International Conference on Autonomous Infrastructure, Management and Security*, Twente, The Netherlands, 2009. Springer-Verlag.

[2] J. R. Douceur. The sybil attack. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 251–260, London, UK, 2002. Springer-Verlag.

[3] M. Latapy, O. Allali, and C. Magnien. Measurement of edonkey activity with distributed honeypots. In *HOTP2P 2009. Sixth International Workshop on Hot Topics in Peer-to-Peer Systems*, Rome, Italy, 2009.

[4] H. Lee and T. Nam. P2p honeypot to prevent illegal or harmful contents from spreading in p2p network. volume 1, pages 497–501, Feb. 2007.

[5] P. Maymounkov and D. Mazieres. Kademlia: A peer-to-peer information system based on the xor metric. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 53–65, London, UK, 2002. Springer-Verlag.

[6] M. Steiner, T. En Najjary, and E. W. Biersack. A global view of KAD. In *IMC 2007, Internet Measurement Conference, October 23-26, 2007, San Diego, USA*, Oct 2007.

[7] M. Steiner, T. En Najjary, and E.W. Biersack. Exploiting KAD: possible uses and misuses. *Computer communications review, Volume 37 N5, October 2007.*

[8] D. Stutzbach and R. Rejaie. Capturing accurate snapshots of the gnutella network. In *8th IEEE Global Internet Symposium*, March 2005.

[9] D. Stutzbach and R. Rejaie. Understanding churn in peer-to-peer networks. In *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 189–202, New York, NY, USA, 2006. ACM.

[10] P. Wang, J. Tyra, E. Chan-Tin, T. Malchow, D. Foo Kune, N. Hopper, and Y. Kim. Attacking the kad network. In *SecureComm 2008: the 4th International Confere on Security and Privacy in Communication Networks*, Istanbul, Turkey, 2008. ACM.