Une attaque par rejeu sur le protocole SEND

Tony Cheneau (tony.cheneau@it-sudparis.eu)*
Jean-Michel Combes (jeanmichel.combes@orange-ftgroup.com) †

Résumé : L'IPv6 commence à se déployer. Dans un futur proche, la transition sera pleinement entamée. IPv6 et plus particulièrement son protocole de découverte de voisins (Neighbor Discovery Protocol) est sujet à de nombreuses attaques (souvent héritées de l'IPv4). Conscient de ce problème, l'IETF a abouti à la normalisation du protocole de Secure Neighbor Discovery (SEND). Nous verrons dans cet article que ce protocole est sujet à une nouvelle attaque par rejeu. Plusieurs parades seront aussi proposées.

Mots Clés : IPv6, NDP, SEND, attaque, rejeu

1 Introduction

Ce court article porte sur la description d'une attaque sur le protocole de découverte de voisin (Neighbor Discovery Protocol, NDP) en IPv6 efficace même lors de l'utilisation du mécanisme de SEcure Neighbor Discovery (SEND). La première partie décrit la fonctionnalité de détection d'adresse dupliquée (Duplicate Address Detection, DAD). La deuxième partie explique une attaque déjà connue sur celle-ci. La troisième partie présente de quelle façon le mécanisme SEND améliore la sécurité du protocole de découverte de voisin et du mécanisme de détection d'adresse dupliquée en particulier. La quatrième partie détaille l'attaque que nous avons trouvé et qui reste valide même en utilisant les protections offertes par le protocole SEND. Et enfin, la dernière partie discute des solutions et des parades que nous proposons afin d'empêcher cette attaque.

2 Présentation d'une des fonctionnalités du NDP : le mécanisme de Duplicate Address Detection

Le mécanisme Neighbor Discovery Protocol (NDP)[NNSS07] fournit en IPv6 un certain nombre de fonctionnalités indispensables au bon fonctionnement du protocole IPv6. La plus connue est la fonctionnalité de résolution d'adresse qui correspond à ce qu'est ARP[Plu82] en IPv4. Ce protocole propose aussi d'autres fonctionnalités. Celle qui va nous intéresser dans notre article, la Duplicate Address Detection (DAD), permet de détecter lorsque deux noeuds veulent utiliser la même adresse et évite la future collision en refusant l'attribution de l'adresse. Cela est équivalent au "gratuitous ARP" en IPv4. Cette fonctionnalité est d'autant plus importante, qu'en IPv6, les nouveaux hôtes peuvent utiliser l'"autoconfiguration sans état" [TNJ07] et s'attribuer eux-même une adresse (autogénérée).

* Institut Télécom et Management SudParis, 9, rue Charles Fourier 91011 Évry Cedex
† France Telecom R&D 38-40, rue du Général Leclerc 92794 Issy-les-Moulineaux

2.1 Principe de fonctionnement du mécanisme DAD

Le mécanisme Duplicate Address Detection s'applique à toutes les adresses de type unicast avant qu'elles ne soient assignées aux interfaces réseaux et ce, qu'elles soient générées de manière manuelle, sans état ("stateless") ou avec état ("stateful" en anglais). Cette fonctionnalité peut néanmoins être désactivée par les administrateurs du système.

Le mécanisme Neighbor Discovery Protocol utilise des messages de type ICMPv6(défini au-dessus du protocole IPv6)[CDG06]. Dans le cadre du mécanisme de DAD, seulement deux types de messages nous intéressent, le message Neighbor Solicitation (NS) et le message Neighbor Advertisement (NA). Lors de la résolution d'adresse, le message Neighbor Solicitation est utilisé pour demander l'adresse physique d'un noeud (p. ex. adresse MAC) avec lequel on désire communiquer en s'adressant à lui grâce à son adresse IP (version 6). Ce message contient un champ cible qui est rempli avec l'adresse (IPv6) du nœud que l'on souhaite contacter. Si cette cible existe, celle-ci répond par un message Neighbor Advertisement destiné au nœud qui avait émis la requête et contient dans un de ses champs une option transportant l'adresse physique de ce nœud pour l'interface réseau concernée. Cette association entre l'adresse logique et l'adresse physique sera ensuite conservée dans la table de cache du voisinage (Neighbor Cache).

Pour le mécanisme DAD, cet échange de requête/réponse est utilisé de manière plus fine. Le nœud ne s'approprie pas l'adresse qu'il désire tant que la procédure n'a pas été satisfaisante ; durant cette procédure, cette adresse sera dite "provisoire" ("tentative" en anglais). Pour être plus précis, si le nœud reçoit du trafic destiné à une adresse "provisoire" il ne doit pas le traiter ni y répondre. La procédure consiste à émettre un message Neighbor Solicitation avec comme cible son adresse "provisoire" et en adresse source, l'adresse de type "non spécifiée" (::). Si quelqu'un répond, à ce message NS par un message Neighbor Advertisement, cela veut dire que l'adresse est déjà prise et qu'un nœud possède déjà cette adresse, on considère que la tentative d'obtention d'adresse échoue : le nœud ne pourra obtenir cette adresse. Il n'y a pas d'autres essais pour obtenir cette adresse, l'administrateur devra intervenir sur le nœud afin de le configurer avec une autre adresse. Il existe un autre cas où l'on peut ne pas obtenir l'adresse : quand le nœud reçoit un message NS avec comme adresse cible l'adresse "provisoire" que l'on désire. Cela signifie qu'un autre noeud effectue aussi une procédure DAD pour la même adresse. Dans ce cas, aucun des deux nœuds effectuant le mécanisme de DAD sur la même adresse ne pourra l'obtenir.

Le mécanisme de DAD n'est pas infaillible, notamment si celui-ci se déroule durant le temps où plusieurs nœuds du même réseau sont temporairement "séparés" (perte de connexion ou chute d'un lien entre les noeuds) et qu'un ou plusieurs des noeuds effectuent une procédure DAD. Ceux-ci pourront s'attribuer la même adresse sans que la procédure ne détecte la collision.

2.2 Détail des échanges du mécanisme de DAD

Pour le nœud, la procédure commence en écoutant le groupe multicast "tous les nœuds" ("all-nodes multicast") et le groupe multicast du nœud sollicité ("Solicited Node multicast"). Le premier lui permet recevoir les demandes de résolution d'adresse ("Address Resolution") pour cette adresse et le second lui permettra de recevoir les messages émis par d'autres nœuds faisant eux aussi une DAD sur cette adresse. Afin d'écouter ces derniers, le noeud doit émettre une requête de Multicast Listener Discovery (MLD) [DFH99] ; cette dernière sera non sécurisée.

Lorsqu'un nœud déclenche la procédure de DAD, il envoie un message Neighbor Solicitation, message de type ICMPv6. Il est important de noter le contenu de certains champs de l'en-tête IPv6[DH98] dans ce paquet :

- la valeur du champ **nombre de sauts limite** est fixée à la valeur maximale de 255 (comme pour tous les messages du NDP). Cette valeur est décrémentée lorsque l'on passe un routeur. Ainsi, il devient possible en vérifiant ce champ de ne communiquer qu'avec les noeuds à 1 "saut" de distance (les voisins) [GHM+07].
- l'**adresse source** du paquet IPv6 est l'adresse de type non spécifiée (::) ;
- l'**adresse de destination** est l'adresse multicast du nœud sollicité ("Solicited-Node Multicast Address") de l'adresse "provisoire", c'est-à-dire les 3 derniers octets de l'adresse provisoire que l'on concatène avec le préfixe FF02::1:FF00:0/104.

Lors de l'envoi de ce message NS, on observe pour l'en-tête ICMPv6 :

- le champ **adresse cible** est rempli avec l'adresse "provisoire" ;
- l'option **adresse couche liaison de la source** n'est pas utilisée. Ainsi, deux nœuds peuvent chacun émettre un même message NS identique.

Ce message sera envoyé "DupAddrDetectTransmits" fois. Par défaut, cette valeur est fixée à 1 comme indiqué dans la spécification [TNJ07].

Suite à ce message, Un nœud réagira suivant le type de paquets reçu en réponse :

- Dans le cas où la réponse est un message NS valide (comme décrit dans la spécification [NNSS07]), où l'adresse source du message NS est l'adresse non spécifiée et où l'adresse destination est celle du groupe multicast "Solicited-Node", alors le nœud expéditeur effectue aussi une procédure DAD(cf fig. 1) : cela conduit à un échec de la procédure de DAD ;
- Dans le cas où il s'agit d'un message NA valide (au sens de la spécification [NNSS07]) et où l'adresse cible est notre adresse "provisoire", alors un autre noeud l'a déjà attribué sur une interface : cela conduit à un échec de la procédure de DAD ;
- Dans tous les autres cas, les paquets ne seront pas analysés et après DupAddrDetectTransmits essais, l'adresse sera considérée comme unique et pourra être assignée à l'interface.

Avec l'autoconfiguration sans état, il est important de noter que si le mécanisme de DAD échoue alors il n'y a pas d'autres essais et une nouvelle adresse devra être assignée autrement. En particulier, pour les adresses qui auront été construites automatiquement grâce au format EUI-64 modifié [HD06].

3 L'attaque sur le mécanisme de DAD

Une attaque sur le mécanisme de DAD a été identifiée dans la RFC 3756, nommée "IPv6 Neighbor Discovery (ND) Trust Models and Threats" [NKN04]. Ce document a été écrit par le groupe de travail SEND de l'IETF afin d'identifier les différentes attaques possibles sur le protocole NDP. L'attaque se compose comme il suit : l'attaquant va leurrer le mécanisme de DAD et le faire aboutir dans un des deux cas où il échoue afin que la victime ne puisse pas s'attribuer d'adresse. Comme il y a un nombre fini d'essais pour réobtenir une adresse, le mécanisme DAD finit toujours par échouer, c'est une attaque de type DoS. Pour que l'attaque soit réalisable, il faut que l'attaquant puisse écouter sur le réseau toute requête nécessaire pour effectuer la procédure de DAD (comme p. ex. les messages NS avec comme adresse source l'adresse non spécifiée sont caractéristiques de la

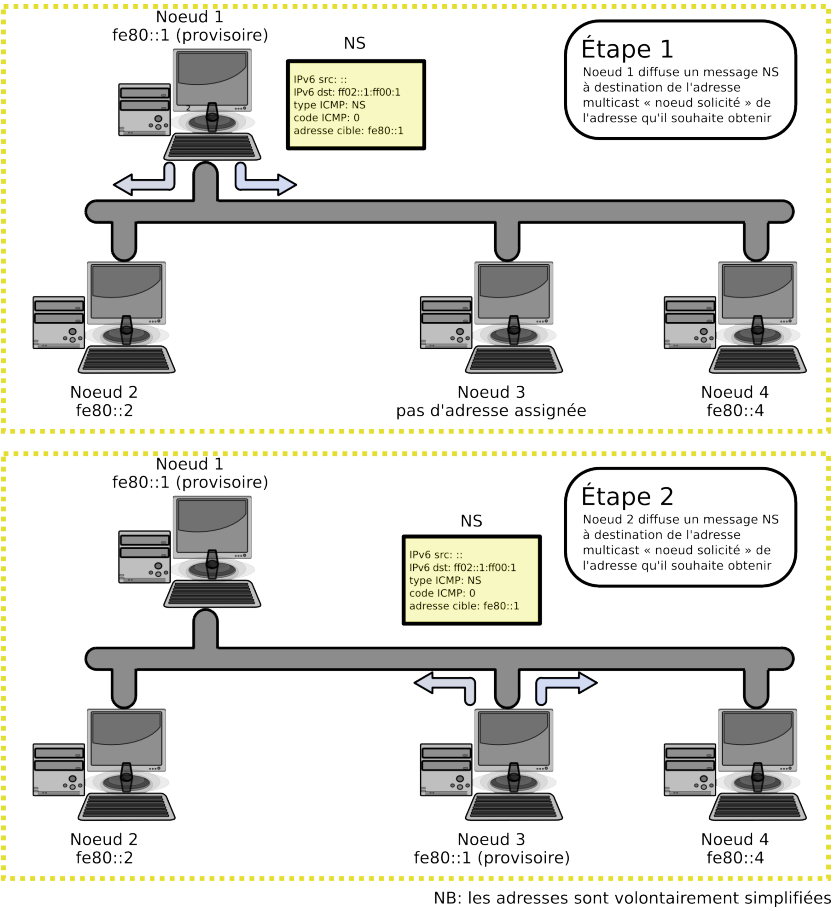

Fig. 1: La machine 1 et la machine 3 souhaitent une adresse qui n'est pas attribuée, mais qui est la même et ne pourront pas l'obtenir.

procédure de DAD ; cela implique de pouvoir joindre le groupe multicast "Solicited-Node"). Il a ensuite deux choix, il peut envoyer un message NS avec, comme adresse source, l'adresse non spécifiée et, comme adresse cible, l'adresse de la victime (cas de la fig.1) ou un message NA avec, comme adresse cible, l'adresse "provisoire" de la victime. Il peut ainsi empêcher l'arrivée de nouveaux nœuds n'ayant pas encore d'adresse. L'efficacité de l'attaque dépend fortement du type des liens, car il faut que l'attaquant puisse recevoir la première NS envoyée par la victime et qu'il puisse y répondre. En effet l'attaquant doit être capable de joindre le groupe multicast "Solicited-Node", ce qui n'est pas aisé dans le cas d'une technologie de niveau 2 de type "point à point", par exemple l'ADSL.

Cette attaque a été implementée par le groupe de "The Hacker's Choice"[1]. Elle est téléchargeable sur le site web et fait partie d'une compilation d'outils d'attaque sur l'IPv6

[1] `http://www.thc.org/`

nommée “THC IPv6 Attack Toolkit”. Le programme à exécuter est *dos-new-ipv6*.

4 Les fonctionnalités de SEND protégeant DAD

4.1 Les protections apportées par SEND

Un avantage de SEND est la possibilité de prouver à ses correspondants que l'on est propriétaire d'une adresse IPv6. Une paire de clef publique/clef privée est générée par chaque nœud, la clef publique servant ensuite à générer une adresse. Une nouvelle option du NDP, l'option CGA, est utilisée pour porter l'information de clef publique de l'hôte. Cette option combinée avec l'option de signature prouve la possession de l'adresse.

4.1.1 Les CGA

SEND utilise des adresses générées à base d'identifiants cryptographiques. Celles-ci sont nommées Cryptographically Generated Addresses (CGA). L'avantage de ce type d'adresse est de pouvoir prouver leur possession, car on repose sur des propriétés cryptographiques. Une paire de clefs publique/privée est créée. Un condensé de la clef publique concaténée à d'autres paramètres (par exemple : le préfixe du sous-réseau) est utilisé pour la valeur de l'identifiant d'interface. Parmi les multiples paramètres qui peuvent influencer ce condensé, on trouve un nombre aléatoire qui permet à deux hôtes ayant la même clef publique (très improbable) d'avoir peu de chances de générer la même adresse. Ces paramètres sont tous stockés dans une structure de données nommée “Paramètres CGA” (voir la fig. 2). Celle-ci est transportée par tous les paquets générés dans le cadre de l'utilisation de SEND à l'intérieur de l'option CGA. Cette option permet à n'importe quel nœud recevant le paquet de pouvoir extraire la clef publique du paquet et de recalculer le condensé servant d'identifiant d'interface, créant de par ce fait un lien unique entre l'adresse et la clef. Par la suite, nous verrons que la clef privée sert à signer le message, prouvant ainsi la possession de l'adresse et l'intégrité d'une partie du message.

Les CGA sont bien plus complexes que ce résumé, mais dans le cadre de cet article, il n'est pas nécessaire d'aller plus en détail. On notera tout de même qu'il existe, dans le document RFC 3972 [Aur05], d'autres mécanismes protégeant les CGA afin de rendre plus difficile le travail d'une attaque par “force brute”.

4.1.2 La signature numérique : l'option RSA

L'option RSA permet d'assurer que l'on possède la partie secrète de notre paire de clefs publique/privée et que cette paire est la même que celle utilisée pour la création de la CGA. Elle permet de prouver à la fois la possession de l'adresse (par la preuve de la possession du secret) et l'intégrité du message. Cette option comporte une signature numérique qui englobe les parties importantes du message. La signature est au format PKCS#1 v1.5, c'est-à-dire une signature RSA. Cette signature porte sur les adresses source et destination du paquet, l'entête ICMP du paquet, toutes les options NDP jusqu'à l'option RSA (qui doit être la dernière option ICMP). L'option CGA est présente en même temps que l'option RSA (permettant la vérification d'un message par lui même).

4.1.3 L'anti-rejeu : options horodatage et nonce

Il serait inutile d'avoir une protection sur l'intégrité si un attaquant pouvait rejouer les paquets. En effet, il pourrait par exemple réutiliser un ancien message authentifié d'un

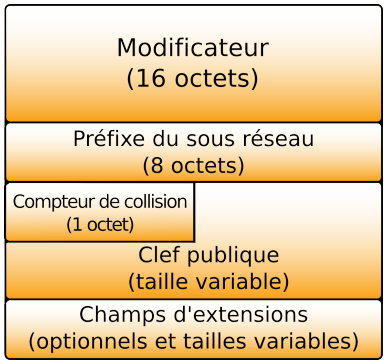

Fig. 2: Les paramètres CGA, transportés par l'option CGA et protégés par la signature numérique RSA

voisin afin de faire croire aux voisins qu'un nœud est toujours connecté. Dans SEND, pour s'assurer que l'on ne puisse rejouer les paquets, on a ajouté deux types de protections anti-rejeu : la première, l'horodatage permet de savoir si un message est "périmé", car trop ancien, et le second, le *nonce*, permet de vérifier qu'une réponse est bien liée à une requête précédemment émise.

L'option d'horodatage ("timestamp") permet de s'assurer que les "advertisements" réguliers (comme l'annonce d'un préfixe par un routeur) ne sont pas rejoués. SEND impose que tous les messages possèdent une option horodatage.

Un format compatible UNIX qui représente le nombre de secondes écoulées depuis le 1er janvier 1970, d'une granularité de 1/64000 de secondes, est utilisé pour représenter l'horodatage dans le paquet. Cette granularité est suffisamment importante pour que deux paquets consécutifs ne puissent pas avoir le même horodatage. En pratique, on ne se sert pas de la valeur de l'horodatage mais d'un écart autour de celle-ci. À la réception d'un nouveau paquet, on compare l'horodatage du paquet avec la valeur de l'horloge de la machine pour vérifier si l'écart n'est pas trop important. Si c'est le cas, le paquet n'est pas traité. Seule exception, si le paquet est un message NS alors le destinataire peut répondre, mais ne doit pas mettre son cache de voisinage à jour. À la réception d'un message SEND valide d'un voisin, le nœud peut calculer la différence d'horloge entre eux. Cette différence permet d'évaluer la pertinence de l'horodatage des paquets à venir dans l'échange. Ainsi, un hôte pourrait recevoir un paquet avec une valeur d'horodatage qui serait inférieur à son horloge interne (si par exemple les horloges des deux hôtes ne sont pas synchronisées par des serveurs de temps). Grâce à cet écart, l'hôte peut vérifier que le paquet reçu a bien été émis après le dernier paquet valide reçu de ce même voisin. Il est ainsi possible de calculer si le message est un ancien paquet qui est rejoué (horodatage expiré).

Le *nonce* (nombre aléatoire) a pour but de prouver au destinataire d'un paquet que le paquet reçu est une réponse légitime. Chaque nouvelle "solicitation" génère un nouveau *nonce* et celui-ci sera recopié pour toutes réponses ("advertisement") à ce message. Cela permet de savoir que les échanges sont des échanges "frais". L'option *nonce* est obligatoire pour tous les messages sauf pour les messages de type multicast émis généralement par les routeurs (bien qu'il soit conseillé de l'utiliser même dans ce cas).

Les paquets reçus qui ne possèdent qu'une seule de ces deux options seront traités comme non sûrs : certaines implémentations rejetteront ces paquets.

4.2 Les protections de SEND appliquées au NDP

Le document RFC 3971[AKZN05] "SEcure Neighbor Discovery (SEND)" contient les modifications suivantes à propos du mécanisme de DAD spécifié dans le RFC 4862 [TNJ07] :

- un nœud sécurisé par le protocole SEND utilisera, comme adresses unicast pour ses interfaces, des adresses CGA ;
- si la procédure DAD indique que l'adresse "provisoire" est déjà utilisée, le nœud devra regénérer une nouvelle adresse CGA "provisoire" ;
- si après trois essais consécutifs, aucune adresse unique n'a pu être générée, le système consignera une erreur dans ses journaux (logs) et abandonnera la génération d'adresse pour cette interface.

Les trois essais ne sont pas tous effectués de la même façon, en effet :

- lors de la procédure de DAD sur la première adresse provisoire générée, le nœud accepte les messages NS et NA de tous les hôtes, qu'ils utilisent le protocole SEND ou non ;
- lors du deuxième et du troisième essai, le nœud ne tiendra plus compte que des messages protégés par le protocole SEND.

Ce même document ajoute que : le protocole SEND permet de contrer l'attaque connue sur le mécanisme de DAD (présentée section 3) en rendant obligatoire à un message Neighbor Advertisement d'utiliser l'option de Signature RSA. Ainsi, un attaquant ne pourrait plus forger un message NA en réponse au message NS envoyé par le nœud. L'option de Signature RSA prouvera qu'un éventuel nœud possédant l'adresse testée a bien le droit de l'utiliser. Sans cette option, le nœud ne traitera pas les réponses (après le premier essai). Cette partie met aussi en valeur que le protocole SEND n'écoute les messages venant de nœuds n'utilisant pas SEND que pour la première fois, au risque d'entrer en collision lors de la génération d'une adresse pour le deuxième ou le troisième essai avec un nœud n'utilisant pas SEND et possédant également cette adresse.

Le document RFC 3972[Aur05] "Cryptographically Generated Addresses (CGA)" définit la structure de données "paramètres CGA" (voir fig 2) qui contient un paramètre "compteur de collisions" permettant de compter le nombre de collisions réalisées durant la procédure de DAD. Cette structure est ensuite incluse dans le paquet et protégée par la signature numérique.

Les deux grandes améliorations du mécanisme de DAD en utilisant SEND vis à vis du NDP classique sont :

- lors d'une collision durant la procédure de DAD, il n'y a plus d'échec immédiat de la procédure, mais génération d'une nouvelle adresse et nouveaux essais (jusqu'à trois fois) ;
- les échanges de la procédure de DAD sont sécurisés par les options du NDP de SEND.

5 L'attaque remise au goût du jour pour SEND

L'attaque consiste à écouter les messages NS émis sur le réseau qui correspondent aux messages NS utilisés lors de la procédure de DAD (caractérisés par l'adresse source non

spécifiée). Ceux-ci sont protégés par les 4 types d'options de SEND. Il suffit, pour que l'attaque fonctionne, de rejouer immédiatement ces paquets. En effet, ceux-ci possèdent des propriétés de signature totalement valide, l'horodatage est correct et le *nonce* n'est pas vérifié dans ce cas précis (il est normalement utilisé pour être sûr qu'un "advertisement" est bien une réponse liée à une "solicitation" émise par le nœud).

Les protections offertes par SEND sont toutes déjouées :

- le paquet peut être rejoué au niveau de la couche 2 ou de la couche 3, la signature de l'option RSA ne porte que sur des champs de la couche 3, la recopie offre un paquet avec une signature valide ;
- il n'y a aucune sémantique associée à l'option *Nonce* lors de la réception de message NS ;
- l'horodatage est valide, car à la réception d'un nouveau message NS lors du mécanisme de DAD, un nœud comparera la différence entre la valeur de l'horodatage de ce message NS et la valeur de l'horodatage du dernier message SEND reçu et accepté. Comme l'on rejoue le message NS de la victime, son horodatage va contenir la valeur de son horloge interne. La victime va ensuite comparer cet horodatage (son horloge interne) avec l'horodatage du dernier paquet arrivé. L'écart sera normalement inférieur avec la valeur du Delta maximum permis. Cette valeur par défaut est fixée à 5 minutes.

En revanche, il est important de noter encore une fois que l'attaquant doit joindre le groupe multicast "Solicited-Node" ou être sur le même lien que la victime afin de pouvoir détecter l'exécution d'une procédure de DAD. L'écoute des paquets des voisins est très facilement réalisable sur les réseaux sans fils ouverts (comme les *Hot Spot* Wifi), les réseaux où les noeuds sont dans le même domaine de diffusion ou même quand ils sont reliés par un switch ne gérant pas le "MLD Snooping" en IPv6 (dans ce cas, les switchs effectuent souvent de la diffusion sur tous leurs ports des paquets à destination d'une adresse multicast, comme mentionné dans le document Internet Draft nommé "IPv6 Multicast Deployment Issues" [Sav05]).

La procédure de DAD dure "DupAddrDetectTransmits $*$ RetransTimer" millisecondes avant d'être validée. Par défaut, DupAddrDetectTranmsits vaut 1 et la valeur RetransTimer vaut 1000. Il faut donc rejouer le paquet en moins d'une seconde pour que la victime n'ait pas le temps de valider sa procédure de DAD.

On pourrait penser que le nœud ne va pas traiter ce paquet qui est une copie du sien ou détecter un rejeu. Il y a en effet des règles à appliquer dans le cadre d'interfaces qui possèdent des propriétés de "loopback".

Le document RFC 4862[TNJ07] "IPv6 Stateless Address Autoconfiguration" définit le comportement attendu et la sémantique du "loopback" : si le nombre réel de messages Neighbor Solicitations reçus excéde le nombre attendu basé sur une sémantique de "loopback" (c.-à-d. pour une interface ne faisant pas de "loopback" sur ses paquets, la réception d'une sollicitation ou plus), l'adresse "provisoire" est dupliquée. Ce cas se produit lorsque deux nœuds effectuent une procédure DAD simultanément, envoyant ainsi un message de Neighbor Solicitation identique au même instant.

L'attaque est donc valide et le rejeu de message NS n'est finalement pas déjoué par le protocole SEND dans ce cas. Les conséquences de l'attaque mènent à un DoS qui empêche l'arrivée de nouveaux nœuds dans le réseau.

Cette attaque a été testée sur la seule implémentation publique à ce jour, celle de

NTT DoCoMo [2] : l'attaque ne fonctionne pas. En effet, la procédure de DAD dans cette implémentation, comme expliqué à la fin du manuel utilisateur, n'est pas complète.

Afin de démontrer la faisabilité de l'attaque, nous mettons à disposition la preuve de concept suivante, écrite grâce à l'outil Scapy6[3] (lui même basé sur Scapy[4] auquel il rajoute un support de l'IPv6). Celle-ci fonctionne aussi quand le NDP n'est pas protégé par SEND.

Listing 1: Preuve de concept de l'attaque de dénie de service sur la procédure de DAD (fonctionne aussi quand cette dernière est protégée par SEND)

```
import scapy6

# interface réseau sur laquelle on va écouter/rejouer les paquets
conf.iface = 'eth0'

# écoute le trafic d'une interface réseau
sniff(store=0, filter="ip6",
      # pour n'écouter que les messages NS utilisés lors du DAD
      lfilter = lambda x: x.haslayer(ICMPv6ND_NS) \
                and x.getlayer(IPv6).src== "::",
      # rejoue le paquet quand celui-ci rempli les critères
      prn = lambda x: sendp(x), count=0)
```

6 Solutions et parades à l'attaque

Une parade simple consisterait à donner une sémantique au *nonce* lors de la procédure de DAD. Il est possible de ne tenir compte, dans les messages que l'on reçoit durant le procédure de DAD, que des messages NS ayant un *nonce* différent des messages émis. Cela permettrait de rester rétro-compatible avec les implémentations actuelles.

Une autre solution possible, plus radicale mais toujours rétro-compatible, est de valider l'adresse après trois essais infructueux.

Concernant cette solution, le document RFC "Optimistic DAD" [Moo06] apporte quelques résultats, quant aux probabilités qu'une collision se produise réellement. La formule de calcul de la possibilité qu'une collision se produise si un nœud se déplace dans un réseau où un certain nombre de nœuds sont déjà présents est :

$Pc(n, k, m) = 1 - [(1 - k/n)^m]$

Où, **n** est le nombre d'adresses possibles dans l'espace d'adresse, **k** est le nombre de nœuds existants dans ce réseau et **m** est le nombre de fois où le nœud se déplace et change d'adresse.

Pour un nœud se déplaçant d'un réseau de 5000 nœuds à un autre une fois par minute pendant 100 ans, la probabilité de faire une collision est un peu moins d'une sur 1 million. La probabilité de trois collisions successives semble alors relativement faible.

Il serait ainsi raisonnable de décider d'attribuer l'adresse après trois collisions successives, considérant que ces collisions émanent d'une attaque. Cela permettrait d'améliorer

[2] `http://www.docomolabs-usa.com/lab_opensource.html`

[3] `http://namabiiru.hongo.wide.ad.jp/scapy6/`

[4] `http://www.secdev.org/projects/scapy/`

la procédure de détection d'adresse en réduisant les risques pour que cela soit une collision réelle.

Les deux solutions proposées seraient totalement compatible avec les implémentations actuelles. Ainsi, il suffirait de "patcher" progressivement les implémentations afin qu'elles soient insensibles à l'attaque et celles-ci seraient capables de communiquer avec les versions non "patchées" durant la phase de transition.

Coupler les deux solutions proposées augmenterait la robustesse du mécanisme en cas de faille dans le système cryptographique permettant de prévoir les clefs RSA utilisées. Un attaquant connaissant la clef privée ne pourrait plus empêcher un noeud de joindre le réseau mais serait à même d'utiliser la majorité des attaques citées dans le document RFC 3756 [NKN04] (car il aurait la possibilité de signer tous les paquets pour prouver sa "possession de l'adresse"). S'apercevoir de trois collisions successives serait alors un indice permettant de détecter qu'un autre noeud partage la même clef et pourrait déclencher de nouvelles actions tel que la génération d'une nouvelle clef ou un message utilisateur indiquant qu'une attaque est potentiellement en cours sur son adresse.

La solution de désactiver le DAD est envisageable mais elle est à proscrire car elle enlèverait alors tout possibilité de détecter une duplication d'adresse (potentiellement légitime).

7 Conclusion

Cette nouvelle attaque sur la procédure de DAD entraîne des conséquences assez importantes sur la connectivité d'un nœud rejoignant un réseau, rendant ce dernier injoignable. On pourrait imaginer un scénario où un attaquant décide d'interdire l'accès au réseau à tous les nœuds utilisant SEND, en espérant que ceux-ci se connectent sans utiliser SEND, afin de pouvoir ensuite utiliser une des attaques "Man In The Middle" réalisables sur le mécanisme de NDP.

Certaines implémentations ne supportent pas la collision lors d'une procédure de DAD, et s'attribuent alors la première adresse générée, ce qui constitue une parade assez efficace. On pourra dès lors se demander l'intérêt du DAD.

L'Institut National des Standards et de la Technologie américain et le Département de la Défense des États-Unis propose des recommandations (sujettes à des évolutions futures) imposant l'utilisation de SEND afin de sécuriser la découverte de voisins[5]. En parallèle, un groupe de travail est en train d'émerger au sein de l'IETF afin de maintenir et d'améliorer le protocole SEND. Les auteurs ne manqueront pas de faire noter l'attaque présente dans cet article dès que la formation du groupe sera achevée.

Remerciements

Nous remercions Patrick Cegielski et Maryline Maknavicius pour leurs corrections, leurs conseils avisés et leurs propositions pour l'amélioration de cet article.

Références

[AKZN05] J. Arkko, J. Kempf, B. Zill, and P. Nikander. SEcure Neighbor Discovery (SEND). RFC 3971 (Proposed Standard), IETF, March 2005.

[5] `http://www.antd.nist.gov/usgv6/sp500-267-draft2.html`

[Aur05] T. Aura. Cryptographically Generated Addresses (CGA). RFC 3972 (Proposed Standard), IETF, March 2005. Updated by RFCs 4581, 4982.

[CDG06] A. Conta, S. Deering, and M. Gupta. Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification. RFC 4443 (Draft Standard), IETF, March 2006. Updated by RFC 4884.

[DFH99] S. Deering, W. Fenner, and B. Haberman. Multicast Listener Discovery (MLD) for IPv6. RFC 2710 (Proposed Standard), IETF, October 1999. Updated by RFCs 3590, 3810.

[DH98] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard), IETF, December 1998. Updated by RFC 5095.

[GHM+07] V. Gill, J. Heasley, D. Meyer, P. Savola, and C. Pignataro. The Generalized TTL Security Mechanism (GTSM). RFC 5082 (Proposed Standard), IETF, October 2007.

[HD06] R. Hinden and S. Deering. IP Version 6 Addressing Architecture. RFC 4291 (Draft Standard), IETF, February 2006.

[Moo06] N. Moore. Optimistic Duplicate Address Detection (DAD) for IPv6. RFC 4429 (Proposed Standard), IETF, April 2006.

[NKN04] P. Nikander, J. Kempf, and E. Nordmark. IPv6 Neighbor Discovery (ND) Trust Models and Threats. RFC 3756 (Informational), IETF, May 2004.

[NNSS07] T. Narten, E. Nordmark, W. Simpson, and H. Soliman. Neighbor Discovery for IP version 6 (IPv6). RFC 4861 (Draft Standard), IETF, September 2007.

[Plu82] D. Plummer. Ethernet Address Resolution Protocol : Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware. RFC 826 (Standard), IETF, November 1982.

[Sav05] P. Savola. IPv6 Multicast Deployment Issues. Internet-Draft draft-ietf-mboned-ipv6-multicast-issues-02, IETF, February 2005. Work in progress.

[TNJ07] S. Thomson, T. Narten, and T. Jinmei. IPv6 Stateless Address Autoconfiguration. RFC 4862 (Draft Standard), IETF, September 2007.