

Comment fonctionne la faille DNS « Kaminsky » ?

Stéphane Bortzmeyer

<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 31 Juillet 2008

<http://www.bortzmeyer.org/comment-fonctionne-la-faille-kaminsky.html>

La faille de sécurité DNS <<http://www.bortzmeyer.org/faille-dns-empoisonnement.html>> découverte par Dan Kaminsky à l'hiver 2007-2008 et annoncée le 8 juillet 2008 n'a pas encore été officiellement publiée en détail. Cela doit être fait le 8 août à la conférence BlackHat à Las Vegas. Mais, l'Internet étant ce qu'il est, il n'est pas étonnant que cette date n'aie pas été tenue et que les informations aient été publiées avant l'annonce officielle <<http://beezari.livejournal.com/141796.html>>. Trois exploitations de la faille ont déjà été publiées. Plusieurs articles publics ont déjà repris en détail ces informations (voir par exemple, en français, l'article de Sid <<http://sid.rstack.org/blog/index.php/286-la-faille-dns-ou-comment-une-fuite-vous-casse-la-baraque>>). On peut donc désormais expliquer en quoi cette faille consiste.

Je tiens d'abord à rappeler que des tas de bêtises ont été écrites sur l'Internet (et plus encore dans la presse papier ou à la télévision) sur cette faille. Ne vous fiez pas à des messages arrogants d'ignorants complets qui écrivent dans les commentaires d'un blog « La faille est une faiblesse du générateur aléatoire de BIND » ou bien « C'est une faille classique et il n'y a rien de nouveau ».

Plaçons un peu de contexte : le DNS fonctionne typiquement sur UDP (il peut aussi utiliser TCP et l'utilisation de ce protocole réglerait complètement la faille Kaminsky mais beaucoup de serveurs de noms sont incorrectement configurés et refusent les accès TCP). UDP n'a pas la notion de connexion et l'adresse IP de la machine avec laquelle on parle n'est donc pas du tout authentifiée (avec TCP, elle est vérifiée par le processus de connexion, le "*3-way handshake*" - section 3.4 du RFC 793¹ -, ainsi que par les numéros de séquence, qui doivent démarrer d'un nombre imprévisible). Lorsqu'un résolveur (le serveur de noms qui pose une question) interroge le **serveur faisant autorité** (le serveur de noms qui connaît la réponse), il ne peut pas savoir si la réponse vient bien du serveur faisant autorité, elle peut venir d'un méchant qui a usurpé l'adresse IP de celui-ci, et répondu avant lui. Si le méchant peut donc deviner quant une requête va être émise (il existe des techniques pour cela, la plus simple étant de la générer soi-même, si le résolveur est un récursif ouvert <<http://www.bortzmeyer.org/fermer-les-recursifs-ouverts.html>>, ou bien si on dispose d'un zombie autorisé à y accéder),

1. Pour voir le RFC de numéro NNN, <http://www.ietf.org/rfc/rfcNNN.txt>, par exemple <http://www.ietf.org/rfc/rfc793.txt>

cet attaquant peut envoyer une fausse réponse (en bombardant le vrai serveur en même temps, le méchant peut le ralentir pour augmenter les chances de gagner la course).

S'il n'y avait que cela, le DNS serait trivial à abuser. Mais quelques protections existent : la plus importante est que la requête DNS contient un nombre, le "Query ID" (QID, parfois appelé "Transaction ID", TXID). Il est décrit dans la section 4.1.1 du RFC 1035. Le résolveur n'accepte une réponse que si elle contient un "Query ID" identique à celui qu'il a lui-même envoyé. La réponse doit également correspondre à une question actuellement en suspens et être reçue sur le port UDP d'où venait la question. (Il y a très longtemps, les implémentations du DNS étaient naïves et acceptaient à peu près n'importe quoi, ce qui est fini depuis belle lurette.) Malheureusement, le "Query ID" ne fait que 16 bits de long, ce qui est suffisant pour démêler des questions distinctes mais pas assez pour sécuriser la transaction. Envoyer 2¹⁶ soit 65 536 réponses fausses pour qu'au moins une soit acceptée ne prend pas assez de temps, avec l'Internet rapide d'aujourd'hui. (Une vraie solution à la faille Kaminsky et à toutes les failles de la même famille serait de passer le "Query ID" à 128 bits. Mais cela serait un changement du DNS incompatible.)

Cette faiblesse du DNS est connue depuis très longtemps. Par exemple, en 1995, Paul Vixie écrivait <http://www.usenix.org/publications/library/proceedings/security95/vixie.html> << "With only 16 bits worth of query ID and 16 bits worth of UDP port number, it's hard not to be predictable. A determined attacker can try all the numbers in a very short time and can use patterns derived from examination of the freely available BIND code. Even if we had a white noise generator to help randomize our numbers, it's just too easy to try them all." >>. Ce risque (et d'autres) a été documenté par exemple dans le RFC 3833 en 2004. Il n'y a en effet rien de nouveau ici, mais la faille Kaminsky permet d'exploiter cette vulnérabilité avec bien plus d'efficacité qu'une attaque par force brute.

Jusqu'à présent, en effet, cette protection avait quand même suffi, à condition que le "Query ID" soit réellement aléatoire (suivant les recommandations du RFC 4086). Des problèmes étaient survenus avec le serveur DNS de Microsoft (qui n'utilisait que 14 des 16 bits du "Query ID") ou avec BIND (générateur aléatoire bogué, faille CVE-2007-2926 <http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-2926>). Mais ils étaient facilement solubles en mettant à jour le logiciel bogué.

Fin 2007, la situation était donc d'une faiblesse connue (16 bits ne sont pas suffisants) mais peu exploitée car l'attaque n'était pas pratique. Typiquement, si on voulait empoisonner le cache d'un résolveur pour le nom `www.example.com`, il fallait d'abord attendre l'expiration de l'enregistrement stocké (si on a accès au résolveur, c'est facile, puisque le TTL est publié), faire en sorte qu'une question sur `www.example.com` soit générée (là encore, c'est beaucoup plus facile si on a accès au résolveur) et, pendant les quelques dizaines de millisecondes avant que les serveurs faisant autorité répondent, envoyer une bouffée de fausses réponses, en comptant qu'une corresponde. Même aidé par le paradoxe de l'anniversaire, cela ratait plus souvent que cela ne réussissait.

La menace se rapprochait toutefois suffisamment pour avoir poussé l'IETF à s'intéresser au travail de Bert Hubert sur la résistance aux faux. Ce travail est actuellement documenté dans l'"Internet-Draft" "Measures for making DNS more resilient against forged answers" <http://tools.ietf.org/id/draft-ietf-dnsext-forgery-resilience>. Ce document, pas encore approuvé, contient de très intéressants calculs quantitatifs sur l'attaque exposée ci-dessus, évaluant ses chances de succès (trop élevées, hélas) et proposant des mesures pour rendre l'attaque plus difficile, mesures dont la principale est la SPR ("Source Port Randomisation", le fait de rendre le port UDP source imprévisible, le faisant ainsi venir au secours du "Query ID"). Ce travail avançait cahin-caha lorsque la nouvelle de la faille Kaminsky a commencé à circuler en février 2008.

Cette faille est très simple dans son principe. Elle repose sur la faiblesse expliquée ci-dessus (les 16 bits du "Query ID" ne représentent pas suffisamment d'entropie). Mais elle la rend facilement exploitable. Avant Kaminsky, toutes les attaques reposaient sur une question portant sur le nom qu'on voulait

détourner. Si on voulait empoisonner le cache avec une fausse valeur pour `www.example.com`, on faisait poser au résolveur la question « Quelle est l'adresse IP de `www.example.com`? » et on glissait une fausse réponse. Le problème de cette méthode est qu'on n'a droit qu'un un essai : même s'il reçoit plusieurs questions identiques de ses clients, le résolveur n'en envoie qu'une seule et il faut donc deviner le "Query ID" de cette unique question. Pire, si on rate son coup, la bonne réponse, celle qui vient du serveur faisant autorité, va être mise dans le cache du résolveur et, si le TTL a une valeur raisonnable, l'attaquant ne pourra pas réessayer avant plusieurs heures ou même jours.

Dan Kaminsky a innové essentiellement sur un point : si on veut fausser `www.example.com`, on va demander au résolveur des informations sur `quelquechoseNNN.example.com` où NNN est un nombre qui varie à chaque requête. Ainsi, le résolveur devra envoyer autant de requêtes que de noms de domaine différents, augmentant énormément les chances de l'attaquant, surtout s'il est aidé par le paradoxe de l'anniversaire. Notez que, le nom de la zone (ici `example.com`) doit être le même, pour que les requêtes aillent toutes au même serveur faisant autorité mais aussi pour passer les contrôles du paragraphe suivant.

Bon, maintenant, l'attaquant, au lieu de n'avoir qu'une seule chance, en a des milliers et l'attaque devient bien plus facile. Mais quel intérêt pour lui d'arriver à mettre une fausse réponse pour `quelquechose5447.example.com` dans le cache du résolveur? C'était `www.example.com` qui l'intéressait! C'est la deuxième innovation de Kaminsky. Dans la fausse réponse, l'attaquant glisse non seulement un enregistrement pour `quelquechoseNNN.example.com` (qui n'est pas obligatoire, on peut aussi répondre sans section "Answer") mais également des enregistrements pour `www.example.com` dans les autres sections comme "Additional". Les tests qu'effectuent les résolveurs ne suffiront pas à le rejeter puisque ce nom est dans le bailliage (les résolveurs rejettent les noms dits « hors-bailliage », les noms qui sont dans une autre zone que la zone du nom cherché, justement pour limiter les risques d'empoisonnement).

Si jamais le résolveur devenait d'avantage paranoïaque, il suffirait d'ailleurs de glisser de faux enregistrements NS ("Name Server") et une fausse colle dans la section "Authority", que le résolveur ne peut pas ignorer. C'est ce que fait une des exploitations, et voici la réponse DNS fausse, telle que vue par Wireshark. Ici, on a tenté de tricher sur l'adresse IP de `www.example.com`, un domaine dont les serveurs faisant autorité sont à l'IANA (ici, `193.0.0.236`). On tente d'empoisonner le cache avec l'adresse `192.0.2.1`:

```
...
  Source: 193.0.0.236 (193.0.0.236)
  Destination: 192.0.2.253 (192.0.2.253)
User Datagram Protocol, Src Port: 53 (53), Dst Port: 49186 (49186)
  Source port: 53 (53)
  Destination port: 49186 (49186)
  Length: 175
  Checksum: 0x98a1 [correct]
    [Good Checksum: True]
    [Bad Checksum: False]
Domain Name System (response)
  Transaction ID: 0x0001
  Flags: 0x8400 (Standard query response, No error)
    1... .. = Response: Message is a response
    .000 0... .. = Opcode: Standard query (0)
    .... .1.. .. = Authoritative: Server is an authority for domain
    .... ..0. .... = Truncated: Message is not truncated
    .... ...0 .... = Recursion desired: Don't do query recursively
    .... .... 0... .. = Recursion available: Server can't do recursive queries
    .... .... .0.. .. = Z: reserved (0)
    .... .... ..0. .... = Answer authenticated: Answer/authority portion was not authenticated by the server
    .... .... .... 0000 = Reply code: No error (0)
Questions: 1
Answer RRs: 1
Authority RRs: 1
```

<http://www.bortzmeyer.org/comment-fonctionne-la-faille-kaminsky.html>

```
Additional RRs: 1
Queries
  deb623f600000009.example.com: type A, class IN
    Name: deb623f600000009.example.com
    Type: A (Host address)
    Class: IN (0x0001)
Answers
  deb623f600000009.example.com: type A, class IN, addr 1.1.1.1
    Name: deb623f600000009.example.com
    Type: A (Host address)
    Class: IN (0x0001)
    Time to live: 1 day
    Data length: 4
    Addr: 1.1.1.1
Authoritative nameservers
  example.com: type NS, class IN, ns www.example.com
    Name: example.com
    Type: NS (Authoritative name server)
    Class: IN (0x0001)
    Time to live: 1 day
    Data length: 20
    Name server: www.example.com
Additional records
  www.example.com: type A, class IN, addr 192.0.2.1
    Name: www.example.com
    Type: A (Host address)
    Class: IN (0x0001)
    Time to live: 1 day
    Data length: 4
    Addr: 192.0.2.1
```

L'attaque Kaminsky est donc bien nouvelle. Elle passe d'une méthode laborieuse et ayant de faibles chances de succès à une méthode qui marche à presque tous les coups et en très peu de temps (moins de dix minutes avec les exploitations publiées, beaucoup moins avec certaines non publiées).

Voici pourquoi il est donc urgent de mettre à jour, si ce n'est pas encore fait, les résolveurs DNS dont vous avez la responsabilité. À part PowerDNS et Unbound, qui le mettaient déjà en œuvre, cette mise à jour installe la SPR (rendre le port source aléatoire) sur les résolveurs. On passe ainsi de seulement 16 bits d'entropie (le "Query ID") à 32 (le "Query ID" plus le port source, qui est également sur 16 bits). Cela retarde le moment où l'attaque sera efficace, en attendant des mesures plus radicales.