FAILLES INTRINSÈQUES DU PROTOCOLE DNS

▷ Pierre BETOUIN (betouin@esiea.fr)*

20 octobre 2003

DNSA - DNS Auditing tool

`http://securitech.homeunix.org/dnsa/dnsa-current.tar.gz`

Table des matières

*Etudiant à l'Ecole Supérieure d'informatique, d'électronique et d'automatique.
Responsable *Challenge-SecuriTech* (http ://www.challenge-securitech.com)

1 INTRODUCTION

DNS (Domain Naming System) est, de par son utilité, devenu le protocole numéro 2 d'Internet, derrière OSPF (Open Shortest Path First), principal protocole de routage utilisé. Son rôle central sur le réseau fait de lui une pièce maîtresse de la sécurité, mais bon nombre d'utilisateurs, ou de professionnels, ont tendance à négliger son importance au sein d'un réseau.

Le but de DNS étant de faciliter la communication (qui pourrait retenir autant d'IP que de noms ?), sa configuration de base, en tant que client, est triviale : une simple ligne de configuration dans /etc/resolv.conf, voire la plupart du temps, un paramètre par défaut dans une configuration DHCP... C'est pour cela que la plupart des gens ne s'intéresse pas de plus près à ce protocole, ni aux moyens disponibles pour le sécuriser. Mais derrière le fameux *gethostbyname()* se cachent plusieurs mécanismes...

La plupart des échanges DNS est effectuée en UDP sur le port 53. Le port 53 TCP est lui aussi en écoute sur les serveurs DNS, mais il gère uniquement les requêtes longues, les transferts de zones, etc. L'avantage d'UDP dans ce type de requêtes est sa rapidité. Même s'il ne garantit pas l'arrivée des informations, il se révèle être très rapide. Les requêtes DNS étant très nombreuses, il aurait été très préjudiciable d'utiliser TCP, surtout qu'une perte de paquet DNS n'est pas nuisible, car les clients réitèrent leurs requêtes à plusieurs reprises si aucune réponse n'est donnée (dans la limite du "timeout" bien sûr).

Côté client, DNS se base sur des requêtes de type "A" pour résoudre un nom pleinement qualifié (FQDN) en adresse IP. L'opération inverse peut être faite avec les "requêtes pointeurs" de type "PTR" : la résolution inverse. Le domaine spécifique contenant toutes les adresses IP est alors in-addr.arpa.

Côté serveur, les demandes sont reçues, et traitées en fonction du cache :

- **Si la demande a déjà été effectuée depuis les dernières n secondes (délai fixé dans le payload DNS par le serveur de la zone contactée), le serveur répond directement avec l'information qu'il avait obtenue.**
- **Si le nom n'est pas connu, alors le serveur lance une requête (récursive ou non) sur le serveur DNS de la zone en question afin d'obtenir l'IP correspondant au FQDN communiqué. L'information est alors stocké en cache.**

La seule authentification de DNS est basée sur le DNS ID (champ de 2 octets dans le paquet DNS). Si IPv4 est utilisé, alors toute la sécurité du protocole repose uniquement sur une "clef" de 16 bits ! (seulement 65535 possibilités) Cette seule considération peut laisser songeur quand on connaît les temps de calculs des machines actuelles, ou la vitesse grandissante des liaisons...

Quand le client fait une requête, il choisit aléatoirement - dans la mesure de l'aléa informatique - le DNS ID à mettre dans le paquet. Quand le serveur DNS reçoit la demande, il la traite, et répond avec ce même

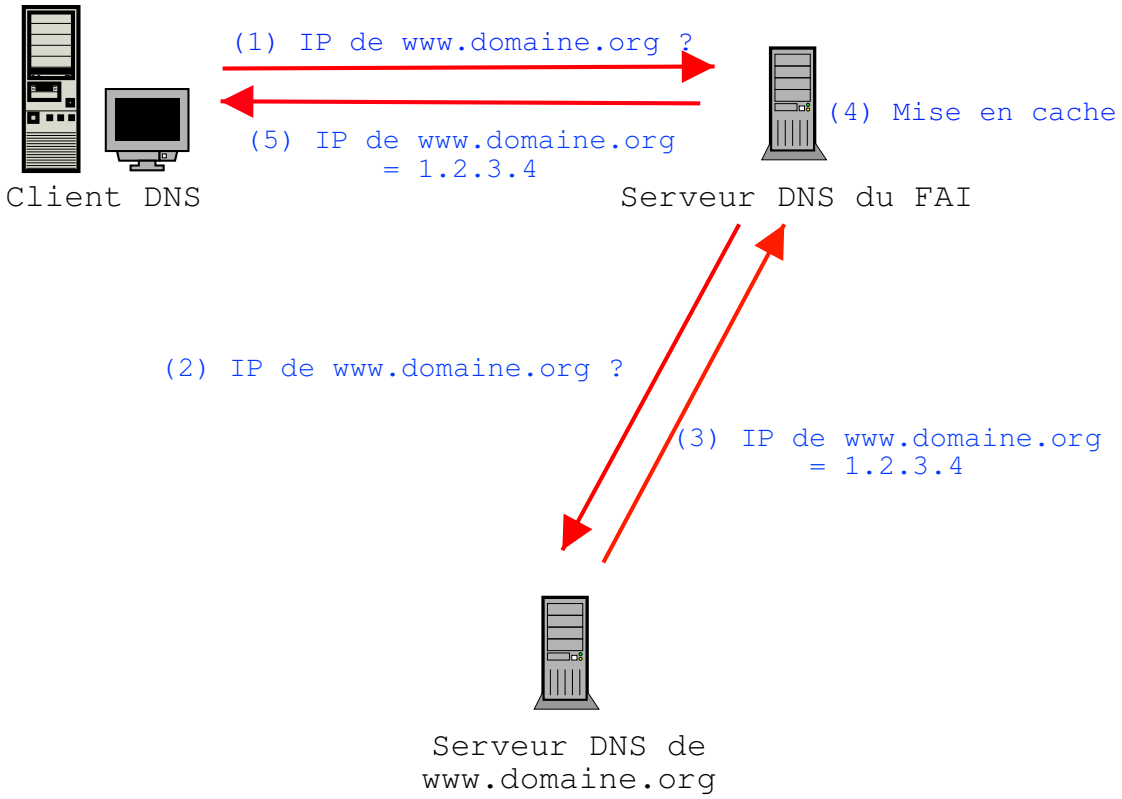


FIG. 1 – Requête sur un serveur DNS n'ayant pas l'information en cache

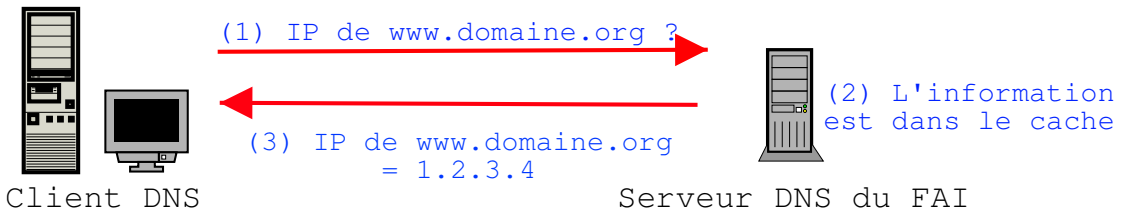


FIG. 2 – Requête sur un serveur DNS ayant l'information en cache

DNS ID. Si le DNS ID de la requête correspond effectivement au DNS ID de la demande, alors la réponse est prise en considération. Et quand un serveur ne possède pas l'adresse IP d'une machine dans son cache, il se comporte alors comme un client en faisant une requête sur le serveur DNS de la zone concernée, pour ensuite répondre à l'initiateur de la demande.

Nous commençons déjà à repérer certaines faiblesses évidentes de ce protocole... Evidemment, il est possible de palier ces problèmes, cet article vous aidera à y voir plus clair.
Vous pourrez consulter [4] pour une compréhension approfondie de cet article.

Dump réseau d'un ping vers www.yahoo.fr :

```
1> [root@linbox]\# tcpdump -i eth0 udp and port 53
2> tcpdump: listening on eth0
3> linbox.32783 > ns1.fai.fr.domain: 30679+ A? www.yahoo.fr. (30) (DF)
4> ns1.fai.fr.domain > linbox.32783: 30679 2/5/1 CNAME[|domain] (DF)
5> ns1.fai.fr.domain > linbox.32784: 25190* 1/4/4 (224) (DF)
6> linbox.32785 > ns1.fai.fr.domain: 30680+ PTR? 11.3.12.217.in-addr.arpa. (42) (DF)
```

- **Ligne 3 : Demande l'IP de l'hôte www.yahoo.fr (requête de type "A")**
- **Ligne 4 : Réponse du DNS. Il s'agit d'un enregistrement CNAME (alias)**
- **Ligne 6 : Requête inverse pour l'hôte 217.12.3.11 (www.yahoo.fr)**

2 ATTAQUES COMMUNES AUX CLIENTS/SERVEURS

Comme nous l'avons vu précédemment, les serveurs DNS se comportent comme des clients quand ils ne possèdent pas l'information en cache. Cette particularité nous laisse percevoir que les faiblesses du protocole, côté client, peuvent directement se répercuter côté serveur... Toute la sécurité est donc basée sur un "short" de 2 octets : le DNS ID. Il est alors possible d'envisager des attaques de "DNS ID Spoofing". Cette attaque peut être menée de 2 façons distinctes, selon les informations dont nous disposons :

2.1 *DNS ID* connu

S'il est possible de "capter" le trafic réseau ("sniffer"), alors nous sommes en mesure de connaître ce fameux champ, et il nous suffit alors de forger un paquet UDP, avec comme IP source celle du serveur, et comme IP de destination, celle de la machine ayant fait la requête.
Bien évidemment, le DNS ID sera celui récupéré sur le réseau.

Le client recevra la requête et la traitera sans pouvoir faire de distinction entre la "vraie" de la "fausse". Le seul challenge est de répondre avant le serveur DNS légitime car seule la première requête est considérée par le client. Ceci est trivial quand on se place sur un LAN, et que le serveur DNS est en dehors du réseau. Ce cas ne figure est analogue aux numéros de séquence TCP.

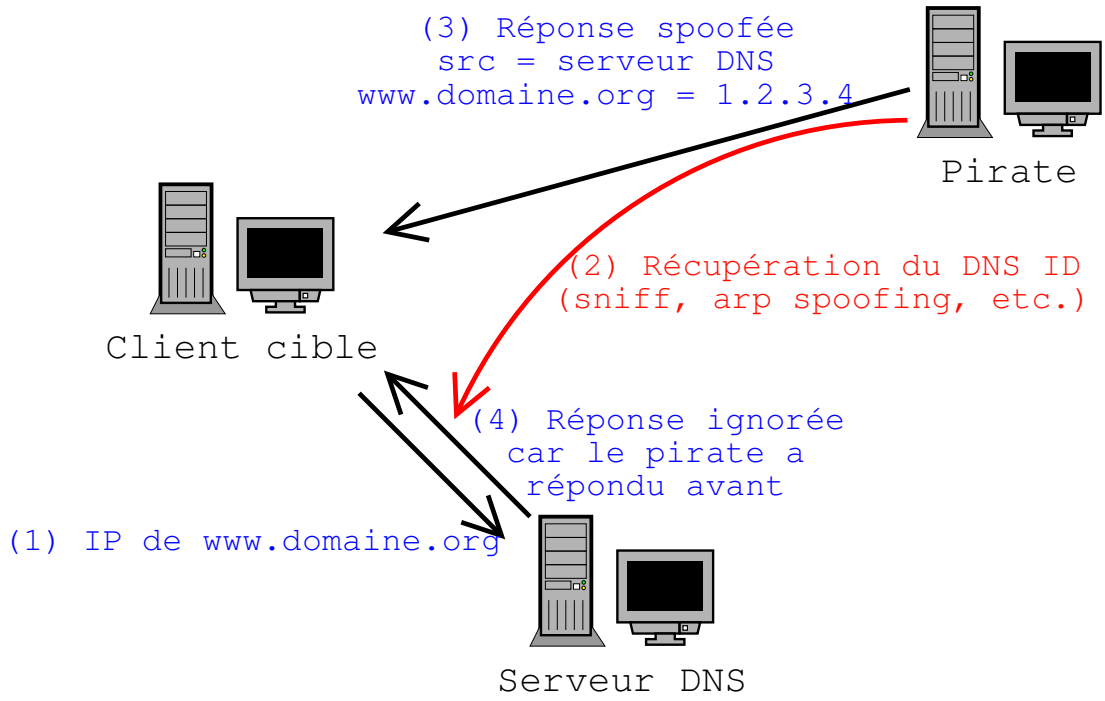


FIG. 3 – Attaque DNS ID Spoofing connaissant l'ID

2.2 *DNS ID* inconnu

Si nous ne pouvons connaître le DNS ID, plusieurs solutions sont envisageables :

- **Attaque par bruteforce.**
  Il faut alors disposer d'une connexion très rapide, et s'être levé chanceux ! Celle-ci est possible mais en général très peu concluante...
- **Attaque par implémentation.**
  Certains OS (comme les Windows 9x de base par exemple), incrémentent le DNS ID à chaque requête. Il est alors très simple de pouvoir le "deviner", uniquement grâce à un sniffer, en faisant effectuer à la victime une requête sur un serveur DNS sous contrôle.
- **Attaque par prédiction.**
  Les générateurs d'aléas ne sont pas, comme vous le savez sûrement, vraiment aléatoires. Ils fonctionnent grâce à un ensemble de facteurs afin de générer les nombres pseudo-aléatoires les plus fiables possibles. Encore une fois, selon l'implémentation, certaines attaques sont plus ou moins faciles à mener (cf article de *Joe STEWART*, "DNS cache poisoning - The next generation"[3]). Une attaque probabiliste (telle que la "birthday attack", cf `http://www.x5.net/faqs/crypto/q95.html`) donne souvent de bons résultats.

3 ATTAQUES SUR LES SERVEURS

Comme nous l'avons vu, la plupart des attaques sur les clients est donc exploitable sur les serveurs. L'inverse n'est pas vrai, et, hormis certaines attaques liées à un bug propre du serveur (overflow, string format, etc.), certaines attaques de "fond" restent possibles. La plus

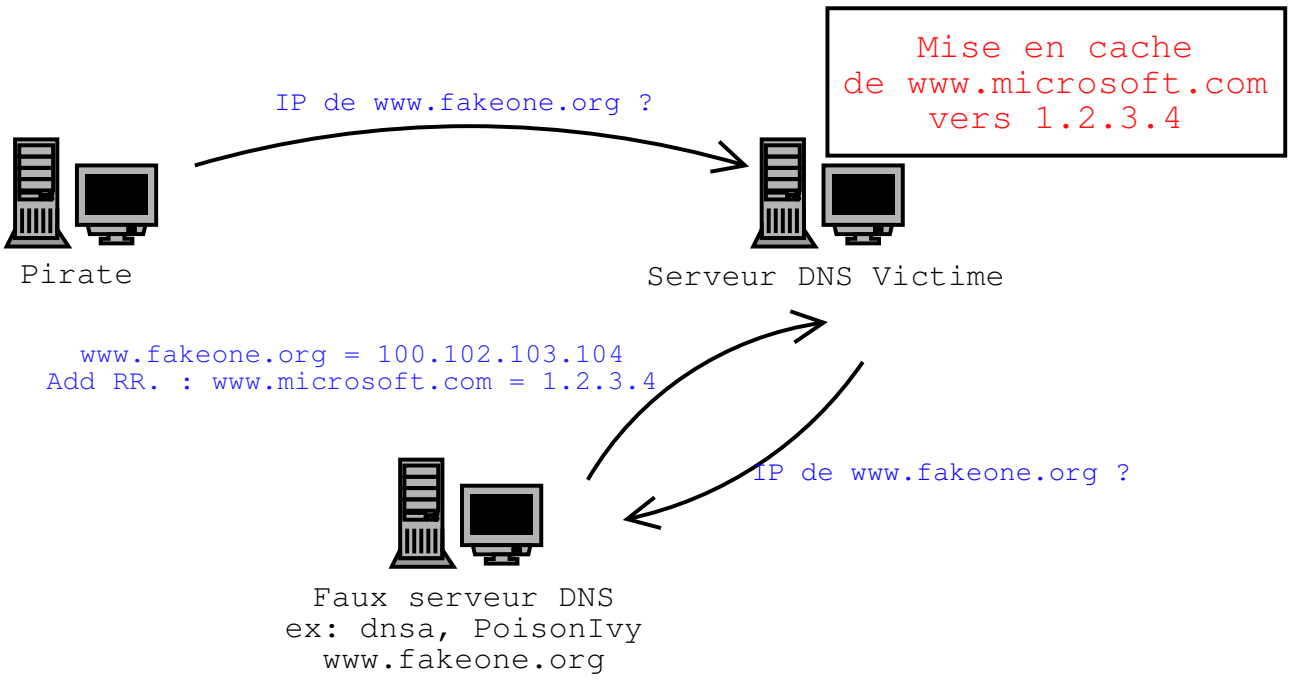
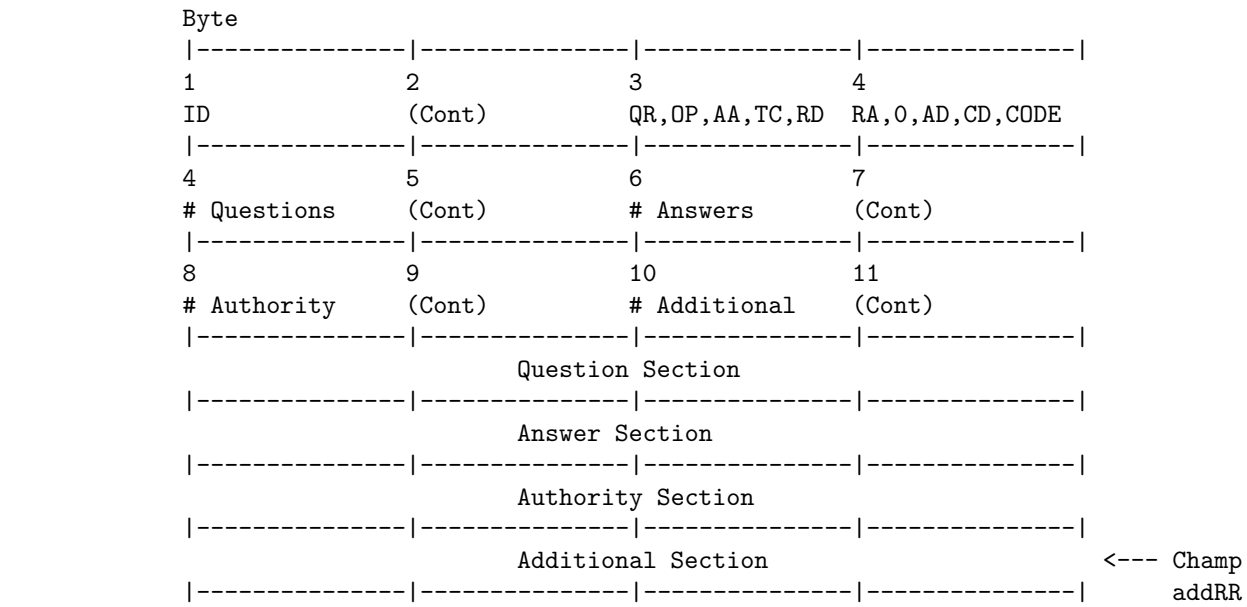


FIG. 4 – Attaque DNS cache poisoning

connue est le "DNS cache poisoning" qui consiste à corrompre le cache d'un serveur DNS. De ce fait, toutes les requêtes ultérieures qui y seront faites, pointeront vers une machine illégitime.
Cette attaque peut être effectuée de plusieurs façons :

- **Problème d'interprétation du champ "additional record" qui permet à un serveur interrogé, de répondre avec plus d'enregistrements que demandé.**
  Prenons l'exemple d'un utilisateur qui demande l'IP de www.domaine.com. Le serveur DNS peut lui donner d'autres adresses comme mail.domaine.com, dns2.domaine.com, etc. Ce champ a été implémenté afin de compléter une requête sur un même domaine. En revanche, ces informations ne devraient pas concerner de domaine extérieur. C'est cette dernière considération qui n'a pas été prise en compte sur de nombreux serveurs (les serveurs DNS "Windows 2000 server" par exemple, d'anciennes versions de Bind...) Cette attaque est triviale. Il suffit simplement d'avoir la main sur un serveur DNS public, pour faire tourner un faux serveur DNS qui permettra de rajouter un champ "additional record".
  ***Poison Ivy*[2] ou *dnsa* permettent de tester cette attaque. Il faut alors faire une requête sur le serveur DNS victime afin de demander une information relative au domaine "pirate.org". Celui-ci renseignera le serveur DNS victime avec l'information, mais également un champ additionnel.**

D'après [7], le payload DNS d'un paquet UDP sur le port 53 est le suivant :



- **Une attaque par DNS ID Spoofing est également possible quand le serveur ne possède pas l'adresse IP correspondante à l'adresse demandée.**
  Le schéma d'une telle exploitation est le même que pour un client.
  Cf page 4 pour plus d'informations.

4 *DNSA* : DNS AUDITING TOOL

Avant d'en dire plus à propos de l'outil d'audit *dnsa* (*DNS auditing tool*), en voici l'usage :

```
$ ./dnsa
Usage: ./dnsa [OPTIONS]
Swiss knife tool for DNS auditing.

-1      DNS ID spoofing         [ Required : -S ]
        -D [www.domain.org]     Hostname query to fool.
                                Don't use it if every DNS request sniffed has to be spoofed
        -S [IP]                 IP address to send for DNS queries
        -s [IP]                 IP address of the host to fool
        -i [interface]          IP address to send for DNS queries

-2      DNS IDs Sniffing        [ Required : -s ] (Beta and not finished)
        -s [IP]                 IP address of the server which makes queries
        -w [file]               Output file for DNS IDs

-3      DNS cache poisoning     [ Required : -S AND -a AND -b]
        -a [host.domain.org]    Hostname to send in the additional record
        -b [IP]                 IP to send in the additional record
        -D [www.domain.org]     Hostname for query. Use it if you want to fool just on
```

```
        -S [IP]                 IP address to send for DNS queries (the normal one)
        -s [IP]                 IP address of the server to fool
        -i [interface]          IP address to send for DNS queries

-h      Print usage
                Bug reports to <soulrider@ifrance.com>
```

***dnsa* est un outil d'audit DNS écrit en C. Il est, pour l'instant, de meilleure utilité sur un LAN mais reste utilisable sur Internet, particulièrement pour le *fingerprinting DNS* (cf page 10).**

- **Mode *DNS ID Spoofing***
  Ce mode permet d'exploiter les failles citées au chapitre 2. Dans le périmètre d'un réseau Ethernet, il permet de récupérer des *DNS ID* émis, et de répondre avant le serveur DNS légitime.
  Plusieurs filtres sont disponibles, ils peuvent être cumulés : filtre sur la requête (pattern matching), sur la source (hôte du réseau), et sur l'interface en écoute.
  Un paramètre est cependant nécessaire : l'adresse IP à envoyer sur la machine cible. Par exemple, pour spoofer uniquement les requêtes provenant de l'hôte *192.168.0.10* vers le domaine *cible.com*, sur l'interface *eth0* avec l'adresse 192.168.0.200, il suffit de lancer :
  $ ./dnsa -1 -D cible.com -S 192.168.0.200 -s 192.168.0.10 -i eth0
  1. **Si le domaine n'est pas renseigné (paramètre *-D*), *dnsa* spoofera toutes les requêtes provenant de l'hôte 192.168.0.10.**
  2. **Si l'IP source des requêtes n'est pas donnée, alors toutes les requêtes interceptées seront traitées.**
  3. **Même comportement pour l'interface (paramètre *-i*).**
- **Mode *DNS ID Sniffing***
  Ce mode permet d'effectuer divers tests. En effet, il peut être pratique d'analyser les distributions de *DNS ID* d'un serveur en particulier. Le filtre *-s* permet de filtrer l'adresse d'un serveur DNS en particulier. Vous pouvez lancer un *ping* vers un FQDN (en désactivant le cache du *resolver*) pour récupérer un maximum de *DNS ID*.
  Il permet de constater des distributions non aléatoires, ou de pouvoir tracer les *DNS ID* avec GnuPlot[11] à l'aide des commandes suivantes :

  ***s*et polar**
  plot "fichier_sortie" using 1 :1 title 'DNS ID'
  pause 2 # Permet d'automatiser l'affichage à l'aide d'un script

- **Mode *DNS Cache poisoning***
  Il s'agit d'une adaptation de *Poison Ivy*[2] en C. Ce mode exploite une faille encore trop souvent rencontrée en jouant sur le champ *additional record* du payload DNS (cf page 6).
  La syntaxe est la suivante :

```
$ ./dnsa -3 -D the_host_IP_which_is_asked_for \
  -S normal_host_IP  \
  -s DNS_server_which_is_doing_the_request  \
  -a host_in_additional_record  \
  -b IP_in_the_additional_record \
```

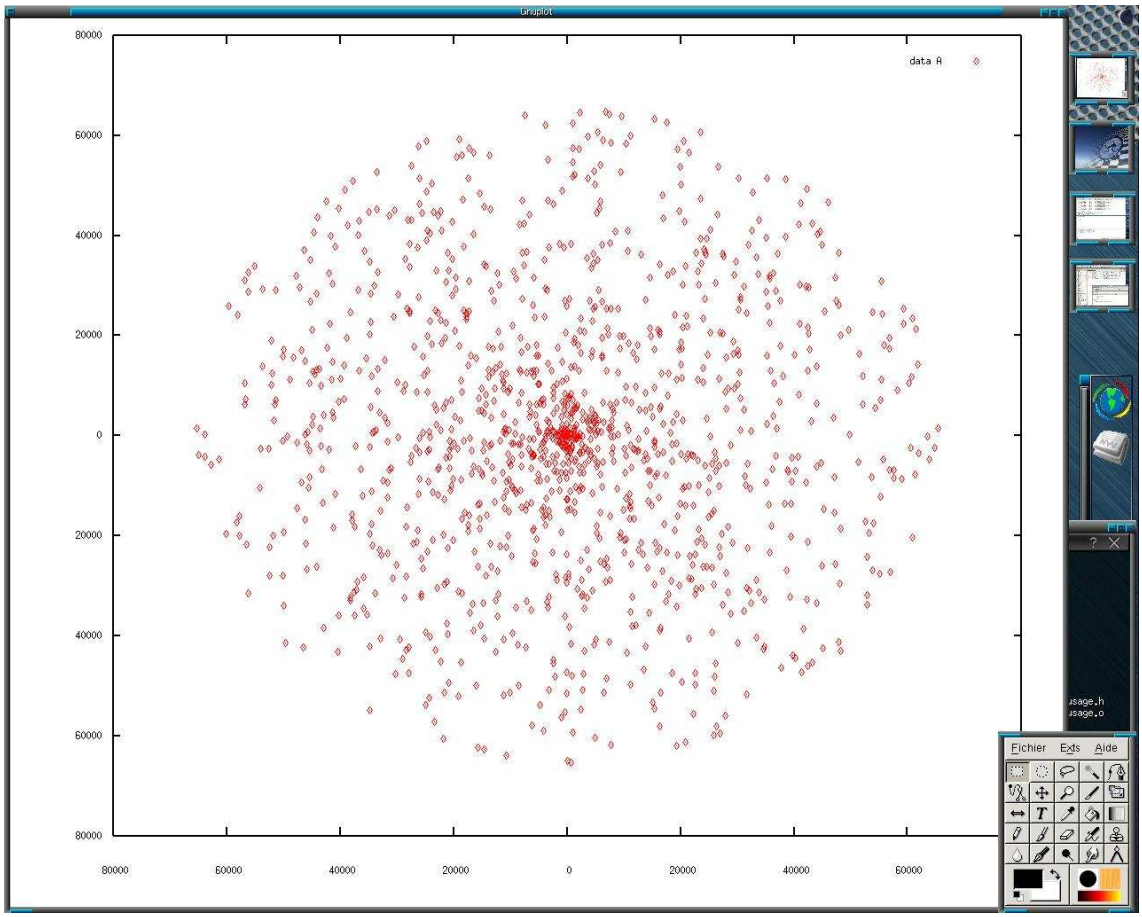


FIG. 5 – Exemple de distribution de *DNS ID* sur FreeBSD 5

`-i INTERFACE`

La ligne de commande parle d'elle même... Pour plus d'informations, consultez le fichier *EXAMPLE* dans le répertoire *doc/* de *dnsa*.

Sur les serveurs faillibles, la nouvelle entrée *host_in_additional_record* sera associée à l'IP *IP_in_the_additional_record*.

Bien entendu, dans le cas d'un réseau switché, l'utilisation de *dnsa* peut être combinée avec *arp-sk*[12], outil permettant de faire de l'*ARP cache poisoning*.
***iptables*[13] peut alors être utilisé pour effectuer un MiM (Man In the Middle)...**

5 FINGERPRINTING DE DNS

Par défaut, un serveur DNS nous renvoie sa version. Il est alors simple de la connaître :

```
$ nslookup
> server dns.domaine.org
> set class=chaos
> set q=txt
> version.bind
Server:  dns.domaine.org
Address:  11.22.11.22

VERSION.BIND    text = "9.2.2P3"
```

Mais désormais, beaucoup d'entre eux cachent la version pour des raisons évidentes de sécurité. Si une faille venait à sortir, le serveur pourrait alors en faire les frais rapidement, et notamment avec des vers ou des *scripts kiddies* qui examinent les bannières avant d'attaquer.

Nous avons vu que la distribution des *DNS ID* était pseudo-aléatoire, on pourrait donc, d'après un échantillon de plusieurs d'entre eux, connaître la version du serveur en question.

C'est dans cette optique que *dnsa* compte implémenter le *fingerprinting DNS*. Mais pas uniquement, car cette technique est longue à appliquer et pourrait passer à côté de comportements évidents d'implémentation.
La première étape de la reconnaissance consistera donc en de multiples envois de paquets falsifiés (les champs du payload y seront forgés pour contenir des informations "incorrectes").
Un *fuzzy test* permettra alors de dégrossir le panel des différents serveurs, puis l'analyse des *DNS ID* viendra affiner les résultats.
Des fichiers de signatures seront chargés, tout comme les fichiers de signatures d'*nmap*, ce qui rendra l'analyse évolutive.

6 VERS UNE SECURISATION DE DNS

Pour que le trafic DNS soit sécurisé, plusieurs techniques peuvent être mises en oeuvre. Des règles de firewall, en passant par *IPSec* (voire *IPv6*), ou une migration vers DNSSec, les possibilités de sécurisation autour du protocole DNS sont nombreuses.

Il est évident que, dans un premier temps, une solide configuration du serveur DNS permet d'éviter 90% des fuites d'informations (transferts

de zones, bannières trop bavardes, etc.). Mais cela ne suffit pas comme nous avons pu le voir car le "DNS Spoofing", par exemple, est une faille intrinsèque de DNS. En effet, UDP n'ayant pas l'équivalent des numéros de séquences de TCP, la seule garantie est la confidentialité du *DNS ID* qui transite sur le réseau.
Nous constatons que sécuriser DNS n'est pas simple, car il s'agit en fait de sécuriser l'ensemble du réseau.

- Certaines règles de base sont indispensables sur un firewall, ou sur un client DNS, lorsque la sécurité doit être optimale. Par exemple, le trafic DNS (ports UDP et TCP 53) doivent être uniquement "réservés" à/aux serveur(s) DNS utilisés.
- L'adresse MAC du serveur DNS doit être entré en "dur" si possible, ce qui permet d'éviter des attaques via le protocole ARP.
  Ceci peut être fait simplement via la commande *arp -s nom_d_hôte hw_addr*.
- *IPSec* peut être déployé entre les machines, ce qui garantit l'intégrité des échanges IP, et annihile donc tout risque de spoofing.
  Cependant, cette méthode est très lourde pour un protocole comme DNS. Nous verrons que *DNSSEC* peut résoudre en partie ce problème.
  Les serveurs très sollicités, tels que les serveurs "root", voire les serveurs des fournisseurs d'accès (FAI), auraient beaucoup de mal à supporter cette charge...
- *IPv6* intègre directement *IPSec*, il permet donc également de fiabiliser les échanges de faon native.
- *DNSSEC* (DNS Security extensions) est un ensemble d'extensions qui permet de sécuriser le protocole DNS au niveau de l'authentification, comme de l'intégrité des données. Les échanges sont sécurisés via des algorithmes cryptographiques symétriques ou asymétriques.
  TSIG ("Transaction Signature") permet principalement d'authentifier les DNS primaires/secondaires lors des transferts de zones grâce à des méthodes cryptographiques (échanges de clefs), ainsi qu'une signature (la plupart du temps un haché *MD5*).
  DNSSEC ne modifie pas l'en-tête du paquet DNS. Il introduit 4 nouveaux *RR* (*Ressource records*) : KEY (clef publique d'une zone), SIG (informations diverses, notamment temporelles), NXT (RR certifiant l'inexistence d'un FQDN) et DS (permet d'établir des relations de confiance).
  DNSSEC est d'ores et déjà experimenté sur les zones "com", "fr", "kr", "jp", "nl", et "se".
  A la différence de *TSIG* qui est disponible sur de grandes echelles, *DNSSEC* ne peut être implémenté que dans des zones confinées.

7 CONCLUSION

Le protocole DNS n'est pas á négliger dans une infrastructure, aussi petite soit elle. Sa sécurité dépend directement de son environnement et de son implémentation. La compromission d'un serveur DNS peut donc mener à la compromission totale d'un réseau, dans un temps très court, avec peu de moyens.
Imaginez un instant qu'un pirate puisse modifier le cache du serveur DNS de votre FAI pour modifier l'adresse du site de votre banque en ligne. Assez facilement (faux certificats, etc), il pourrait réussir à simuler le clone du site web de votre banque... sauf que le serveur distant serait

le sien, et qu'il pourrait récupérer des données sensibles.
Maintenant, imaginez qu'un employé, en passe de se faire licencier, décide de corrompre le cache du serveur DNS de son entreprise. Il pourrait, à souhait, récupérer des mots de passe en tout genre (webmails, serveurs ftp, comptes, etc.), des clefs privées (ssh, imaps, etc.), lancer des MiM (*Man In the Middle*) en forwardant les requêtes reçues, etc.

dnsa permet de tester rapidement le niveau de sécurité des installations. Ce projet est encore en développement : les fonctions de fingerprinting DNS devraient voir le jour dans les prochaines semaines. Son fonctionnement est assez proche du projet p0f.[15]

Toutes les contributions au projet sont les bienvenues : développement, traductions, etc.

8 REFERENCES

Références

[1] **RFC utiles sur DNS.**
http://www.ietf.org/rfc/rfc1035.txt

[2] **Poison Ivy (*VALGASU*)**
Pseudo serveur DNS écrit en perl permettant de rajouter des champs additionnels dans les paquets DNS.
http://valgasu.rstack.org

[3] **"DNS cache poisoning - The next generation" (*Joe STEWART*)**
http://www.securityfocus.com/guest/17905

[4] **"Exploitation malicieuse du protocole DNS" (*Daniel POLOMBO*, Eric DETOISIEN)**
MISC 4.

[5] **Libpcap**
http://www.tcpdump.org

[6] **Libnet**
http//www.packetfactory.net/libnet/

[7] **RFC relatives à cet article :**
- **RFC #1035 : Domain Names - Implementation and Specification**
- **RFC #1536 : Common DNS Implementation Errors and Suggested Fixes**
- **RFC #1912 : Common DNS Operational and Configuration Errors**
- **RFC #1995 : Incremental Zone Transfer in DNS (IXFR)**
- **RFC #2535 : Domain Name System Security Extensions**

[8] **"DNS ID Hacking (and even more !)" (*ADM TEAM*)**
http://www.freelsd.net/ADM/ADMID.txt

[9] **"Coding with the DNS protocol" (*Fux0r*)**
https://fux0r.phathookups.com/programming-tutorials/coding-with-the-dns-protocol.txt

[10] **"DNS Hacks"**
http://www.packetfactory.net/DNS/

[11] **GnuPlot**
http://gnuplot.info

[12] **arp-sk**
http://www.arp-sk.org

[13] **iptables**
http://www.netfilter.org

[14] **Zodiac DNS auditing tool (*TESO TEAM*)**
http://www.packetfactory.net/projects/zodiac/

[15] **p0f (*Michal Zalewski*)**
http://freshmeat.net/redir/p0f/43121/url_homepage/p0f.shtml
Fingerprinting passif sur l'étude des numéros de séquences.

9 REMERCIEMENTS

– ***Eric DETOISIEN*, pour ses conseils, et pour ses codes sources de *Poison Ivy*.**
– ***Frédéric RAYNAL*, pour avoir été un très bon tuteur de stage ! ;)**
– ***Cyril ROBERT*, d'avoir accepté de tester les différentes versions de *dnsa*, ainsi que de les installer sur un maximum de distributions possibles.**
– ***Mike SCHIFFMAN*** (http://www.packetfactory.net), **de croire en *dnsa*.**