Le 28 novembre 2008

Faille(s) DNS

Les petits tracas d'un grand protocole

Clément Collin – Elias Yegdjong

RICM3, Polytech'Grenoble

Plan

1. Introduction
2. Failles du protocole DNS
3. La faille de Kaminsky (+ démo)
4. Des mesures de protection
5. Conclusion

1. Introduction

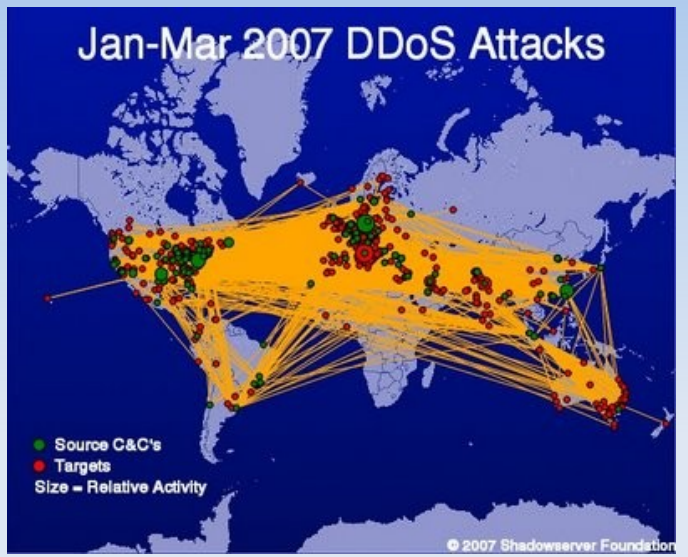
1. **Introduction**
2. Failles du protocole DNS
3. La faille de Kaminsky (+ démo)
4. Des mesures de protection
5. Conclusion

1. Introduction

- DNS : LE service de nommage d'Internet
  - Principes : récursif, itératif... Vous connaissez !
  - Sans DNS, le Web s'écroule
- Juillet 2008 : le buzz de Kaminsky
  - Un bug dans 90% des serveurs DNS
  - Panique à bord !
- Objectifs de la présentation
  - Ludique (titiller un protocole vu en cours)
  - Didactique (dangers et mesures de sécurité)

1. Introduction

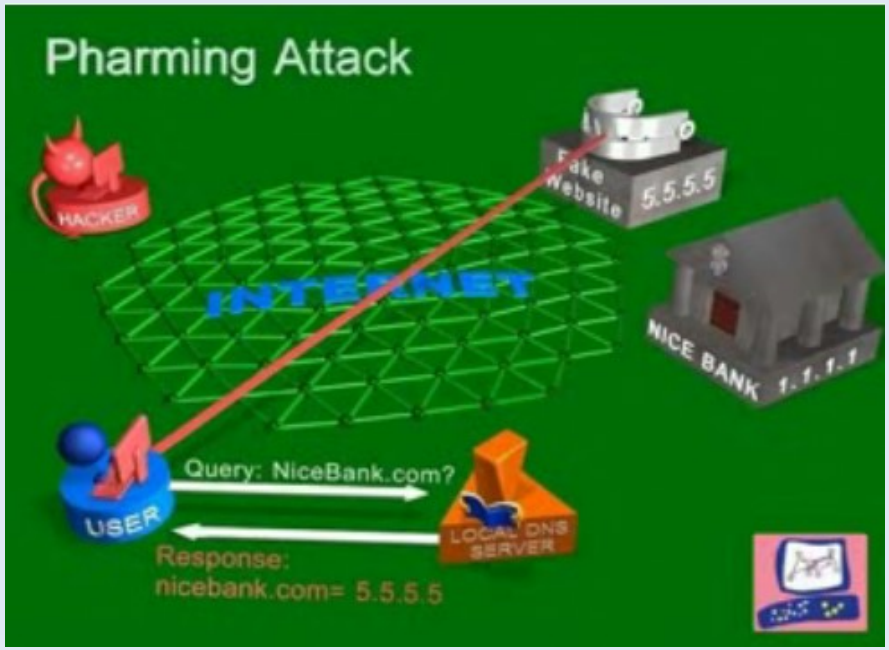
- Ce dont on ne parlera pas :
  - Attaques par DoS et DDoS
    - Objectif : **faire tomber le système**
    - Dernière grosse attaque en date : 11-2007 (les 13 serveurs racines pendant 12h, Corée du Sud suspectée)



  - Tunnels DNS
    - Objectif : **camoufler du trafic malfaisant**
    - La piraterie génère beaucoup de trafic qui peut être détecté
    - Les paquets DNS n'attirent pas l'attention

1. Introduction

- Ce dont on parlera :
  - Spoofing + Cache Poisoning
    - *To spoof* = parodier, faire une blague
    - *Cache Poisoning* = empoisonnement de cache
    - But : **rediriger du trafic** à l'insu de l'utilisateur (*pharming*)
    - Objectifs indirects :
      - *phishing*
      - *sniffing*
      - propagation de virus
      - installation de backdoors
      - surveillance de mails
      - ...



© CheckPoint
http://www.checkpoint.com/defense/advisories/public/dnsvideo/index.ht

2. Failles du protocole DNS

1. Introduction
2. **Failles du protocole DNS**
    1. **Fonctionnement du protocole DNS**
    2. **Les failles intrinsèques du DNS**
    3. **Exploitations malicieuses des failles**
3. La faille de Kaminsky (+ démo)
4. Des mesures de protection
5. Conclusion

2.1 Fonctionnement du protocole DNS

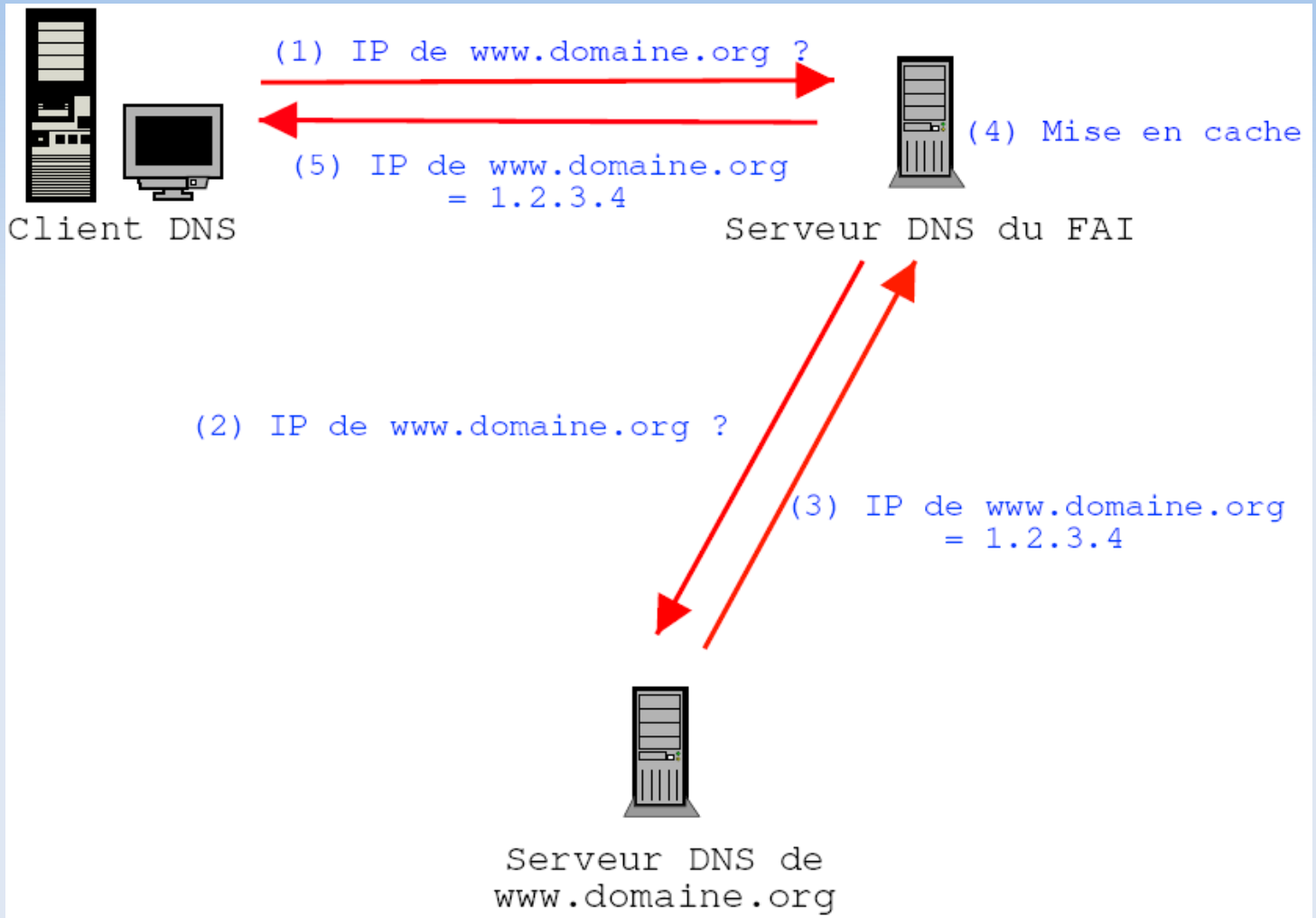


FIG. 1 – Requête sur un serveur DNS n'ayant pas l'information en cache

Source : http://www.packetfactory.net/projects/dnsa/ArticleDNS.pdf

2.1 Fonctionnement du protocole DNS

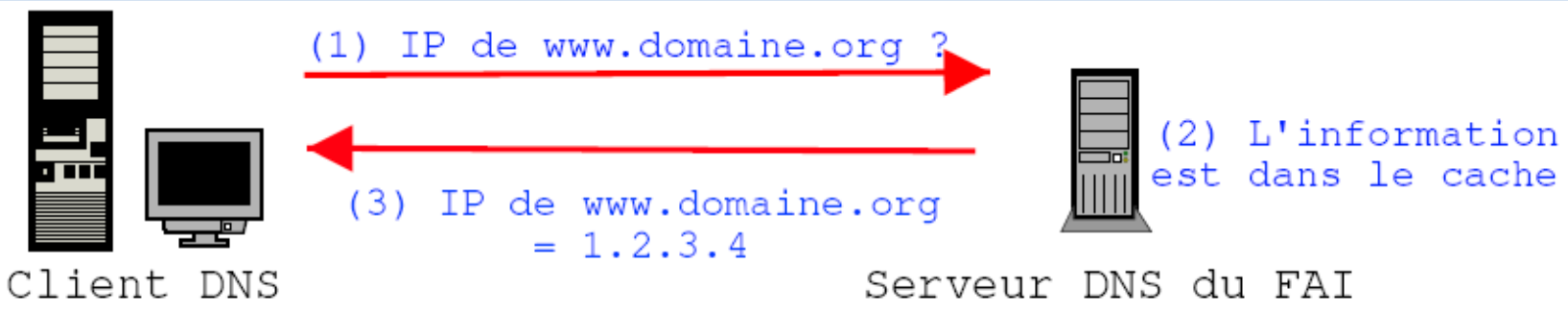


FIG. 2 – Requête sur un serveur DNS ayant l'information en cache

Source : http://www.packetfactory.net/projects/dnsa/ArticleDNS.pdf

2.2 Failles intrinsèques du DNS

- Le DNS est basé sur le protocole UDP.
  - La plupart des échanges sont en UDP.
  - Mais si requête longue ou transfert de zones,
    - Utilisation de TCP (port 53).
- Avantages de l'UDP :
  - Rapidité des requêtes, par rapport à TCP.
  - Une perte de paquets DNS n'est pas nuisible.
    - Le client réitère sa requête (dans la limite du "timeout").

2.2 Failles intrinsèques du DNS

- Problème de l'UDP :
  - Protocole non fiable, car "non connecté".
    - Peut être facilement “spoofé”.
    - Usurpation facile de l'adresse IP source.
  - Intégrité des réponses non garantie.

- *Unique* solution proposée par le protocole DNS :
  - Le DNS ID, ou Transaction ID (TXID).

2.2 Failles intrinsèques du DNS

- **Le DNS ID (TXID) :**
  - C'est l'identifiant d'une requête DNS.
    - Valeur identique dans la réponse.
  - **But :** garantir l'intégrité d'une réponse DNS.
    - La client choisit *aléatoirement* sa valeur à chaque requête.
- ***Est-ce la panacée?***
  - TXID = champ de 2 octets : soit 65536 valeurs possibles seulement !
    - Trop peu, comparé à la vitesse des processeurs actuels, à la vitesse des liaisons...

2.3 Exploitations malicieuses des failles

- **Synthèse :**
  - Le DNS est bâti sur le modèle client/serveur.
  - Des faiblesses du client : UDP, TXID court (2 bytes).
  - Les serveurs DNS peuvent devenir des clients.
    - Faiblesses côté client répercutées sur le côté serveur...
- **Deux types d'attaques envisageables.**
  - Les attaques du client DNS, type ”DNS ID *Spoofing*”.
  - Les attaques du serveur DNS, comme le ”DNS Cache *Poisoning*”.

2.3 Exploitations malicieuses des failles

- **Le ”DNS ID *Spoofing*”.**
  - **But :** trouver le DNS ID des requêtes client, pour y répondre avant le serveur légitime.
  - **Principe :**
    - ”Capter” (ou ”*sniffer*”) le trafic réseau.
    - Forger un (ou plus) paquet(s) de réponse UDP *complet(s)*.
    - Répondre au client avant le serveur DNS légitime.
  - **Autres méthodes d'attaque :**
    - Par force brute,
    - par implémentation,
    - par prédiction.

2.3 Exploitations malicieuses des failles

Illustration du DNS ID *Spoofing*

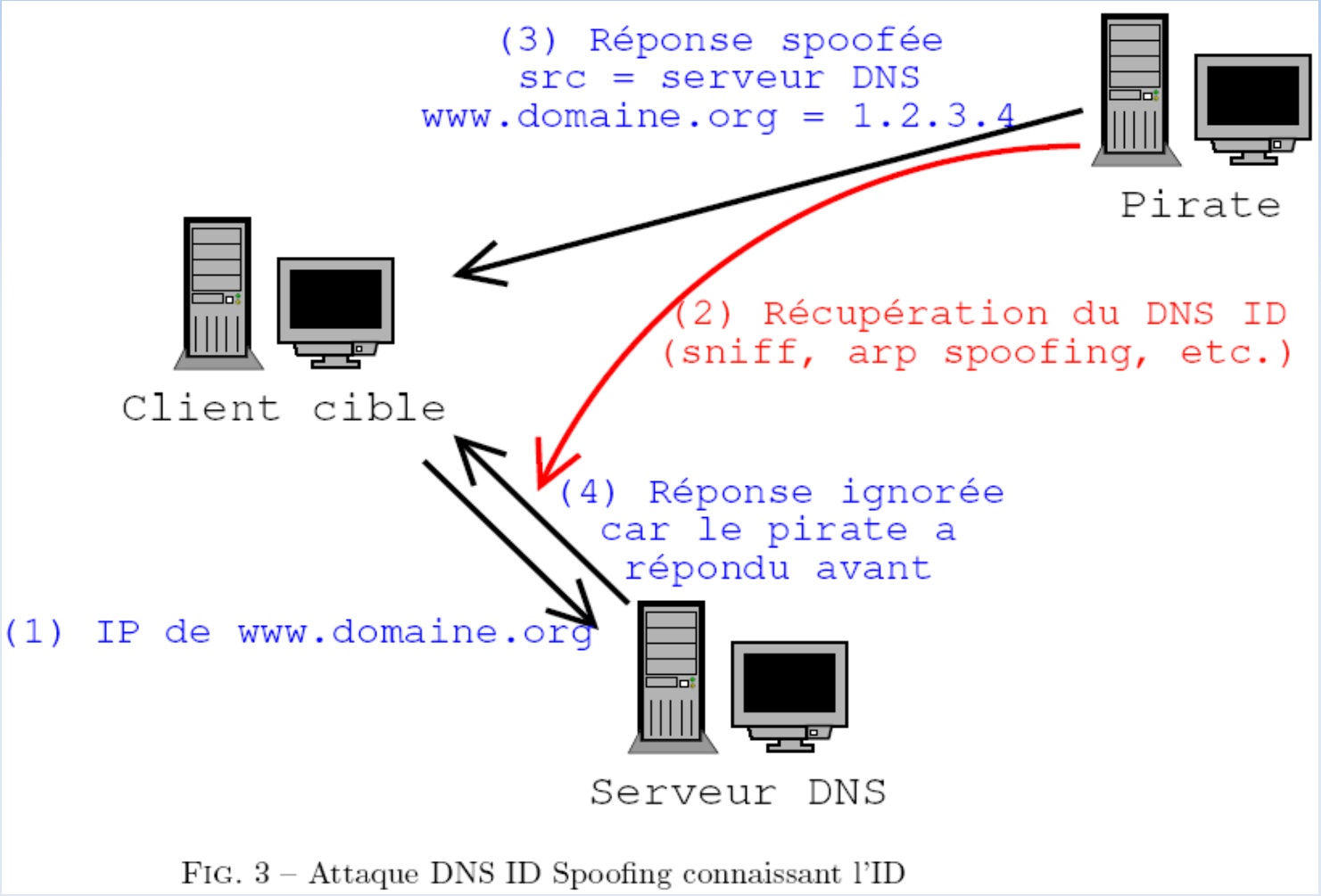


FIG. 3 – Attaque DNS ID Spoofing connaissant l'ID

Source : http://www.packetfactory.net/projects/dnsa/ArticleDNS.pdf

2.3 Exploitations malicieuses des failles

- **Le ”DNS Cache *Poisoning*”.**
  - **But :** corrompre le cache d'un serveur DNS.
  - **Principe :**
    - Faire tourner un faux serveur DNS qui permettra de rajouter un champ "*additional record*".
    - Faire une requête sur le serveur DNS victime.
      - Cette requête devra être redirigée vers le faux serveur DNS.
    - Grace au champ "*additional record*", le faux serveur interrogé répondra avec plus d'enregistrements que demandé.

2.3 Exploitations malicieuses des failles

Illustration du DNS Cache *Poisoning*

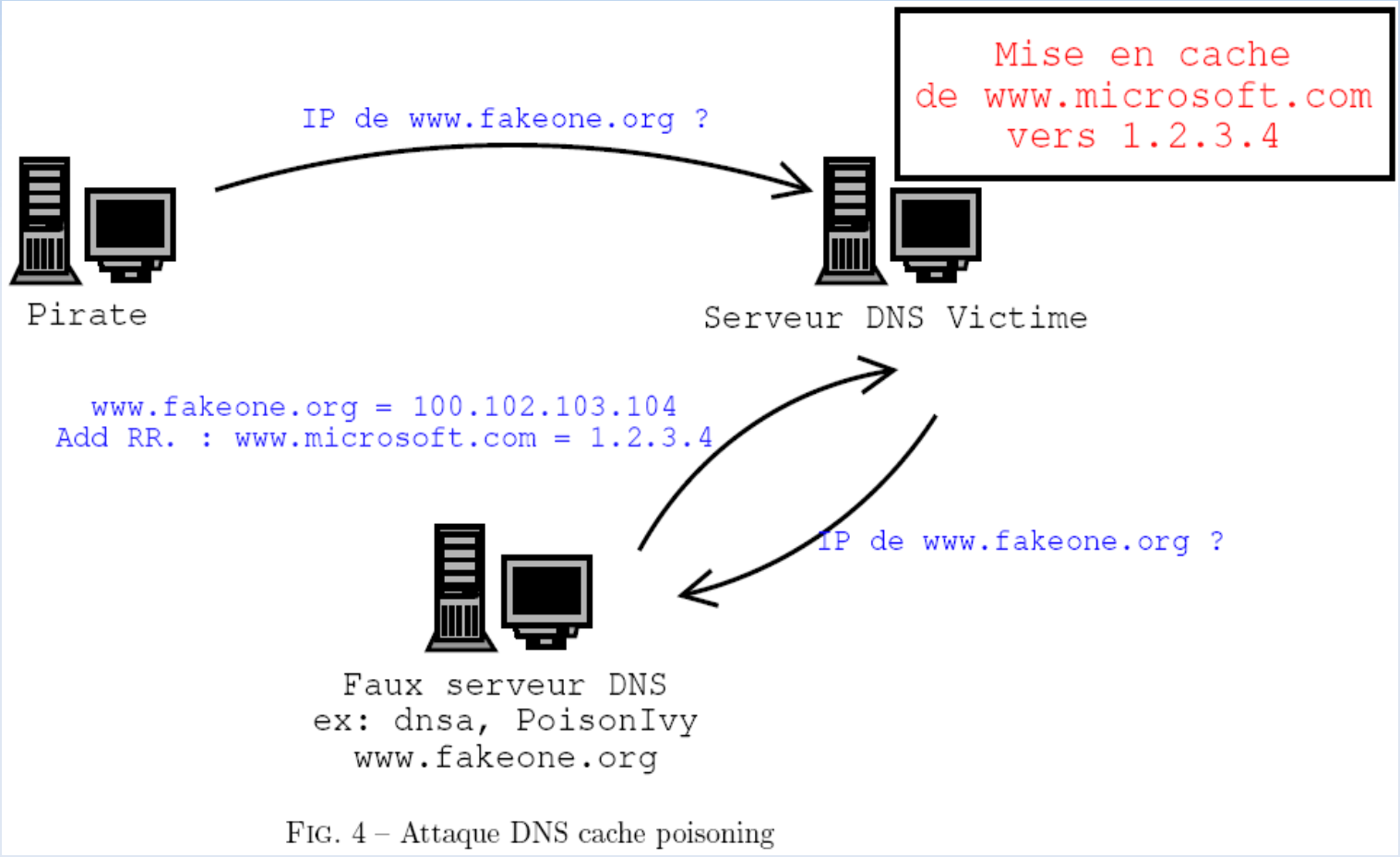


FIG. 4 – Attaque DNS cache poisoning

Source : http://www.packetfactory.net/projects/dnsa/ArticleDNS.pdf

3. La faille de Kaminsky

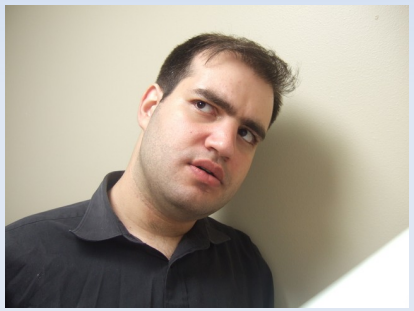
1. Introduction
2. Failles du protocole DNS
3. **La faille de Kaminsky (+ démo)**
    1. **Le buzz**
    2. **La théorie**
    3. **La pratique**
4. Des mesures de protection
5. Conclusion

3.1. Le buzz

- Fin 2007 : Début d'un travail de réflexion
  - Constat : il y a des problèmes de sécurité avec DNS
  - IETF + Bert Herbet :
    - Internet-Draft *Measures for making DNS more resilient against forged answers*
    - Complément à la RFC 1034
    - (http://tools.ietf.org/id/draft-ietf-dnsext-forgery-resilience-07.txt)

3.1. Le buzz

- Pendant ce temps là, chez IOActive
  - Dan Kaminsky découvre un moyen d'empoisonner 90% des serveurs récursifs du marché.
  - Il se tait et met au point des patchs.



- Février 2008 : la fuite
  - Kaminsky : '*J'ai détecté un bug énorme, mais n'essayez surtout pas de le trouver, attendez sagement la BlackHat de Vegas en août ! Patchez-vous, ça va être la zone !'*
  - 51h + tard : 1er exploit
  - La Websphère s'enflamme

3.1. Le buzz

- Enorme campagne de patching
  - Efficacité énorme
  - Cf. /home/clement/Bureau/DNS/Clarified_Networks_Kaminsky_DNS_repair_visualisation.flv

3.1. Le buzz

- Août 2008, Las Vegas (BlackHat)
  - 65% des serveurs patchés

La bataille est terminée... ?

3.2. La théorie

(d'après la présentation de Kaminsky réalisée à Las Vegas en 08/2008)

- But : empoisonner le cache d'un DNS récursif
- C'est une course
  - Entre le **gentil** DNS et le **méchant** DNS pour répondre au DNS **cible**
  - Si le méchant est à côté de la cible (même LAN), il gagne facilement (peut lire le TXID)
  - Sinon doit essayer de deviner...

3.2. La théorie

(d'après la présentation de Kaminsky réalisée à Las Vegas en 08/2008)

- Le gentil a un sacré avantage dans cette course
  - Le gentil a un avantage de 65536 contre 1 sur le méchant (TXID sur 16 bits)
  - Une fois que le gentil a gagné, le méchant doit attendre que le TTL expire avant de participer à une nouvelle course
- Petit calcul avec TTL = 1 jour
  - 124 ans pour 50% réussite (cf. annexe)
  - Ca reste faiblard...

3.2. La théorie

(d'après la présentation de Kaminsky réalisée à Las Vegas en 08/2008)

- Mettons-nous à la place du méchant, comment faire pour changer la donne ?
- ...
- ...
- ...
- ...
- 4 idées, dont 2 vraiment nouvelles

3.2. La théorie

(d'après la présentation de Kaminsky réalisée à Las Vegas en 08/2008)

- **1) Si c'est une course, donnons le départ !**
  - On verra comment par la suite
  - On balance notre fausse réponse immédiatement à la suite du déclencheur
- Qu'est-ce que ça donne ?
  - :-) On est sûr d'arriver le premier
  - :-(On n'est toujours pas sûr d'avoir le bon TXID

3.2. La théorie

(d'après la présentation de Kaminsky réalisée à Las Vegas en 08/2008)

- **2) On ajoute des cartouches**
  - Qui a dit qu'on n'avait le droit qu'à un essai ?
  - Si on arrive à balancer 100 réponses avant l'arrivée de celle du gentil, nos chances changent :
  - On passe de *65536 contre 1* à *655 contre 1*
- Qu'est-ce que ça donne ?
  - :-) Meilleure probabilité de succès (cf. birthday attack)
  - :-(Si TTL = 1 jour, 454 jours = **1 an et 3 mois**

3.2. La théorie

(d'après la présentation de Kaminsky réalisée à Las Vegas en 08/2008)

- **3) N'attendons plus la fin du TTL !**
  - On ne peut essayer qu'une fois par TTL d'empoisonner *www.toto.com*
  - Par contre on peut essayer *1.toto.com*, *2.toto.com*, *3.toto.com*, etc.
- Qu'est-ce que ça donne ?
  - :-) On a réussi à empoisonner *3842.toto.com* !
  - :-(Ca nous fait une belle jambe, on voulait empoisonner *www.toto.com*

3.2. La théorie

(d'après la présentation de Kaminsky réalisée à Las Vegas en 08/2008)

- **4) L'astuce**
  - Un DNS peut répondre 3 choses :
    - *Je connais 3842.toto.com, c'est 1.1.1.1*
    - *Connais pas 3842.toto.com, dommage !*
    - *T'as qu'à demander à www.toto.com qui est en 2.2.2.2*
  - Et là, la cible est forcée d'écraser son cache !
  - En effet :
    - 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.
- GAME OVER

3.3. La pratique

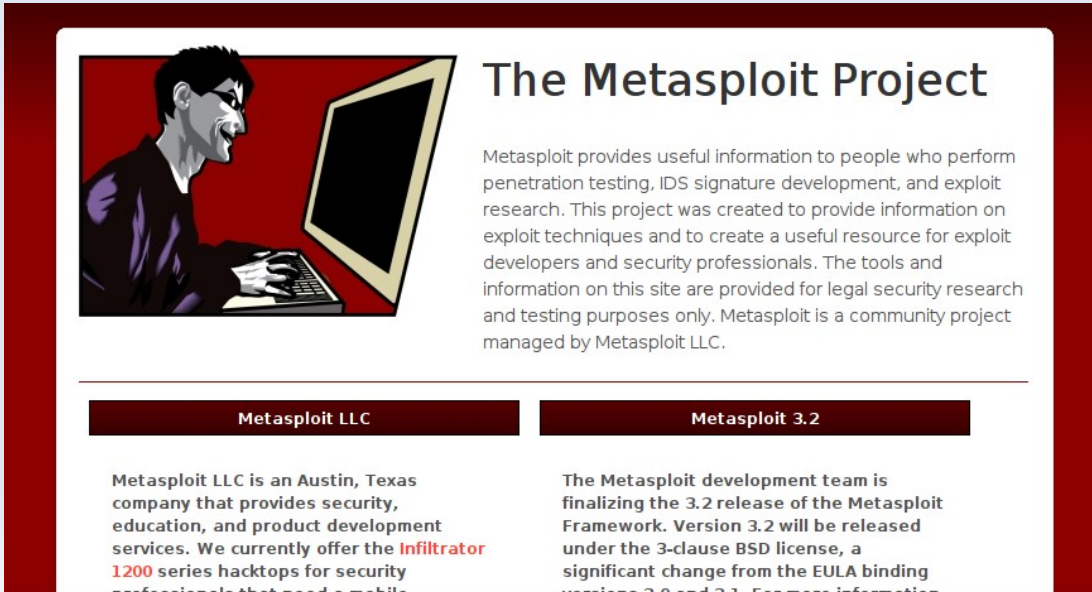
- Protocole expérimental
  - Choix d'un exploit
    - Faciles à trouver sur le Net
    - Mais lequel choisir ? (C, Python, Ruby...)
    - Celui de *Computer Academic Underground* exécuté sous *Metasploit Framework SNAPSHOT*
      - Le plus commenté
      - Le plus paramétrable
      - Le plus complet

Adresse de l'exploit :

http://www.caughq.org/exploits/CAU-EX-2008-0002.txt

Repository SVN Metasploit :

svn co http://metasploit.com/svn/framework3/trunk/



3.3. La pratique

- Protocole expérimental (suite...)
  - Choix d'une cible
    - Ubuntu 8.04
      - Mais attention : mises à jour automatiques
    - Bind 8.4.7
      - La 9.4.1 aurait suffit mais on se méfie des patchs
      - Serveur configuré en récursif
      - Simple forwarder vers DNS de l'IMAG (le gentil)
  - Choix d'une architecture
    - *cible* et *méchant* sur même machine (<= temps)

3.3. La pratique

- Procédé
  - Lecture et compréhension du code de l'exploit
  - Configuration et lancement de BIND
  - Configuration de l'exploit
  - Essai d'empoisonnement de www.google.fr
  - Observation des traces via Wireshark

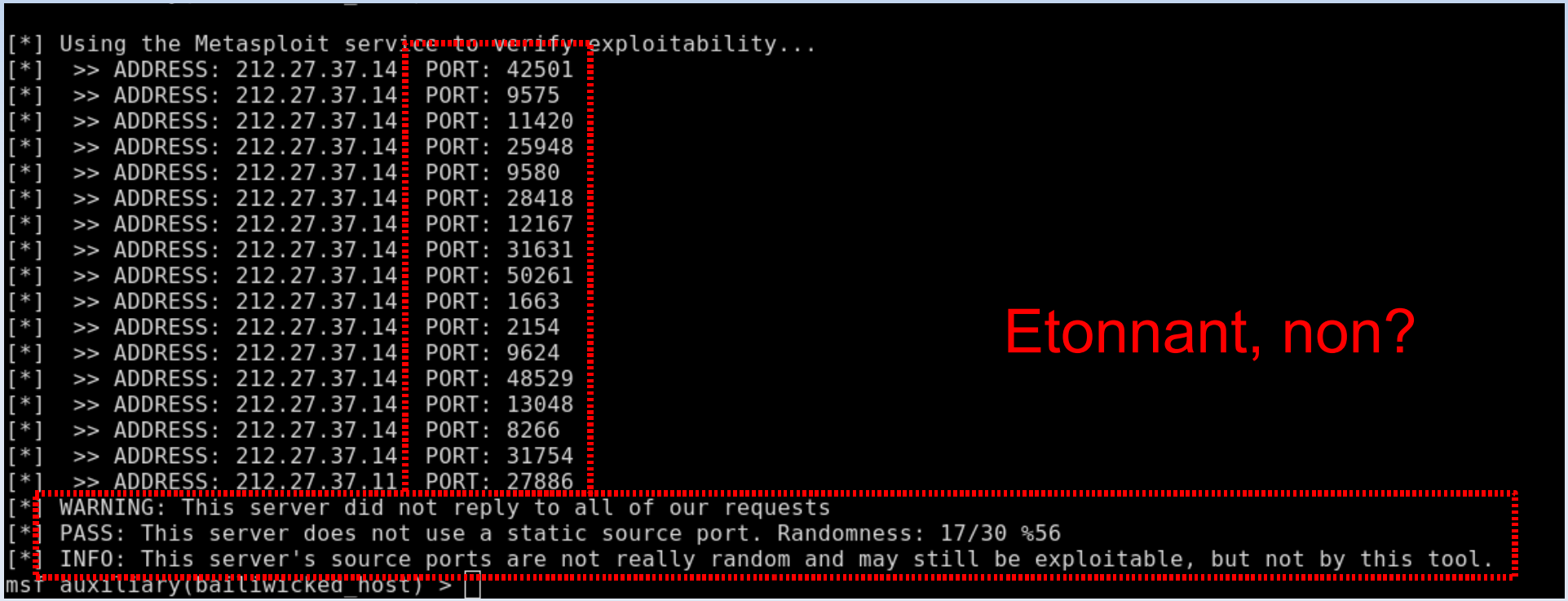
Captures d'écran

Les paramètres de l'exploit

```
msf auxiliary(bailiwicked_host) > show options

Module options:

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   HOSTNAME  www.google.fr    yes       Hostname to hijack
   NEWADDR   212.27.63.167    yes       New address for hostname
   RECONS    208.67.222.222   yes       The nameserver used for reconnaissance
   RHOST     192.168.0.1      yes       The target address
   SRCADDR   Random           yes       The source address to use for sending the queries (accepted: Real, Random)
   SRCPORT   47140            yes       The target server's source query port (0 for automatic)
   TTL       36361            yes       The TTL for the malicious host entry
   XIDS      0                yes       The number of XIDs to try for each query (0 for automatic)

msf auxiliary(bailiwicked_host) >
```

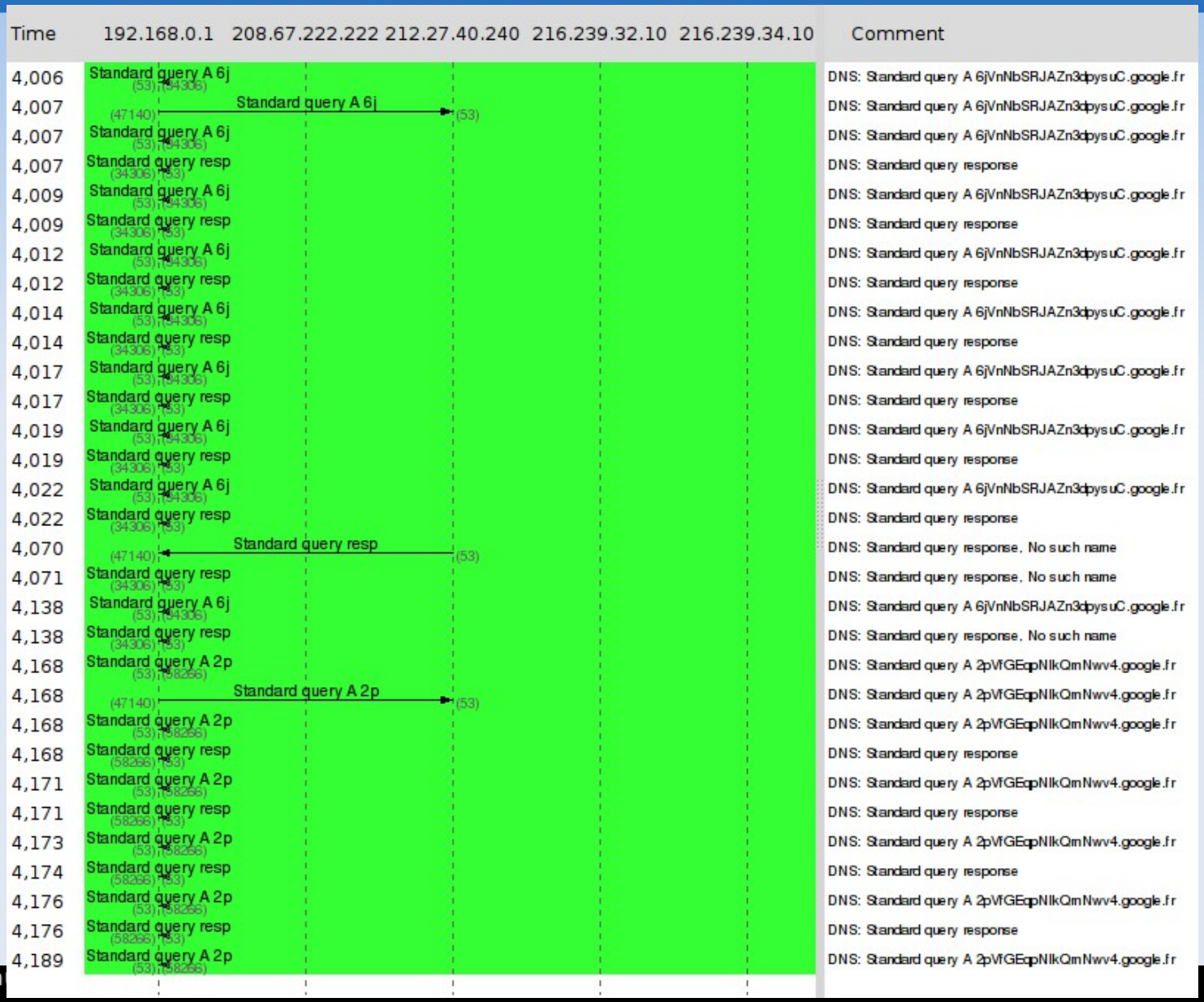
Captures d'écran

La vérification de la vulnérabilité



Captures d'écran

Calculs préalable pour la course

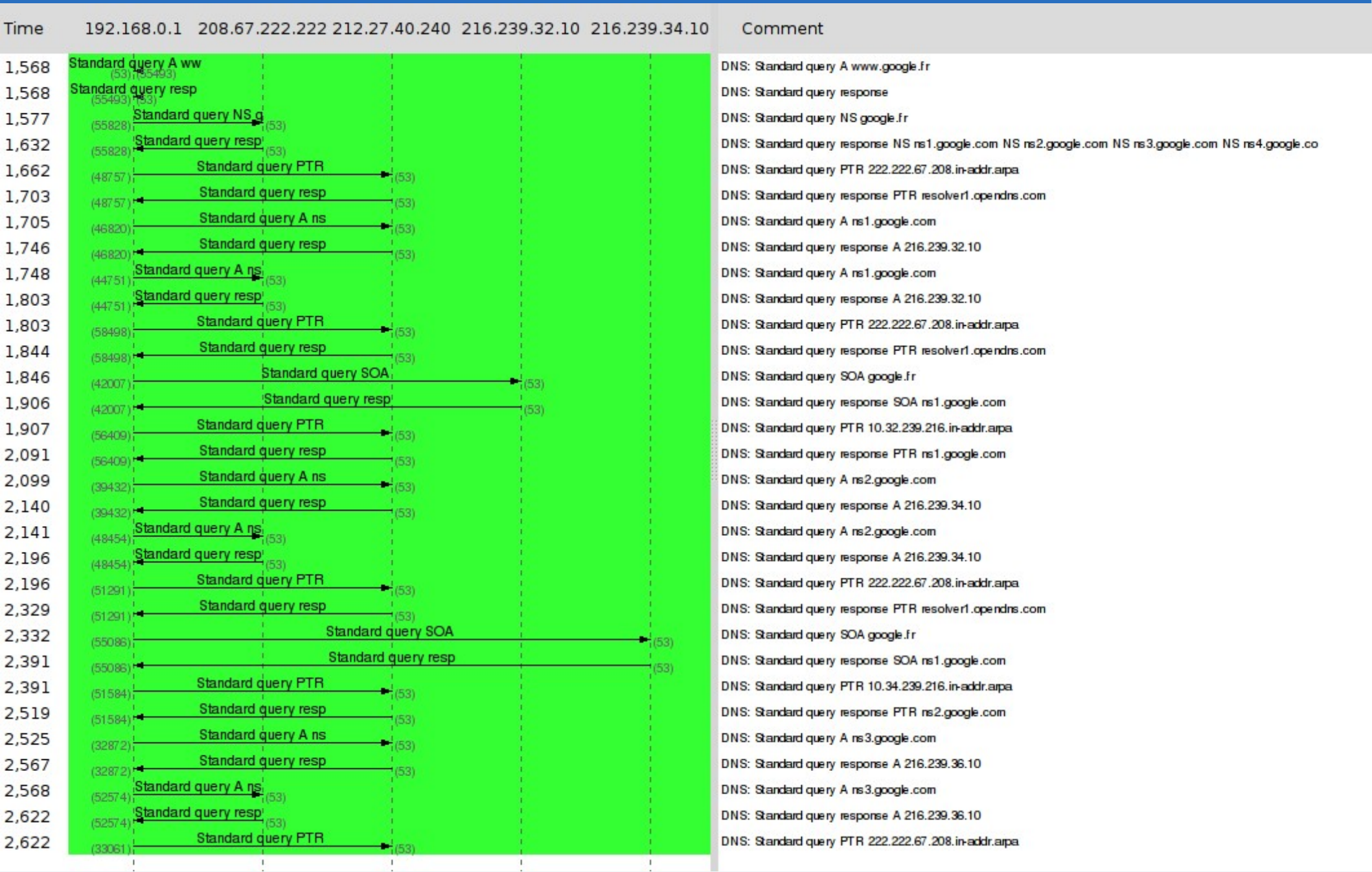


Time	192.168.0.1	208.67.222.222	212.27.40.240	216.239.32.10	216.239.34.10	Comment
4,006	Standard query A 6j (53) (34306)					DNS: Standard query A 6jVnNbSRJAZn3dpysuC.google.fr
4,007	(47140) Standard query A 6j (53)					DNS: Standard query A 6jVnNbSRJAZn3dpysuC.google.fr
4,007	Standard query A 6j (53) (34306)					DNS: Standard query A 6jVnNbSRJAZn3dpysuC.google.fr
4,007	Standard query resp (34306) (53)					DNS: Standard query response
4,009	Standard query A 6j (53) (34306)					DNS: Standard query A 6jVnNbSRJAZn3dpysuC.google.fr
4,009	Standard query resp (34306) (53)					DNS: Standard query response
4,012	Standard query A 6j (53) (34306)					DNS: Standard query A 6jVnNbSRJAZn3dpysuC.google.fr
4,012	Standard query resp (34306) (53)					DNS: Standard query response
4,014	Standard query A 6j (53) (34306)					DNS: Standard query A 6jVnNbSRJAZn3dpysuC.google.fr
4,014	Standard query resp (34306) (53)					DNS: Standard query response
4,017	Standard query A 6j (53) (34306)					DNS: Standard query A 6jVnNbSRJAZn3dpysuC.google.fr
4,017	Standard query resp (34306) (53)					DNS: Standard query response
4,019	Standard query A 6j (53) (34306)					DNS: Standard query A 6jVnNbSRJAZn3dpysuC.google.fr
4,019	Standard query resp (34306) (53)					DNS: Standard query response
4,022	Standard query A 6j (53) (34306)					DNS: Standard query A 6jVnNbSRJAZn3dpysuC.google.fr
4,022	Standard query resp (34306) (53)					DNS: Standard query response
4,070	(47140) Standard query resp (53)					DNS: Standard query response, No such name
4,071	Standard query resp (34306) (53)					DNS: Standard query response, No such name
4,138	Standard query A 6j (53) (34306)					DNS: Standard query A 6jVnNbSRJAZn3dpysuC.google.fr
4,138	Standard query resp (34306) (53)					DNS: Standard query response, No such name
4,168	Standard query A 2p (53) (58266)					DNS: Standard query A 2pVfGEqpNIkQmNwv4.google.fr
4,168	(47140) Standard query A 2p (53)					DNS: Standard query A 2pVfGEqpNIkQmNwv4.google.fr
4,168	Standard query A 2p (53) (58266)					DNS: Standard query A 2pVfGEqpNIkQmNwv4.google.fr
4,168	Standard query resp (58266) (53)					DNS: Standard query response
4,171	Standard query A 2p (53) (58266)					DNS: Standard query A 2pVfGEqpNIkQmNwv4.google.fr
4,171	Standard query resp (58266) (53)					DNS: Standard query response
4,173	Standard query A 2p (53) (58266)					DNS: Standard query A 2pVfGEqpNIkQmNwv4.google.fr
4,174	Standard query resp (58266) (53)					DNS: Standard query response
4,176	Standard query A 2p (53) (58266)					DNS: Standard query A 2pVfGEqpNIkQmNwv4.google.fr
4,176	Standard query resp (58266) (53)					DNS: Standard query response
4,189	Standard query A 2p (53) (58266)					DNS: Standard query A 2pVfGEqpNIkQmNwv4.google.fr

```
msf a
[*]  race calc: 50 queries | min/max/avg time: 0.06/0.23/0.09 | min/max/avg replies: 9/40/22
msf auxiliary(bailiwicked_host) > exploit
```

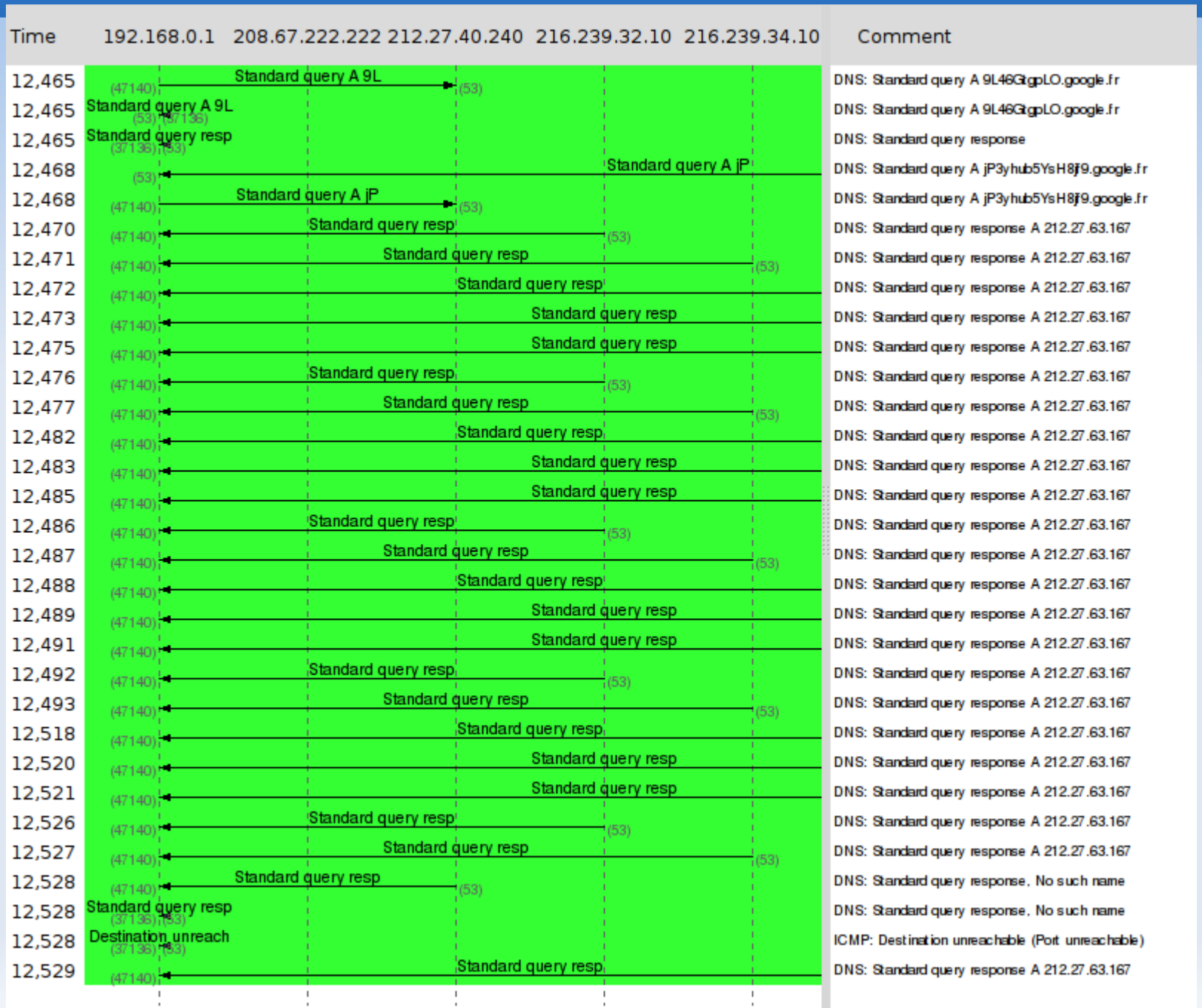
Captures d'écran

Initialisation de l'attaque : serveurs d'autorité



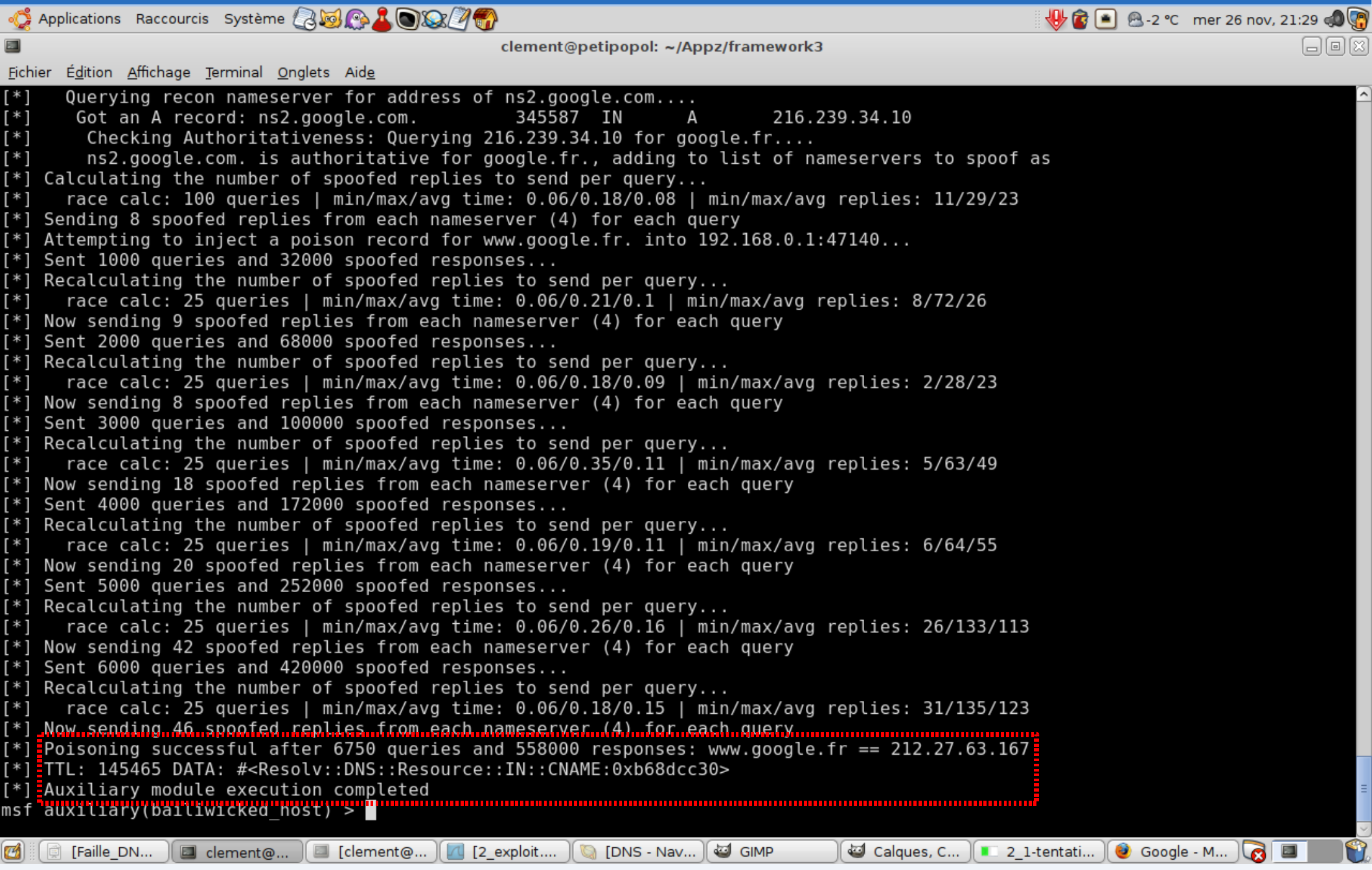
Captures d'écran

Aspect d'une tentative avec 5 réponses spoofées



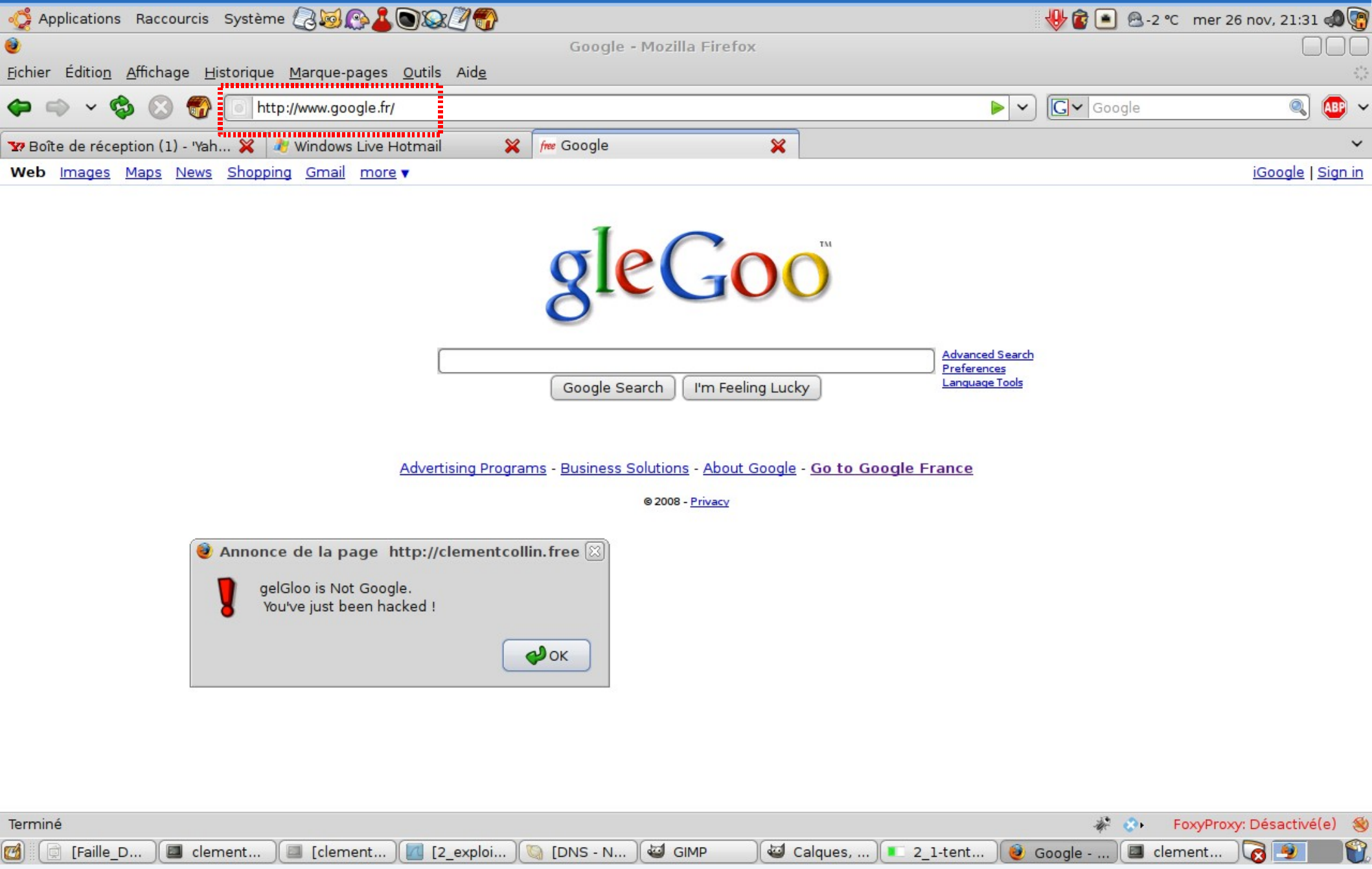
Captures d'écran

Après 25 minutes d'exécution...



Captures d'écran

...la fin des haricots



4. Les mesures de protection

1. Introduction
2. Failles du protocole DNS
3. La faille de Kaminsky (+ démo)
4. **Des mesures de protection**
    1. **Quelques mesures de base**
    2. **Pour aller plus loin...**
    3. **Et contre la faille de Kaminsky ?**
5. Conclusion

4. Des mesures de protection

- Attention : aucune mesure n'est infaillible !
  - L'augmentation du TTL :
    - Efficace dans la plupart des cas;
    - Mais inefficace devant la faille de Kaminsky.
- Même des ”*patches*” de sécurité peuvent ajouter des vulnérabilités au protocole.

4.1 Quelques mesures de base

- Mettre à jour les serveurs DNS (*”patches”*)
- Sur les systèmes non autorisés à être des serveurs DNS :
  - Désactiver le démon de noms BIND (*named*),
  - Supprimer ce démon si besoin est, pour éviter les *”backdoor*”.
- Activer les options de filtrage avancé.
  - Bloquer certains ports de communication via le pare-feu.

4.2 Pour aller plus loin...

- Utiliser, si possible, une table ARP *statique* sur le serveur DNS :
  - Evite les attaques via l' ”*ARP spoofing*”, par exemple.
- Exécuter BIND (*named*) en tant qu'utilisateur non privilégié.
  - L'exécuter dans une structure ”***chroot()ée***”.
- Cacher la version de BIND.
- Surveiller le trafic DNS...

4.2 Pour aller plus loin...

- Utiliser les extensions de sécurité DNS (*DNSSEC*) : RFC 4033, 4034, 4035.
  - Ajoute l'authentification de la source.
    - Nouveaux RR (*Resource Records*)
    - Modifications du protocole DNS.
  - Mais c'est pas la panacée (exploit BIND 8.2 NXT)
    - Le pirate peut exécuter du code ”root” sur le serveur.
    - Il peut utiliser la méthode de Kaminsky pour changer le serveur d'autorité du domaine visé.

4.3 Et contre la faille de Kaminsky ?

- Contre Kaminsky → **appliquer les correctifs !**
  - Ports des requêtes récursives aléatoires
    - On ”allonge” le TXID de 16 bits (environ)
- ...

5. Conclusion

1. Introduction
2. Failles du protocole DNS
3. La faille de Kaminsky (+ démo)
4. Des mesures de protection
5. **Conclusion**

5. Conclusion

- Réels problèmes de sécurité dans DNS (pas pensé pour ça)
- Globalement bien protégé (ajouts successifs)
- Le buzz de Kaminsky est avant tout un buzz
- Sécuriser le DNS revient aussi à sécuriser le réseau
- Avenir : DNSSEC + TCP

Annexes

1. Bibliographie
2. Quelques calculs...

Bibliographie

- Généralités :

  - COLE E., *Hackers, attention danger !*, CampusPress, Paris, 2001
  - RUSSEL R. et al., *Stratégies anti-hackers*, Eyrolles, 2002
  - LIU C., ALBITZ P., *DNS & BIND*, O'Reilly, Paris, 2006
  - BETOUIN P., *Failles intrinsèques du protocole DNS*, 2003, http://www.packetfactory.net/projects/dnsa/ArticleDNS.pdf
  - SAINSTITUTE, *Attacking the DNS Protocol – Security Paper v2*, Security Associates Institute, 2003, http://www.net-security.org/dl/articles/Attacking_the_DNS_Protocol.pdf
  - SAIZ J., *DNS, retour sur une vulnérabilité hors-norme*, Security Vibes, 2008, http://fr.securityvibes.com/vulnerabilite-dns-kaminski-retour-article-924.html
  - XMCO Partners, *L'actuSécu N° 20 : UPNP, un protocole dangereux ?*, Conseil en Sécurité en Informatique, 2008, http://www.xmcopartners.com/actu-secu/XMCO-ActuSecu-20-UPNP.pdf

Bibliographie

- Faille de Kaminsky :
  - KAMINSKY D., *It's The End Of The Cache As We Know It*, BlackHat, Las Vegas, 2008, http://www.doxpara.com/DMK_BO2K8.ppt
  - DOUGHERTY C. R., *Vulnerability Note VU#800113 : Multiple DNS implementations vulnerable to cache poisoning*, US Computer Emergency Readiness Team, 2008, http://www.kb.cert.org/vuls/id/800113
  - BORTZMEYER S., *Comment fonctionne la faille DNS « Kaminsky » ?*, 2008, http://www.bortzmeyer.org/comment-fonctionne-la-faille-kaminsky.html
  - SAIZ J., *DNS, retour sur une vulnérabilité hors-norme*, Security Vibes, 2008, http://fr.securityvibes.com/vulnerabilite-dns-kaminski-retour-article-924.html

Bibliographie

- RFCs *(Disponibles sur http://www.bind9.net/rfc)* :
  - RFC **1034** - Domain Names - Concepts and Facilities
  - RFC **1035** - Domain Names - Implementation and Specification
  - RFC **4033** - DNS Security Introduction and Requirements
  - RFC **4034** - Resource Records for the DNS Security Extensions
  - RFC **4035** - Protocol Modifications for the DNS Security Extensions

Quelques calculs...

Soit *Pk* la proba de trouver la bonne réponse au bout de *k* tentatives infructeuses.

Soit *m* le nombre de combinaisons de TXID possibles.

Soit *n* le nombre de réponses que le pirate peut donner pour une tentative.

On a : $P_k = P_{k-1} + (1 - P_{k-1}) \times n/m$

Avec : $P_0 = n/m$

Objectif : trouver k (le nombre de tentatives) en fonction de la probabilité de réussite désirée.

Quelques calculs...

- Ceci nous donne :
- Pour n = 1 et m = 65536
  - P > 50%
    - K = 45426 tentatives soit si TTL = 1 jour : **124 ans**
  - P > 90%
    - K = 150 902 tentatives soit si TTL = 1 jour : **413 ans**

Quelques calculs...

- Expérimentalement : 1 tentative toute les 75 ms
- 16 réponses spoofées par tentative
- Pour n = 16 et m = 65536
  - P > 90%
    - k = 9431 tentatives soit si 1 tentative dure 75 ms : **700 sec** (moins de 11 minutes !)
- Pour n = 6 et m = 65536
  - P > 90%
    - k = 25150 tentatives soit si 1 tentative dure 75 ms : **1886 sec** (plus de 30 minutes)

Quelques calculs...

- Réponses spoofées = 1120 bits
- 10Mbits/s
- Envoi d'1 spoof = 0,112 ms
-
- Or ici 1 ms sépare chaque spoof
- => mauvais