

# DNS Poisoning

## Pollution de cache sur des serveurs DNS

Xavier Dalem, Adrien Kunysz, Louis Plair

Université de Liège

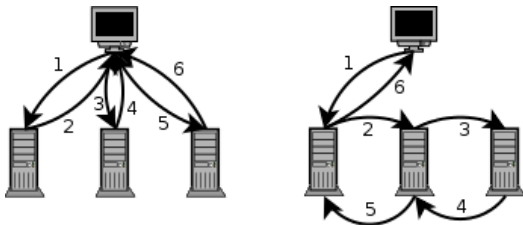
15 mars 2007

# Table des matières

- 1 Introduction
- 2 Différents types d'attaque
  - Sniffing & Spoofing
  - Blind spoofing
  - Additional records
- 3 Mise en œuvre
  - Sniffing & Spoofing
- 4 Conclusion

# DNS : rappel

- utilisé pour associer des données à un nom de domaine
- pas forcément qu'une adresse (MX, TXT,...)
- système de requête/réponse sur UDP
- hiérarchie de serveurs (un serveur est un client)
- requête récursive ou itérative



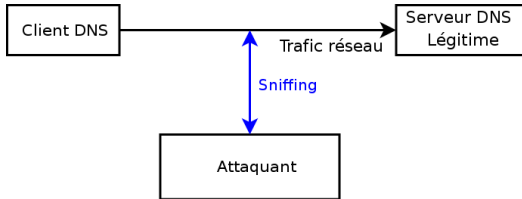
# DNS poisoning

- *empoisonnement* du cache DNS avec des adresses usurpées
- redirection des utilisateurs du service vers un serveur arbitraire
- empoisonnement de toute la hiérarchie sous-jacente

# Table des matières

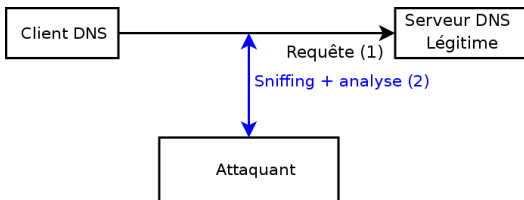
- 1 Introduction
- 2 Différents types d'attaque
  - Sniffing & Spoofing
  - Blind spoofing
  - Additional records
- 3 Mise en œuvre
  - Sniffing & Spoofing
- 4 Conclusion

# Principe...



sniffing : écoute des requêtes DNS

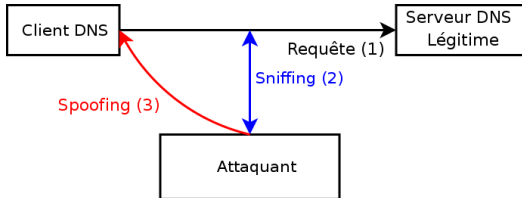
# Principe...



sniffing : écoute des requêtes DNS

analyse : récupération de l'identifiant de transaction DNS et des adresses/ports de source et de destination

# Principe...

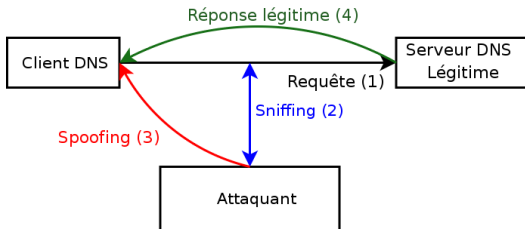


sniffing : écoute des requêtes DNS

analyse : récupération de l'identifiant de transaction DNS et des adresses/ports de source et de destination

spoofing : réponse à la place du serveur légitime

## Principe (suite)



- L'attaquant doit répondre avant le serveur légitime.
- Nécessite de pouvoir sniffer.
- Le client peut être un serveur pour d'autres clients.

# Table des matières

- 1 Introduction
- 2 Différents types d'attaque
  - Sniffing & Spoofing
  - **Blind spoofing**
  - Additional records
- 3 Mise en œuvre
  - Sniffing & Spoofing
- 4 Conclusion

# Principe

- Même principe que le Sniffing & Spoofing. . .

# Principe

- Même principe que le Sniffing & Spoofing...
- ... mais sans le Sniffing !

# Principe

- Même principe que le Sniffing & Spoofing. . .
- . . . mais sans le Sniffing !
- prédiction des identifiants de transactions (attaque d'anniversaire)
- plus besoin d'être sur le même LAN
- efficacité dépendante de l'algorithme de génération des identifiants

# Attaque de l'anniversaire (brute force)

- Le problème : prédire l'identifiant de transaction
- Solution : brute force

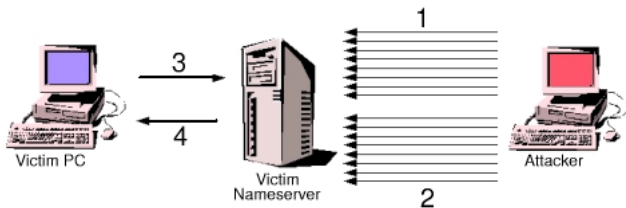
## Attaque de l'anniversaire (brute force)

- Le problème : prédire l'identifiant de transaction
- Solution : brute force
- transaction ID codé sur 16 bits → 65536 possibilités
- $P[\text{determiner ID en un 1 coup}] = \frac{1}{65536} \simeq 1,5 \cdot 10^{-5}$   
→ impossible
- Solution : Paradoxe de l'anniversaire

## Attaque de l'anniversaire (paradoxe)

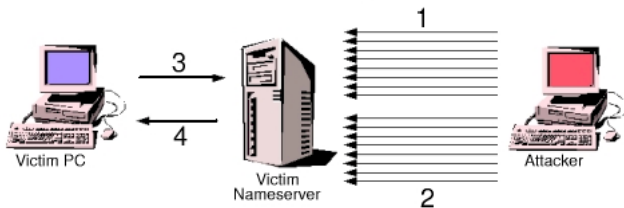
- Paradoxe : La probabilité que deux personnes dans un groupe de 23 aient la même date d'anniversaire est supérieure à 1/2.
- Général : Soit une fonction définie par  $f : x \rightarrow y \text{ mod } K$  ( $K$  valeurs possibles), on peut s'attendre à avoir deux fois la même valeur de sortie après  $1,2K1/2$  évaluations.
- $P[\text{collision}] = 1 - \left(1 - \frac{1}{t}\right)^{\frac{n(n-1)}{2}}$  où  $t$  représente le nombre de valeurs possible (365 dans le cas de l'anniversaire) et  $n$  représente le nombre d'essais (23 dans le cas du paradoxe)

## Attaque de l'anniversaire (cas du DNS)



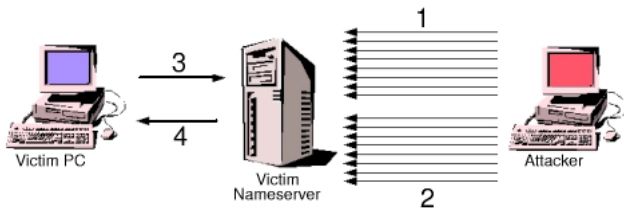
- BIND : pour N demandes identiques non présentes en cache Bind réeffectue N sous-demandes au serveur DNS d'autorité.

## Attaque de l'anniversaire (cas du DNS)



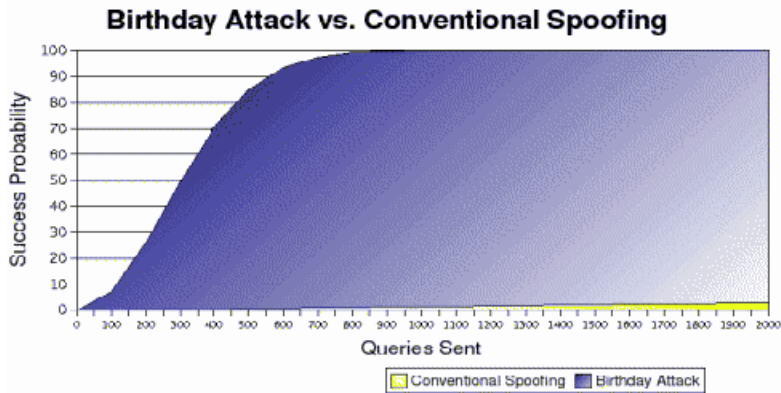
- BIND : pour N demandes identiques non présentes en cache Bind réeffectue N sous demandes au serveur DNS d'autorité.
- Attaque : (1) effectuer N demandes au serveur DNS et N réponses en parallèle (2) (ID transaction aléatoire)

## Attaque de l'anniversaire (cas du DNS)



- BIND : pour N demandes identiques non présentes en cache Bind réeffectue N sous-demandes au serveur DNS d'autorité.
- Attaque : (1) effectuer N demandes au serveur DNS et N réponses en parallèle (2) (ID transaction aléatoire)
- Réussi : une des N réponses match une demande à l'autorité → Empoisonnement du cache.

## Attaque de l'anniversaire (comparaison)



*Illustration 2: Birthday Attack vs. Conventional Spoofing*

Référence : DNS Cache Poisoning – The Next Generation. Joe Stewart. GCIH

# Table des matières

- 1 Introduction
- 2 Différents types d'attaque
  - Sniffing & Spoofing
  - Blind spoofing
  - **Additional records**
- 3 Mise en œuvre
  - Sniffing & Spoofing
- 4 Conclusion

# Principe

- requête sur le serveur de l'attaquant

# Principe

- requête sur le serveur de l'attaquant
- réponse de l'attaquant
- inclut des informations *additionnelles* empoisonnées

# Principe

- requête sur le serveur de l'attaquant
  - réponse de l'attaquant
  - inclut des informations *additionnelles* empoisonnées
  - pas d'interception ou de condition de course
- l'attaquant a seulement besoin d'un serveur DNS

# Table des matières

- 1 Introduction
- 2 Différents types d'attaque
  - Sniffing & Spoofing
  - Blind spoofing
  - Additional records
- 3 Mise en œuvre
  - Sniffing & Spoofing
- 4 Conclusion

## dnsspoof. . .

- fait partie de dsniiff par Dug Song
- simple sniffing & spoofing
- un fichier de configuration contenant les associations (*adresse, nom*)

# dnsspoof : utilisation

```
$ cat dnsspoof.hosts
82.165.176.145 www.google.com
$ dnsspoof -f dnsspoof.hosts
dnsspoof: listening on eth0 [udp dst port 53 and not src 192.168.1.11]
192.168.1.32.23204 > 139.165.12.5.53: 12725+ A? www.google.com
```

## dnsspoof : utilisation

```
$ cat dnsspoof.hosts
82.165.176.145 www.google.com
$ dnsspoof -f dnsspoof.hosts
dnsspoof: listening on eth0 [udp dst port 53 and not src 192.168.1.11]
192.168.1.32.23204 > 139.165.12.5.53: 12725+ A? www.google.com
```



# Perdu sur l'Internet ?

## Pas de panique, on va vous aider

\* <----- vous êtes ici

# dnsspoof : détection

- les réponses générées par dnsspoof ont toujours un TTL d'une minute. . .

## dnsspoof : détection

- les réponses générées par dnsspoof ont toujours un TTL d'une minute... ou pas

```
221c221
```

```
< memcpy(p, "\xc0\x0c\x00\x01\x00\x01\x00\x00\x00\x3c\x00\x04",  
---
```

```
> memcpy(p, "\xc0\x0c\x00\x01\x00\x01\x00\x37\x5f\x00\x00\x04",
```

## dnsspoof : détection

- les réponses générées par dnsspoof ont toujours un TTL d'une minute... ou pas

```
221c221
```

```
< memcpy(p, "\xc0\x0c\x00\x01\x00\x01\x00\x00\x00\x3c\x00\x04",
```

```
---
```

```
> memcpy(p, "\xc0\x0c\x00\x01\x00\x01\x00\x37\x5f\x00\x00\x04",
```

- détecter les identifiants de transaction en double
- vérifier aussi les adresses et ports ?
- ! certains clients ne considèrent que l'identifiant pour accepter la réponse
- ! exprimable avec une règle Snort ? un module ?

# Questions

Questions ?