dig .com AXFR +dnssec

Lister l’Internet avec DNSSEC

Aris Adamantiadis, SSTIC 2024

Aris Adamantiadis <aris@badcode.be>
https://infosec.exchange/@aris

Freelance sécurité des appareils embarqués

90=4x20+10

SOLAR WINE

Un peu de contexte

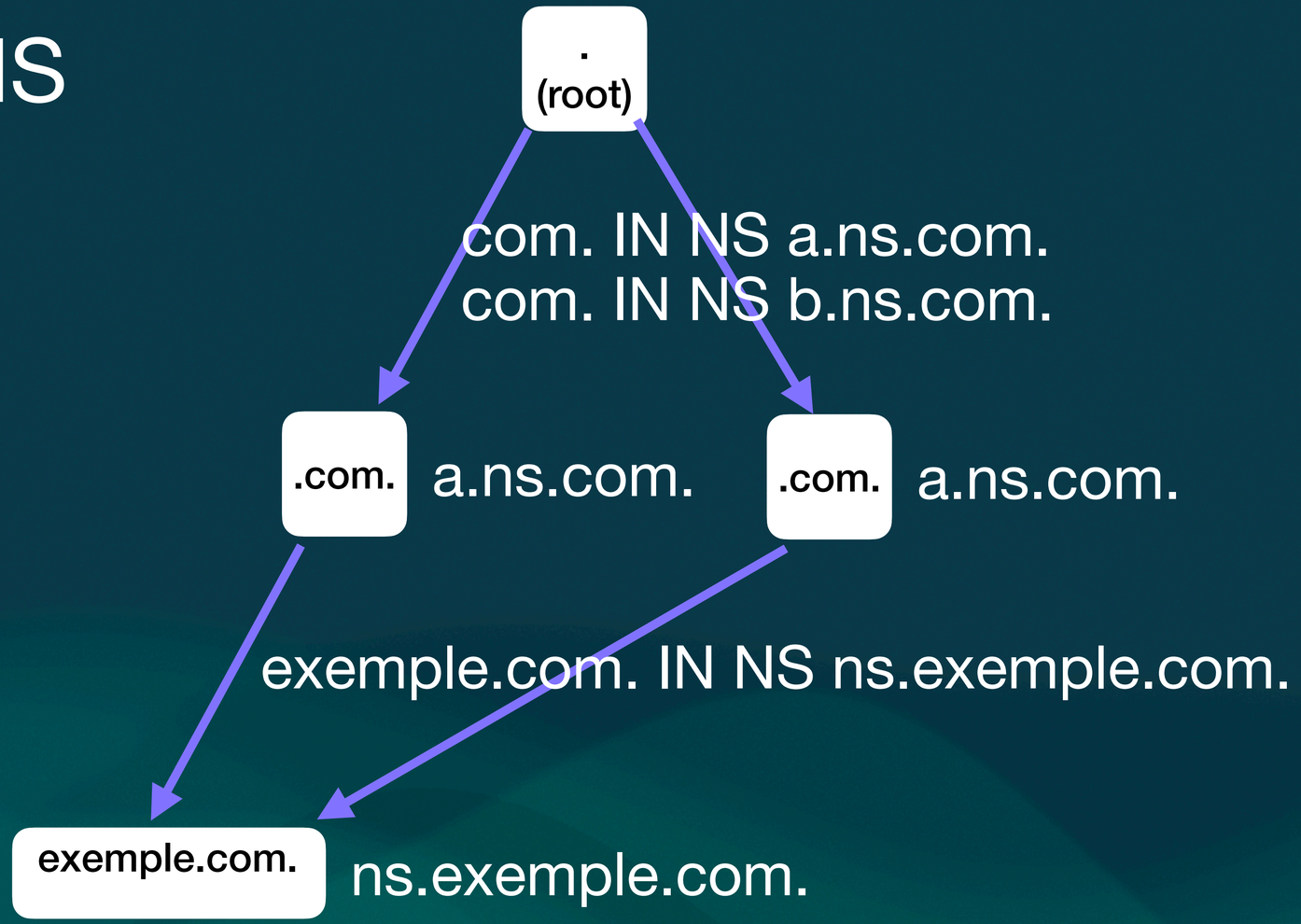
Un pirate éthique récupère des données sensibles grâce au rachat de 107 domaines expirés

Le pirate informatique et spécialiste en cybersécurité Inti De Ceukelaire (29 ans), partisan du "piratage éthique", a affirmé ce mercredi avoir pu acheter 107 noms de domaines sur internet qui appartenaient jusqu'ici à des zones de police, des hôpitaux, des services sociaux ou juridiques, mais qui n'ont pas été prolongés. Il a pu de cette manière accéder à 848 adresses mails de collaborateurs et via celles-ci à de nombreux courriers électroniques et documents sensibles. "C'est pénible de voir combien de données sensibles peuvent ainsi être achetées en ligne. Ce ne devrait pas être possible", conclut De Ceukelaire.

Anne François, Belga

mer. 22 mai 16:32

Rappel DNS



Racine

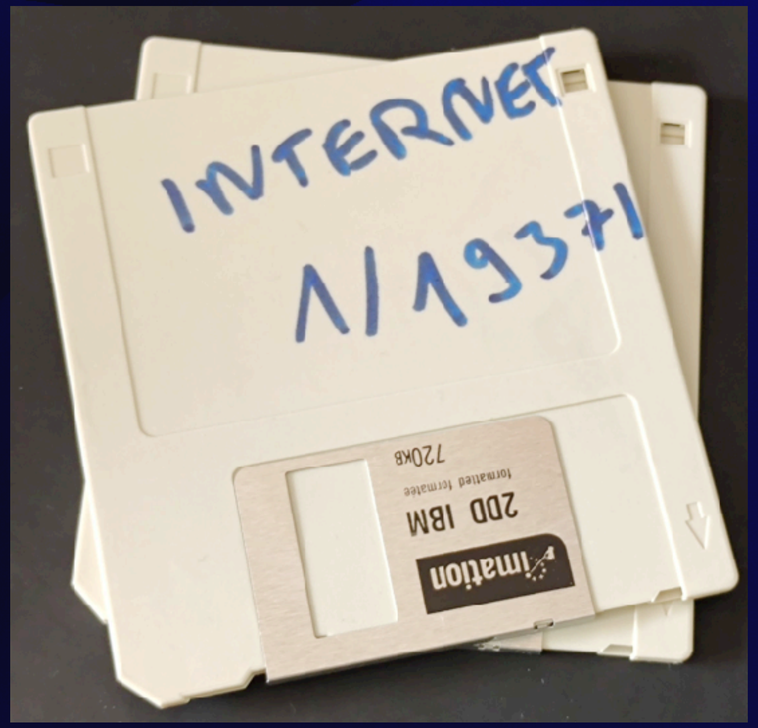
TLD
Domaines de premier niveau

2LD
Domaines de second niveau

Trivia : Combien de TLD existent aujourd’hui ?

1449

Ma question :
Est-il possible de lister la totalité des noms de domaine ?



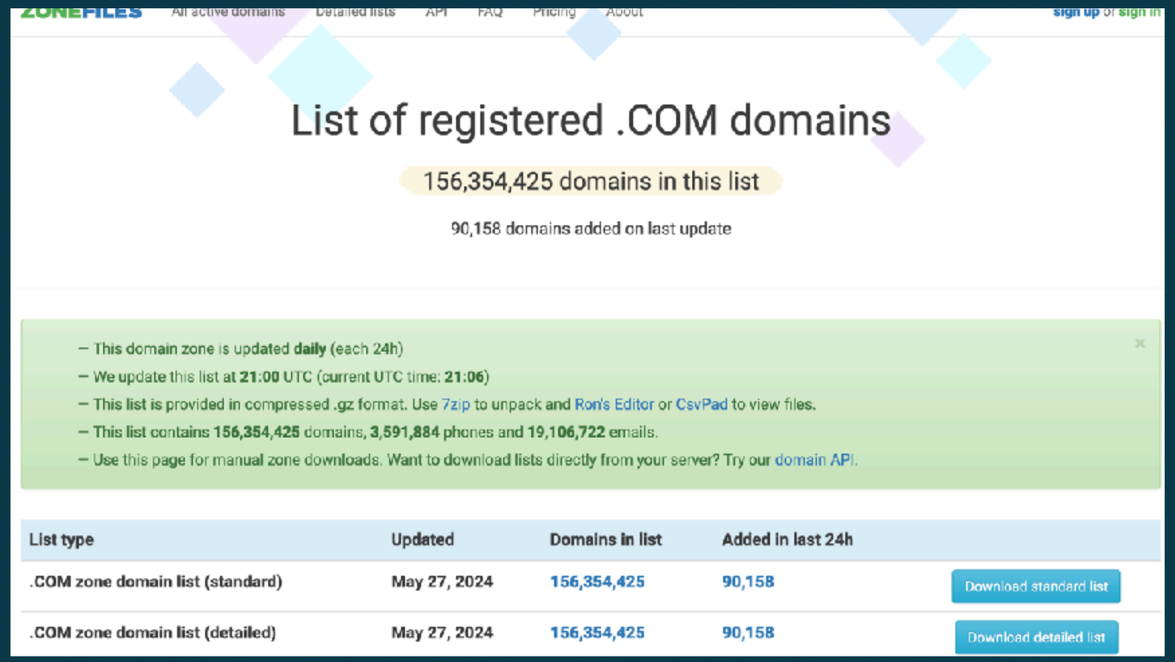
Extraire les zones des TLDs

Trois méthodes
Le bon
La brute
Le truand

Le bon

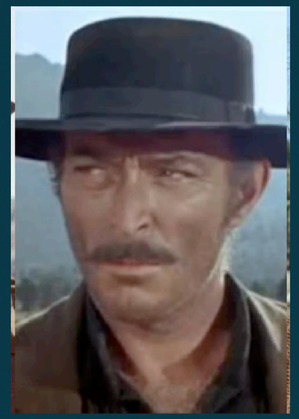
- **L'ICANN fournit un accès limité à ses membres**
- **Certains sites vendent des listes**
- **Certaines zones (.li, .ch) sont en open data**

Pas très « hacker » tout ça !




```
1
2  # filename ch_zonedata.key
3  key tsig-zonedata-ch-public-21-01 {
4      algorithm hmac-sha512;
5      secret "stZwEGApYumtXkh73qMLPqfbIDozWKZLkqRvcjKSpRnsor6A6M \
6      xixRL6C2HeSVBQNfMW4wer+qjSOZSfiWiJ3Q==";
7  };
8
9  # filename li_zonedata.key
10 key tsig-zonedata-li-public-21-01 {
11     algorithm hmac-sha512;
12     secret "t8GgeCn+fhPaj+cRy1epox2Vj4hZ45ax6v3rQCkkfIQNg5fsxu \
13     U23QM5mzz+BxJ4kgF/jiQyBDBvL+XWPE6oCQ==";
14 };
15
16 dig -k ch_zonedata.key @zonedata.switch.ch +noall +answer \
17 +noidnout +onesoa AXFR ch. > ch.txt
```

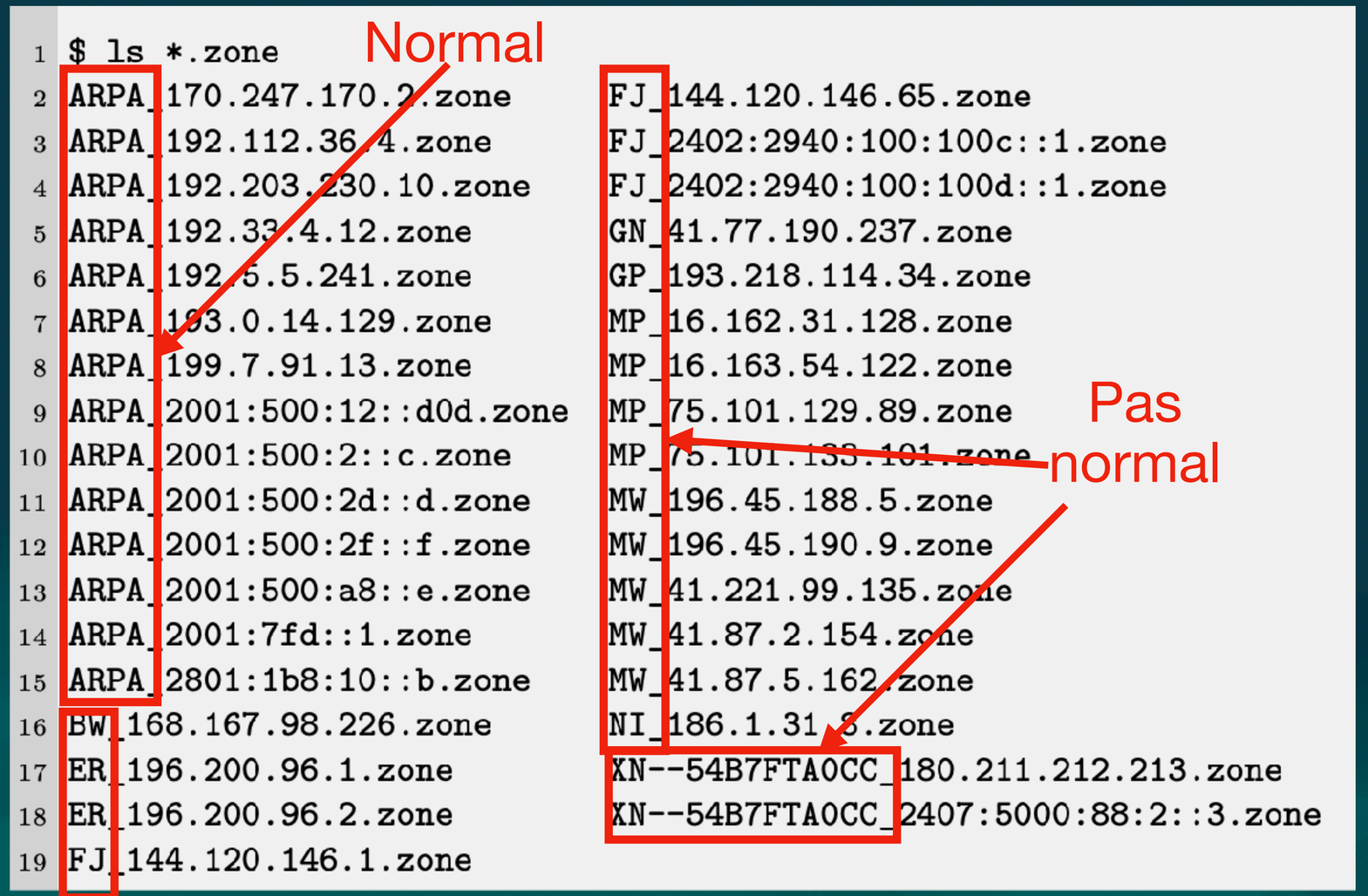
La brute



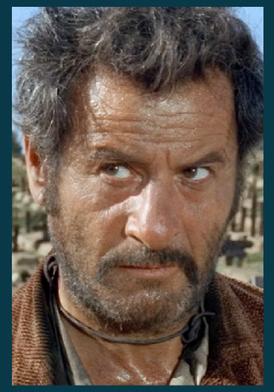
- On télécharge la liste des TLD
- On tente un transfert de zone AXFR sur chaque serveur NS de la zone

```
#!/bin/bash

TLDs=$(curl 'https://data.iana.org/TLD/tlds-alpha-by-domain.txt' |
↪  grep -v '^#')
for tld in $TLDs
do
    echo "Doing TLD $tld"
    for ns in $(dig $tld NS +short)
    do
        #echo "$tld : $ns"
        ip4=$(dig $ns A +short)
        ip6=$(dig $ns AAAA +short)
        for ip in $ip4 $ip6
        do
            #echo "$tld: $ns: $ip"
            FILENAME="${tld}_${ip}.zone"
            dig axfr $tld @$ip > "$FILENAME"
            if grep -q "SOA" "$FILENAME" ; then
                echo "Match for $tld !"
            else
                rm -f "$FILENAME"
                #echo "No match for $tld"
            fi
        done
    done
done
```

* AXFR/Transfert de zone : mécanisme de réplication de zones DNS


Normal
Pas normal
$ ls *.zone
ARPA_170.247.170.2.zone
ARPA_192.112.36.4.zone
ARPA_192.203.230.10.zone
ARPA_192.33.4.12.zone
ARPA_192.5.5.241.zone
ARPA_193.0.14.129.zone
ARPA_199.7.91.13.zone
ARPA_2001:500:12::d0d.zone
ARPA_2001:500:2::c.zone
ARPA_2001:500:2d::d.zone
ARPA_2001:500:2f::f.zone
ARPA_2001:500:a8::e.zone
ARPA_2001:7fd::1.zone
ARPA_2801:1b8:10::b.zone
BW_168.167.98.226.zone
ER_196.200.96.1.zone
ER_196.200.96.2.zone
FJ_144.120.146.1.zone
FJ_144.120.146.65.zone
FJ_2402:2940:100:100c::1.zone
FJ_2402:2940:100:100d::1.zone
GN_41.77.190.237.zone
GP_193.218.114.34.zone
MP_16.162.31.128.zone
MP_16.163.54.122.zone
MP_75.101.129.89.zone
MP_75.101.133.101.zone
MW_196.45.188.5.zone
MW_196.45.190.9.zone
MW_41.221.99.135.zone
MW_41.87.2.154.zone
MW_41.87.5.162.zone
NI_186.1.31.3.zone
XN--54B7FTA0CC_180.211.212.213.zone
XN--54B7FTA0CC_2407:5000:88:2::3.zone

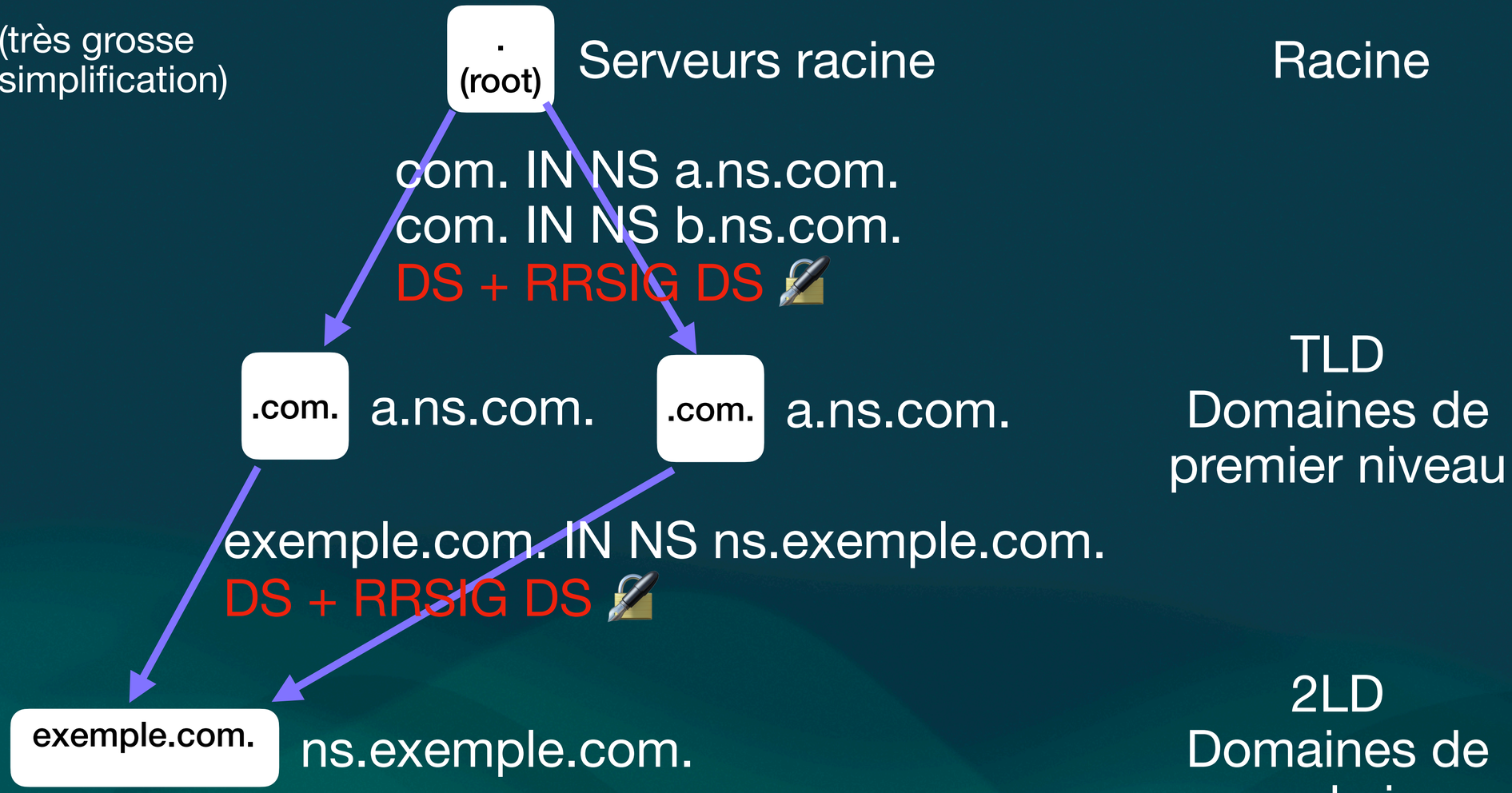
Le truand



- Vulnérabilité connue de DNSSEC : Zone Walking
- Pas une nouvelle attaque. Daniel Bernstein (DJB) l’a démontrée en 2008
- L’outil public nsec3map permet de l’exploiter
- Les outils publics ne fonctionnent pas bien sur de très larges zones (p.e .nl avec +10M entrées)

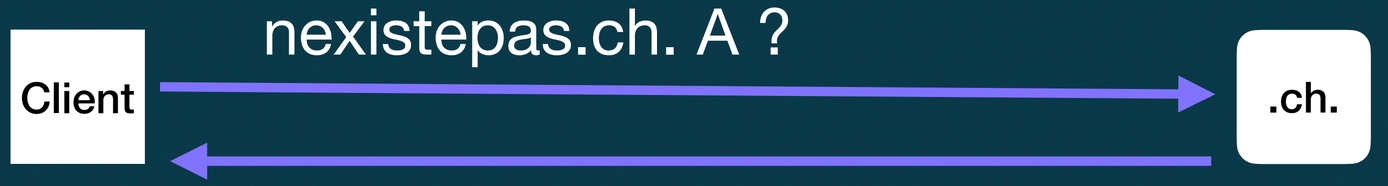
DNSSEC

(très grosse simplification)



```
$ dig nexistepas.ch A +dnssec @a.nic.ch
; <<>> DiG 9.18.18-0ubuntu0.22.04.1-Ubuntu <<>> nexistepas.ch A +dnssec @a.nic.ch
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 9005
;; flags: qr aa rd; QUERY: 1, ANSWER: 0, AUTHORITY: 6, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 1232
;; QUESTION SECTION:
;nexistepas.ch.     IN  A

;; AUTHORITY SECTION:
ch.     900 IN  SOA a.nic.ch. dns-operation.switch.ch. 2024020400 900 600 1209600 900
ch.     900 IN  RRSIG SOA 13 1 900 20240304221506 20240203220143 30091 ch. Q2WrYq2qw5Bj5lIdynu1s7G/
+p+tOYUeRzxXKxlpc+dmDA664TTjTSSp Pb2ACr8pnPZN7krXZkR5yA5WQxZ9kg==
ch.     900 IN  NSEC  0-0.ch. NS SOA RRSIG NSEC DNSKEY
ch.     900 IN  RRSIG NSEC 13 1 900 20240304141705 20240203140142 30091 ch.
hBT85gA6WAknVCt4+2Pg+DvNfJ4gSz4D2EdnD2BXnU2BFdGFoHvxW+/P eb8pacxfq7G7U4ODwbdOPhCeFIDJVA==
nexiskill.ch.   900 IN  NSEC  nexisvizzera.ch. NS RRSIG NSEC
nexiskill.ch.   900 IN  RRSIG NSEC 13 2 900 20240226035542 20240127030211 30091 ch.
jH7kPR+Fxmt9oUofkKdGp0iI2ZvvZv5M/w/mE4eor3hnPD5CCTA1U4xq wviTKyUTolEkLBPOuyr+NwDrouwRuw==

[...]
```

Quand le domaine n’existe pas



NXDOMAIN

NSEC nexiskill.ch. — nexisvizzera.ch.
RRSIG NSEC

La réponse s’applique à tous les domaines entre ces deux bornes. Ces deux domaines existent.

nexiskill.ch. 900 **IN NSEC nexisvizzera.ch.** NS RRSIG NSEC

NSEC3

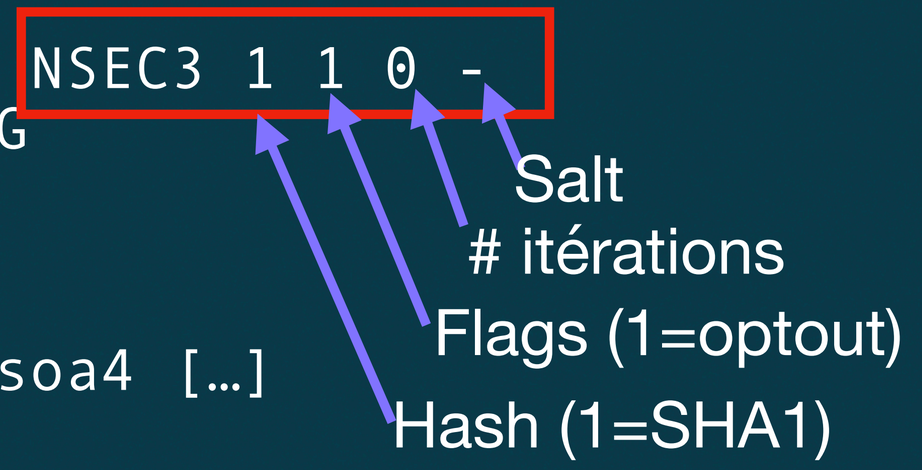
- NSEC est problématique au niveau de la confidentialité des domaines
- Cela empêchait le passage à DNSSEC
- RFC 5155 (NSEC3) introduit de la cryptographie pour rendre le listage des domaines plus compliqué
- Rajout d’une fonction de hachage (SHA-1 avec itérations et Salting)

- Nous ignorerons totalement la partie intégrité (signatures, RRSIG etc.)

$ dig **nexistepas.be**. A +dnssec @b.nsset.be
[...]
;; ->>HEADER<<- opcode: QUERY, status: **NXDOMAIN**, id: 41310
[...]
;; AUTHORITY SECTION:
n83q05hv1lpj19e1m5s6kph55sc5lh66.be. 600 IN NSEC3 1 1 0 - N840INOT5TF2DS37ILFA0FGOT0R535HO NS DS RRSIG
n83q05hv1lpj19e1m5s6kph55sc5lh66.be. 600 IN RRSIG NSEC3 8 2 600 20240215005951 20240124150612 62823 be. dEG3BEoRWf3oRVl4Y5xvjcto00Qnl0E/yfl5kXviHxBelKkGbnLFQJfC p6j0cjElBe+s0cXBNuiX7jkzDJ0CKhcQisRggBzDof0EWQGJ+kPSRlQE MpO/ wZGevLRHVI8z9mFEFbFZMQiLg52maueyO2+FwS4Zy7VYaEHIGy01 A0w=
pdrpqgs0t19r8qul7c2h4bhnb7agh1b8.be. 600 IN NSEC3 1 1 0 - PDRRMAHC5LLB04DHP1J3TFP2JERSM6G0 NS SOA RRSIG DNSKEY NSEC3PARAM
pdrpqgs0t19r8qul7c2h4bhnb7agh1b8.be. 600 IN RRSIG NSEC3 8 2 600 20240214204813 20240123183525 62823 be. nPnQHEmJBECYie4Yo45Z12SYS+zCTzMJ1VNeoH5Fok2hTHgTp5MAsB9B QadbBiZKYxr+5t8Rowfj6pZYXA7HUD/zcnhcIKJWhJEwW1CqxdPg5Y5/ W26qW0HyucC+qxqJpmfjLmvQy07qHGnNC1gsFcMiUCUnHb7/X0maxj7y p+Q=
e43jlmfjm7b0ob359cku1oek7gqqrdgd.be. 600 IN NSEC3 1 1 0 - E43KEJDV5NITCUR0E43HTLAGRLHJPV44 NS DS RRSIG
e43jlmfjm7b0ob359cku1oek7gqqrdgd.be. 600 IN RRSIG NSEC3 8 2 600 20240224010526 20240202102556 62823 be. ERjSUn0EWUUDTKIsLpLcI5M7fsyrxHhpQLEHV0Tl0ujrvx4ZZ7Z2/0pv 0DoMpoZW3BddsxbMQ3iUayy+/IfR/q8NzoymZ8P3JuhIfvwsLz3Ppg// kk2tFcXq3XWgJhU8sTMdyqbB8UqOZDQvq0RobhkHH9QV+lUNmxMRmzYy hNE=

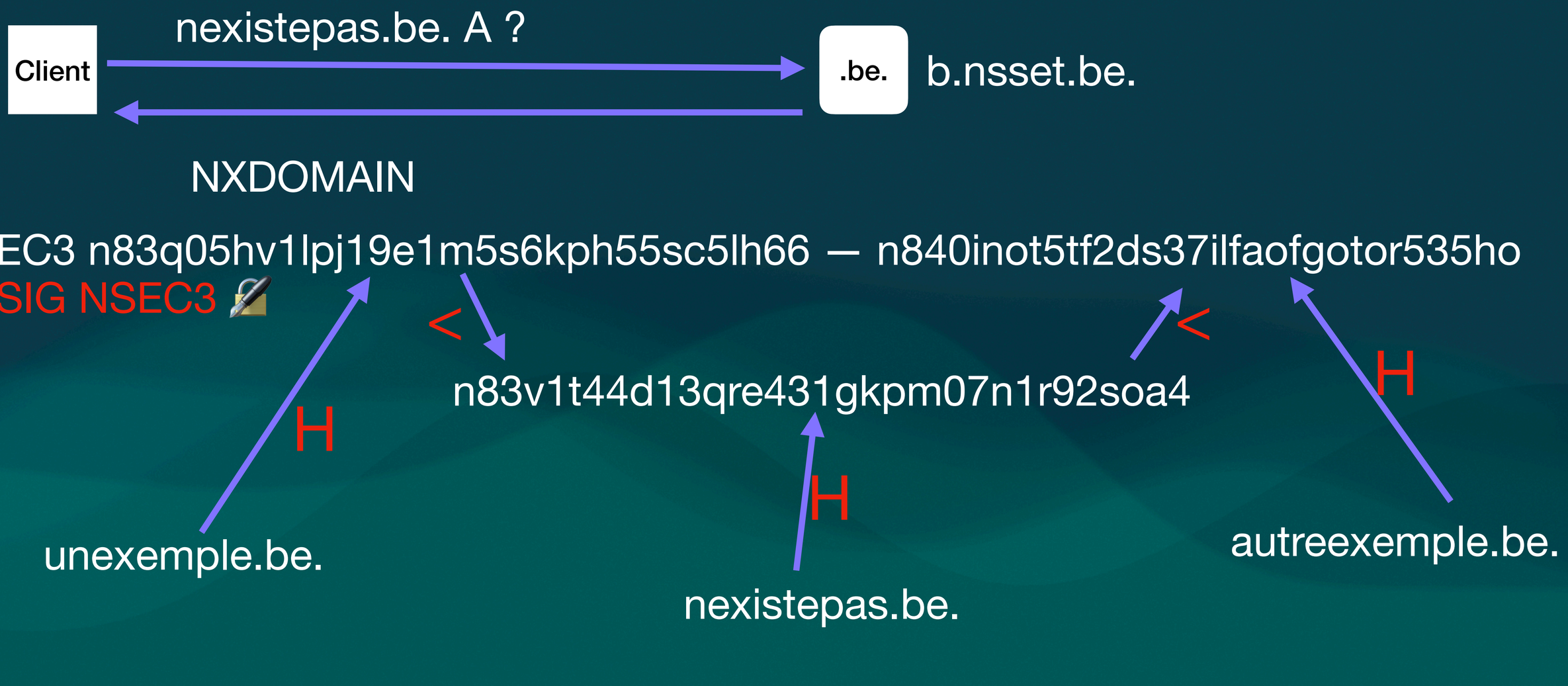
[...]

```
$ dig nexistepas.be. A +dnssec @b.nsset.be
[...]


n83q05hv1lpj19e1m5s6kph55sc5lh66.be. 600 IN NSEC3 1 1 0 -
n840inot5tf2ds37ilfaofgotor535ho NS DS RRSIG
[…]

$ ./fastnsec3/nsec3hash.py nexistepas.be 0
nexistepas.be: n83v1t44d13qre431gkpm07n1r92soa4 […]
```



- Le domaine (FQDN) est haché avec les paramètres de la zone.
- Hash = base32(H(fqdn, salt, itérations))
- C’est ce hash qui est comparé à la base de données DNS

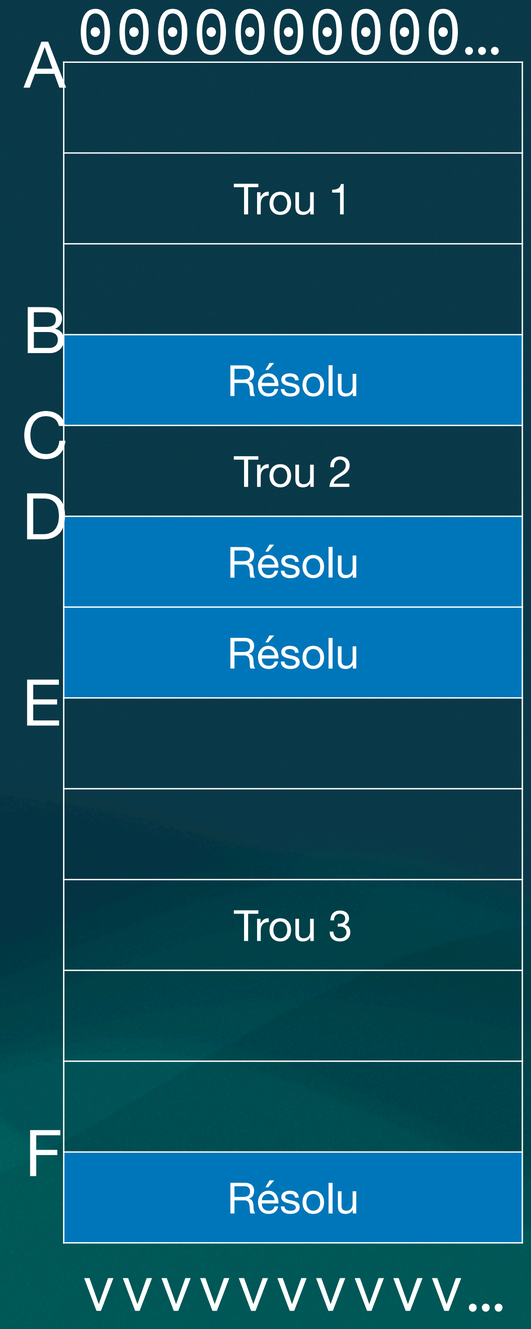
Quand le domaine n’existe pas (NSEC3)



Comment lister une zone NSEC3 (Algorithme de base)

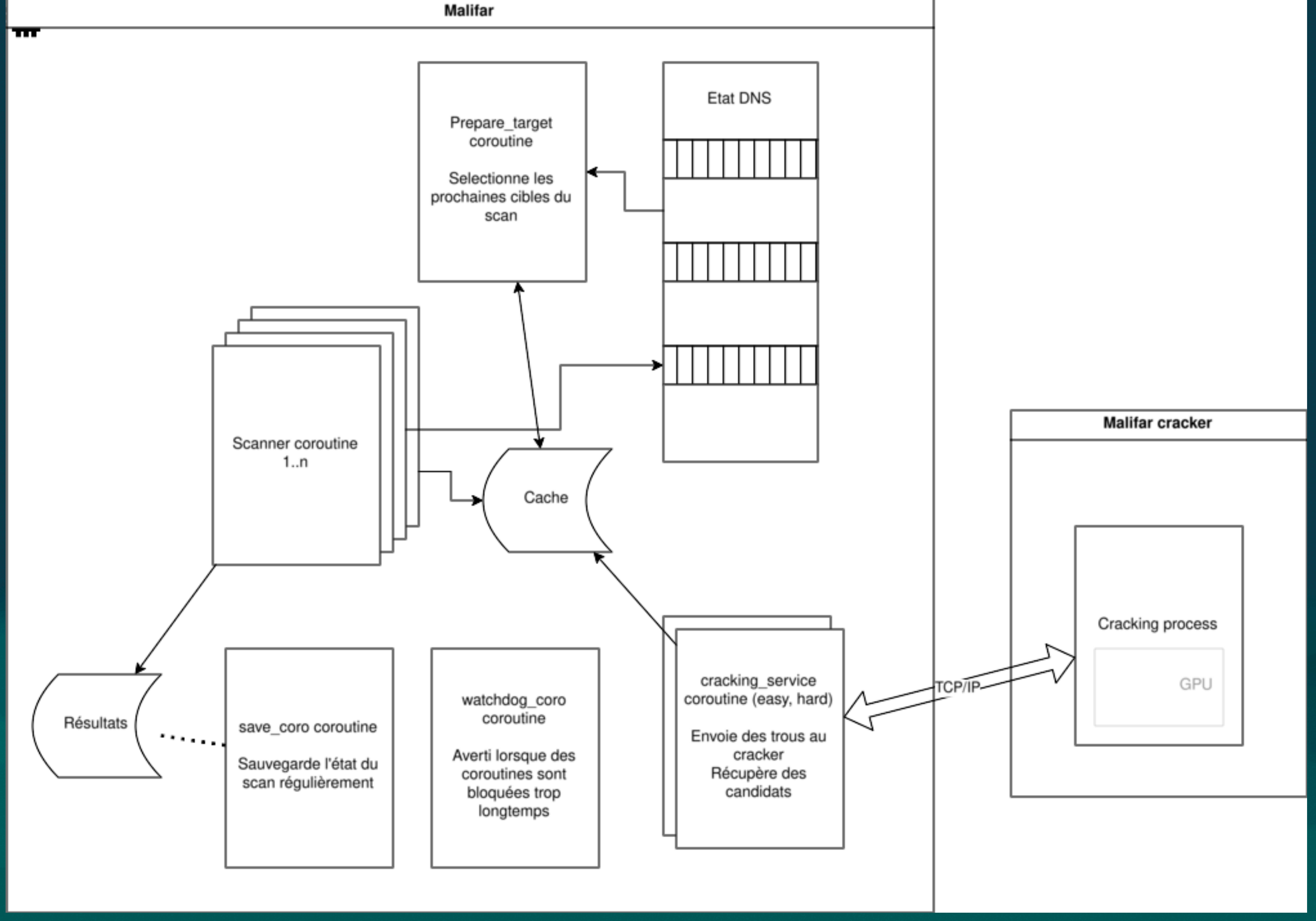
- Choisir un nom de domaine au hasard
- Hachage (SHA1, itérations, salt)
- Si le hash est dans une zone non découverte (trou), on envoie au nameserver
- On marque l'intervalle entre les deux réponses comme résolu

Difficultés: l'espace de recherche est immense (7,000,000 .com)
Un grand nombre de hash nécessaires (jusqu'à 2^{45})
Consommation de RAM et CPU



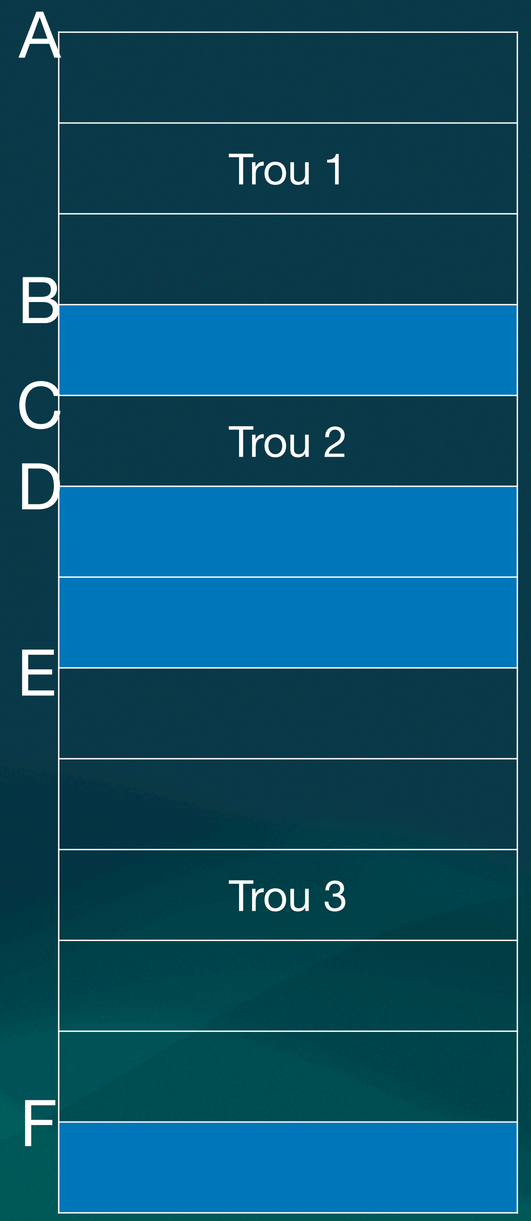
Le scanner NSEC3 Malifar

- Python, asyncio
- TCP DNS (vs UDP), IPv4 + IPv6
- Configuration par TLD
- Système de cache pour le stockage des hash intermédiaires
- Stockage efficace des résultats
- Aperçu temps réel de l’état d’avancement
- Crackage des hashes intermédiaires sur GPU déporté sur une autre machine
- Optimisation mémoire
- Stop/restore rapide
- C’est sur GitHub :) https://github.com/arisada/malifar

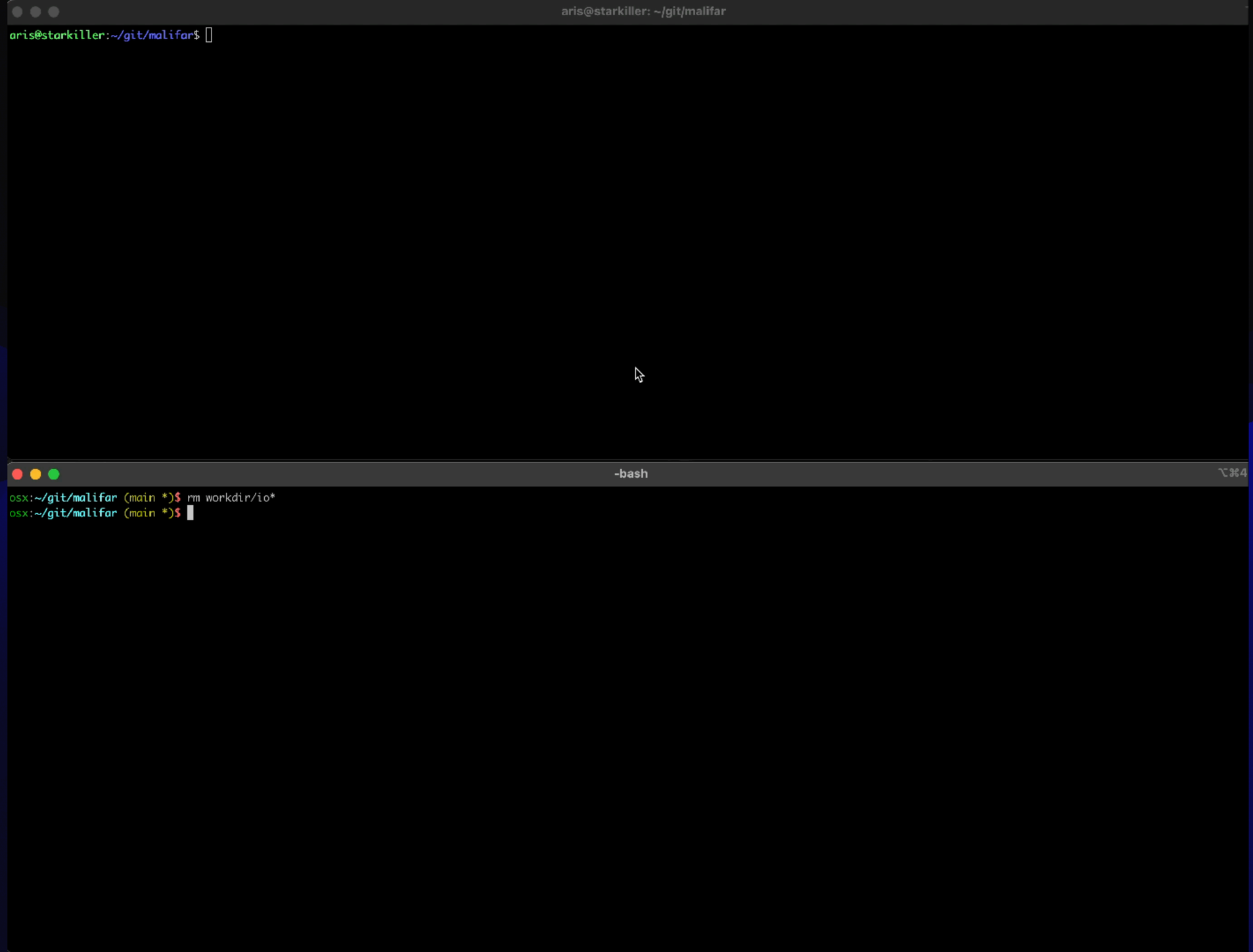

Malifar
Prepare_target coroutine
Selectionne les prochaines cibles du scan
Etat DNS
Scanner coroutine 1..n
Cache
Résultats
save_coro coroutine
Sauvegarde l'état du scan régulièrement
watchdog_coro coroutine
Averti lorsque des coroutines sont bloquées trop longtemps
cracking_service coroutine (easy, hard)
Envoie des trous au cracker
Récupère des candidats
TCP/IP
Malifar cracker
Cracking process
GPU

Cracking GPU

- Implémentation en OpenCL avec PyOpenCL
- Basé sur une implémentation SHA-1 sous license MIT
- Le serveur reçoit une liste de trous, par ex. (A,B), (C,D), (E,F)
- Ces bornes sont stockées dans des entiers 64 bits
- Renvoie des candidats FQDN qui rentrent dans ces trous
- Performance: 2GH/s on RTX4090
- Certains trous complexes (2^{45}) peuvent prendre une heure ou plus
- Les résultats sont stockés dans une db locale pour accélérer les prochains scans



Démo


aris@starkiller: ~/git/malifar
aris@starkiller:~/git/malifar$
-bash
⌥⌘4
osx:~/git/malifar (main *)$ rm workdir/io*
osx:~/git/malifar (main *)$

Crackage des noms de domaine

- Malifar génère des fichiers compatibles avec Hashcat
- NSEC3 est implémenté depuis des années
- 15GH/s sur RTX4090
- Espace de caractères a-z 0-9 -
- Beaucoup de domaines basés sur des mots du dictionnaire

Crackage des noms de domaine

- Malgré le jeu de caractère réduit, crackstation.txt et rules, on n’arrive qu’à 50% au bout de 24h sur .com
- Travail à réaliser pour améliorer les listes de mots :
  - Récolter des listes de domaines d’autres sources
  - Juxtaposer des mots avec et sans -
  - Prendre en compte punycode (caractères unicode) avec noms en `xn--`
  - Noms de domaines composés, p.e. abc.co.uk

Quelles protections sur les TLD ?

- **Grande majorité avec NSEC3**
- **La plupart utilise les paramètres par défaut**
  - **Pas de sel, 0 ou 1 itération**
  - **3 TLDs changent le salt toutes les 20 minutes**
  - **Un groupe de TLD utilisent 100 itérations mais le même salt**
- **La plupart utilisent le « Opt out flag » (pas d'entrées pour les délégations non sécurisées)**

Type of DNSSEC	TLD Number
Pas de DNSSEC	196
NSEC*	48
NSEC3	1299
Other	13

* certaines mitigations comme white lies apparaissent comme NSEC

Mitigations trouvées

- Rotation des salts (toutes les 20 min)
  - Empêche les longs dumps, rend les caches moins utiles
  - Contre-mitigation : lister le TLD en moins de 20 min
- Limites sur les requêtes
  - Certains serveurs ont des limites en temps et nombres de requêtes par session TCP
  - Contre-mitigation : réglages « doux »
- Protections anti DDOS
  - Utiliser un nombre raisonnable de flux par IP (2)

Mitigations trouvées

- Itérations
  - Rendent les hashes intermédiaires plus durs à trouver
  - Rendent le cracking beaucoup plus long.

- RFC9276 "Guidance for NSEC3 Parameter Settings »
  "In short, for all zones, the recommended NSEC3 parameters are as shown below :
  ; SHA-1, no extra iterations, empty salt :
  bcp.example. IN NSEC3PARAM 1 0 0 -"

Mitigations

- White Lies (RFC4470, 2006)
  - Technique basée sur NSEC
  - Génère des réponses signées à la demande
  - Par exemple : NXDOMAIN pour foo.com:

fon\255\255\255[…]\255\255.example.com 3600 IN NSEC \000.foo.example.com (NSEC RRSIG)

  - Nécessite la clef privée sur le serveur DNS
- Black Lies
  - "Compact Denial of Existence in DNSSEC" (https://datatracker.ietf.org/doc/draft-ietf-dnsop-compact-denial-of-existence/)
  - Implémenté par cloudflare, variation de White Lies

Nouvelle mitigation ?

- **Proposition : Rajouter des hashes factices intermédiaires**
- **Imaginons que le serveur DNS répond:**

```
nexistepas.be: n83v1t44d13qre431gkpm07n1r92soa4
 n83q05hv1lpj19e1m5s6kph55sc5lh66.be. 600 IN NSEC3 1 1 0 -
 N840INOT5TF2DS37ILFA0FGOT0R535HO NS DS RRSIG
```

- **Rajouter des fausses entrées entre les vrais hash**

```
n83q05hv1lpj19e1m5s6kph55sc5lh66.be.
n83q05hv1lpj19e1m5s6k00000000000.be.
n83q05hv1lpj19e1m000000000000000.be.
n83q05hv1lpj1000000000000000000.be.
n83q05hv1000000000000000000000.be.
n840in0t5000000000000000000000.be.
n840in0t5tf2d000000000000000000.be.
N840in0t5tf2ds37i00000000000000.be.
```

Deux valeurs retournées

H(nexistepas.be.)

Mitigations

- Approche pragmatique
  - Partir du principe que ces informations sont déjà publiques et pas à protéger
  - Suivre l’exemple de Switch (.ch) et publier les zones en open data
  - Plus de scans

Travaux futurs

- Quelques points laissés en suspend
  - Publier les scans crackés en ligne
  - Scripter les scans et le cracking
  - Publier automatiquement les différences
  - Site web avec statistiques et téléchargements
  - API pour rechercher des nouveaux domaines sur base de similarité (typosquatting)

Des questions ?

https://github.com/arisada/malifar

Lisez l’article des actes !