Déni de service algorithmique

Cédric Lauradoux

17 novembre 2014

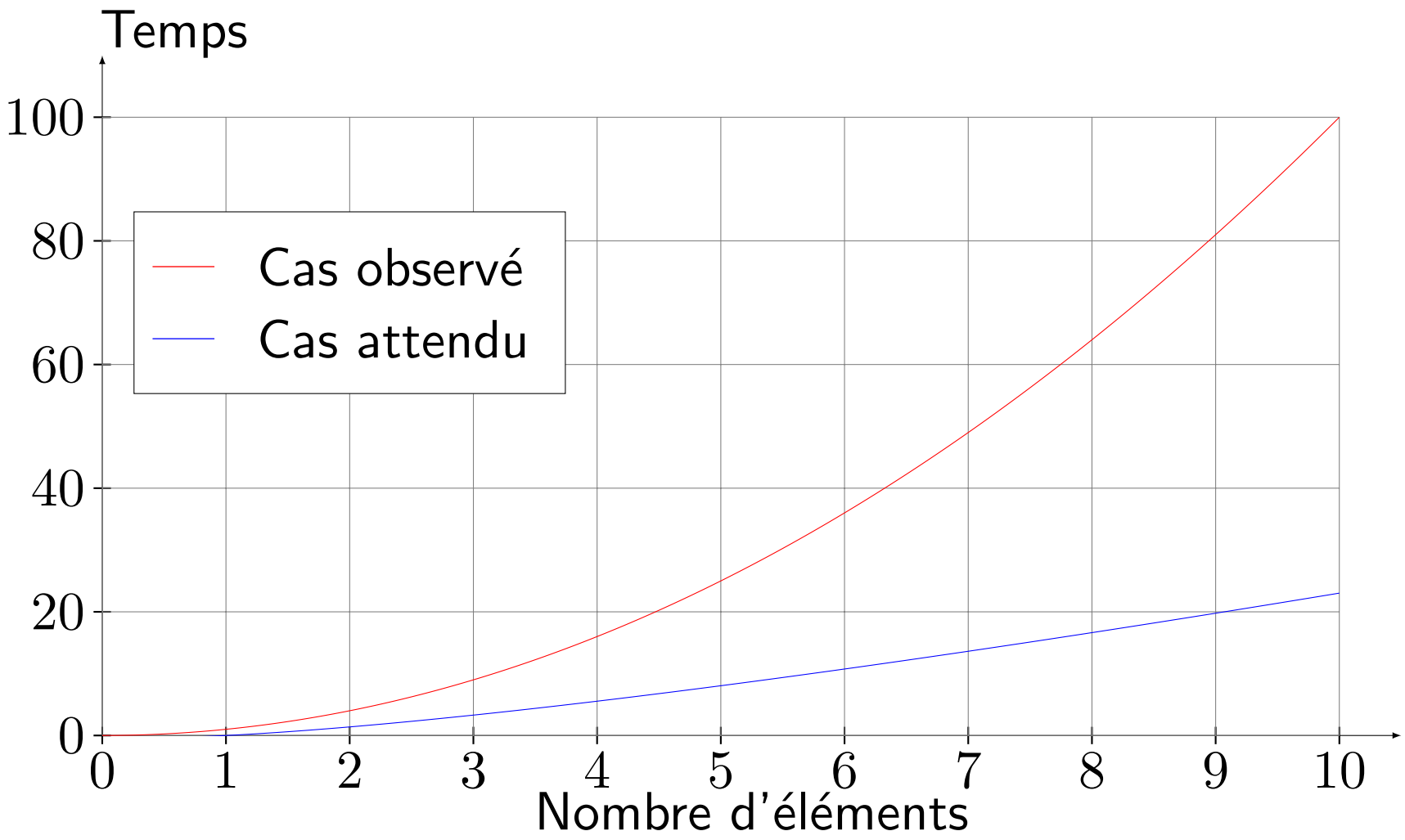
Déni de Service Algorithmique

Trier un tableau de n entiers

	Complexité en temps			
Algorithme	Meilleur	Moyen	Pire	Mémoire
Quicksort	$O(n\log(n))$	$O(n\log(n))$	$O(n^2)$	$O(n)$
Mergesort	$O(n\log(n))$	$O(n\log(n))$	$O(n\log(n))$	$O(n)$
Heapsort	$O(n\log(n))$	$O(n\log(n))$	$O(n\log(n))$	$O(1)$
Bubble Sort	$O(n)$	$O(n^2)$	$O(n^2)$	$O(1)$
Insertion Sort	$O(n)$	$O(n^2)$	$O(n^2)$	$O(1)$

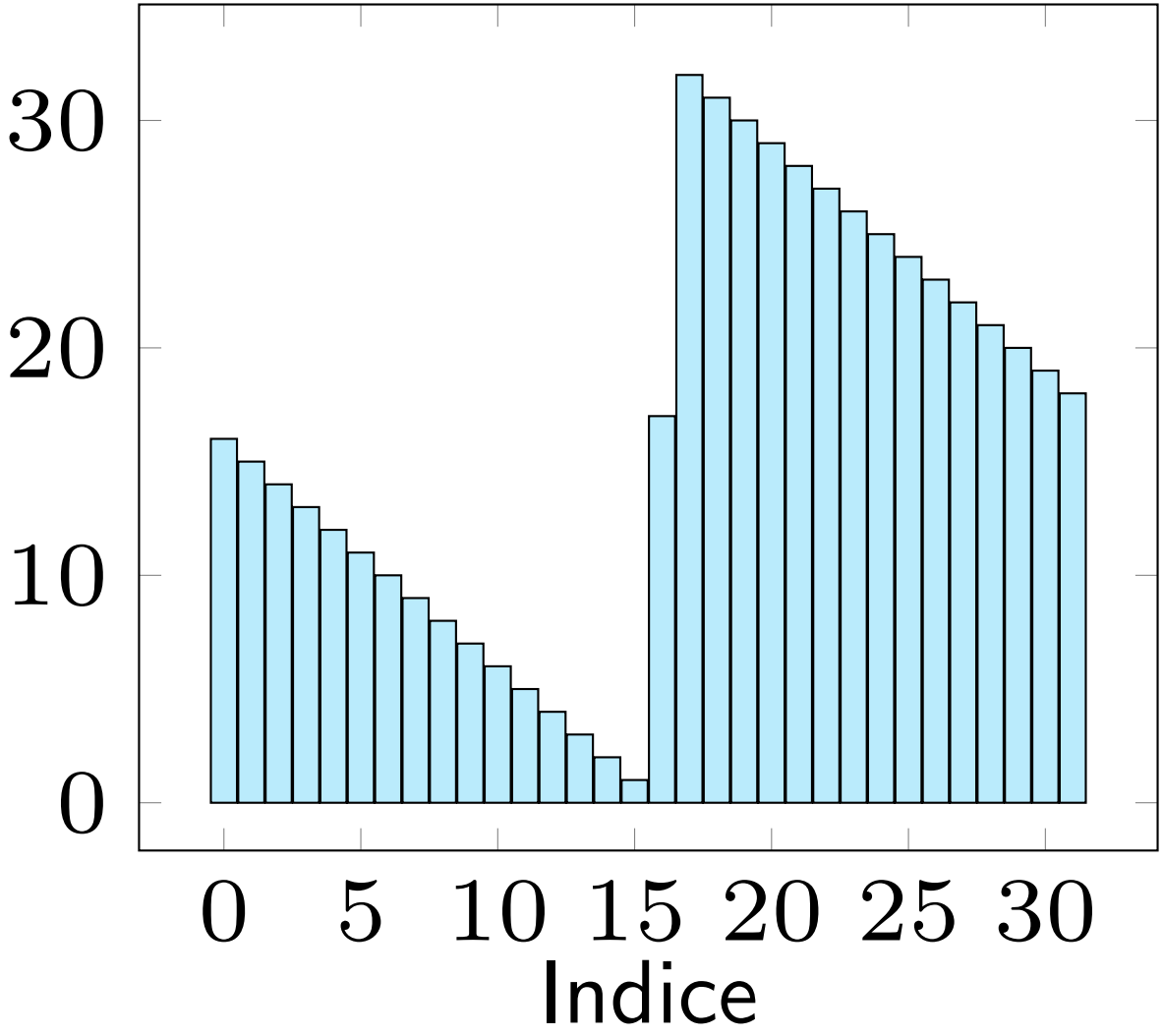
- Les programmeurs oublient souvent la complexité dans le pire cas de leur algorithme.
- Les attaquants rappellent aux programmeurs le pire cas !

Cas de `qsort` (FreeBSD 8.1.0)

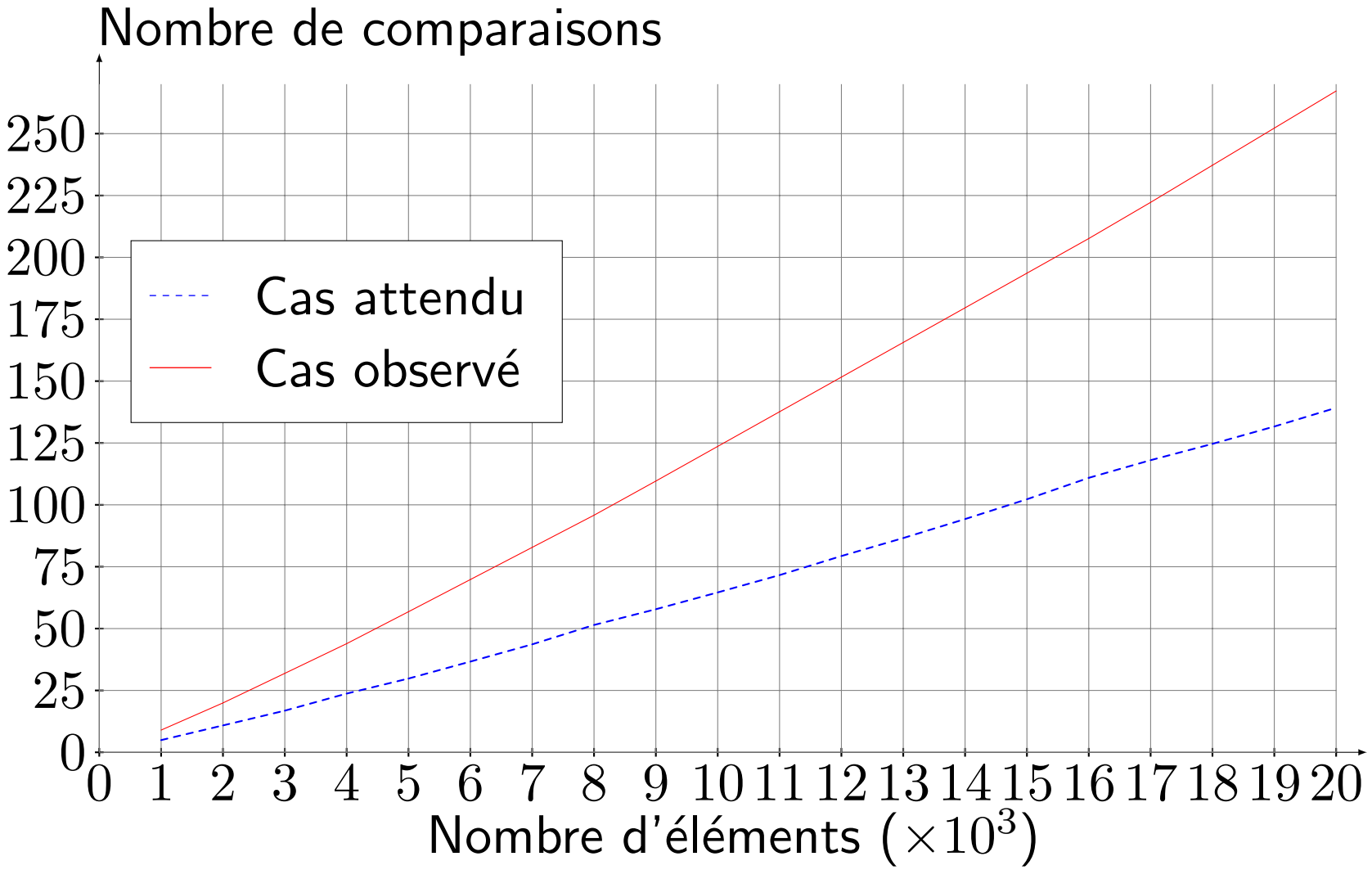


Comment atteindre le pire cas ?

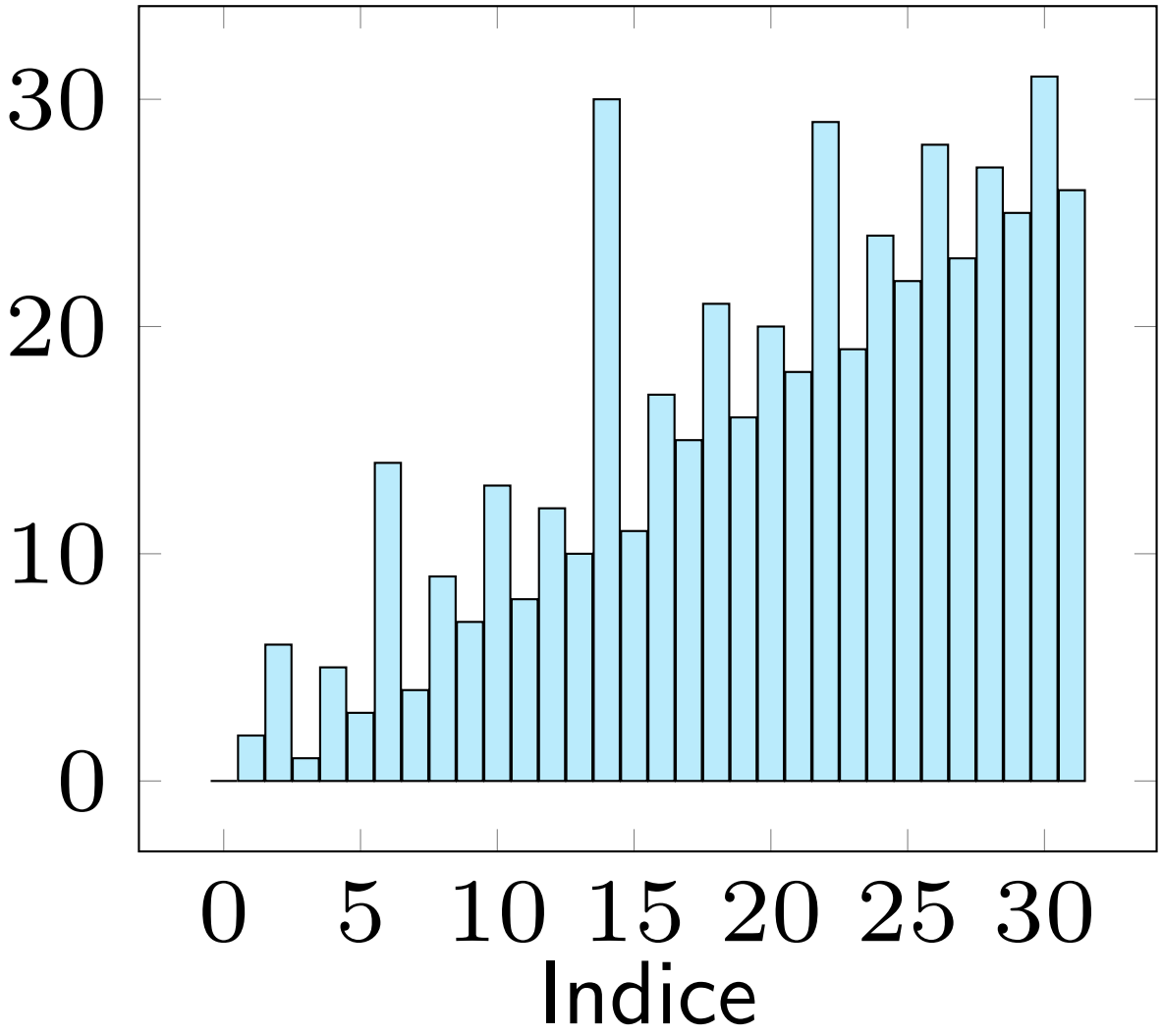
[McIlroy]



Cas de `qsort` (`glibc` 2.19)



Profil



Ça sert a quoi ?

- **Attaques contre les applications Web :**
  - base de données (**[Crosby et Wallach]**)
- **Système d'exploitation :**
  - Ordonnancement
  - Routage
- **Attaques contre les filtres :**
  - Intrusion Detection System
  - Deep Packet Inspection
  - Anti-malwares

Google Safe Browsing

- **Mise en service** en 2008 pour les navigateurs :
  - Google Chrome
  - Mozilla Firefox
  - Apple Safari
  - Opera
- Conçu par le *Safe Browsing group* de Montréal.
- **Impact :** plus d'un milliard d'utilisateurs selon Google
- **Objectifs :** prévenir les utilisateurs d'atteindre des sites de
  - *phishing*
  - *malwares*
- **Approche :** blacklist
- API compatible avec C#, Python et PHP

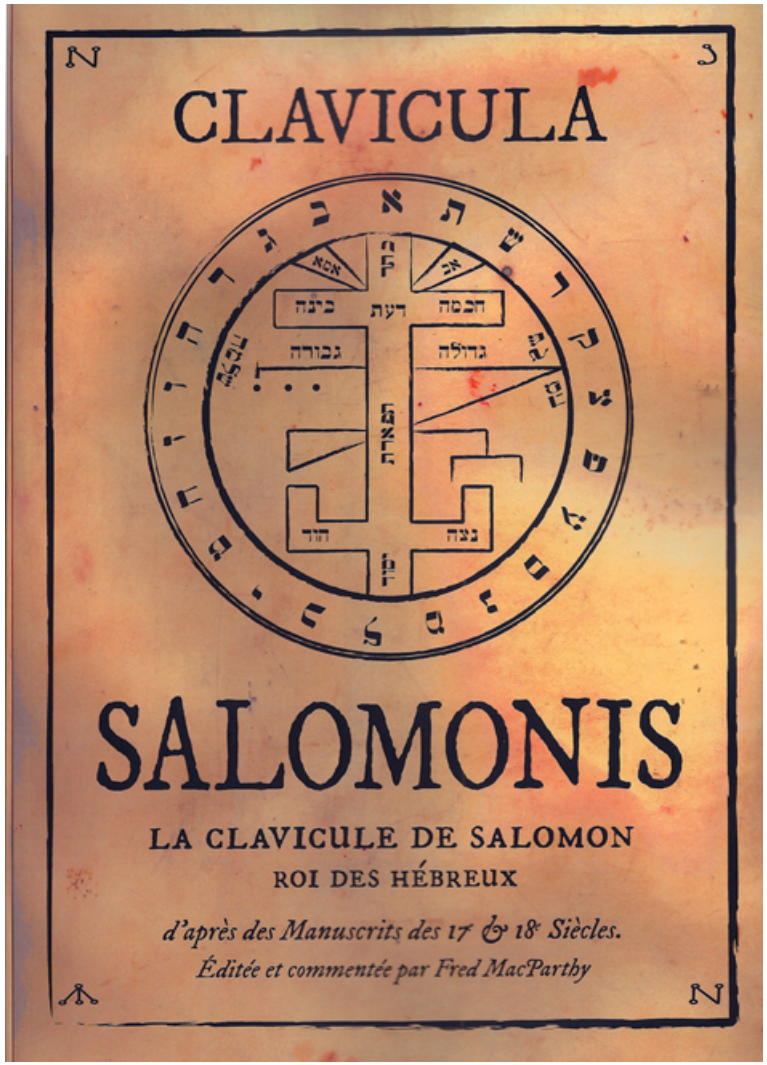
Safe Browsing Lookup API

- Google **crawle** le web à la recherche des sites de phishing ou de malwares pour **maintenir une blacklist** sur ces serveurs.
- **Utilisation la plus simple :** on demande aux serveurs de Google si un site est malicieux avec un simple HTTP GET.

```
https://sb-ssl.google.com/safebrowsing/api/lookup?
```

- **Problèmes :**
  - passage a l'échelle mauvais
  - problème de protection de la vie privée

La première blacklist de l'histoire



72 Démons de Salomon

Agares

Aim

Alloces

Amdusias

Amon

Amy

Andras

Andrealphus

Andromalius

⋮

Comment reconnaître un démon ?

- **Problème :** tout le monde ne peut pas disposer de la clavicule de Salomon dans sa poche. **Comment faire alors ?**

- **Solution :** faire de la **compression avec perte**.

Ag
Ai
Al
Am
An
⋮

- On passe de 72 noms à 50 préfixes (30% de compression).
- On passe de 518 caractères à 150 (70% de compression).

Les faux positifs

- `Hollande` $\rightarrow$ `Ho` n'est pas dans le dictionnaire. Donc `Hollande` n'est pas un démon.

- `Vals` $\rightarrow$ `Va` **est dans le dictionnaire**. Pourtant `Vals` n'est pas dans la liste complète. C'est un faux positif !

- Si une valeur est dans la liste raccourcie, il faut **vraiment ouvrir la Clavicule de Salomon** pour avoir la liste de tous les démons. Pour `Va`, on aurait : `Valefar`, `Vapula` et `Vassago`.

- **La solution est intéressante si on a peu de faux positifs.**

- Si tout est clair, il est temps de revenir a Safe Browsing.

Safe Browsing API v3

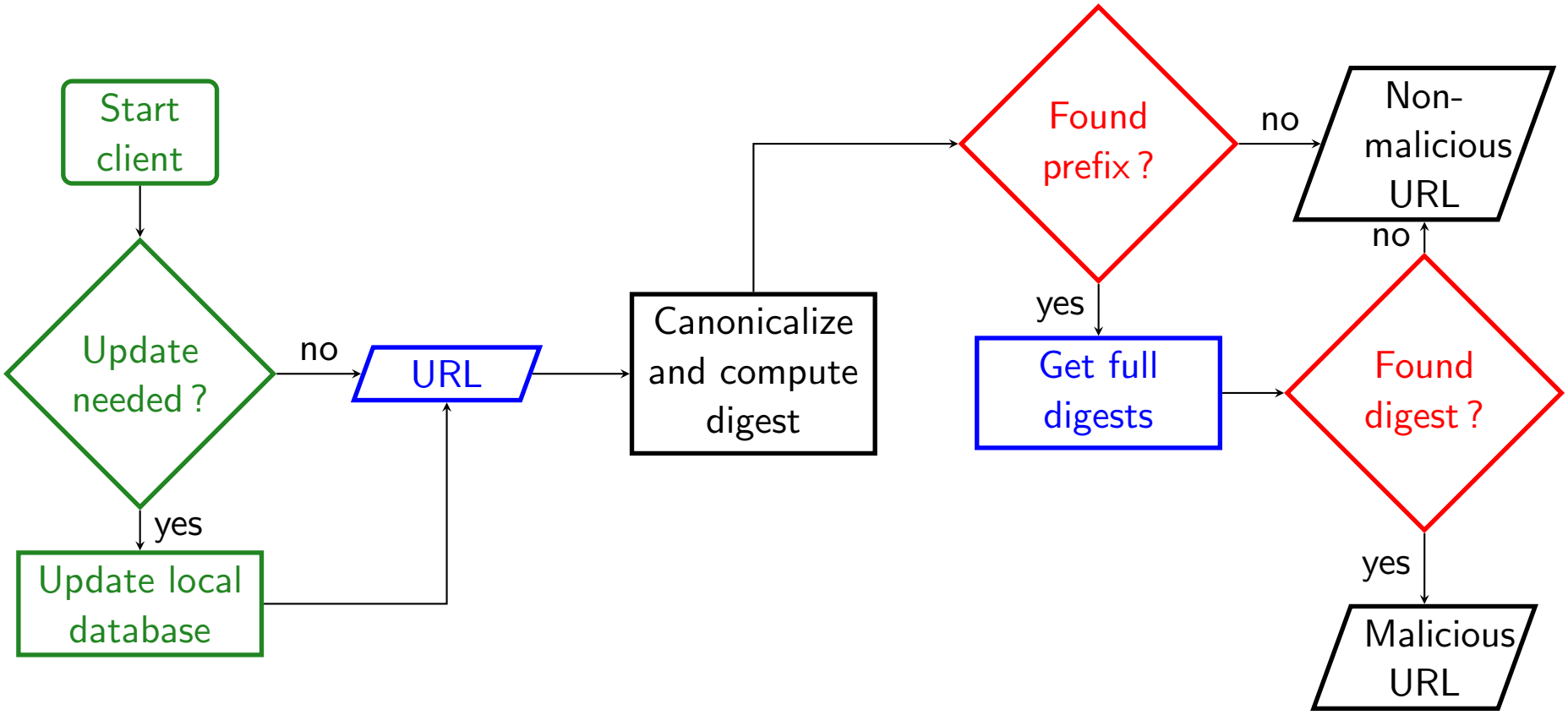
- Une partie de la vérification est faites en local dans 2 fichiers :
  — `googpub-phish-shavar`
  — `goog-malware-shavar`

- **Environ 650000 entrées au total.**

- On ne travaille pas avec directement sur les URLs. On utilise les 4 premiers octets de l'empreinte SHA-256 de ces dernières (32 octets).

  `Prefix32(SHA256(www.example.com/))=0xd59cc9d3`

Safe Browsing API v3



Caractéristiques de la blacklist

# Empreintes/prefix	Malwares	Phishing	Total
0	23	178	201
1	176191	459554	635683
2	2	22	42

- Il y a **223 URLs** du TopAlexa qui provoquent des **faux positifs**.

La blacklist réduite locale

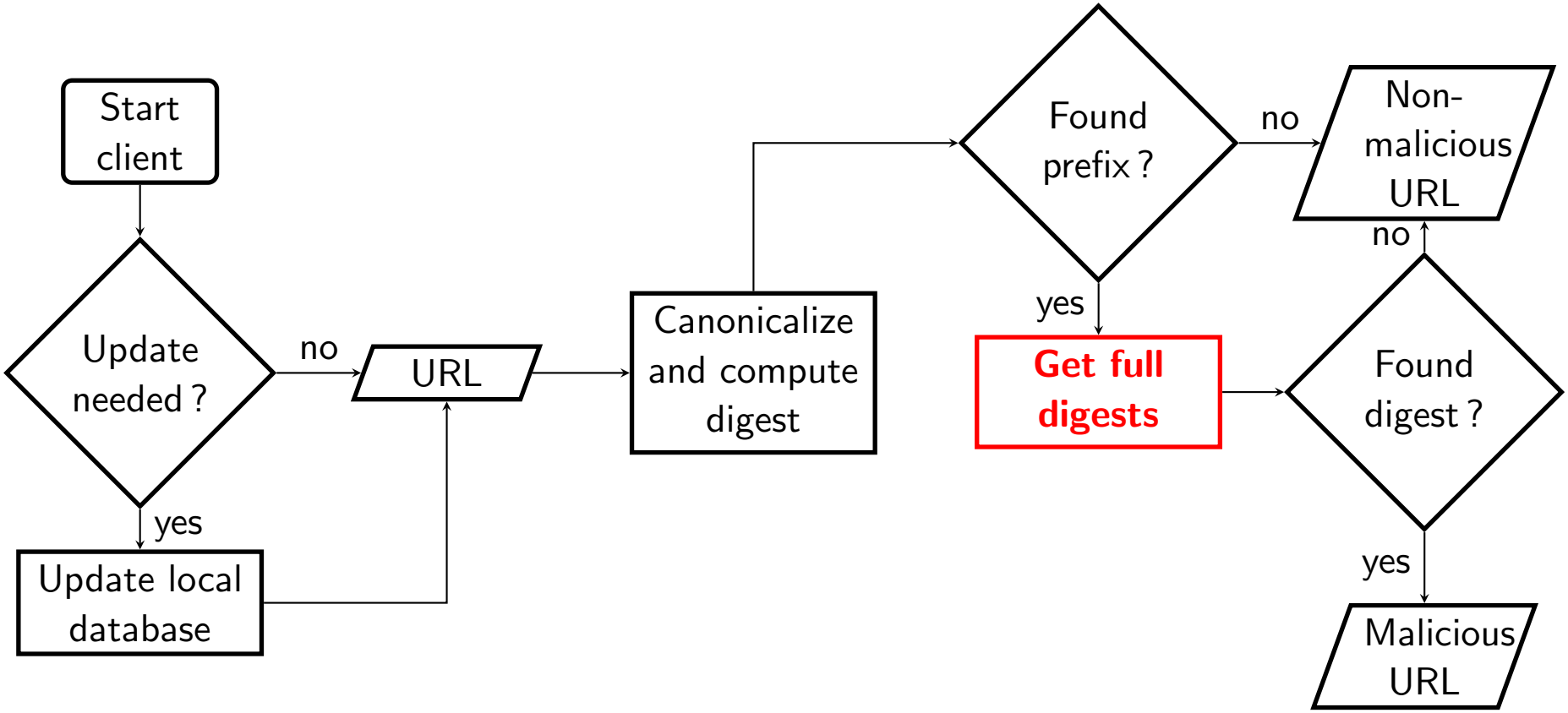
- **Implantation dans Chromium :**
  - codage delta
  - filtre de Bloom (abandonné)

		Structure (Mo)			
		Codage Delta		Filtre de Bloom	
Préfixe (bits)	**Don. brutes**	**taille**	**Compr.**	**taille**	**Compr.**
32	2. 5	1.3	1.9		0.8
64	5.1	3.9	1.3		1.7
80	6.4	5.1	1.2	3	2.1
128	10.2	8.9	1.1		3.4
256	20.3	19.1	1		6.7

- **La taille est le point critique !**

Que retenir de tout ce que je viens de raconter ?

- La solution est intéressante si on a peu de faux positifs.



Genèse de l'attaque

- **Acte I :** engendrer des faux positifs. Par exemple, `Hollande` est un nom souvent recherché. On cherche des noms qui ont la même empreinte :

  `Hochart`
  `Houssin`
  `Hoareau`
  `Hocquet`
  `Horn`
  ⋮

- **Acte II :** transformer les noms en démon et les faire ajouter à la Clavicule de Salomon.

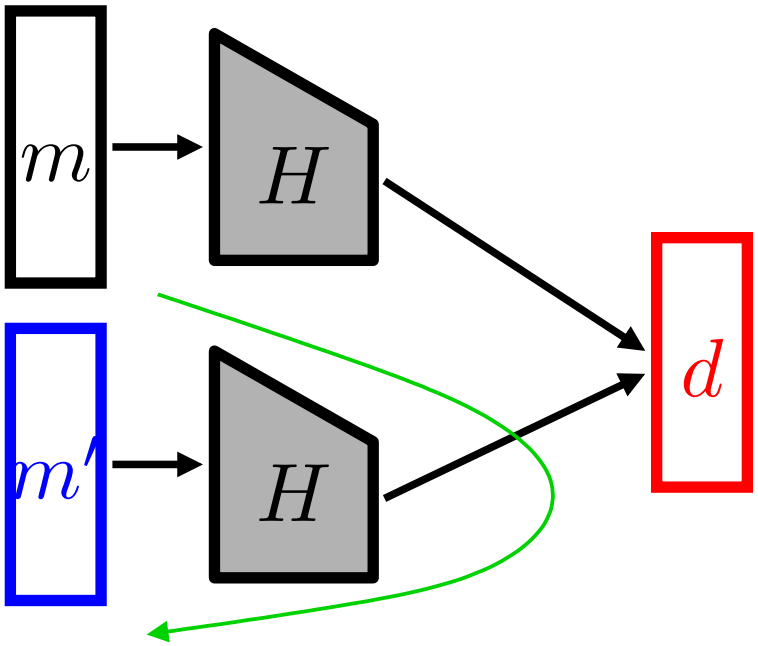
- **Acte III :** constater le résultat.

Acte I : Faux positifs

Seconde pré-image

- Étant donné une URL m, trouver $m' \neq m$ tel que :

$$\texttt{Prefix32(SHA256(}m\texttt{))} = \texttt{Prefix32(SHA256(}m'\texttt{))}$$



- 2^{32} calculs pour trouver un tel m'.

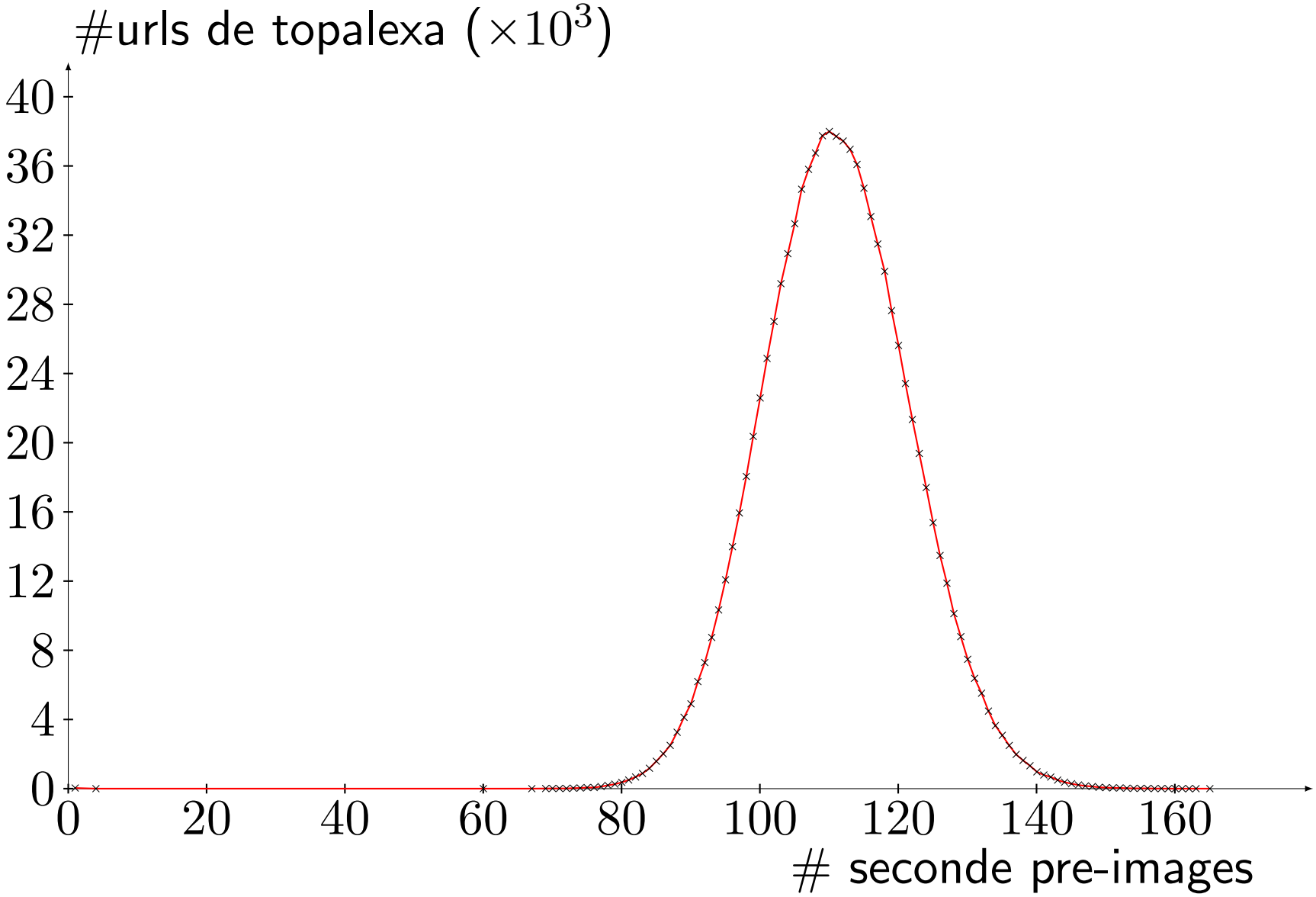
Acte I : Faux positifs

Cible

- Top 10^6 d'Alexa (`www.alexa.com/topsites`)

- Un mois de calcul sur 32 nœuds :
  - `Python`
  - `fake-factory 0.4.2`

Seconde pré-image



Seconde pré-image

Prefix	#	Alexa Site
0xd8b4483f	165	http://getontheweb.com/
0xbbb9a6be	163	http://exqifm.be/
0x0f0eb30e	162	http://rustysoffroad.com/
0x13041709	161	http://meetingsfocus.com/
0xff42c50e	160	http://js118114.com/
0xd932f4c1	160	http://cavenergie.nl/

- **Exemples d'URLs produites :**
  - http://62574314ginalittle.org/
  - http://chloekub.biz/id9352871

Acte II : inscription

- **Rapporter leurs existences à Google :**
  - `www.google.com/safebrowsing/report_badware/`
  - `www.google.com/safebrowsing/report_phish/`

- **Rapporter leurs existences des sources de Google :**
  - `www.phishtank.com/`
  - `www.stopbadware.org`

- **Google Webmaster tools**

- Enregistrement est la partie la plus discutable de nos travaux.

Conséquences

- Augmentation du trafic vers `sb-ssl.google.com`.
- **Discount :** 4 octets envoyés, 5280 reçus.

	Amplification
Pire cas	2
Cas moyen	800
Meilleur cas	1320

- **Bonus :** pollution du cache du navigateur !
  - Un préfixe ne peut être demandé que toutes les 45 min.
  - Le navigateur doit conserver la liste de toutes les empreintes correspondantes dans un cache pendant 45 min.
  - **Consommer de la mémoire !**

- Pas besoin de `botnet`.
- Uniquement bien choisir l'URL de ces malwares.

Conclusion

Notification à Google

We have notified the team about this issue ; they will review your report and decide whether they want to make a change or not. Thanks for letting us know. Regarding our Vulnerability Reward Program, the panel decided this issue has very little or no security impact, and therefore we believe that it is not in scope for the program, so we won't be issuing a reward or an entry in the hall of fame at this time.

Références

- **The Power of Evil Choices in Bloom Filters.** T. Gerbet, A. Kumar et C. Lauradoux. Rapport de recherche INRIA 8627, nov. 2014, `https://hal.inria.fr/hal-01082158`.

- **(Un)Safe Browsing**. T. Gerbet, A. Kumar et C. Lauradoux. Rapport de recherche INRIA 8594, sept. 2014, `https://hal.inria.fr/hal-01064822`.

- **You Can't See Me : Invisible Network Flows**. A. Kumar et C. Lauradoux. Rapport de recherche INRIA XXXX, nov. 2014.