Safe Browsing Services: to Track, Censor and Protect

Thomas Gerbet, Amrit Kumar, Cédric Lauradoux

Safe Browsing Services: to Track, Censor and Protect

Thomas Gerbet, Amrit Kumar, Cédric Lauradoux

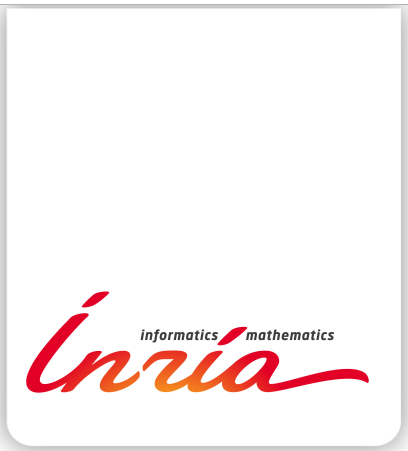
RESEARCH REPORT

N° 8686

February 2015

Project-Teams PRIVATICS

ISSN 0249-6399 ISRN INRIA/RR--8686--FR+ENG



Safe Browsing Services: to Track, Censor and Protect

Thomas Gerbet, Amrit Kumar, Cédric Lauradoux

Project-Teams PRIVATICS

Research Report n° 8686 — February 2015 — 20 pages

Abstract: Users often inadvertently click phishing or malware website links, and in doing so they sacrifice secret information and sometimes even fully compromise their devices. In light of the increasing number of such URLs, new strategies have been invented to detect them and prevent users from harm. At the forefront lies the *Safe Browsing* mechanism, which identifies unsafe URLs and notifies its users in real-time. In this work, we show how Safe Browsing services work and how they put user's privacy at stake. We study the privacy of high impact Safe Browsing services including Google and Yandex, and demonstrate that Google's privacy policies are incorrect when they claim that Google can not recover URLs visited by its users from its Safe Browsing service. It implies that Google and Yandex Safe Browsing can be used to track specific classes of individuals or to censor websites. Our investigations on the data currently included in Yandex Safe Browsing show that some URLs can already be re-identified.

Key-words: Malware, phishing, surveillance

RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Safe Browsing Services: to Track, Censor and Protect

Résumé : Les utilisateurs cliquent parfois par inadvertance sur des sites de phishing ou de malware et ce faisant ils sacrifient des informations secrètes et parfois même compromettent leur sécurité. Les créateurs de ces sites sont très motivés et sont capables de construire des URLs intelligemment scriptées. Compte tenu du nombre sans cesse croissant de tels URLs, de nouvelles stratégies ont été inventées pour les détecter et informer l'utilisateur final quand il est tenté d'accéder à un tel lien. Au premier rang de ces techniques se trouve les services de Safe Browsing, qui interrogent une liste noire de sites dangereux et qui avertissent les utilisateurs leur permettant de se protéger.

Dans cet article, nous montrons comment les services Safe Browsing endommagent la vie privée de leurs utilisateurs. Nous étudions la vie privée de systèmes Google et Yandex pour comprendre quelles informations ils peuvent recueillir. Nous avertissons leurs utilisateurs que ces les services peuvent être facilement utilisés pour la surveillance de suivre des classes spécifiques d'individus ou de censurer certains sites. Google et Yandex ne sont pas actuellement responsables des entrées inclus dans leur liste noire. Jusqu'à une avancée majeure dans le domaine de la recherche d'informations privées, les services de Safe Browsing ne peuvent pas être respectueux de la vie privé. Nous fournissons des preuves que Yandex a toutes les sondes en place pour suivre les utilisateurs de sites Web ciblés.

Mots-clés : Malware, phishing, surveillance

1 Introduction

Communications purporting to be from popular social web sites, payment processors or IT administrators are rampantly used to lure unsuspecting public. These attempts known as *phishing*, aim to acquire sensitive information such as passwords and credit card details. In general, these attacks rely on seemingly authentic URLs to entice a user to transmit sensitive data to the back-end malicious entities or to install harmful piece of code, aka *malware* on his device.

These malicious URLs are often hard to be visually detected, whence the *raison d'être* of a reliable, automated and robust way of detecting them. *Safe Browsing* is such a service that enables browsers to test a URL against suspected phishing or malware pages' database. The browser can then warn the user about the potential danger of visiting the page. Essentially, the browser computes digests of the provided URL and checks if the 32-bit prefixes of the digests match a local database. If there is a hit, a server is queried to eliminate errors.

All the major browsers such as Chrome, Firefox, Internet Explorer, Safari, Opera and Yandex.Browser include Safe Browsing as a feature. Furthermore, the service has a large user base, for instance, Google claims more than a billion users of its Safe Browsing services until date [Inc14].

In this paper, we investigate high impact Safe Browsing services and in particular, we describe several privacy issues with Google and Yandex. To this end, we provide an overview of the existing Safe Browsing services in Section 2 and present some popular services which are by design privacy unfriendly, in the sense that they know the webpages visited by a user. Most prominent among such services include Microsoft SmartScreen Filter, Web of Trust, Norton Safe Web and McAfee SiteAdvisor. We then provide a comprehensive description of Google and Yandex Safe Browsing in Section 3 and Section 4 respectively. Our first contribution consists in demonstrating how URLs can be leaked to Google and Yandex (Section 5). We use balls-into-bins arguments to analyze the anonymization scheme of Google and Yandex. We show that the k-anonymity property hence achieved does not hold in certain cases. We also provide a countermeasure to restore some sort of k-anonymity.

Our second contribution is the analysis of the current databases of Google and Yandex (Section 6). We exhibit several URLs which can be re-identified by Yandex: it can know if a user prefers the adult site `www.wickedpictures` or `www.mofos` because they both trigger Safe Browsing (false positive) and communicate too much information to the server. By crawling the database of Google and Yandex servers, we obtain a list of "suspicious" prefixes that we call *orphans*. Orphans trigger communications with the Google and Yandex server but, no full digest corresponds to them. They can be the result of an inconsistency between the server and the client or they might have been deliberately inserted to track specific URLs. The percentage of orphans is rather low for Google but very high for some databases of Yandex. We even observe that some very sensitive URLs have been blocked by Yandex. It would be very easy for Google and Yandex to censor a URL by including it in Safe Browsing. This is the case for the NGO, *Secours Islamique*'s webpage blocked by Yandex.Browser but not by more popular ones including Chrome, Chromium, Firefox, Opera or Safari.

2 Safe Browsing: An Overview

The essential goal of any Safe Browsing (SB) mechanism is to warn and dissuade an end user from accessing malicious URLs. All the existing SB services filter malicious links by relying either on a dynamic blacklist of malicious URLs, domains and IP addresses or a whitelist of *safe-to-navigate* websites. The dynamic nature of this list incorporates the fluctuating behavior of malicious

domains: a safe-to-navigate domain transforming into a malicious one and vice versa. Since SB features are included in the browsers, the service has been implemented in a way that it does not affect the browser's usability. The robustness of the service relies on fast lookup data structures which may generate *false positives*: a non-malicious URL getting detected as a malicious one.

In addition to providing the basic warnings to the users, the SB services often provide webmaster tools. These tools allow a user to report malicious links unknown to the service. The submitted link upon analysis may get included in the blacklists. Symmetrically, an administrator may explicitly ask the service to remove the malicious flag from a given domain.

Since the inception of the GOOGLE Safe Browsing in 2008, other prominent web service providers have proposed similar solutions. This includes YANDEX Safe Browsing [Yan], MICROSOFT *SmartScreen URL Filter*, *Web of Trust* [WS], Norton *Safe Web* [Symb] and McAfee *SiteAdvisor* [McAa].

In the following sections, we discuss the SB services which agree that they recover each URL visited by a user. These are by design privacy-unfriendly and can track each user and monitor his activities over the web. This includes MICROSOFT SmartScreen filter, Web of Trust, Norton Safe Web and McAfee SiteAdvisor. We postpone the discussion of the remaining two services namely by GOOGLE and YANDEX to Section 3 and Section 4 respectively, since they somewhat try to anonymize the requests recieved by the client.

2.1 Microsoft SmartScreen Filter

SmartScreen filter [Mica] was first introduced as a malware and phishing filter in Internet Explorer (IE)8 and is disabled by default. It is a reputation based system checking URLs against a whitelist. It requires the client to send the URL to the server and obtain a response. It also checks downloaded files: the SHA-256 digest of the file is sent to the server. Since the release of Windows 8, SmartScreen has become a system feature of Windows, and hence is not solely restricted to IE. With the advent of IE9, a new feature has been added to the SmartScreen: *SmartScreen Application Reputation*. It aims to protect users from unknown malware programs. This software reputation scheme is backboned by the *Authenticode signatures* [Micb, Micc].

MICROSOFT's privacy policy for IE reads: "*When you use SmartScreen Filter to check websites automatically or manually, the address of the website you are visiting will be sent to* MICROSOFT *with standard computer information and the SmartScreen Filter version number. Information that may be associated with the address, such as search terms or data that you entered in forms, might be included.*" For files, MICROSOFT has a similar statement. MICROSOFT asserts that the afore-mentioned information is only used to analyze performance and to improve the service. However, this is the only guarantee available to a user.

2.2 Web of Trust

Web of Trust (WOT) [WS] is a free browser addon released in 2007. The addon makes query to a reputation database to know whether or not a website is trustworthy. It uses several sources including Phishtank (`phishtank.com`), GOOGLE and YANDEX Safe Browsing. The addon automatically creates a random identifier for a user. At each query, this identifier, the date and time of the request and the host name of each site a user visits are sent to the WOT servers to recover the rating information. WOT can track a user with the data that it collects.

2.3 Norton Safe Web

Norton Safe Web (NSW) [Symb] is developed by Symantec and delivers information about websites based on automated analysis and user feedback. The feature is included in its product Norton

Internet Security and Norton 360. A limited standalone version of Safe Web, known as Norton Safe Web Lite is also available as a freeware. Safe Web operates as a browser plugin and requires IE6 or Firefox 3 or later or Chrome. Recently, Symantec has teamed up with Ask.com to provide on top of the Safe Web, a search environment called Norton Safe Search (NSS) [Syma] which offers visual site ratings within search results. Even when NSS is disabled, NSW continues to annotate searches on the default search engine.

Symantec asserts that: "*Ask.com collects information including: IP address, the origin of the search ... and may share this information with a third party*".

2.4 McAfee SiteAdvisor

McAfee SiteAdvisor (SA) [McAa] is a plugin available for Windows (Firefox, Internet Explorer and Chrome) and for Mac OS (Firefox and Safari). An extended and paid version of the software comes with SA Live [McAc]. With the SA Live software, even emails and instant messages get checked for malicious links. It may also scan downloaded files for threats. Depending on the chosen protection level, download protection will stop the download and warn users of the risks. Protection against malicious downloads is not available in the standard SA freeware. SA employs a lookup mechanism to contact the rating server [McAb] and generates a log file which is sent to the ePolicy Orchestrator for reporting purposes. SA currently sends logs only through HTTP which can be easily eavesdropped.

As a conclusion, these services are not privacy friendly: the URL or a part of it is sent in clear to the servers. To our knowledge, Google and Yandex provide the only SB services with built-in privacy features.

3 Google Safe Browsing

Google Safe Browsing (GSB) [sba] is the most widely used and the most robust of all. According to a 2012 report [Pro], Google detects over 9500 new malicious websites everyday and provides warnings for about 300 thousand downloads per day. Fig. 1 shows a simplified architecture of the service. Google crawlers continuously harvest malicious URLs on the web and then transmit them to the GSB servers. Clients can then consult the server to check if a link is malicious.

GSB classifies malicious URLs into two lists: malware and phishing. The names and the number of entries per list is given in Table 2 together with the lists provided by Yandex. They contain SHA-256 digests of malicious URLs and can either be downloaded partially to only update the local copy by adding new hashes and removing old ones or can be downloaded in its entirety. The lists can be accessed by the clients using two different APIs, software developers choose the one they prefer according to the constraints they have. In the sequel, we describe in detail these APIs.

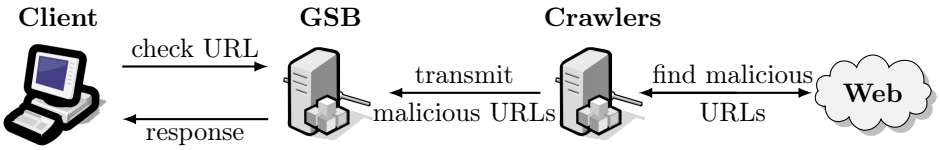


Figure 1: High level overview of Google Safe Browsing.

3.1 Lookup API

The Lookup API is a simple interface to query the state of a URL. Clients send URLs they need to check using HTTP GET or POST requests and the server's response contains a direct answer for each URL. The response is generated by looking up in the malicious lists stored on the distant server. This is straightforward and easy to implement for developers, but has drawbacks in terms of privacy and performance. Indeed, URLs are sent in clear to the servers and each request implies latency due to the network round-trip. To solve these issues, Google offers another API: Google Safe Browsing API described below.

3.2 Safe Browsing API

The Google Safe Browsing API is now the reference API to use GSB. It was positively received by the community as a major improvement for the privacy.

In contrast to the Lookup API, the client now does not handle a URL directly. Instead, the URL is canonicalized following the URI specifications [BLFM05] and hashed with SHA-256 [Nat12]. Digest is then checked against a locally stored database which contains 32-bit prefixes of malicious URL digests. If the prefix is not found to be present in the local database, then the URL can be considered safe. However, if there is a match, the queried URL is not necessarily malicious: it can be a false positive. Consequently, the client must query the GSB server to get all the full digests corresponding to the given prefix. Finally, if the full digest of the client is not present in the list returned by the server, the URL could be considered safe. Fig. 2 summarizes a request through the GSB API.

We note that the lookup requires the client to test several URLs for a given URL. This is because the complete URL might not have been included in the blacklists. For instance, in order to check whether `http://a.b.c/1/2.html?param=1` is malicious, the client will lookup for the following 8

decompositions:	`a.b.c/1/2.html?param=1`	`b.c/1/2.html?param=1`
	`a.b.c/1/2.html`	`b.c/1/2.html`
	`a.b.c/`	`b.c/`
	`a.b.c/1/`	`b.c/1/`

If any of the above URLs creates a hit in the local database, then the initial link is considered as suspicious and the prefix can be forwarded to the Google server for a confirmation. If there are more than 1 hits, then all the corresponding prefixes are sent. After receiving the list of full digests corresponding to the suspected prefixes fragments, they are locally stored until an update discards them. Storing the full digests prevents the network from slowing down due to frequent requests. To maintain the quality of service and limiting the amount of resources needed to run the API, Google has defined for each type of requests (malware or phishing) the frequency of queries that the clients must restrain to.

3.3 Local Data Structures

The choice of the data structure to store the prefixes is constrained by two factors: fast query time and low memory footprint. Google has deployed two different data structures until now: *Bloom filters* [Blo70] and *Delta-coded tables* [MKD+02].

In early versions of Chromium (discontinued since September 2012), a Bloom filter was used to store the prefixes' database on the client's side. This solution was abandoned to be replaced by delta-coded tables. Unlike the classical Bloom filter, this data structure is dynamic, does not have any "intrinsic" false positive probability and yet incurs less memory. However, its query time is slower than that of Bloom filters. Even though the delta-coded table does not entail false

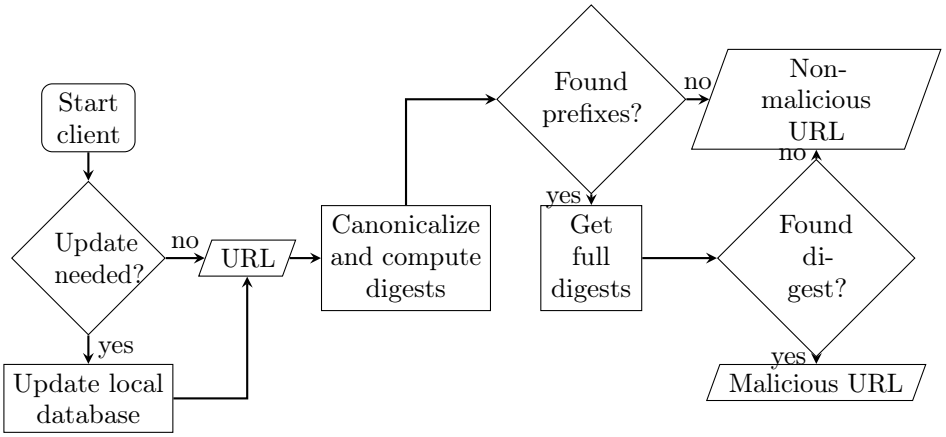


Figure 2: Google Safe Browsing API: Client's behavior flow chart.

positives, its use to store 32-bit prefixes indeed leads to false positives. False positives arise since several URLs may share the same 32-bit prefix.

We have implemented all the data structures to understand why Bloom filters were abandoned and why Google has chosen 32-bit prefixes. The results are shown in Table 1. If 32-bit prefixes are stored, the raw data requires 2.5 MB of space. Storing these prefixes using a delta-coded table would only require 1.3 MB of memory, hence Google achieves a compression ratio of 1.9. For the same raw data, a Bloom filter would require 3 MB of space. However, starting from 64 bit prefixes, Bloom filter outperforms delta-coded table. This justifies Google's choice of delta-coded tables over Bloom filters and the choice of 32-bit prefixes.

Table 1: Client's cache size for different prefix sizes.

		Data structure (MB)			
		Delta-coded table		**Bloom filter**	
Prefix size (bits)	**Raw data (MB)**	**size**	**comp. ratio**	**size**	**comp. ratio**
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	1.06		6.7

3.4 Safe Browsing Cookie

GSB has been criticized since the beginning because when implemented inside web browsers, each request to the API also sends a cookie which identifies a user. The cookie sent by the browsers is the same as the one used by other services provided by Google especially the social features such as the *+1 button*. To these criticisms, Google responded that the cookies were not used to track users but only to monitor the performance of the service on the server-side and to catch bugs.[1] Since they are needed by Google to operate the service, the browsers can not disable it.

[1] `https://code.google.com/p/chromium/issues/detail?id=103243`

Chromium and Firefox have decided to isolate the SB cookie from the others with the purpose of achieving maximum privacy.[2]

4 Yandex Safe Browsing

Yandex Safe Browsing (YSB) service comes in the form of an API [Yan] and also as a security feature in its browser called *Yandex.Browser*. The Yandex Safe Browsing API is compatible with C#, Python and PHP and is a verbatim copy of the GSB API with the only difference that in addition to the phishing and the malware lists provided by Google, the API also includes 17 other blacklists. Each of these lists contains malicious or unsafe links of a given category.

Table 2: Blacklists provided by Google and Yandex. The information could not be obtained for cells marked with (*).

	List name	**Description**	**#prefixes**
Google	goog-malware-shavar	malware	317807
	goog-regtest-shavar	test file	29667
	goog-whitedomain-shavar	unused	1
	googpub-phish-shavar	phishing	312621
Yandex	goog-malware-shavar	malware	283211
	goog-mobile-only-malware-shavar	mobile malware	2107
	goog-phish-shavar	phishing	31593
	ydx-adult-shavar	adult website	434
	ydx-adult-testing-shavar	test file	535
	ydx-imgs-shavar	malicious image	0
	ydx-malware-shavar	malware	283211
	ydx-mitb-masks-shavar	man-in-the-browser	87
	ydx-mobile-only-malware-shavar	malware	2107
	ydx-phish-shavar	phishing	31593
	ydx-porno-hosts-top-shavar	pornography	99990
	ydx-sms-fraud-shavar	sms fraud	10609
	ydx-test-shavar	test file	0
	ydx-yellow-shavar	shocking content	209
	ydx-yellow-testing-shavar	test file	370
	ydx-badcrxids-digestvar	.crx file ids	*
	ydx-badbin-digestvar	malicious binary	*
	ydx-mitb-uids	man-in-the-browser android app UID	*
	ydx-badcrxids-testing-digestvar	test file	*

Table 2 provides the name and description of the blacklists with the number of prefixes present in each. We highlight that the lists `goog-malware-shavar` and `ydx-malware-shavar` provided by Yandex are identical. The same holds for the malware blacklists for phones: `goog-mobile-only-malware-shavar` and `ydx-mobile-only-malware-shavar`, and for phishing: `goog-phish-shavar` and `ydx-phish-shavar`. By comparing the `goog-malware-shavar` lists of Google and Yandex, we observed that there are only 36547 prefixes in common. Similarly, the phishing lists `googpub-phish-shavar` and `goog-phish-shavar` of Google and Yandex respectively have only 195 prefixes in common. These anomalies might be because the lists are not up-to-date on the Yandex server.

[2] `https://bugzilla.mozilla.org/show_bug.cgi?id=368255`

5 Tracking with GSB and YSB

Google Chrome Privacy Notice [Goo14] includes a section on Safe Browsing. It states that: "'*Google cannot determine the real URL from this information* (to be read prefixes)." This statement has been re-iterated by Google in a document concerning GSB usage in Mozilla Firefox [Goo12]. In this section, we show that this statement is incorrect: Google and Yandex can recover URLs in certain cases.

After motivating why Google and Yandex would turn Safe Browsing into a tracking service, we describe how information are leaked to their respective servers. We conclude this section by explaining how to mitigate this problem.

5.1 Motivations

Google has several open opportunities to include tracking technologies in its products, especially in Chrome. The motivation for Google to use Safe Browsing to track users is that its use is not restricted to Google products (Chrome and Google Analytics) but also in many other browsers including Chromium, Firefox, Safari and Opera. According to statcounter[3], these represent 65% of all the browsers in use. Twitter and Bitly also use GSB to prevent users from disseminating malicious URLs. Fig. 3 describes where GSB and YSB are used.

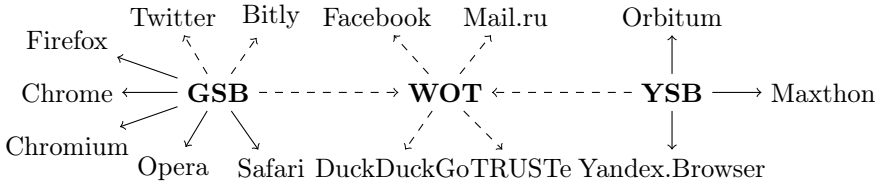


Figure 3: The Safe Browsing ecosystem.

The plain arrow in Fig. 3 indicates that the information sent to Google or Yandex are known. The users of the services or the softwares connected by a plain arrow are directly impacted by any Safe Browsing leaks. The dashed arrow indicates that we do not know the data communicated to the servers. The leakage depends on the cookie sent to Google by the companies (*e.g.* Twitter). It is interesting to observe that WOT uses both GSB and YSB as third-party sources and many services including Facebook use WOT to check URLs. Consequently, even when a user does not directly use a Google or Yandex product, there is still a chance that the user's data is sent to the backend Safe Browsing service.

5.2 k-Anonymity Argument

A simple way to understand GSB and YSB consists in considering them as a probabilistic test run by the browser to filter malicious URLs. When the test executed by the browser is positive, Google or Yandex are contacted to remove any ambiguity. While a negative test leaks no information about the URL to Google and Yandex, a positive test sends prefix(es) of the decompositions of the URL to the server. The exact number of prefixes sent depends on the number of hits in the local database. It is important to know if those prefixes are enough to recover the URL, hence allowing the service to learn about a user's activity.

[3] `statcounter.com`

Table 3: Decompositions of a URL and its different prefixes.

URL	32-bit prefix
esorics2015.sba-research.org/call-for-papers/	0xec158785
esorics2015.sba-research.org/	0x0fbdbaa8
sba-research.org/call-for-papers/	0x0dc8825f
sba-research.org/	0x52592f0d

Let us consider the example of esorics2015.sba-research.org/call-for-papers/. Its decompositions are shown in Table 3. We also suppose that the second decomposition creates a hit in the local database, i.e., 0x0fbdbaa8 is present in one of the lists. To determine if the prefix is enough for GOOGLE or YANDEX to recover the corresponding URL, we need to know the maximum number of URLs which share the same prefix. To this end, we use a result from Raab and Steger [RS98] on *balls-into-bins* to compute the maximum number of URLs that can share the same ℓ-bit prefix.

Theorem 1 (Raab and Steger [RS98]) *Let M be the random variable that counts the maximum number of balls into any bin. If we throw m balls independently and uniformly at random into $n = 2^\ell$ bins, then $Pr[M > k_\alpha] = o(1)$ if $\alpha > 1$ and $Pr[M > k_\alpha] = 1 - o(1)$ if $0 < \alpha < 1$, where:*

$$k_\alpha = \begin{cases} \frac{\log n}{\log \frac{n \log n}{m}} \left(1 + \alpha \frac{\log^{(2)}\left(\frac{n \log n}{m}\right)}{\log \frac{n \log n}{m}}\right), & \text{if } \frac{n}{\text{polylog}(n)} \leq m \ll n \log n, \\ (d_c - 1 - \alpha) \log n, & \text{if } m = c \cdot n \log n, \\ \frac{m}{n} + \alpha \sqrt{2 \frac{m}{n} \log n}, & \text{if } n \log n \ll m \leq n \cdot \text{polylog}(n), \\ \frac{m}{n} + \sqrt{\frac{2m \log n}{n} \left(1 - \frac{1}{\alpha} \frac{\log^{(2)} n}{2 \log n}\right)}, & \text{if } m \gg n \cdot (\log n)^3. \end{cases}$$

The equation for computing d_c can be found in [RS98].

If $n = 2^\ell$ is the number of possible prefixes and m is the number of unique URLs, then M is the maximum number of URLs matching a given prefix. M is also the maximum uncertainty (in terms of number of URLs) that GOOGLE or YANDEX can have over re-identifying the URL for a received prefix. It can be viewed as a k-anonymity argument [Swe02] to support the fact that the URLs are anonymized through hashing-and-truncation to ℓ bits. A more correct argument from the user's point of view would be to consider the bin having the least number of balls. However, we are taking the attacker's point of view: for GOOGLE and YANDEX, M is the worst case uncertainty for URL re-identification.

In 2008, GOOGLE [goo] claimed to know 1 trillion unique URLs. Since then, GOOGLE officials have reported 30 trillion URLs in 2012 and 60 trillion in 2013. These values are summarized in the first two row of Table 4. The table also presents the number of domain names provided by VERISIGN in its reports [Inc] and the number of active hostnames provided by NETCRAFT.

Using the previous theorem and the previous Internet facts, we compute the values M for unique URLs, domain names and active hostnames for different prefix sizes over the last few years. When GSB was started in 2008, at most 443 URLs matched a given 32-bit prefix. It has increased over the years to reach 14757 in 2013. In the worst case, it is hard for GOOGLE and YANDEX to re-identify a URL from a single 32-bit prefix. The case of domain names and active hostnames is slightly different because the space of domain names is much smaller and its dynamic is far slower than the one of URLs. In the worst case, two domain names will collide to the same prefix. Domain names can hence be re-identified with high certainty. However, the

server does not know if the received prefix corresponds to a domain name or to a URL. So far, the solution seems to be privacy preserving as long as we only reveal a single prefix.

Table 4: M for URLs and domain names with different prefix sizes ℓ.

	URLs (10^{12})			**domains** (10^6)			**hosts** (10^6)		
	2008	2012	2013	2008	2012	2013	2008	2012	2013
Number	1	30	60	177	252	271	70	185	184
ℓ (bits)	M **for URLs**			M **for domains**			M **for hosts**		
16	2^{28}	2^{28}	2^{29}	3101	4196	4498	1253	3123	3107
32	443	7541	14757	2	3	3	1	2	2
64	2	2	2	1	1	1	1	1	1
96	1	1	1	1	1	1	1	1	1

5.3 Tracking by URL Re-identification

In this section, we show that in certain cases, the afore-described k-anonymity argument does not hold. This implies that the data sent to the server is sufficient to re-identify certain URLs and hence can be used to track users. The crucial issue with GSB and YSB is that the server can receive several prefixes for a URL: multiple decompositions of a URL can create hits in the local database. Randomly choosing two URLs for which the decomposition matches exactly the same 2 prefixes is unlikely, with probability 2^{-64}. It implies that if there is more than one match for a URL, then it can be re-identified using its prefixes.

In fact, it is also possible to re-identify a URL even when only a single prefix among the decompositions matches the database. This can be achieved by exploiting the temporal correlation between the queries of a user. Yandex and Google can identify the requests of a given user thanks to the SB cookie. A user visiting: `esorics2015.sba-research.org/call-for-papers/` is very likely to visit the submission website: `esorics2015.sba-research.org/submission/`. Instead of looking at a single query, one would need to correlate two queries. A user making two queries for the prefixes `0xec158785` and `0x1626224a` in a short period of time is planning to submit a paper.

A tracking system based on GSB or YSB will work as follows. First, they build a shadow database of prefixes corresponding to at least two decompositions of the targeted URLs. Second, they insert/push those prefixes in the client's database. Google and Yandex can identify individuals each time their server receives a query with at least two prefixes present in the shadow database.

To illustrate our discovery let us use the URL of ESORICS conference Call-for-Paper (CFP) webpage. We assume that the corresponding prefix: `0xec158785` is included in the prefix list. As per Table 4, if the server receives this prefix, it can not tell for sure that the user has visited the CFP webpage. Let us now assume that `0xec158785` and `0x0fbdbaa8` (for `esorics2015.sba-research.org/`) are both in the prefix list. When a user enters the CFP URL, it will be decomposed into the four URLs as shown in Table 3. Two out of the four prefixes corresponding to the decompositions now match the database. The server will receive the two prefixes and re-identify the URL by querying a shadow database.

5.4 Hashing for Privacy

Rendering the tracking capability of SB useless implies that Google or Yandex can not distinguish the prefixes corresponding to a targeted URL from those of other irrelevant URLs.

To achieve these goals we need to create URLs with legitimate content sharing the same 32-bit prefixes as those in the blacklists proposed by the services. This essentially consists in finding *multiple (second) pre-images* over 32-bit prefixes. It restores the k-anonymity argument used earlier to some extend. The factor k of uncertainty depends on the number of multiple (second) pre-images found.

We have written a second pre-image search engine in Parallel Python. In order to ensure that the URLs are human readable, we have employed fake-factory [fak], a python package to generate fake but human readable URLs. Our scheme can be tuned to generate URLs for a given existing domain or for free domain names. Our experiments were performed on a cluster running Centos Linux 2.6.32 and is powered by an Intel QEMU Virtual CPU (cpu64-rhel6), with 32 cores running at 2199 MHz.

The goal of the search engine was to search for second pre-images for the 1 million popular webpages of the Alexa list. At the end of 1 week, the total number of second pre-images found was $111,027,928$. Since the Alexa list contains 1 million entries, as expected the number of second pre-images found per URL was highly concentrated around 100. For around 38000 URLs in the Alexa list, we found 110 second pre-images per URL. Finding second pre-images for a 32-bit prefix is clearly fast and inexpensive.

To give a concrete use case, we can create fake profiles on `xhamster` to protect the URL `http://fr.xhamster.com/user/kmille`, which corresponds to the channel of a user 'kmille'. This URL is listed in Appendix B since the prefixes of its decompositions `fr.xhamster.com` and `xhamster.com` are in a prefix list of Yandex. The channel 'kmille' on `xhamster` is not in the database but let us assume it is. In order to protect the privacy of a user visiting the channel, we create fakes profiles on `xhamster` which share the same prefix and decompositions. A typical example is: `http://fr.xhamster.com/user/1375396439` having the prefix `0xe235ec60`. The link has the same prefix as that of `http://fr.xhamster.com/user/kmille`. It took us only a few hours of computation to obtain the fake link. These fake profiles hide a user's visit to the channel since it is impossible for the SB servers to know from the prefixes only if a user has visited the authentic URL or one of the fakes.

It is interesting to notice that Google or Yandex can use the same idea to justify the inclusion of a prefix in the database. For a given targeted website, they can create several malware pages matching the site's prefixes. If one accuses them of tracking, they may still use the URLs of their malwares as an alibi.

6 Blacklist Analysis

As a first step in our analysis, we recover the prefix lists of Google and Yandex. We then use these lists to query their servers using the API. This allows us to obtain the list of all the full digests. Our second step is an attempt to identify the URLs which correspond to these prefixes. To this end, we harvested phishing and malware URLs, domains, and IP addresses from several sources and tested for their belonging to the blacklists of prefixes. The list of all our sources can be found in [Cor]. In total, we harvested 1240300 malware URLs, 151331 phishing URLs and 2488828 URLs of other categories. We then query GSB and YSB with those URLs. The results of our findings are shown in Table 5. The findings reveal that the reconstruction for Google prefixes is rather inconclusive, 5.9% for malwares and 0.1% for phishing websites. For Yandex the situation is better but the majority of the database still remains unknown.

To know how many URLs generate more than 2 hits in the blacklists, we have used the Alexa list of 1 million most popular websites (`bit.ly/1yhXcgL`) and the BigBlackList of malicious URLs. In case of the Alexa list, we found nothing for Google, i.e., all the URLs create at most

Table 5: Matches found with 1240300 malware websites, 151331 phishing websites and 2488828 URLs and IPs in BigBlackList (`http://urlblacklist.com/`).

		Malware list		**Phishing list**		**BigBlackList**	
	list name	#matches	%matches	#matches	%matches	#matches	% matches
Google	goog-malware-shavar	18785	5.9	351	0.1	6208	1.9
	googpub-phish-shavar	632	0.2	11155	3.5	816	0.26
Yandex	ydx-malware-shavar	44232	15.6	417	0.1	11288	3.9
	ydx-adult-shavar	29	6.6	1	0.2	33	7.6
	ydx-mobile-only-malware-shavar	19	0.9	0	0	17	0.8
	ydx-phish-shavar	58	0.1	1568	4.9	153	0.47
	ydx-mitb-masks-shavar	20	22.9	0	0	1	1.1
	ydx-porno-hosts-top-shavar	1682	1.6	220	0.2	11401	11.40
	ydx-sms-fraud-shavar	66	0.6	1	0.01	22	0.20
	ydx-yellow-shavar	43	20	1	0.4	8	3.8

1 hit in the malicious list by Google. The result for Yandex is left in Appendix B (Table 7). To summarize, we found 12 URLs for which 2 prefixes in the decomposition create hits in the malicious lists. One of these URLs creates a hit in `ydx-malware-shavar` while the remaining are hits in `ydx-porno-hosts-top-shavar`. Hence, these URLs are re-identifiable. Furthermore, Yandex can know when a user attempts to login to the adult site `m.mofos.com` and when he logs out. Yandex can also know the nationality of a person by the version of `xhamster` he visits. In case of BigBlackList, we found 103 URLs creating 2 hits in the Yandex prefix lists. Moreover, we found one URL which creates 3 hits and another one which creates 4 hits in the database. The complete list is available at `privatics.inrialpes.fr/~amkumar/misc.html`.

6.1 Orphan Prefixes

We now look for *orphan prefixes* in Yandex and Google Safe Browsing. We call an entry in the prefix list as orphan if no 256-bit digest matches it in the corresponding list of full digests. It is worth noticing that these prefixes cannot be termed as false positives. We also look for URLs in the Alexa list which generate an orphan prefix. Table 6 presents the results of our findings. Both Google and Yandex have orphans. While Google has 159 orphan prefixes, for Yandex the numbers are astonishingly high. 43% for `ydx-adult-shavar`, 99% for `ydx-phish-shavar`, 95% for `ydx-sms-fraud-shavar` and 100% of the prefixes in `ydx-mitb-masks-shavar` and `ydx-yellow-shavar` are orphans.

We did not find any URL in the Alexa list matching a Google orphan prefix. But there are 572+88 URLs of the Alexa list with one parent: the prefix matches one full digest. For Yandex, we found 271 URLs matching an orphan prefix and 20220 URLs with one parent.

The presence of orphan prefixes is very difficult to justify. Moreover, the behavior of a browser on these prefixes is not consistent. Some of the orphan prefixes are considered as false positives by Yandex while others are declared as true positives. There are three possible explanations to argue the presence of orphans. First, that there is inconsistency between the prefix lists and the corresponding lists of full digests. This could be due to a misconfiguration or the result of a development error. Second, that the services have intentionally noised the database in order to mislead attackers who may try to re-identify URLs from the prefixes. The last argument being that, Google and Yandex have tampered with their prefixes' database for tracking purposes. In order to track a user's activity on a webpage, Google and Yandex add an orphan prefix corresponding to the page. Consequently, whenever a user visits the webpage, the orphan prefix is sent to the server and hence Google and Yandex may track the user. We observed GSB for five months and the presence of orphans is persistent and their number is constant. We can not

Table 6: Distribution of prefixes provided by GOOGLE and YANDEX as the number of full hashes per prefix. There are 36 orphan prefixes in `goog-malware-shavar`. Collision with the Alexa list is also given.

		#full hash per prefix				#Coll. with Alexa list			
	list name	0	1	2	Total	0	1	2	Total
GOOGLE	goog-malware-shavar	36	317,759	12	317,807	0	572	0	572
	googpub-phish-shavar	123	312,494	4	312,621	0	88	0	88
YANDEX	ydx-malware-shavar	4,184	279,015	12	283,211	73	2,614	0	2,687
	ydx-adult-shavar	184	250	0	434	38	43	0	81
	ydx-mobile-only-malware-shavar	130	1,977	0	2,107	2	22	0	24
	ydx-phish-shavar	31,325	268	0	31,593	22	0	0	22
	ydx-mitb-masks-shavar	87	0	0	87	2	0	0	2
	ydx-porno-hosts-top-shavar	240	99,750	0	99,990	43	17,541	0	17,584
	ydx-sms-fraud-shavar	10,162	447	0	10,609	76	3	0	79
	ydx-yellow-shavar	209	0	0	209	15	0	0	15

however prove with certainty that they are actually tracking their users using orphan prefixes but the presence itself shows that they have all probes in place to do so.

6.2 Pseudo-censorship

The SB services by design block malicious URLs. This might not be considered as classical censorship, since despite the warnings, a user may still visit the webpage. We refer to it as *pseudo-censorship*. In the past, some incidents have occurred with GSB: some legitimate and safe websites were accidently blocked. The most famous example was BITLY[4], in October 2014. To (pseudo)censor a URL, the services push the prefixes into the client's database and store the full digests on their servers. Without the support from GOOGLE and YANDEX, it is hard to re-identify the URLs from the digests. Hence, it is difficult to know if these services abuse the mechanism to deliberately censor non-malicious websites.

We attempt to check if any such abuse occurs. We limit our investigation to the URLs from the Alexa list which trigger SB. We categorize these URLs into four categories: religion, war/crime, disease and sex. The result of our classification can be found in Appendix C(Table 8). Among these URLs, we discover several interesting cases. The webpage of the French NGO, *Secours Islamique* is a notable one. YSB considers their webpage `secours-islamique.org` as malicious. It is however considered as safe by GSB, WOT, MICROSOFT filter and others. We eventually found two records for `secours-islamique.org` at `stopbadware.org`. URLs: `~/ramadan-2011` and `~/projet` of the domain had been blacklisted by GOOGLE for 23 and 146 days in December and August 2013 respectively.

There are two explanations behind the presence of the NGO's webpage in the YANDEX malware list. Either, the list is not updated or the prefix has been intentionally kept, so as to censor the webpage. After contacting the administrators of *Secours Islamique*, we realized that they were unaware of the situation.

The root of the problem is that GOOGLE and YANDEX are the only ones to control the databases of dangerous websites and their prefixes. Until this situation is not changed, suspicion will stay on these services. All the prominent SB providers are involved in the `stopbadware.org` initiative. It advertises 1,610,355 malwares in their Clearinghouse database. However, this database can only be queried online and is not distributed. It limits the transparency of the initiative. A data sharing program exists but it is rather limited and comes with a lot of usage

[4] `https://twitter.com/bitly/status/526015781533134848`

restrictions. GOOGLE, YANDEX, MICROSOFT and the others must be made *accountable* for their blacklists. A step toward accountability consists in making the databases public.

7 Related Work

Browsers have always been under scrutiny especially for their privacy implications. Aggarwal et al. [ABJB10] provide an analysis of *private browsing mode* in Firefox, Internet Explorer, Chrome and Safari. They show that in most cases, private browsing does not hide the IP address and the browser fingerprint and that some browsers make half-hearted attempt if not at all to hide the public state of the browser. Several works on privacy enhancing techniques such as [SRG97] and Tor-based extensions such as [Per], [Rom] focus on hiding the client's IP address. *Doppelganger* [SK06] is a tool that focuses on cookie privacy. *Bugnosis* [AM02] is a Firefox extension that warns users about server-side tracking using web bugs.

A key question for the privacy of the SB services is the practicability of *Private Information Retrieval* (PIR). If PIR schemes were usable and deployable at a large scale then SB could be made private. The practicability of PIR is highly debated in the community. Olumofin and Goldberg [OG12] refute the general conclusion by Sion et al. [SC07] that no PIR scheme can be more efficient than the trivial one. The authors show that for certain bandwidths, multi-server PIR scheme by Goldberg [Gol07] is more efficient than the trivial PIR. However, it is hard to imagine a company using multi-server approach for its SB services: *it appears that PIR is not yet suitable for a large scale deployment.*

We highlight that the SB services are only one of the many prevalent web-malware detection systems: *Virtual Machine client honeypots* [WBJ+06], *Browser Emulator client honeypots* [CKV10], *Classification based on domain reputation* [APD+10], [FKP10] and *Anti-Virus engines*. The main advantage of these alternatives to the SB is that they can be run locally on the users' device without disclosing any sensitive information to a third party. However, we have not checked if these alternative solutions are privacy-friendly.

8 Conclusions

Safe Browsing services are valuable tools to fight malwares, phishing and other online frauds. However, their use raises serious privacy concerns. GOOGLE and YANDEX have made sincere efforts to make their services as private as possible. Unfortunately, these efforts have not yielded satisfactory results. We have shown in this paper that the design of these two services is fundamentally flawed: hashing and truncation have failed. Our observations on YANDEX database and to a lesser extent on that of GOOGLE show that it is possible to tamper with these Safe Browsing services. These may be deliberate attempts or the results of development errors/misconfigurations. In any case, these can lead to tracking through a mechanism embedded in many software solutions.

We have two solutions to protect ourselves from Safe Browsing services: disabling them or improving them. Disabling Safe Browsing exposes users' private information to malwares and phishing. This is however the radical choice Tor Browser developers. To improve Safe Browsing services, we need privacy and accountability. The trivial PIR, *i.e.* distributing the databases in clear to all the clients, is the only current practical solution providing both the properties. Unfortunately, it comes with a memory/bandwidth overhead. Until any advance in computationally secure PIR is achieved, this is the only available solution.

References

[AB12] Jean-Philippe Aumasson and Daniel J. Bernstein. SipHash: A Fast Short-Input PRF. In *Progress in Cryptology - INDOCRYPT 2012*, Lecture Notes in Computer Science 7668, pages 489–508, Kolkata, India, December 2012. Springer.

[ABJB10] Gaurav Aggarwal, Elie Bursztein, Collin Jackson, and Dan Boneh. An Analysis of Private Browsing Modes in Modern Browsers. In *19th USENIX Security Symposium, Washington, DC, USA, August 11-13, 2010, Proceedings*, pages 79–94, 2010.

[AM02] Adil Alsaid and David Martin. Detecting web bugs with bugnosis: Privacy advocacy through education. In *Privacy Enhancing Technologies, Second International Workshop, PET 2002, San Francisco, CA, USA, April 14-15, 2002, Revised Papers*, pages 13–26, 2002.

[APD+10] Manos Antonakakis, Roberto Perdisci, David Dagon, Wenke Lee, and Nick Feamster. Building a dynamic reputation system for DNS. In *19th USENIX Security Symposium, Washington, DC, USA, August 11-13, 2010, Proceedings*, pages 273–290, 2010.

[App10] Austin Appleby. smhasher - Test your hash functions., 2010. `https://code.google.com/p/smhasher/`.

[BLFM05] T. Berners-Lee, R. Fielding, and L. Masinter. Uniform Resource Identifier (URI): Generic Syntax. RFC 3986 (INTERNET STANDARD), January 2005. Updated by RFCs 6874, 7320.

[Blo70] Burton H. Bloom. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Commun. ACM*, 13(7):422–426, 1970.

[CKV10] Marco Cova, Christopher Kruegel, and Giovanni Vigna. Detection and analysis of drive-by-download attacks and malicious javascript code. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 281–290, New York, NY, USA, 2010. ACM.

[CLRS09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[Cor] Zelster Security Corp. `https://zeltser.com/malicious-ip-blocklists/`.

[CW03] Scott A. Crosby and Dan S. Wallach. Denial of Service via Algorithmic Complexity Attacks. In *USENIX Security Symposium*, Lecture Notes in Computer Science 7668, pages 3–3, Washington, USA, December 2003. USENIX Association.

[Dan12] Quynh Dang. Recommendation for Applications Using Approved Hash Algorithms. Technical Report SP 800-107 Revision 1, National Institute of Standards & Technology, august 2012.

[fak] Python fake factory. `https://pypi.python.org/pypi/fake-factory/0.2`.

[FKP10] Márk Félegyházi, Christian Kreibich, and Vern Paxson. On the potential of proactive domain blacklisting. In *LEET*, 2010.

[Gol07] I. Goldberg. Improving the Robustness of Private Information Retrieval. In *Security and Privacy, 2007. S&P '07. IEEE Symposium on*, 2007.

[goo] http://googleblog.blogspot.fr/2008/07/we-knew-web-was-big.html.

[Goo12] Google. Google Safe Browsing Service in Mozilla Firefox Version 3. https://developers.google.com/safe-browsing/firefox3_privacy?csw=1, April 2012.

[Goo14] Google. Google Chrome Privacy Notice. https://www.google.com/intl/en/chrome/browser/privacy/, November 2014.

[Inc] Verisign Inc. http://www.verisigninc.com/en_US/innovation/dnib/index.xhtml.

[Inc14] Google Inc. Google Transparency Report. Technical report, Google, June 2014. https://bit.ly/1A72tdQ.

[Jen96] Robert Jenkins. A Hash Function for Hash Table Lookup, 1996. http://www.burtleburtle.net/bob/hash/doobs.html.

[McAa] McAfee. McAfee Site Advisor. http://www.siteadvisor.com/.

[McAb] McAfee. McAfee Site Advisor FAQ. https://kc.mcafee.com/corporate/index?page=content&id=KB73457.

[McAc] McAfee. McAfee Site Advisor Live. http://home.mcafee.com/store/siteadvisor-live.

[Mica] Microsoft. SmartScreen Application Reputation. http://blogs.msdn.com/b/ie/archive/2011/05/17/smartscreen-174-application-reputation-in-ie9.aspx.

[Micb] Microsoft. SmartScreen: Authenticode Code Signing. http://blogs.msdn.com/b/ieinternals/archive/2011/03/22/authenticode-code-signing-for-developers-for-file-downloads-building-smartscreen-applicati aspx.

[Micc] Microsoft. Windows Logo. https://msdn.microsoft.com/en-us/windows/dd203105.aspx.

[MKD+02] J. Mogul, B. Krishnamurthy, F. Douglis, A. Feldmann, Y. Goland, A. van Hoff, and D. Hellerstein. Delta encoding in HTTP. RFC 3229, RFC Editor, January 2002. http://tools.ietf.org/html/rfc3229.

[MVO96] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., 1st edition, 1996.

[Nat12] National institute of standards and technology. Secure Hash Standard (SHS). Technical Report FIPS PUB 180-4, National Institute of Standards & Technology, march 2012.

[Nat14] National institute of standards and technology. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. Technical Report FIPS PUB 202, National Institute of Standards & Technology, may 2014. draft.

[OG12] Femi Olumofin and Ian Goldberg. Revisiting the Computational Practicality of Private Information Retrieval. In *Financial Cryptography and Data Security*. Springer Berlin Heidelberg, 2012.

[Per] Mike Perry. Torbutton. `https://www.torproject.org/docs/torbutton/`.

[Pro] Niels Provos. Safe Browsing - Protecting Web Users for 5 Years and Counting. `http://googleonlinesecurity.blogspot.fr/2012/06/safe-browsing-protecting-web-users-for.html`.

[Rom] Sasha Romanosky. FoxTor: helping protect your identity while browsing online. `cups.cs.cmu.edu/foxtor`.

[RS98] Martin Raab and Angelika Steger. "Balls into Bins" - A Simple and Tight Analysis. In *Proceedings of the Second International Workshop on Randomization and Approximation Techniques in Computer Science*, RANDOM '98, pages 159–170, London,UK, 1998. Springer-Verlag.

[sba] Safe Browsing API. `https://developers.google.com/safe-browsing/`.

[SC07] Radu Sion and Bogdan Carbunar. On the Practicality of Private Information Retrieval. In *Proceedings of the Network and Distributed System Security Symposium – NDSS 2007*, San Diego, CA, USA, February 2007. The Internet Society.

[SK06] Umesh Shankar and Chris Karlof. Doppelganger: Better browser privacy without the bother. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, pages 154–167, New York, NY, USA, 2006. ACM.

[SRG97] Paul F. Syverson, Michael G. Reed, and David M. Goldschlag. Private web browsing. *Journal of Computer Security*, 5(3):237–248, 1997.

[Swe02] Latanya Sweeney. k-Anonymity: A Model for Protecting Privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, 2002.

[Syma] Symantec. Norton Safe Search. `http://nortonsafe.search.ask.com/`.

[Symb] Symantec. Norton Safe Web. `https://safeweb.norton.com/`.

[WBJ+06] Yi-Min Wang, Doug Beck, Xuxian Jiang, Roussi Roussev, Chad Verbowski, Shuo Chen, and Samuel T. King. Automated web patrol with strider honeymonkeys: Finding web sites that exploit browser vulnerabilities. In *NDSS*, 2006.

[WS] Ltd WOT Services. Web of trust. `https://www.mywot.com`.

[Yan] Yandex. Yandex Safe Browsing. `http://api.yandex.com/safebrowsing/`.

A Hash functions

A hash function compresses inputs of arbitrary length to digest/hash of fixed size. It is a very popular primitive used in algorithms [CLRS09] and in cryptography/security [MVO96]. The design of a "*good hash function*" depends on the field of application. Non-cryptographic hash functions such as MurmurHash [App10] or Jenkins hash [Jen96] are designed to be fast, to uniformly distribute their outputs and to satisfy several other criteria such as the avalanche test [App10]. The SMHASHER suite [App10] provides a good overview of these functions and the tests they must satisfy.

The design of a cryptographic hash function is very different. Cryptographic hash functions are slower than their non-cryptographic siblings. Furthermore, in order to resist to an adversary, a cryptographic hash function h must verify the following properties:

- **Pre-image resistance:** Given a digest d, it is computationally infeasible to find an input x, such that $h(x) = d$.
- **Second pre-image resistance:** Given an input x and the digest $h(x)$, it is computationally infeasible to find another input $x' \neq x$, such that $h(x) = h(x')$.
- **Collision resistance:** It is computationally infeasible to find two inputs $x \neq x'$ such that $h(x) = h(x')$.

Non-cryptographic hash functions are not designed to satisfy these properties [CW03, AB12]. Let us consider a hash function h with ℓ-bit digests. The choice of ℓ is critical for cryptographic hash functions because the basic complexities for finding pre-image, second pre-image and collisions are 2^ℓ, 2^ℓ and $2^{\ell/2}$ respectively. A cryptographic hash function is considered secure if there is no attack with a lower complexity. The NIST recommendation [Dan12] for cryptographic hash functions are SHA-256, SHA-384, SHA-512 [Nat12] and SHA-3 [Nat14].

We provide in this paragraph two natural extensions of the previous concepts: *truncated digests* and *multiple pre-images* or *multiple second pre-images*. In fact, many applications need to reduce the size of the full digest. Truncated digests must be carefully used as explained in [Dan12], since it is commonly assumed that for a truncated digest of ℓ' bits the security is reduced at least to $2^{\ell'}$ (pre-image and second pre-image) and $2^{\ell'/2}$ (collision). If ℓ' is too small, computation of pre-image, second pre-image or collisions are feasible. For instance, in GOOGLE Safe Browsing (Section 3), SHA-256 is applied to URLs but the digests are truncated to 32 bits *vs* 256 bits of SHA-256. With 32 bits of digest, pre-image, second pre-image can be computed after 2^{32} brute force computations. We also define *multiple pre-images*, where, given a digest d, we wish to compute n pre-images x_i such that $h(x_i) = d$. Accordingly, we define *multiple second pre-images*, where given x' and $h(x')$, we wish to compute n second pre-images x_i such that $h(x_i) = h(x')$.

B Multiple prefix URLs

We present in Table 7, URLs in the Alexa list for which more than 1 decompositions were found to be present in the malicious prefix lists of YANDEX. The first URL:

`http://torr.comoj.com`

creates two hits in the `ydx-malware-shavar` file of YANDEX. The remaining 11 create two hits in the `ydx-porno-hosts-top-shavar` file of YANDEX.

C Characterizing Alexa prefixes

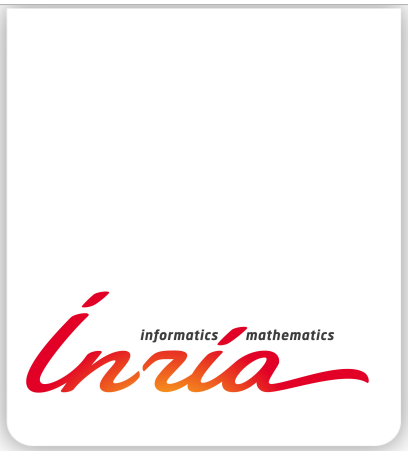
We present here Table 8 which characterizes Alexa prefixes which are considered as malicious. We classify the corresponding URLs into four important categories: religion, war/crime, disease, and sex.

URL	matching decomposition	prefix
http://torr.comoj.com/	torr.comoj.com/	0x18e3177e
	comoj.com/	0xc748b1b8
http://rpc.weblogs.com/	rpc.weblogs.com/	0xfb8c1dea
	weblogs.com/	0x58c465bd
http://fr.xhamster.com/user/video	fr.xhamster.com/	0xe4fdd86c
	xhamster.com/	0x3074e021
http://nl.xhamster.com/user/video	nl.xhamster.com/	0xa95055ff
	xhamster.com/	0x3074e021
http://m.mofos.com/user/login	m.mofos.com/	0x6e961650
	mofos.com/	0x00354501
http://fr.xhamster.com/user/kmille	fr.xhamster.com/	0xe4fdd86c
	xhamster.com/	0x3074e021
http://de.xhamster.com/user/video	de.xhamster.com/	0x0215bac9
	xhamster.com/	0x3074e021
http://nl.xhamster.com/user/ppbbg	nl.xhamster.com/	0xa95055ff
	xhamster.com/	0x3074e021
http://mobile.teenslovehugecocks.com/user/join	mobile.teenslovehugecocks.com/	0x585667a5
	teenslovehugecocks.com/	0x92824b5c
http://nl.xhamster.com/user/photo	nl.xhamster.com/	0xa95055ff
	xhamster.com/	0x3074e021
http://m.mofos.com/user/logout	m.mofos.com/	0x6e961650
	mofos.com/	0x00354501
http://m.wickedpictures.com/user/login	m.wickedpictures.com/	0x7ee8c0cc
	wickedpictures.com/	0xa7962038

Table 7: URLs with multiple matching prefixes.

		Orphans				One parent			
	list name	**religion**	**war/crime**	**disease**	**sex**	**religion**	**war/crime**	**disease**	**sex**
Google	goog-malware-shavar	na	na	na	na	1	6	2	34
	googpub-phish-shavar	na	na	na	na	0	2	0	1
Yandex	ydx-malware-shavar	4	0	3	11	17	21	22	79
	ydx-adult-shavar	0	1	0	32	0	0	0	13
	ydx-mobile-only-malware-shavar	1	0	1	1	0	0	0	4
	ydx-phish-shavar	0	0	1	0	na	na	na	na
	ydx-mitb-masks-shavar	0	0	0	0	na	na	na	na
	ydx-porno-hosts-top-shavar	0	0	0	43	26	58	26	6882
	ydx-sms-fraud-shavar	0	0	0	20	0	0	0	0
	ydx-yellow-shavar	0	0	0	3	na	na	na	na

Table 8: Characterising TopAlexa websites whose 32-bit prefix is present in the Yandex and Google lists.



RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399