CrowdSurf: Empowering Transparency in the Web

Hassan Metwalley
DET, Politecnico di Torino, Italy
hassan.metwalley@polito.it

Stefano Traverso
Politecnico di Torino, Italy
stefano.traverso@polito.it

Marco Mellia
Politecnico di Torino, Italy
mellia@polito.it

Stanislav Miskovic
Symantec Corp., USA
stanislav_miskovic@symantec.com

Mario Baldi
Politecnico di Torino, Italy
mario.baldi@polito.it

ABSTRACT

Individuals lack proper means to supervise the services they contact and the information they exchange when surfing the web. This security task has become challenging due to the complexity of the modern web, of the data delivering technology, and even to the adoption of encryption, which, while improving privacy, makes in-network services ineffective. The implications are serious, from a person contacting undesired services or unwillingly exposing private information, to a company being unable to control the flow of its information to the outside world.

To empower transparency and the capability of taking informed choices in the web, we propose CROWDSURF, a system for comprehensive and collaborative auditing of data exchanged with Internet services. Similarly to crowdsourced efforts, we enable users to contribute in building awareness, supported by the semi-automatic analysis of data offered by a cloud-based system. The result is the creation of "suggestions" that individuals can transform in enforceable "rules" to customize their web browsing policy. CROWDSURF provides the core infrastructure to let individuals and enterprises regain visibility and control on their web activity. Preliminary results obtained executing a prototype implementation demonstrate the feasibility and potential of CROWDSURF.

Keywords

Privacy, Crowdsourced systems, Web Browsing

1. INTRODUCTION

Users can not do without being online. They rely on web services to organize their work and life, to retrieve information and products, to establish and maintain social relationships, etc. To keep up with this trend, technology has considerably evolved. On the one hand, producers build smarter terminals, which consistently interact with multiple web sources to fetch content and information for the user. On the other hand, the web has grown in complexity as well. To the end of improving interaction and design, modern websites consist of numbers of different objects which are often served by multiple, possibly third party, sources. Hence, the infrastructure delivering the web, i.e., the Internet, e., between the user and the web. Due to this change, the Internet nowadays consists of a complicated mix of (pervasive) distributed systems such as CDNs, cloud services, virtualized resources, etc. The way people access the Internet is also more complicated than in the past, with smartphones running apps instead of one browser in a PC.

taking part in it. This scenario dramatically complicates the task of controlling what services are accessed and which information users share with the providers offering such services. Over the past years network middleboxes have been deployed throughout the Internet to filter inappropriate/undesired content, and to protect users and organizations from possible threats. The capabilities of these devices are more recently being curbed by the growing deployment of traffic encryption. According to recent estimates [1], about 50% of web traffic is HTTPS, and thus, encrypted. For instance, according to the same study, child protection through the use of Internet Watch Foundation[1] blacklists has become ineffective, with just 5% of entries still being blocked when HTTPS is deployed (e.g., by blocking the entire domain name).

In parallel, the increasing trend of users "living online" has favored a business around the collection of personal information. The web includes indeed hundreds of tracking services which implicitly or explicitly extract data from the user's web browsing activity. The data collection business has reached striking volumes [2, 3, 4, 5], and the phenomenon is so pervasive that users encounter the first tracker as soon as they start browsing the web [5]. This results in involuntary leakage of information that users and companies would like to keep private, in people being exposed to dubious third party services, as well as in web companies (sometimes illegitimately) profiling users, and selling private information. While encryption helps in protecting users' privacy, it also prevents third parties to observe and possibly control what (personal) data is exchanged, which is fundamental for instance in corporate scenarios.

In this paper we advocate the need for a model where users are explicitly offered the freedom to i) know which services their data is exchanged with and how, and ii) decide what information can be accessed and shared, and with whom. We present as such CROWDSURF, a system that allows end-users and enterprises to regain visibility on the information they exchange on the web and, possibly, to control it. CROWDSURF's approach is holistic, and we design it following principles that are sound and practical, rather than revolutionary. CROWDSURF targets HTTP, the new "narrow waist of the Internet" [6], and, hence, users accessing the web using a browser or running a smartphone app, whether connected to a public WiFi hotspot, residential gateway or a corporate network.

We design the CROWDSURF client as a layer that sits right below the application layer, where information has not yet been encrypted when TLS is being used. The CROWDSURF backend is a cloud service which builds on crowdsourcing principles, whereby CROWDSURF can semi-automatically learn from contribution users. Any user and/or device running CROWDSURF can contribute according to a personal level of expertise or convenience, from teams of security researchers who can collaboratively setup experiments to observe intricate signs of behind-the-scenes communications, to novice users who voluntarily offer anonymized samples of their

This work was conducted under the Narus Fellow Research Program.

[1] https://www.iwf.org.uk

traffic, or vote on the legitimacy of data leaving their devices. All the data gathered and processed enables the CROWDSURF cloud to compute *suggestions* about the trustfulness of web services. The CROWDSURF client translates these suggestions in *rules* which the user can install and customize in order to enable control on the services contacted and on the information exchanged. Users can choose to block undesired services, remove information they would like to keep private, or explicitly embrace third party services for the advanced functionalities they offer, in spite of the personal information they gather.

For this paper, we implement CROWDSURF in a prototype that we use to show the feasibility of the approach. We do not design CROWDSURF to target a specific threat, and we demonstrate its flexibility by providing different application examples. In particular, we show its capability to impose blacklists (see Sec. 2.4), to assess the security mechanisms offered by services transporting users' personal information (see Sec. 3.2), or to automatically detect the presence of user identifiers in HTTP traffic (see Sec. 4.1).

Our work is preliminary, and aims at stimulating a debate toward the creation of a sustainable ecosystem with increased transparency and control. If widely deployed, CROWDSURF would allow users to acquire consciousness of which services they contact, and which data they expose and to whom. On the other hand, CROWDSURF would lead web services to embrace more transparent and loyal communication mechanisms.

We believe CROWDSURF's potential is wide, and that it can be employed in several contexts. Indeed, CROWDSURF would let a parent decide which websites her children can access, or a company define and impose policies for employees (including the case when BYOD – Bring Your Own Device – policy is allowed). CROWDSURF allows also third parties to offer novel services, where users voluntarily opt for advanced services, e.g., for using an accelerating proxy managed by an ISP for specific types of traffic (e.g., when watching videos, but not when accessing her bank account). CROWDSURF could even be instrumental in enabling users to monetize on their personal information, should they decide for it, as proposed in [7].

The remainder of the paper is organized as follows. Sec. 2 first provides a high level description of CROWDSURF, detailing the design of the client, and the cloud components. Sec. 3 describes the prototype of CROWDSURF we build. We rely on this prototype to analyze performance implications on the user experience, and to present a simple application example in which CROWDSURF observes how users' credentials are exchanged with popular services. Then, we focus on the cloud-side of CROWDSURF in Sec. 4: we present an algorithm for the detection of user-identifiers embedded in HTTP requests, and then analyze the time the system would take to collect enough information to build suggestions. Then, we discuss the open problems of CROWDSURF in Sec. 6 and, finally, Sec. 7 concludes the paper.

2. CROWDSURF

CROWDSURF is a crowdsourced system in which users can opt to collaborate by providing explicit (e.g., their opinion) and implicit (e.g., traffic samples) information about the web services they use. Requiring the involvement of users, we follow a short list of simple design principles:

1) **Privacy-safe**: it must never compromise users' privacy, and any contribution must be purged from any personal information.
2) **Automated**: it must support the generation of suggestions by running algorithm on the cloud.
3) **Client centric**: it must be available on any device, as the default tool to enhance transparency and enforce users' choices.

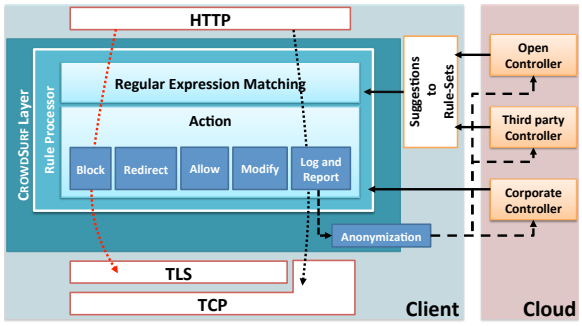

Figure 1: CROWDSURF layer in the networking stack and its high level structure.

2.1 CROWDSURF Architecture Overview

A *controller* – running in the cloud – collects pieces of information provided by the volunteers and feeds an automatic *data analyzer*, which runs data mining algorithms to produce *suggestions*. A suggestion contains indications about the reliability of one or more web services. For instance, the data analyzer can flag services collecting/leaking users' personal information, or known to host malicious software. A federated group of experts forms the *advising community*, which inspects the results provided by the data analyzer and interacts with it to generate the suggestions. Following a collaborative approach similar to Wikipedia and the Electronic Frontier Foundation (EFF), users are invited to increase the wisdom of the system. On the other hand, thanks to the CROWDSURF client, the users can supervise the information they exchange with services, and can contribute in forming the suggestions. They can voluntarily donate portions of their browsing activity, i.e., anonymized HTTP-level traces, to support the analysis of the community. Or they can simply use the system, without contributing to it, as the large majority of users does for Wikipedia.

2.2 CROWDSURF on the Client-side

We envision as CROWDSURF cornerstone a new layer to add to the stack. We expect users' terminals, mobile devices and personal computers alike, to embed the CROWDSURF layer in their operating system. The Client box of Fig. 1 shows the high level architecture. It sits between the HTTP and transport layers, where it has access to HTTP traffic, before it is possibly encrypted. This choice is motivated by the fact that today HTTP is "the" application layer protocol [6].

Users asynchronously and periodically obtain suggestions from the advising community and are free to decide to what level to take them into account. This feature is implemented by the `Suggestions to Rule-Sets` block in Fig. 1. It enables controlling how suggestions are translated in a set of *rules*, or *rule-set*. A rule consists of a *regular expression* (which may apply on all or specific fields of the HTTP requests) and one or more *actions*. For each HTTP request, the `Rule Processor` looks for matches and applies the corresponding actions, for instance `Block`, `Redirect`, `Modify`, `Log&Report`, with `Allow` being the default one. We provide some examples of rule-sets in Sec. 2.4. This simple pattern matching/action process has been proved very flexible and very efficient. It is at the root of successful technologies such as the ones used in firewalls, antiviruses, traffic classifiers, etc.

Given that CROWDSURF leverages a crowdsourced approach, the `Log&Report` block is vital. It enables the collection of data samples. The layer collects traffic samples at user's will and under user's full control. The layer temporarily stores the collected data locally, and periodically transmits it to the controller. We adopt different approaches to limit private information leakage. The anonymization block in Fig. 1 is responsible for this. First, it implements

sampling policies, e.g., by logging only samples of traffic at random. This also reduces the amount of data to transfer. Second, it assigns each user a unique random identifier, CS-UID, rotated periodically (e.g., every day). Third, it alters any piece of personal information it can find, such as, e.g., username/password credentials, cookies, etc. For instance, by default the block replaces all key values found in HTTP requests with random strings obtained by using cryptographic hash functions with user's identifier as seed to guarantee consistency for the data analysis. Fourth, users can leverage the pattern/action mechanism to implement anonymization practices. For instance, this enables the definition of custom policies like "never collect data when browsing my online bank account, or when checking my email". Fifth, data on the controller will be stored for only the time needed to process it (also to limit the storage). At last, since even information available at the network layer (e.g., IP addresses) could be exploited to trace back the identity of the user, transmission to the controller takes place through an encrypted channel established with a peer randomly picked (from a list provided by the controller), i.e., by employing other devices running the CROWDSURF layer as relays. These anonymization mechanisms are meant to prevent occasional attacks and can not be considered as 100% reliable. Unfortunately, we are not aware of privacy preserving policies that allow us to remove all sensitive data from HTTP traces. In our ongoing effort we are investigating the adoption of data sharing mechanisms based on differential privacy [8]. However, we remark that the user voluntarily opts to contribute, and she has full control on which data she intentionally decides to share.

2.3 CROWDSURF in the Cloud

The cloud controller is responsible for semi-automatically analyzing the traffic samples and the reports provided by users who participate CROWDSURF, and for generating the suggestions. In particular, we devise three different cloud controllers, depending on the possible deployment scenario:

Open controller: In the most common case, a user accessing the Internet receives suggestions from the *advising community* and uses them to regulate her access to web services. If user's preference allows it, traffic samples are sent to the controller so to contribute to the advising community. The latter may be offered and supported by public bodies or non-profit organization like EFF.

Third party controller: In a second scenario, suggestions could also be generated by a *third party advisor* run by an independent, third party entity which offers custom suggestions to users. This opens a "market of suggestions". Or alternatively, a third party advisor can offer advanced suggestions in change of data voluntarily offered by users for running other businesses, e.g., targeted advertisement [7].

Corporate controller: When CROWDSURF is deployed in a corporate scenario, the *corporate controller* does not create suggestions, but it directly imposes rule-sets which are installed on all devices connected to the corporate network. The same rule-set can be easily imposed on any corporate-owned device, even when connected from other networks. We expect the employee not to be allowed to modify the rule-sets imposed by the corporate authority. To handle the BYOD case, the presence of the corporate controller must be automatically identified by any device when connected to the corporate network. This can be achieved for instance using DHCP extensions, or using standard DNS names that forces the CROWDSURF client to connect to the corporate controller.

2.4 Application Examples

In the following we provide examples of CROWDSURF applications in both the public Internet, and the corporate network. We use the same examples to run the experiments in Sec. 3.1.

		Actions		
		`block`	`redirect`	`log&report`
Web Services	Facebook	C		
	Twitter			C
	Dropbox			C
	Google		C(→Bing)	
	YouTube	C		
	Ebay+Amazon	C		
	Adult Sites	C, K		
	Trackers	P		K
	Ads+NoJS	P		

Table 1: Rules for the (P)aranoid, (K)id and (C)orporate profiles.

We consider three user profiles, each exploiting CROWDSURF to customize their browsing experience. A summary of rules is available in Tab. 1. We define a "Paranoid Profile" (P) that opts for blocking all advertisement sites, to not download offending Javascript code, and to use private navigation mode on the browser. This profile is the equivalent of running browser plugins like AdBlock Plus and NoScript. This user decides to not share any traffic samples with the community.

A second profile is called "Kid Profile" (K): the user activates parental control by installing the suggestions provided by the advising community. For the experiment we describe in Sec. 3.1, we simply use the list of the Alexa top 50 "Adult Sites" augmented by other manually verified adult sites. The user contributes also to manually signal other offending websites/objects she gets into. Finally, she volunteers to enable random logging and reporting of three popular online trackers (*doubleclick.net*, *scorecardresearch.com*, and *yieldmanager.com*). By doing so she supports the advising community that would like to understand which kind of information these trackers collect.

A third profile impersonates the "Corporate Profile" (C): rules are imposed by the network administrator, and i) do not allow employees to access Facebook (also removing Facebook buttons from any website), ii) redirect all requests from Google Search to Bing, iii) block the usage of adult sites, Ebay, Amazon, and YouTube, and iv) all HTTP(S) requests exchanged with Dropbox and Twitter are reported to the corporate controller.

3. PROOF OF CONCEPT

As a proof of concept, we develop a preliminary prototype in which the controller is a Java-based web service where the SOAP protocol is used to communicate with CROWDSURF-enabled devices. For the sake of ease, we implement the CROWDSURF client as Firefox plugin based on the Mozilla Javascript SDK. Our plugin works on both standard PCs and mobile devices, supports rules, and the `allow`, `block`, `redirect`, `log&report` actions.[2] The plugin supports the encryption of all communications. Traffic logs are compressed and anonymized before being transmitted. At startup, the data analyzer provides each CROWDSURF instance with a randomly assigned ID (the CS-UID). The controller component receives and stores reports that we process to extract information.

3.1 Performance Implications

We exploit our implementation to evaluate the performance overhead a user would pay when running CROWDSURF on her device. Given the not-optimized implementation of the prototype, our benchmark is meant to show the feasibility of the approach rather than being considered as a thorough testing. We consider the three profiles described in Sec. 2.4. For baseline, we take a plugin-free Firefox configuration. We setup a testbed based on Selenium Web-

[2]The code building our software is available upon request.

Driver[3] to automatize the browsing of webpages. We consider i) the Alexa top 10 global websites, ii) 8 popular news portals, and iii) 6 portals which do not include any online tracker. We run the experiment from a standard PC and instrument the browser to visit each website 20 times, emptying the cache at every run. Then, we measure the average time needed to render the webpage.

We notice that the Paranoid profile is favored, as it blocks advertisement and some Javascript content download, thus speeding up the rendering of the webpage in many cases. For the case of *google.com*, the Corporate profile shows much better performance than the Paranoid, since in the former profile requests are redirected to *bing.com*, which in our measurements is faster at rendering. In general, results show that the extra load generated by a possible CROWDSURF implementation is fairly minimal, especially considering the low optimization of our code. The Paranoid profile is 1.07 times faster than the baseline, while Corporate and Kid configurations show slightly worse performance being 1.08 and 1.17 times slower, respectively. In summary, our result shows that clients today have enough power to easily execute CROWDSURF without harming users' browsing experience. We invite the interested reader to refer to our technical report [9] for a thorough description of the results.

3.2 Checking HTTPS Information Handling

We use our implementation to run a kind of analysis we want CROWDSURF to offer, i.e., the capability of verifying to which services personal data is sent, and how. In particular, we are interested in understanding whether information such as user credentials and credit card data might be shared with third party services, and how these are exchanged with the first party.

We collect a dataset by browsing a catalog of websites with a CROWDSURF enabled browser to log all HTTP and HTTPS requests. We consider a list made of the Alexa top sites in the Global, Banking, Gambling, and Shopping categories. We investigate a total of 160 websites. For each of them, we manually attempt to log in with the dummy credentials "MyName:MyPassword". We then analyze the logs the client collects.

We observe that the credentials are handled by the first party service only, and never exchanged (at least not it plain text) with third parties. However, we find that still 10% among the most popular websites in the Global rank do not use HTTPS to exchange users' credentials. Only two of these apply some client-side custom encryption/obfuscation technique before transmitting them to the server. Even more surprisingly, when HTTPS is used, we notice that users' credentials are sent in plain text over the encrypted channel. Assuming HTTPS offers a secure channel, no guarantees are given on how the server handles and stores credentials. Indeed, the server could store those in plain text, posing severe security risks if the server gets compromised. Unfortunately, this is not a rare event. The most recent incident involved a giant like eBay [10]. 90% of both Gambling and Shopping categories do not hash the credentials. Even for the Bank category, 75% of websites transmit credentials in plain text, totally trusting the HTTPS channel. Interestingly, some of those do implement two-step strong authentication methods based on PIN or token, which are sent in plain text through the HTTPS channel.

These findings strengthen the need for CROWDSURF to warn users and services about the weaknesses that are unfortunately present on (popular) websites exchanging credentials. As discussed in Sec. 5, no other tool offers this kind of capability.

We also run a second experiment where we examine the key

[3] `http://www.seleniumhq.org`

Algorithm 1 Automatic third party tracker identifier.

#HTTP request log and target website
Input: $\mathbf{HS}$, $target$
#List of possible third party trackers and their user-tracking keys
Output: $\mathbf{TS}$
#Init hashtable of user identifiers
$\mathbf{H_u} \leftarrow$ init_hash_table()
#Init hashtable of key-value pairs
$\mathbf{H_{kv}} \leftarrow$ init_hash_table()
#Read HTTP request logs
while h in $\mathbf{HS}$ **do**
  #Extract fields of interest
  $h \leftarrow$ csUid, host, path, referer
  #Check target is third party
  if $target$ not in h.host and $target$ in h.referer **then**
    #Extract key, value pair from the URL path
    $K, V \leftarrow$ extract_keys(h.path)
    #Iterate all key names and values
    while k, v in K, V **do**
      #Create hash for $\mathbf{H_u}$
      $host_key_csUid \leftarrow$create_hash(h.host,k,h.csUid)
      #Create hash for $\mathbf{H_{kv}}$
      $host_key_value \leftarrow$create_hash(h.host,k,v)
      #Insert all key-value pairs in $\mathbf{H_u}$
      ADD_DISTINCT($\mathbf{H_u}[host\_key\_csUid]$,v)
      #Insert CS-UID in $\mathbf{H_{kv}}$
      ADD_DISTINCT($\mathbf{H_{kv}}[host\_key\_value]$,h.uid)
    end while
  end if
end while
#Iterate over $\mathbf{H_u}$
while hash in $\mathbf{H_u}$ **do**
  #Iterate over values mapped to current hash
  while value in $\mathbf{H_u}$[hash] **do**
    #Check current hash refers to one value only
    if LEN($\mathbf{H_u}$[hash]) == 1 **then**
      #Decode hash into host, key and CS-UID
      $host, key, csUid \leftarrow$ decode_hash(hash)
      #Create an auxiliary hash using host, key and value
      $hash_{aux} \leftarrow$ create_hash($host, key, value$)
      #Check the auxiliary hash in $\mathbf{H_{kv}}$ contains only one CS-UID and
      #check this corresponds to the one in $\mathbf{H_u}$
      if LEN($\mathbf{H_{kv}}[hash_{aux}]$) == 1 and $\mathbf{H_{kv}}[hash_{aux}] == csUid$ **then**
        #Add host and key to the output list
        ADD($\mathbf{TS}$, $host$, key)
      end if
    end if
  end while
end while
return $\mathbf{TS}$

names of parameters in HTTP requests that carry the username and password to create two regular expressions to automatically match those keys. Surprisingly, a regular expression of only 5 (6) terms allows to match 98% of keywords used for username (password), with no false positives in any other part of this dataset. This means that services usually employ standard names for keys carrying credentials, e.g., `uid`, `login`, `pwd`, etc. This is a simple example to show the potentiality of CROWDSURF to support data extraction and suggestion creation.

4. CROWDSURF AT SCALE

To better understand how CROWDSURF would work at a larger scale, we have to consider more sizable portions of traffic than those we collected with our prototype. For this, we consider a dataset obtained from passive measurements where HTTP requests generated by 19,000 households in a ISP network are monitored (see [5] for all details).

4.1 Automatic Tracker Detector

One of the CROWDSURF challenges is the need of automatic means to detect services that possibly offend users' policies. This

acmetrack.com		User$_1$	User$_2$	…	User$_n$
Visit-1	key1 ✗	y$_1$	y$_2$	…	y$_n$
	key2 ✗	z	z	…	z
	key3 ✓	v$_1$	v$_2$	…	v$_n$
Visit-2	key1 ✗	y$_1$'	y$_2$'	…	y$_n$'
	key2 ✗	z	z	…	z
	key3 ✓	v$_1$	v$_2$	…	v$_n$
…	…	…	…	…	…
Visit-m	key1 ✗	y$_1$''	y$_2$''	…	y$_n$''
	key2 ✓	z	z	…	z
	key3	v$_1$	v$_2$	…	v$_n$

Figure 2: Example of key-value pairs the algorithm 1 checks. `key3` is the only one the algorithm labels as user-identifying, as it maintains the same value for the same user across multiple visits, but it shows different values across different users.

section presents a simple automatic solution the data analyzer running in the CROWDSURF cloud can use to generate suggestions. As an example, we present an unsupervised methodology to identify possible third party trackers that users unknowingly contact while browsing a given website. The results provided by the algorithm can be used by the advising community to create suggestions that notify the users about the tracking services they encounter.

The algorithm we design builds on the observation that third party trackers often exchange user-identifiers (UID) as URL parameters. Then, the algorithm looks for parameters in HTTP GET requests that look like UIDs.[4] Let us consider the third party service `acmetrack.com`, and the HTTP query example `http://acmetrack.com/query?key1=X&key2=Y`. The algorithm first extracts `key1` and `key2`, with values `X` and `Y`, respectively. Then, it looks for the keys whose value is uniquely associated to the CS-UID (the identifier CROWDSURF assigns to the user), i.e., one value must be associated to one CS-UID, and values must be different for different CS-UIDs. Fig. 2 reports intuitive examples of keys our algorithm processes for third party site `acmetrack.com`. The URL contains three keys. `key1` shows different values for different users, but these are not equal across visits, so it cannot be a user identifier. `key2` takes the same value across different users (and visits), so we discard it. `key3` is the only key whose values are different for different users, and constant for same user across different visits. Thus `key3` is considered a possible user-identifier.

A simple way of designing the algorithm could be by building a matrix as depicted in Fig. 2, and iterate it to pinpoint user-identifying keys. However, this approach is hardly implementable since it requires large memory. Then, for our implementation, we decide to rely on hash tables as illustrated in Alg. 1.

More in detail, our algorithm leverages the HTTP requests reported to the controller by users who activated CROWDSURF's `Log&Report` action on their devices. These requests form the dataset $\mathbf{HS}$ used as input. From $\mathbf{HS}$, URLs having *target* appearing in the `Referer` field, but not in the `Host` string are considered (lines 3-14). In other words, the data analyzer extracts all third party URLs a user contacts when accessing pages of the *target* website. For each matching URL, we extract all HTTP key-value pairs (K, V) contained in each GET request (line 6). For each service, and for each key, we investigate one-to-one mapping between the CS-UID and the observed values. This is obtained using two hash tables, $\mathbf{H_u}$ and $\mathbf{H_{vk}}$. The first uses as key the hash of (tracker hostname, key name k, CS-UID), and contains all distinct values v seen in the logs. The second instead uses as key the hash of (tracker hostname, key name k, key value v), and contains the distinct CS-UIDs (lines 8-11). Finally, the algorithm iterates over the values contained in the two hash tables to elect the keys which show a one-to-one mapping with CS-UIDs (line 17-20), add them, together with their hostnames, to the output list (line 21), which is finally returned at the end of the execution (line 26).

[4]The methodology can be easily extended to process the data the client transmits to the servers via POST requests, or embedded in the cookies, or to seek for user identifiers based on combinations of multiple keys.

Website	# of distinct trackers
Portal1	26
News1	13
E-Commerce1	12
E-Commerce2	9
E-Commerce3	4
Portal2	4
Porn	3
SportNews	1
SearchEngine	1

Third party service	Keys
c1.adform.net	`xid`
atemda.com	`bidderuid`
x.bidswitch.net	`user_id`
googlesyndication.com	`afp,tdl,tme`
www.77tracking.com	`pagehittag,rand`
track.movad.net	`us`
ovo01.webtrekk.net	`cs2`
ums.adtech.de	`providerid`
metrics.nt.vc	`cg,hu`
ib.adnxs.com	`xid`
p.rfihub.com	`bk_uuid`
pixel.mathtag.com	`check`
dis.criteo.com	`uid`

Table 2: Number of distinct third party trackers for each of the 9 websites in our experiment (left), and the third party trackers and the user-tracking keys our algorithm detects for website $News1$ (right).

Given the output, we further process it to guarantee statistical evidence. In particular, we consider as actual UIDs those keys for which we observe at least 10 different CS-UIDs, and 10 visits each. We chose these thresholds experimentally. A comprehensive study of their sensitivity can be found in [11].

We validate our approach using our passive HTTP trace. We target popular portals. For each of them, we look for third party services, and we extract those keys whose values show a one-to-one mapping with the CS-UID.

The left part of Tab. 2 shows the number of third party trackers found in each website in our experiment. In total, we identify 65 distinct trackers. Surprisingly, we find that 27 of them are actual tracking services not present in the blacklist of Ghostery: *a.tfxiq.com*, *m.webtrends.com* and *track.movad.net* to name a few.

Considering for instance $News1$ service, we check the third party hostnames associated to it that were discovered using user-identifiers. We report them in the right part of Tab. 2. Key names are shown on the left column. For $News1$ the algorithm identifies 13 distinct third party services. By inspecting their hostname, we observe that all are (well-known) tracking services, with most of key names suggesting the exchange of possible user identifiers.

These results, even if far from conclusive, are promising, and show that the availability of large data enables automatic detection of personal information leakage. Our experiments are carried over HTTP traffic. CROWDSURF allows us to check for privacy leakages also in case HTTPS is in place.

4.2 A Feasibility Check

As described in Sec. 2, the crowd feedback is vital for CROWDSURF. Therefore, in this section we run a simple study to compute how long the system would take to build a large enough set of user reports to generate reliable suggestions. We consider the case in which we aim at collecting reports involving N different services, e.g., websites. We say a service data to be reliable when we have collected at least K reports. We assume that a user sends its report back to the controller when she visits that service. We consider sampling, so that only a fraction of visits are reported.

Understanding how many reports we need to obtain K distinct

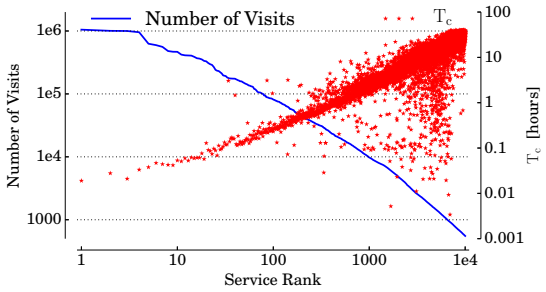

Figure 3: Popularity of services found in a portion of our trace, and corresponding average time T_c (in hours) to collect at least 100 reports with a sample ratio equal to $1/10$.

reports for each of the N services belongs to the family of Coupon Collector's problem. In particular, we have to refer to the Newman-Shepp generalization [12]. Let $E[V]$ be the expectation of the total number of coupons (reports) to collect so that, for each of the N goods (services), at least K coupons are collected. Then, $E[V]$ is given by:

$$E[V] = N \log N + (K-1)N \log\log N + O(N). \quad (1)$$

The model assumes coupons are uniformly extracted among N goods. By targeting $K = 100$ reports for $N = 10,000$ services, $E[V] \approx 2.3$ M coupons. In our experiment, the total number of HTTP requests that are logged considering one day is ≈ 45 M. Thus, assuming i.i.d. requests, we obtain that the system would take ≈ 1.2 hours to collect at least K=100 reports for each of the N=10,000 selected services in the catalog. If we apply a sampling rate of 1/10, then the number of reports reduces to 4.5 M, and thus the time to collect enough reports becomes 12 hours.

In reality, the probability of visiting a service is not i.i.d., and it typically follows a heavy-tailed distribution. Consider the left y-axis of Fig. 3. It reports the number of visits of the top $10,000$ services. As expected, it follows the typical Zipf-like distribution. Thanks to this, the top $10,000$ services correspond to 88.13% of total visits. We run a trace-driven experiment using the actual trace to evaluate the average time T_c needed to collect 100 reports for each of the top $10,000$ services in the trace. We assume CROWDSURF clients are configured with sampling ratio equal to $1/10$. We focus on the top $10,000$ popular services in the first day of our trace. For each request, we measure T_c, averaging over 12 independent runs, i.e., repeating the experiment assuming the collection starts at a different random initial time. As soon as K=100 reports are collected, the data collection is said to be reliable for the target service.

Red dots in Fig. 3 reports T_c (in hours, right-y axis) for each service. As shown, CROWDSURF would take few hundred seconds to collect 100 visits for the most popular service, e.g., 87 s for *www.google.com*, much less than the time predicted by Eq.(1). Less than 48 h are needed in the worst case, i.e., for those services whose visit rate is much smaller than the average rate considered in Eq.(1). Observe that some services show very bursty traffic patterns that considerably decrease T_c. Indeed, when the clients access those services, we collect a large number of reports in few time. The overall average value of T_c is 12.57 h.

This simple experiment shows that even with a population of only $19,000$ contributors that are reporting $1/10$ of their activity, CROWDSURF can easily collect enough reports to compute suggestions. We can also envision smarter sampling policies to, e.g., avoid to keep collecting samples from most popular sites while only asking sample contributions for other services.

	Ghostery	AdBlock Plus	WoT	DNTMe	PrivDog
Block ads	Yes	Yes	No	Yes	Yes
Disable tracking	Yes	Yes	Yes	Yes	Yes
Parental control	No	No	Yes	No	No
Filter sent data	No	No	No	No	No
Customizable	Yes	Yes	Yes	No	Yes
Support mobiles	Firefox	Proxy	No	Custom	No
Collect traffic	Yes	Yes	No	Yes	No
Collect rating	No	No	Yes	No	No
Open-source	No	Partially	No	No	No
Corporate ver.	Yes	No	No	Yes	No
Neutral authority	No	No	No	No	No

Table 3: Feature comparison with selected client-based solutions.

5. CROWDSURF AND SIMILAR SOLUTIONS

To the best of our knowledge, we are the first to propose an open cloud-based system relying on crowdsourced principles to increase transparency in the web. The only proposal that could be compared to CROWDSURF is Privee [13], but it presents significant differences. First, it specifically targets the classification of web services based on the privacy policies they publish on their websites. Privee combines users' reports and automatic text-mining algorithms to perform this classification. CROWDSURF follows similar principles, but has a much wider scope. For instance, Privee does not leverage HTTP traffic samples, but only users' reports, and, more importantly, do not offer a client-side component which translates classification results in concrete actions. The browser plugin Privee provides simply labels potential privacy-offending services. Differently, CROWDSURF provides information users can exploit to manipulate their HTTP traffic.

Focusing on tools made available by the industry, we can find several solutions which help users filter the HTTP traffic they generate. However, these tools are mainly oriented to protect users from online trackers or advertisement. They are all implemented as browser plugins. The most popular are DoNotTrackMe (DNTMe), WebOfTrust, Ghostery, EFF's Privacy Badger PrivDog, AdBlock Plus and NoScript. Tab. 3 reports some of the features we want CROWDSURF to offer against their availability in a subset of the aforementioned tools. As reported, each tool offers different subsets of features. Notably, no plugin investigates which kind of data is exchanged with the server, mostly blocking the transactions based on the contacted hostname only. None of them assess the security level of communications carrying users' sensitive information (e.g., how users' credentials are transmitted to the server – see Sec. 3.2). Not all of them offer a good level of customization. Only a few of them collect users' contributions and provide information about website trustfulness. For instance, AdBlock Plus collects users' traffic samples (but the users has no control of which data is shared) and WoT collects users' ratings. None of them is open-source, and AdBlock Plus is the only one sharing with the community the list of regular expressions it uses to build the blacklist of services. Few solutions offer advanced functionality which may be suitable for the administrators of corporate networks (e.g., a business version is available). Only EFF's Privacy Badger (not in the table) is developed and maintained by a neutral authority. All other tools are developed by third parties, which the users are compelled to trust, sometimes disappointingly [14].

Importantly, only three of them can run on mobile devices, but with severe limitations. For instance, they run either as stand-alone web browsers, or as Firefox plugins. Thus, they have no visibility on the traffic generated by other apps. AdBlock Plus runs as a proxy inside the mobile terminal, so to process HTTP traffic generated by applications. However, it has no visibility on HTTPS traffic. This is in common with network-based solutions such as proxies and firewalls. They do not provide the functionality we want CROWDSURF to implement. Indeed, despite both have been designed to process

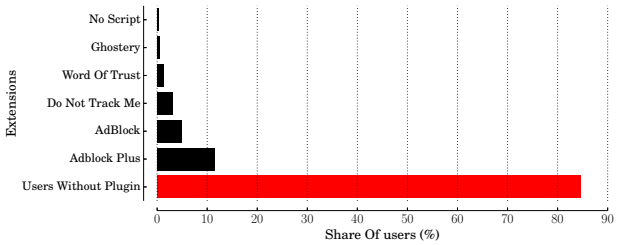

Figure 4: Shares of users adopting "popular" privacy-preserving extensions.

traffic, they have no visibility when encryption is enabled. Man-in-the-Middle solutions are also available, but appear to be very intrusive (and cumbersome).

We check the popularity of some extensions among Internet users by leveraging the ISP dataset. Fig. 4 show the results. 85% of households do not have any of these plugins installed on any device they use. Only AdBlock Plus seems to be quite popular, being installed in 11% of households. However, by checking the user-agents in the requests, we observe it is installed mostly on Personal Computers, leaving tablets and smartphones unprotected even in those households. This observation demonstrates, first, that the vast majority of users are not conscious of their information being exposed when surfing web, and, second, that those who actually care about this problem are not aware of means to supervise their surfing when using mobile terminals.

In summary, we lack a comprehensive solution capable of offering Internauts visibility and control of the information they send over the Internet, independent on device/OS they use. CROWDSURF's challenge is in offering a unified, not-optional, cross-device and cross-platform system.

6. DISCUSSION

CROWDSURF design presents some practical challenges that must be faced, and ingenuity must be used to find appropriate solutions. The research community as a whole is called to design efficient algorithms, and propose scalable implementations. CROWDSURF offers this possibility, allowing anyone to contribute.

i) Protecting CROWDSURF users' privacy: For CROWDSURF is vital to maintain a good degree of visibility on the traffic sample users decide to share. However, it is fundamental to guarantee that *all* users' sensitive information is filtered out from the traffic sample before this is shared. To overcome limitations of the current CROWDSURF's anonymization mechanisms, we are investigating a methodology based on differential privacy [8].

ii) Protection from malicious biases: CROWDSURF's suggestion system might be polluted by malicious services to increase their reputation or gain popularity among users. Clearly, mechanisms to to prevent this kind of attacks must be adopted. We leave this for our future work.

iii) Usability: A key factor to let CROWDSURF reach a wide population of users is to combine CROWDSURF features with a simple, easy to use, yet effective interface to quickly inspect contacted services, customize rules, generate suggestions, etc.

iv) Web industry reaction: We believe that CROWDSURF, if widely deployed, would not incite services to contrast or boycott it. Instead, we believe CROWDSURF would lead to a better, more transparent web, where providers would be induced to improve the reliability of their services. In turn, users would reward trustful providers with their loyalty. Moreover, CROWDSURF has the potential of creating room for new services and opportunities such as, e.g., a new "market of suggestions".

iv) Complexity: We are aware that CROWDSURF introduces a considerable amount of new "mechanisms" to guide users during their online activity. However, we believe that this represents a little technological cost in comparison to the large flexibility and advantages it would introduce.

7. CONCLUSION

This paper presented CROWDSURF, a crowdsourced holistic system which allows users and companies to supervise the information exchanged in the web.

We have shown that CROWDSURF is feasible. We presented real data to support how we can build a crowdsourced knowledge supported by automatic algorithms. As a proof of concept, we implemented CROWDSURF as a Firefox plugin, whose benefits can come at a marginal performance cost for the user. Second, we provided an example of algorithm for the automatic generation of suggestions for the users.

We are aware that our idea is ambitious, as, first, CROWDSURF shall pass through a long and difficult design and standardization process to get accepted as a compelling technology by the industry. It shall undergo a deep engineering effort to convince users about its reliability, usability and effectiveness. However, as the research community is becoming more and more conscious that data constitutes a vital asset in modern Web, we are confident that the unified solution offered by CROWDSURF represents a good starting point to protect (and possibly endorse) such asset.

8. REFERENCES

[1] D. Naylor, A. Finamore, I. Leontiadis, Y. Grunenberger, M. Mellia, K. Papagiannaki, and P. Steenkiste, "The Cost of the "S" in HTTPS," in *ACM CoNEXT*, 2014.

[2] G. Acar, C. Eubank, S. Englehardt, M. Juarez, A. Narayanan, and C. Diaz, "The Web Never Forgets: Persistent Tracking Mechanisms in the Wild," in *ACM SIGSAC*, 2014.

[3] B. Krishnamurthy, K. Naryshkin, and C. E. Wills, "Privacy leakage vs. Protection measures: the growing disconnect," in *W2SP*, 2011.

[4] T.-F. Yen, Y. Xie, F. Yu, R. P. Yu, and M. Abadi, "Host Fingerprinting and Tracking on the Web: Privacy and Security Implications," in *NDSS*, 2012.

[5] H. Metwalley, S. Traverso, M. Mellia, S. Miskovic, and M. Baldi, "The online tracking horde: a view from passive measurements," in *TMA*, 2015.

[6] L. Popa, A. Ghodsi, and I. Stoica, "HTTP As the Narrow Waist of the Future Internet," in *ACM HotNets*, 2010.

[7] C. Riederer, V. Erramilli, A. Chaintreau, B. Krishnamurthy, and P. Rodriguez, "For Sale : Your Data: By : You," in *ACM HotNets*, 2011.

[8] B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu, "Privacy-preserving data publishing: A survey of recent developments," *ACM Comput. Surv.*, vol. 42, pp. 14:1–14:53, June 2010.

[9] H. Metwalley, S. Traverso, M. Mellia, S. Miskovic, and M. Baldi, "CrowdSurf: Empowering Informed Choices in the Web," *ArXiv e-prints*, Feb. 2015.

[10] Hackers steal vast eBay user database, including passwords, `http://www.bdlive.co.za/world/americas/2014/05/23/hackers-steal-vast-ebay-user-database-including-passwords`.

[11] H. Metwalley, S. Traverso, and M. Marco, "Unsupervised detection of web trackers," in *IEEE GLOBECOM*, 2015.

[12] D. J. Newman, "The double dixie cup problem," *American Mathematical Monthly*, 1960.

[13] S. Zimmeck and S. M. Bellovin, "Privee: an architecture for automatically analyzing web privacy policies," in *Proceedings of the 23rd USENIX conference on Security Symposium*, pp. 1–16, USENIX Association, 2014.

[14] Google, Microsoft, and Amazon are paying to get around Adblock Plus, `http://www.theverge.com/2015/2/2/7963577/google-ads-get-through-adblock`.