Cookies that give you away: Evaluating the surveillance implications of web tracking (Draft: April 2, 2014)

Dillon Reisman, Steven Englehardt, Christian Eubank, Peter Zimmerman, and Arvind Narayanan

Department of Computer Science, Princeton University, Princeton, NJ, USA
{dreisman,cge,ste,peterz,arvindn}@princeton.edu

Abstract. We investigate the ability of a passive network observer to leverage third-party HTTP tracking cookies for mass surveillance. If two web pages embed the same tracker which emits a unique pseudonymous identifier, then the adversary can link visits to those pages from the same user (browser instance) even if the user's IP address varies. Using simulated browsing profiles, we cluster network traffic by transitively linking shared unique cookies and estimate that for typical users over 90% of web sites with embedded trackers are located in a single connected component. Furthermore, almost half of the most popular web pages will leak a logged-in user's real-world identity to an eavesdropper in unencrypted traffic. Together, these provide a novel method to link an identified individual to a large fraction of her entire web history. We discuss the privacy consequences of this attack and suggest mitigation strategies.

1 Introduction

We investigate the power of a passive network adversary to track an individual target user or surveil users *en masse* by observing HTTP cookies in transit. This is a strong threat model, but we believe that it is well-motivated in light of recent reports of NSA "piggybacking" on advertising cookies. As per those reports, the NSA utilized its knowledge of users' Google 'PREF' cookies to target known individuals [23]. While the details revealed about the specific attacks are scant, we seek to address a more general question along these lines. Our goal is to quantify what an adversary with purely passive access to network traffic (say, on the Internet backbone) can learn about users based on their web browsing activity.

Our work starts with two insights. First, the presence of third-party cookies on most web pages, albeit pseudonymous, can tie together all or most of a user's web traffic without having to rely on IP addresses. Thus the adversary can separate network traffic into clusters, with each cluster corresponding to

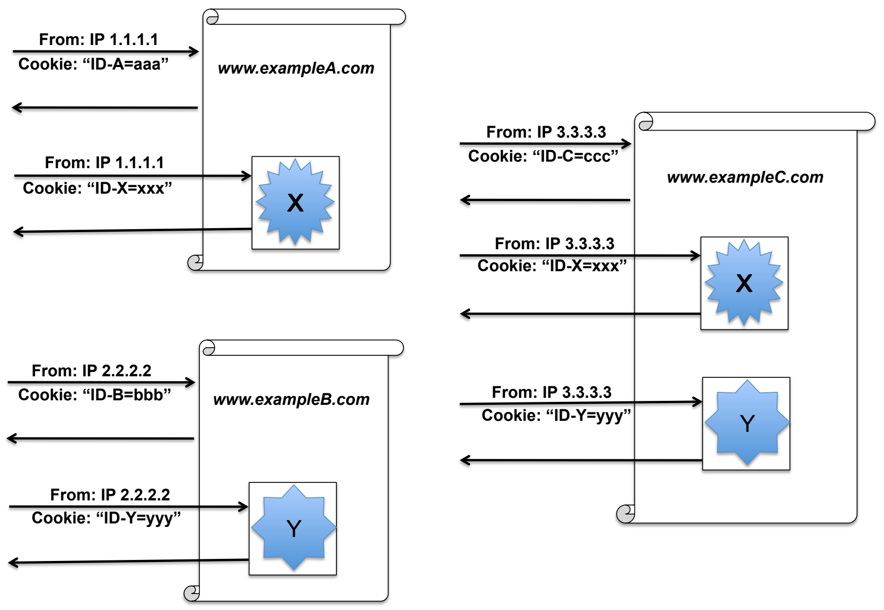


Fig. 1. Illustration of link between each of a single browser's visits to three first-party pages using two different third-party tracking cookies. The user accesses the web at three different times, behind three different IP addresses.

only one user (or more precisely, one browser instance). A single user's traffic, however, may span more than one cluster if the linking is imperfect.

Second, although most popular web pages now deploy HTTPS for authentication, many web pages reveal an already logged-in user's *identity* in plaintext. Thus, an adversary that can wiretap the network can not only cluster together the web pages visited by a user, but can then attach real-world identities to those clusters. This technique relies on nothing other than the network traffic itself for identifying targets. Even if a user's identity isn't leaked in plaintext, if the adversary in question has subpoena power they could compel the disclosure of an identity corresponding to a cookie, or vice versa.

Figure 1 illustrates the basis for our work. The adversary observes the user visit three different web pages which embed trackers X, Y or both. The user's IP address may change between visits to each page, though we assume it is consistent for the request to site A and the request to A's embedded tracker X. But there is no way to tie together her visits to pages A and B until she visits C after which all three visits can be connected. The unique cookie from X connects A and C while the one from Y connects B and C. We assume here that the user has visited pages with both trackers before so that cookies have already been set in her browser and will be sent with each request.

While IP address is a *convenient* method to link a request to a first party page to the corresponding request to an embedded third party tracker, it is not necessary. In Section 5.1 we show how this linkage can be achieved even if the IP address cannot be observed at all or if an IP address is shared by many users.

Contributions. Our contributions are both conceptual and empirical. First, we identify and formalize a new privacy threat from packet sniffing. While the technique of utilizing cookies to target users is well known, we forumlate the attack concretely in terms of the following steps: (1) automatically classifying cookies as unique identifiers (2) using multiple ID cookies to make *transitive* inferences and *clustering* HTTP traffic, and (3) inferring real-world identity from HTTP traffic itself.

Second, we rigorously evaluate the above attack model by simulating realistic browsing behavior and measuring the actual cookie ecosystem. This requires nuanced techniques along at least three fronts: (1) a crawling infrastructure based on *browser automation* to more closely simulate real users (2) a model of browsing history derived from real user behavior, and (3) analytic techniques for cookies to determine which cookies in the observed dataset are unique identifiers.

Results. We simulate users that make 0 to 300 web page visits spread out over a 2–3 month period. For each such set of visits, we perform clustering using the method described above and find the "giant connected component."

At a high level, our results show that for web pages that have any third-party tracking cookies at all, over 90% of visits fall into this connected component. The clustering effect is extremely strong and is robust to differences in the models of browsing behavior and cookie expiration. It applies even if the adversary is able to observe only a small, random subset of the user's requests. We find that on average, over two-thirds of time, a web page visited by a user has third-party trackers.

Second, we measure the presence of identifying information in plaintext among popular (Alexa Top 50 U.S.) websites. 60% of websites transmit some form of identifying information in plaintext once a user logs in, whether first name, a combination of first and last name, a username or an email address. These identifiers can be trivially sniffed by an adversary. While a name can help narrow the possibilities for the anonymous user's identity, a username or email address is often a unique identifier. 44% of websites present such unique identifiers in the clear.

Implications An adversary interested in targeted surveillance can proceed as follows: (1) Either scan for the target's identity in plaintext HTTP traffic, or use auxiliary methods to obtain the target's cookie ID on some first-party page (2) From this starting point, transitively connect the target's known first-party cookie to other third-party and first-party cookies of the target. On the other hand, an adversary interested in *en masse* surveillance can first cluster all observed HTTP traffic, albeit at a high computational cost, and then attach identities to these clusters using the methods above. Our attacks show the feasibility of either adversary's goal.

After attaching an identity to a cluster of web traffic, the adversary can go in one of three directions. First, the browsing history itself could be the sensitive information of interest—for example, one of the leaked Snowden documents refers to a plan to discredit 'radicals' based on browsing behavior, such as pornography [3]. Second, further sensitive information could be gleaned from preferences, purchase history, address, etc. that are transmitted unencrypted by some web pages. Finally, it could be used as a stepping point for active attacks such as trying to get a target user to download malware [23].

Our measurement of the prevalence of popular web pages revealing identity in plaintext to logged-in users has implications for other, related adversaries, such someone running a packet sniffer in a coffee shop. Our results suggest that such an adversary can determine the identities of customers within a few minutes of typical browsing activity, although follow-up experiments on live user sessions are required for an accurate quantification. Such deanonymization violates users' intuitive expectations of privacy and can be a stepping stone to a real-life social engineering attack.

2 Background and threat model

Cookies. When a user visits `example.com`, her web browser will send HTTP requests to `example.com`'s server as well as any third-party domains whose content is embedded on the page. Third-party plugins and specialized trackers are ubiquitous on the web. In addition to advertisements, content hosted on CDNs, and third-party content such as social plug-ins, there are also invisible trackers such as advertising networks and exchanges, analytics services and data brokers [17]. One study estimated that the average top-50 website contains 64 independent tracking mechaisms [25].

Each of these domains may place cookies on her browser, respectively called first-party cookies and third-party cookies. It is common for servers to set a cookie containing a pseudonymous identifier so that the client can always be uniquely identified. Cookies may exist only for a single browsing session (a "session cookie") or be stored for a server-specified amount of time (a "persistent cookie"). If a third-party advertiser, for instance, has ads on multiple web pages, then a persistent cookie containing the same unique identifier will be sent alongside a visit to any of those pages. It is these unique and persistent cookies, especially third-party cookies, that are of interest to us.

NSA leaks. Two of the leaked NSA documents motivate our work. A stand-alone leaked slide says that the agency "provided TAO/GCHQ with WLLids/DSL accounts, Cookies, GooglePREFIDs to enable remote exploitation" [23]. A presentation titled "Tor stinks" notes that while the Tor browser prevents linking based on cookies, if Tor is used improperly, cookies – including `doubleclick.com` cookies – can be used to link a user's Tor browsing to her non-Tor browsing [4]. Another slide in the same presentation describes potential active attacks on Tor users. Based on this scant information, there has been much speculation about these attacks.

Threat model. We are not specifically concerned with what the NSA or any other specific entity may or may not be doing in practice. Rather, we aim to quantify the power of the technical capabilities that we know to be possible. We consider only passive attacks for several reasons. First, passive attacks appear to be more powerful than generally realized, and we wish to highlight this fact. Second, passive attacks are easier to study rigorously because the attack model is simple and well-defined, whereas the spectrum of active attacks is open-ended in terms of the attacker's abilities. Third, even an active attack requires some form of eavesdropping as a first step. Finally, almost all active attacks carry some risk of detection, making passive attacks much easier to mount.

We consider a powerful adversary with the ability to observe a substantial portion of web traffic, either at an ISP or on the Internet backbone. We also assume that the adversary can't routinely compromise HTTPS, so cookies or other identifying information sent over HTTPS are of no use.

The adversary may have one of two goals: first, the adversary might want to target a specific individual for surveillance. In this case the adversary knows either the target's real-world identity or a single ID cookie known to belong to the target (whether on a domain that's typically a first party or on a tracker domain). Second, the adversary might be engaged in mass surveillance. This adversary would like to "scoop up" web traffic and associate real-world identities with as much of it as possible.

The adversary's task is complicated by the fact that the IP addresses of the target(s) may change frequently. A user's IP address could change because she is physically mobile, her ISP assigns IP addresses dynamically or she is using Tor. Browsing from a smartphone is a case worth highlighting: Balakrishnan et al. find that "individual cell phones can expose different IP addresses to servers within time spans of a few minutes" and that "cell phone IP addresses do not embed geographical information at reasonable fidelity" [9].

To link users across different networks and over time, the adversary aims to utilize first-party and third-party unique cookies assigned to browser instances by websites. He can easily sniff these on the network by observing the "Cookie" field in HTTP request headers and the "Set-Cookie" field in HTTP response headers. Cookies set by an "origin" (roughly, a domain) that have not expired are automatically sent as part of requests to the same origin.

To learn a user's real-world identity, the adversary could sniff an authentication cookie that is transmitted in the clear, and present the sniffed cookie to the website, thus essentially logging in. We consider this and other active attacks out of scope, both because there is a high risk of detection, especially if the attack is carried out *en masse*, and because it is difficult to rigorously evaluate (for instance, websites may tie login cookies to IP addresses, defeating the attack).

3 Methodology

Our methodology consists of the following distinct parts: the creation of a realistic model of browsing behavior, a browser automation and measurement platform and an analysis of the resulting data. On a high level, we wish to simulate real users browsing over a period of time, detect the creation of unique identifiers, and measure the flow of both unique pseudonymous identifiers and real-world identifiers. This necessitates the creation of realistic user profiles, an infrastructure with which to deploy the profiles and a platform to perform an analysis in an automated and scalable fashion. The infrastructure used in this project is an instance of a general platform which we developed, building upon cutting-edge browser automation and measurement tools.

3.1 Infrastructure

Requirements. This study has a wide range of often-conflicting requirements, presenting a significant system design and engineering challenge. First, we wish to replicate a user's interaction with the web as faithfully as possible, both in the technologies used and in the patterns of use, in order to be able to draw inferences about real users based on our simulated experiments. Second, the platform must be fully automated and scalable – our experiments required hundreds of thousands of page visits overall. Third, the platform must provide profile management functionality, providing the ability to save and load a simulated user's history and cookie databases in a modular, scripted way. Fourth, the platform requires a full picture of cookie interaction – we need to monitor all third party requests, responses, and cookies. Fifth, the platform must be portable and headless, as crawls need to be able to run in parallel on local and remote machines regardless of display capabilities. Lastly, the collected data must be stored in an easily searchable, structured format – we need the capability to search cookie strings using regular expression patterns and link cookie interactions to the involved first and third parties.

A major hurdle for automated web measurement in general is that nearly all of the tools available are meant for real user interaction or interaction on a semi-automated level. Web browsers and web pages are meant for manual interaction by users (e.g. dealing with a pop-up or alert window) or canceling and refreshing a page when it doesn't load fully. Browser automation tools are often built for web development testing on a single site and may have limited feature sets for general searching or handling sites with a wide array of load times and objects. An additional requirement of our platform is to incorporate a layer of abstraction that removes the need for manual interaction and monitoring from these underlying tools and provides stability in an automated setting.

Design and implementation. We drive our data collection and measurement infrastructure (see Figure 2) using a web crawler built around the Selenium WebDriver.[1] Selenium WebDriver is a browser automation tool which can control most modern web browsers by hooking into the browser's WebDriver API to enable automation of simulated user behavior from several languages, including Python and Java.[2] The WebDriver API provides an interface for scripted tools

[1] `http://docs.seleniumhq.org/projects/webdriver/`
[2] `http://www.w3.org/TR/webdriver/`

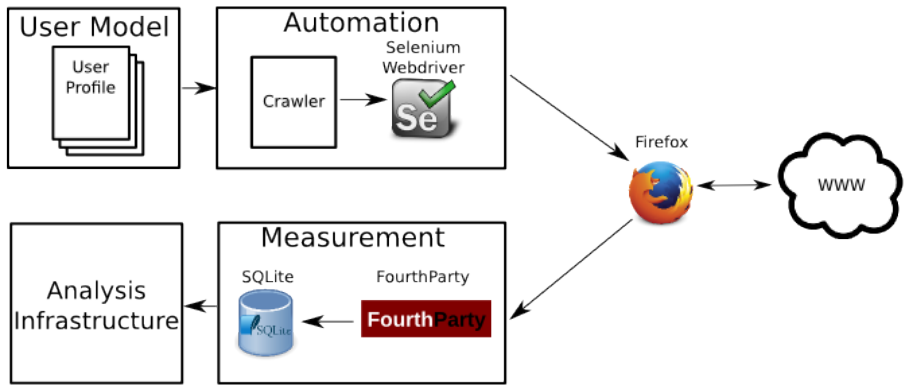


Fig. 2. Data collection and measurement platform - our automation infrastructure drives a Firefox browser in order to simulate a user profile from our user model and automatically analyzes data collected throughout browsing sessions.

to execute common browsing tasks such as loading a page and clicking elements, as well as an interface to search and parse the contents of a page.

In our study, we use Selenium to drive Firefox instances, since Firefox is a modern browser used by many users on the web and also provides a strong plugin infrastructure. This allows us to use FourthParty, a web measurement platform built by the Stanford Security Lab, which serves as our primary method for logging browser data.[3] FourthParty is a Firefox extension which monitors all dynamic web content on the browser, including HTTP requests and responses, cookie interactions and JavaScript calls. In particular, we hooked into FourthParty's HTTP request and response tables to determine which cookies are transmitted under each page load, and the cookie table to detect and monitor unique identifiers. FourthParty's SQLite database enables quick aggregation and searching of cookie values through standard SQL. FourthParty offers usability benefits over a pure HTTP proxy approach and also tracks all cookie read, delete and edit operations.

When compared to driving crawls exclusively through the Selenium WebDriver, our architecture presents a few distinct advantages. A profile management system simplifies the task of loading and saving browser histories for crawls and easily scales to support the simulation of any number of users from any source of history. This level of scale is extended to the automated processing of FourthParty databases, a tool which is designed for aggregation and analysis on an individual user level. We make use of pyvirtualdisplay to interface with Xvfb, a virtual X display server, and run Firefox instances entirely headless. This allows ease of portability between machines enables simultaneous crawls to occur on a remote machine with no X display server.

Our infrastructure continually monitors the state of deployed crawls to handle exceptions, stalls, extraneous windows and other issues which often occur during

[3] `http://fourthparty.info/`

browsing and is able to smoothly recover to the same point with very little downtime. If an issue is detected or a process timeout is triggered, it saves the current crawl state before killing the Selenium and Firefox instances. A fresh WebDriver instance is created with the same Firefox state re-loaded, ensuring that the crawl continues with the same user profile and state it had prior to the issue.

We had many browser automation options to choose from when building our infrastructure. Other browser automation tools such as PhantomJS and SlimerJS also provide full history, cookie and JavaScript support. However, unlike those frameworks, Selenium provides a more extensive range of web technologies, including user plugins, addons and HTML5 features. Potential differences in the underlying parsing engine, user agent string, and interaction with web technologies also motivated our choice to use Selenium to drive a consumer web browser over the stripped down alternatives. More lightweight tools, such as *wget*, were not considered as they do not provide the necessary range of supported features.

3.2 Browsing profiles

Our browser automation and measurement infrastructure requires as input a model of browsing behavior. One of our models was a naive one – the user visits random subsets of the Alexa top 500 sites. We created more realistic browsing profiles using the AOL search query log dataset which we describe below. This is almost identical to the method that was used in [16].

The AOL search query log dataset contains the queries made by 650,000 anonymous users over a three month period (March–May 2006). Our procedure for creating a browsing profile from a user's search query profile is as follows. First, we remove repeated queries. Next, for every search query performed by the user, we submit the query to Google search and retrieve the links for the first five results. Users were selected on the basis that they performed between 50 to 100 unique queries which resulted in browsing profiles of 250 to 500 URLs.

Of course, only a subset of real users' web browsing results from web searches. Nevertheless, we hypothesize that our profiles model two important aspects of real browsing histories: the distribution of popularity of web pages visited, and the topical interest distribution of real users. Popular websites may embed more trackers on average than less popular sites, and websites on the same topic may be more interconnected in terms of common embedded trackers. Therefore failure to model these aspects correctly could skew our results.

The reason we recreate the users' searches on a current search engine rather than simply using the sites visited by the AOL users (which are recorded in the AOL dataset) is that the distribution of websites visited by real users changes over time as websites rise and fade in popularity, whereas the distribution of users' interests can be expected to be more stable over time.

The ideal way to create browsing profiles would be to directly acquire real user browsing histories. We know of two such datasets [14, 5]. We attempted to acquire these, but found them either no longer available or prohibitively expensive.

3.3 Detecting unique identifier cookies

Cookie text files contain several pieces of information, including the host (the domain who owns it) and a name-value string. For our analysis, we consider a *cookie* to be the unique (owner's domain, name) pair and the cookie's *value* to be the value component of the name-value string (see the Cookie text in the PREF cookie below). Our most fundamental analytical task was to effectively identify cookies with value strings that correspond to unique identifiers.

```
Host : www.google.com
User-Agent : Mozilla/5.0 (X11; Ubuntu;
Linux x86_64; rv:26.0) Gecko/20100101
Firefox/26.0
...
Referer : http://www.unity3d.com/gallery
Cookie : PREFID=5834573d6649ab5
```

To be useful to the adversary as identifiers, cookie values must have two important properties: persistence over time and uniqueness across different browser insetances. Based on these criteria we develop heuristics that classify cookies as identifiers and attempt to avoid false positives. Our heuristics are intentionally conservative, since false positives risk exaggerating the severity of the attack. Our method does have some false negatives, but this is acceptable since it is in line with our goal of establishing lower bounds for the feasibility of the attack.

We define a cookie to be an identifier cookie if its value string:

- **Is long-lived** – the cookie's observed expiry date at time of creation is sufficiently far in the future
- **Remains stable over time within a browser instance** – once set, the cookie value does not change over time and over multiple visits to the cookie domain and even if the browser is restarted
- **Varies across browser instances** – has a different value in all the crawls we have observed
- **Passes the entropy test** – value strings are sufficiently different across different browser instances so as to serve a globally unique identifier
- **Is of constant length** – the length of the value string is invariant across browser instances

Long-lived cookies are non-session cookies with expiry times longer than three months. The three month cut-off was chosen to reflect the fact that the AOL dataset on which a majority of our crawl data is based was collected over a period of three months. A simple accounting for expiry time would require a cookie set at the beginning of the period of observation to survive until the end. Our baseline ignores cookies that fail this criterion as they are too transient to track long-term user behavior. During our analysis however we tune this parameter to better explore other methods of modeling expiry time by analyzing all non-session cookies.

Stable cookies have value strings that, for a given browser, remain stable across multiple browsing sessions. Cookies with dynamic value strings may simply be logging timestamps and other non-identifying information. We believe that other non-static cookies contain identifying information but do not fit within our study. For example, (google.com, __utma) changes when data are sent to Google Analytics.[4] Although we believe that cookies encoding some part of an individual's browsing patterns may indeed leak personal information, their dynamic nature excludes them for our definition of persistent identifier.

User-specific cookies have value strings that are unique across different browser instances in our dataset. Cookies that fail this criterion only mark a given browser as belonging to larger set of browsers that have been marked with a given string. An extreme case is (doubleclick.com, test_cookie), which has the same value (CheckForPermission) across all users in our sample.

Low-entropy cookies have values that differ across different users' browsers in our measurements but are not sufficiently different enough as to be truly unique identifiers. For instance, value strings that incorporate fixed timestamps (e.g. the time that the cookie was created) may differ across different users by a few digits but are likely to not be globally unique. To mitigate this issue, we used the Ratcliff-Obershelp [11] algorithm to compute similarity scores between value strings. We filtered out all cookies with value strings that were more than 90% similar to value strings from the corresponding cookies from different crawls.

Constant-length cookies have value strings with the same, fixed length across all our datasets. We believe that unique cookie identifier strings are typically generated in a standard, fixed-length format. This belief is motivated by patterns seen during manual inspection and the likelihood of third party libraries generating identifiers. As such, we only consider constant length cookies, keeping in mind that this heuristic may cause false negatives that render our analysis in fact overly conservative.

To collect data to identify unique cookies, we ran two simultaneous crawls using identical sets of websites visited in the same order. By conducting simultaneous measurements, we avoid the problem of sites changing their cookie interaction behavior depending on a user's browsing time. For instance, in relation to the entropy heuristic, cookies with values that depend on time stamps will be easier to detect and ignore if the crawls have nearly the same timestamps for all actions.

Once the data were collected, we applied our heuristics when comparing cookies with the same keys across the different datasets in order to identify the cookies most likely to be unique identifiers. Since our passive adversary in our model cannot observe traffic sent over HTTPS, we also disregarded cookies with the is_secure flag set to 1.

[4] `https://developers.google.com/analytics/devguides/collection/analyticsjs/cookie-usage`

3.4 Crawls

Our crawling infrastructure ran headless crawls on a remote Linode instance running Linux over a four-day period in early 2014. We ran 45 crawls based on AOL user profiles and another 20 crawls based on random 200-site subsets of the Alexa Top 500 US sites. Each crawl was performed twice in parallel for the purpose of finding unique identifiers.

Web pages were visited approximately once every ten seconds. We set the timeout in our crawler to restart the browser under the same Firefox user profile should the browser crash and save the crashed browser's Fourthparty database.

When the crawls are complete the "cookie" tables from each crawl's collection of Fourthparty databases is analyzed for unique identifiers using our heuristics described above.

Linking first-party pages via third-party trackers Once we determine which cookies contain unique identifiers, we use the Fourthparty databases (specifically the HTTP request and response headers tables) from the user's crawl to take count of which first-party pages transmit persistent, identity-bearing cookies. We construct a bipartite graph made up of first-party nodes of type (first_party_page) and of third-party nodes of type (cookie_host, cookie_name). Two nodes share an edge when a tracker's cookie is transmitted while the browser fetches a certain first-party page, giving us the foundation for the clustering effect. We consider the giant connected component to be the largest collection of first-party pages connected via common third-party trackers.

3.5 Survey of popular websites for identity-leakers

The privacy problem of connectivity in the graph of would be further deepened if the adversary could link the connected graph of a user's web pages to a real-world identity. We set out to determine the likelihood that our HTTP-observing adversary would see some aspect of real-world identity leaked in plaintext across the network in the course of wiretapping. To do so we conducted a survey of popular web pages that are the most likely to have user accounts. This survey led us to identify 50 of the Alexa top 68 US sites that allow for account creation and signed up test accounts when possible. The top web pages are a useful representation for how heavily-used sites with user accounts manage data as seemingly benign as a user's first name, and are also more likely to occur in a typical user's browsing history.

4 Results

In the course of analyzing our crawl data we found that out of 15,225 page visits, 10,138 of the page visits had cookies that we classified as a unique identifier. Throughout this section, we only take into account those page visits that embed a unique tracking cookie and consider what proportion of those are located in the giant connected component. Visits to pages without tracking cookies cannot possibly be connected to other page visits using our technique.

4.1 Clustering

Measuring clustering for a single user We are primarily interested in the number of web pages visited by a user located in the giant connected component (GCC) relative to the number of pages with tracking cookies visited by the user. We focus on large connected components/clusters because the probability that a cluster will have at least one page visit that transmits the user's real-world identity in plaintext increases linearly with the size of the cluster. Manual inspection shows that page visits not in the large connected components belong to very small clusters, typically singletons, and are thus unlikely to be useful to the adversary.

We must also consider that the adversary's position on the Internet backbone might not give them a full view of any user's traffic. A user's changing location on the network or alternative routes taken by packets through the network might result in gaps in the adversary's data collection. To model the adversary's partial view of the user's browsing activity, we repeat our analysis with random subsets of web pages of various sizes.

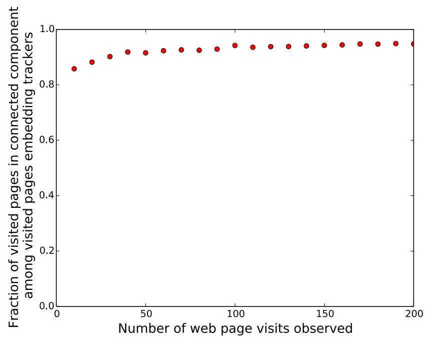


Fig. 3. AOL User 9472615, random subsets of page visits

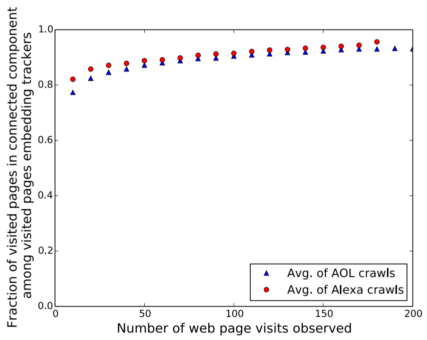


Fig. 4. AOL and Alexa, random subsets, 3 month cut-offs

For illustration, Figure 3 shows how the GCC of a single AOL user's page visits (y-axis) grows as we vary the completeness of the adversary's view of the user's HTTP traffic (x-axis). Each data point was computed by taking 50 independently random samples of page visits. For each sample we apply the clustering algorithm and compute the fraction contained in the GCC. We then average the fractions across the 50 samples. Since we wish to simulate these page visits being spread out over time, only cookies with expiration times at least three months into the future were included when computing the clusters. Thus for each (x, y) pair we can say that if the adversary captures x web page visits by a user in the course of a wiretap, they could link approximately $y\%$ of those visits into a single cluster. The numbers we see for this user are typical — the fraction is around 80% for even very small clusters and exceeds 90% as the cluster size increases.

All AOL Users Now we present our primary result. We repeated the process described above for 45 user profiles from the AOL dataset, with the results averaged across all user profiles. Figure 4 shows that a random sample of only ten web pages visited by the user led to 77.3% of the web pages being located in the GCC on average. After 100 web page visits observed by the adversary, the GCC reaches 90% coverage of those pages.

Effect of the model: Alexa profiles, explicit cookie expiry, and chronological visits Next, to see if our results are significantly affected by the specifics of the model we used, we varied several parameters and recomputed our results. First, Figure 4 compares the use of Alexa profiles (i.e., random subsets of Alexa US top 500 websites) against the AOL profiles; the results were very similar. Second, we modeled the adversary seeing subsets of web pages that the user visited in chronological order, rather than a random subset (perhaps the adversary was only able to observe the user for a short period of time). The results, shown in Figure 4.1, are again similar except that there is an even higher degree of clustering than observing random subsets. We hypothesize that this is because users visit multiple pages on the same domain in contiguous periods, resulting in common sets of trackers and consequently greater clustering.

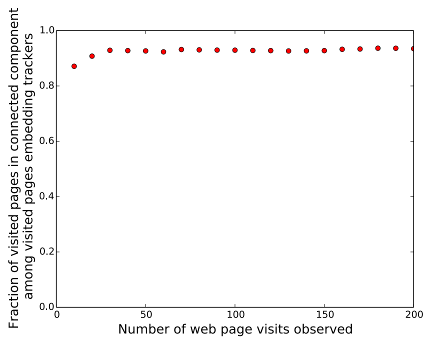


Fig. 5. AOL user page visits analyzed in chronological order

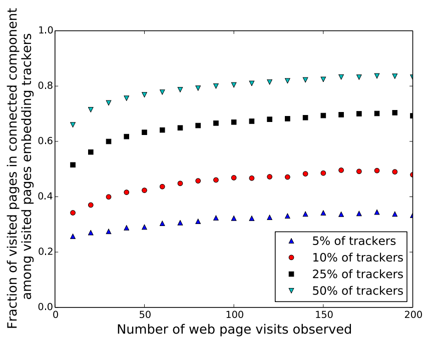


Fig. 6. Reduced size random samples of third-party trackers

Third, rather than the somewhat crude (but conservative) heuristic of assuming that unique cookies with expiry times of over three months can be used by the adversary to link visits, we implemented an explicit model of cookie expiry, which we describe in detail in Appendix A.1. The results were visually indistinguishable from the 3-month cookie expiry model, so we omit them (in fact, the distance between the two distributions was less than 10^{-3}).

Effect of mitigation attempts by trackers We now investigate what happens if some, but not all, trackers switch to HTTPS or employ one of the other mitigation methods described in Section 5.2. Figure 6 shows that a random 50% of trackers cleaning up their unique IDs makes very little difference to the attack, and it takes 90% of attackers to change in order to bring the fraction of clustered page visits to under 50% of all page visits.

In summary, we find that most of a user's observed page visits can be linked together by the adversary even if only a small number of page visits are observed.

The results are largely unaffected by the model of user browsing behavior; the only factor that makes a difference is if a very large fraction of web trackers employ mitigation mechanisms.

4.2 Identity Leakage

Table 1 summarizes our results from a manual survey of the Alexa US sites. We picked the top 50 sites that support account creation. 34 of the 50 websites used HTTPS to secure login pages,[5] and only 15 of those sites continued to use HTTPS to secure future interactions with the user.

Plaintext Leak Type	**Percentage of Sites**
`First Name`	28%
`Full Name`	12%
`Username`	30%
`Email Address`	18%
`At least one of the above`	60%

Table 1. Leakage of Identifiers on Alexa Top 50 supporting user accounts

Although a majority of sites secure user credentials on login pages, personally identifying information (name, username, email address) is transmitted much more frequently via HTTP. Over half of the surveyed sites leak at least one type of identifier, and 44% (not shown in table) leak either username or email address, which can be used to uniquely infer the user's real-world identity.

A representative example of the web's identity-leak problem is `www.youtube.com`. As a Google property, users sign in via a secure login screen with their Google account. Other Google domains like `www.gmail.com` will continue to interact with the browser using HTTPS by default, but depending on a user's settings `www.youtube.com` will default to a non-HTTPS page after login and transmit the user's full name and email address (Appendix A.2).

Furthermore, we verified that pages from these popular sites that leak identity occur in the clusters of web pages found in our attack. Specifically, at least 6 of the 30 sites we found to leak some type of identity were found in the giant connected component of *every one of the 45 AOL user browsing profiles*. Of course, there are likely also many sites outside the top 50 that leak identity and are found in these clusters, but we did not measure these.

Taken together with our results on clustering, our measurements show that a passive attack is highly feasible: after observing only a fraction of a user's web traffic spread out over time and source network location and interspersed with traffic from millions of others, the adversary will be able to link the majority of the user's web requests together and furthermore, use the contents of the responses to infer the user's real-world identity.

[5] We did not separately measure if login form submission was via HTTPS. If the form is loaded insecurely but submitted securely, the site is vulnerable to an active attack that steals login credentials, but not a passive one.

5 Discussion

5.1 Linking without IP address

So far we have assumed that the adversary sees the same source IP address on a request to a first-party site and its corresponding third-party tracker, and that this can be used to link the two requests. There are at least two scenarios in which this assumption is problematic. The first is a NAT. If two users, Alice and Bob, behind the same NAT visit the same web page at roughly the same time, the adversary sees the same IP address on all ensuing HTTP requests. Therefore it is not clear how to disambiguate Alice's traffic from Bob's, and, in particular, disambiguate Alice's cookies from Bob's. The user-agent headers on the requests might be different but this approach may not be reliable enough on its own.

The other scenario is when the user employs Tor without application layer anonymization. (If the user is using a properly configured Tor setup, such as the Tor browser bundle, this attack does not work at all). Here, the attack works only if the adversary is located on the Internet backbone, rather than, say, at the user's ISP, because Tor encrypts circuits all the way to the exit node and the adversary can sniff cookies only on the path from the exit node to the receiver. Since Tor will, in general, use different circuits for communicating with different servers, the adversary will see different source IPs for the two requests (or may be able to observe only one of the requests).

We propose a simple technique to link requests without using the IP address. It relies on the following observation: if a cookie value a associated with page A's domain and a cookie value x associated with an embedded tracker domain X are observed *multiple times* near-simultaneously (e.g. within 1 second of each other), then a and x are probably associated with the same user. This is because it is unlikely that different users will visit the same page near-simultaneously at various different times.

Unlike the use of IP address for linkage, this method incurs a few false positives. Intuition suggests that for all but the busiest of web pages, two or three visits may be sufficient to link the first-party and tracker cookies with each other. However, this claim cannot be rigorously evaluated without access to large-scale HTTP traffic and so we leave this as a hypothesis.

5.2 Mitigation by trackers

Trackers may be able preventing a passive eavesdropper from piggybacking on their unique identifiers if they are only transmitted over HTTPS. Deploying HTTPS for all connections can be expensive, though some trackers already support HTTPS for embedding in HTTPS pages as this is necessary to avoid mixed content warnings. There are also subtle issues, such session tickets used for TLS session resumption, which can be used by an eavesdropper to link multiple HTTPS connections to the same browser instance just like a tracking cookie.

Without deploying HTTPS, trackers could use *ephemeral cookies* which vary every time they are submitted. The simplest way to achieve this is to set a new cookie with every request so that each cookies is only sent once. However, this is highly vulnerable as an eavesdropper with all traffic can trace the chain of cookies sent and received from each user and potentially bridge gaps from missing traffic by linking traffic from pages with distinct sets of trackers.

A better approach is for the browser to dynamically generate ephemeral cookies. This could be achieved by the tracker transmitting a public key, and using Javascript to randomly encrypt a long-term identifier to generate an ephemeral identifier for each connection. The downside of this approach is that public key operations can relatively expensive to compute in Javascript. An alternate approach if trackers can deploy HTTPS on a limited scale is to transmit a large number of one-time use tokens over HTTPS and have the browser transmit one with each subsequent HTTP connection, re-connecting over HTTPS to request more tokens when needed. This requires far less computation, but more data transmission and some HTTPS deployment.

Unfortunately, a large fraction of trackers would need to deploy such mitigation strategies for thems to make a dent in the adversary's overall chances. As we showed in Section 4.1, the feasibility of traffic clustering is barely affected if cookies from the top 10 most prevalent trackers are unavailable.

5.3 Implications

Our attack shows the feasibility of both targeted and *en-masse* surveillance of web traffic. The computational requirements are not onerous in the context of surveillance – while it requires inspection of packet contents, the information needed for further analysis, namely, URLs, IP addresses and cookie headers, are compact and can easily be extracted from streaming data using regular-expression filters. Since there is an ongoing policy debate about the acceptability of storing "metadata," we remark that HTTP headers indeed fall into this category.

Our results cast even more doubt on the argument that third-party web tracking does not raise privacy issues because it is "anonymous." As we show, even if no single tracker connects a user's browsing history to her identity, this veil of pseudonymity is easily pierced by a traffic-sniffing adversary. In the absence of third-party tracking cookies, the attacker's ability would be significantly circumscribed by changes in the user's IP address which can happen for a variety of reasons.

Our work adds to the evidence that in many cases, utilizing the infrastructure created by private companies might be the technologically cheapest, but such a strategy presents a much more efficient way for governments to carry out surveillance. It underscores the importance of deploying HTTPS for web privacy; the ramifications go beyond any single domain. Similarly, our study highlights the difficulty of proper application-layer anonymization and problems with the use of Tor without proper attention to the application layer. In particular, it suggests that technologies like Safeplug that act as a "Tor appliance" are ineffective against an eavesdropper.

5.4 Limitations

A couple of important limitations of the attack must be pointed out. First, using the Tor browser bundle likely defeats it. “Cross-origin identifier unlinkability” is a first-order design goal of the Tor browser, which is achieved through a variety of mechanisms such as double-keying cookies by first-party and third-party [21]. In other words, the same tracker on different websites will see different cookies. However, our measurements on identifier leakage on popular websites apply to Tor browser usage as well. Preventing such leakage is not a Tor browser design goal.

Simply disabling third-party cookies will also deter the attack, but it is not clear if it will completely stop it. There are a variety of stateful tracking mechanisms in addition to cookies [12, 15, 27, 17], although most are not as prevalent on the web as cookies are.

We also mention two limitations of our study. First, while a significant fraction of popular sites transmit identities of logged-in users in the clear, we have not actually measured how frequently typical users are logged in to the sites that do so. Anecdotal evidence suggests that this number must be high, but experiments on actual user sessions are required for an accurate estimation of vulnerability.

Second, while we take pains to model the adversary’s view of our simulated users’ traffic as if it were spread out over a 2-3 month period (as the searches in the AOL logs were), our actual crawls were conducted over a short time span (10 seconds between visits). There is one way in which our results might differ if our crawls were spread out: cookies might list expiry times well into the future, but servers might expire them ahead of schedule. To check that this would not be the case, we utilized the independent cookie measurement database *Cookiepedia.* [6] For a sample of cookies in our dataset with long expiry times, we manually verified that they are actually long-lived as measured in the wild by Cookiepedia.

5.5 Ongoing and future work

We now describe several threads of ongoing and planned work to explore related aspects of the problem.

Linking different devices of the same user. Our attack as described does not allow the adversary to link two different browsers or devices of the same user, except by attaching real-world identities to both clusters. However, there could be other mechanisms: if two sets of cookies are repeatedly seen together on a variety of different networks, this probably represents the same user who is switching IPs (for example, traveling with a smartphone and a laptop). Another possibility is that the set of websites that the user tends to visit frequently could serve as a fingerprint.

Effectiveness of user mitigation mechanisms. While using the Tor browser and blocking all third-party cookies are good mitigation mechanisms,

[6] `http://cookiepedia.co.uk/`

they come at a cost in terms of usability and functionality. But there are other browser privacy measures that are more palatable for users to deploy: (1) an extension such as "HTTPS everywhere" which makes requests over HTTPS whenever the server supports it and (2) Safari-style cookie blocking which is a limited form of third-party cookie blocking that breaks less functionality. Since these are essentially modifications to browser behavior, they can be studied using our approach.

Large-scale analysis of identifier leakage. Our measurements of the ability of eavesdroppers to observe identities were done manually. Creating a "census" of identifier leakage across thousands of websites would require automated account creation which is both technically infeasible and ethically and legally problematic. However, there is a shortcut: we could use federated authentication (such as Google/Facebook OAuth) to log in to any website supporting such authentication without having to create accounts on per-site basis. Automatically logging in requires sophisticated browser automation and is an engineering challenge, but we have made significant headway in solving it.

Leakage of sensitive attributes in HTTP traffic. As mentioned in Section 1, one of the adversary's goals might be to infer sensitive attributes about the user that are transmitted by between the browser and websites in plaintext, such as preferences, purchase history, address, etc. We plan to develop heuristics to identify such data in HTTP traffic and measure the prevalence of such leakage. A further step would be to extend the measurement of unencrypted transmission of sensitive information to the domain of mobile apps.

6 Related Work

The most closely related body of work to our study is the emergent field of *web privacy measurement*, which aims to detect and measure privacy-infringing data collection and use online. While most research on web privacy measurement is concerned with what websites learn about users, our concern is an eavesdropper. Nonetheless, the methods we use are heavily drawn from this field of study.

There have been several notable results on detection and measurement of various types of online tracking, including the detection of Google bypassing Safari cookie blocking [7], various studies of flash cookies (LSOs) including respawning behavior [22, 8, 18]. Some studies that have utilized large-scale automated web measurement platforms include a measurement of browser and device fingerprinting [20, 6] and a survey of mobile web tracking [13].

Some privacy studies have focused on how personal data is used online rather than how it is collected, including detecting search and price discrimination based on browsing history [19] and other factors [24], measurement of targeted advertising [16, 26], a study of the effectiveness of privacy tools for limiting behavioral advertising [10].

There are various client-side tools to block, limit or visualize third-party tracking, built by researchers, hobbyists, or companies. These are too numerous to list exhaustively, but a sampling include Adblock Plus, Ghostery, ShareMeNot [1], Lightbeam and TrackingObserver [2].

7 Conclusion

While much has been said from a legal, ethical and policy standpoint about the recent revelations of NSA tracking, many interesting *technical* questions deserve to be considered. In this paper we studied what can be inferred from the surveillance of web traffic logs and established that utilizing third-party tracking cookies enables an adversary to attribute traffic to users much more effectively than methods such as considering IP address alone.

We hope that these findings will inform the policy debate on both surveillance and the web tracking ecosystem. We also hope that it will raise awareness of privacy breaches via subtle inference techniques. Finally, while it is unrealistic to expect a large percentage of web trackers to switch away from unique cookies transmitted in plaintext, the problems we identified with non-use of HTTPS on first party sites are more readily fixable and we hope that our results will provide an impetus for doing so.

Acknowledgments We would like to thank Joseph Bonneau, Ed Felten, and Jonathan Mayer for their feedback and input.

References

1. ShareMeNot: Protecting against tracking from third-party social media buttons while still allowing you to use them. `https://sharemenot.cs.washington.edu`. Accessed: 2014.
2. TrackingObserver: A browser-based web tracking detection platform. `http://trackingobserver.cs.washington.edu`. Accessed: 2014.
3. NSA 'planned to discredit radicals over web-porn use'. `http://www.bbc.co.uk/news/technology-25118156`, November 2013.
4. 'tor stinks' presentation - read the full document. `http://www.theguardian.com/world/interactive/2013/oct/04/tor-stinks-nsa-presentation-document`, October 2013.
5. Databases in WRDS - comScore. `http://wrds-web.wharton.upenn.edu/wrds/about/databaselist.cfm`, December 2014.
6. Gunes Acar, Marc Juarez, Nick Nikiforakis, Claudia Diaz, Seda Gürses, Frank Piessens, and Bart Preneel. FPDetective: dusting the web for fingerprinters. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 1129–1140. ACM, 2013.
7. Julia Angwin and Jennifer Valentino-Devries. Google's iPhone tracking: Web giant, others bypassed Apple browser settings for guarding privacy. `http://online.wsj.com/news/articles/SB10001424052970204880404577225380456599176`, February 2012.
8. Mika Ayenson, Dietrich J. Wambach, Ashkan Soltani, Nathan Good, and Chris J/ Hoofnagle. Flash cookies and privacy II: Now with HTML5 and ETag respawning. *World Wide Web Internet And Web Information Systems*, 2011.

9. Mahesh Balakrishnan, Iqbal Mohomed, and Venugopalan Ramasubramanian. Where's that phone?: geolocating IP addresses on 3G networks. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 294–300. ACM, 2009.
10. Rebecca Balebako, Pedro Leon, Richard Shay, Blase Ur, Yang Wang, and L/ Cranor. Measuring the effectiveness of privacy tools for limiting behavioral advertising. In *Web 2.0 Workshop on Security and Privacy*, 2012.
11. Paul E. Black. Ratcliff/Obershelp pattern recognition. `http://xlinux.nist.gov/dads/HTML/ratcliffObershelp.html`, December 2004.
12. Elie Bursztein. Tracking users that block cookies with a HTTP redirect. `http://www.elie.net/blog/security/tracking-users-that-block-cookies-with-a-http-redirect`, 2011.
13. Christian Eubank, Marcela Melara, Diego Perez-Botero, and Arvind Narayanan. Shining the floodlights on mobile web tracking - a privacy survey. Web.
14. Sharad Goel, Jake M. Hofman, and M. Irmak Sirer. Who does what on the web: A large-scale study of browsing behavior. In *ICWSM*, 2012.
15. Manoj Hastak and Mary J. Culnan. Persistent and unblockable cookies using HTTP headers. `http://www.nikcub.com/posts/persistant-and-unblockable-cookies-using-http-headers`, 2011.
16. Bin Liu, Anmol Sheth, Udi Weinsberg, Jaideep Chandrashekar, and Ramesh Govindan. AdReveal: improving transparency into online targeted advertising. In *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*, page 12. ACM, 2013.
17. Jonathan R. Mayer and John C. Mitchell. Third-party web tracking: Policy and technology. In *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 413–427. IEEE, 2012.
18. Aleecia M McDonald and Lorrie Faith Cranor. Survey of the use of Adobe Flash local shared objects to respawn HTTP cookies. *ISJLP*, 7:639, 2011.
19. Jakub Mikians, László Gyarmati, Vijay Erramilli, and Nikolaos Laoutaris. Detecting price and search discrimination on the internet. In *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*, pages 79–84. ACM, 2012.
20. Nick Nikiforakis, Alexandros Kapravelos, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. In *Security and Privacy (SP), 2013 IEEE Symposium on*, pages 541–555. IEEE, 2013.
21. Mike Perry, Erinn Clark, and Steven Murdoch. The design and implementation of the Tor browser [draft]. `https://www.torproject.org/projects/torbrowser/design`, March 2013.
22. Ashkan Soltani, Shannon Canty, Quentin Mayo, Lauren Thomas, and Chris Jay Hoofnagle. Flash cookies and privacy. In *AAAI Spring Symposium: Intelligent Information Privacy Management*, 2010.
23. Ashkan Soltani, Andrea Peterson, and Barton Gellman. NSA uses Google cookies to pinpoint targets for hacking. `http://www.washingtonpost.com/blogs/the-switch/wp/2013/12/10/nsa-uses-google-cookies-to-pinpoint-targets-for-hacking`, December 2013.
24. Jennifer Valentino-Devries, Jeremy Singer-Vine, and Ashkan Soltani. Websites vary prices, deals based on users' information. `http://online.wsj.com/news/articles/SB10001424127887323777204578189391813881534`, December 2012.

25. Jennifer Valentino-Devries, Jeremy Singer-Vine, and Ashkan Soltani. What they know. `http://online.wsj.com/public/page/what-they-know-digital-privacy.html`, 2012.
26. Craig E. Wills and Can Tatar. Understanding what they do with what they know. In *Proceedings of the 2012 ACM Workshop on Privacy in the Electronic Society*, pages 13–18. ACM, 2012.
27. Michal Zalewski. Rapid history extraction through non-destructive cache timing (v8). `http://lcamtuf.coredump.cx/cachetime/`. Accessed: 2014.

A Appendices

A.1 Modeling cookie expiration time

We limited our analysis to cookies with a three-month-plus lifespan as an efficient way to ensure that tracking cookies survived for the duration of the user's browsing. But to more accurately model cookie expiration and its effect on the growth of the GCC, we use the AOL dataset to map the dates in which we collected data to the original timestamps spanning the actual three month's worth of search queries from 2006. We consider the timestamps of the original visits as if we were visiting those pages in "real-time," and from there attempt to model the effect of the cycle of a cookie being set and reset on the growth of the giant connected component.
When we first encounter a cookie under this model, we use the lifespan of the cookie to determine how far in the future the cookie will persist. Any web page having this cookie that is encountered before the end of the cookie's lifespan will be connected to other sites having that same cookie in the same lifespan. If eventually the simulated time of the crawl progresses past the end of the cookie's life and the same cookie is encountered again, we must presume that the unique identifier in the cookie would be reset, preventing us from connecting this new cookie from past instances of the same cookie.

A.2 Example of identity leak

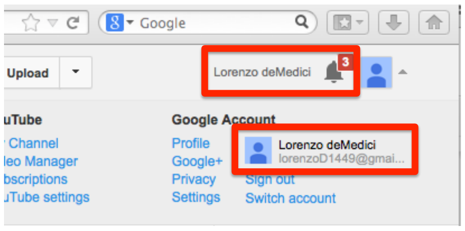
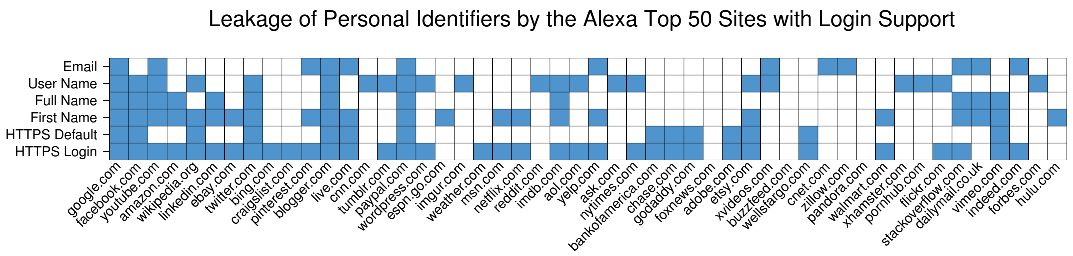


Fig. 7. Identity leak of a pseudonymous account on an unencrypted YouTube page

A.3 Full Summary of Personal Identifier Leaks


Leakage of Personal Identifiers by the Alexa Top 50 Sites with Login Support
Email
User Name
Full Name
First Name
HTTPS Default
HTTPS Login
google.com
facebook.com
youtube.com
amazon.com
wikipedia.org
linkedin.com
ebay.com
twitter.com
bing.com
craigslist.com
pinterest.com
blogger.com
live.com
cnn.com
tumblr.com
paypal.com
wordpress.com
espn.go.com
imgur.com
weather.com
msn.com
netflix.com
reddit.com
imdb.com
aol.com
yelp.com
ask.com
nytimes.com
bankofamerica.com
chase.com
godaddy.com
foxnews.com
adobe.com
etsy.com
xvideos.com
buzzfeed.com
wellsfargo.com
cnet.com
zillow.com
pandora.com
walmart.com
xhamster.com
pornhub.com
flickr.com
stackoverflow.com
dailymail.co.uk
vimeo.com
indeed.com
forbes.com
hulu.com