HTTP Cookie Hijacking in the Wild: Security and Privacy Implications

Suphannee Sivakorn, Jason Polakis* and Angelos D. Keromytis*
[suphannee, polakis, angelos]@cs.columbia.edu
Columbia University, New York NY, USA
**Joint primary authors*

The widespread demand for online privacy, also fueled by widely-publicized demonstrations of session hijacking attacks against popular websites, has spearheaded the increasing deployment of HTTPS. However, many websites still avoid ubiquitous encryption due to performance or compatibility issues. The prevailing approach in these cases is to force critical functionality and sensitive data access over encrypted connections, while allowing more innocuous functionality to be accessed over HTTP. In practice, this approach is prone to flaws that can expose sensitive information or functionality to third parties.

In this work, we conduct an in-depth assessment of a diverse set of major websites and explore what functionality and information is exposed to attackers that have hijacked a user's HTTP cookies. We identify a recurring pattern across websites with partially deployed HTTPS; service personalization inadvertently results in the exposure of private information. The separation of functionality across multiple cookies with different scopes and inter-dependencies further complicates matters, as imprecise access control renders restricted account functionality accessible to non-session cookies. Our cookie hijacking study reveals a number of severe flaws; attackers can obtain the user's home and work address and visited websites from Google, Bing and Baidu expose the user's complete search history, and Yahoo allows attackers to extract the contact list and send emails from the user's account. Furthermore, e-commerce vendors such as Amazon and Ebay expose the user's purchase history (partial and full respectively), and almost every website exposes the user's name and email address. Ad networks like Doubleclick can also reveal pages the user has visited. To fully evaluate the practicality and extent of cookie hijacking, we explore multiple aspects of the online ecosystem, including mobile apps, browser security mechanisms, extensions and search bars. To estimate the extent of the threat, we run IRB-approved measurements on a subset of our university's public wireless network for 30 days, and detect over 282K accounts exposing the cookies required for our hijacking attacks. We also explore how users can protect themselves and find that, while mechanisms such as the EFF's HTTPS Everywhere extension can reduce the attack surface, HTTP cookies are still regularly exposed. The privacy implications of these attacks become even more alarming when considering how they can be used to deanonymize Tor users. Our measurements suggest that a significant portion of Tor users may currently be vulnerable to cookie hijacking.

1 Introduction

In the past few years, there has been much discussion within the research community regarding the necessity of securing web connections from adveraries. Firesheep [1] demonstrated how easily attackers can hijack a user's session. Many major services moved to critical user activities to mandatory HTTPS connections. Nonetheless many services still serve content over unencrypted connections, which exposes the users' HTTP cookies to attackers mornitoring their traffic.

Not enforcing ubiquitous encrypted connections may be attributed to various reasons, ranging from potential increases to infrastructure costs and the loss of in-network functionality [2] to maintaining support for legacy clients. If access control policies correctly separated privileges of authenticated (e.g., session cookies) and non-authenticated cookies (e.g., persistent tracking cookies), stolen HTTP cookies would not allow attackers to obtain any personal user information. However, that is not the case in practice [3]. The problem can be partly attributed to the things become worse as services continue to sacrifice security over usability.

In this paper, we explore the extent and severity of the unsafe practice followed by major services of partially adopting encrypted connections, and its ramifications for user privacy. We demonstrate how HTTP *cookie hijacking*

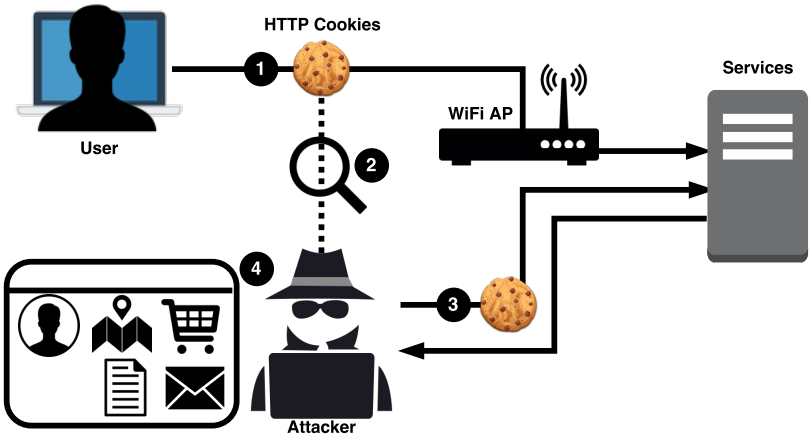


Figure 1: HTTP cookie hijacking attack

attacks not only enable access to private and sensitive user information, but can also circumvent authentication requirements and gain access to protected account functionality.

2 Background

In this section we provide a background of cookie hijacking in practice, and current security mechanisms in use that can mitigate cookie hijacking attacks.

2.1 HTTP Cookie Hijacking

The adversary monitors the traffic (e.g., in a public wireless network, proxy, middlebox). Figure 1 presents the workflow of a cookie hijacking attack. The attack starts when the user's browser appends the HTTP cookies to the HTTP requests sent over unencrypted connections (1). As the traffic is being monitored by the attacker, the unencrypted cookies can be easily extracted from the network trace (2). The adversary then connects to the vulnerable services using the stolen cookies from the trace (3). The services "identify" the user from the cookies and offer a personalized version of the website, thus, exposing the user's personal information and account functionality to the adversary (4).

The cookie hijacking attacks require the user to have previously logged into the service as the attack requires the presence of cookies from the user. The adversary can store the cookies and continuously obtain up-to-date information until the cookies' expiration date.

Active Adversary. The attacker can also opt for more active approaches and take advantage of WiFi Portals [4] and compromise access points [5], to enable more invasive attacks. For example, injecting content that forces the user's browser to send requests to specific vulnerable websites and expose the user's cookies even if the user does not explicitly visit those sites while being monitored.

2.2 Current Security Mechanisms

Browsers include support for various security mechanisms that are designed to protect user from a range of attacks that target communications to online services. Here we explore mechanisms that are relevant to our attack.

2.2.1 HTTPS and Secure Cookie

Because HTTP traffic is unencrypted, any data sent over HTTP can be read and modified by anyone with access to the network. HTTPS relies on SSL/TLS for encrypting the data, and is often combined with cookies that are

flagged as secure. HTTP cookies are used to identify users, and provide access to a more personalized version of a website, without requiring the user to log in. The secure flag is an option that is set by the server when sending a new cookie to a user in an HTTP response. The secure flag is to prevent cookies from being sent over unencrypted connections. Both HTTPS and cookies with the secure flag prevent the transmission of unencrypted HTTP data. Listing 1 shows an example `Set-Cookie` HTTP header response of a non-secure cookie and a cookie with the `secure` flag.

Listing 1: Example of header for setting non-secure and secure cookies.

```
Set-Cookie: SID=XXXXXXXXXX; Expires=Mon, 01 Jan 1970 00:00:01 GMT; Path=/;
            Domain=.google.com
Set-Cookie: SSID=YYYYYYYYYY; Expires=Mon, 01 Jan 1970 00:00:01 GMT; Path=/;
            Domain=.google.com; secure; HttpOnly
```

2.2.2 HSTS

The HTTP Strict Transport Security mechanism (HSTS) [6] allows websites to instruct browsers to only communicate over HTTPS. This is done through the `Strict-Transport-Security` HTTP header response. HSTS is currently supported in all major browsers and certain mobile browsers [7].

HSTS Preload. Due to the fact that the cookie is appended in the user's initial request, before the HSTS header is received, the user is exposed to hijacking if the initial connection is over HTTP. HSTS Preload is designed to prevent such an attack. Major browsers rely on the *preload list* which is a list of hardcoded domains, which are always contacted over HTTPS, including the initial request. The list is currently maintained by Chrome and shared with other major browsers (i.e. Firefox, Safari, Opera, IE 11 and Edge) [8]. Certain browsers, like Firefox, create custom versions of the preload list, based on Chrome's version.

Certificate Pinning. In addition, HSTS Preload also incorporates certificate pinning, which was designed to prevent attacks that use rogue SSL certificates. Browsers will only trust specific certificates and ignore certificates that are signed by other parties, as adversaries can create or obtain fraudulent certificates (e.g., in the event of a CA compromise) and impersonate websites as part of man-in-the-middle attacks [9].

2.2.3 HTTPS Everywhere

HTTPS Everywhere is a browser extension from EFF and the Tor Project that was released in 2010 [10]. The extension is designed to enforce browsers to always connect to domains over HTTPS, based on a set of rulesets written by the community. The rulesets in HTTPS Everywhere are a set of regular expressions that aim to rewrite any URLs from HTTP to HTTPS in order to force connection to be encrypted based on the rule indicated. HTTPS Everywhere currently has 1.7 Million users (May 2016) and can be installed in most major browsers: Chrome, Firefox, Firefox for Android and Opera. The extension is pre-installed in the Tor Browser.

3 Real-World Privacy Leakage

In this section, we present the major findings from our study on HTTP cookie hijacking attacks in real websites. For a more complete list, we refer readers to [11].

First we explore browser behavior that results in the exposure of cookies over unencrypted connections, for websites that support HTTPS. We then audit numerous top Alexa websites [12] and find that HTTP cookie hijacking attacks affect the majority of popular websites we tested.

3.1 Browser Behavior

For HTTP cookie hijacking attack to work, the adversary must observe an unencrypted connection to the server. However, if the website is running on HTTPS, the attacker should not observe any HTTP request. A very typical scenario is for the HTTP cookie to be sent over HTTP is when the victim uses the browser's address bar (Figure 2). The flow starts with the user typing a URL (`www.google.com`) in the address bar(1). The browser by default will

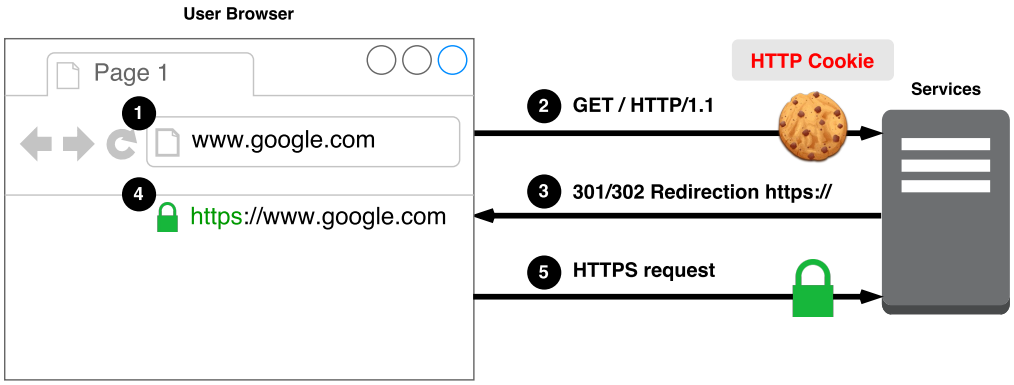


Figure 2: HTTP cookie sent unencrypted with HTTP request before redirect to HTTPS.

Table 1: Browser behavior for user input in address bar.

Browser	**Connect over** HTTP
Desktop	
Chrome (v. 45)	✓
Firefox (v. 41)	✓
Safari (v. 8.0)	✓
Internet Explorer (v. 11)	✓
Opera (v. 32)	✓
Mobile	
Safari (iOS 9)	✓
Chrome (v.46, Android 5.1.1)	✗ (conditionally)

***user input:** {google.com, www.google.com}

send an HTTP request for the given URL(2). As this request is over HTTP, the user's HTTP cookies are appended to this request. Since the server supports HTTPS, it sends an HTTP redirection to its HTTPS page (3). The user's browser will receive the response from the server and automatically change `http://` to `https://` in the address bar(4). After that, the browser completes the SSL/TLS handshake and can communicate securely. This process seems to be very secure to users as the server and browser co-operatively redirect the user to a secure connection. However, step 2 leaves a window of opportunity for attackers to steal the cookies. This also means that even when users see `https://` in the address bar and the other visual clues of a trusted connection, they might still have been exposed to a cookie hijacking attack during the initial request.

To understand the conditions under which this occurs, we explore how popular browsers handle user input in the address bar, when trying to visit `google.com`. As shown in table 1, for straightforward user input, popular browsers will connect to `google.com` over HTTPS. Due to the autocomplete feature of certain browsers (e.g., Firefox), even if the victim only types `"google"`, the auto-complete mechanism will add `".com"`, and the browser will again connect over HTTP. Therefore, under common browsing patterns, the existing design will expose a user's cookie. Interestingly, while the default iOS browser (Safari) exhibits the same behavior, Chrome on Android will connect to Google over HTTPS to securely prefetch page resources. However, if users turn this option off to improve performance[1], Android Chrome will also connect over HTTP.

3.2 Audited Services and Results

We audit the top Alexa website from a varied collection of categories using test accounts and find that HTTP cookie hijacking attacks affect the majority of popular websites that do not enforce ubiquitous encryption. Table 2 presents an overview of the services and our results, including the private information and account functionality that we are able to access with the stolen cookies.

[1] `https://support.google.com/chrome/answer/1385029`

Table 2: Overview of the audited websites and services, the feasibility of cookie hijacking attacks, and the type of user information and account functionality they expose.

Service	HTTPS Adoption	Cookie Hijacking	XSS Cookie Hijacking	Information and Account Functionality Exposed
Google	partial	✓	✗	first and last name, username, email address, profile picture, home and work address, search optimization, click history of websites returned in search results
Baidu	partial	✓	✓	username, email address, profile picture, entire search history, address of any saved location
Bing	partial	✓	✓	first name, profile photo, view/edit search history (incl. images and videos), links clicked from search results, frequent search terms, saved locations, information in interest manager, edit interest manager
Yahoo	partial	✓	✓	username, full name, email address, view/edit search history, view/edit/post answers and questions in Yahoo Answers (anonymous or eponymous), view/edit finance portfolio, view subject and sender of latest incoming emails, extract contact list and send email as user
Youtube	partial	✓	✗	view and change (through pollution attacks) recommended videos and channels
Amazon	partial	✓	✓	view user credentials (username, email address or mobile number), view/edit profile picture, view recommended items, view user wish lists, view recently browsed items, view recently bought items, view/edit items in cart, view shipping name and city, view current balance, view user's review (even anonymous), send email of products or wishlist on behalf of user, obtain email addresses of previously emailed contacts
Ebay	partial	✓	✓	delivery name and address, view/edit items in cart, view/edit purchase history, view items for sale, view previous bids, view user's messages, view/edit watch list and wish lists
MSN	partial	✓	✓	first and last name, email address, profile picture
Walmart	partial	✓	✓	first name, email address, view/edit items in cart, view delivery postcode, write product review
Target	partial	✓	✓	first name, email address, view/edit items in cart, recently viewed items, view and modify wish list, send email about products or wish list
CNN	partial	✓	✓	view/edit profile (full name, postal address, email address, phone number, profile picture) view/edit linked Facebook account, write/delete article comments, recently viewed content on iReport
New York Times	partial	✓	✓	username, email address, view/edit basic profile (display name, location, personal website, bio, profile picture) username, email address, view/edit list of saved articles, share article via email on behalf of user
Huffington Post	partial	✓	partial	profile can be viewed and edited (login name, profile photo, email address, biography, postal code, location, subscriptions, fans, comments and followings). change account password, delete account
The Guardian	partial	✓	✓	username, view public section of profile (profile picture, bio, interests), user's comments, replies, tags and categories of viewed articles, post comments on articles as user
Doubleclick	partial	✓	✓	ads show content targeted to user's profile characteristics or recently viewed content
Skype	partial*	✗	✗	-
LinkedIn	partial*	✗	✗	-
Craigslist	partial*	✗	✗	-
Chase Bank	partial*	✗	✗	-
Bank of America	partial*	✗	✗	-
Facebook	full	✗	✗	N/A
Twitter	full	✗	✗	N/A
Google+	full	✗	✗	N/A
Live (Hotmail)	full	✗	✗	N/A
Gmail	full	✗	✗	N/A
Paypal	full	✗	✗	N/A

*While these services do not have ubiquitous HTTPS, no personalization is offered over HTTP pages.

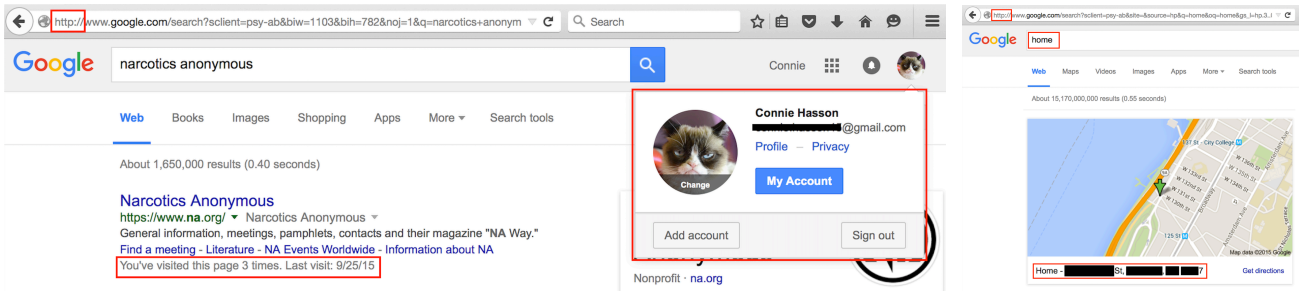
3.3 Google

3.3.1 Google Services

Due to the cookie, Google considers the victim logged-in, resulting in personal information being leaked.

Personal information. As can be seen in Figure 3(a), we gain access to the user's name and surname, Gmail address, and profile picture.

Location. Google Maps allows users to set their Home and Work addresses, for easily obtaining directions to/from other destinations. While Google Maps requires HTTPS, which prevents us from acquiring any information, if the adversary connects to Google over HTTP and searches for "`home`" or "`work`", the search results will contain a widget of Google Maps revealing the respective address. An example can be seen in Figure 3(b). Accessibility to location information can expose the user to physical threats [13, 14].



(a) Profile and History

(b) Location

Figure 3: Private information obtainable from user's Google account through HTTP cookie hijacking.

Browsing history. Using the stolen cookie, the adversary can start issuing Google searches for various terms of interest. If the search results contain links that the user has previously visited through the search engine, Google will reveal how many times the page has been visited and the date of the last visit. Users can opt-out of historical information being included in their search results, however, this option is enabled by default. If enabled, the adversary can search for a variety of terms and infer sensitive data about the user. Figure 3(a) shows an example scenario where the adversary obtains such information. Depending on the attacker's goal, she could employ a precompiled dictionary of sensitive keywords for finding sensitive web activity, or a dictionary of the most popular Google search terms for recovering parts of the user's web visiting history. While previous work demonstrated that unencrypted sessions could enable attackers to reconstruct a user's Google search history [15], this is the first, to our knowledge, attack that discovers webpages *visited* by the user through Google.

Exploiting search optimization. Google search may return results that have been personalized for the user, either by inserting specific entries, or changing the rank of specific results. Previous work has demonstrated a methodology for measuring personalization in Google search results [16]. By adapting this technique, the adversary can extract entries from the search results that have been returned based on characteristics of the victim's profile.

Shopping. Using the HTTP Google cookie when visiting Google's shopping page, which runs mainly over HTTP, will reveal the user's first and last name, Gmail handle, Google profile. It also allows viewing and editing the user's shortlist (items under consideration).

Pollution attacks. If the attacker issues search queries using the stolen cookies, the search terms are treated as if originating from the user and added to the search history. This allows the adversary to affect the victim's contextual and persistent search personalization through pollution attacks [17].

Solving reCAPTCHA. The stolen Google cookies can also be used to bypass reCAPTCHA challenges [18], as the user's browsing history will most likely result in high confidence by the advanced risk analysis system.

3.3.2 Youtube

Youtube exhibits a strange behavior that we did not come across in other services. If the victim is logged in, the stolen cookie does not reveal any information. However, if the victim is not logged in, the cookie that is exposed gives access to the user's recommended channels and videos, which can be changed through pollution attacks. Furthermore, information about the user's music interests can be used to infer private attributes [19].

3.4 Bing

According to a recent report [20], Bing handles approximately 20.4% of the desktop searches originating from the U.S. Bing is also the default search engine for Siri and all Microsoft-based products. When auditing Bing we found that, by default, all connections are served over HTTP, i.e., all searches are sent in clear-text. Users have to explicitly type `https` in the browser's address bar to be protected from eavesdropping.

Personal information. Bing will expose the user's first name and profile photo. The profile photo can be used to obtain more information of the user through face recognition and publicly available data in other websites [21].

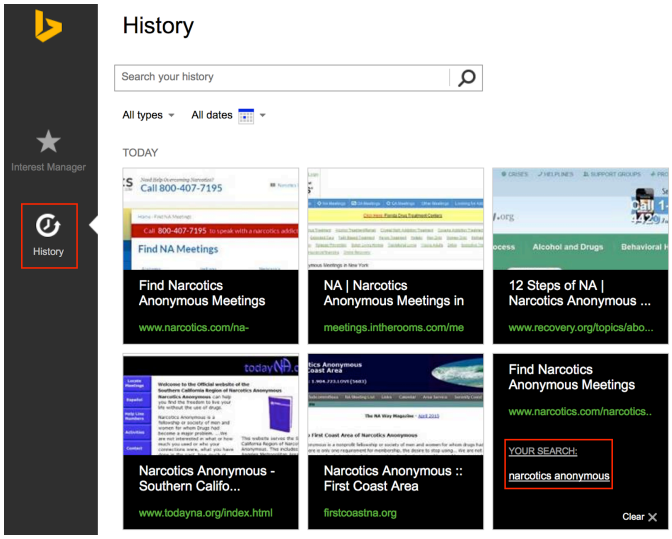


Figure 4: Bing's search and visit history seen by attacker.

Location. If the victim has saved any locations on Bing Maps they are also exposed. Apart from the work or home addresses, this may include other locations the user has visited in the past (e.g., bars, health clinics).

Interest Manager. This recently introduced feature, allows users to select interests from a variety of topics. Based on the category, this can reveal private information including financial assets and political inclination.

Search and browsing history. Once the adversary steals the cookie, she can retrieve the user's search history, including those in the images and videos categories. Apart from a widget displaying the users most recent and most frequent search queries, the search history page also reveals the page that the user visited from each search (Figure 4).

Pollution attacks. The attacker can also issue search queries for conducting a pollution attack and, subsequently, delete those entries for stealthiness. This will remove any trails of the attack, and prevent the victim from detecting it.

3.5 Yahoo

Links in the main Yahoo page are all redirected through `http://hsrd.yahoo.com`. Even if the user explicitly connects to Yahoo over HTTPS, if any link in the homepage is clicked, it will connect to that subdomain over an unencrypted connection. Therefore, regardless of how the victim connects, we have identified three HTTP cookies (Y, F, T) that are exposed to eavesdroppers and enable access to sensitive information and functionality. We first describe the information and functionality that attackers can access, and then how we perform a cookie forging attack to remove the requirements for the user to browse specific subdomains while being monitored.

Yahoo currently moved their site to mainly served on HTTPS. However we are still able to obtain the HTTP cookies from browser redirection behavior and able to expose user's information and functionalities.

Personal information. The Y and T cookies set for `yahoo.com` allow the attacker to obtain the user's first name. The full last name and email address can also be obtained, as we explain below.

Yahoo Mail. To facilitate sharing posts with friends, articles in Yahoo contain an "Email to friends" button, which presents a popup window in which the adversary can add an arbitrary message, as shown in Figure 5. Furthermore, the `Sender` field has auto-complete functionality, which allows us to obtain the victim's complete contact list. These features combined can be leveraged for deploying effective phishing or spam campaigns. The contacts' emails can be used for acquiring information about those users from other services and deploying personalized spam campaigns [22]. The widget also contains the user's full name and email address. Extracting the contacts requires all three cookies set for the main domain, while sending the email requires them for the `news.yahoo.com` or the `finance.yahoo.com` subdomain depending on which section the article is located in.

If the user hovers over or clicks on the mail notification button, the attacker can also access the incoming mail

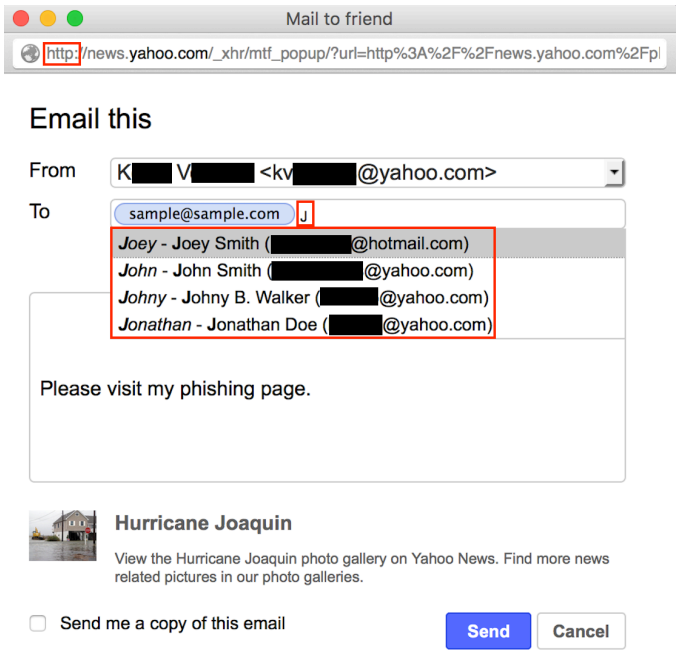


Figure 5: Extracting contact list and sending email from the victim's account in Yahoo.

widget, which reveals the `Sender` and partial `Subject` (up to 21 characters) of the 8 most recent incoming emails. This is due to a cookie being attributed an "authenticated" status. This lasts approximately one hour, after which it cannot access the widget. If at any point the user accesses the notification button again, the hijacked cookie is re-authorized.

Yahoo Search. Having acquired the main domain and search subdomain Y and T cookies, the adversary can gain access to the victim's complete search history. Apart from viewing the searched terms, these cookies allow editing the history and removing previous searches. However, Yahoo explicitly states that even if past searches are deleted, user search data is still logged. This enables *stealthy* pollution attacks; after issuing search queries for influencing the personalization profile of the user, the adversary can then delete all issued searches and remove traces of the attack.

Yahoo Answers. One of the many services offered by Yahoo, is a popular "question and answer" site, where users can ask any type of question, and other members of the community can provide answers (albeit sometimes with questionable quality [23]). Users posting questions or answers, may choose to remain anonymous for a given question, especially if the topic is considered sensitive [24]. Upon auditing Yahoo, we found that the victim's HTTP cookie allows partial control over the account; the adversary is able to ask or answer questions (either eponymously or anonymously), and also to view and edit previous questions and answers posted by the victim. Thus, the adversary can effectively "deanonymize" posts and obtain potentially sensitive information about the victim, which was posted under the assumption of anonymity. The adversary can also post comments as the victim in the comment section of news articles. This requires the Y, T cookies for the `yahoo.com` domain and the `answers.yahoo.com` subdomain.

3.6 Amazon

The homepage follows the common approach of redirecting to HTTPS if connected to over HTTP. However, product pages are served over HTTP and, as a result, users' cookies will be exposed during their browsing sessions.

Personal Information. The adversary can obtain the information used by the victim for logging in; this includes the victim's username, email address and/or cell phone number. Furthermore, when proceeding to checkout items in the cart, Amazon also reveals the user's full name and city (used for shipping). The adversary can view the user's reviews (which may include sensitive items), thus, breaking the pseudonymous nature of those reviews.

Previous work has demonstrated the privacy risks of recommender systems and experiments in Amazon indicated that sensitive purchases can be inferred from the user's review history [25].

Account History. The user's HTTP cookie is sufficient for accessing private information regarding previous actions. Specifically, the adversary can obtain information regarding recently viewed items, and recommendations that are based on the user's browsing and purchase history. The wish-lists where the user has added items of interest are also accessible. Furthermore, the adversary can obtain information regarding previously purchased items either through the recommendation page or through product pages (which depict date of purchase). In an extensive study on privacy-related aspects of online purchasing behavior [26], users rated the creation of a detailed profile from their purchase history and other personal information as one of the most troubling scenarios.

Shopping Cart. The user's cart is also accessible, and the adversary can see the items currently in the user's cart. Additionally, the cart can be modified, and existing items can be removed, and other items can be added.

Vendor-assisted spam. We also found that the cookie exposes functionality that can be leveraged for deploying spam campaigns to promote specific items that are presented as "endorsed" by the victim. The widget has an auto-complete feature that reveals the contacts that the user has emailed in the past. The attacker can either send emails about a specific item or a wish-list, and can add text in the email's body. URLs can be included; while the email is sent as simple text, email providers such as Gmail render it as a click-able link. Since the emails are actually sent by Amazon (`no-reply@amazon.com`), they are most likely to pass any spam detection heuristics. Furthermore, the `From` field, contains the victim's username, further strengthening the personalized nature of the spam/phishing email.

3.7 Indirect Information Exposure - Ad Networks

Online ads account for a significant portion of website real estate, and their ubiquitous nature has been discussed extensively in the context of user tracking (e.g., [27, 28]). Here we focus on Doubleclick, which is owned by Google. An interesting aspect of hijacking ad-network cookies is that they result in *side-channel information leakage*.

3.7.1 Information leakage.

We conducted a small number of manual experiments for identifying cases of personal information being leaked by Doubleclick. Previous work has shown that ads presented to users may be personalized based on their browsing history and interests, which may be based on the user's profile characteristics [29], associated to sensitive topics [30, 31] (e.g., substance abuse, health issues, sexual inclination), and that advertisers can even obtain private user information not explicitly provided by the service [32].

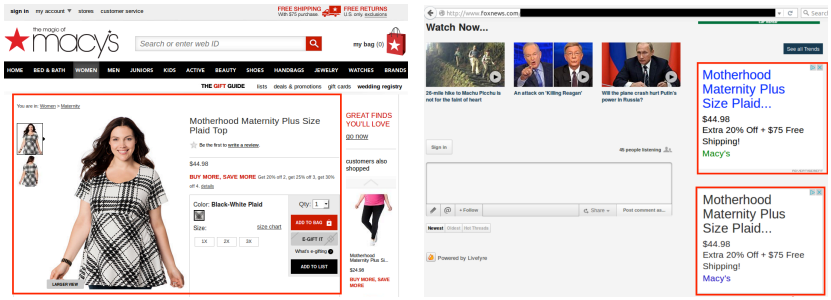
Here we describe one of our experiments for indirect information exposure. We browsed maternity clothes on a popular e-commerce website, and visited the page of a few returned products (figure 6(a)). We, then browsed other sites from a different machine connected to a different subnet, and appended the Doubleclick HTTP cookie from the previous browsing session. We were presented with ads from the e-commerce website advertising women's clothing. Several ads even advertised a specific maternity product whose page we had visited (figure 6(b)). Depending on the time lapsed between the user browsing the e-commerce site and the attacker browsing with hijacked cookies, there is a decrease in the frequency of ads that contain the viewed product. However, we found that even after several hours we received ads that continued to promote the exact product, and women's clothing ads even after several days.

3.8 Collateral Cookie Exposure

In this section we explore other means by which a user's HTTP cookies may be exposed.

3.8.1 Browser Extensions and Apps

According to a manifest file analysis of over 30K Chrome extensions [33], a higher number of extensions requested permission for connecting to Google over HTTP compared to HTTPS. The same was true for wildcarded (`http://*/*`) permission requests. This indicates that a considerable number of extensions may be weakening security by connecting over unencrypted connections to websites that also support encrypted connections. To that



(a) Browsed page

(b) Attacker receives ads exposing browsed page.

Figure 6: Side-channel leak of user's browsing history by the Doubleclick ad network.

Table 3: Cookie exposure by popular browser extensions and apps.

Name	Type	Browser	#	Cookie leaked
Google Maps	app	Chrome	N/A	✓
Google Search	app	Chrome	N/A	✓
Google News	app	Chrome	1.0M	✓
Amazon Assistant	extension	Chrome	1.1M	✓
Bing Rewards	extension	Chrome	74K	✓
eBay for Chrome	extension	Chrome	325K	✓
Google Dictionary	extension	Chrome	2.7M	✓
Google Hangouts	extension	Chrome	6.4M	✗
Google Image Search	extension	Chrome	1.0M	✗
Google Mail Checker	extension	Chrome	4.2M	✗
Google Translate	extension	Chrome	5.5M	✗
Yahoo Mail Notification	extension	Chrome	1.2M	✗
Amazon	default search bar	Firefox	N/A	✓
Bing	default search bar	Firefox	N/A	✗
Ebay	default search bar	Firefox	N/A	✓
Google	default search bar	Firefox	N/A	✗
Yahoo	default search bar	Firefox	N/A	✗
Amazon 1Button	extension	Firefox	157K	✓
Bing Search	extension (unofficial)	Firefox	28K	✓
eBay Sidebar	extension	Firefox	36K	✓
Google Image Search	extension	Firefox	48K	✓
Google Translator	extension (unofficial)	Firefox	794K	✓
Yahoo Toolbar	extension	Firefox	31K	✓

end, we explore whether browser components expose users to cookie hijacking attacks. We analyze a selection of the most popular browser components, for Chrome and Firefox, that have been released by major vendors we have audited.

Table 3 lists the web components we have evaluated, their reported number of downloads if available, and if they leak the cookies required for our hijacking attacks. Our experiments yield a number of surprising findings. The 3 Chrome apps released by Google we tested expose the HTTP cookies, while their extensions present mixed results with 4 out of 9 leaking the cookie. As one of those is Google Dictionary, with over 2.7 million downloads, a significant number of Chrome users is vulnerable to considerable risk.

Every Firefox extension we tested, along with two of the default search bars, actually expose the required HTTP cookies over unencrypted connections. Interestingly, Google's Search by Image extension is secure for Chrome but not for Firefox. As there is no official Bing app for Firefox, we test the most popular one, and we also audit a popular unofficial Google translator extension with over 794K users, both of which turn out to be vulnerable. Overall, these findings highlight the privacy threats that millions of users face due to browser components.

3.8.2 Mobile Apps

To explore the feasibility of our HTTP cookie hijacking attacks against users on mobile devices, we audited the official iOS and Android apps for the most popular services that we found to expose private information and account

Table 4: Cookie exposure by official mobile apps.

Application	Platform	Version	#	Cookie leaked
Amazon	iOS	5.3.2	N/A	✗
Amazon	iOS	5.2.1	N/A	✓
Amazon	Android	28.10.15	10-50M	✗
Bing Search	iOS	5.7	N/A	✓
Bing Search	Android	5.5.25151078	1-5M	✓
Spotlight (Bing)	iOS	iOS9.1	N/A	conditionally
Siri (Bing)	iOS	iOS9.1	N/A	✗
Ebay	iOS	4.1.0	N/A	conditionally
Ebay	Android	4.1.0.22	100-500M	conditionally
Google	iOS	9.0	N/A	✗
Google	Android	5.4.28.19	1B+	✗
Gmail	iOS	4.1	N/A	✗
Gmail	Android	5.6.103338659	1-5B	✗
Google Search Bar	Android	5.4.28.19	N/A	✗
Yahoo Mail	iOS	4.0.0	N/A	conditionally
Yahoo Mail	Android	4.9.2	100-500M	✗
Yahoo News	iOS	6.3.0	N/A	✓
Yahoo News	Android	18.10.15	10-50M	✗
Yahoo Search	iOS	4.0.2	N/A	✗
Yahoo Search	Android	4.0.2	1-5M	✗
Yahoo Sports	iOS	5.7.4	N/A	✓
Yahoo Sports	Android	5.6.3	5-10M	✗

Table 5: Statistics of outgoing connections from a subset of our campus' public wireless network for 30 days.

Protocol	Connections	Requests	Vulnerable Requests*	Exposed Accounts
HTTP	685,500,365	1,398,044,178	29,908,099	282,459
HTTPS	772,562,024	–	–	–

*HTTP requests to domains that we have audited and found to be vulnerable.

functionality. The overview of our results is shown in Table 4. Spotlight, the system-wide search feature of iOS, is also powered by Bing. When the user issues a search query, Spotlight connects over HTTPS to Apple servers. However, the search results contain a "Show more in Bing" button and, if clicked, will open the browser showing the search results and leak the user's HTTP Bing cookie. Once again Yahoo follows poor security practices as 3 out of 4 iOS apps leak the user's cookies. As expected both versions of Gmail protect the cookies, while iOS Amazon apps prior to version 5.3.2 expose the cookie. First, Ebay sellers are allowed to customize their item pages and often add links to other items they are selling; if the seller has added an HTTP Ebay link to those items, the cookie will be exposed if a link is clicked by the user. Empirically we found that that these HTTP links are common. The other scenario is if the user clicks on the "Customer Support" menu.

4 Evaluation

4.1 Public WiFi Data Collection

The feasibility of cookie hijacking attacks through eavesdropping depends on the browsing behavior of users when connected to public wireless networks. If users only visit websites with ubiquitous encryption or employ VPN tunneling solutions, HTTP cookie hijacking can be prevented. Our goal is to understand the browsing patterns of users connecting to public wireless networks, and measure the feasibility of explore the potential impact of cookie hijacking attacks in practice. We conduct an exploratory study of the traffic passing through the public wireless network of our university's campus. Before beginning our experiments, we got approval from our Institutional Review Board. We worked closely with the Network Security team of our university's IT department for conducting the data collection and analysis in a secure and privacy-preserving manner.

Data collection. In order to collect the data, we setup a logging module on a network tap that received traffic

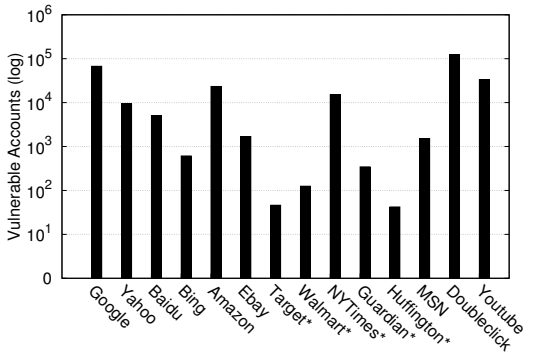


Figure 7: Number of unique vulnerable accounts per service.

from multiple wireless access points positioned across our campus. The RSPAN was filtered to only forward outgoing traffic destined to TCP ports 80 and 443, and had a throughput of 40-50 Mb/s, covering approximately 15% of the public wireless outgoing traffic. Our data collection lasted for 30 days. We used the number of TCP SYN packets to calculate the number of connections. When the connection is over HTTP or HTTPS, we capture the destination domain name through the HTTP host header and the TLS SNI extension respectively. For each HTTP request we log the destination domain, and the name of any HTTP cookies appended (e.g., SID). We also calculated a HMAC of the cookie's `value` (the random key was discarded after data collection). The cookie names allow us to verify that users are logged in and susceptible to cookie hijacking for each service, as we have explored the role of each cookie and also identified the subset required for the complete attack.

While we do not log the cookie value for privacy reasons, the keyed hash value allows us to distinguish the same user within a service to obtain a more accurate estimation of the number of exposed accounts. We must note that our approach has limitations, as the numbers we estimate may be higher than the actual numbers; a user's cookie value may have changed over the course of the monitoring period or the user may use multiple devices (e.g., laptop and smartphone). However, some services employ user-identifier cookies, which we leverage for differentiating users even if the other cookie values have changed. Furthermore, we cannot correlate the same user across services as we do not collect source IP addresses or other identifying information; thus, we refer to vulnerable *accounts*. Nonetheless, we consider this to be a small trade-off for preserving users' privacy, and consider our approximation accurate enough to highlight the extent of users being exposed when browsing popular services.

Findings. Table 5 presents the aggregated numbers from the data collected during our study. During our monitoring, we observed more that 29 million requests towards the services that we have found to be vulnerable. This resulted in 282,459 accounts exposing the HTTP cookies required for carrying out the cookie hijacking attacks and gaining access to both their private information and account functionality. Figure 7 breaks the numbers down per service. Search engines tend to expose many logged in users, with 67,201 Google users being exposed during our experiment. Every category of services that we looked at has at least one very popular service that exposes over ten thousand users during the monitoring period. Ad networks also pose a significant risk, as they do not require users to login and ads are shown across a vast number of different websites, which results in Doubleclick exposing more than 124K users to privacy leakage.

4.2 Tor Data Collection

We investigate if more privacy-conscious users are protected against our presented cookie hijacking attacks. Specifically, we explore how users employing the Tor bundle can be deanonymized by adversaries. In this case, we consider a scenario where the adversary monitors Tor exit nodes instead of public wireless access points.

While the Tor bundle offers significant protection against a variety of attacks, its effectiveness in mitigating cookie hijacking attacks varies greatly depending on each website's implementation. Even with all protection mechanisms enabled, users still face the risk of deanonymization when visiting popular sites. Therefore, the threat they face greatly depends on their browsing behavior, which we try to evaluate next.

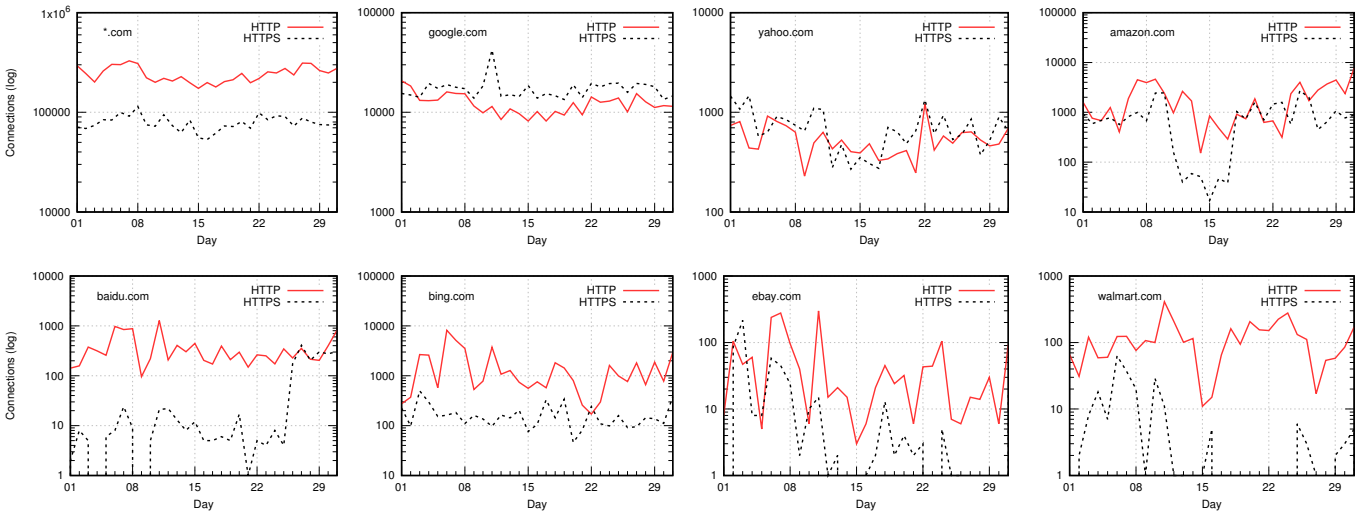


Figure 8: Number of encrypted and unencrypted connections per day, as seen from a freshly-deployed Tor exit node.

4.3 Evaluating Potential Risk

We want to explore whether privacy-conscious users actually visit these major websites over the Tor network, or if they avoid them due to the lack of ubiquitous encryption.

Ethics. We obtained IRB approval for our experiments. However, due to our ethical consideration for Tor users (as they are not members of our university nor connecting to our public wireless network), we do not replicate the data collection we followed in our experiment from university open wifi. We opt for a coarse-grained non-invasive measurement and only count the total connections towards the websites we audited in Section 3, using the port number to differentiate between HTTP and HTTPS. *We do not log other information, inspect any part of the content, or attempt to deanonymize any users.* Furthermore, *all data was deleted* after calculating the number of connections to the websites evaluated in Section 3. Since we do not look at the name of the cookies sent in the HTTP connections, we cannot accurately estimate the number of users that are susceptible to cookie hijacking attacks. Our goal is to obtain a rough approximation of the number and respective ratio of encrypted and unencrypted connections to these popular websites.

Tor exit node. We deployed a fresh exit node for this experiment, hosted on Amazon EC2. The number of outgoing connections were measured over 1 month, on a fresh exit node with a default reduced exit policy[2] and bandwidth limited to 300 KB/s.

Measurements. Figure 8 presents the number of total connections and broken down for some services. The number of connections over HTTP account for 75.4% of all the connections we saw, with an average of 10,152 HTTP and 3,300 HTTPS connections per hour. While non-HTTP traffic may be contained within the total connections, we do not distinguish it as that would require a more invasive approach. For most of the services, the unencrypted connections completely dominate the outgoing traffic to the respective domains. On the other hand, for Google we observe an average of 508 HTTP connections per hour as opposed to 705 HTTPS connections. Similarly we logged 23 unencrypted connections to Yahoo per hour and 36 encrypted connections. We do not consider the Doubleclick side channel leakage attack for Tor, as the double key session cookies employed by the Tor browser affect third party cookies and their ability to track users across domains.

5 Security Mechanism Problems and Limitations

In this section, we present certain limitations of current security mechanisms that can mitigate cookie hijacking attacks.

[2] `https://trac.torproject.org/projects/tor/wiki/doc/ReducedExitPolicy`

5.1 HSTS and HSTS Preload

5.1.1 HSTS Adoption

As HSTS suffers from the initial insecure request, HSTS Preload is a more secure option. However, in order to be on the HSTS Preload list, websites have to satisfy certain submission requirements [8]. Apart from redirecting HTTP to HTTPS and having a valid certificate, websites must serve all subdomains over HTTPS. This means that websites needs to make sure all URLs and APIs on the domain are accessible over HTTPS. While this requirement may be easy for newly deployed services, older and larger sites that still need to maintain compatibility to outdated clients, might have considerable obstacles to migrating to HTTPS.

A study by Selvi presented at Black Hat Euro 2014 demonstrated how HSTS can be bypassed, by altering the system time to be after that set by the HSTS max-age directive (possible on systems that rely on an unencrypted network time protocol [34]). Another work from Bhargavan et al. [35] also showed how the HSTS header could be partially truncated, resulting in the expiration of the HSTS entry within seconds. In addition, HSTS is in a very early state of adoption. Recent work reported that many websites fail to implement HSTS correctly [36]. The study also pointed out a very low percentage of adoption even on the Alexa top sites.

5.1.2 HSTS Partial Adoption

When HSTS is implemented only on subdomains (not on the base domain), or without the `includeSubdomains` flag, there is considerable risk for users as the site's cookies might be sent unencrypted on the part of the website that is not on HSTS, while authorizing access to different parts of the website.

For example, in the case of `google.com`, the preloaded HSTS policy for Chrome does not actually force the browser to connect to `google.com` over HTTPS. It does however employ certificate pinning; it requires an acceptable certificate if the browser is already connecting over HTTPS. This is applied to all local country-based variations of Googles search engine, and the main page itself. On the other hand, critical Google subdomains support HSTS preloading and are explicitly forced to connect over HTTPS.

Listing 2: Subset of rules in Chrome's HSTS-preload file.

```
//(*.)google.com, iff using SSL, must use an acceptable certificate.
{ "name": "google.com", "include_subdomains": true, "pins": "google" },

//Now we force HTTPS for subtrees of google.com.
{ "name": "mail.google.com", "include_subdomains": true,
    "mode": "force-https", "pins": "google" },
```

As a result, when a user visits `google.com/mail`, the connection might be sent unencrypted over HTTP, while visiting `mail.google.com` is forced to HTTPS.

5.2 HTTPS Everywhere

HTTPS Everywhere is perhaps the most well-known user-protection approach. The main limitation of HTTPS Everywhere is the coverage of the ruleset. The rulesets are created and maintained by the community, which requires a significant amount of manual effort and can result in incomplete rules. HTTPS Everywhere cannot protect the case where websites contain pages or subdomains whose functionality breaks over HTTPS, resulting in the unavoidable unencrypted connection. Therefore, user accounts are likely to be exposed even with this extension in place, since a single HTTP request is enough.

Quantifying impact. To simulate the potential impact of HTTPS Everywhere, we use the network trace collected from our campus' public WiFi, and calculate the number of accounts that would remain exposed due to URLs not handled by HTTPS Everywhere rulesets (version 5.1.0). We found that over 77.57% of all the collected HTTP traffic would remain over HTTP even if HTTPS Everywhere was installed in every users' browser. Due to those connections, 207,271 accounts remain exposed to our cookie hijacking attacks. Table 6 breaks down the numbers per targeted service. The largest impact is seen in Youtube where less than 1% of the users remain exposed while Ebay, Doubleclick and numerous news sites are not impacted at all. Surprisingly, even though Google's main page is

Table 6: Accounts from our public wireless trace (Section 4) that remain exposed even with HTTPS Everywhere installed.

Services	Exposed Accounts	Reduction
Google	31,729	53.12%
Yahoo	5,320	43.55%
Baidu	4,858	4.63%
Bing	378	38.03%
Amazon	22,040	5.68%
Ebay	1,685	0%
Target	46	0%
Walmart	97	23.62%
NYTimes	15,190	0%
Guardian	343	0.29%
Huffington	42	0%
MSN	927	39.25%
Doubleclick	124,352	0%
Youtube	264	99.21%
Total	207,271	26.62%

protected, over 46% of the users remain exposed when visiting a Google service. For the remaining search engines, the impact has a varying degree, with over 95% of the Baidu users remaining susceptible to cookie hijacking.

6 Conclusion

In this paper we presented our extensive in-depth study on the privacy threats that users face when attackers steal their HTTP cookies. We audited a wide range of major services and found that cookie hijacking attacks are not limited to a specific type of websites, but pose a widespread threat to any website that does not enforce ubiquitous encryption. Our study revealed numerous instances of major services exposing private information and protected account functionality to non-authenticated cookies. This threat is not restricted to websites, as users' cookies are also exposed by official browser extensions, search bars and mobile apps. To obtain a better understanding of the risk posed by passive eavesdroppers in practice, we conducted a measurement study and detected that a large portion of the outgoing traffic in public wireless networks remains unencrypted, thus, exposing a significant amount of users to cookie hijacking attacks. We also evaluated the protection offered by popular browser-supported security mechanisms, and found that they can reduce the attack surface but can not protect users if websites do not *support* ubiquitous encryption. The practicality and pervasiveness of these attacks, also renders them a significant threat to Tor users, as they can be deanonymized by adversaries monitoring the outgoing traffic of exit nodes.

7 Acknowledgements

We would like to thank the CUIT team of Joel Rosenblatt and the CRF team of Bach-Thuoc (Daisy) Nguyen at Columbia University, for their technical support throughout this project. Finally we would like to thank Georgios Kontaxis, Vasileios P. Kemerlis and Steven Bellovin for informative discussions and feedback. This work was supported by the NSF under grant CNS-13-18415. Author Suphannee Sivakorn is also partially supported by the Ministry of Science and Technology of the Royal Thai Government. Any opinions, findings, conclusions, or recommendations expressed herein are those of the authors, and do not necessarily reflect those of the US Government or the NSF.

References

[1] E. Butler, "Firesheep," 2010, http://codebutler.com/firesheep.

[2] D. Naylor, A. Finamore, I. Leontiadis, Y. Grunenberger, M. Mellia, M. Munafò, K. Papagiannaki, and P. Steenkiste, "The Cost of the "S" in HTTPS," in *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '14. ACM, 2014, pp. 133–140.

[3] K. Singh, A. Moshchuk, H. J. Wang, and W. Lee, “On the Incoherencies in Web Browser Access Control Policies,” in *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, 2010.

[4] A. Dabrowski, G. Merzdovnik, N. Kommenda, and E. Weippl, “Browser history stealing with captive wi-fi portals,” in *Proceedings of Workshops at IEEE Security and Privacy 2016, Mobile Security Technologies (MoST)*, 2016.

[5] N. Heninger, Z. Durumeric, E. Wustrow, and J. A. Halderman, “Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices,” in *Proceedings of the 21st USENIX Security Symposium*, Aug. 2012.

[6] J. Hodges, C. Jackson, and A. Barth, “HTTP Strict Transport Security,” RFC 6797, 2012.

[7] Can I use. HSTS Browser Support. http://caniuse.com/#feat=stricttransportsecurity.

[8] L. Garron. HSTS Preload. https://hstspreload.appspot.com/.

[9] M. Stevens, A. Sotirov, J. Appelbaum, A. Lenstra, D. Molnar, D. A. Osvik, and B. De Weger, “Short chosen-prefix collisions for MD5 and the creation of a rogue CA certificate,” in *Advances in Cryptology-CRYPTO 2009*, 2009, pp. 55–69.

[10] EFF. HTTPS Everywhere. https://www.eff.org/https-Everywhere.

[11] S. Sivakorn, I. Polakis, and A. D. Keromytis, “The Cracked Cookie Jar: HTTP Cookie Hijacking and the Exposure of Private Information,” in *Proceedings of the 37th IEEE Symposium on Security and Privacy*, ser. SP ’16, 2016.

[12] Alexa Top Site. http://www.alexa.com/.

[13] R. Gross and A. Acquisti, “Information Revelation and Privacy in Online Social Networks,” in *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society*, ser. WPES ’05, 2005.

[14] I. Polakis, G. Argyros, T. Petsios, S. Sivakorn, and A. D. Keromytis, “Where’s wally? precise user discovery attacks in location proximity services,” in *CCS ’15*, 2015, pp. 817–828.

[15] C. Castelluccia, E. De Cristofaro, and D. Perito, “Private Information Disclosure from Web Searches,” in *Privacy Enhancing Technologies*, ser. PETS ’10, 2010.

[16] A. Hannak, P. Sapiezynski, A. Molavi Kakhki, B. Krishnamurthy, D. Lazer, A. Mislove, and C. Wilson, “Measuring Personalization of Web Search,” in *Proceedings of the 22nd International Conference on World Wide Web*, ser. WWW ’13, 2013.

[17] X. Xing, W. Meng, D. Doozan, A. C. Snoeren, N. Feamster, and W. Lee, “Take This Personally: Pollution Attacks on Personalized Services,” in *Proceedings of the 22nd USENIX Security Symposium*, 2013.

[18] S. Sivakorn, I. Polakis, and A. D. Keromytis, “I am Robot: (Deep) Learning to Break Semantic Image CAPTCHAs,” in *2016 IEEE European Symposium on Security and Privacy (EuroS P)*, 2016.

[19] A. Chaabane, G. Acs, and M. A. Kaafar, “You Are What You Like! Information Leakage Through Users Interests,” in *Proceedings of the Network and Distributed System Security Symposium*, ser. NDSS ’12, 2012.

[20] comScore. (2015, Aug.) July 2015 U.S. Desktop Search Engine Rankings. http://www.comscore.com/Insights/Market-Rankings/comScore-Releases-July-2015-U.S.-Desktop-Search-Engine-Rankings?

[21] A. Acquisti, R. Gross, and F. Stutzman, “Faces of facebook: Privacy in the age of augmented reality,” *BlackHat*, 2011.

[22] I. Polakis, G. Kontaxis, S. Antonatos, E. Gessiou, T. Petsas, and E. P. Markatos, “Using Social Networks to Harvest Email Addresses,” in *Proceedings of the 9th Annual ACM Workshop on Privacy in the Electronic Society*, ser. WPES ’10, 2010.

[23] F. M. Harper, D. Raban, S. Rafaeli, and J. A. Konstan, “Predictors of Answer Quality in Online Q&Amp;A Sites,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI ’08, 2008.

[24] D. Pelleg, E. Yom-Tov, and Y. Maarek, “Can You Believe an Anonymous Contributor? On Truthfulness in Yahoo! Answers,” in *SOCIALCOM-PASSAT ’12*, 2012.

[25] J. A. Calandrino, A. Kilzer, A. Narayanan, E. W. Felten, and V. Shmatikov, “You Might Also LIke: Privacy Risks of Collaborative Filtering,” in *Proceedings of the 2011 IEEE Symposium on Security and Privacy*, 2011.

[26] J. Y. Tsai, S. Egelman, L. Cranor, and A. Acquisti, “The Effect of Online Privacy Information on Purchasing Behavior: An Experimental Study,” *Info. Sys. Research*, vol. 22, no. 2, 2011.

[27] J. R. Mayer and J. C. Mitchell, “Third-Party Web Tracking: Policy and Technology,” in *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, 2012.

[28] F. Roesner, T. Kohno, and D. Wetherall, “Detecting and Defending Against Third-party Tracking on the Web,” in *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI ’12, 2012.

[29] P. Barford, I. Canadi, D. Krushevskaja, Q. Ma, and S. Muthukrishnan, “Adscape: Harvesting and Analyzing Online Display Ads,” in *Proceedings of the 23rd International Conference on World Wide Web*, ser. WWW ’14, 2014.

[30] A. Datta, M. C. Tschantz, and A. Datta, “Automated Experiments on Ad Privacy Settings: ATale of Opacity, Choice, and Discrimination,” *Proceedings on Privacy Enhancing Technologies*, vol. 2015, no. 1, 2015.

[31] M. Lécuyer, G. Ducoffe, F. Lan, A. Papancea, T. Petsios, R. Spahn, A. Chaintreau, and R. Geambasu, “XRay: Enhancing the Web’s Transparency with Differential Correlation,” in *Proceedings of the 23rd USENIX Security Symposium*, 2014.

[32] A. Korolova, “Privacy violations using microtargeted ads: A case study,” in *Proceedings of the 2010 IEEE International Conference on Data Mining Workshops*, ser. ICDMW ’10, 2010.

[33] A. Kapravelos, C. Grier, N. Chachra, C. Kruegel, G. Vigna, and V. Paxson, “Hulk: Eliciting Malicious Behavior in Browser Extensions,” in *Proceedings of the 23rd USENIX Security Symposium*, 2014.

[34] J. Selvi, “Bypassing HTTP Strict Transport Security,” *BlackHat-EU*, 2014.

[35] K. Bhargavan, A. Delignat-Lavaud, C. Fournet, , A. Pironti, and P.-Y. Strub, “Triple Handshakes and Cookie Cutters: Breaking and Fixing Authentication over TLS,” in *Proceedings of the 2014 IEEE Symposium on Security and Privacy*, 2014.

[36] M. Kranch and J. Bonneau, “Upgrading HTTPS in Mid-Air: An Empirical Study of Strict Transport Security and Key Pinning,” in *Proceedings of the Network and Distributed System Security Symposium*, ser. NDSS ’15, 2015.