Hidden in Plain Sight: An Analysis of Browser-Supplied Privacy

Matt Brenman

Fall 2013, Tufts University

Abstract

Almost every web browser available today comes with a setting for a private mode. This leads users into believing that their web activity cannot be monitored, traced or stored while under the protection of private browsing when, in fact, there are many ways that all browsing data can be seen. This article will discuss what levels of privacy these services actually do and do not provide as well as many ways to break these systems and view the activity from a machine running an private window, such as automatic screen capturing software, browser add-ons/extensions, sniffing and availability of browsing information at different levels of network access. In conclusion, we will discuss whether or not the current state of affairs is an irrecoverable truth and why it happened.

mattbrenman@gmail.com
Mentor: Ming Chow, Tufts University

Contents

Introduction

Before analyzing the provided methods of privacy of modern web browsers, we must establish a formal definition of what constitutes private browsing since private can be a relatively ambiguous term. Confusion arises due to the many levels at which a user's privacy can be violated. Privacy is needed in many places, such as the computer, network, and outside of that connection, and also different times, which are during and after the network activity. To avoid this problem, we will assume that to count as true privacy, all browsing must be fully invisible to everybody other than the target user during and after the network activity. That is, it should be impossible for anybody to view, record, or recover the data during or after the fact.

To the community

With technological changes such as massive social media and cheap storage that allows for massive data collection, individuals suffer a massive loss in privacy. Some reasons for this loss are conscious, such as putting vast amounts of personal data on the web, while others, such as the collection of metadata, are subtler. A large part of increasing privacy and security comes from increasing awareness, and the misconceptions surrounding browser-supplied privacy modes leads to trust of the services without a legitimate basis. While some of the methods that can be used to break private browsing require knowledge of technical tools like Wireshark (or another sniffer) or forensic tools, users should be aware that these methods exist. Also, some ways of breaking private browsing can happen accidentally and can be performed/protected against in non-technical ways by the user.

Browser-supplied Private Modes

Many users have different reasons to desire a private browsing session, and a popular explanation for implementing the service is that users can buy gifts without worrying that the recipient will stumble upon the evidence. A study by Aggarwal et al. discovered that users were much more likely to use private browsing for adult sites than for gift purchases [1].

Where do the misconceptions come from?

People's misunderstandings of the functionalities of private browsing modes could either result from ignorance of the users or deceit (intentional or not) on the part of the browser suppliers. Many users mistakenly assume that a browser's built-in privacy mode supplies true privacy despite the fact that each browser gives information on what their service provides. Google Chrome claims that sites accessed in a private window "won't appear in your browser history or search history, and they won't leave other traces, like cookies, on your computer after you close all open incognito windows" [2]. Firefox, Safari, and Internet Explorer all make similar claims, and Firefox and Chrome actually warn of ways that their services may be broken. Safari and IE, however, do not obviously imply that there are imperfections to the systems [3][5][6].

It seems, based on when users are mostly using the private browsing services, that they do not understand what the services provide, despite the explicit description. This may be due to not reading the information or not understanding the technical differences, but in either case, the services are misunderstood and misused. The browsers actually do a reasonably good job on what they claim to offer, but the problem is that they take on a very limited threat model which does not encompass everything that "private browsing" implies [8].

The Threat Models and Attackers

In order to show exactly where browser-supplied privacy modes fail and succeed, we will adopt a threat model that splits attackers into two types, slightly modifying a threat model used by Aggarwal et al. A local attacker takes control of the machine at any arbitrary time T, after private mode is exited, and if the privacy mode is successful, they should not be able to gain any information about the private browsing activities on the machine before time T. A web attacker, as described by Aggarwal et al., controls a website that the user visits. A web attacker does not have access to the user's computer, but does have all of the powers that the website has, some of which should be mitigated by the browser [1]. I will add a third threat to the model, a neighboring attacker, who controls a machine on the same network but does not have direct access to the user's machine or control of any sites that the user visits. The neighboring attacker, therefore, can remain relatively secret, since they do not have to physically acquire the machine, nor would they be more or less suspect based on the user's actions like a web attacker would, due to the obvious connection between the two. The final threat to the robustness of privacy is the user themselves, who could accidentally and unknowingly break the privacy barrier in multiple ways.

How can we test privacy?

We can test the security of private browsing methods by assuming the role of the attacker in each scenario and seeing what we can do to violate the goals of the software.

Ways to Break Private Browsing

Threat: Local Attacker

The definition of a local attacker only allows for the attacker to control the computer after the private browsing session has been ended. If the attacker were allowed access before the network activity, then violating privacy would be trivial. The attacker could install a key logger [1] or one of multiple parental control software packages that allow for screen captures or live screen monitoring. Therefore, we must limit the local attacker to only having access to the target computer after the fact. The major browsers all work with the most obvious methods of privacy violation; they do not store data in obviously targeted places, such as the history, the cache, and the cookies accessible from within a new browser window.

In a study at CMU by Aditya Mahendrakar, James Irving, and Shivam Patel, the browsers' privacy modes were testing against a forensic analysis of the machine before, during, and after the browsing activity. The study used a test website that used many potential ways that privacy could be leaked, including SSL certificates, form passwords, form text entries, HTML files (up to 16MB), JPEG files from 100KB to 16MB and cookies of varying sizes [4]. When analyzing just the browser memory, Firefox only left the URL in memory, with all other memory zeroed out (and therefore irretrievable). Chrome and IE left a few HTML pages, which held less than 1MB in data, with all other memory zeroed out. Safari, however, was almost identical to the data seen while private browsing was still active. It seemed as though safari only changed the page entry tables and page directory, and they did not zero out any other data [4]. Next, the researchers performed a forensic analysis of the entire memory since the browser needed to interact with the kernel, another way that information could be leaked. In this analysis, all browsers performed significantly worse. For Firefox, the researchers were able to find cookies, form passwords, form data, the SSL certificate, and a small amount of HTML text. They were not able to recover any extra URLs that the target visited or any images. IE and Chrome performed similarly to Firefox, except all visited URLs were recovered. Since Safari performed as it did on the first test, not much more could be found, but the researchers did find the generated cookies and form passwords [4].

Threat: Web Attacker

Until recently, any website that a user visits through a private browsing mode could track its visitors through methods such as Local Shared Objects [7]. Many users know that cookies aid sites in tracking, but since they are stored differently than HTTP cookies, the normal methods of deletion would not work, so websites could track users even after they deleted their cookies or went into a privacy mode [7]. Adobe has since attempted to remedy this privacy breach. Despite this method being fixed, all traffic from a user must flow through their ISP, who can easily track the computers from which the requests were sent. If the data sent is not encrypted, they can also read that data. Due to the fact that all information flows through the ISPs, the idea of true privacy cannot be ever fully fulfilled.

Companies benefit greatly from knowing more about their users, as they can target advertising, find out the most-used services, and make business decisions based on what their users would want [7]. Netadmins of businesses also benefit from tracking their users' activity; they can monitor for illegal activity and find the user responsible to avoid liability and lawsuits.

Threat: Neighboring Attacker

A nearby adversary on the same network (switched or unswitched) can also track a user's browsing data. In an unswitched network, a computer only needs to have promiscuous mode turned on to sniff the data, but in a sniffed network, all data is routed through a switch so that the previous method does not work. In a switched network, an attacker can poison the ARP cache to gain access to everything that they would see if the network were unswitched [9]. By sniffing the network, a neighboring attacker can see source and destination IP addresses, timestamps, packet data, and much more. Private browsing modes do not perform any spoofing of the IP addresses and do not encrypt any data, so everything that would be exposed without private browsing (such as usernames, passwords, credit card numbers, and other identifiers) is still exposed. This would allow anybody sharing the network to set up a simple program to track the users and log all of their browsing history without their knowledge or consent, since many sniffers are passive and therefore cannot be detected [9].

Threat: User

Since the user is utilizing a privacy mode, the user likely would not want to hinder their goal by breaking the privacy barrier themselves, despite the many ways to do this. Many of the browsers warn the users of these potential breaches, but without understanding the limitations of private browsing, a user can easily sabotage his or her privacy. Extensions and Add-ons can sneakily break the privacy barrier if the browser does not block them in private mode; Chrome defaults to blocking extensions unless the user gives them access, but Firefox defaults to not blocking Add-ons so usability will not change between the modes [1]. An extension that may seem harmless can add code to track a user's browsing if allowed to function in private mode, which the user could easily overlook. Also, downloads and bookmarks are not removed upon exit, which allows users to accidentally expose their activity if they are not careful. Finally, if a user logs into a non-private service, such as a Facebook or Google Account, their activity could be recorded by those services and associated to their username [2].

Action items for privacy

- Disable extensions and add-ons in private browsing/incognito mode
- Use the Chrome/Firefox/Opera extension called *HTTPS Everywhere*, a tool made be The Tor Project and the Electronic Frontier Foundation, to encrypt communication between the user and the target website
- Do not put everything online
- Use Tor to help anonymize web browsing by routing through many nodes
- Be cognizant of easy to make mistakes
- Enable the Do Not Track Header, which is an HTTP header that requests that the user not be tracked. There are, however, no legal consequences for websites that completely disregard the message
- Spread awareness to help eliminate the User section

  NOTE: Many of these action items (such as Tor and the Do Not Track header) have potential issues, but they provide a first step towards privacy.

Conclusion

The overestimation of the power that comes from browser-supplied privacy modes leads to grave misconceptions about what the services do provide. As Christopher Soghoian points out, the browsers clearly state their threat models, but users do not read or do not understand the implications, and they misuse the software [8]. After learning about the reality of these modes, it is important to use them for their intended purposes while being aware of their limitations. They simply make it difficult for a non-tech savvy user to view your network activity after the browser is closed. If a greater level of privacy or anonymity is needed, then other methods should be used, but they should be thoroughly researched as to avoid unforeseen issues in the future.

Proof of Concept

To illustrate how easy it is to break the privacy barrier of a modern web browser, we will look at a PoC chrome extension, *IncogNOTo*, that silently sits in the background and records the hosts of every site that the user visits. I will also include a menu that will allow the user to view the data that has been collected. First, every Chrome extension requires a JSON manifest file that provides general information about the extension, such as:

```
{
 "manifest_version": 2,
 "name": "IncogNOTo",
 "description": "This extension demonstrates how an extension can
                 break the privacy barrier of Incognito mode.",
 "version": "1.0",
 "content_scripts": [
   {
    "matches": ["http://*/*", "https://*/*"],
    "js": ["track.js"]
   }
 ],
 "permissions": [
   "storage"
 ],
 "browser_action": {
   "default_icon": "lock.png",
   "default_popup": "popup.html"
 }
}
```

The relevant pieces to this example are the “content scripts” and the “permissions” sections. The content scripts are JavaScript files that are run on the pages that match the “matches” headings, which in our case means that “track.js” runs on URLs containing “http://*/*” or “https://*/*”. The “permissions” section allows the extensions to use the Chrome APIs, and in our case we need to use the chrome.storage API so that we can persist data between different windows and sessions. The storage of the data happens in the track.js file, which is:

```
// Run on every page that matches: "http://*/*" and "https://*/* when loaded
hostname = window.location.host
chrome.storage.sync.get(null, function (items) {
      count = items[hostname];
      if (!count){count = 0}
      count += 1;
      //Create a new object to hold the count value
      var obj = {};
      obj[hostname] = count;
      chrome.storage.sync.set(obj);
});
```

Whenever a matching webpage loads, the hostname (which is the target data for our extension) is stored in the hostname variable by accessing the page information. The call to *chrome.storage.sync.get(...)* pulls the data that we have already stored, and it utilizes a callback function, which is the majority of the code in track.js. First, the hostname count is defined, which is one more than the previous count (defaulting to zero). That count is then stored by calling *chrome.storage.sync.set(...)*, where the argument holds the hostname and the updated count. With this code alone, an extension could track your activity, and it could easily be hidden amongst many other calls or files.

To view this data, we need to add two pieces to the extension, which is a popup page that displays the data when the extension button is clicked and another JavaScript file to populate that page with our data. The HTML page that is shown is:

```
<!doctype html>
<html>
  <head>
    <title>Your History</title>
    <style>
      body {
        min-width: 357px;
        overflow-x: hidden;
      }
    </style>
    <script src="popup.js"></script>
  </head>
  <body>
      <p>History</p>
      <ul id="myList"></ul>
  </body>
</html>
```

This page just has a header and an empty list which are populated by popup.js:

```
var historyTrackerWindow = {
  showHistory: function (e) {
    chrome.storage.sync.get(null, function (items) {
      for (var key in items){
        var li = document.createElement('li');
        var text = key + " : " + items[key]
        var textNode = document.createTextNode(text);
        li.appendChild(textNode);
        document.getElementById("myList").appendChild(li);
      }
    });
  },
};
// Run the history display script as soon as the document's DOM is ready.
document.addEventListener('DOMContentLoaded', function () {
  historyTrackerWindow.showHistory();
});
```

When the DOM is loaded for the popup, the *showHistory()* function is called, which pulls all of the data that we previously stored with track.js and formats and adds it to the previously empty list. With this, the extension is complete, and now all user data will be tracked where the extension is running. If the user disables the extension in Incognito mode, then the extension cannot run, but some users may want the capabilities that the extension provides to extend into the private windows, which is where a malicious code could track all user activity.

References

[1]: Aggarwal, Gaurav, Elie Bursztein, Collin Jackson, and Dan Boneh. "An Analysis of Private Browsing Modes in Modern Browsers." N.p., n.d. Web.

[2]: "Incognito Mode (browse in Private)." *Chrome Help*. Google, n.d. Web. Dec. 2013. <https://support.google.com/chrome/answer/95464?hl=en>.

[3]: "InPrivate Browsing." Windows IE9 Features. Microsoft, n.d. Web. Dec. 2013. <http://windows.microsoft.com/en-us/internet-explorer/products/ie-9/features/in-private>.

[4]: Mahendrakar, Aditya, James Irving, and Shivam Patel. "Forensic Analysis of Private Browsing Mode in Popular Browsers." Carnegie Mellon University, n.d. Web.

[5]: "Private Browsing - Browse the Web without Saving Information about the Sites You Visit." *Firefox*. N.p., n.d. Web. <https://support.mozilla.org/en-US/kb/private-browsing-browse-web-without-saving-info>.

[6]: "Safari 5.1 (OS X Lion): Browse privately." Apple Suppoer. Apple, n.d. Web. Dec 2013. <http://support.apple.com/kb/ph5000>.

[7]: Soltani, Ashkan and Canty, Shannon and Mayo, Quentin and Thomas, Lauren and Hoofnagle, Chris Jay, Flash Cookies and Privacy (August 10, 2009). Available at SSRN: http://ssrn.com/abstract=1446862 or http://dx.doi.org/10.2139/ssrn.1446862

[8]: Soghoian, Christopher. "Why Private Browsing Modes Do Not Deliver Real Privacy." Center for Applied Cybersecurity Research, Indiana University, n.d. Web. <http://www.consumerwatchdog.org/sites/default/files/resources/soghoian2.pdf>.

[9]: Spangler, Ryan. "Packet Sniffer Detection with AntiSniff." University of Wisconsin, May 2003. Web. <http://207.228.3.42/Sniffer.Detectors/snifferdetection.pdf>.