# Un-Incognito

Avoiding Detection of Incognito Mode

Abhishek Modi (akmodi2)
Stephen Sullivan (sksulli2)
Paul Rachwalski (rchwlsk2)

## Introduction

In 2016, privacy is no longer an assumed property. Modern web browsers have a feature called “incognito mode” that a user must explicitly enter to ensure their privacy when browsing the internet. Without it, there is a clear record kept of a user’s actions on whichever websites they visited when the user leaves the computer.

According to “An Analysis of Private Browsing Modes in Modern Browsers,” incognito mode can protect a user from two types of attackers: local attackers and web attackers. A local attacker is one that has full access to the user’s computer only after the user browses the web (if they have access beforehand, a simple keylogger gets around any incognito defenses). A web attacker is within the website itself and tries to link data from the user browsing in incognito mode to the same user’s public data. Due to these defenses, we believe incognito mode is indispensable in today’s web landscape.

### Google Chrome’s Incognito Mode

Google Chrome’s Incognito Mode follows many of the standard methods of information removal to prevent local attackers from obtaining useful information. Some of these methods include deleting history, cookies, HTML5 local storage, and caches from the user’s incognito session. It also does not allow the user to access the password database or form autocompletion. Any of these could be used by local attackers to see which websites and pieces of information were accessed during an incognito session if not removed.

When it comes to web attackers, Chrome protects users by disallowing cookies set in incognito mode from being available in public mode, nor does it allow cookies set in public mode to be available in incognito mode. Incognito mode is not a fully anonymous, however, since IP address can be used to link data from public and incognito sessions, and there is little the browser can do so mitigate this.

Another useful and important feature of Chrome is that it disables extensions in incognito mode by default. The user must explicitly allow the use of individual extensions. This prevents the possibility of malicious extensions from gaining and storing/distributing personal data without the user’s consent.

# Detection Methods

The following methods to detect incognito mode and defeat said detection were discovered throughout this project across various websites.

## CSS

### How it Works

Sites can use the computed styling of links to determine if a browser is in incognito mode. To this end, a site can force a browser to visit a link by using a <xxx src> or a <link> tag. If the browser is not in incognito mode, it will remember that this is a link that has been visited and will render any link to that url as a visited link. A browser in incognito mode however, will render it as a link that hasn’t been visited. For instance:

| | |
|---|---|
| link-visited | link-not-visited |

Sites can have hidden elements or even 0px sized elements that hold these links. A client side script can then check the computed styling to determine if a browser is or isn’t in incognito mode.

### Countermeasure

It isn’t currently possible to check exactly which CSS rules have been applied to an element using JavaScript. Only the final computed styling can be checked. This is to say that a script can’t check if the ‘link visited’ rules or the ‘‘link not visited’ rules are the ones that have actually been applied. They can only check things like the color or font.

This means that we can create a rule that overrides the <a> tag styling. This rule will make every link have the computed styling of visited links. This is fairly simple to do using a chrome extension.

## RequestFileSystem

### How it Works

JS can access a sandboxed filesystem on the client machine. Calling `window.requestFileSystem()` normally returns a FileSystem object. However in incognito mode, the function throws an exception. Sites can therefore check the result of this function call to check if one's browser is in incognito mode.

### Countermeasure

A browser extension can swizzle the filesystem api to change how it behaves when the browser is in incognito mode. In particular, it can create storage that is handled by the extension and return FileSystem objects that interface with this specially handed storage.

In this way, the FileSystem can be made to appear as though it is working in expected manner even if the browser is in incognito mode.

# Cases of Interest

We analyzed several websites, in particular, that provide popular services using incognito mode detection to block users.

## Netflix

Netflix, the incredibly popular movie and TV show streaming service denys service to users browsing in incognito mode. They report that incognito mode "prevents Netflix from using cookies to store information needed to stream video." We think that this should not be the case, as other DRM protected streaming services operate just fine in the presence of incognito mode, and users should have the right not to be tracked, even on Netflix.

The Netflix clientside javascript player is a complex beast to wrangle. Comprised of roughly 32k lines of obfuscated javascript, we were able to reduce the problem down to the DRM decoding performed clientside. This process fails in incognito mode, for reasons we do not completely understand, and prevents video playback in Netflix. Our direct attempts to circumvent this consisted of injected content scripts through the chrome extension. We tried swizzling several methods in the DRM decryption process to return success values when normally they would return failure values, i.e. navigator.requestMediaKeySystemAccess and methods on the MediaKeySession interface. We encountered difficulties swizzling the native code methods in

Chrome, so perhaps a better approach would be to fork the Chromium browser project and reimplement the DRM decryption methods in native C++, modifying the browser's source code.

Our indirect method using the Mac application was also unsuccessful, as the Netflix player requires Microsoft Silverlight support, which is unavailable in Mac OS X's Cocoa WebView interface. Even with system Silverlight support in place, the mac app cannot view Netflix streams.

### HBO Go

Another popular online streaming service that disallows use in incognito mode is HBO Go. Browsing the site is possible, as is viewing the intro video to every TV show, but the user is presented with the error message "This browser/platform combination does not allow DRM protected playback when in incognito mode" as soon as the show is supposed to start.

HBO Go relies on Adobe Flash Player to play the videos for the users. This poses a problem when it comes to bypassing DRM protection. Flash Player is installed by users on their own computers, which means that the code for it runs outside of the browser. The problem that arises from this configuration is determining how Flash Player detects that the user is in incognito mode in order to disable DRM protected content. After testing several of our previous methods for bypassing incognito detection (such as swizzling the RequestFileSystem call), we arrived at the conclusion that Flash Player is likely using Chrome's internal flags to determine if it is in incognito mode. This means that bypassing the detection would involve modifying/swizzling native code, in alignment with our discoveries about Netflix.

Our indirect method using the Mac app defeats HBO's incognito detection schemes.

## Generated Applications

### Mac App

(reproduced in part from the README)

This native Mac OS X application implements a special browser that employs host-agnostic circumvention to defeat any cache, local storage and cookie based incognito detection schemes by totally clearing any stateful information stored on the client browser upon navigation to each new domain. This scheme is important because it a) allows you to use sites as you would normally that require session information, i.e. logging in to a website with an account, while b) preventing tracking information from transferring into, out of, or within the browsing session. This app, for instance, allows you to log into HBO Now and stream video content while also preventing your viewing history from appearing in ads on Amazon.com.

### Chrome Extension

(reproduced from the README)

Designed to defeat a small subset of the incognito mode detection schemes, this Chrome extension can be loaded into Google Chrome and then enabled in incognito mode (regard the source to ensure we don't do anything nefarious) to defeat the file system request verification, and the media session verification. The test pages in `Un-incognito-tests` show off some of this functionality, i.e. without the extension running, in incognito mode, the pages will show indications that one's browser is in incognito mode, but they will not appear if in incognito mode with the extension running.

## Conclusion

Ultimately, defeating incognito detection schemes is, in some cases, relatively straightforward if you know the techniques, but some services implement detection schemes that are even more sophisticated than can be easily circumvented, cognisant even of the current literature's cutting edge detection techniques. It may be that proprietary, non-open source techniques are used in some of these cases, and the surmountability of these techniques is questionable, given the difficulty in ascertaining the behavior of modern clientside javascript applications. We found this to be the case despite recent advances in java decompilers, ubofuscaters, and the like.

A generalized approach, however, may be the best solution to this problem, as most incognito mode detection schemes rely on a nonstandard browsing mode to detect incognito mode. By merely clearing information stored on the client repeatedly when browsing, we can emulate the beneficial features of incognito mode while using a standard browsing mode. This gives most websites no detection vector; though there are exceptions. Further developments to our Mac app browser could be made to increase its efficacy, especially in the area of 3rd party application/add-on support.

## Sources

Aggarwal, Gaurav, et al. "An Analysis of Private Browsing Modes in Modern Browsers." USENIX Security Symposium. 2010.

Amatriain, Chen, et al. “Why doesn't Netflix allow you to watch movies with Chrome's incognito mode?” Quora. 2011-2015.

W3C. “Encrypted Media Extensions.” https://w3c.github.io/encrypted-media. W3C Editor’s Draft. 26 April 2016.

Google Chrome Team. “chrome.fileSystem.” https://developer.chrome.com/apps/fileSystem. Chrome Developer Documentation. 2016.