Search History Privacy Though Obfuscation: Introducing the Haystack Project

David Huerta
david@haystackproject.com

January 10th, 2009

Abstract

This paper illustrates a proposal for obfuscating search queries with machine-generated queries that are designed to be as close to indistinguishable from the human generated ones as possible. This is ultimately to discourage the use of search histories as a source of user profiling and incrimination. The architecture and specific techniques to be used by the Haystack Project's search tools to do this are explained here in technical detail.

Introduction

In 2005, the US Department of Justice issued a subpoena duces tecum to major search engines asking for large samples of search queries which were to be theoretically used in analyzing the availability of pornography to minors. The subpeona included a demand for "all queries that have been entered on your company's search engine between June 1, 2005 and July 31, 2005, inclusive."[1] It did not ask for IP addresses or user accounts linking those search terms to particular users. The inclusion of this would have allowed the federal government an ability to profile users based on their search histories. Given the eager and bipartisan disdain for privacy protection in the US Government in the years since then, however, we may not be so lucky next time. Across the pond, recent data retention laws in the European Union impose regulations on search engines that would force them to keep an archive of search histories that would be all too tempting for governments to abuse. This project aims to make the EU's collection of search-related ISP activity less than useful.

The idea of using machine-generated search queries to hide human-generated searches and obfuscate search histories is not a new one. It has been discussed recently by Martin Eberhard in the Spring 2008 issue of 2600. Many of his ideas, even the name, are used in the Haystack Project (or HayProj, for short). Two implementations for obfuscated search are proposed: A web-based search tool, and a more powerful browser plug-in, which is functionally closer to Eberhard's proposal. Mobile applications might also arise in the future and, depending on the capabilities of the mobile device, function basically the same way the browser plug-in will.

What Hay Looks Like

Machine-generated anything can lead to predictability, which can lead to pattern detection, which can lead to the magnetic fist of Big Brother finding your search shenanigans in our proverbial haystack. A bored 12-year old on an FBI job shadowing field trip could easily distinguish the human-generated search from the fake ones in this bit of search history:

```
2012-11-17 14:05     Plum
2012-11-17 14:10     Camera
2012-11-17 14:15     Quorum
```

[1] American Civil Liberties Union v. Alberto R. Gonzales http://www.google.com/press/images/subpoena_20060317.pdf

```
2012-11-17 14:17     cascadian independance army Portland
2012-11-17 14:20     Platypus
2012-11-17 14:25     Xylophone
```

With only the most rudimentary addition of basic dictionary words, its relatively easy to spot which term our suspect was actually looking for. Here's why:

- Search term source is a dictionary. We know this because its in the program itself, and the word list never changes.
- Generated terms are only single words.
- Generated terms start with an upper-case letter.
- Generated terms are perfectly spelled.
- Searches all take place on predictable time intervals.
- Seriously, who searches for “xylophone?”

This has but one source, a dictionary. It has only one location for the contents, and the contents never change. No matter how human some search terms may appear, if the list is a single, static list, simply writing a script to check which terms in a search history are not in the source term list will be enough to find which ones could not have been generated.

HayProj's raw search term sources will consist of a myriad of websites and web services that can dynamically change their contents given some parameters. These parameters will be randomized to further deter the predictability of what the contents will look like. For its initial release, Haystack will use Google's Hot Trends page, which can provide lists of popular search terms generated by other humans on a given date between a few years ago and now. The Google Hot Trends web service would be a cleaner and more elegant means of getting this data, but it currently does not accept a date for a parameter. Because the time frame that can be queried for search terms gets larger as time goes on, Hot Trends makes this approach more effective in daily increments, until the oldest available date is changed, which happens occasionally. For its initial implementation, HayProj will scrape terms from Google Hot Trends from two random dates with a random division of the total number of searches from each date, so that the possibility of very current search topics can't be ruled out as not being machine-generated if the surrounding searches are somewhat dated (e.g. finding “Obama re-election” surrounded by searches related to the 2008 primary election).

Hot Trends retrieves the top 100 terms for a given day, but running 100 searches can put a dent in search performance, adding a lot of wait time. Haystack will retrieve a random number of randomly chosen terms from each source, but only to a maximum total sum of 20 search terms and minimum of 5. The maximum term limitation may be relaxed in the future as browsers become better at rendering JavaScript faster.

As large and extensive as Google's Hot Trends is as a source of search terms, we can make it harder to build a source to compare search histories with by adding similar sites and services to the mix that can retrieve content dynamically. News website articles could be scraped by changing date parameters, random Wikipedia articles could be used as a term source, or even content from highly active messaging boards and systems too large to archive their content could be used.

Beyond Google Hot Trends

Hot Trends neatly retrieves real search terms, which vary in the total number of words in a term and might not always be spelled correctly. This isn't the case with other sources. Scraping the Drudge Report, for example, would require some way to combine words together to make phrases. Since the subject of a news article is likely to be mentioned more than once, Haystack can check for the number of times its mentioned and filter out non-nouns—this means searches for “and” and “in” by themselves might not be protected, which I'd imagine isn't exactly going to be a deal-breaker for most users. Another easy shortcut would be to look for the keyword-ad links that are being placed on news articles if available in the page. To make it easier for sites to be added to HayProj as term sources, HayProj will standardize its scraping code to work for any site using a particular publishing platform, such as Wordpress, so that HayProj will already know which part of the site to look for content in, and what formats are available for looking at content written in a particular date, having a standard set of permalink formats, etc.

Phrases collected by machines might leave a bit of predictability in the order of words in a phrase, and even having this predictability in human searches leaves an ability to search for phrases or multi-word parts of phrases if a forensic analyst was to look for a particular phrase in a theoretical collection of every possible content source. To deter this, HayProj has an option to randomize the order of the search terms, which, at least in Google, doesn't usually affect the search results much. If other search engines are less robust, maybe that's why no one uses them.

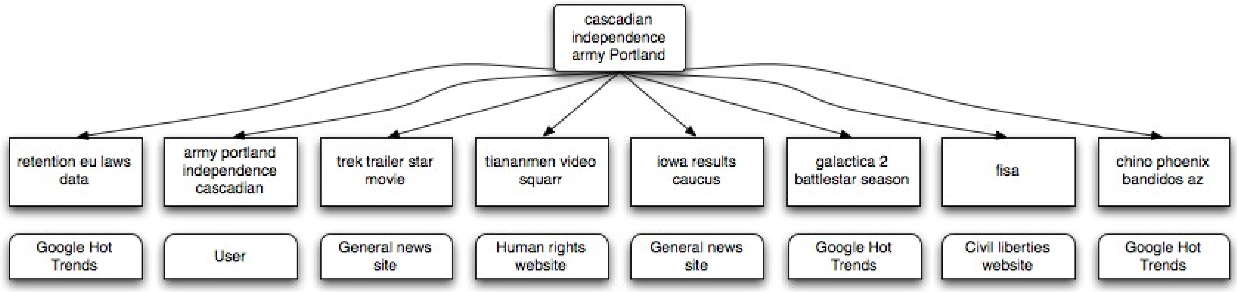
The spelling and punctuation in a term also poses a similar risk. To deter the problem of punctuation, all letters in a search term are made lower case. Spelling errors can be addressed by a system of random letter replacement that will have a bias towards replacing a random total of letters between 0 and half the total number of letters in each word with letters closest to them in a QWERTY keyboard layout, but still leave some smaller possibility of a letter being replaced by any random character that can be typed. Due to the possibility of this being a performance hit and the fact that spelling errors can carry a certain personal touch, the user will be an option to choose whether to include typos or not.

Ideally, the number of web pages and web services used as sources should increase indefinitely within reason—the reason being that a browser plug-in with over 9000 content grabbing scripts would be huge and slow. Since that cloudy technical limitation exists, a fair portion of the sources should be focused on retrieving content that would be suspect by most governments, to increase the protection for journalists and political activists who would have a more politically or socially active search history. I predict—I hope this sort of search privacy tool will be popular if useful in that crowd.

A search made in HayProj will be accompanied by a random number of generated searches under between 5 and whatever number is bigger than 5 and smaller than make-your-browser-crawl.

Currently the limit is 20. The order that the human search term and generated search terms is submitted to the search engine (currently only Google) is also randomized, but only the results for the human-generated search are displayed. Each burst of searches can be triggered by a human-initiated search or by a combination of human-initiated searches and machine-generated searches that are initiated at random intervals of time between 1 to 10 minutes after the previous search runs. In the web interface, machine-generated search bursts run after the first human-initiated search is ran and continue until that tab or window is closed. In the browser plug-in, machine-generated searches will begin automatically and continue while the browser is open.

An example of a user-generated search is illustrated below, generated keyword sources are listed below each search term. Machine-generated searches create the same type of search burst.



There's Definitely Definitely No Logic; To Human Behavior

Actually there is, in search behavior. Humans click results. Sometimes more than one result. Sometimes they search for related topics with a short period of time. This presents some interesting challenges. Using Google's Web History tool, you can look into a history of searches you've made in addition to which result you looked at. The way this is addressed in HayProj is by choosing a random number of results between 0 and the total number of results returned to navigate to behind the scenes. Since most users won't click more than half of the search results returned, a bias can be added to make this number more likely be around the 0 to 5 range, while still leaving some wiggle room for people insistent on looking at more pages.

To fake topic-related searches, HayProj will use similar techniques used to retrieve search terms from non-GHT websites, to retrieve a random number of search phrases from the descriptions of search results and include those terms in future search bursts that will be chosen at random.

No Need for Trust

The HayProj plug-in and web application will be Free software with a capital F, licensed under the GPL. In addition to this, the plug-in and web interface[2] code is written in client-side code (JavaScript), so that the user can see precisely what happens when they run it. This addresses the

[2] The web interface will use a limited amount of server-side code to work around most browsers' XMLHttpRequest cross-domain security policy that applies to web content. This code will be publicly available and will be written in a way that allows the user to audit it with instructions on how to do that posted on the Haystack Project website.

concern that HayProj might be hiding something nefarious in server-side code that may be open-source, but still different than the source being publicly distributed.

This still entails a certain amount of trust on the underlying browser and operating system to run the plug-in's code without any special interpretation or back doors. HayProj is currently being developed for the open source Mozilla Firefox browser. Ideally, the browser should also run in an open source operating system.