John Pires

SE 4930

January 10, 2012

Article Review 2:  
 I still Know What You Visited Last Summer: Leaking Browser History Via User Interaction and Side Channel Attacks

**Summary**

Browsers make a distinction between visited and unvisited URLs when rendering a page. This distinction is meant to provide users with a simple, visual indicator of which links have already been visited. History Sniffing is a Web exploit in which a malicious site uses JavaScript to check the rendered color value of a list of URLS. These values can then be used to determine which of the sites on the list have been visited. There are a number of uses for such an exploit, some malicious. For instance, sites, such as banking sites, which are in danger of being impersonated, could use the technique to check against URLs of known phishing sites. Sites which make use of federated login can streamline the process by checking against URLS such as Facebook, OpenID, Connect, etc. The exploit can be used for advertising purposes or to check whether or not users are visiting competing websites. The list goes on and on, and although some of the uses are less malicious then others, they are all violations of privacy.

The URL sniffing exploit has been known since 2002 and until David Baron of Mozilla developed a defense in 2010 it was completely un-defendable. Baron’s defense blocks all automated techniques for this attack, but still allows Browsers to distinguish between visited and unvisited sites. However, this defense does not defend at all against interactive attacks. These interactive are much slower and can only check small lists of URLS but unlike automated URL sniffing they remain undefended against. Furthermore, sites utilizing URL sniffing generally only concern themselves with relatively small lists of relevant URLS: popular social networking sites, direct competition, important financial sites etc. That is to say, generally the number of sites relevant to a particular URL Sniffing attack is still well within the limit of the still undefended interactive URL sniffing exploits. The purpose of this paper is to discuss automated exploits and their defenses as well as the still-undefendable interactive exploits.

In order to give context to the rest of the article the authors first cover the mechanics of the now obsolete automated history sniffing exploit. This technique allowed sites to sniff history “automatically, rapidly, and invisibly.” Three techniques are discussed: Direct Sniffing, indirect sniffing, and side channel sniffing. In direct sniffing a JavaScript program examines the page it is embedded in to obtain all CSS properties which influence the drawing of the page. It then uses the color properties to determine which URLS out of a specified list have been visited. Indirect sniffing is when the difference between visited and unvisited links is used to cause changes to other elements on the page. For instance they could be made to take up different amounts of space which would cause unrelated elements to shift or the different links could cause certain images to load etc. Side channel sniffing is when a system “leaks information through a mechanism that wasn’t intended to provide that information, bypassing the system’s security policy.” For instance, when a cache returns a piece of information faster than it could be retrieved from the source it reveals that the resource was previously accessed.

The defenses developed by Baron block all known techniques for automated attacks. Direct sniffing is thwarted by having the API pretend that all links are unvisited. In order to prevent indirect and side channel sniffing the ability of CSS to control the unvisited/visited distinction has been limited to prevent all known exploits. Also any algorithms which can be used in timing exploits have been adjusted to always take the same amount of time. However, these defenses make no attempt to defend against attacks in which the victim’s own actions reveal browsing history.

Four interactive attacks were tested and discussed for the article. The attacks were designed to probe 8 to 100 links, which is still within the useful range for a malicious site. The first attack makes use of CAPTCHA technology. Words are generated based on a URL list and displayed in the CAPTCHA. Words based on unvisited links are styled the same color as the background, so the user only types words based on visited sites. A second use of the CAPTCHA exploit victims are asked to type what appears to be a string of letters, numbers and dashes but which in reality is four characters superimposed into one which mimics 7 segment LCD symbols. For instance, 4+5=9, 4+f=A etc. This design tests 3 links per character, so a 12 character CAPTCHA will probe 36 sites. Another exploit also makes use of user validation. A chessboard is presented to the user and the user is asked to click on each pawn to prove that they are human. However, each pawn is in fact every square contains a pawn, but each is a hyperlink to a different website, and only the pawns corresponding to visited sites are made visible. Another exploit asks the user to select two images which are then assembled to produce a composite image. The image fill colors depend on the “visitness” of four hyperlinks. There are four choices for each of the two images to be selected; together, they exhaust the sixteen possible appearances of the composite image.

The results showed that all of these exploitation techniques have the potential to be successful. However, some were more successful than others, mainly due to user frustration with the presented task. The Chessboard exploit was the most successful with nearly 100% accuracy and nearly 1000 queries per minute. The worst was the character CAPTCHA which proved to be so frustrating to users that many gave up, or could not be bothered to carefully enter the correct information.

A different, but very interesting, type of exploit presented made use of webcams. The exploit involves making use of the user’s webcam (potentially through posing as a video chat site) and then changing the color of an area of the screen (potentially a fake advertisement) based on a list of URLS. The webcam is used to monitor the light reflected off the user and determine whether the current color represents a visited or unvisited link. This technique proved to be very successful in the best conditions but in unfavorable conditions, such as a dark room or a moving person the accuracy dropped significantly.

The article concludes by stating that automated sniffing attacks have been blocked. However attacks which make use of user interaction remain possible and a single attack can be used to probe a significant and useful number of sites. There are currently no defenses against such attacks.

**What was learned**

I did not learn much technically from this article but it was incredibly interesting reading about the different techniques and exploits presented in the paper. I did learn quite a bit about history sniffing in general and certain attacks in particular. The ideas behind some of the attacks were ingenious and also incredible in their simplicity.

This article will absolutely help in future software development. For one thing I learned about certain specific browser vulnerabilities. However what was really useful about this article is that it got me to think outside the box. Many of the exploits were technically simple but also clever and effective. For instance, the side channel attacks which worked through recording response time was able to get useful information through a seemingly unrelated metric. The article really got me thinking about how ambiguous and not-straightforward things can be in software.

The biggest questions I have regarding this article would be:

1. Are there any big-name sites which made use of these attacks?
2. What research is being done to defend against the interactive attacks outlined in the article?

Works Cited

Weinberg, Zachary, ERic Y. Chen, Pavithra Jayaraman, and Collin Jackson. "I Still Know What You Visited Last Summer (Leaking Browsing History via User Interaction and Side Channel Attack)." *2011 IEEE Symposium on Security and Privacy* (2011). Print.