Nabi - A Preventative Solution Against JavaScript and Scriptless Tabnabbing Attacks

Michael Baraboo, William Schattgen, Christopher To

4 May 2015

Abstract

1 Introduction

2 Background

2.1 TabShots

One approach to combating tabnabbing is a Google Chrome browser extension known as TabShots, which was created by Philippe De Ryck, Nick Nikiforakis, Lieven Desmet, and Wouter Joosen [1]. To detect whether a tab has changed or not while not in focus, TabShots uses visual comparison of the tab's appearance before and after the change in the user's focus. The extension records screenshots of the currently focused tab at regular intervals, using a simple Google Chrome API call and storing the screenshot as a data URL. The program also keeps record of the tab's favicon, or the icon displayed in the tab's url and title space. These two sets of images provide the basis for comparison when a tab regains focus.

When any tab regains focus, TabShots compares a new screenshot of the tab and it's current favicon to the most recent stored screenshot and favicon that was recorded before the tab lost focus. The small favicons are compared by source, and the screenshots are compared in a visually divided manner using the HTML5 canvas element. The screenshots are divided

in a raster of fixed-size tiles, and each tile in the new screenshot is compared to the most recently stored screenshot of the tab. The tiles are sized as 10x10 pixels, which are deemed by the authors as a balance between performance and precision. If the tiles do not match exactly, that area is marked as changed. The HTML5 canvas element provides powerful image manipulation capabilities that allowed the rastering and comparison algorithms to be implemented.

Once the differences in the current tab from its previous version are calculated, an overlay is injected into the page that shows the differences in semi-transparent red. The overlay is both transparent in both visibility and mouse events, so no mouse and keyboard events will be affected it. The overlay is constantly checked for its presence by the extension whenever an element is removed, ensuring that a malicious page will not be able to remove the overlay without notifying the extension and user. TabShots also places an icon onto the browser's toolbar that changes color in response the amount of changes on the currently focused tab. The icon serves as an unobtrusive security layer that will remain in view should the overlay be removed.

Finally, there is also an intended component for blacklisting sites that have confirmation of tabnabbing attempts. The developers have created an optaionl server-side component which can pull reported URLs from individual users, and add the confirmed URLs to a database. If the user decides that the current page is attempting tabnabbing and provides their explicit approval, the application can be signaled to send the accused page's URL, the image before the user switched tabs, and the image of the page after the user switched tabs. The server-side application uses its own automated comparison process and the further verification of a human analyst to decide if the page is indeed a phishing page, and whether the URL should be added to a subscription model blacklist.

According to the testings of the creators, TabShots is said to provide large compatibility with most major sites on the internet at a seemingly acceptable processing speed. The extensions was used on the top 1000 websites listed on Alexa.com, and the results showed

that 78% of these sites fall within the safe threshold of 5% or less in changed blocks, meaning no compatability issues exist with that page. About 19% of these sites have moderate changes less than 40%, which is the threshold for high amounts of changes, and 3% report more than 40% differences in changed blocks. While the last two numbers may seem high in terms of false positives, the authors note that TabShots never interferes with the functionality of a page, and does provide a whitelist to providing overlays for these sites. In terms of speed, testing showed that TabShots was able to prepare and calculate the image comparisions on a 1366×768 window within an average time of 284ms, which should provide little impact on browser operating time.

There are some problems with this application. First, the performance testing of TabShots was done on a 1366×768 window. Many users today have monitors with much larger resolutions such as 1920×1080 or 2560×1440, which provide two or four times as many pixels. This means that TabShots will run considerably longer on these types of monitors and provided a much larger performance strain. TabShots is designed to calculate the comparisons every time the user changes tab, which lead to very frequent occurrences of calculations depending on user behavior, and can further exaggerate TabShot's slower performance on higher resolution monitors. Finally, the visual comparison property of TabShots leads to many occurrences of false positives that lead to the high numbers mentioned above. Examples of non-malicious behavior that would trigger overlay occurrences would be videos, GIFs, image slideshows, overlays, and other dynamic advertisements. The authors note that slowly images and files would flag large amounts of tile changes. In addition, if an element was added or removed from the page, every other element on the page would shift, which would flag a major change in the picture comparison algorithm. The uncertain performance of TabShots on large monitors or frequently changing tabs, as well as its high opportunities for false positives make TabShots a potentially unideal solution to tabnabbing.

2.2 TabSol

2.3 NoTabNab

3 Proposal and Methodology

4 Results

5 Conclusion

References

[1] Lieven Desmet Wouter Joosen Philippe De Ryck, Nick Nikiforakis. Tabshots: Client-side detection of tabnabbing attacks, 2013. 978-1-4503-1767-2.