Kepler - Raising Browser Security Awareness

Thomas Wahlberg, Petri Paakkola, Christian Wieser, Marko Laakso and Juha Röning
University of Oulu, Department of Computer Science and Engineering
OUSPG
Email: ouspg@ee.oulu.fi

***Abstract*—Web browser security is lacking and browsers are often unable to detect what is unwanted traffic, like contacting tracking sites. Without our direct knowledge, we are exposed to different kinds of security risks during web browsing. Browsing the web causes our browser to render a web page from HTML, CSS and JavaScript but many things that we cannot see happens during this. The web page may have saved data on our computer, tracked sites we have visited, sent us unwanted advertisements and maybe even executed a malicious script. This triggers the need for users to be able to determine themselves which risk they are willing to take. To improve browsing security, third party developers are developing browser extensions. By giving the users detailed information about web requests, we raise awareness and help users make decisions themselves about which websites are secure. To this end, a prototype for displaying this information was developed. The prototype is a Google Chrome extension which gathers and displays detailed information about web sites' activities in real-time. With the extension we are able to inspect every request made by the browser in detail. We can also filter the data according to country. The data is presented in a human readable form using geolocation. With the help of this prototype, users' awareness of web browsing security is raised in an informative and interesting way.**

***Keywords*—Browsers, Computer security, Data visualization**

I. Introduction

A. Background

Web browsers are the main way to view and inspect resources available in the Internet. These browsers are built to send HTTP (HyperText Transport Protocol) requests, interpret responses and render resources, such as images or HTML documents. The most common browsers in descending order are Chrome, FireFox, Internet Explorer, Safari and Opera [1].

To allow the user view performed activities, major browsers support different built-in or add-on inspectors, like the Chrome Developer Tools [2] or FireBug [3]. Add-ons for browsers are software that add features to the browser and can can be developed by anyone. These add-ons helps us get more knowledge about what exactly happens during browsing, including where resources are requested from, how long the requests take and what kind of responses were returned. The information in these inspectors is cluttered, unorganized and unfiltered. Creating a visualization about the data makes it more accessible to every user. Also representing data in a visual form tends to be more convenient and pleasing, thus raising the awareness of web security. Filtering the data is also one key point in helping users to see exactly what they want to focus on. Findings from such a visualization can help to notice unwanted traffic or find saved cookies that track the browsing even between sessions. To notice such events it is required to be aware of the details, such as the destination of the web requests. Filtering and pointing out where data is received from can help us detect unwanted traffic. For one to create such a visualization, the main thing is that the data must be available.

One approach is to make a browser extension or add-on. The terminology varies between browser developers. Using a popular browser as a platform for writing an extension to it makes this kind of development fast and the final software easily distributed. Browsers have their own application stores so the distribution channel is already at the hands of millions of users. Also the browsers have their dedicated APIs (Application Programming Interface) to make such a visualization possible.

Our goal was to make an extension to help non-technical users find out what is happening behind the scenes during web browsing. Using appealing presentation directly within a browser aims to raise interest and improve informativeness of the data.

B. Web requests

A basic form of communication in the web is the client-server model. Clients send requests of data they want to a server while the server is listening for any client's requests and replying to them accordingly. A web browser is a client program which sends requests to web servers. Web servers in turn are programs run on server machines. The user types in a URL and the browser sends a web request to the server behind the URL. The server replies with a message which usually contains the structure of a web page. The browser reads the reply sent by the server and renders it.

The server may have replied with information about a resource located on another server. This triggers the web browser to send another request to the destination defined in the first server's reply. This happens very often on some pages. For example, a local news magazine's web site can cause almost a total of 300 different web request to several different countries. This information is completely hidden from the average user. In the worst case this unawareness is taken advantage of and the user is given malicious, harmful or spam-like data.

Presenting the user with an informative, filterable list of all their web requests raises awareness of web traffic. When doing online banking the user should be seeing encrypted (HTTPS) communication only, preferably to one server. Finding insecure communication to foreign countries should give reasonable

evidence for the user to be doubtful of the web site's intentions and security.

II. Related work

A. Users

The field of web security and how well users perceive issues related to that has been studied [4], [5]. In a research regarding users' conceptions about web security it was shown that even people with a technology background make often mistakes when trying to identify the security of a website [4]. Tests were conducted with different groups of people: suburban, rural and high-technology. The groups were given tasks to tell whether a website was secure or nonsecure. 50% to 67% of the participants (depending on background) from these groups correctly recognized a secure connection. The tests also indicate that recognizing non-secure connections are easier than secure connections. The two most common indicators in recognition of secure connections are the use of HTTPS protocol and the corresponding icon (lock or key icon). In contrast the icon was also the most frequently used indicator in wrongly evaluated connection type. This happened when the website in the task was not really secure, but used HTTPS as the transport protocol, thus displaying the icon.

The use and effectiveness of these indicators has been studied further [5]. With the help of eye-tracking data from a series of tests it was possible to analyze the awareness and behaviour of users. Participants in the tests browsed various sites and completed multiple tasks, including logging into a webmail account and using credit cards as if they were at home or at work. After the tests the users filled out a questionnaire regarding the security of web services. The test was redone so that participants were adviced to focus on the security of the browsing.

The results were inconclusive for the first parts of the tests as the researchers were not able to simulate natural web browsing. The second part indicated that in addition to overall type of site the lock icon was the most used visual cue used for security. This result is in line with the research conducted in the year 2002 [4]. It is also stated that certificate data is rarely viewed and understood. Also, after a login, people tend not to seek any further security information. The results indicated in both that there is need for a more easily understandable way to detect secure web connections.

B. Cookies

Cookies are something that makes our web browsing a stateful experience. They can be used to store data between page loads to keep track of the items added to shopping cart in an online store. Cookies can even carry on some information for next session, for example, remembering that the user left the website last time logged in so he/she does not have to type in login credentials again. But with them also comes along a new problem concerning privacy. Cookies can also be used to find out a user's browsing history. There is an option in major browsers to disable cookies, but some sites just will not work without them. Therefore cookies are a type of information relevant to web browsing which users could benefit to be more transparent.

The methods that inform the user about of the usage and the purpose of the cookies have evolved. But these methods still have some ground for improvement. An attempt to improve the methods is the introduction of a concept of informed consent. The concept states that the decisions made while browsing should be able to be made with enough information and voluntariness. The idea of "informed" includes the terms disclosure and comprehension, while the idea of consent can be analysed by terms voluntariness, coercion and agreement. When these conditions are met, the user should have enough information and power to decide whether or not to accept the cookies. Also a language, the P3P Protocol [6], for web sites to specify their privacy policies is discussed. Although P3P mechanism has not received widespread adoption, some browsers let users congure the type of sites they are willing to interact with. [7]

Having a notification pop up every time we set a cookie is cumbersome and time consuming. An attempt to fix this problem is a software called The Doppelganger [8]. Doppelganger is a Firefox extension that helps us work with cookies in a more convenient way by making an invisible parallel session of the users web browsing session. The description of this "mirroring" technique used in Doppelganger is described in the following way by the authors: "When Doppelganger encounters a domain in the user's browsing session for which it hasn't determined a cookie policy, it mirrors the user's web session in a hidden parallel session whose only difference is the cookies accepted and sent. We refer to this hidden parallel session as the fork window since it represents a forking of the browser state." Doppelganger reveals the fork window and asks the user to observe the differences when it detects that the main window and the fork window differ. The seconds technique used in Doppelganger is the Fix me -button, which is a rewind-and-playback mechanism.

The Doppelganger manages in the authors' test scenario to fix all third-party cookies. Third-party cookies are something that are not originating from the web page's domain itself, but rather from another domain. Many advertisement sites works this way. This can also be fixed by prompting the user whenever a third-party cookie is about to be set. But this is a very tiring and slowing process, since there may be several such cookies on one site.

C. Collusion extension

A similar extension to the prototype developed in this article for improving user awareness of web traffic is Collusion [10]. Collusion is an add-on for FireFox but also exists for Safari and Chrome. Collusion's goal is to inform us about the websites that get information about our browsing. The front-end is done with displaying a network graph that shows the sites we have visited. The visited sites are then connected to a network of other sites that have been contacted by the original web site.

Collusion also has blocking features for sites users want to disable. The add-on has a list of known tracking sites and they can be disabled with one click.

With the help of collusion users are easily able to see what web sites connect to tracking or advertisement sites. This raises awareness of the browsing and helps users to get more information about possible security risks.

D. Ghostery extension

Ghostery is also a Chrome extension that monitors the web sites during browsing [11]. It catches web request in the same manner as Kepler, with the help of the chrome.webRequest API. Ghostery is also updating in real-time and searches for predefined advertisement sites. Ghostery doesn't provide information about the security of the requests. Instead it uses a Ghostery cache where users are redirected to ghostery.com web site for detailed inspection about a web request that was caught by Ghostery.

E. Recx Security Analyzer

Recx Security Analyzer is a Chrome extension [12]. It gives the user security related information about a web page they are currently browsing by focusing on inspecting the HTTP security headers. It displays with checkmarks whether a connection is secure or not. Recx Security Analyzer also displays information about cookies on set by the current page and their security related attributes.

III. Kepler

The goal of Kepler is to raise user awareness of web browser security considerations. While understanding that no browser can be bulletproof and that the server side implementation affects also security, we decided to make a prototype that helps users make decisions about security themselves. Kepler is implemented as a Google Chrome extension. Chrome extensions were chosen as the development platform because of its good developer tools, comprehensive documentation and the fact that it is the most commonly used browser at the moment. To set up a Chrome extension the user has to modify a manifest file in JSON format to include the wanted name and version. After that it is all about writing the functionality in JavaScript and the presentation with HTML, JS and CSS.

An alternative implementation could have been done with a proxy server. A proxy server could be given a URL of a web site that the user wants to inspect. This could be implemented for example, with Node.js [13]. This way we could avoid the loading of any extensions by the user. The proxy could also be used to gather the data from many different sites at once by giving it a list of URLs. However, this would require Kepler to send information over the internet. This is always a security concern. Also, when requests are sent via proxy, the response is based on the information given by the proxy, not the user. This may affect for example, advertisements received as the location information between the proxy and the user may differ. This results in different kinds of data received from the server. By using a Chrome extension we are able to see in real-time all the requests that take place side by side with the web page we are interested in.

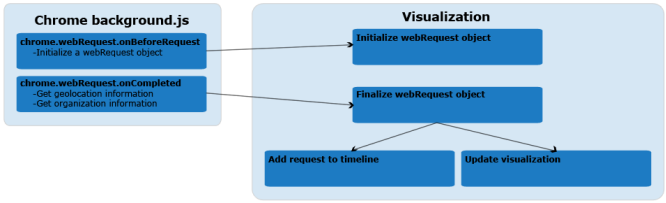

Fig. 1. The architecture of Kepler.

A. Architecture and back-end

Kepler uses several existing chrome APIs to gather information about the browser's network traffic [14]. The tabs API is used for reading Chrome window and tab information, such as active tab, URL of the tab and identification.

All the web traffic is gathered with the Google Chrome's webRequest API. Kepler utilizes two event listeners: the first listener triggers when a new request is made and the second listener triggers after the request is fully completed. This way the request duration can be calculated and different information gathered. After a web request is completed, a function call to the visualization page is made and the data gets visualized. The architecture of Kepler is visualized in Figure 1. When a web request is about to happen the onBeforeRequest listener gets called. This in turn initializes a custom made webRequest JavaScript object. After the web request is completed the rest of the available data is saved in the webRequest object. The following information is available from the completed event: requestId, url, method, frameId, parentFrameId, tabId, type, timeStamp, ip, fromCache, statusCode, responseHeaders and statusLine. By parsing this data from the completed web request we are able to determine the protocol (wheter it is using HTTP or HTTPS) and data type that we display in the Kepler visualization window. The extension also catches errors and displays them in red in the timeline.

Kepler has geolocation functionality for IP-addresses. Such were provided by MaxMind [15]. Their CSV files were converted to a JavaScript array with a small Python script. This array is included in the visualization page and used for looking up the country codes for IP-addresses. MaxMind provides the IP-addresses ranges in integer form. This made it easy to use a binary search algorithm to find the desired range for each IP-address. Therefore the geolocation function isn't too time consuming. Further, found IP-addresses are cached in a JavaScript array to avoid searching for the same IP-address twice. This is important because plenty of requests we see on a web page goes to the same IP-address.

B. Design and front-end

The front-end of Kepler consist of HTML, CSS and JavaScript. Kepler's main view is like any tab on Chrome as it can be pinned next to any other tab or opened as a new window. This makes it possible to browse on websites while having Kepler opened in another window to view the ongoing requests real-time.

To support extensibility and fast development, the front-end was implemented using open source frameworks like Twitter Bootstrap [16] and jQuery [17]. Twitter Bootstrap is used to layout the information in the visible extension window, and the DOM (Document Object Model) manipulation is implemented mainly using jQuery.

Kepler is designed to support different kinds of visualizations. The user can select the current visualization from a navigation bar located in the top of the extension window. The navigation bar contains also a dropdown list from which the user can filter which tabs' requests are included in the visualization.

Kepler is based on a two column design. On the left hand side rests what we call the timeline. It contains all the requests captured with the webRequest API with the current filtering. The requests are listed in descending order with the latest request on top. A request appears as green if it is using HTTPS and blue if it is a regular HTTP request. Left to the request is an icon that describes the request type. The other column acts as the main visualization area and contains all the elements of the currently selected visualization.

C. Statistics visualization

The main motivation behind the StatisticsVisualization is to provide how many HTTP and HTTPS requests there is generated and which country and organization they are originated while browsing or idling on specific website. The main visualization area categorizes the request to HTTP and HTTPS requests and further divides the them into subcategories by their type (image, script, etc.). The visualization also maintains and updates a table with countries which requests are originating, sorted by their frequency.

IV. Evaluation

We evaluated Kepler by browsing different web sites and observing the real-time information. Kepler displayed information about all the countries that were contacted from each web site and whether that information was encrypted or not. After filtering the requests and observing the details of the suspicious requests we were able to determine what kind of data is sent to third parties. The severity of the unwanted web requests is for the user to decide.

A. Use case: Evaluating the amount of information given to third parties

This is an example of a local news magazine's website. With the help of Kepler we are easily able to determine that requests happen to several countries and almost all of them via non secure channels. Figure 2 shows the results of loading the web page. An interesting observation here is the fact that there is 278 requests that take place when we load the page. While most of the requests are harmless, consisting of advertisements and social media functionality, it is easy to miss a harmful requests. A feasible way to find a harmful request is to filter the requests by their countries, starting from those which have leasts requests.

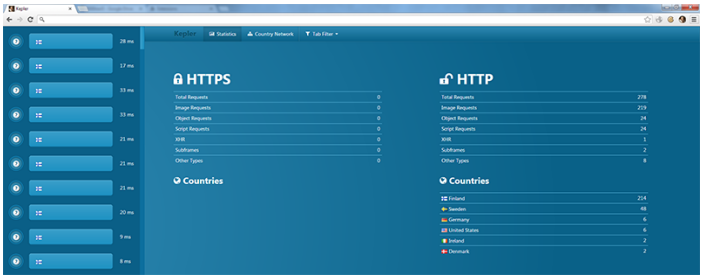

Fig. 2. Kepler displaying the requests from a local news magazine's web site.

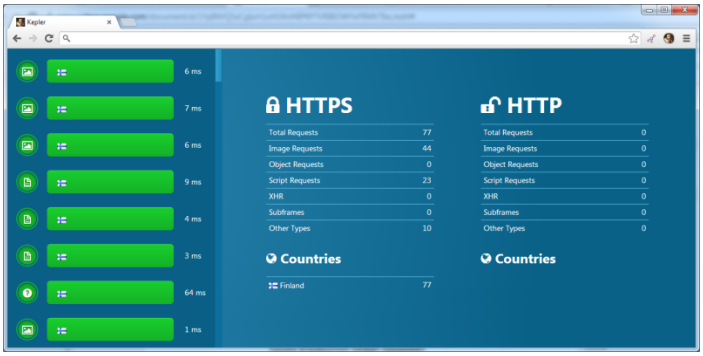

Fig. 3. Kepler status after loading an online banking web site.

B. Use case: Evaluating online banking services

Online banking must use encrypted channels for the communication between the browser and the web server. The risks with non-encrypted traffic could be devastating. This could lead to monetary theft and identity theft. In this test two Finnish online banking sites were evaluated. With Kepler we were able to see the differences in security between the banking sites. The first site showed no signs of suspicious traffic. Only one country was sent requests and all the requests went to the same domain. No tracking sites were contacted. With the help of Kepler we were able to conclude that no suspicious activity was detected. In Figure 3 we can see the results of loading the first banking web site.

With the second banking web site we are able to immediately see that the bank's web site is hosted in another country. This is not necessarily a problem, but what was interesting is that a second country was also sent a request and the fact that all the traffic happened unencrypted. When inspecting this request we could see from the URL of the request that it was a known tracking site that had been contacted. The same site was being contacted many times when logged into the bank account. The results from loading the second bank's web site can be seen in Figure 4. In Figure 5 is the details of the suspicious request made by the bank's web site.

In this test Kepler has helped the user to analyze the security of the different online banking web sites. This is a major step towards safer online banking and web browsing. Web traffic and details about web requests require technical knowledge about many things in order to make sense for most people.

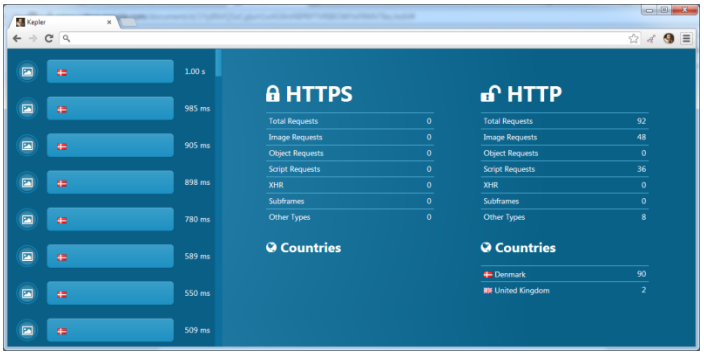

Fig. 4. The second online banking web site after loading.

Fig. 5. The details of the suspicious web request.

Kepler takes a step forward with bringing detailed information available easily and in human readable form. With Kepler it is enough for the user to understand the basics of web browsing to make conclusions about the security of their activities on the web. With a simple description of Kepler's functionality users should be able to improve their browsing security. In addition Kepler aims to please the eye by being as appealing as possible to achieve a bigger audience. Making browsing more lively and intriguing via nice front-end design helps us achieve the goal of a better web. This is something most of the browsers are lacking. A simple and dull lock-icon in the lower corner does not exactly raise interest or awareness.

Compared to similar extensions Kepler has a clear approach. The goal is to provide technical data to inexperienced users in a clear way and to display where the data is coming from. In this Kepler succeeded. By developing Kepler further we could adopt some of the features from other extension such as the Recx Security Analyzer. Recx Security Analyzer allows the user to click on the icons it displays for further information about the issue which is what Kepler will also provide in future versions. Similar applications have also focused a lot on listing the 3rd party tracking sites that get contacted while surfing the web. Kepler is lacking this feature and it could also be implemented in the future versions.

V. Conclusion and Further Work

The Kepler extension raises user awareness of web security. Cryptic information presented in a more human readable form makes Kepler a tool for everyone. With Kepler users are able to make conscious decisions about what web services to use. Users get a better understanding of how web pages work and what data is received and where the data is coming from.

At the moment Kepler only a prototype for giving out information. More visualizations could be added to give the user a better view of different kinds of information. Also evaluating the user experience with more testers would lead to improvements of the current user interface and the addition of new features.

References

[1] w3schools.com 2012, *December*10. *Browser Statistics* [Online]. Available: http://www.w3schools.com/browsers/browsers_stats.asp

[2] Google Developers (2012, December 10). *Chrome Developer Tools - Google Developers* [Online]. Available: https://developers.google.com/chrome-developer-tools/

[3] Mozilla (2012, December 10). *Firebug* [Online]. Available: http://getfirebug.com/

[4] B. Friedman et al., "Users' Conceptions of Web Security: A Comparative Study," in *CHI 2002*, Minneapolis, MN, 2002, pp. 746-747.

[5] T. Whalen and K. M. Inkpen, "Gathering Evidence: Use of Visual Security Cues in Web Browsers," in *Proceedings of Graphics Interface 2005*, Victoria, BC, Canada, 2005, pp. 137-144.

[6] L. Cranor and R. Wenning, (2012, December 10). *P3P: The Platform for Privacy Preferences* [Online]. Available: http://www.w3.org/P3P/

[7] L. I. Millett et al., "Cookies and Web Browser Design: Toward Realizing Informed Consent Online," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Seattle, WA, 2001, pp. 46-52.

[8] U. Shankar and C. Karlof, "Doppelganger: Better Browser Privacy Without the Bother," in *Proceedings of the 13th ACM conference on Computer and communications security*, Alexandria, VA, 2006, pp. 154-167

[9] G. Aggarwal et al., "An Analysis of Private Browsing Modes in Modern Browsers," in *Proceedings of the 19th USENIX conference on Security*, Washington, DC, 2001, pp. 46-52.

[10] Disconnect, Inc. (2012, December 10). *Do not track — Online privacy tools — Disconnect* [Online]. Available: https://disconnect.me/tools

[11] Evidon, Inc. (2013, February 27). *Ghostery* [Online]. Available: http://www.ghostery.com/

[12] Recx Ltd. (2013, February 27). *Google Chrome Plugins — Recx* [Online]. Available: http://www.recx.co.uk/products/chromeplugin.php

[13] Joyent, Inc. (2012, December 10). *node.js* [Online]. Available: http://nodejs.org/

[14] Google (2012, December 11). *chrome.* APIs - Google Chrome* [Online]. Available: https://developer.chrome.com/extensions/api_index.html

[15] MaxMind, Inc. (2012, December 10). *MaxMind - IP Geolocation and Online Fraud Prevention* [Online]. Available: http://www.maxmind.com/en/home

[16] M. Otto (2012, December 10). *Twitter Bootstrap* [Online]. Available: http://twitter.github.com/bootstrap/

[17] The jQuery Foundation (2012, December 10). *jQuery: The Write Less, Do More, JavaScript Library* [Online]. Available http://jquery.com