

Midterm Report for ECE1776 Project

by

Robert Ma
990342054
robertma@gmail.com

Dmitry Denisenko
980432520
dmitry.denisenko@utoronto.ca

October 17, 2006

User Interface

In this project, a Mozilla web browser plug-in is developed to detect phishing websites and warn users when they are identified. The design of our plug-in will try to be clean and simple but at the same time address the common problems found in existing anti-phishing solutions. First, to minimize the effect that can be made to users' web browsing experience, this plug-in will be lightweight in processing and should take up as little space as possible. In our design, the plug-in will reside in the status bar of the web browser window, as shown in Figure 1.

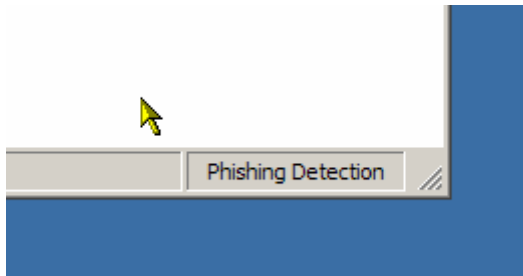


Figure 1 - Phishing Detection Plug-In reside in the status bar of the browser

While located in the status bar, the plug-in will not take up any space of the main web window. Users will have options to turn on and off the phishing detection in our plug-in by accessing the context menu of the plug-in status bar as shown in Figure 2. In order to be lightweight, currently our plug-in will only do a phishing analysis when there is a domain-name change in the URL of the current browser.

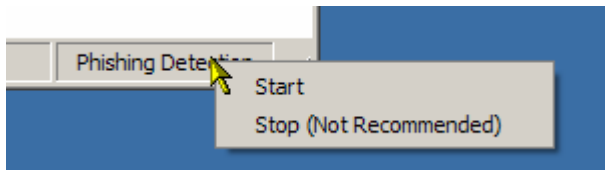


Figure 2 - Context Menu of Phishing Detection Plug-In

There are many efforts spent in developing tools that prevent Internet users from falling into phishing traps. However, most solutions are ineffective against phishing attacks because their user-interfaces were not well designed. In our project, our primary focus of the UI design is to tackle the deficiencies found in other similar tools.

Most anti-phishing tools are located in a peripheral area in the browser hence warning indicators in them become insignificant when compared to the web content. Furthermore, security is rarely user's main focus when browsing web sites. It's unlikely that a user will continuously pay attention to these indicators. Therefore, when the current website is determined to be suspicious, we will pop-up a dialog box, which interrupts the current task the user is trying to accomplish as shown in Figure 3. As a result, it would be impossible that users would miss a potential warning and carelessly fall into traps of phishing websites.

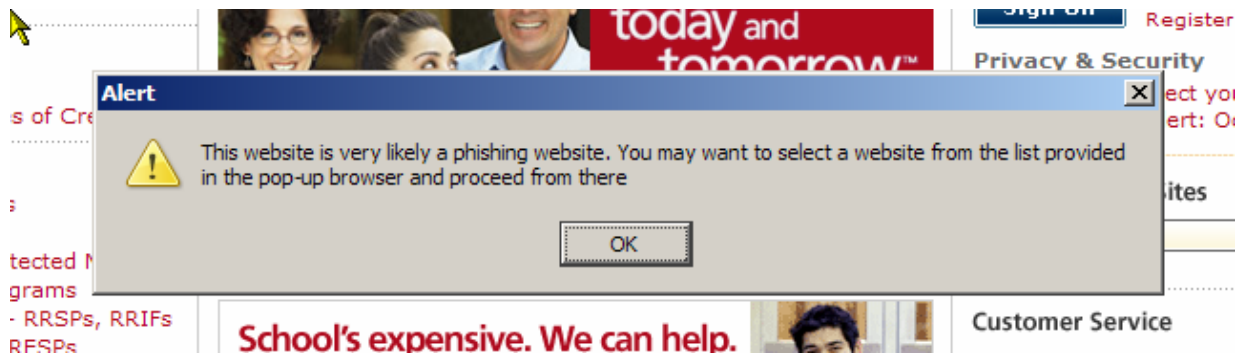


Figure 3 - Pop-up dialog suggests to user that the current web site is a phishing website

When user presses “OK” in this dialog, the plug-in will bring up a browser window containing the search result we obtain from Google, as shown in Figure 4. Users can then select websites from the search result.

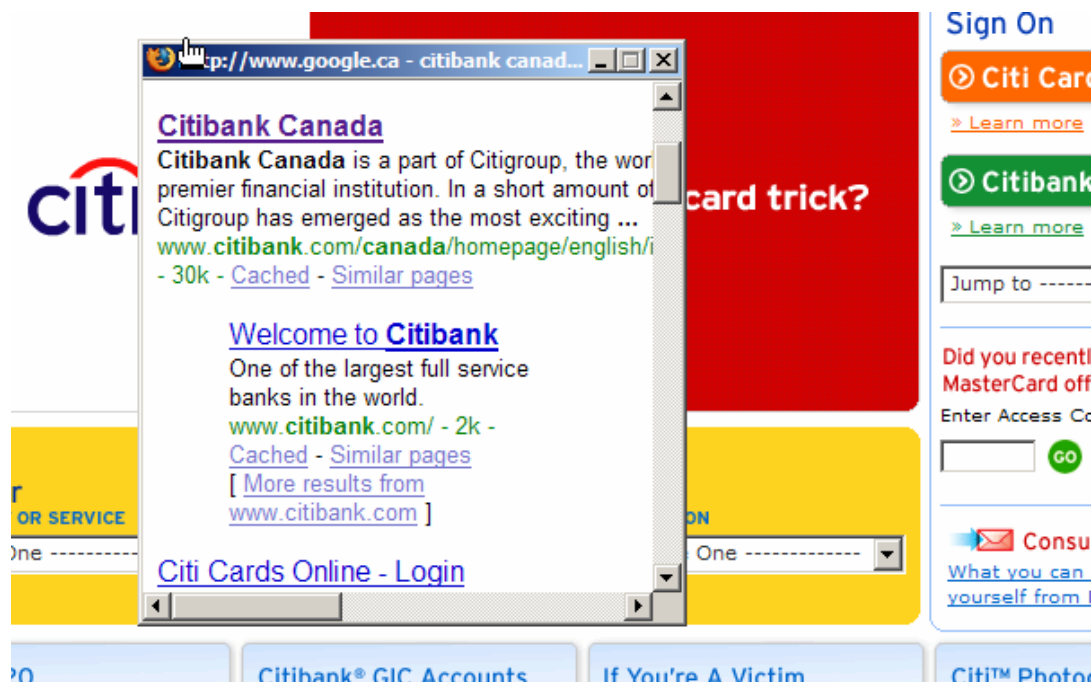


Figure 4 - Pop-up browser window with Google search results

By providing search results in the pop-up browser, our plug-in solves the problem that most indicators usually show something is wrong and advise user not to proceed, but they do not suggest good alternatives. This may encourage users to risk submitting their information anyway, since they don't see any other way to accomplish their goal.

Automatic Query Generation

In this section we describe how we form a query to Google based on loaded web page's content. We first explain what assumptions we are making when building the query. Then, we describe what we have learned and implemented so far.

The main assumption underlying the effectiveness of our plug-in is that of similarity between the phishing site and the real site the phisher is trying to imitate. The user is responsible for “enforcing” this assump-

tion, i.e. we are assuming that if the user encounters a site that looks very dissimilar to the real site, he will realize the illicit nature of the site and leave the site immediately.

Since we are relying on the user to verify the general appearance of the site, we can only query the *visible* portions of the site. The invisible parts, such as metadata and embedded scripts, can be easily changed without user's notice and our plug-in, if it relied on such parts, would be thwarted. We also found that Google does not give good results if queried with keywords from metadata section.

We found that a query that contains site's title works extremely well. Unfortunately, not all sites have enough information in their title to uniquely identify them. For example, the login page for CIBC is titled "Online Banking." Therefore, our query should contain other words from the page. In particular, it seems necessary to find the company name inside the webpage.

With that in mind, we currently have the following implementation for the query generator:

1. Extract the title of the webpage. Let's call it T.
2. For capitalized words that follow a period or a '>', change the capital letter to lowercase.
3. Remove all scripts, html tag, links, and link names, leaving only visible text.
4. Search the remaining text for words containing a capital letter in the hope of getting the company name.
5. From the list of selected words, remove words from a built-in list of common words, such "agreement," "contact," "we," "yes," and "customer." Let's call the remaining words w_1, \dots, w_n .
6. Form the following query: "T"+ $w_1+\dots+w_n$. Note that the quotes are used around the title to tell Google to search for web pages where the title words appear together and in the same order.

Step 2 is used to cut down on the number of words in the query by de-capitalizing starts of sentences. There is a chance that the company name will appear at the start of the sentence. However, we find that the company name appears so many times within the page that there is still a good chance of finding it in a middle of a sentence.

In step 5, we eliminate common words for the following reason: most login sites have them and they skew Google results towards some random pages. Google seems to work best when it is queried by a few specific words, rather than by the same words followed by a number of common ones.

Quality Of Results Evaluation

Doing live tests on real phishing sites is difficult due to short life-span of such sites (the average life-span is reported to be around 3 days). However, to fairly judge effectiveness of query techniques, we have to compare the results they generate on exactly the same input set. Since the plug-in development will take about two months, efficient query technique evaluation on live phishing sites is impossible.

To circumvent this problem, we use our visual similarity assumption: since the phishing site is assumed to look very similar to the real site, we might as well use the real site for testing purposes. This is not a perfect approach but it should be sufficient.

Following this idea, we compiled a list of nine test cases: login sites for major financial institutions and retailers. Here it is:

Amazon.ca
Dell
PayPal

CIBC
eBay
Royal Bank

Citibank
ING Direct
Wells Fargo

Note that eBay and PayPal are two most-targeted sites for phishing attacks.

For each query technique, we run the plug-in through all nine login sites and see how many queries point to that site as the first few hits in the Google result set. For faster testing, the sites were saved as text-only html pages on a local hard-drive and the plug-in was run as a JavaScript in shell mode. Only actual Google query evaluation was done using the network.

We found that no matter how good the query is, we almost never found the exact page we were looking for. We judged a query to be successful if we get the same domain as the input page among the first few results returned by Google.

With our current implementation, for 7 out of 9 sites we can find the right domain name.

Future Work

Our query-generating module is currently not integrated with the GUI, it is run as a stand-alone JavaScript shell script on locally saved pages. Integrating it with the GUI and shaking out integration of the GUI with the Firefox browser will need to get done.

Even though our query generator is not expected to take any noticeable time, querying Google every time a user changes a web page is very network-intensive. Furthermore, such frequent querying can result in Google blocking our user from further queries.

We currently have not implemented, but plan to use, the following time-saving techniques:

- Do not do anything if a site doesn't have a password field (specified as `<input type="password">` in HTML).
- If the location of the site contains an IP address (and 1/3 of phishing sites don't have domain names), there is no point in going to Google -- the site is most likely illegitimate.
- If there is no encryption on the site (and it asks for a password), the site clearly doesn't care about the confidentiality of user data.

The query-generator is not working perfectly. It will need to be tuned better. In particular, the following areas will be explored:

- Common word removal: which ones?
- (De-)Capitalization usefulness: if we can remove common words better, there might not be a need to do fancy de-capitalization that can hide important words
- Many sites seem to mention their top-level addresses in the visible text (E.g. "Go to www.dell.com for more info"). Using these addresses intelligently is a challenge.
- Analyzing linking structure of the site. We found that some phishing sites would only place the actual login page on the rogue server, linking to the real site for everything else (such as the typical "Privacy Policy," "Contact Us," "About," and others). Using domain names from these links might be useful.
- We found that all sites use pictures to show a company's logo. Since graphics stands out much more than text (and hence is paramount in the deception created by the phishing site), if we can find a way to use that graphics, it would greatly enhance the effectiveness of our plug-in.

- Many international companies have separate domains for each country. For example, there is ebay.com and ebay.ca. Automatically telling if both ebay.com and ebay.ca belong the same company is a challenge. Using information from page's certificate (if there is one) might allow us to link ownership of two sites.

Extensive testing of the plug-in on real phishing sites and other legitimate sites is also needed.