Computer Science and Engineering

University of Nevada, Reno

Creating a malicious attack to steal users facebook cookies and then post on their behalf

Enzo Arata, Zachary McIlwain,

Instructors: Iman Vakilinia

May 10th, 2019

**Our Method:**

In order to solve the issue of stealing a users facebook cookies and then using them ourselves we used a few different strategies. To start we simplified our code into two separate scripts, one of the scripts would be designed to gather the targets cookies and save them. The other scripts intention was to use the stolen cookies and than login to the targets facebook account and finally to post on to their facebook. We then created a bat file which can be launched to start both scripts on a computer. The idea behind this was we would be able to put the bat file on a flashdrive, plug it into a targets computer and launch the malicious scripts.

To create the malicious scripts we used the scripting language AutoHotkey. The way the first script works is that it launches google chrome on the target's computer and sets it to fullscreen. It then sends the user to the extension website for “EditThisCookie”, the script then waits a couple of seconds for the web page to load. After the wait period finishes it installed the extension by clicking on various locations in the google chrome window. The script then waits again for the extension to install, after that the script has google chrome load facebook. Again the script has to wait for the web page to load. Once the wait has concluded, the script clicks on various locations in the google chrome window so that it can gather cookie information from the extension “editThisCookie”. The script then saves the cookie data to the clipboard, it then will open a notepad file from within the flash drives location. Once the notepad file is open it pastes the contents of the clipboard into it(the users cookies for facebook) and then saves the notepad and closes it out. Now we have gathered the cookies for a users facebook login and have them saved onto our flashdrive.

The second script is designed to only be able to work if the first script was successful, basically if you don't have cookies saved into the notepad file on the flashdrive it wont work. This script starts by launching the notepad file with the saved cookies and saving the cookies to the clipboard. After that the script launches google chrome and loads into facebook and then waits for a the webpage to load. Once the web page has loaded the script clicks on various locations on the webpage and pastes the contents of the clipboard(Stolen facebook cookies) into “Editthiscookie” and then refreshed the page. Once the page is loaded the script clicks on locations of the window and posts to the users facebook “Hacked by Cookie Stealer”. This concludes what our project does. Overall we are happy with the results of our project, we enjoyed working on it as well as testing it.

**What we learned:**

It took a lot for us to find a working solution for stealing cookie data because it seems like a lot of the old techniques for doing this no longer work and caused us many issues. For example some of the options that we saw were stealing cookie data over the network with wireshark, injection of a script on facebook and using something like an alert to trigger the stealing of the cookies, or we even tried to use headless chrome so the user could potentially even not know the attack was running but we ran into issues with all of these techniques. First when using wireshark to steal you have to check if the site uses HTTPS which will encrypt the cookies when they are sent over the internet and unfortunately Facebook was our target and it uses HTTPS. Technically using wireshark would still be viable but only if the site is not protected. As for the other options we tried to test the ability of the options by manually testing it and it seems that the function call to return the cookies in both cases seems to only return non-https cookies and those were pretty much useless for us.

In the end we decided to use autohotkey as it was a program that we were somewhat familiar with. However this does not mean we didn’t run into to many issues along the way. Autohotkey has some libraries that give it access to selecting web elements but these only exist for internet explorer and our goal was to steal cookies from chrome instead. To solve this issue we had to use coords to be able to click elements on the screen but we soon found out that these screen coords are not the same for every computer. The things that cause screen coords to change can be the screen resolution chromes zoom and even windows scaling which was one that had us stumped for the longest time. Currently our script only works on computers that are 1080p resolution and at 100% windows scaling but it would be possible to add scripts that could change the resolution but that would increase the scope of the project beyond what we had time for. In the end we learned a lot about the limitations of web scraping using autohotkey and about the improvements to network/web security to make an exploit like this much more challenging to pull off.