Author: rupp,scd
Project:super-Evercookies
Date: 10/20/12

Super-Evercookies

Index

1. Introduction

This document describes the modification of the Evercookies Javascript API. We have two ways to improve the robustness of the Evercookie API:

1) finding new way for localstorage
2) modifying the existing javascript code to make the API less exposed to removal

2. Additional Ways to perform local storage on a machine

2.1. QuickTime Local Storage (and Real Player/Windows Media Player)

- ***THIS METHOD MAY NOT BE USED AS ADDITIONAL EVERCOOKIE LOCAL STORAGE***
- ***DO NOT REQUIRE USER INTERACTION***

QT plugin is using special place in hard drive to store cached movies.

"This file type seems to have been introduced with QuickTime Player 7.x. The file is normally stored in any sub-directories of the following location:

C:\Documents and Settings\<username>\Local Settings\Application Data\Apple Computer\QuickTime\downloads

E.G.:

C:\Documents and Settings\Administrator\Local Settings\Application Data\Apple Computer\QuickTime\downloads\08\02\82580278-44f0f184-c8c435b8-526b4f79.qtch"

A movie could be stored on this emplacement with storage data embedded in the QT format then retrieved later by the plugin.

Even if technically possible this may be very slow due to QT plugin slowness.

Besides it may be very complicated to implement.

2.2. Adobe Reader plugin Local Storage

==> this is in fact the flash extended storage and this mechanism is been used already by evercookies

2.3. IE User Data Behavior

==> used by evercookies

2.4. Form Autocompletion/Autofill

- ***THIS METHOD MAY NOT BE USED AS ADDITIONAL EVERCOOKIE LOCAL STORAGE***
- ***REQUIRE USER INTERACTION***

The underlying idea is to use the browser autocompletion / autofill system to store data locally.

The autocomplete data are ciphered and stored on the disk. We cannot directly access the storage by javascript but we want to use the following idea:

1) we trigger keys and/or mouse events to select a form, having the browser autocomplete

suggestions being listed

2) then sending a keydown event we select the data we want to read and fill the form with it
3) then we just have to read the content of the form in javascript.

http://stackoverflow.com/questions/9356689/jquery-autocomplete-get-value

The HTML5 autocomplete appears not to work with present browsers (too new) even with chrome21 which is the best HTML5 compliant browser at the moment.

We try an other strategy as shown partially from the previous articles and demos : try to select the form and get the autocomplete by a keydown event , all this down in javascript.

```
function simulateKeyPress(character) {
 jQuery.event.trigger({ type : 'keypress', which : character.charCodeAt(0) });
}
```

At first glance, this may work , keydown event will be triggered by javascript so this should select the autocompletion value and then we may use this as additional local storage, using the autocompletion local storage in browsers.

This is a demo how to simulate keys events : http://bililite.com/blog/2011/01/23/improved-sendkeys/#demo

The idea is to create a hidden test field (or any invisible text field will do) then fill it with the content of the cookie. at next visit the browser will have kept this cookie data for autocompletion and we will simulate a select of the form and a keydown event so that the hidden/invisible text field is filled with the content of the cookies, the only thing left to be done is to retrieve the content of the text field.

We create a demo at http://192.162.69.236/tmp/test1/index2.html using the key press simulation. it sends a 'a' key in a text field. can we get autocompletion for this field and can we simulate pressing the keydown so we fill the field with the autocompleted data?

We get no good result with sending keydown events by javascript ,the keydown event does not move the autocomplete nor does any keypress event spawn this menu. Events must be sampled by the browser at a superior level.

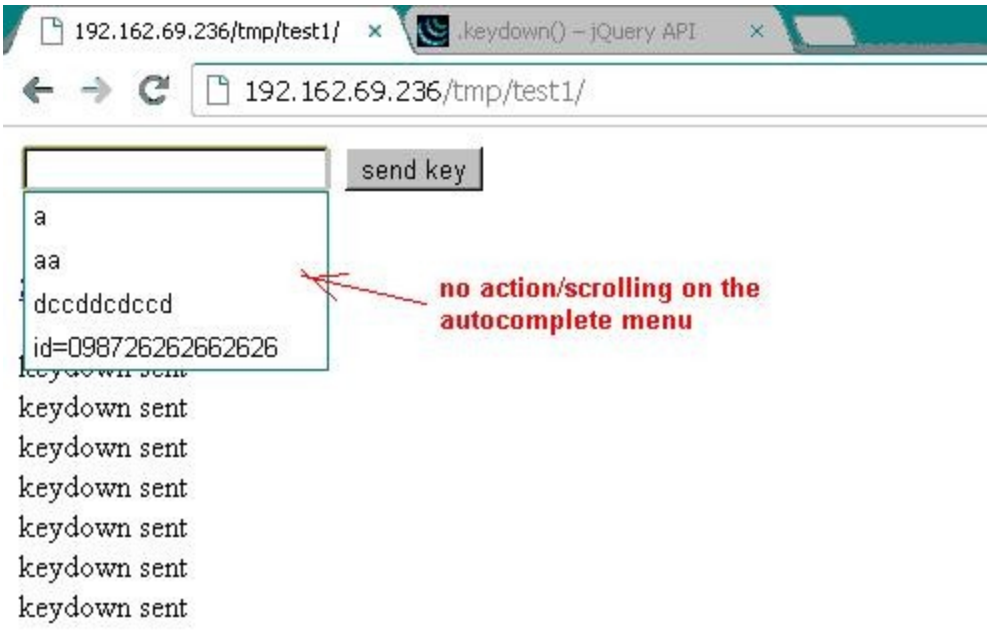
javascript code to triggers keydown/keypress events:

```
function sendArrowDown() {
 e = jQuery.Event("keydown");
e.which=40;
```

```
e.KeyCode=40;
$(t1).trigger(e);

$(t1).keydown();

....
```



it seems so far that the browser autocomplete menu is unreachable by javascript action. Therefore we may not be able to use this as additional storage.

2.5. Local vcard

- ***THIS METHOD MAY NOT BE USED AS ADDITIONAL EVERCOOKIE LOCAL STORAGE***
- ***REQUIRE USER INTERACTION***

Using Vcard as a new way to store local information:

"When the user inputs information into a form and submits it, Internet Explorer creates a vCard. When you input a piece of information like a username or an e-mail address, the browser creates files on your system called 'VCARD_NAME' and 'VCARD_E-MAIL'. These cards are created after the entire form is filled and submitted. After this, whenever you fill a form that has attributes name or e-mail, the vCard will suggest the name and e-mail for you. If you do not submit a form with these attributes, no

vCard will be created on your system.”

Unfortunately if we can force the browser suggesting the existing vcard entries, these data will appear only in a suggestion menu and the form won't be filled without the user action.

2.6. Javascript File System API

- ***THIS METHOD MAY BE USED AS ADDITIONAL EVERCOOKIE LOCAL STORAGE***
- ***REQUIRE USER INTERACTION***

Looking at FILE javascript api .This feature has not yet been used by evercookies API

http://www.htmlgoodies.com/beyond/javascript/read-text-files-using-the-javascript-filereader.html#fbid=_Xq2rX9dxkV

This is an example of a modified version of the JS file system API :

http://html5-demos.appspot.com/static/filesystem/filer.js/demos/index.html

This requires user interaction(click on button or drag and drop)

If we could get automatically a file reference explicitly without user interaction we could use that technique to add a new type of local storage. Could we get an explicit file reference to an image in the cache for example or any kind of local file ? If we could we may use that file as a local storage to store information about the user .

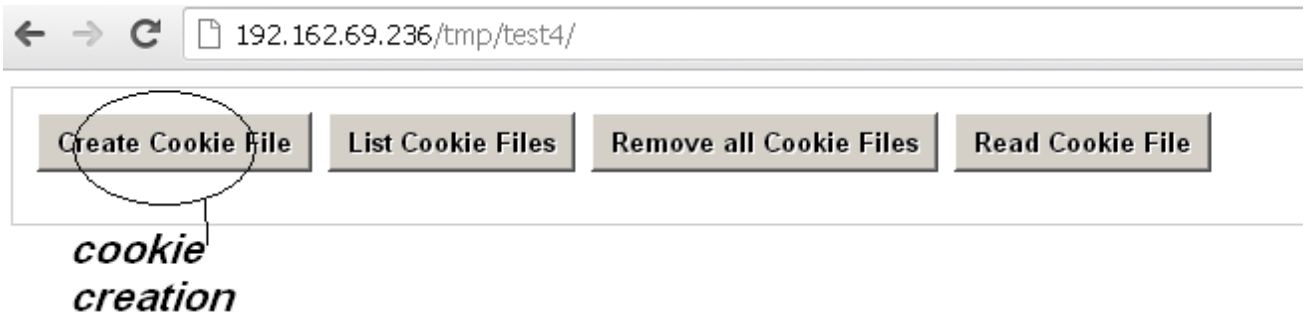
There are script that read and write *any* files which are been selected by the user (by listening the global file selection events). That mean that as soon as a browser user drag and drop or use a select file button for some files, these files may be immediately used as local storage. This is anyway not of interest since we still need user interaction.

2.7. Google Javascript FileWriter API

- ***THIS METHOD MAY BE USED AS ADDITIONAL EVERCOOKIE LOCAL STORAGE***
- ***DO NOT REQUIRE USER INTERACTION***

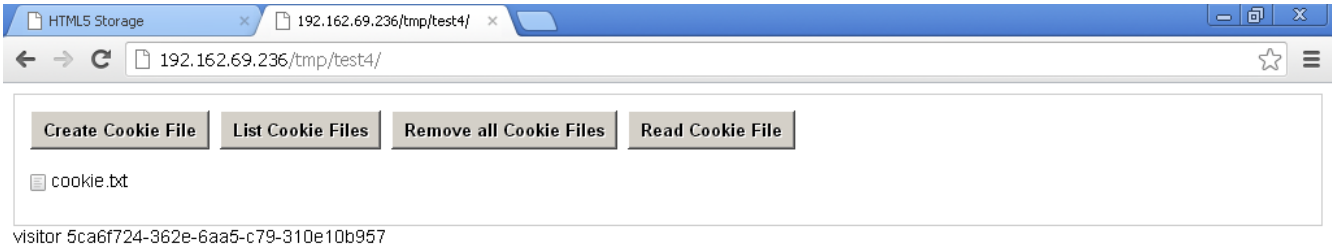
The FileWriter API provides a new way to locally store data .(not being used by evercookies)

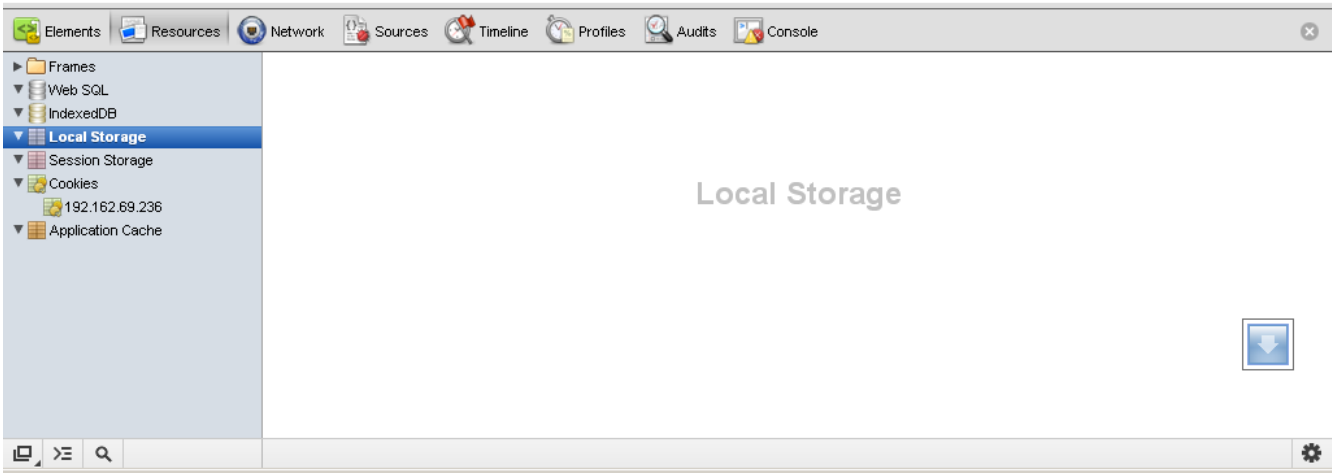
We write a UUID to a file in the chrooted filesystem we have previously requested.We can name this file as we want . We can also create folders and tree structure (Google Filesystem Disk Quota will apply)



visitor 5ca6f724-362e-6aa5-c79-310e10b957

We can check that no local/global storage / WebSQL / indexedDb or cookies have been created. The file belongs to a dedicated filesystem.





2.8. BruteForcing Cache Data

==>Technique already used by evercookie API

2.9. Local Resources (JS execution ,...)

- ***THIS METHOD MAY NOT BE USED AS ADDITIONAL EVERCOOKIE LOCAL STORAGE***
- ***REQUIRE USER INTERACTION***

We try to launch javascript in a local context (ex: file:// protocol) so we can read and write files using several techniques. The loading of javascript in local context must be initiated by the user and not by a remote website, this is a strong security requirement for the browsers, thus even trying all sort of ways to bypass this, it will always fail.

2.10. Force Download/ Download History

We can force download of data to the browser without any user interaction using the following script:

```
window.location.assign("http://server//pathtofile");
```

This will force-download a file without user interaction in several browsers.

Can we use this mechanism to store local data?

Windows.history.length is a publicly accessible method. This returns the amount of history entries in the browser.

If we can by automatic action, increase this number to a 100bit number , we could code the cookie value in the last 90 first bits for example, assuming that the user will not change this value significantly and that the last 90 bits will stay unchanged.

A 90 bit number will be around 2^90 =1237940039285380274899124224 which would represent several billions of zillions history entries! So unfortunately it seems hardly possible to do this.

2.11. Grease Monkey localstorage

- ***THIS METHOD MAY BE USED AS ADDITIONAL EVERCOOKIE LOCAL STORAGE***
- ***REQUIRE USER INTERACTION***

Greasemonkey is a popular script engine for firefox (and eventually with other navigators such as Chromium)

The function GM_setValue , according to http://wiki.greasespot.net/GM_setValue allows data to be stored on the hard drive by greasemonkey . of course that will only work with browsers that have greasemonkey installed or the browser having native support for a greasemonkey script.(.user.js)

It is unclear how many browser will support the greasemonkey localstorage script even if they do not have greasemonkey installed.

greasemonkey will store local storage in Firefox preference store (and in simlar location for the other browsers) which is completely independent of the browser cache. wo that it will resist to a cleaning.

But greasemonkey scripts installation and/or execution require user interaction.

2.12.Javascript Protocol Handling/ContentHandling

- *THIS METHOD MAY BE USED AS ADDITIONAL EVERCOOKIE LOCAL STORAGE*
- *REQUIRE USER INTERACTION*

The idea behind using protocol registration is to register an imaginary protocol which will represent the cookie data, like:

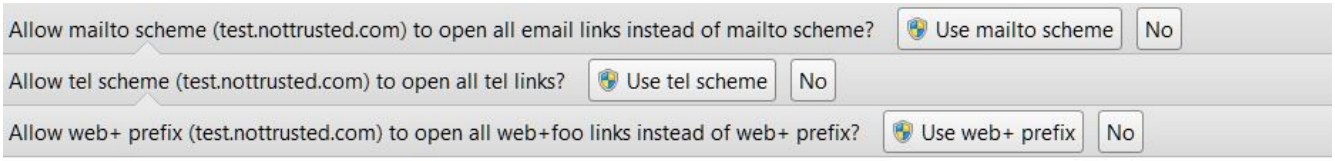
```
navigator.registerProtocolHandler("web+qwertyuiop",
 "http://192.162.69.236/tmp/test5",
 "Cookie");
```

where qwertyuiop is a data cookie.

It would be therefore possible to do a brute force attack on the protocol registered or to get the value of the 'Cookie' protocol.

Alternatively the cookie data may be passed as the name of the imaginary protocol.

Unfortunately registering such an application handler will trigger user interaction:



Same thing happens for content handling.

2.13.ActiveX/Vbscript

- *THIS METHOD MAY BE USED AS ADDITIONAL EVERCOOKIE LOCAL STORAGE*
- *REQUIRE (STRONG) USER INTERACTION*

http://192.162.69.236/tmp/test6/

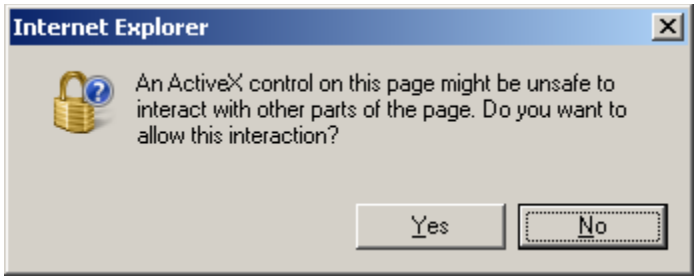
We use the Vbscript language as an alternative to javascript. (will work only with IE therefore)

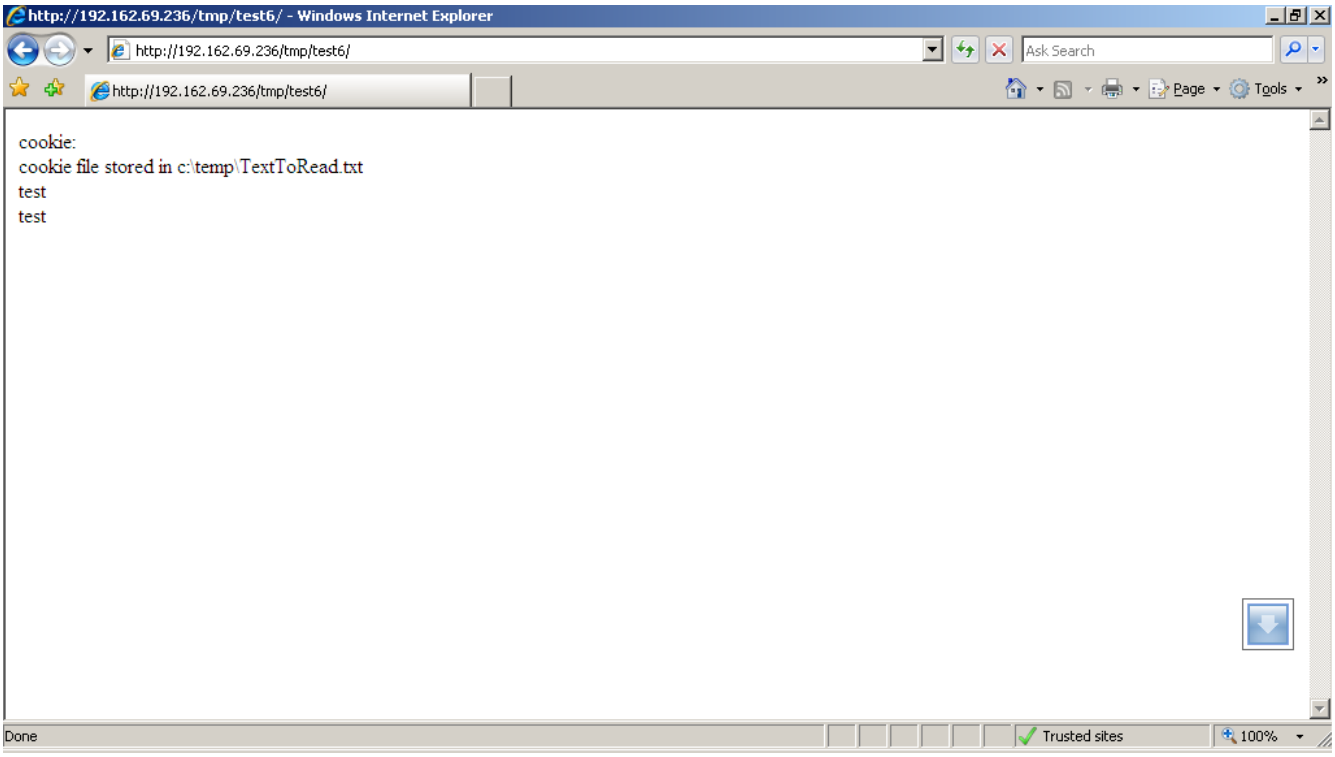
We can have the Vbscript writing and reading local files but :

1) this is only possible if the security settings of IE allows activeX on this page
2) in all cases user is beeing warned and user acceptance is required for the script to be executed.

```
<html>
<script language="vbscript">
Option Explicit
Const conForReading = 1
'Declare variables
Dim objFSO, objReadFile,objTextFile, contents
Set objFSO = CreateObject("Scripting.FileSystemObject")
Const ForAppending = 8
Set objTextFile = objFSO.OpenTextFile("C:\Temp\TextToRead.txt", 8, True)
' Writes a string every time you run this VBScript
objTextFile.WriteLine("test<br>")
objTextFile.Close
'Set Objects
Set objReadFile = objFSO.OpenTextFile("C:\Temp\TextToRead.txt", 1, False)
'Read file contents
contents = "cookie:"+objReadFile.ReadAll
'Close file
objReadFile.close
'Display results
document.write(contents)
'Cleanup objects
Set objFSO = Nothing
Set objReadFile = Nothing


</script>
</html>
```



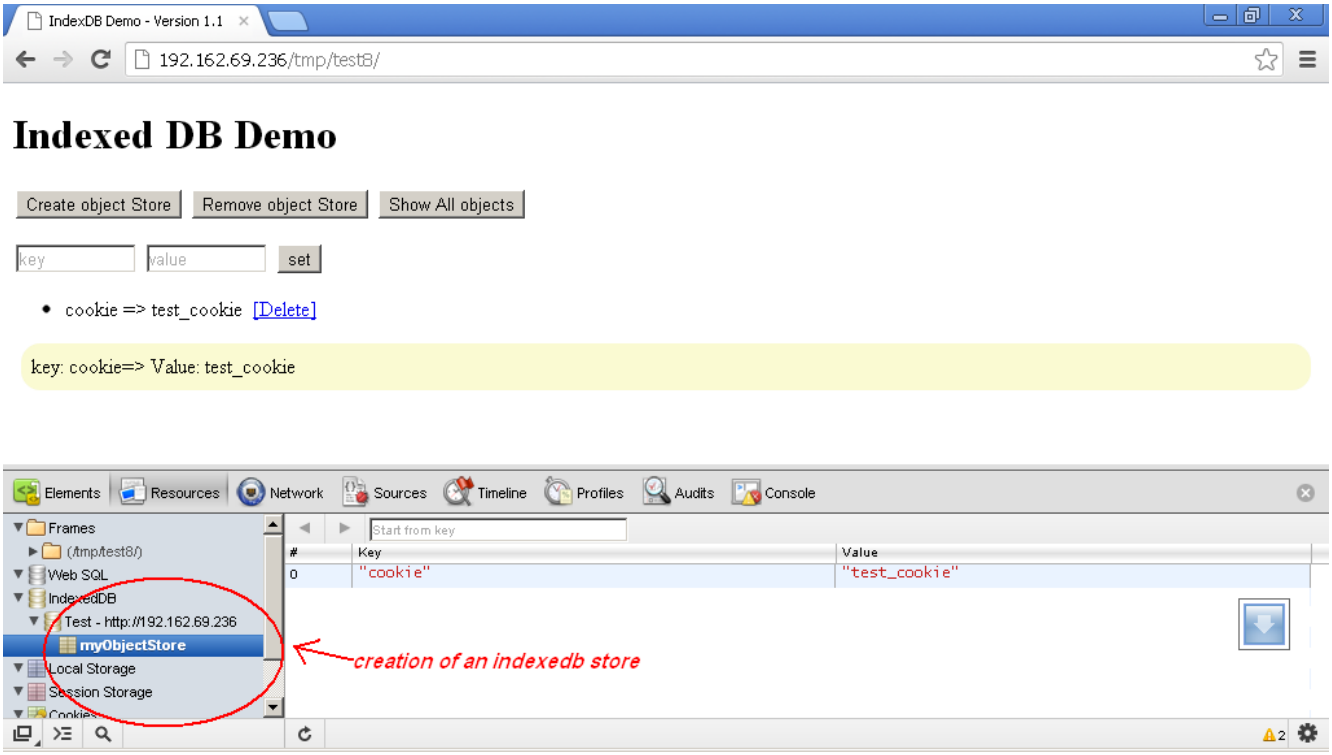


2.14. indexedDb

- ***THIS METHOD MAY BE USED AS ADDITIONAL EVERCOOKIE LOCAL STORAGE***
- ***DO NOT REQUIRE USER INTERACTION***

- indexedDb is a replacement specification for webSQL. The indexeddb storage is different from the local storage, global storage and Websql storage.

http://192.162.69.236/tmp/test8/



2.15. Google WebKitStartActivity/Webstore

- *THIS METHOD MAY BE USED AS ADDITIONAL EVERCOOKIE LOCAL STORAGE*
- *REQUIRE USER INTERACTION*

This technique consists in using the google-specific APIS.

1) WebKitStartActivity

This allow to connect an application service provider to the web page in the context of 'web intents'.Unfortunately User Interaction is required to get the connection to App service provider.

Other techniques for similar google APIs requires user interaction.

2.16. Window.Name localstorage Hack

- *THIS METHOD MAY NOT BE USED AS ADDITIONAL EVERCOOKIE LOCAL STORAGE*
- *DO NOT REQUIRE USER INTERACTION*

The window.name variable is NOT read-only. It will survive to a refresh of the page. Therefore a string may be stored in that variable. But the variable will not survive a Browser restart, this is only a session-based way to store data.

2.17. Google Gear LocalStorage

Google gear provides a local storage. We will be using dojo.storage in order to access this feature when available.

http://192.162.69.236/tmp/test7/

3. Evercookies Modifications

3.1. Ofuscation