Analysis of setting cookies for third party websites in different browsers

Tuesday January 20, 2015

tl;dr

This post discusses the results from our research into the ability of third party websites setting cookies for first party websites across different web browsers. The ability to be able to set cookies in this manner not only facilitates tracking but also opens up other opportunities and avenues of attack.

Introduction

Cookies are one of the most common sources user supplied input for web applications, with browsers sending their latest values to their relevant server with every normal HTTP request. Like any other forms of user input, cookies can be used to attempt a number of different classes of attack including cross-site scripting (XSS) and SQL Injection (SQLi).

Exploiting client side issues (such as an XSS) via cookie parameters is more difficult than using URL parameters (QueryString) or body of POST requests as an attacker needs to find a way to set a malicious cookie value in victims' browsers in the first place.

Risks posed an XSS issue when an attacker can set a malicious cookie for victims are less than a normal stored XSS but higher than a reflected one.

This type of attack is normally possible by using another reflected XSS (in the same domain or another subdomain) or by exploiting a CSRF issue in a page that that stores users supplied input in a cookie.

It should be noted that it is not uncommon for websites to keep users' supplied data that is received from GET or POST requests in their cookies.

For instance, "Remember me" or "Keep me signed in" feature of a login page is a common place to see this behaviour in which the provided username might be kept in the cookies even when the provided username is incorrect.

In this research different methods by which it is possible to set cookies via a CSRF style attack through an IFRAME tag have been reviewed.

In order to send an HTTP request to a target website via an IFRAME, a list of HTML/SVG tags that can accept URL in their attributes were extracted from the following resources:

http://www.w3.org/TR/html4/index/attributes.html (http://www.w3.org/TR/html4/index/attributes.html)

http://www.w3.org/TR/html5/index.html#attributes-1 (http://www.w3.org/TR/html5/index.html#attributes-1)

http://www.w3.org/TR/SVG/attindex.html (http://www.w3.org/TR/SVG/attindex.html)

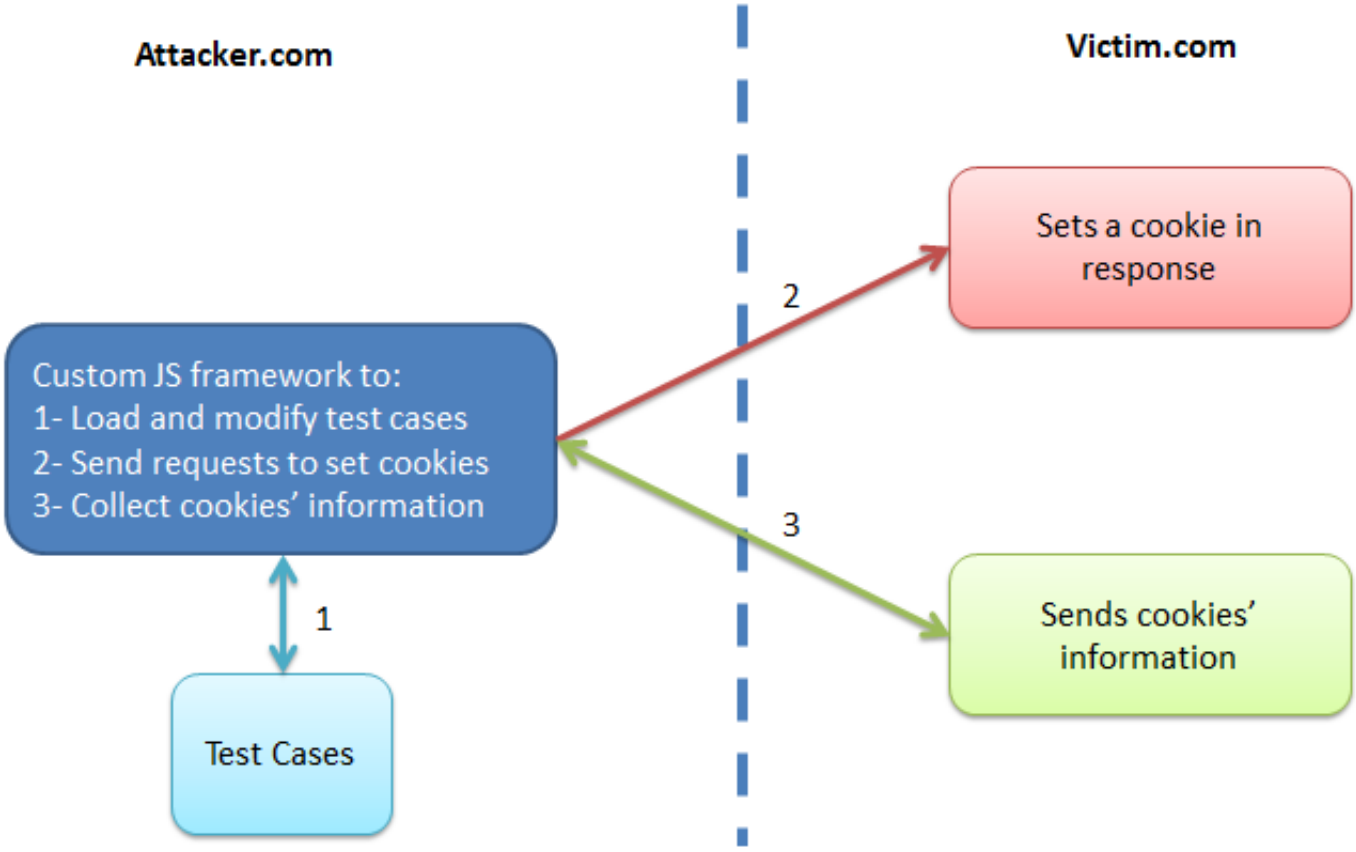
A testing framework in JavaScript was also designed and implemented to test different cases automatically. This framework operates in the following way:

First it opens HTML test cases which are designed for setting cookies.

Then it changes the required parameters such as the target URL in the HTML codes and loads them in different IFRAME tags.

Finally, it checks the target website to see whether the cookie parameters were set successfully or not on the target website.

These steps are shown in the image below.



The test website was also protected with the following defensive HTTP security headers to remove trivially exploitable scenarios:

```
X-Frame-Options: DENY
X-Content-Security-Policy: allow 'self'; frame-ancestors 'none'
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
```

The results for the different browsers are as follows (with no user interaction):

Test Case	Firefox 32	Chrome 38	IE 11	Opera 24
Using SRC in IMG tag - GET				
Using SRC in SCRIPT tag - GET				
Using SRC in IFRAME tag - GET				
Using SRC in SOURCE tag in AUDIO tag - GET				
Using SRC in AUDIO tag - GET				
Using SRC in FRAME tag in FRAMESET tag - GET				
Using BACKGROUND in BODY tag - GET				
Using DATA in OBJECT tag (no closing tag required) - GET				
Using HREF in LINK tag to load page style				
Using CONTENT in META tag to refresh the page - GET				
Using SRC in IMAGE tag				
Using SRC in INPUT tag with type=image - GET				
Using SRC in VIDEO tag - GET				
Using BACKGROUND in STYLE in DIV tag - GET				
Using SRC in SOURCE tag in VIDEO tag - GET				
Using SRC in TRACK tag in VIDEO tag - GET				
Using POSTER in VIDEO tag - GET				
Using SRC in EMBED tag (no closing tag required) - GET				
Using xlink:href in IMAGE tag in SVG tag - GET				
Using SRCDOC in IFRAME tag - GET				
Using SRC in IMG tag with CROSSORIGIN - GET				
Using ACTION in FORM tag to target an internal IFRAME - POST				
Using SRC in IFRAME tag with a Flash file - GET				
Using ACTION in FORM tag to target an internal OBJECT tag - POST				
Loading a FORM tag inside an OBJECT tag to use action in FORM tag - POST				
Using ACTION in FORM tag to target an internal IFRAME which points to a local redirect with 307 status - POST				
Using ACTION in FORM tag to target an internal IFRAME which points to a local redirect with 308 status - POST				
Using ACTION in FORM tag to target an internal IFRAME which points to a local redirect with 301 status - GET				

Lessons Learned

In order to send a GET request to set a cookie in multiple browsers (tested on Firefox, IE, Chrome, and Opera), the following methods can be used:

Using the DATA attribute in an OBJECT tag (no closing tag is required):

```
<OBJECT data="http://target.com/setcookie?cookie=value">
</OBJECT>
```

Using the SRC attribute in an EMBED tag (no closing tag is required):

```
<EMBED src="http://target.com/setcookie?cookie=value">
</EMBED>
```

Using the SRC attribute of an IFRAME that targets a local Flash file. This Flash file reads data from a local page that redirects the request to the target website that sets a cookie:

```
<IFRAME src="./localFlash.swf?input=./localRedirect?rdir=http://target.com/setcookie?cookie=value "></IFRAME>
```

Using the ACTION attribute in a FORM tag to target an internal IFRAME. This IFRAME uses a local page that redirects the request to the target website. In this case, although the FORM tag still uses the POST method, it sends a GET request to the target after being redirected with status code of 301 (or 302):

```
<FORM name="myFORM" action="./localRedirect?rdircode=301&rdir=http://target.com/setcookie%3Fcookie=value"
method="POST" target="internalFrame">
<input type=submit />
</FORM>
<IFRAME name="internalFrame"></IFRAME>

<script>
myFORM.submit();
</script>
```

It was not possible to find one solution for all the browsers to send a POST request to the target page. However, two solutions could be used at the same time to cover all the tested browsers.

Solution one: using an OBJECT tag as an IFRAME. Although IE used the OBJECT tag as an IFRAME just like other browsers, the FORM tag could not use it as its target (it tried to open a new page which was blocked by the IE popup blocker).

```
<FORM name="myform" action="http://target.com/setcookie?cookie=value" method="POST"
target="testObjectframe">
<input type=submit />
</FORM>
<OBJECT data="/" name="testObjectframe" title="0" onload="if(this.title==0) {this.title=1;myform.submit();}">
</OBJECT>
```

Solution two: Browsers redirect a POST request with its parameters when they receive a HTTP status code 307 or 308. Among the tested browsers, Firefox needed user interaction for this to happen. IE 11 also did not support status code 308.

```
<FORM name="myform" action="./localRedirect?rdircode=307&rdir=http://target.com/setcookie%3Fcookie=value"
method="POST" target=" internalFrame">
<input type=text name=test value=test />
<input type=submit />
</FORM>

<IFRAME name=" internalFrame" onload="confirmPagLoad()"></IFRAME>

<script>
myform.submit();
</script>
```

Internet Explorer (IE) was very different than other browsers during this research. This difference was because of implementing the Platform for Privacy Preferences Project (P3P) protocol.

Briefly, IE does not allow any third party website (a website which is opened in an IFRAME and its domain is different from its parent for example) to set any cookies when it does not have an appropriate P3P header in HTTP responses.

You can read more about P3P in its **Wikipedia page (http://en.wikipedia.org/wiki/P3P)**. However, this research identified some methods with which it was possible to circumvent this policy in IE even when its privacy setting was set on "Medium High".

A live demo of the designed JS framework for this research can be found in the following URL:

http://0me.me/demo/cookie-test/testcase_runner.html (http://0me.me/demo/cookie-test/testcase_runner.html)

Note: This page sends its requests to the sdl.me server (an IIS server).

Source Code

The source code for the framework can be found on GitHub page here
- **https://github.com/nccgroup/SetCookieAnalysisInBrowsers (https://github.com/nccgroup/SetCookieAnalysisInBrowsers)**

Vendor Disclosure

This research was performed in September 2014 and the P3P policy bypass issues were reported to Microsoft in October 2014. Microsoft confirmed that we can disclose the details publicly in January 2015.

Future Research

On the pile for future research includes further test cases for different techniques and client side OBJECTs such as SVG, SWF, Silverlight, and so on. Additionally Non-standard HTML tags (browser specific ones) should also likely be added to the test cases.

Engaging NCC Groups Security Research

NCC Group Security Research is regularly engaged by our clients to undertake projects ranging from a few days through to multi-year endeavours. Your organization could benefit from working with one of the world's most unique companies when it comes to breadth, scale, global reach and technical specialism.

Want a chat?

Contact us through our website: **/en/contact-us/ (/en/contact-us/)**

Call us: +44 (0)161 209 5200

Request an expert **call you back (/en/callback/)**

Tags

cookies (/en/blog/?tag=cookies) **Web browser (/en/blog/?tag=Web+browser)**

WRITTEN BY **SOROUSH DALILI (/EN/BLOG/?AUTHOR=SOROUSH DALILI)**, AT 15:26