About myself

• Lieven Desmet

• Research manager of the iMinds-DistriNet Research Group (KU Leuven, Belgium)
  – Software security lab with 80+ researchers
  – Dedicated team on Web App Sec

• Active participation in OWASP:
  – Board member of the OWASPBelgium Chapter
• Integrating JavaScript
• Large-scale analysis of script inclusions
• Overview of mitigation techniques
  – HTML5 Sandbox/CSP-enabled security architecture
  – JSand: Server-driven sandboxing of JavaScript
• Conclusion
INTEGRATING JAVASCRIPT
<html><body>
...
<script src="http://3rdparty.com/script.js"></script>
...
</body></html>

Security model:
Third-party JavaScript is everywhere

- Advertisements
  - Adhese ad network
- Social web
  - Facebook Connect
  - Google+
  - Twitter
  - Feedsburner
- Tracking
  - Scorecardresearch
- Web Analytics
  - Yahoo! Web Analytics
  - Google Analytics
  - ...
“88.45% of the Alexa top 10,000 websites included at least one remote JavaScript library”

CCS 2012
Malicious third-party scripts can ...
And it happens in practice...

If you downloaded the qTip2 library between 8th December 2011 and 10th of January 2012, please make sure to re-download the library as the site was compromised during these dates due to malicious code injected via a Wordpress bug. Apologies for any inconvenience caused. This sort of attack as usual vulnerabilities like this can only be pro-actively remedied as they occur.

Download latest: 1.0.0-rc3

- Production - YUICompressed source code - 38KB
- Development - Uncompressed source code - 83KB
- Debugger - qTip debug plugin for easier development - 5KB
- jQuery 1.3.2 - Tested and recommended for qTip - 56KB

Download! 94KB

32 days...

LARGE-SCALE ANALYSIS OF SCRIPT INCLUSIONS
• Crawled over 3,300,000 pages belonging to the Alexa top 10,000

• Discovered:
  – 8,439,799 remote inclusions
  – 301,968 unique JS files
  – 20,225 uniquely-addressed remote hosts
How many remote hosts?
### Popular JavaScript libraries and APIs

**OWASP**  
The Open Web Application Security Project

<table>
<thead>
<tr>
<th>Offered service</th>
<th>JavaScript file</th>
<th>% Top Alexa</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web analytics</td>
<td><a href="http://www.google-analytics.com/ga.js">www.google-analytics.com/ga.js</a></td>
<td>68.37%</td>
</tr>
<tr>
<td>Dynamic Ads</td>
<td>pagead2.google syndication.com/pagead/show_ads.js</td>
<td>23.87%</td>
</tr>
<tr>
<td>Web analytics</td>
<td><a href="http://www.google-analytics.com/urchin.js">www.google-analytics.com/urchin.js</a></td>
<td>17.32%</td>
</tr>
<tr>
<td>Social Networking</td>
<td>connect.facebook.net/en_us/all.js</td>
<td>16.82%</td>
</tr>
<tr>
<td>Social Networking</td>
<td>platform.twitter.com/widgets.js</td>
<td>13.87%</td>
</tr>
<tr>
<td>Social Networking &amp; Web analytics</td>
<td>s7.addthis.com/js/250/addthis_widget.js</td>
<td>12.68%</td>
</tr>
<tr>
<td>Web analytics &amp; Tracking</td>
<td>edge.quantserve.com/quant.js</td>
<td>11.98%</td>
</tr>
<tr>
<td>Market Research</td>
<td>b.scorecardresearch.com/beacon.js</td>
<td>10.45%</td>
</tr>
<tr>
<td>Google Helper Functions</td>
<td><a href="http://www.google.com/jsapi">www.google.com/jsapi</a></td>
<td>10.14%</td>
</tr>
<tr>
<td>Web analytics</td>
<td>ssl.google-analytics.com/ga.js</td>
<td>10.12%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>JS Action</th>
<th># of Top scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reading Cookies</td>
<td>41</td>
</tr>
<tr>
<td>document.write()</td>
<td>36</td>
</tr>
<tr>
<td>Writing Cookies</td>
<td>30</td>
</tr>
<tr>
<td>eval()</td>
<td>28</td>
</tr>
<tr>
<td>XHR</td>
<td>14</td>
</tr>
<tr>
<td>Accessing LocalStorage</td>
<td>3</td>
</tr>
<tr>
<td>Accessing SessionStorage</td>
<td>0</td>
</tr>
<tr>
<td>Geolocation</td>
<td>0</td>
</tr>
</tbody>
</table>
New Attacks?

• 8.5 million records of remote inclusions
• Are there new attack vectors to exploit the script-inclusion pattern?

• 4 new attack vectors
  – Cross-user & Cross-network Scripting
  – Stale domain-based inclusions
  – Stale IP-based inclusions
  – Typo-squatting Cross-Site Scripting
• What happens when you trust a remote site and the domain of that site expires?
  – Anyone can register it, and start serving malicious JS
  – Equal in power to stored XSS
• 56 domains found, used in 47 sites
Shopping spree!

- Registered some of the stale domains:
  - blogtools.us -> goldprice.org (4,779th in Alexa)
  - hbotapadmin.us -> hbo.com

<table>
<thead>
<tr>
<th></th>
<th>Blogtools.us</th>
<th>Hbotapadmin.com</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visits</td>
<td>80,466</td>
<td>4,615</td>
</tr>
<tr>
<td>Including domains</td>
<td>24</td>
<td>4</td>
</tr>
<tr>
<td>Including pages</td>
<td>84</td>
<td>41</td>
</tr>
</tbody>
</table>
• Typo-squatting
  – registering domains that are mistypes of popular domains
  – Serve ads, phishing, drive-by downloads etc. to users that mistype the domain
• Unfortunately... developers are also humans
  – <script src="http://googlesyndicatio.com/..."></script>
<table>
<thead>
<tr>
<th>Intended domain</th>
<th>Actual domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>google syndication.com</td>
<td>google syndicatio.com</td>
</tr>
<tr>
<td>purdue.edu</td>
<td>purude.edu</td>
</tr>
<tr>
<td>worldofwarcraft.com</td>
<td>worldofwaircraft.com</td>
</tr>
<tr>
<td>lesechos.fr</td>
<td>lessechos.fr</td>
</tr>
<tr>
<td>onegrp.com</td>
<td>onegrp.nl</td>
</tr>
</tbody>
</table>

Examples found...

- **Googlesyndicatio.com**
  - Unique visitors: 163,188
  - Including domains: 1185
  - Including pages: 21,830
OVERVIEW OF MITIGATION TECHNIQUES
Existing mitigation techniques?

- Limit third-party code to safe subset of JavaScript
  - Facebook JS, ADSafe, ADSafety, ...

- Browser-based sandboxing solutions
  - ConScript, WebJail, Contego, ...

- Server-side transformations of scripts to be included
  - Google Caja, Jacaranda, BrowserShield, ...

  - No compatibility with existing scripts
  - Browser modifications imply short-term deployment issues
  - No direct script delivery to browser
    - Changes architecture of the web
Emerging solutions: Client-side security architectures

• JavaScript security architecture on top of mainstream browsers
  – Sandboxing/isolation of untrusted JavaScript code
  – Policy-controlled mediation to the actual DOM

• HTML5 sandbox/CSP-enabled security architecture

• TreeHouse: web workers sandbox architecture

• JSand: SES-enabled sandbox architecture
Based on the talk of Mike West at Devoxx 2012
Securing the Client-Side: Building safe web applications with HTML5
https://mikewest.org/2013/02/securing-the-client-side-devoxx-2012

HTML5 SANDBOX/CSP-ENABLED SECURITY ARCHITECTURE
Content Security Policy (CSP)

- Issued as HTTP response header
  - `Content-Security-Policy: script-src 'self'; object-src 'none'`
- Specifies which resources are allowed to be loaded as part of your page
- Extremely promising as an additional layer of defense against script injection
Example of sandboxing unsafe JavaScript

Main site

Sandboxed iframe
- Runs in unique origin
- Allowed to run JS

Web Messaging

Sandboxed JS execution environment

Secured with CSP

Delegates insecure executions to the sandboxed iframe

“Used in office document reader on Chrome OS”

“Securing the Client-Side: Building safe web applications with HTML5” (Mike West, Devoxx 2012)
Content-Security-Policy: script-src 'self'

```html
<html><head>
    <script src="main.js"></script>
</head>
<body>
    <a href="#" id="sandboxFrame"/>Click here</a>
    <iframe id="sandboxFrame" sandbox="allow-scripts" src="sandbox.html">
    </iframe>
    <div id="#content"></div>
</body></html>
```
<html>
<head>
  <script>
    window.EventListener('message', function(event) {
      var command = event.data.command;
      var context = event.data.context;
      var result = callUnsafeFunction(command, context);
      event.source.postMessage({
        html: result}, event.origin);
    });
  </script>
</head>
</html>
document.querySelector('#click').addEventListener('click',
    function(){
        var iframe = document.querySelector('#sandboxFrame');
        var message = { command: 'render'; context: {thing: 'world'});
        iframe.contentWindow.postMessage(message, '*');
    });

window.addEventListener('message', function(event){
    //Would be dangerous without the CSP policy!
    var content = document.querySelector('#content');
    content.innerHTML = event.data.html;
});
Pieter Agten et. al. **JSand: Complete Client-Side Sandboxing of Third-Party JavaScript without Browser Modifications.** In proceedings of the Annual Computer Security Applications Conference (ACSAC 2012).

**JSAND: SERVER-DRIVEN SANDBOXING OF JAVASCRIPT**
JSand: Server-driven sandboxing of JavaScript

site

Browser

JS

JSand

policy1

policy2

JS1

script provider 1

JS2

script provider 2

OWASP
The Open Web Application Security Project
Jsand: under the hood

Embedding page

1. 3rd party JavaScript
2. DOM
3. JSand
• Secure ECMAScript library (SES)
  – Developed by Google CAJA Team
  – Provides object-capability functionality within JavaScript

• JS Proxy API
  – Provides transparent proxy capabilities in wrapping native functionality

• Membrane pattern
  – Guarantees that no object capabilities (i.e. References) leak through the sandbox perimeter
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>JSP Page</title>
  <jsand:initialize/>
  <jsand:sandbox policy="my embedded script">
    <jsand:code>alert("inline code on the page");</jsand:code>
  </jsand:sandbox>
</head>
<body>
  <h1>Hello World!</h1>
</body>
</html>
<jsand:sandbox policy="googlemapsNoGeolocation">
  <jsand:code>
    canvasID = "map_canvas2";
    failcity = "New York";
    failpos = new google.maps.LatLng(40.69, -73.95);
  </jsand:code>
  <jsand:script src="googlemaps-geolocation.js"/>
</jsand:sandbox>
Evaluation on legacy scripts

• Google Analytics
  – Needs 1 client-side JS AST transformation

• Google Maps
  – Needs support for dynamic script loading
  – Needs 3 client-side JS AST transformation

• JQuery

Demo available at http://demo-jsand.websand.eu/
CONCLUSION
• Most common way of integrating 3\textsuperscript{rd} party JavaScript
  – More than 88% of websites integrate 3\textsuperscript{rd} party scripts
  – Google is the absolute #1 script provider

• Malicious or compromised script providers obtain full control over websites on which they are integrated
  – E.g. qTip2, googlesyndycatio.com, blogtoos.us, ...
Existing mitigation techniques

- None of them can be integrated seamlessly
  - Require browser modifications
  - Require server-side processing
  - Require re-architecting the application
  - Have restrictions on JS the language features

- Showed some insights in 2 promising directions
  - iFrame/CSP based sandboxing
  - Server-driven sandboxing with JSand
• The work is partially funded by the European FP7 projects WebSand, STREWS and NESSoS.

• With the financial support from the Prevention of and Fight against Crime Programme of the European Union.