Sandboxing JavaScript

Lieven Desmet – iMinds-DistriNet, KU Leuven
Lieven.Desmet@cs.kuleuven.be
OWASP BeNeLux Days 2012 (29/11/2012, Leuven)






KU LEUVEN
DISTRINET RESEARCH GROUP
iMinds
CONNECT.INNOVATE.CREATE

About myself

@lieven_desmet

- Lieven Desmet
- Research manager of the iMinds-DistriNet Research Group (KU Leuven, Belgium)
- Active participation in OWASP:
  - Board member of the OWASP Belgium Chapter
  - Co-organizer of the academic track on past OWASP AppSec Europe Conferences

Earlier results:
CSRF protection: CsFire

- Implemented as a FireFox/Chrome extension
- Available at the Mozilla Add-ons website
  - 45K+ download
  - 3500+ daily users

- Since iMinds – the conference, also available for Chrome!

Sandboxing JavaScript: Outline

- Integrating JavaScript
- Large-scale analysis of script inclusions
- JSand: Server-driven sandboxing of JavaScript
- Challenge: *How to support Google Maps?*
- Evaluation on legacy scripts
- Conclusion

Integrating JavaScript

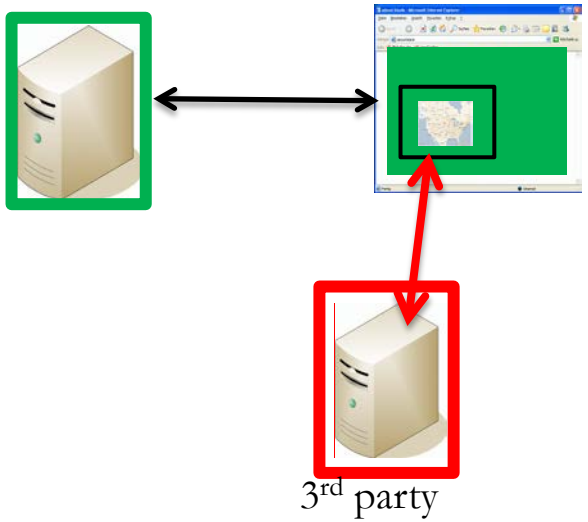





Two basic composition techniques

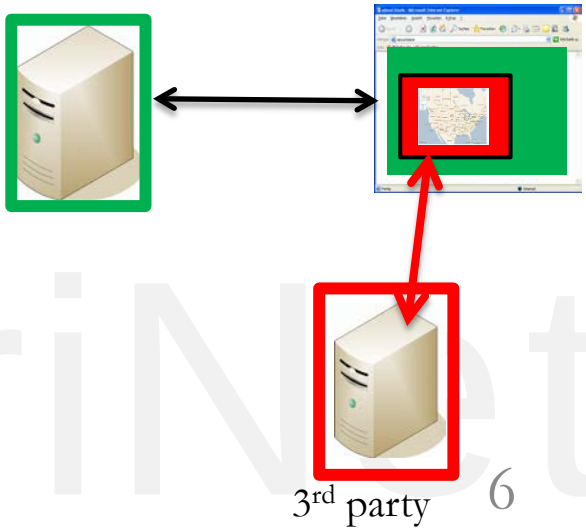
Script inclusion

```
<html><body>
…
<script src="http://3rdparty.com/script.js">
</script>
…
</body></html>
```



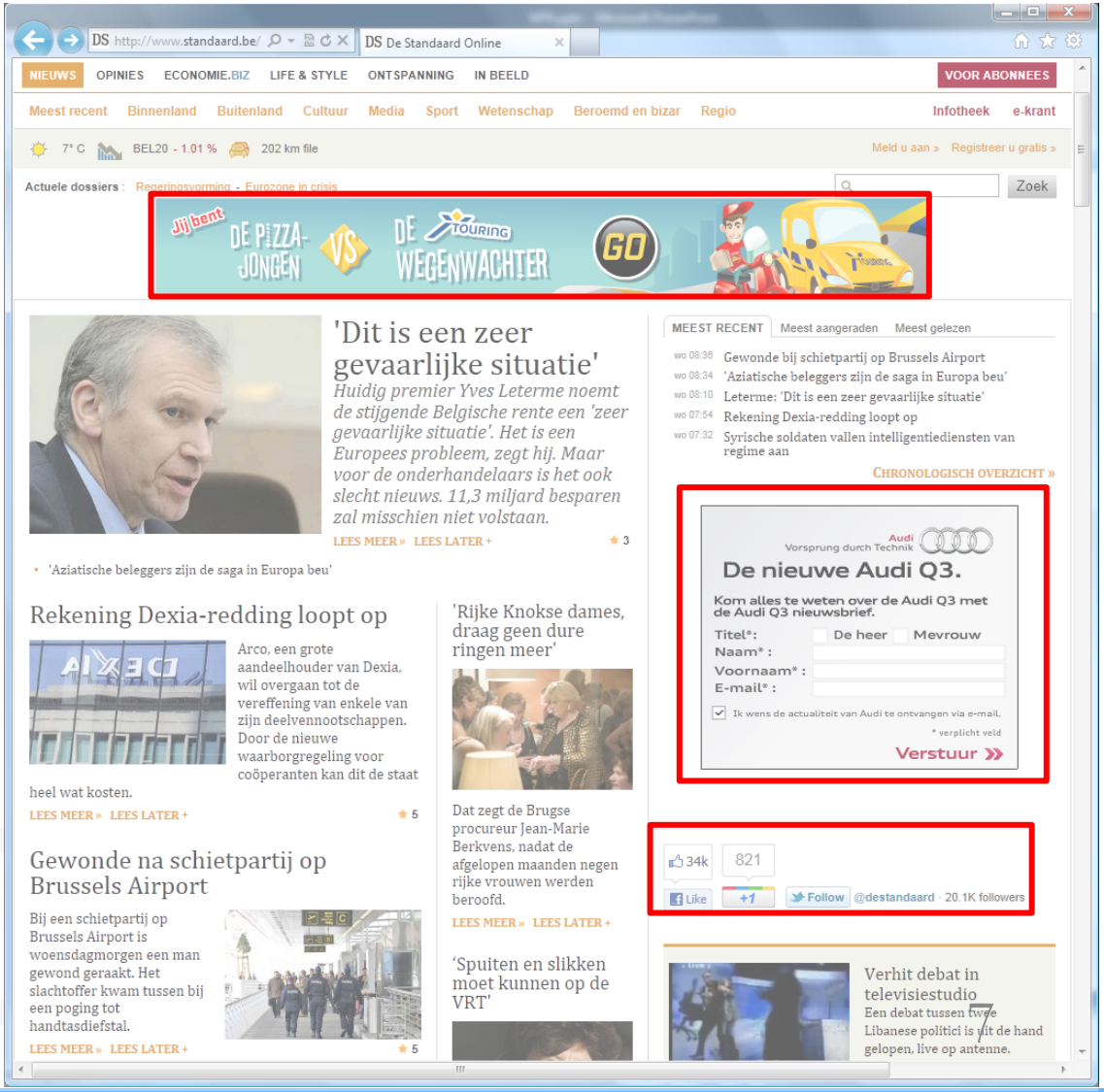
Iframe integration

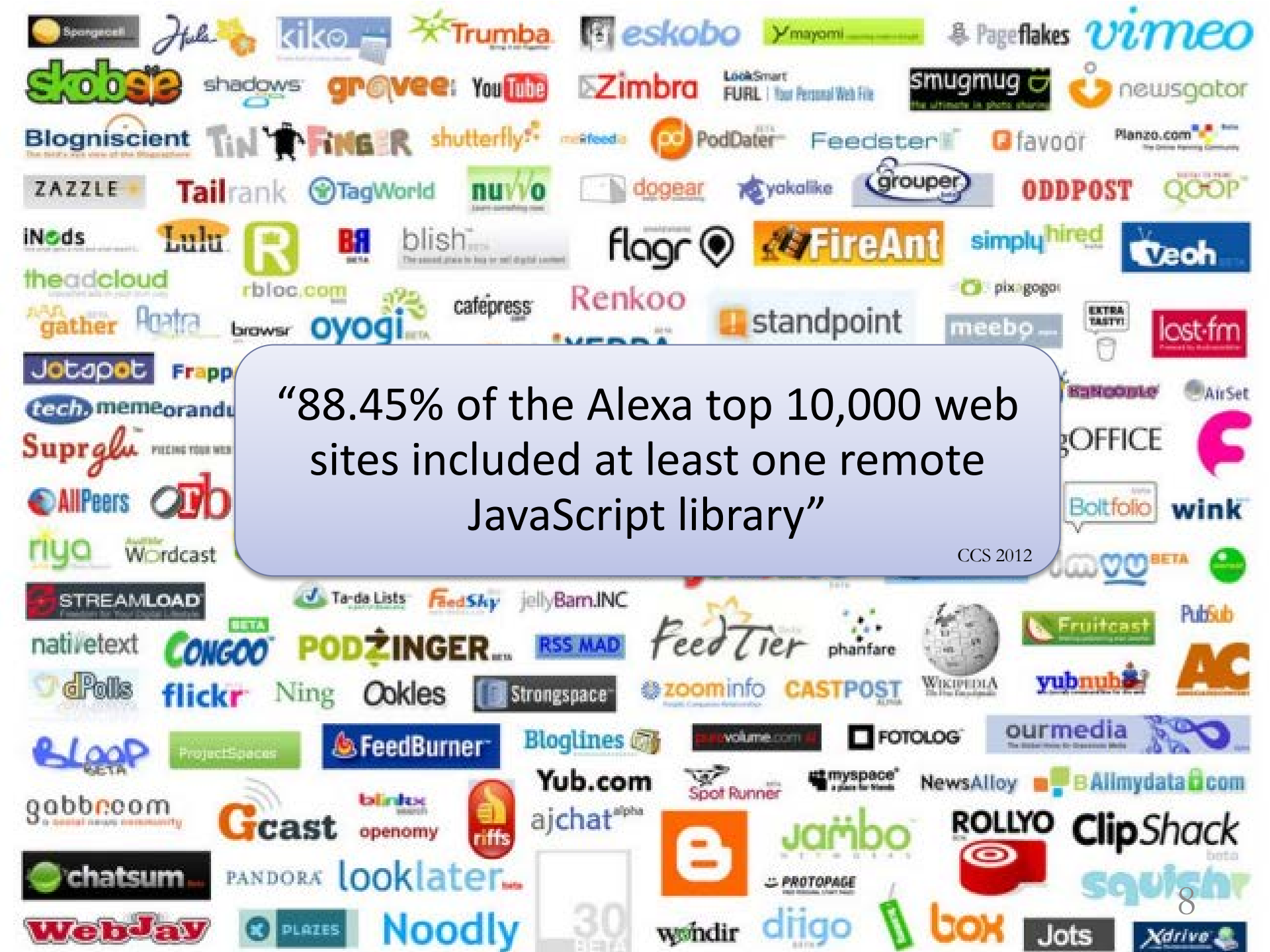
```
<html><body>
…
<iframe src="http://3rdparty.com/frame.html">
</iframe>
…
</body></html>
```



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 web sites included at least one remote JavaScript library"
CCS 2012

Malicious third-party scripts can …

And it happens in practice…

32 days…

Existing solutions?

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

No compatibility with existing scripts

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

Browser modifications imply short-term deployment issues

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

No direct script delivery to browser

Changes architecture of the web

Large-scale analysis of script inclusions

Nick Nikiforaki *et. al*. **You are what you include: Large-scale evaluation of remote JavaScript inclusions**. In *Proceedings of the ACM Conference on Computer and Communications Security*. 2012.






Large-scale analysis of script inclusions

- Data collection experiment
- Crawling results
- New remote inclusion attacks

- More detail in the CCS 2012 paper *"You are what you include: Large-scale evaluation of remote JavaScript inclusions"*

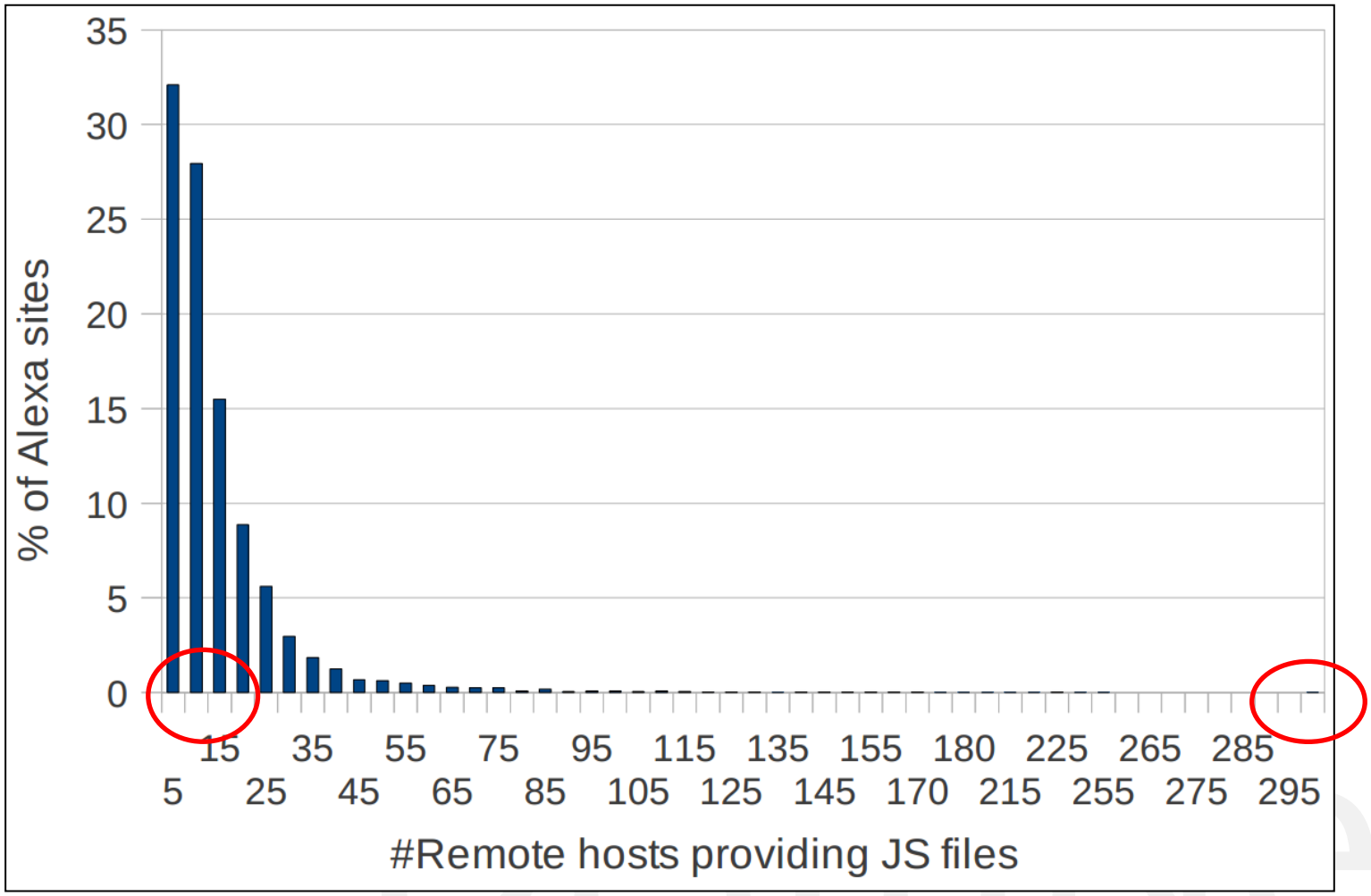
Data Collection Experiment

- Discovering remote JavaScript inclusions (aka trust relationships)
- Alexa Top 10,000
  - Up to 500 pages from each
  - Pages chosen by Bing
    - Query “site:google.com”
- Crawler based on HtmlUnit
  - GUI-less Java browser with JavaScript support

Crawling results

- 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
    - Addressed by domain-name
    - Addressed directly by IP address

How many remote hosts?



Popular JavaScript libraries and APIs

Offered service	JavaScript file	% Top Alexa
Web analytics	www.google-analytics.com/ga.js	68.37%
Dynamic Ads	pagead2.googlesyndication.com/pagead/show_ads.js	23.87%
Web analytics	www.google-analytics.com/urchin.js	17.32%
Social Networking	connect.facebook.net/en_us/all.js	16.82%
Social Networking	platform.twitter.com/widgets.js	13.87%
Social Networking & Web analytics	s7.addthis.com/js/250/addthis_widget.js	12.68%
Web analytics & Tracking	edge.quantserve.com/quant.js	11.98%
Market Research	b.scorecardresearch.com/beacon.js	10.45%
Google Helper Functions	www.google.com/jsapi	10.14%
Web analytics	ssl.google-analytics.com/ga.js	10.12%

JS Action	# of Top scripts
Reading Cookies	41
document.write()	36
Writing Cookies	30
eval()	28
XHR	14
Accessing LocalStorage	3
Accessing SessionStorage	0
Geolocation	0

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

Stale domain-based inclusions

- 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 the, almost extinct, 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

	Blogtools.us	Hbotapadmin.com
Visits	80,466	4,615
Including domains	24	4
Including pages	84	41

Typo-squatting XSS

- 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/...>

Examples found…

Intended domain	Actual domain
googlesyndication.com	googlesyndicatio.com
purdue.edu	purude.edu
worldofwarcraft.com	worldofwaircraft.com
lesechos.fr	lessechos.fr
onegrp.com	onegrp.nl

	Googlesyndicatio.com
Unique visitors	163,188
Including domains	1185
Including pages	21,830

JSand: Server-driven sandboxing of JavaScript

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).



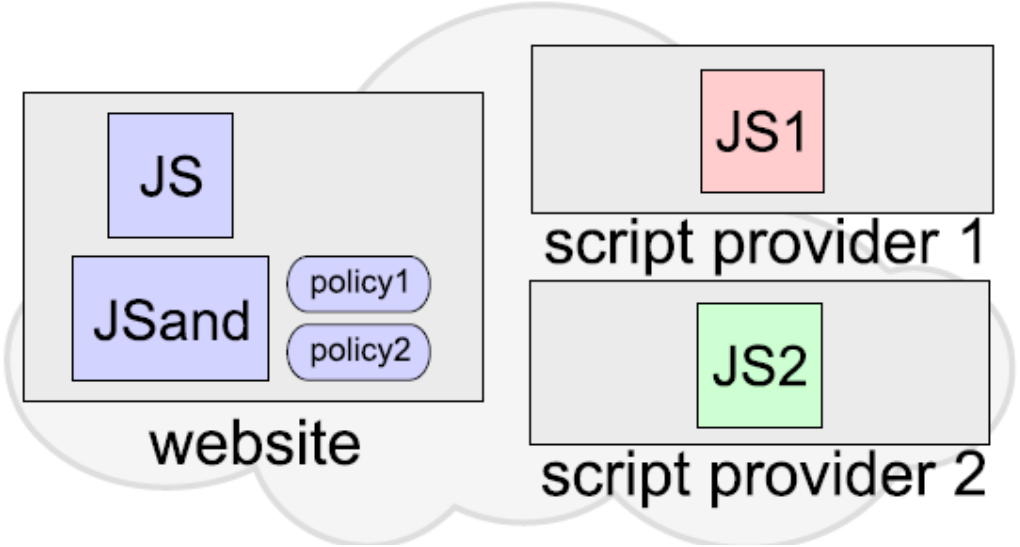



JSand Requirements

- Secure integration of 3rd party JavaScript
- Under control of the website owner

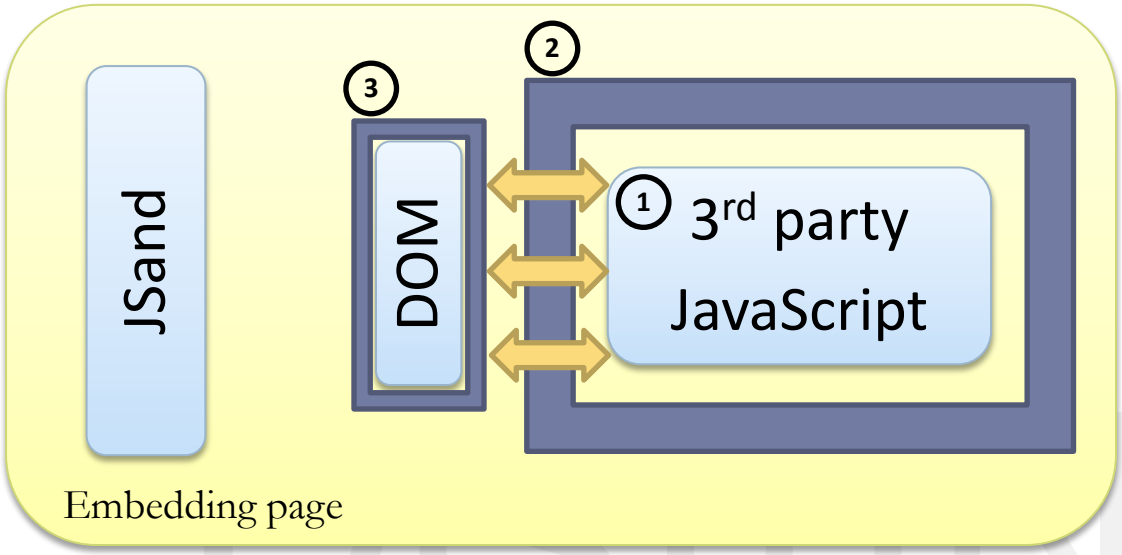
1. Complete mediation
   - All security sensitive operations are completely mediated (DOM, JS APIs, …)
2. Backward compatible
   - No browser modifications
   - Compatible with direct script delivery to the browser
   - Support for legacy scripts
3. Reasonable performance

JSand high-level architecture



Under the hood

1) Download third-party script directly to browser
2) Load script in **isolated** object-capability environment using Google's Secure ECMAScript
3) Enable access to outside using *membrane* around DOM
   Policy determines permitted operations



SES example

Secure ECMAScript library usage (simplified):

```
var scriptCode = "window.alert('Boo!');";
ses.execute(scriptCode);
```

with block

strict mode

```
var catchAll = makeCatchAll();

with (catchAll) {
  (function() {
    "use strict";
    // ** Example malicious code **
    window.alert('Boo!');
    // ****************************
  })();
}
```

Jsand wrapper proxy example

Example (highly simplified):

```
function wrap(target, policy) {
  var handler = {
    get: function(propertyName) {
      if (policy.isPropertyAllowed(target, propertyName)) {
        return wrap(target[name], policy);
      }
      return undefined;
    }
  }
  return Proxy.create(handler);
}

var windowProxy = wrap(window, somePolicy);
windowProxy.alert("Foo");
```

Challenge:
How to support Google Maps?






Several Implementation challenges

- Secure ECMAScript restrictions
- Dynamic script loading
- Remote script fetching

Challenge 1: Secure ECMAScript restrictions

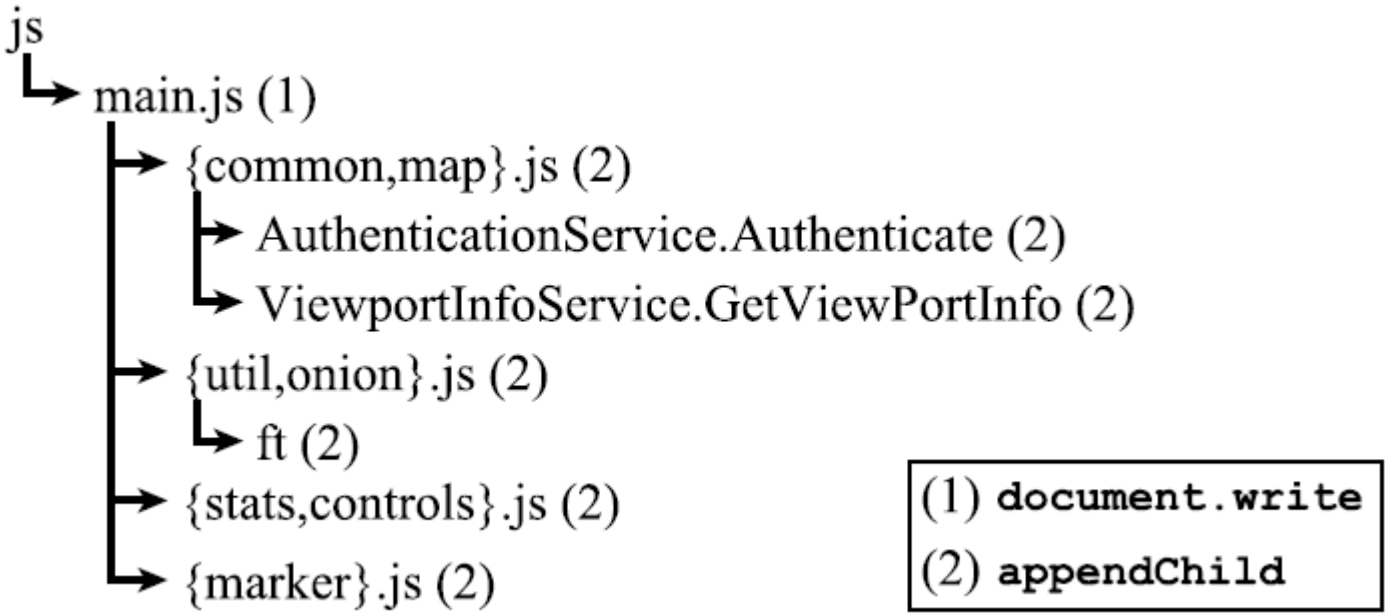
- Global variables
  - Global variables are no longer aliased by properties on the global object and vice versa
- Strict mode enforcement
  - Drops support for *with*
  - Prevents variable introduction via *eval*
  - No binding of *this* in functions calls
- How to support legacy scripts?

JS transformations to the rescue

- Client-side AST transformations using Uglify
  - T1: global alias for each property of window
  - T2: property of window for each global variable
  - T3: binding this to window in function calls
- No full translation from ES5 to SES, but a sufficient approximation
- The set of transformations expected to be extended to support more legacy scripts

Challenge 2: Dynamic script loading in JavaScript

- Example from Google Maps



Secure dynamic script evaluation

- Special handlers to intercept all methods that allow script tags to be added
  - node.appendChild, node.insertBefore, node.replaceChild, node.insertAfter
  - document.write, …

1. Parse partial DOM tree/HTML
2. Execute scripts in the sandbox environment

Different parsing techniques

- Technique 1: Via a sandboxed iframe
  1. Create sandbox iframe
  2. Set content via srcdoc attribute
  - → More performant
  - → Parsed exactly as will be interpreted by browser
  - → Executed asynchronously (!)
- Technique 2: Via a HTML parsing library in JavaScript

Loading additional code in the sandbox

- Several use cases require external code to be executed in a previously set up sandbox
  - Loading API + glue code
  - Dynamic script loading
- Two new operations:
  - innerEval(code)
  - innerLoadScript(url)
- Dynamic variable analysis needed in SES

Challenge 3: Remote script loading

- The JSand framework needs to be able to load script from remote script providers

- Inherent problem for all JS security architecture
- Current prototype relies on:
  - CORS/UMP headers set by the script provider
  - Server-side JavaScript proxy

Evaluation on legacy scripts






Evaluation on legacy scripts

- Google Analytics
  - T2 to make _gaq available as window._gaq
  - Hosting website can access sandbox script via innerEval
- Google Maps
  - Dynamic script loading
  - T1+T2+T3 are needed to function
- JQuery

Performance benchmarks

- Micro benchmarks
  - JSand loadtime: 48.5 ms
  - JQuery loadtime: 1350.6 ms
    - Mainly due to AST script rewriter
    - JQuery loadtime (w/o AST trans): 598.2 ms
  - Membrane transition cost: 7.1 µs
- Macro benchmarks
  - Google Maps loadtime: 1432.8 ms
    - vs 308.0 ms outside JSand
  - Google Maps interaction delay: 420.0 ms
    - vs 320.2 ms outside JSand

Conclusion






Conclusion

1. Complete mediation ✓
   - All security sensitive operations must be completely mediated (DOM, JS APIs, …)
2. Backwards compatible ✓
   - No browser modifications
   - Direct script delivery to the browser
   - Support for legacy scripts
     - Google Analytics, Google Maps, JQuery
3. Reasonable performance overhead ✓