You are what you include:

Large-scale evaluation of remote JavaScript inclusions

Nick Nikiforakis, Luca Invernizzi, Alexandros Kapravelos, Steven Van Acker, Wouter Joosen, Christopher Kruegel, Frank Piessens, Giovanni Vigna
Introduction: my USB stick
Introduction: browsers don’t care
You are what you include:
Large-scale evaluation of remote JavaScript inclusions

Nick Nikiforakis, Luca Invernizzi, Alexandros Kapravelos, Steven Van Acker, Wouter Joosen, Christopher Kruegel, Frank Piessens, Giovanni Vigna
Outline

- JavaScript in a browser
  - ... and motivation for an experiment
- Our experiment
- Our results
  - Some unsurprising results
  - Some weirdness
- Countermeasures
JavaScript in the browser
JavaScript in a browser: origins

Origin: http, facebook.com, 80

Origin: http, google-maps.com, 80
JavaScript in a browser: inclusions

Origin: http, facebook.com, 80

Origin: http, google-maps.com, 80
Motivation...

32 days...
Our experiment
Our experiment: questions

Given that remote JS inclusions happen...

... Should sites be trusting remote providers?

- Which third-party vendors do they currently trust?
- Are JS providers capable of securing their website? What is the quality of maintenance profile of each JS provider?
  - Could a provider be attacked as a way of reaching a harder-to-get target?
- Are there attack vectors, in relation to remote inclusions, that we were not aware of?
- How can one protect his web application?
  - Are coarse-grained sandboxes sufficient?
Our experiment: crawler

- Crawler requirements:
  - Download webpages
  - Log JavaScript inclusions
  - Execute JavaScript for dynamic inclusions

- HTMLUnit: JS-enabled headless browser in Java

- Queried Bing for max 500 pages of Alexa top 10000
Our experiment: some numbers

- Crawled over 3,300,000 pages belonging to the Alexa top 10,000

- Discovered:
  - 8,439,799 remote inclusions
  - 88.45% of Alexa top 10k uses at least 1 remote JS library
  - 301,968 unique JS files
  - 20,225 uniquely-addressed remote hosts
Results: unsurprisingly...
Results: how many remote hosts?
## Results: Popular JavaScript includes

<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.googlesyndication.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>
Results: quality of maintenance?

- Assumption: Unmaintained websites are easier to attack
- QoM indicator comprised of these factors:
  - Availability: DNS not expired, publicly-routable IP address
  - Cookies (at least one):
    - HttpOnly?
    - Secure?
    - Path & Expiration?
  - Anti-XSS & Anti-Clickjacking headers?
  - TLS/SSL implementation
    - Weak ciphers
    - Valid certificates
    - Strict Transport Protocol
  - Cache control when using TLS/SSL?
  - Outdated web servers?
Results: QoM in color!
Results: like attracts like
Results: weirdness!
Results: weirdness?

- In about 8.5 million records of remote inclusions, is there something that we didn’t know?

- 4 Things! 😊
  - Cross-user & Cross-network Scripting
  - Stale domain-based inclusions
  - Stale IP-based inclusions
  - Typo-squatting Cross-Site Scripting
Weirdness: Cross-user Scripting

```html
<script src="http://localhost/script.js">

- 133 records were found
- 131 specified a port (localhost:12345), always greater than 1024

Attack:
- Setup a web-server, listen to high ports, hack other users
```
Weirdness: Cross-network Scripting

- `<script src='http://192.168.2.3/script.js'>`  
  - 68 of them
  - Same as before, but now you just need to be in the same local network

- Who is doing that?
  - akamai.com
  - virginmobileusa.com
  - gc.ca (Government of Canada)
Weirdness: Stale IP-based remote inclusions

- What if the IP address of the host which you trust for JavaScript, changes?
  - The including page’s scripts must also change
  - Do they?

- Manual analysis of the 299 pages
  - 39 addresses had:
    a) Not changed
    b) no longer provided JavaScript
      a) In 89.74%, we got a “Connection Timeout”
Weirdness: 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
  - Try proving in court that someone hacked you with that

56 domains found, used in 47 sites

- 6 were identified as special cases (TXSS)

Scared yet?
Weirdness: Typo-squatting XSS (TXSS)

- Unfortunately... developers are humans
  - `<script src="http://googlesyndicatio.com/...">`

- Typo-squatting
  - registering domains that are mistypes of popular domains
  - Serve ads, phishing, drive-by downloads etc. to users that mistype the domain
Weirdness: TXSS examples found...

<table>
<thead>
<tr>
<th>Intended domain</th>
<th>Actual domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>google syndication.com</td>
<td>google syndication.com</td>
</tr>
<tr>
<td>purdue.edu</td>
<td>purdue.edu</td>
</tr>
<tr>
<td>world of warcraft.com</td>
<td>world of warcraft.com</td>
</tr>
<tr>
<td>lesechos.fr</td>
<td>lesechos.fr</td>
</tr>
<tr>
<td>onegrp.com</td>
<td>onegrp.nl</td>
</tr>
</tbody>
</table>

- **Unique visitors** 163,188
- **Including domains** 1185
- **Including pages** 21,830
Countermeasures
Countermeasures

- Problems with remote inclusions
  - Never the visitor’s fault
  - A developer can mess up
    - Cross-user, cross-network and TXSS
  - The remote host can mess up
    - Low security, expiration of domain names

- How to protect one’s self?
  i. Sandbox remote scripts
  ii. Download them locally
Countermeasures: sandboxing

- Is it feasible?
- What are the current requirements of legitimate scripts?
- Study the top 100
  - Automatically study each script
    - JavaScript wrappers + stack trace
  - Find out what sensitive resources they access
    - Cookies, Storage, Geolocation, Eval, document.write
  - Is containment possible?
... sandboxing: Access to resources

<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><code>document.write()</code></td>
<td>36</td>
</tr>
<tr>
<td>Writing Cookies</td>
<td>30</td>
</tr>
<tr>
<td><code>eval()</code></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>

Coarse-grained sandboxing is useless here, legitimate scripts and attackers act the same way 😞
Countermeasures: local copies

- Study the frequency of script modifications
  - Discover overhead for administrator
- Top 1,000 most-included scripts (803)
  - Download every script three consecutive times and remove the ones that changed all three times
  - Study the rest for a week
- 10.21% were modified
  - 6.97% were modified once
  - 1.86% were modified twice
  - 1.83% were modified three or more
- 89.79% was never modified!
- 96.76% at most once
Conclusions

- Remote inclusions mean, almost unconditional, trust
  - Think twice before including something from a remote host

- **Do NOT:**
  - Include from 127.0.0.1 or private networks
  - Include from IP addresses
  - Include from stale domains
  - Include from typodomains
  - Include from questionable JS providers

- **Do:**
  - Make local copies
  - Sandbox 3rd party JS if it is feasible
  - Have hope: sleep sound tonight
Thank you!

Questions?