INCIDENT RESPONSE & INVESTIGATION SERVICES • PENETRATION TESTING • SECURITY RESEARCH •
SOFTWARE ENGINEERING SECURITY ASSURANCE • SECURITY ARCHITECHTURE & DESIGN ASSURANCE • TRAINING

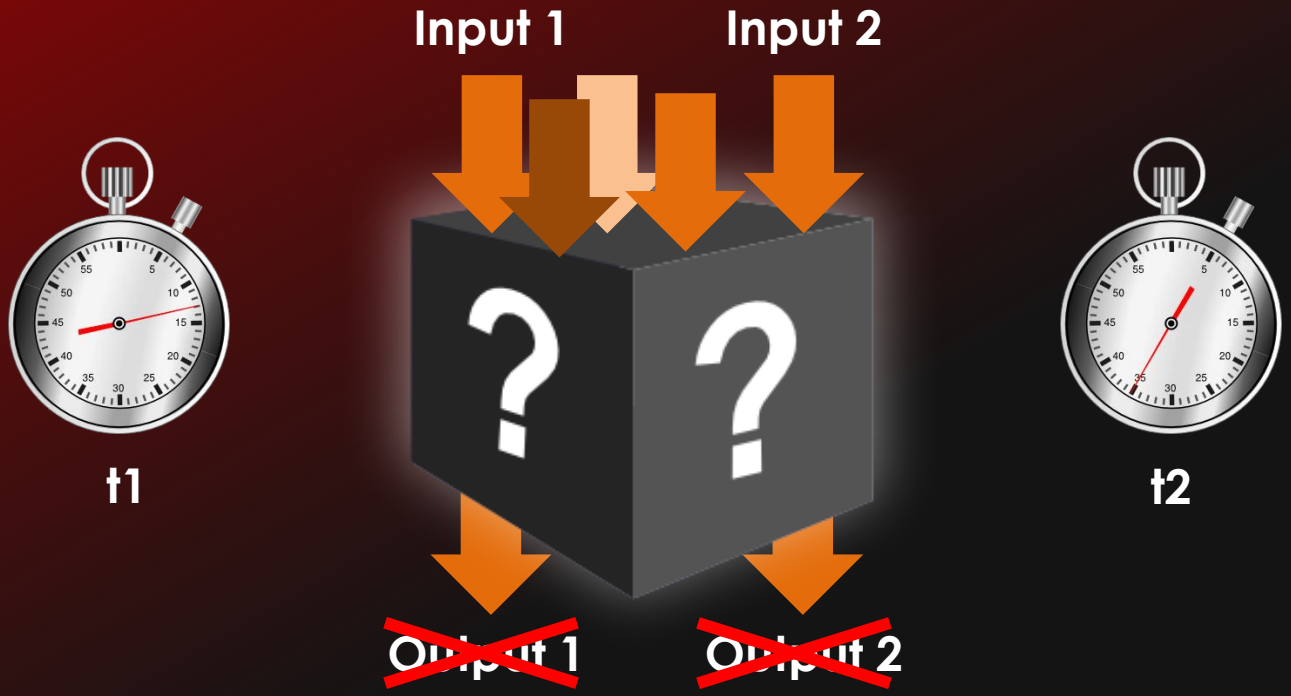
Pixel Perfect Timing Attacks

Paul Stone (@pdjstone)

Research. Response. Assurance. @CTXIS

Timing Attacks

Using timing information to discover the secrets of a ‘black box’



Browser Black Boxes

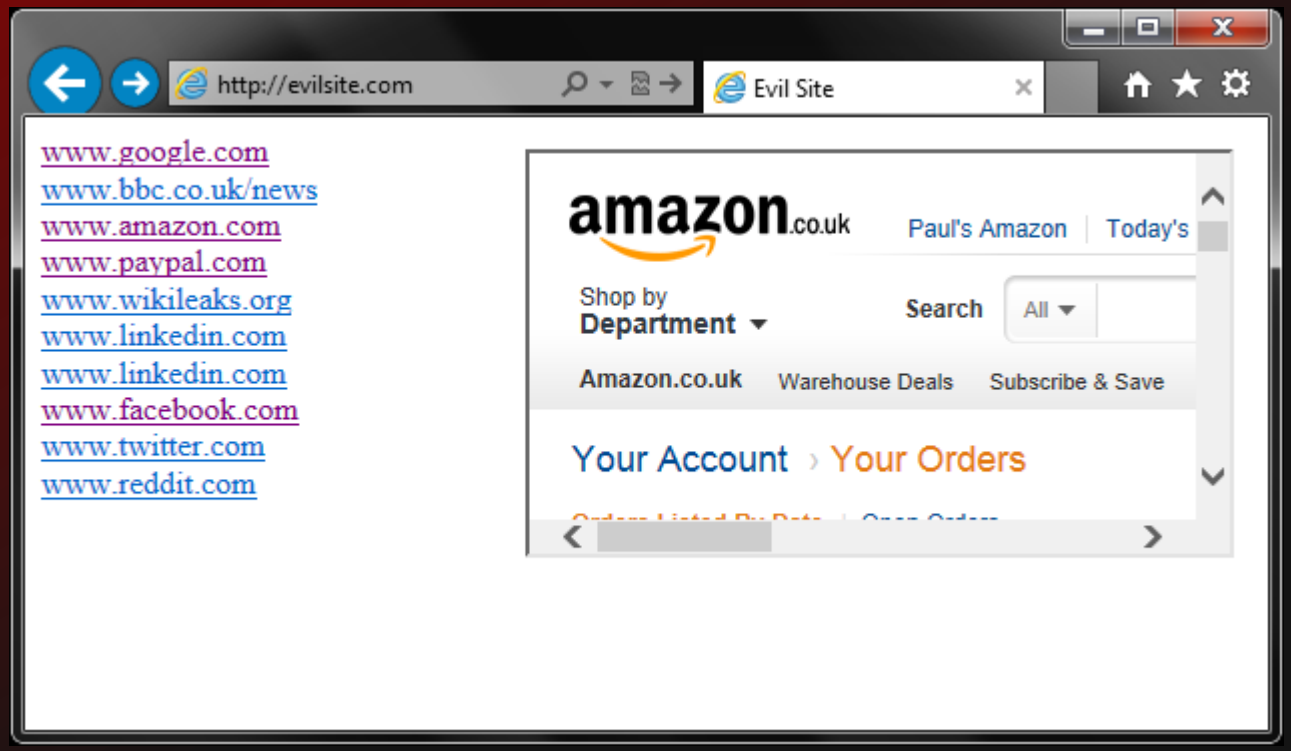
- Same Origin Policy: Site A cannot read or modify data from site B
- Can still *make requests* to other sites
  - <img src="...">
  - <script src="...">
  - XMLHttpRequest
- But cannot (usually) read results

Browser Black Boxes

- Link Colours – information from browser history
- Iframes – load 3rd party site inside your own
- But browser restrictions prevent page JavaScript from ‘seeing’ these things

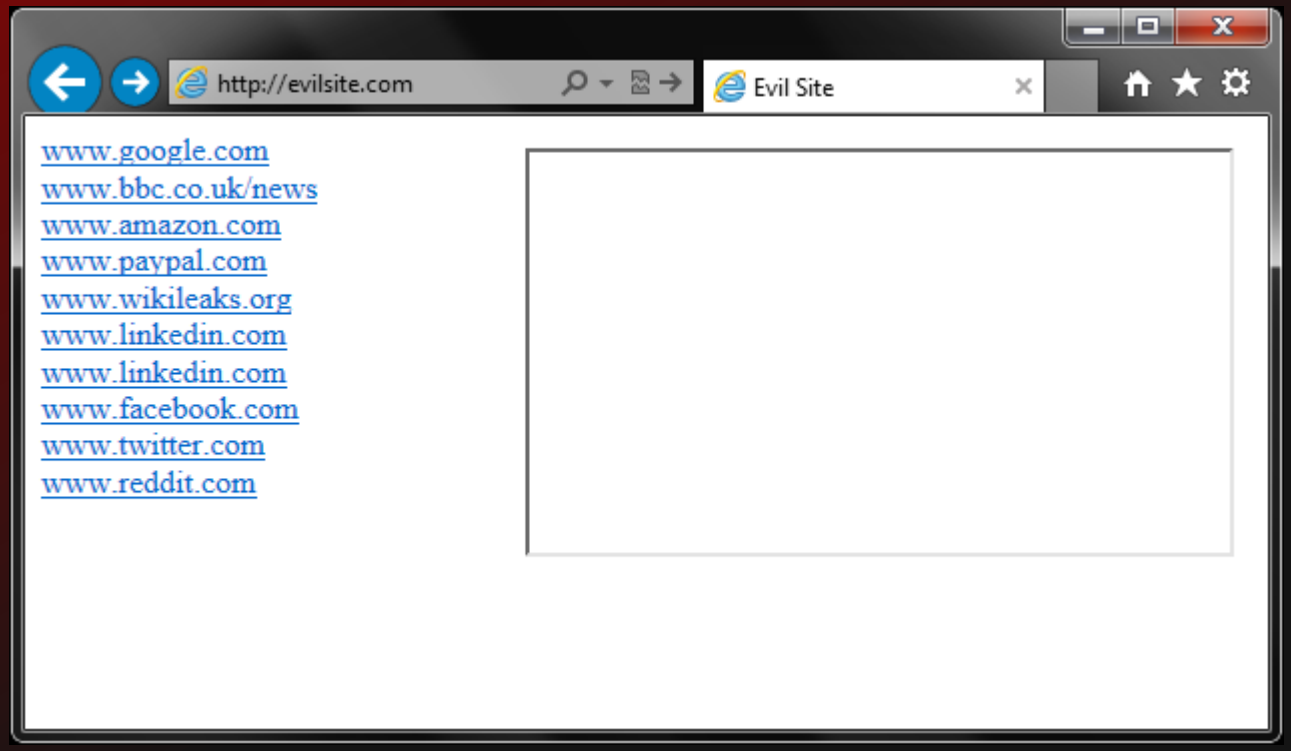
Browser Black Boxes

- How much private information is shown here?



Browser Black Boxes

- What the page ‘sees’:



In this talk

- Browser History Sniffing via Timing Attack
- Reading pixels from frames via Timing Attack
- Using new browser features (HTML5-ish)

Page Request Timing

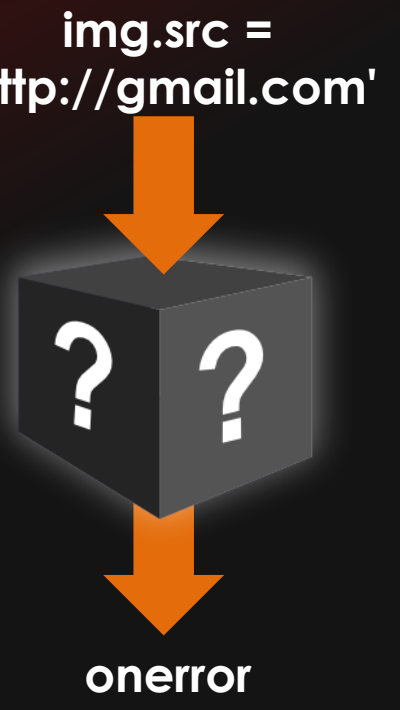
- Is the user logged into GMail?

```
var start = Date.now();         // current time in ms
var img = new Image();

img.onerror = function() {      // callback function
  var t = Date.now() - start;
}

img.src = 'http://gmail.com'; // not actually an image
```

http://crypto.stanford.edu/~dabo/papers/webtiming.pdf

Page Request Timing

- Image request is our black box
- URL is our input
- onerror callback is our output
- Date.now() is our stopwatch

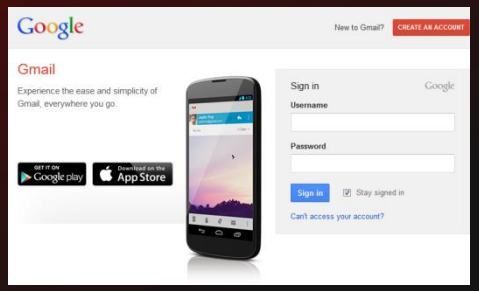


start = Date.now()

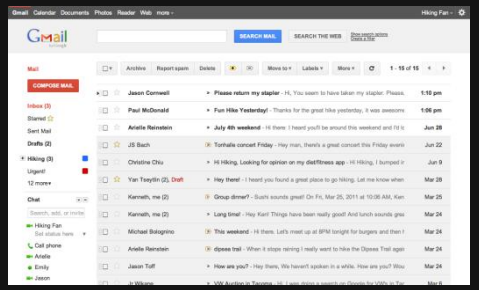
t1 = Date.now() - start

Page Request Timing

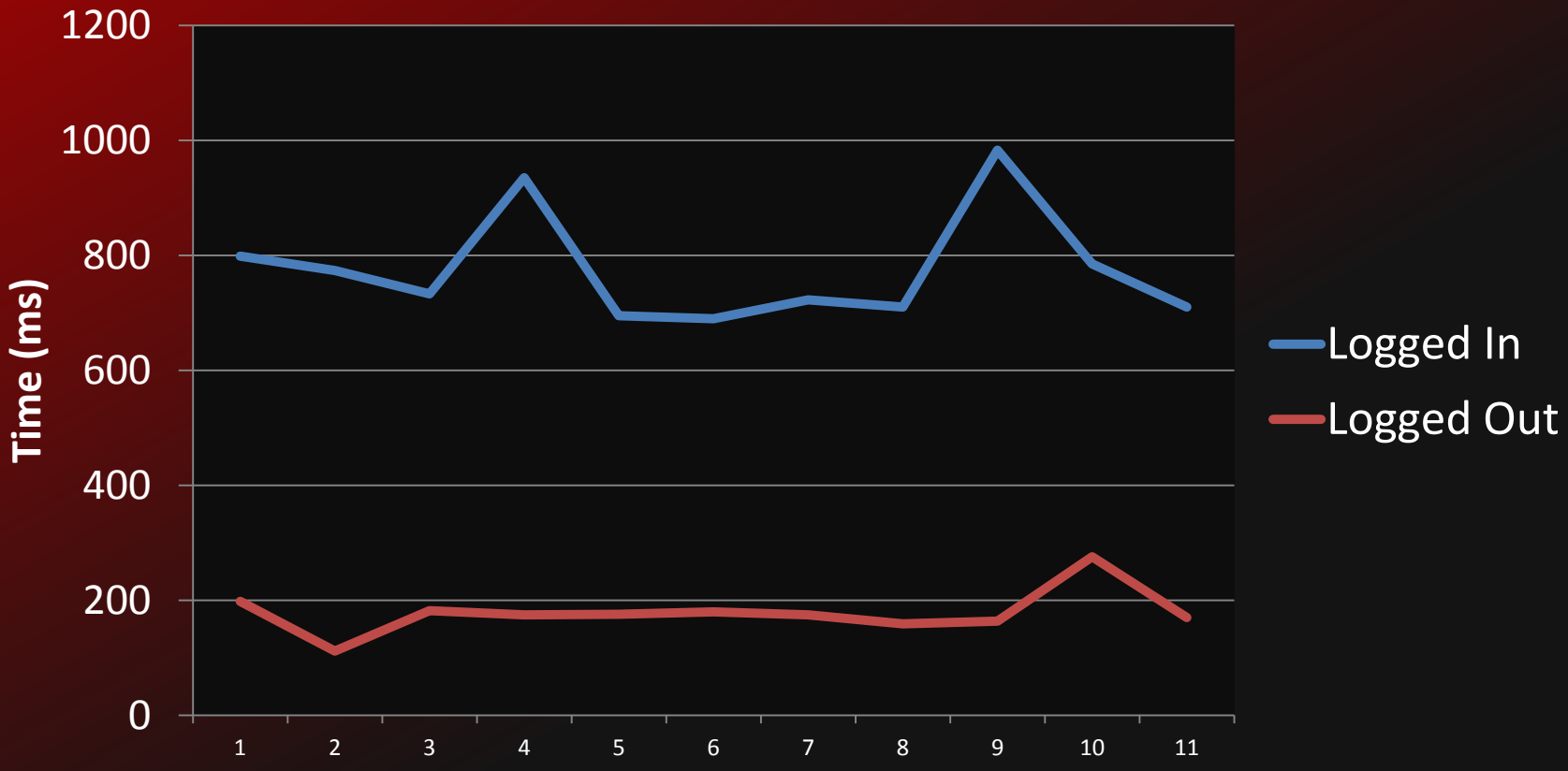
URL	Status	Domain	Size	Remote IP	Timeline
GET mail	302 Moved Temporarily	mail.google.com	352 B	173.194.34.86:80	137ms
GET Servic	200 OK	accounts.google.com	24.1 KB	173.194.66.84:443	284ms
2 requests			24.4 KB	(24.1 KB from cache)	429ms



URL	Status	Domain	Size	Remote IP	Timeline
GET mail	302 Moved Temporarily	mail.google.com	352 B	173.194.34.86:80	55ms
GET Servic	302 Moved Temporarily	accounts.google.com	360 B	173.194.66.84:443	71ms
GET ?pli=1	302 Moved Temporarily	mail.google.com	445 B	173.194.34.86:443	79ms
GET ?pli=1	302 Moved Temporarily	mail.google.com	0 B	173.194.34.86:443	80ms
GET ?shva	200 OK	mail.google.com	12 KB	173.194.34.86:443	340ms
5 requests			13.1 KB		730ms



Page Request Timing



Page Request Timing

- ***Can*** I tell if you're logged into Gmail?
- I measure a time of 500 ms on your computer
- Is that logged in or not?

Timing Attack Problems

- Network latency, jitter
- Unknown baseline
  - How long does server take to respond?
  - How fast is the user's connection?
  - How fast is the user's computer
- Unstable local environment
  - Other running programs
  - Other open browser tabs
  - Other network traffic

Timing Attack Problems

- Network latency, jitter
- Unknown baseline
  - How long does server take to respond
  - How fast is user's connection?
  - How fast is user's computer
- Unstable local environment
  - Other running programs
  - Other open browser tabs
  - Other network traffic

Take multiple measurements

Calibrate against known target

Wait until idle

Part 1 – Sniffing History

CSS History Sniffing

- Long long ago… (before 2010)

```
<style>a         { color: blue }
       a:visited { color: red  } </style>

<a href="http://paypal.com"  id="l">

<script>
var link = document.getElementById('l');
window.getComputedStyle(link).color;
```

CSS History Sniffing

- Study in 2010 surveyed top 50,000 sites
- 485 inspected history via CSS
- 46 were confirmed to be doing history sniffing
- Sites were testing between 20 – 200 URLs

http://cseweb.ucsd.edu/~d1jang/papers/ccs10.pdf

CSS History Sniffing

CSS History Sniffing

- But now?
  - History sniffing is history…
  - Fix proposed by Mozilla in 2010
  - All browsers have implemented it
  - Can only change color of visited links, not text size, background image etc..
  - getComputedStyle will lie to you about link color!

History Sniffing 2013

- History sniffing was fun, let's bring it back!
- …using a timing attack

requestAnimationFrame

- Like setTimeout, but linked to refresh rate of display
- Registers a function that is called just before the next frame is painted
- Will be called back roughly 60 times per second (or every 16.66… ms)

*** not technically part of HTML5 – see http://www.w3.org/TR/animation-timing/**

requestAnimationFrame

- Can use it to measure frame rate of web page
- If JS or rendering is too slow, frame rate will drop
- Can rendering time be used for a timing attack?

Your Browser, In Slow Motion

Your Browser, In Slow Motion

2. JS inserts link onto page – www.google.com
3. Browser fires async query in history DB . . .

5. DB query completes
- URL is visited

1. Page begins loading

4. Browser paints link as unvisited

6. Browser re-paints link as visited

Your Browser, In Slow Motion

2. JS inserts link onto page – www.google.com
3. Browser fires async query in history DB . . .

5. DB query completes
- URL is ***not*** visited

1. Page begins loading

4. Browser paints link as unvisited

Detecting Repaints

- If we can detect repaints, we can determine if the link is visited
- …but requestAnimationFrame will do callback whether repaint has happened or not
- We need to ***slow down*** painting so we can detect it

Make Painting Slooooow

- text-shadow: 5px 5px 10px red

offset blur radius

www.google.com

www.google.com
www.google.com
www.google.com
www.google.com
www.google.com
www.google.com
www.google.com
www.google.com

*** This is a lie**

Detecting Repaints

Quick repaints – every frame is equal

16ms 16ms 16ms 16ms 16ms

Detecting Repaints

Slow repaints are now detectable

16ms **60ms** 16ms 16ms **60ms**

The Black Box Analogy (again)

- Page rendering is our black box
- Link URL is our input
- callback is our output
- Delay between frames is our timing data

History Sniffing Timing Attack #1

- For each URL:
  - Make N link elements with text-shadow
  - Use requestAnimationFrame to time next few frames
  - If 1 slow frame, then URL not visited
  - If 2 slow frames, then URL **is visited**

Chrome

- Chrome ***does not do*** async URL lookups
- Does lookup before paint
- But, will repaint if link href changes ***and*** new URL is visited

```
<a href="http://not.visited.xyz" id="l">

var link = document.getElementById('l');
link.href='http://www.google.com';
link.style.color='red';
link.style.color='';     // force restyle
```

History Sniffing Timing Attack #2

- Make N link elements with text-shadow
- For each URL:
  - Update link hrefs to URL
  - Time next frame with requestAnimationFrame
  - If frame was slow, link ***is visited***
  - Update link hrefs to non-visited URL

Link Painting

	Async DB Lookup	Repaint after href changes
	✔	✔
	✔	✘
	✘	✔

History Sniffing Timing Attack

- Practicalities:
  - Need to calibrate number of links and amount of blur for text-shadow
  - We can make links invisible
  - Chrome demo tests ~16 URLs / sec
  - Can we do better?

History Sniffing Timing Attack #3

- Display 1000 different URLs at once
- If repaint is detected, divide in two sets of 500 - A,B
- Display each set separately, check for repaints
- Continue testing + dividing until we get individual URLs

History Sniffing Timing Attack #3

- In IE10 we can test 1000 URLs in ~16 secs
- Roughly 60 URLs per second
- Interclick.com tested ~200 URLs in 2010
- Practical attack would take a few seconds

Part 2 - Reading Pixels

SVG

- Scalable Vector Graphics
- XML graphics format
  - <circle>, <rect>, <path>
- Supported by all recent browsers
- HTML5 allows mixing SVG and HTML

*** OK, technically SVG is a separate spec that predates HTML5**

SVG

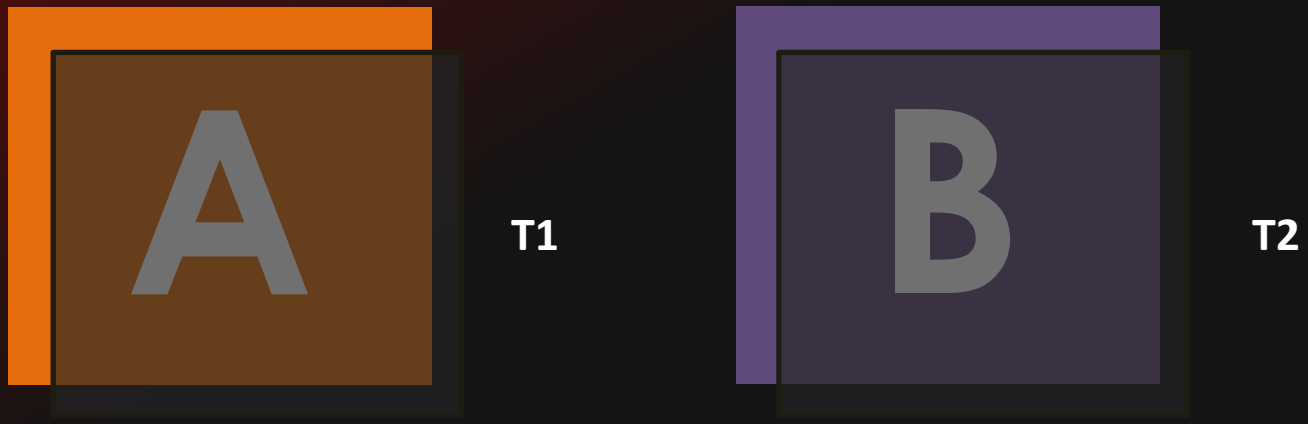
SVG Filter Effects

SVG Filter Effects

- 16 basic operations
  - Convolution, blur, displacement map…
- Combine filters to make fancy effects
  - bump mapping, drop shadow
- Alters element appearance only – JS cannot ‘see’ the result
- Can apply SVG filters to HTML elements!

SVG Filter Timing Attacks?

- SVG filters are complex algorithms
- We can apply a filter to any visual element of a webpage
- Can we find a filter that takes different times for different inputs?

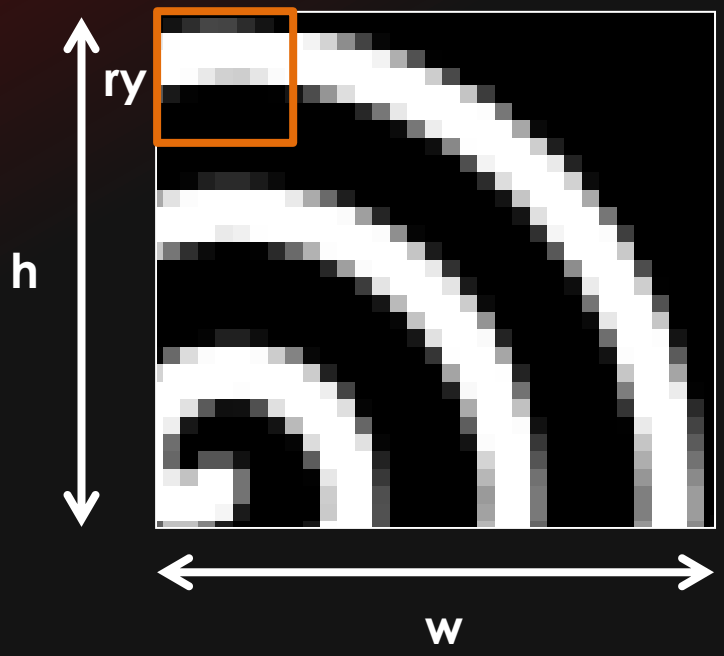


<feMorphology>

- Used to make lines thicker or thinner
- Takes a ‘radius’ parameter that controls the amount of erosion/dilation

<feMorphology>

- Must pass filter box over every pixel of source image
- Set each pixel to value of darkest/lightest pixel within filter box
- Naïve case – w × h × rx × ry comparisons



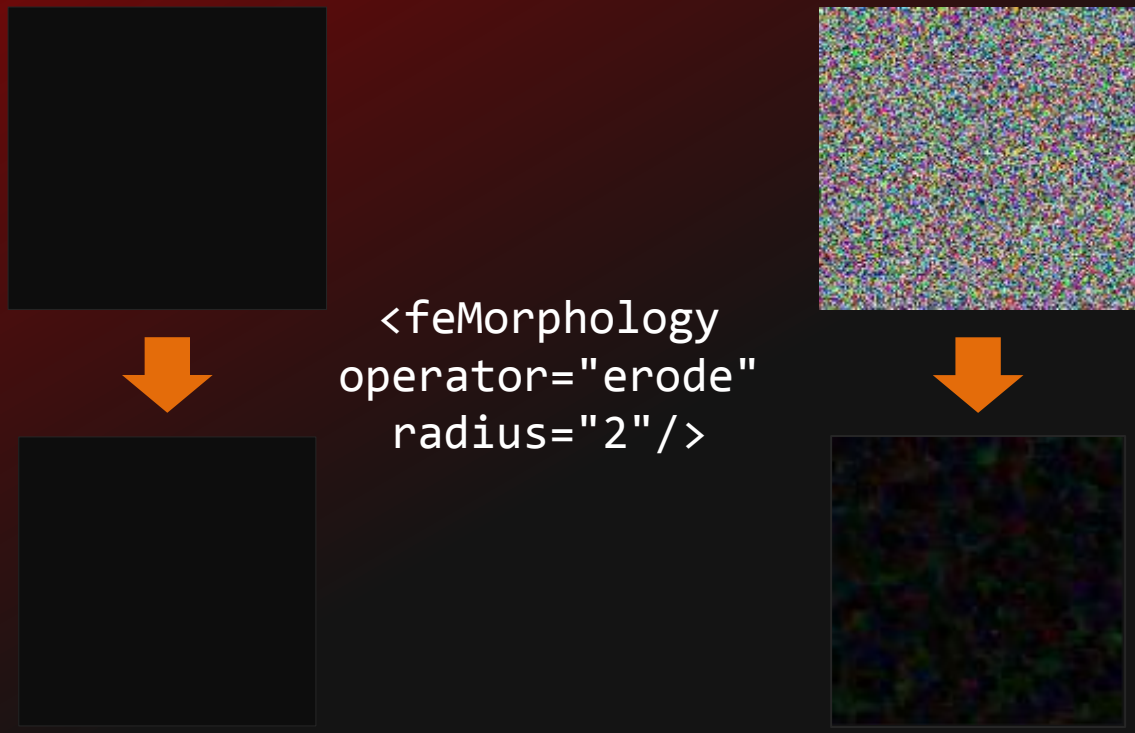
feMorpology - Firefox

```
// We need to scan the entire kernel
if (x == rect.x || xExt[0]  <= startX || xExt[1] <= startX ||
    xExt[2] <= startX || xExt[3] <= startX) {
  PRUint32 i;
  for (i = 0; i < 4; i++) {
    extrema[i] = sourceData[targIndex + i];
  }
  for (PRUint32 y1 = startY; y1 <= endY; y1++) {
    for (PRUint32 x1 = startX; x1 <= endX; x1++) {
      for (i = 0; i < 4; i++) {
        PRUint8 pixel = sourceData[y1 * stride + 4 * x1 + i];
        if ((extrema[i] >= pixel &&
             op == nsSVGFEMorphologyElement::SVG_OPERATOR_ERODE) ||
            (extrema[i] <= pixel &&
             op == nsSVGFEMorphologyElement::SVG_OPERATOR_DILATE)) {
          extrema[i] = pixel;
          xExt[i] = x1;
          yExt[i] = y1;
        }
      }
    }
```

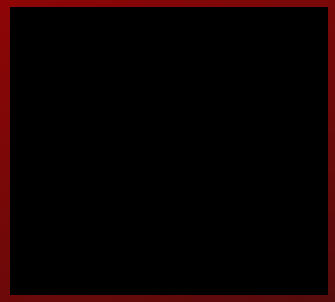
```
          xExt[i] = x1;
          yExt[i] = y1;
        }
      }
    }
  }
} else { // We only need to look at the newest column
  for (PRUint32 y1 = startY; y1 <= endY; y1++) {
    for (PRUint32 i = 0; i < 4; i++) {
      PRUint8 pixel = sourceData[y1 * stride + 4 * endX + i];
      if ((extrema[i] >= pixel &&
           op == nsSVGFEMorphologyElement::SVG_OPERATOR_ERODE) ||
          (extrema[i] <= pixel &&
           op == nsSVGFEMorphologyElement::SVG_OPERATOR_DILATE)) {
        extrema[i] = pixel;
        xExt[i] = endX;
        yExt[i] = y1;
      }
    }
  }
}
```

feMorphology

- Best case – w × h × ry comparisons
- Occurs in areas of flat colour

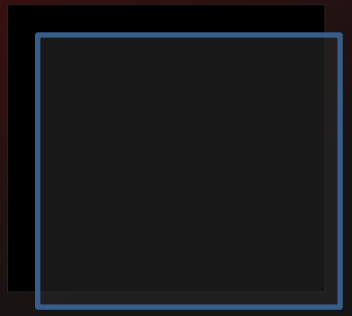


SVG Timing Attack Filter



×

=



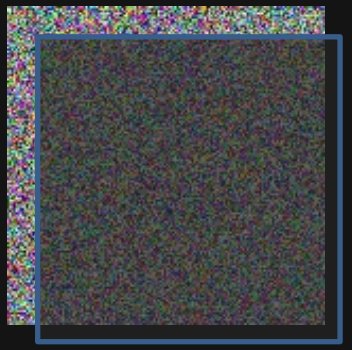
<feComposite
operator="multiply">

<feImage
xlink:href="noise.png">

<feMorphology>

×

=

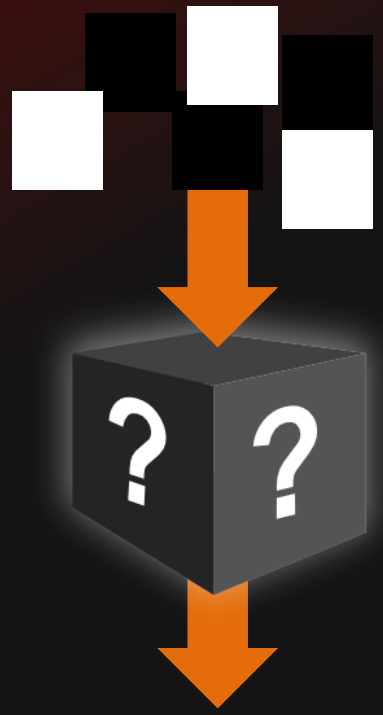


Reading Pixels

- Can we read pixels from iframes?
  - Crop an iframe to a single pixel (0,0)
  - Enlarge pixel by x100
  - Apply SVG filter
  - Time next frame with requestAnimationFrame
  - Move to next pixel (0,1)
  - Repeat for entire iframe

The Black Box Analogy (again)

- SVG filter rendering is our black box
- Pixels are our input
- callback is our output
- Delay between frames is our timing data



Reading Pixels

- SVG <pattern> and background: -moz-element(#el)
- Lets us take a ‘snapshot’ of elements, use as backgrounds
  - Avoids unpredictable timings unrelated to filters
- Apply ‘threshold’ filter to make pixels black or white
- CSS transform: scale(100) to zoom pixel
- Toggle filter to read pixel

Reading Pixels

- Works great!
- Very slow ☹

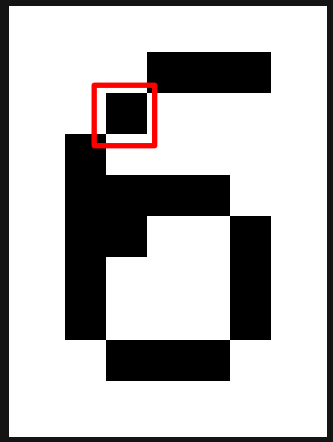
- Can we make some assumptions to speed this up?
  - Known font face, size
  - Fixed width font
  - Known location on page

Pixel Perfect OCR

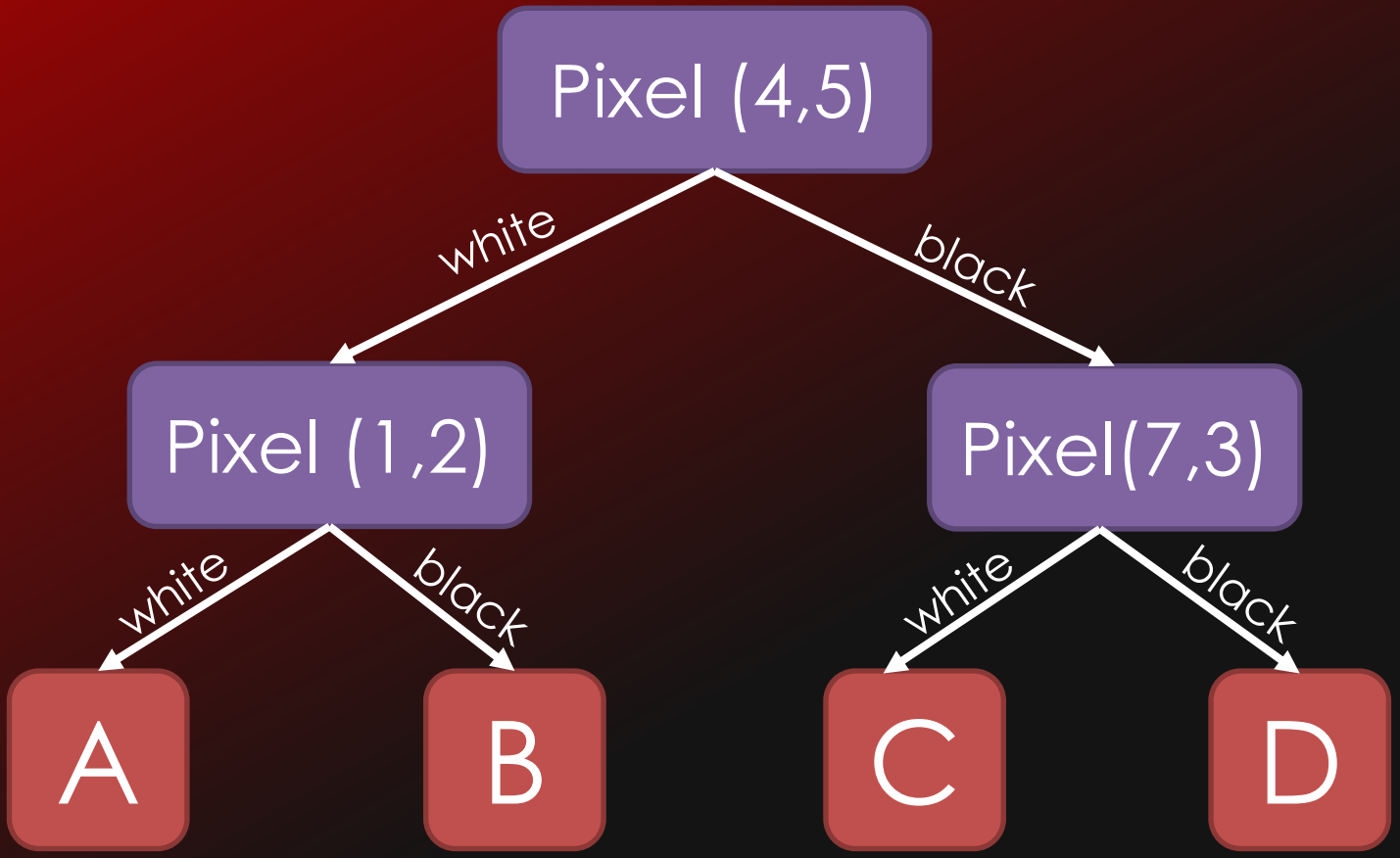
- What can we steal?
  - ‘Secret’ values in HTML source
  - <iframe src="view-source:http://…">
  - CSRF tokens!

Pixel Perfect OCR

- Are certain pixels unique to some chars?
- If this pixel is unique to ‘6’ then we know it’s a ‘6’
- What if there are no unique pixels for some characters?

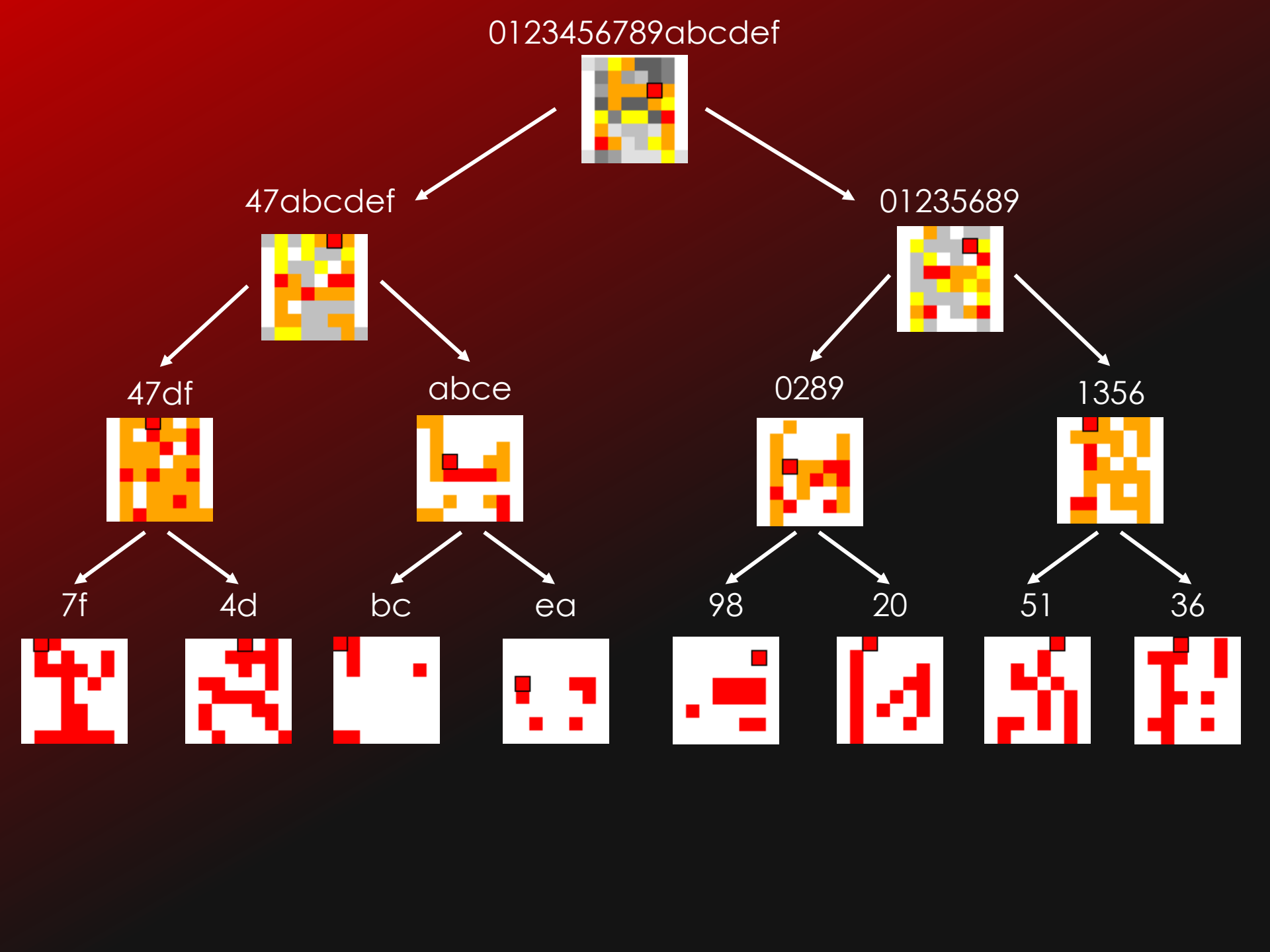


Pixel Perfect OCR – Binary Tree



Pixel-Perfect OCR

0123456789abcdef


0123456789abcdef
47abcdef
01235689
47df
abce
0289
1356
7f
4d
bc
ea
98
20
51
36

Pixel Perfect OCR – Binary Tree

- Can read character set of 2^n characters with n reads
- 16 characters -> 4 reads (hex chars)
- 32 characters -> 5 reads (a-z lowercase + punctuation)
- 64 characters -> 6 reads (base 64, most ascii text)

Pixel Reading

	Apply SVG Filters to HTML	view-source in iframes
	✔	✔
	✘	✘
	✔	✘

Fixing Timing Attacks

- Mozilla have fixed feMorphology in Firefox 22
- Preventing timing differences is tricky
  - Graphics code is performance critical
  - Compiler optimisations
  - CPU cache

- Other ways to prevent
  - Always redraw links – visited or not
  - Prevent filters from applying to iframes, links
  - Render iframes as blank, links as unvisited when applying filters

Fixing Timing Attacks

- Sites can protect themselves with X-Frame-Options
- Users can protect themselves by clearing history, using private browsing

Questions?

www.contextis.co.uk

@pdjstone