EPISODIO 2
994

Finalmente, llegaste hasta tu reclutador: Smith. Como ya sabes, él lleva tiempo operando al margen de los parámetros permitidos, por lo que ahora debéis enfrentar no sólo la amenaza de los rebeldes humanos, sino también las consecuencias de haber decidido desobedecer a Matrix.

Es por ello que hoy tendrás que acceder a información confidencial custodiada por las máquinas. Existe un lugar donde las máquinas registran información acerca de los rebeldes humanos que son liberados. Ellos no lo saben, pero la ecuación de Matrix permite a las máquinas anticipar la información del próximo rebelde que será liberado, antes incluso de que este hecho se produzca.

Es esencial que lleguemos hasta ese rebelde, antes de que lo hagan el resto de humanos o los centinelas de Matrix. Debemos reclutar a ese rebelde para nuestra causa.

El portal de acceso que te ha sido facilitado está relacionado de alguna manera con esa información. Sin embargo, estamos teniendo muchos problemas para poder penetrar en sus sistemas. Además, los mecanismos para llevar a cabo la conexión no son los habituales. Algo en el código parece estar alterando el modo en que se producen las conexiones.

¿Eres capaz de conseguir la información necesaria para anticiparnos a nuestros enemigos? Necesitamos saber la fecha en la que el próximo rebelde será liberado.

¡Cuidado! El servidor banea temporalmente a la gente que hace demasiado ruido.

Web: http://34.247.69.86/matrix/episodio2/index.php

Info: La flag tiene el formato UAM{md5}

Visitamos la página web a la que hacen referencia y nos da este resultado:

Id: 1
Nombre: Morfeo
Sexo: Varon

Nos damos cuenta de varias cosas:

Si accedemos a: http://34.247.69.86/matrix/episodio2/index.php?id=1

Y obtenemos:

Undefined hash

Visto esto, accedemos a:

http://34.247.69.86/matrix/episodio2/index.php?id=1&hash=1

Y obtenemos:

Hash error

Como vemos, los mensajes de error son diferentes, así que seguramente tenemos que averiguar qué hashes tienen que tener los diferentes ids.

El source de la web es el siguiente:

```
<html>
<head>
<script src="index.min.js"></script>
<script src="main.js"></script>
</head>
<body>
Id: 1<br>Nombre: Morfeo<br>Sexo: Varon
<br><br></body>
```

</html>

Vamos a mirar los diferentes js.

Index.min.js está ofuscado, por lo que podemos poner el código en un desofuscador de javascript, por ejemplo en: http://jsnice.org/

Con esto obtenemos el código más claro para poder analizarlo mejor.

Por otra parte, main.js se encarga de cargar un fichero: main.wasm, en definitiva un WebAssembly https://en.wikipedia.org/wiki/WebAssembly

Podríamos descargarlo, pasarlo a código C y examinarlo con más tranquilidad, pero realmente no es necesario al final. Al final el uso que se le da es el de utilizar funciones compiladas en el para diferentes tareas …

Viendo con más detenimiento Index.min.js vemos esta función que nos llama la atención:

```
function doIt(groupField) {
  var removedGroup = OMG(groupField);
  var plot_script = "0x" + removedGroup[_0x3358("0x13")](0, 8);
  var ret = "0x" + removedGroup[_0x3358("0x13")](8, 8);
  var thrown = "0x" + removedGroup[_0x3358("0x13")](16, 8);
  var status = "0x" + removedGroup[_0x3358("0x13")](24, 8);
  return Module[_0x3358("0x14")](plot_script, ret, thrown, status);
}
```

Con la Consola de Google Chrome podemos ver que hace cada cosa … por ejemplo podemos ver que OMG es una función que hace un md5 del string que le llega, luego veríamos que el md5 lo parte en 4 partes y lo envía a la función _calc que esta compilada en el wasm …

Lo importante de aquí es que si en la consola por ejemplo hacemos:

doIt("34.247.69.86/matrix/episodio2/index.php?id=1")
-1758453311

* La string a procesar con doIt la sabemos gracias al Hint que se puso en la prueba

En un primer momento no me cuadraba el valor negativo y el hash ese me daba igualmente error, después vemos que si hacemos:

dolt("34.247.69.86/matrix/episodio2/index.php?id=1")>>>0
2536513985
El operador >>> en javascript es: `>>>` is a right shift *without sign extension*

Así pues, si probamos con la url:
http://34.247.69.86/matrix/episodio2/index.php?id=1&hash=2536513985

Vemos que el resultado es el esperado, y nos muestra los datos del Id 1 de la base de datos.
Si repetimos el proceso para el Id 2, Id 3 .. vemos que todos contienen datos ..
En este momento podemos hacer consultas en la base de datos, pero seguramente haya un Id oculto con la información que nos piden .. aquí podríamos crear un script que fuera incrementando el Id y calculando el hash, pero tenemos el problema que el servidor web nos bloquearía, como pone el enunciado.

Tenemos que buscar algun bug para poder listar todos los datos que contiene la base de datos, probamos una inyección SQL … etc .. y lo que nos da resultado es una inyección NOSQL.

Para ilustrarse podemos mirar:
https://github.com/ctfs/write-ups-2014/tree/master/defkthon-ctf/web-200

Así pues probamos con:
http://34.247.69.86/matrix/episodio2/index.php?id[$ne]=1&hash=2536513985

Id: 2
Nombre: Trinity
Sexo: Mujer

...

Id: 7
Nombre: Mujer de rojo
Sexo: Mujer

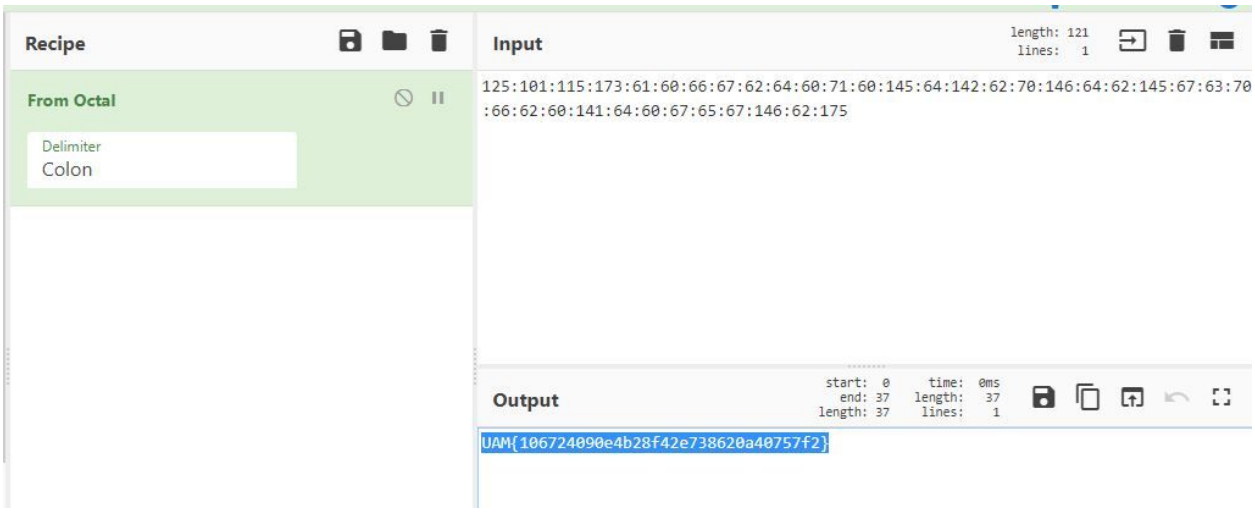
Id: 57069
Nombre:
125:101:115:173:61:60:66:67:62:64:60:71:60:145:64:142:62:70:146:64:62:145:67:63:70:66:62:60:141:64:60:67:65:67:146:62:175
Sexo: XXX

El resultado es que nos deja ver todos los datos de la base de datos, en especial nos interesa el Id 57069.

Lo primero que pruebo es pasar Decimal a ASCII, cosa que da una string que no se interpretar.

Fijándonos más atentamente en las numeraciones, vemos que no hay ningún número por encima de 7, por lo que parece ser código Octal.

https://gchq.github.io/CyberChef



Flag: UAM{106724090e4b28f42e738620a40757f2}

Found : **quinc3d3m4y0d3l2019**
(hash = 106724090e4b28f42e738620a40757f2)

DarkEagle