Propuesta de solución del reto Una-al-mes de septiembre de 2019 de Hispasec por Ramón Sola (@asterixco)

Tras el descanso de agosto, UAM regresa con un nuevo desafío inspirado en **Bola de Dragón**.

Enunciado

El segundo reto se presenta de esta forma:

> Una sospecha tiene a Trunks bastante preocupado. Según cree, una mano negra está controlando la empresa de su familia, Capsule Corp. Jacu, el patrullero galáctico, le ha informado de que investigando este servidor podrá confirmar sus sospechas y descubrir quién mueve los hilos en Capsule Corp.
>
> Ayuda a Trunks a desvelar el misterio que se encuentra en el siguiente servidor.
>
> Info:
>
> http://34.253.120.147:5002/api/list ftp://34.253.120.147:21

Estudio de las API del servicio web

La petición del recurso http://34.253.120.147:5002/api/list arroja la siguiente respuesta:

```
"[{\"http://34.253.120.147:5002/api/friends\": \"True\"},
{\"http://34.253.120.147:5002/api/users\": \"True\"},
{\"http://34.253.120.147:5002/api/zabbix\": \"True\"},
{\"http://34.253.120.147:5002/api/zimbra\": \"True\"},
{\"http://34.253.120.147:5002/api/media\": \"True\"},
{\"http://34.253.120.147:5002/api/endpoints\": \"True\"}]"
```

Su tipo es JSON (*application/json*). Sin embargo, se observa una peculiaridad. Especificaciones antiguas de JSON, como la publicada en la RFC 4627, exigían que el elemento principal fuese un objeto o un *array* (lista). Las revisiones posteriores RFC 7159 y RFC 8259, "*The JavaScript Object Notation (JSON) Data Interchange Format*", indican que un documento JSON puede estar formado por cualquier tipo de valor: objeto, *array*, cadena, número y los identificadores literales **true**, **false** y **null**. Por tanto, una cadena de caracteres construida según las reglas de JSON constituye en sí misma un documento válido, cuestión semántica aparte.

Las URL incluidas en la respuesta envían los respectivos mensajes indicados en la tabla:

API	Respuesta
/api/friends	`"{\"You don't have friends\": \"True\"}"`
/api/users	`"{\"No users available\": \"True\"}"`
/api/zabbix	`"{\"Zabbix is not installed\": \"True\"}"`
/api/zimbra	`"{\"Zimbra is not installed\": \"True\"}"`
/api/media	`"{\"Naranja\": \"True\"}"`
/api/endpoints	`"{\"This is\": \"True\"}"`

Ningún resultado es especialmente elocuente. El porqué de no devolver directamente objetos o *arrays* JSON, sino decidir encapsularlos en cadenas, se encuentra abierto a numerosas interpretaciones e hipótesis. Por otro lado, el servicio FTP no admite acceso anónimo, lo que es bastante obvio. La necesidad de descubrir al mismo tiempo el usuario y la contraseña correctos, así como el retardo intencionado del servidor para su verificación, hace inviable atacar sus credenciales.

A estas alturas, y obviando el chiste de la media naranja, parece razonable que deba existir un recurso oculto que proporcione información más útil. En una ocasión anterior se probó la herramienta Gobuster, escrita en el lenguaje Go y disponible ya compilada para múltiples plataformas. Su modo de empleo más básico exige únicamente la URL que se usará como prefijo y un fichero de palabras; tan simple como lanzar `gobuster dir -u URL -w diccionario.txt`. Después de probar sin éxito varios diccionarios de uso corriente como `big.txt` o `common.txt` del repositorio de dirb, el desánimo hacía mella.

Otro compañero de fatigas de Una-al-mes, mientras tanto, desvelaba una pista camuflada en la lista de las API a modo de acróstico: las iniciales de *friends*, *users*, *zabbix*, *zimbra*, *media* y *endpoints* formaban **FUZZ ME**. Realmente ese era el camino a seguir; tal vez se trataba tan solo de escoger el diccionario adecuado.

En lugar de seguir confiando en *Gobuster*, se decidió probar otra herramienta no usada hasta entonces. Wfuzz, que requiere el entorno de ejecución de Python, cumple el mismo propósito pero es mucho más versátil: no solo permite "atacar" una porción de una URL, sino también otros elementos de las peticiones HTTP como cabeceras, parámetros, cuerpos de peticiones POST, *cookies*, etc. También puede transformar de diversas maneras la "carga útil" (*payload*), indicada con la palabra clave **FUZZ**, mediante codificadores (*encoders*), y aplicar diversos filtros a las respuestas para conservar o destacar solamente las que interesen. En realidad, no hace falta tanta sofisticación para este reto.

El uso más elemental de *Wfuzz* no difiere mucho de *Gobuster* u otros: `wfuzz -w diccionario.txt http://34.253.120.147:5002/api/FUZZ`. Por una parte, la opción `-w` es una notación simplificada para el tipo **file** de carga útil, quizá el más común. Se pueden consultar las demás posibilidades con `wfuzz -e payloads` y `wfuzz -z help`. Por otra, la palabra **FUZZ** se sustituirá por cada elemento del diccionario.

Sin embargo, se observa inmediatamente que la pantalla se llena de resultados con error 404; para evitar mostrarlos, hay que añadir `--hc 404`. Además, se sabe que esas respuestas constan de 232 bytes; en el caso hipotético de que el servidor camuflase un recurso legítimo bajo la apariencia de un error 404, aunque con un mensaje de longitud diferente, la opción `--hh 232` descartaría los 404 genuinos.

Este filtrado manual se puede evitar de forma sencilla. Si a la palabra clave *FUZZ* se le agrega un término entre llaves, este se empleará en la primera petición y su resultado podrá tomarse como referencia (*baseline*) mediante la notación especial `BBB` en opciones como `--hh` o `--hc`. Aquí se ha tomado http://34.253.120.147:5002/api/xxx como ejemplo de petición de un recurso inexistente (error 404), por tanto la URL llevará la partícula `FUZZ{xxx}`.

Finalmente, tras probar otros ficheros de la rama *Discovery/Web-Content* del repositorio SecLists, la familia *raft-large* conduce a una conclusión satisfactoria. En este caso, se probó la línea de comandos `wfuzz -Z --hh BBB --hc BBB http://34.253.120.147:5002/api/FUZZ{xxx} -w raft-large-directories-lowercase.txt`. Sin la opción `-Z`, cualquier fallo transitorio de conexión podría haber interrumpido el proceso.

```
********************************************************
* Wfuzz 2.4 - The Web Fuzzer                           *
********************************************************

Target: http://34.253.120.147:5002/api/FUZZ
Total requests: 56166

=====================================================================
ID           Response   Lines    Word     Chars       Payload
=====================================================================
```

```
000000002:   404        4 L      34 W     232 Ch      "xxx"
000000010:   200        1 L      2 W      26 Ch       "media"
000000131:   200        1 L      4 W      37 Ch       "users"
000000565:   200        1 L      12 W     335 Ch      "list"
000000890:   200        1 L      5 W      41 Ch       "friends"
000052398:   200        1 L      2 W      26 Ch       "wnioski"
```

Los términos *media*, *users*, *list* y *friends* ya eran conocidos. De este modo, tras miles y miles de peticiones infructuosas, se consigue encontrar el recurso oculto *wnioski*, una palabra al parecer de origen polaco.

Gobuster también habría obtenido el mismo resultado: `gobuster dir -u http://34.253.120.147:5002/api/ -w raft-large-directories-lowercase.txt`.

```
===============================================================
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
===============================================================
[+] Url:            http://34.253.120.147:5002/api/
[+] Threads:        10
[+] Wordlist:       raft-large-directories-lowercase.txt
[+] Status codes:   200,204,301,302,307,401,403
[+] User Agent:     gobuster/3.0.1
[+] Timeout:        10s
===============================================================
2019/09/19 08:40:05 Starting gobuster
===============================================================
/media (Status: 200)
/users (Status: 200)
/list (Status: 200)
/friends (Status: 200)
/wnioski (Status: 200)
===============================================================
2019/09/19 08:49:21 Finished
===============================================================
```

Cuando se solicita http://34.253.120.147:5002/api/wnioski, el servidor responde con `"{\"TrUnK5\": \"5w0RD\"}"`. Esto concuerda con unas credenciales para el FTP: usuario **TrUnK5** y contraseña **5w0RD**.

```
Conectado a 34.253.120.147.
220 (vsFTPd 3.0.2)
Usuario (34.253.120.147:(none)): TrUnK5
331 Please specify the password.
Contraseña:
230 Login successful.
```

El servidor FTP ofrece un único fichero con apariencia de imagen PNG.

```
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
trunks.png
226 Directory send OK.
ftp: 12 bytes recibidos en 0,00segundos 12,00a KB/s.
ftp> get trunks.png
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for trunks.png (243198 bytes).
226 Transfer complete.
ftp: 243198 bytes recibidos en 0,22segundos 1125,92a KB/s.
ftp> quit
221 Goodbye.
```

Análisis de la imagen

El fichero *trunks.png* es una imagen de Trunks haciendo la peseta, un montaje que de acuerdo con las búsquedas inversas parece surgir de los tempestuosos submundos de Tumblr. En la imagen original, la mano mostraría los dedos índice y corazón juntos y extendidos.

Una inspección rápida con un visor o editor binario revela una sección de metadatos en forma de código XML.

```
<x:xmpmeta xmlns:x='adobe:ns:meta/' x:xmptk='Image::ExifTool 11.65'>
<rdf:RDF xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'>

 <rdf:Description rdf:about=''
  xmlns:dc='http://purl.org/dc/elements/1.1/'>
  <dc:rights>
   <rdf:Alt>
```

```
     <rdf:li xml:lang='x-default'>MM4N0N3GR4</rdf:li>
    </rdf:Alt>
   </dc:rights>
  </rdf:Description>
 </rdf:RDF>
</x:xmpmeta>
```

Llama la atención el término **MM4N0N3GR4**. La herramienta ExifTool confirma su presencia como parte de los metadatos.

```
File Type                       : PNG
File Type Extension             : png
MIME Type                       : image/png
Image Width                     : 500
Image Height                    : 358
Bit Depth                       : 8
Color Type                      : RGB
Compression                     : Deflate/Inflate
Filter                          : Adaptive
Interlace                       : Noninterlaced
SRGB Rendering                  : Perceptual
Gamma                           : 2.2
Pixels Per Unit X               : 3779
Pixels Per Unit Y               : 3779
Pixel Units                     : meters
XMP Toolkit                     : Image::ExifTool 11.65
Rights                          : MM4N0N3GR4
Image Size                      : 500x358
Megapixels                      : 0.179
```

Se sospecha que **MM4N0N3GR4** pudiera formar parte de unas credenciales de acceso. Si fuese así, la otra pieza debería estar escondida en la propia imagen. Queda descartado *steghide* porque no admite el formato PNG. El examen de las distintas capas de colores con ImageStegano y StegSolve desvela unas letras en la banda negra de la manga de Trunks. No está claro si el carácter central es la letra O o el cero: **CCORP** o **CC0RP**. Es necesario intentar las diferentes combinaciones, comenzando por el usuario **CCORP** (con la letra O) y contraseña **MM4N0N3GR4**. Fracaso.

```
Conectado a 34.253.120.147.
220 (vsFTPd 3.0.2)
Usuario (34.253.120.147:(none)): CCORP
331 Please specify the password.
Contraseña:
530 Login incorrect.
Error al iniciar la sesión.
ftp> quit
221 Goodbye.
```

Usuario **MM4N0N3GR4** y contraseña **CCORP** (con la letra O): también error.

```
Conectado a 34.253.120.147.
220 (vsFTPd 3.0.2)
Usuario (34.253.120.147:(none)): MM4N0N3GR4
331 Please specify the password.
Contraseña:
530 Login incorrect.
Error al iniciar la sesión.
ftp> quit
221 Goodbye.
```

Usuario **CC0RP** (con cero) y contraseña **MM4N0N3GR4**: negativo.

```
Conectado a 34.253.120.147.
220 (vsFTPd 3.0.2)
Usuario (34.253.120.147:(none)): CC0RP
331 Please specify the password.
Contraseña:
530 Login incorrect.
Error al iniciar la sesión.
ftp> quit
221 Goodbye.
```

Finalmente, la combinación del usuario **MM4N0N3GR4** con la contraseña **CC0RP** (con cero) es la correcta.

```
Conectado a 34.253.120.147.
220 (vsFTPd 3.0.2)
Usuario (34.253.120.147:(none)): MM4N0N3GR4
331 Please specify the password.
Contraseña:
230 Login successful.
```

¿La *flag*? No tan rápido.

```
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
flag.txt
226 Directory send OK.
ftp: 10 bytes recibidos en 0,00segundos 10000,00a KB/s.
ftp> get flag.txt
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for flag.txt (101 bytes).
226 Transfer complete.
```

```
ftp: 101 bytes recibidos en 0,00segundos 101000,00a KB/s.
ftp> quit
221 Goodbye.
```

El contenido de *flag.txt* da la impresión de estar codificado en Base64: `RXlWMW9hV2dHVGtPcndFZUR4cU9uYU9JRzJnbkhINGtESjFqQUtPVUh6Y2lxMUFjcFJnSlowU0lJd09PRjFManAwRXZDRGI9Cg==`. El camino parece fácil. Su decodificación produce otro mensaje en Base64: `EyV1oaWgGTkOrwEeDxqOnaOIG2gnHH4kDJ1jAKOUHzciq1AcpRgJZ0SIIwOOF1Ljp0EvCDb=`.

No obstante, el resultado de volver a decodificar es ininteligible, por lo que se presume algún tipo de ofuscación adicional. Se considera como opción más simple un cifrado César y en particular ROT13. Tras aplicar sucesivamente este método y la decodificación de Base64, se llega a `HNZ{626n193pppq005779q10n1oqr74r45r4}`, un formato mucho más familiar.

Una última transformación con ROT13 arroja la ansiada *flag*: **UAM{626a193cccd005779d10a1bde74e45e4}**. El *hash*, según la página https://www.md5online.org/md5-decrypt.html, corresponde a la cadena **TimesGoesBySoSlowly**. Es posible que para Trunks el tiempo fluya de manera distinta, o tal vez le guste la música de Madonna.