Una-al-mes: Silicon Valley - Episodio#3 - Hispasec

Enunciado CTF:

Richard mandó a Gilfoyle montar un servicio oculto que mantuviera a flote "El Flautista" pero este ya no recuerda donde se encuentra. Gracias a dios, como buen sysadmin, siempre hace backup de todo su trabajo, pero se trata de backups un tanto peculiares... Gilfoyle guarda el trabajo que hace en archivos encriptados relacionados con temáticas que le gustan.

Tenemos el fichero que contiene información sobre el servicio. Necesitamos que extraigas la información, accedas al servicio y consigas la flag de UAM. ¡Mucha suerte!

Enlace de descarga: https://drive.google.com/open?id=1qTuI9VndJ24krrO8U1WF3JpS77M4M2hV

Info: La flag tiene el formato UAM{md5}

Resolución:

Nos bajamos el archivo del enunciado y vemos que es el archivo "Avengers Infinity War Soundtrack - DEP Stan Lee.wav".

Tras analizar el espectro con Sonic Visualiser por si oculta información, lo analizamos con un editor hexadecimal como Bless.

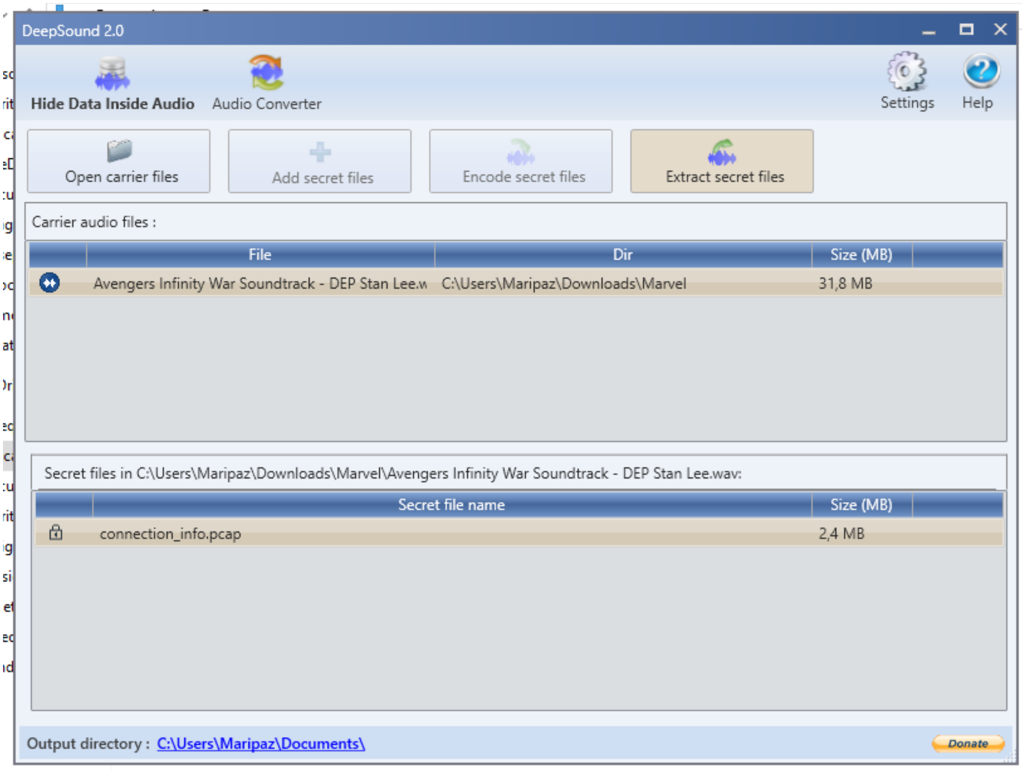
Al final del archivo encontramos la siguiente cadena:

```
U29uaWRvUHJvZnVuZG87KQo=
```

Si la decodificamos en base64 nos da una pista sobre la herramienta que tenemos que utilizar:

```
~$ echo "U29uaWRvUHJvZnVuZG87KQo=" | base64 -d
SonidoProfundo;)
```

Existe una herramienta para Windows llamada DeepSound. Si la usamos con el archivo .wav obtenemos una captura de tráfico de red "connection_info.pcap" oculta en el archivo de audio.



Carrier audio files :

File	Dir	Size (MB)
Avengers Infinity War Soundtrack - DEP Stan Lee.w	C:\Users\Maripaz\Downloads\Marvel	31,8 MB

Secret files in C:\Users\Maripaz\Downloads\Marvel\Avengers Infinity War Soundtrack - DEP Stan Lee.wav:

Secret file name	Size (MB)
connection_info.pcap	2,4 MB

Output directory : C:\Users\Maripaz\Documents\

Abrimos el archivo .pcap con Wireshark y analizamos el tráfico.

Como el enunciado nos habla de un servicio de Gilfoyle, me centro en el tráfico HTTP hacia direcciones IP públicas.

Pulsando sobre los paquetes HTTP y eligiendo la opción de Follow HTTP Stream, podemos ver la petición y la respuesta que se realiza sobre un servidor en el path /server que devuelve en el cuerpo de la respuesta lo que parece un código morse:

```
GET /server HTTP/1.1
User-Agent: Wget/1.17.1 (linux-gnu)
Accept: */*
Accept-Encoding: identity
Host: 192.168.116.130
Connection: Keep-Alive

HTTP/1.1 200 OK
Date: Wed, 14 Nov 2018 16:50:15 GMT
Server: Apache/2.4.18 (Ubuntu)
Last-Modified: Wed, 14 Nov 2018 16:48:15 GMT
ETag: "a6-57aa2b358be7a"
Accept-Ranges: bytes
Content-Length: 166
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive

...-- ....- .--. ..- -. - --- ..--- ....- --... .--. ..- -. - --- -.... ----. .--. ..- -. - --- ---.. -.... -.. ---
... .--. ..- -. - --- ... .---- ...-- ...-- --...
```

1 client pkt(s), 1 server pkt(s), 1 turn(s).

Entire conversation (573 bytes)

Show and save data as ASCII

Find:

Usando la herramienta http://www.unit-conversion.info/texttools/morse-code/ para convertir el código morse a texto plano, obtenemos el siguiente mensaje:

```
# Código Morse
...-- ....- .--. ..- -. - --- ..--- ....- --... .--. ..- -. - --- -....
----. .--. ..- -. - --- ---.. -.... -.. --- ... .--. ..- -. -
--- ... .---- ...-- ...-- --...

# Texto plano
34punto247punto69punto86dospuntos1337

# Dirección del servidor
34.247.69.86:1337
```

Es decir, tenemos una dirección IP y un puerto donde se encuentra el servidor de Gilfoyle.

Al hacer la petición en el browser, recibimos un ERR_INVALID_HTTP_RESPONSE.

Así que usamos *curl* para hacer la petición y vemos que nos está sirviendo en streaming algún archivo binario:

```
~$ curl 34.247.69.86:1337
#��CSf�C��W���#�V�+C##��U�F"L���z����O����Q���>���ۓ�1XQ2<|Qd�X�J�^�
ئس#��X�G#bn#χ��C#�##�@髮�Q�Z�a#��#�i��\<h��Z�8���=��-�#���n[W.�#15L��G�┐Y|��m�
(salida truncada)
```

Procedo a descargar lo que parece un archivo:

```
~$ wget -O ~/Documentos/CTF/SiliconValley3/response 34.247.69.86:1337
--2018-11-16 14:00:32--  http://34.247.69.86:1337/
Conectando con 34.247.69.86:1337... conectado.
Petición HTTP enviada, esperando respuesta... 200 Sin cabeceras, supondremos
HTTP/0.9
Longitud: no especificado
```

Tras esperar un rato, el archivo no deja de crecer, así que detengo la descarga y analizo el archivo parcialmente descargado con el comando *strings*.

[Luego entendí que no era un archivo lo que nos estaba sirviendo, sino un *stream* infinito de datos ¬¬']

```
strings response | grep UAM

UAM:OWY5MTBhNjNiMGRlNWMzNjM4YTA3MTg4MzFiN2JkODk0MGYxN2EyZjZjYTQ4MTE2MDVlYmU0NGMw
ZjNkYjJiNmI2YzQzZjU1NmZhYjYwMWZ8a2V5OjFZRUFS
```

Vemos que aparece repetidamente esta cadena que empieza por UAM, que decodificamos en base64:

```
$ echo
"OWY5MTBhNjNiMGRlNWMzNjM4YTA3MTg4MzFiN2JkODk0MGYxN2EyZjZjYTQ4MTE2MDVlYmU0NGMwZjN
kYjJiNmI2YzQzZjU1NmZhYjYwMWZ8a2V5OjFZRUFS" | base64 -d

9f910a63b0de5c3638a0718831b7bd8940f17a2f6ca4811605ebe44c0f3db2b6b6c43f556fab601f
|key:1YEAR
```

Es una cadena en hexadecimal y nos proporcionan una key. Probamos con XOR y diferentes cifrados y codificaciones simétricas, hasta que con **Blowfish** conseguimos el flag usando esta herramienta online: https://webnet77.net/cgi-bin/helpers/blowfish.pl

```
UAM{227218a71146ab9dc6ac28e5ec50a635}
```

Rafa Martos
@elbuenodefali