EPISODIO 3

300

Con todo el dinero robado, necesitamos escapar dando una distracción a la policia. Para ello, hace falta encontrar la bomba programada en el firmware del sistema informático. Una vez resuelta, podremos acceder al servidor, donde tras buscar bien, conseguiremos la flag final y escaparemos con el premio.

Info: La flag tiene el formato UAM{md5}

TOP 3: 1. 2. 3.

Nos descargamos el fichero firmware.zip y lo descomprimimos y hacemos un file para obtener más información del fichero:

 # file backup.raw

backup.raw: Linux rev 1.0 ext4 filesystem data, UUID=046b9ae6-97df-49fd-8785-2c68de053b05 (needs journal recovery) (extents) (large files) (huge files)

Montamos el sistema de ficheros y vamos a recorrer los directorios. Cuando nos hemos familiarizado con los ficheros allí presentes vemos que hay un ejecutable interesante. Si no queremos buscar por todos los directorios también podemos mirar directamente los ejecutables de la partición con:

 # find . -type f -executable -print

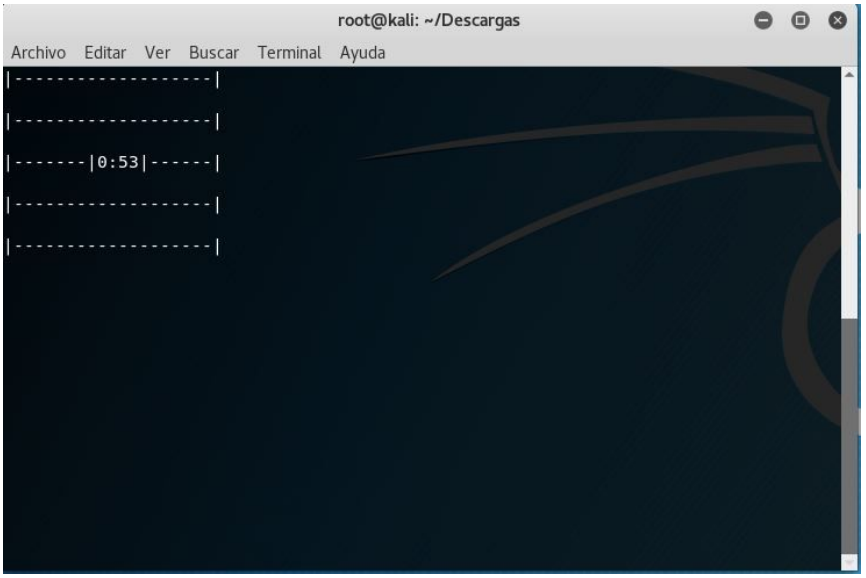
./lib/pushit1.6.7

./lib/libcrypto-1-dev.dev

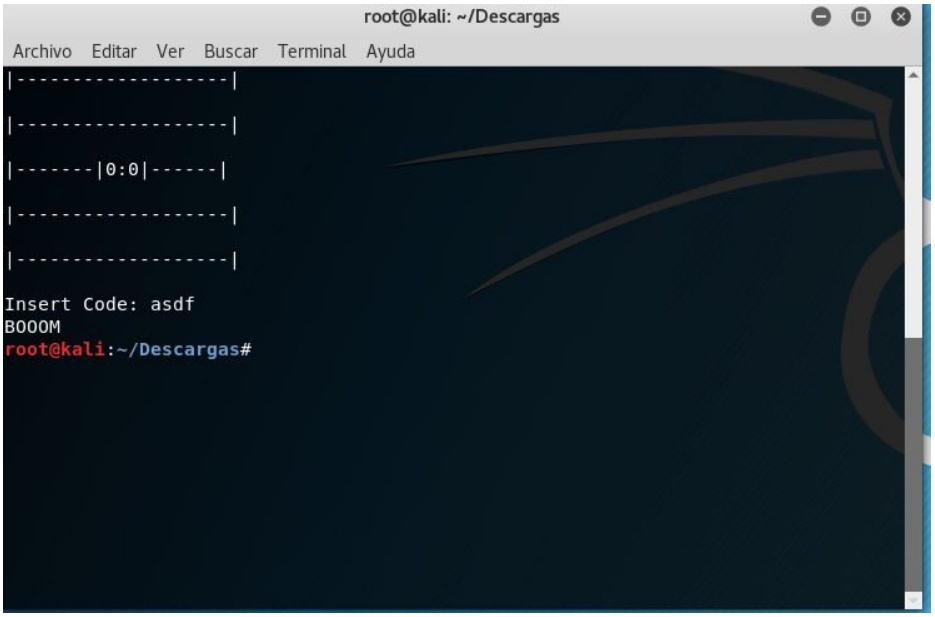
./lib/varr_lib5.1
./lib/resolve9.0.1.5
./lib/uam5.1
./lib/bomb.key
./lib/bomb.0
./lib/flagnothere0.1.2.3
./lib/libcurl64
./lib/bomb.resolve
./lib/unalmesplatform6.0
./lib/kill9.1
./lib/curllib64
./.bomb
./sbin/ls
./sbin/def
./sbin/sec
./sbin/arm
./boot/grub/locale/.bomb

Vemos que hay dos ficheros llamados .bomb y si miramos el hash que tienen, vemos que son el mismo fichero .. así que vamos a ejecutarlo.

./.bomb

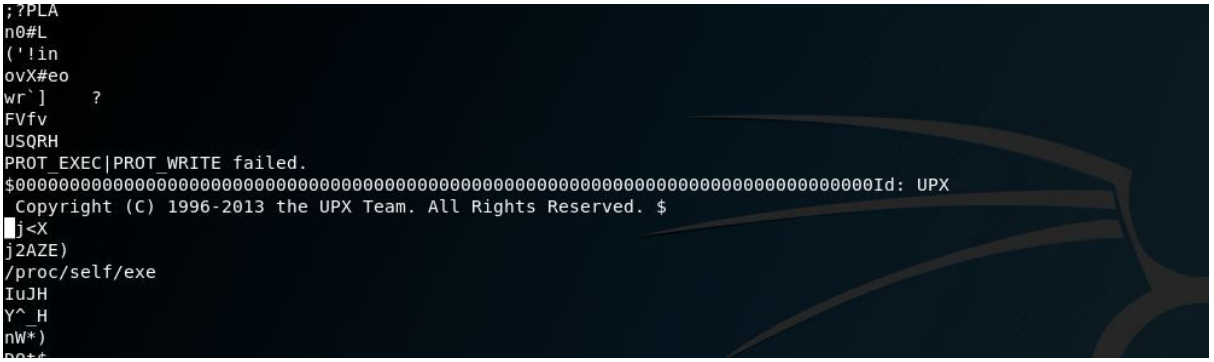


Un contador .. cuando llega a cero pide insertar un código.



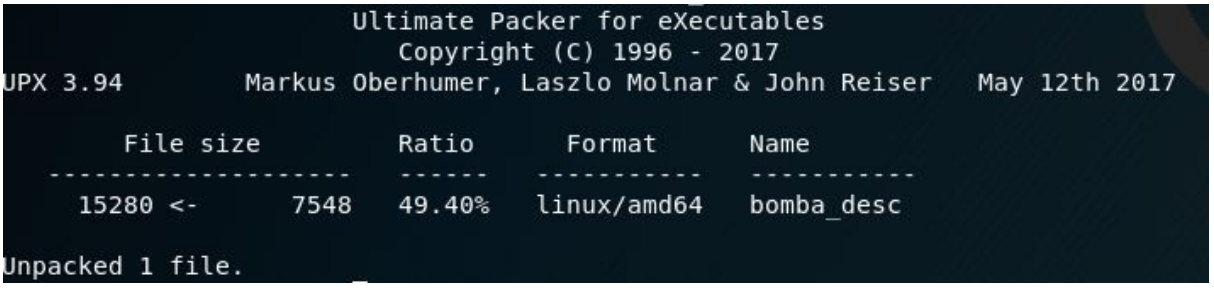
Ok, vamos a mirar los strings del ejecutable:

 # strings .bomb



Vemos que el ejecutable está comprimido con UPX por lo que lo descomprimimos y volvemos a mirar los strings

 # upx -d .bomb -o bomba_desc



En las strings del ejecutable descomprimido vemos esto:

```
_dbf7c981d7e_fe8
_c462eab3c39
_f2b06_fd
Tienes 1 minuto
clear
|------------------|
|-------|
|------|
Insert Code:
italy
BOOOM
```

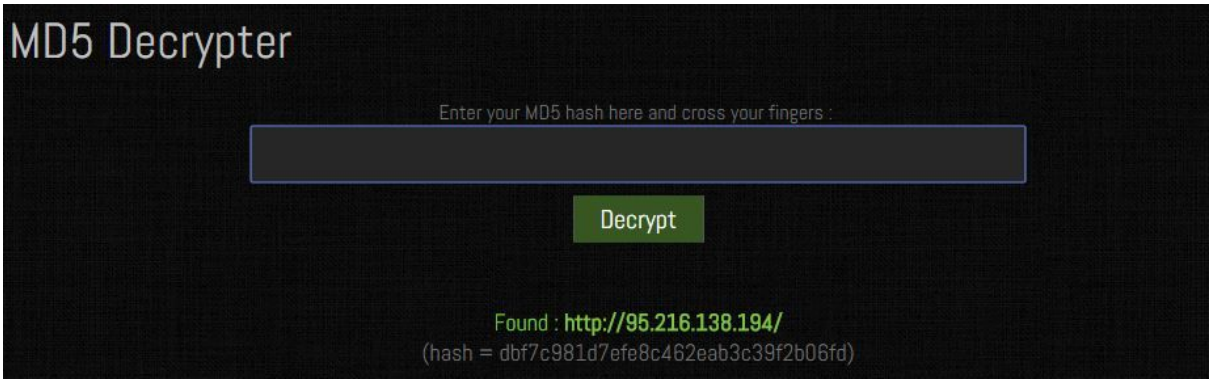
Probamos con el string “italy” como código y el resultado que nos devuelve el programa es justamente los strings que hay más arriba: _dbf7c981d7e_fe8_c462eab3c39_f2b06_fd

Extra: Otra forma de obtener lo mismo es usar radare2 para analizar el ejecutable descomprimido. En la función main (s main) vemos:

Vemos el codigo “italy” en claro con radare2

Una vez aquí .. voy probando muchas cosas … buscando en la plataforma del CTF … buscando en el servidor que siempre habían usado los creadores del reto (un servidor en Amazon) …. Intentando convertir ese string a una url .. pero no avanzo. Paralelamente examino mejor el sistema de ficheros y rescato todos los ficheros borrados … utilizo binwalk en varios ficheros … hexdump … radare2 … no doy con la tecla hasta que me fijo mejor en el string que me había dado la “bomba”.

Analizo que está compuesta por números .. y letras de la a-f … si le saco los “_” tenemos un posible hash md5 de manual, 32 chars hexadecimales.
Primero pruebo en https://crackstation.net/ y no me da ningún resultado … pero en https://www.md5online.org/md5-decrypt.html sí tenemos un resultado positivo:



Ok, ya tenemos el servidor pero al acceder por web no encontramos con esto:

Bad Request

Your browser sent a request that this server could not understand.
Reason: You're speaking plain HTTP to an SSL-enabled server port.
Instead use the HTTPS scheme to access this URL, please.

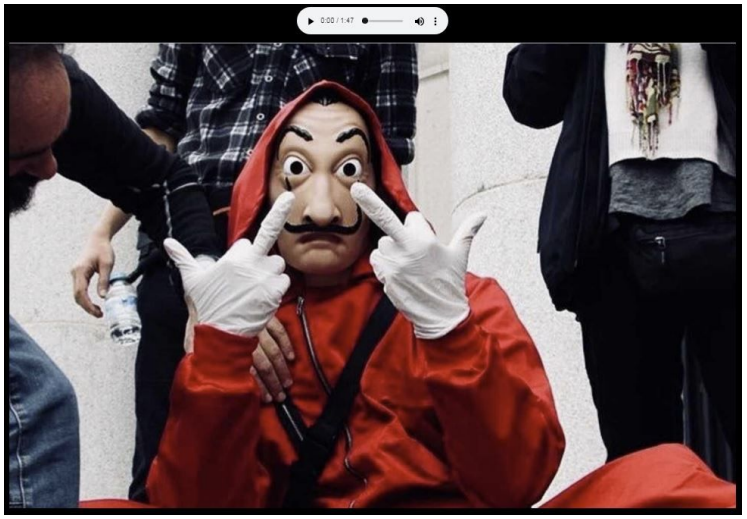
Apache/2.4.25 (Debian) Server at 127.0.1.1 Port 80

Después de alguna búsqueda por internet, vemos que el problema puede estar con el Servername y el certificado SSL … probamos con el host al que apunta la ip:

 # nslookup 95.216.138.194

Non-authoritative answer:
194.138.216.95.in-addr.arpa name = static.194.138.216.95.clients.your-server.de.

Accedemos por el host al servicio HTTPS a ver que pasa:



Y miramos el código fuente:

```
<html>
<head>
    <title>La casa de papel</title>
</head>
<body bgcolor="black">
<!-- <audio src="audio/Bella_Ciao.mp3"></audio> -->
<center>
    <audio controls>
        <source src="audio/Bella_Cia0.wav" type="audio/mp4">
    </audio>
</center>
<br />
<center>
    <img src="images/back.jpg" />
</center>
</body>
</html>
```

Descargamos los dos audios (uno comentado en el código html):

 # wget https://static.194.138.216.95.clients.your-server.de/audio/Bella_Ciao.mp3 --no-check-certificate

 # wget https://static.194.138.216.95.clients.your-server.de/audio/Bella_Cia0.wav --no-check-certificate

Una vez aquí, reproducimos la música desde los altavoces del PC y no se escucha nada raro en ese primer instante … utilizamos aplicaciones de stego … como por ejemplo: DeepSound, MP3Stego, Openpuff, visualizamos el espectro con Sonic Visualiser, AudaCity … no viendo nada extraño.

Volviendo al principio … tenemos dos audios, aparentemente son iguales (duración … espectro …) lo único que cambia es el formato, pero sobre todo el peso de los ficheros. El wav ocupa unos 36 MB y el mp3 unos 2.5 MB. Algo tiene que haber diferente ... Escucho otra vez los dos audios pero esta vez con unos cascos .. rápidamente apreció que en el .wav hay un código morse de fondo … esta vez fácilmente perceptible utilizando los cascos de más calidad que los altavoces del PC.

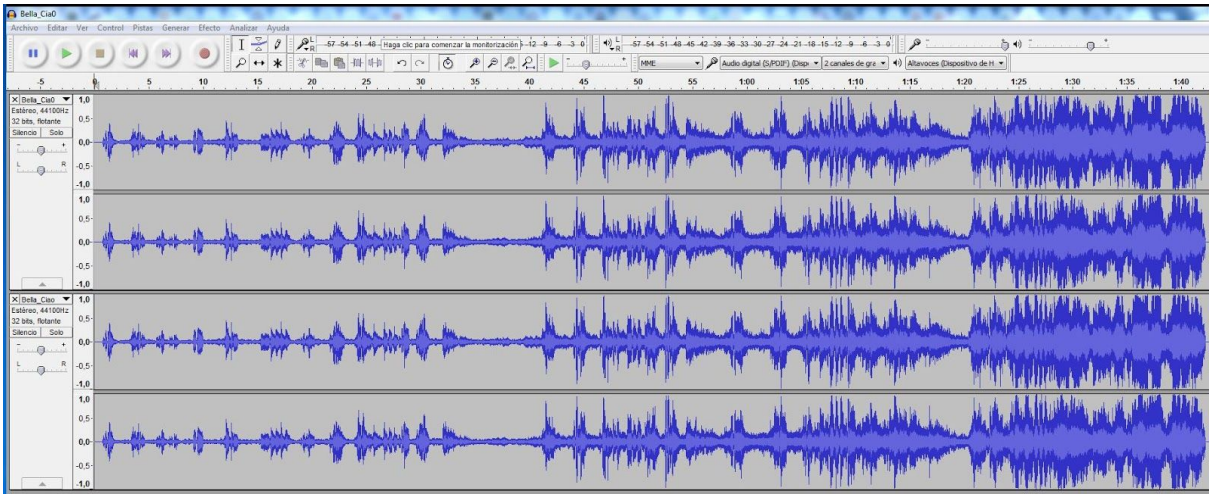
Cuando buscaba stego en los ficheros había pasado los audios por una web que escribe el mensaje por pantalla si la música tiene morse … el problema es que me daba resultados erráticos debido a la música y voces de fondo.

La web en cuestión es esta: https://morsecode.scphillips.com/labs/audio-decoder-adaptive/
Llegados a este punto, y sabiendo cual es el problema, escuchando la solución pero no pudiendo interpretar fácilmente el resultado empiezo a jugar con Audacity …

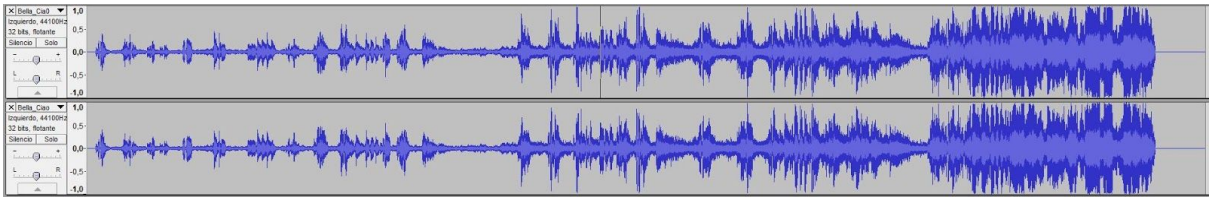
Intento quitar la voz,la música de fondo y solo dejar el morse, pero se me hace muy difícil con los filtros de Audacity, puesto que también me eliminan el código morse (al estar todo mezclado en el mismo canal).

Toca recapacitar otra vez .. al final tiene que ser algo fácil puesto que tenemos un audio donde esta el morse, y otro que no está. Debería ser tan fácil como quedarnos con lo diferente de los dos audios. Esto me hace pensar en la operación xor … La XOR representa la función de la desigualdad, es decir, la salida es verdadera si las entradas no son iguales, de otro modo el resultado es falso

Así pues, aprovecho que el audio es el mismo y lo que hago es, en Audacity, separar los canales de los audios y siempre trabajar con los canales izquierdos. Los canales derechos de los audios los podemos descartar. Una vez hecho esto, invertir uno de los audios. El resultado es la reproducción de lo que es diferente entre los dos audios, es decir el código morse.



Audios sin separar



Audios separados, solo canales izq. Uno invertido

Aspecto del audio resultante donde se ve claramente el cod. Morse

Una vez tenemos el morse separado .. podemos sacar el mensaje de forma manual o automática.

Manual:

Viendo la forma de onda resultante del código morse, podemos obtener el mensaje con ayuda de la siguiente web:

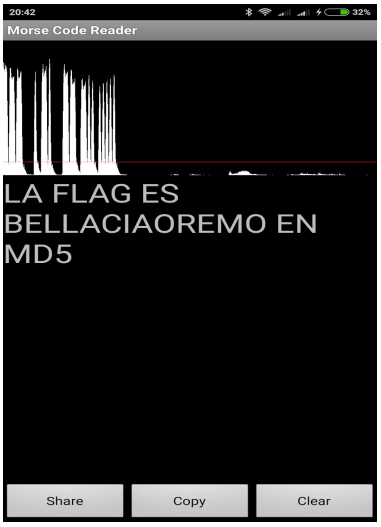
http://www.unit-conversion.info/texttools/morse-code/

Automática:

https://morsecode.scphillips.com/labs/audio-decoder-adaptive/

Tenemos esta web donde subimos el audio y nos dará el mensaje morse que contiene.

Yo opté por una tercera opción, también automática, que fue descargarme la app Morse Code Reader en Android, poner los altavoces del ordenador y reproducir el audio. Automáticamente la App en el móvil iba descifrando el mensaje.



Flag “bellaciaoremo” en MD5: f3b2c8d7436ccb3eaebc832c447f9051

PD: Se podía haber llegado a conseguir el morse de otras formas, como aplicar varios filtros en Audacity sin usar el otro audio, pero los creadores del reto seguramente nos dieron el otro audio para hacerlo así, de forma mucho más efectiva, limpia y rápida.

Al final un reto muy entretenido, divertido y algo novedoso para mí … con muchos callejones sin salida y muchas cosas que ir probando .. cuando al final era más fácil de lo que parecía.

DarkEagle