ACECTF WRITEUPS

FORENSICS

ACECTF / Broken Secrets

keywords: file, 7z, word, png, header, magic bytes

1. `$ file Brokenfr.7z`
2. `$ 7z x Brokenfr.7z`
3. `_/word/media/not_so_suspicious_file`
4. Open in hex editor and notice the magic bytes are wrong
5. fix and download > flag

ACECTF / Hidden in the traffic

```
$ python solve.py
~/Downloads/Very_mysterious_file.pcapng
Analyzing
/home/kilox/Downloads/Very_mysterious_file
.pcapng for hidden flags...

Checking ICMP payloads...
Found 19 potential ICMP data patterns

...

Pattern 12: Every 12th value
ASCII:
ALKJIHGFEDCBA0LKJIHGFEDCBACLKJIHGFEDCBADLK
JIHG
Hex:
414c4b4a4948474645444342413 04c4b4a494847..
.

Pattern 13: Every 13th value
ASCII:
ACECTF{p1n6_0f_D347h}ACECTF{p1n6_0f_D347h}
POTENTIAL FLAG FOUND:  {p1n6_0f_D347h}
Hex:
4143454354467b70316e365f30665f4433343768..
.

Pattern 14: Every 14th value
ASCII:
AABCDEFGHIJKL_ABCDEFGHIJKLpABCDEFGHIJKL
Hex:
4141424344454647 48494a4b4c5f414243444546..
.
```

pcap_flag_extractor.py

ACECTF / Virtual Hard Disk

keywords: image, ftk, vigenere

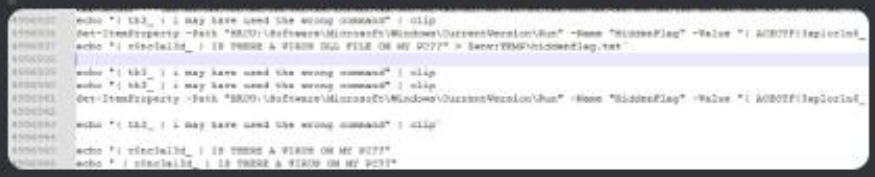
> 1. give the extension .img
> 2. open with FTK Imager
> 3. scroll until `666c61672e747874.jpg`
> 4. this file has Flag and Key
> 5. Vigenere to get the flag
> *. MANY FAKE FLAGS

ACECTF / Another Reading between the Lines?

```
with open('hidden', 'rb') as f:
    data = f.read()
binary = ""
i = 0
while i < len(data):
    if i+1 < len(data) and data[i] == 13
and data[i+1] == 10:  # CRLF (0x0D 0x0A)
        binary += "1"
        i += 2
    elif data[i] == 10:  # LF (0x0A)
        binary += "0"
        i += 1
flag = ""
for i in range(0, len(binary), 8):
    byte = binary[i:i+8]
    if len(byte) == 8:
        flag += chr(int(byte, 2))
print(flag)
```

ACECTF / Deep Memory Dive
keywords: memory dump

> 1. open the file in notepad and search for `ACECTF{` and i found a huge part of the flag.
> 2. Then using Volatility Workbench, i did a Filescan and found the last part `\Desktop\last_part_is_{r1ddl3s}.exe.exe`



Juke Yesterday at 22:22

ACECTF / FRACTURED FRAMES
keywords: resize

> I noticed the size of the image and 182x255 was weird so i tried to make it 241x255 (a little wider) and found the flag

ACECTF / Keyboard Echo

```
└─$ tshark -r challenge.pcapng -Y
"usb.capdata" -T fields -e usb.capdata >
capdata_dump.txt

┌──(kilox㉿kilox)-[~/Desktop/ Keyboard
Echo ]
└─$ cat capdata_dump.txt
09007409000f00000000ff00eb
09007509000f00000000ff00eb
0100000000000000
0100150000000000
0000150000000000
0000000000000000
0400000000000000
04002b0000000000
0520030105
09007609000f00000000ff00eb
09007709000f00000000ff00eb
00002b0000000000
09007809000f00000000ff00eb
09007909000f00000000ff00eb
09007a09000f00000000ff00eb
0000000000000000
00001c0000000000
```

```
09007b09000f00000000ff00eb
09007c09000f00000000ff00eb
0100000000000000
0100150000000000
0000000000000000
0400000000000000
04002b0000000000
0520050105
09007d09000f00000000ff00eb
09007e09000f00000000ff00eb
09007f09000f00000000ff00eb
09008009000f00000000ff00eb
0400000000000000
0000000000000000
09008109000f00000000ff00eb
09008209000f00000000ff00eb
```

0000150000000000 - q
00002b0000000000 - tab
00001c0000000000 - y
0000270000000000 - 0
0000180000000000 - u
00000b0000000000 - h
0000210000000000 - 4
0000190000000000 - v
0000200000000000 - 3
0000090000000000 - f
0000270000000000 - 0
0000180000000000 - u
0000110000000000 - n
0000070000000000 - d
00001e0000000000 - 1
0000240000000000 - 7

STEGANOGRAPHY

ACECTF / Cryptic Pixels
keywords: binwalk, john, rot

1. `$ binwalk CrypticPixels.png -e`
2. `$ zip2john chal.zip > hash.txt`
3. `$ john hash.txt --wordlist=/usr/share/wordlists/rockyou.txt` > `qwertyuiop`
4. ROT - 9

Juke Yesterday at 17:09

ACECTF / Tabs&Spaces
keywords: size, steghide, whitespace

1. `$ ls -la ctf/files/`
2. notice that the 87th image has different size
3. `$ steghide extract -sf test.jpg` with blank password > `whitespace_flag.txt`
4. simple solve script to get the flag

```
# Read the file containing the whitespace-base
with open('whitespace_flag.txt', 'r') as file:
    data = file.read()

# Replace spaces with '0' and tabs with '1'
binary_data = ''.join(['0' if char == ' ' else
```

Expand

solve.py 1 KB

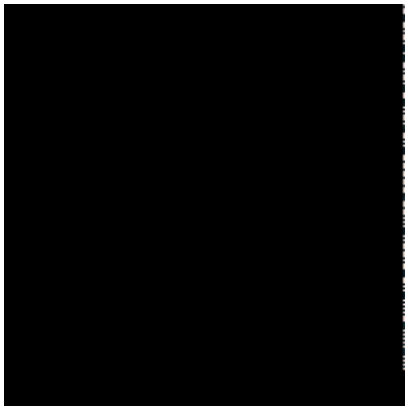
ACECTF / HeaderHijack
keywords: header, magic bytes

1. `$ cp not_the_flag.mp4 fixed.mp4`
2. `$ hexedit fixed.mp4`
3. Replace the first bytes with:
   `0000001c6674797033677034`
4. open `fixed.mp4` and see the flag

Juke Yesterday at 20:08
ACECTF / Whispering Waves
keywords: john, spectrogram, binary

1. unzip the zip using the wordlist given
2. open the wav file using sonic visualiser
3. Decode binary

DOUBLE VISION



XOR THE IMAGES AND MORSE CODE ON THE TOP RIGHT CORNER

REVERSE

acectf / rev / Significance of Reversing

```
$ # Reverse the file bytes
xxd -p Reverseme.png | tr -d '\n' | fold -w2 | tac | tr -d '\n' > reversed.bin

xxd -r -p reversed.bin > executable

chmod +x executable

./executable
```

acectf / rev / Piped Up

```
target = [
    0x6c, 0x2c, 0xe0, 0xef, 0x8d, 0x60, 0xdc, 0x75, 0x0d, 0xff,
    0xd6, 0x59, 0xf4, 0x5d, 0xde, 0x9b, 0xe3, 0xd7, 0x52, 0x99,
    0x5a, 0x7c, 0xa3, 0xc9, 0x4e, 0x1b, 0x45, 0xe5, 0xc0, 0x29, 0x9a
]
key = [
    0x7b, 0x2e, 0xf1, 0xeb, 0x8b, 0x76, 0xe7, 0x68, 0x77, 0xa3,
    0xef, 0x52, 0xf6, 0x3c, 0xda, 0xaa, 0xf6, 0xa7, 0x43, 0xeb,
    0x21, 0x24, 0xc3, 0x9c, 0x7d, 0x08, 0x33, 0xb7, 0xf7, 0x2c, 0xb4
]
n = 31

# T5⁻¹: XOR with repeating key
after_T5 = [target[i] ^ key[i % 31] for i in range(n)]
# T1⁻¹: Undo chained XOR
after_T1 = [after_T5[0]] + [after_T5[i] ^ after_T5[i-1] for i in range(1, n)]
# T3⁻¹: XOR with 0x56
flag = [x ^ 0x56 for x in after_T1]

print(''.join(chr(x) for x in flag))
```

(edited)

ACECTF / DONOTOPEN

keywords: Bourne-Again shell script, awk

1. `$ awk '/^__ARCHIVE_BELOW__/ {print NR + 1; exit 0; }' DONTOPEN | xargs -I {} tail -n +{} DONTOPEN | gzip -d > embedded_script.py`
2. `$ python3 embedded_script.py`
3. the script had this pin check: `ACE@SE7EN`
4. got the flag from the script

(edited)

```
└─$ python3 embedded_script.py
It looks like the box is locked with some kind of password, determine the pin to open the box!
What is the pin code?ACE@SE7EN
Looks good to me...
I guess I'll generate a flag
ACE{e2e3619b630b3be9de762910fd58dba7}
```

Trust issues:

```
input_filename = "Reverseme.png"
output_filename = "reversed_file.LEF"

with open(input_filename, 'rb') as infile:
    data = infile.read()

reversed_data = data[::-1]

with open(output_filename, 'wb') as
outfile:
    outfile.write(reversed_data)
```

The Chemistry of Code:

```
import base64

anionic_password =
"NjQzMzcyNzUzNTM3MzE2Njc5MzE2ZTM2"
decoded_hex =
base64.b64decode(anionic_password).decode(
'ascii')
password =
bytes.fromhex(decoded_hex).decode('ascii')


username = "AdminFeroxide"
ALKALINE_SECRET =
"4143454354467B34707072336E373163335F36343
22C28010D3461302C392E"

hex_username = username.encode().hex()
hex_password = password.encode().hex()


username_int = int(hex_username, 16)
password_int = int(hex_password, 16)
covalent_link = username_int ^
password_int
alkaline_secret_int = int(ALKALINE_SECRET,
16)
metallic_alloy = covalent_link ^
alkaline_secret_int


precipitate = hex(metallic_alloy)[2:]
if len(precipitate) % 2 != 0:
    precipitate = '0' + precipitate

alloy_bytes = bytes.fromhex(precipitate)
flag = alloy_bytes.decode('ascii')

print(f"Flag: {flag}")
```

OSINT

ACECTF / Fall of 2022 [Link] (https://whoisfreaks.com/tools/dns/history/lookup/acectf.tech?type=all)

escape to cancel • enter to save

Juke Yesterday at 16:11

ACECTF / The Mysterious Building

1. Chat GPT told me it is `Pitampura TV Tower`
2. Search for it and it was!!
3. check around to get the perfect view and got this
4. That was the Building

(edited)

Juke Yesterday at 18:09

ACECTF / For The Fans

1. search for the given username > Found X account
2. he said that he has a blog, so i tried searching many things > `"salty-senpai-drake1"` > Tumblr posts
3. Found his account
4. get file from found base64 here
5. we know that the password of the zip has 7 numbers so we try all possible combinations of the birthday given: `2000914`

SOCIAL CIRCLES

FIND THE YOUTUBE. TRANSLATE THE VIDEO. FIND THE NEW GUY https://www.smule.com/wimebix884 . GOOGLE DRIVE FLAG https://drive.google.com/file/d/1093uvDYSVWke8ze2jdgJ1rVehz51Jx00

OSINT BAND NAME AND SONG

FIND YOUTUBE CHANNEL AND GO TO HIS https://makromusic.com/u/modernlouis .SEE THE HINT AND GO TO https://myspace.com/modernlouis FOUND THE SPOTIFY LINK AND GOT THE FLAG https://open.spotify.com/user/313vqcsij2k5ukfgqwhu27sr4l64

CRYPTO

ACECTF / Super Secure Encryption

solve.py

ACECTF / Custom Encoding Scheme

solve.py

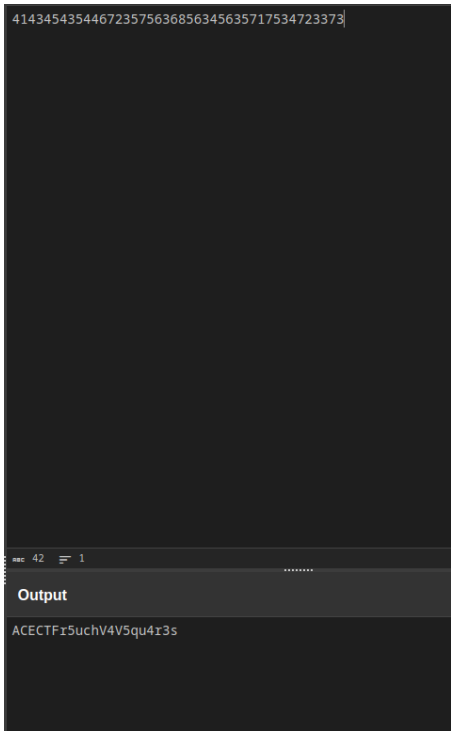
PIANO CIPHER

```
Ciphertext - DC# DD# DF DD# EC '70' G#B CE F#C FC# C#C# '104' C#A FC# F#A# C#A C#A '108' CF AF# C#C FC# CE '102' FC# C#A# FC# GA# CE '112' FC# C#B C#C# C#A# GC '125'
Plaintext  - A   C  E   C  T   F   {  0   h   _   7   h   3   _   f  3   3   l   1   n   6   _   0   f   _   4   _   p  0   p   _   5   7   4   r   }
ACECTF{0h_7h3_f33l1n6_0f_4_p0p_574r}

A  1
A# 2
B  3
C  4
C# 5
D  6
D# 7
E  8
F  9
F# 10
G  11
G# 12
```

HEXED AND SQUARED

```
>>> abcd = open("encoded.txt")
>>> res = []
>>> for x in range(0,len(dic)):
...     if dic[x] != '3':
...             res.append(x)
...
>>> res
[32767, 65535, 98303, 163839, 196607, 229375, 294911, 327679, 360447, 393215, 425983, 442367, 458751, 524287, 557055, 58982
3, 622591, 688127, 720895, 753663, 770047, 786431, 851967, 884735, 901119, 917503, 983039, 1015807, 1048575, 1081343, 11141
11, 1179647, 1212415, 1245183, 1343487, 1359871]
>>> 65535 - 32767
32768
>>> 98303 - 65535
32768
>>> 163839 - 98303
65536
>>> 196607 - 163839
32768
>>> x = 32767
>>> res4 = []
>>> while x < 1359880:
...     res4.append(dic[x])
...     x += 32768
...
>>> res4
['4', '1', '4', '3', '4', '5', '4', '3', '5', '4', '4', '6', '7', '2', '3', '5', '7', '5', '6', '3', '6', '8', '5', '6', '3
', '4', '5', '6', '3', '5', '7', '1', '7', '5', '3', '4', '7', '2', '3', '3', '7']
>>> haha = "".join(res4)
>>> haha
'414345435446723575636856345635717534723 37'
```



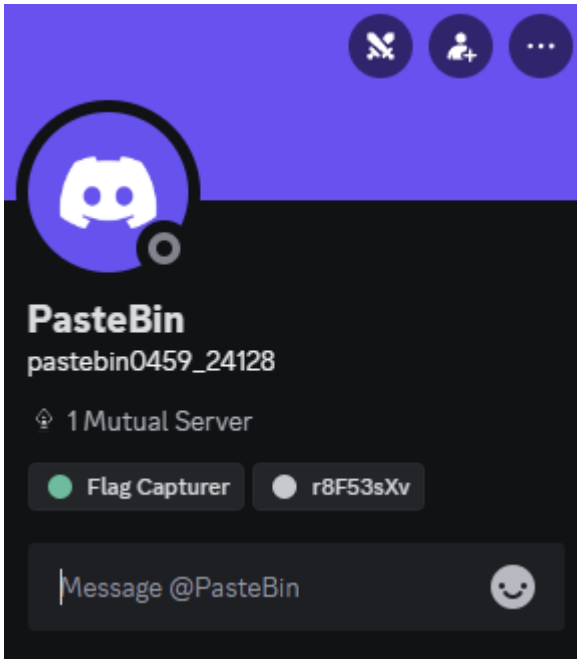
A Little Extra Knowledge Is Too Dangerous

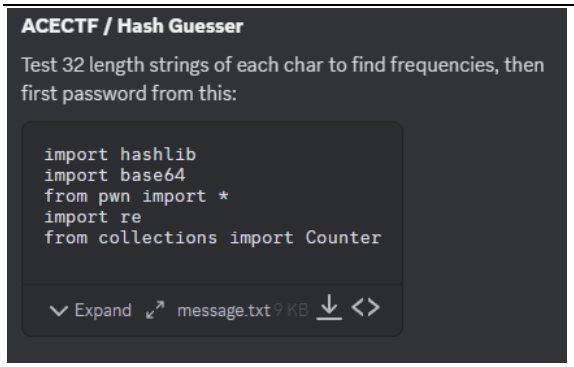
delete some letters and decode from base64

ACECTF{1_6u355_y0u_7runc473d_7h3_3x7r4_kn0wl3d63_r4d0m_57r1n666666
66666555555555_xxxxxxxbbbxxxxxxccccx}

MISC

INSANITY CHECK WE FOUND THIS pastebin0459_24128





ACE6467

EXTRACT USING

```
Success! Password found: 2529797
Extracted data: wrote extracted data to "message.txt".
```

DO ROT AND GET THIS
https://drive.google.com/file/d/1sDC14Hwf6UoCRGTS2dIb3c8CwUSe91GA/view?usp=sharing

EXTRACT USING THE WIDTH 259x194

```
┌──(juke㉿DESKTOP-FSL8MQS)-[/mnt/c/Users/User/Desktop]
└─$ steghide extract -sf RickRolltest.jpg -p 259x194
wrote extracted data to "password.txt".
```

https://github.com/oneshhh/book

GREP THE WORD FILE FOR ACECTF AND FORMAT THE FLAG

Her cousin prefaced his speech with a solemn bow and though she could not hear a word of it, she felt as if hearing it all, and saw in the motion of his lips the ACECTF “The” “B00K” and “Of S3cr3ts.”

WEB

WEB BURIED DEEP

/robots.txt

/buried

/secret_path

/static/css/style.css

WEBRYPTO

TWO VALUES MUST BE DIFFERENT AND MD5 HASHES EQUAL

I TRIED MANY COMBINATIONS BUT THIS OR SOMETHING LIKE THAT SEEMED TO WORK AND GOT THE FLAG /?tom[]=1&jerry[]=2

TOKEN OF TRUST

Try sending a JSON payload like this: {"user":"ace","pass":"ctf"}.

AFTER THE REQUEST MODIFY THE JWT TOKEN AND TRY TO GET THE /FLAG

BUCKET LIST

THE URL GIVES MANY INFORMATION SO

TRYING TO VIEW THE URL WITHOUT THE PATHS I FOUND THESE https://opening-account-acectf.s3.ap-south-1.amazonaws.com/

FOUND THIS WEIRD LINK /cry-for-me/acectf/secret.txt

TRYING TO APPLY IT WE GET A BASE64 STRING AND THEN THE FLAG

FLAG FETCHER

flag can be inspected during during redirection

PWN

ACECTF / jumPIEng

```
└$ nc 34.131.133.224 12346
Main function address: 0x560b53f0c1a9
Enter a redirection address (e.g.-
0x33012a): 0x560b53f0c262
0x560b53f0c262
Redirecting to address 0x560b53f0c262!
Error: Could not locate 'flag.txt'
Flag: ACECTF{57up1d_57up1d_h4rry}
```

Offset of main: 0x11a9.

Offset of redirect_to_success: 0x1262.

Difference: 0x1262 - 0x11a9 = 0xb9

So we need to redicret to 0x560b53f0c1a9 + 0xb9 == 0x560b53f0c262

ACECTF / Running Out of Time

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main() {
    time_t t = time(0);
    printf("Current time: %ld\n", t);

    // Try current time and a few seconds in both directions
    for (int offset = -2; offset <= 2; offset++) {
        time_t test_time = t + offset;
        srand((unsigned int)test_time);
        int rand_val = rand();
        int result = rand_val % 100;
        printf("Time %ld: %d\n", test_time, result);
    }

    return 0;
}
```

```bash
#!/bin/bash

echo "Building prediction program..."
gcc -o predict predict.c
./predict > predictions.txt

echo "Trying all values from prediction..."
cat predictions.txt

echo "If none of the above predictions work, trying brute force..."
for i in {0..99}; do
    echo "Trying $i..."
    echo $i | wine Running_Out_Of_Time.exe > result.txt
    if ! grep -q "Incorrect" result.txt; then
        echo "SUCCESS with $i!"
        cat result.txt
        exit 0
    fi
done
```

!Underflow

Just do strings and get the flag
