CyTaka™
WORLD CYBER
CHAMPIONSHIP

CTF name:	Reverse Engineering
Number of files:	1
Flag pattern:	flag#********
Flag:	flag#9MlQnFHk
Solvable on which OS:	Windows
This solution's OS:	Windows 11
Software:	IDA
Estimated solution time:	20 minutes
Internet access required:	Yes, to download software and to surf to a Base64 decoder.

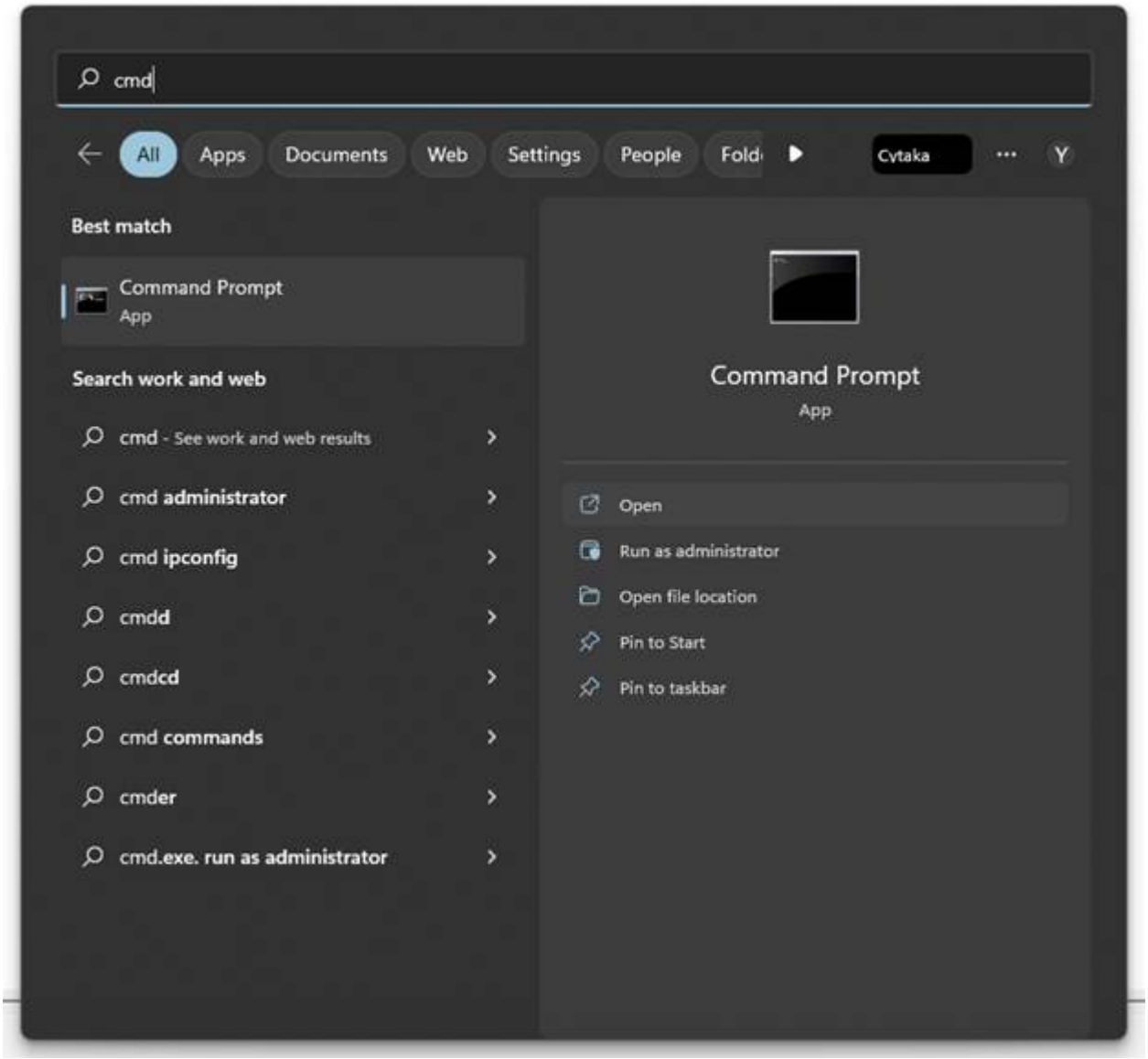
Goal:

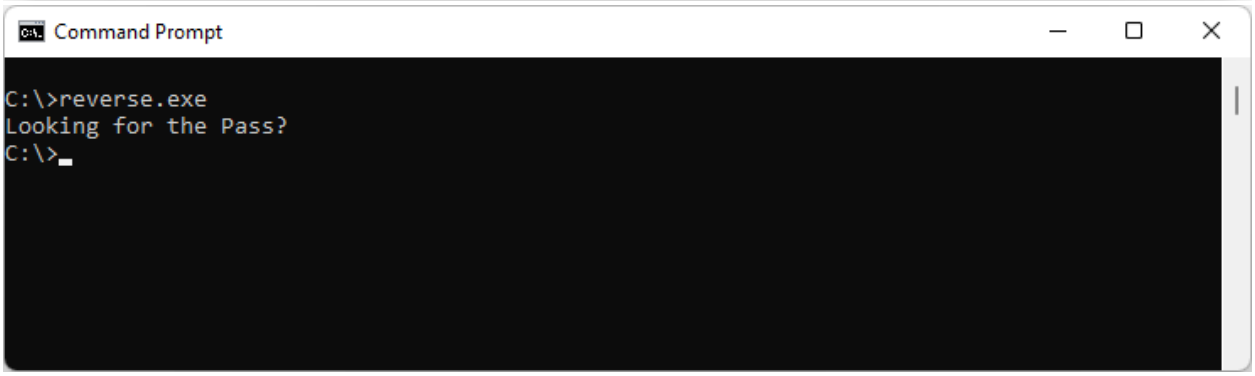
The user has to understand that in order to obtain the flag the .exe file needs to be reverse engineered (e.g., by using IDA or similar tools). After that, the user needs to understand that the flag is encrypted in Base64 and decode it (e.g., by using use online tools, or even manually).

Implementation:

We received an .exe file:

When we run the file in command prompt (cmd) not much happens:


cmd
All
Apps
Documents
Web
Settings
People
Cytaka
Best match
Command Prompt
App
Search work and web
cmd - See work and web results
cmd administrator
cmd ipconfig
cmdd
cmdcd
cmd commands
cmder
cmd.exe. run as administrator
Command Prompt
App
Open
Run as administrator
Open file location
Pin to Start
Pin to taskbar


Command Prompt
C:\>reverse.exe
Looking for the Pass?
C:\>

As is called for in the file name, we may need to reverse engineer the file, using e.g. IDA.

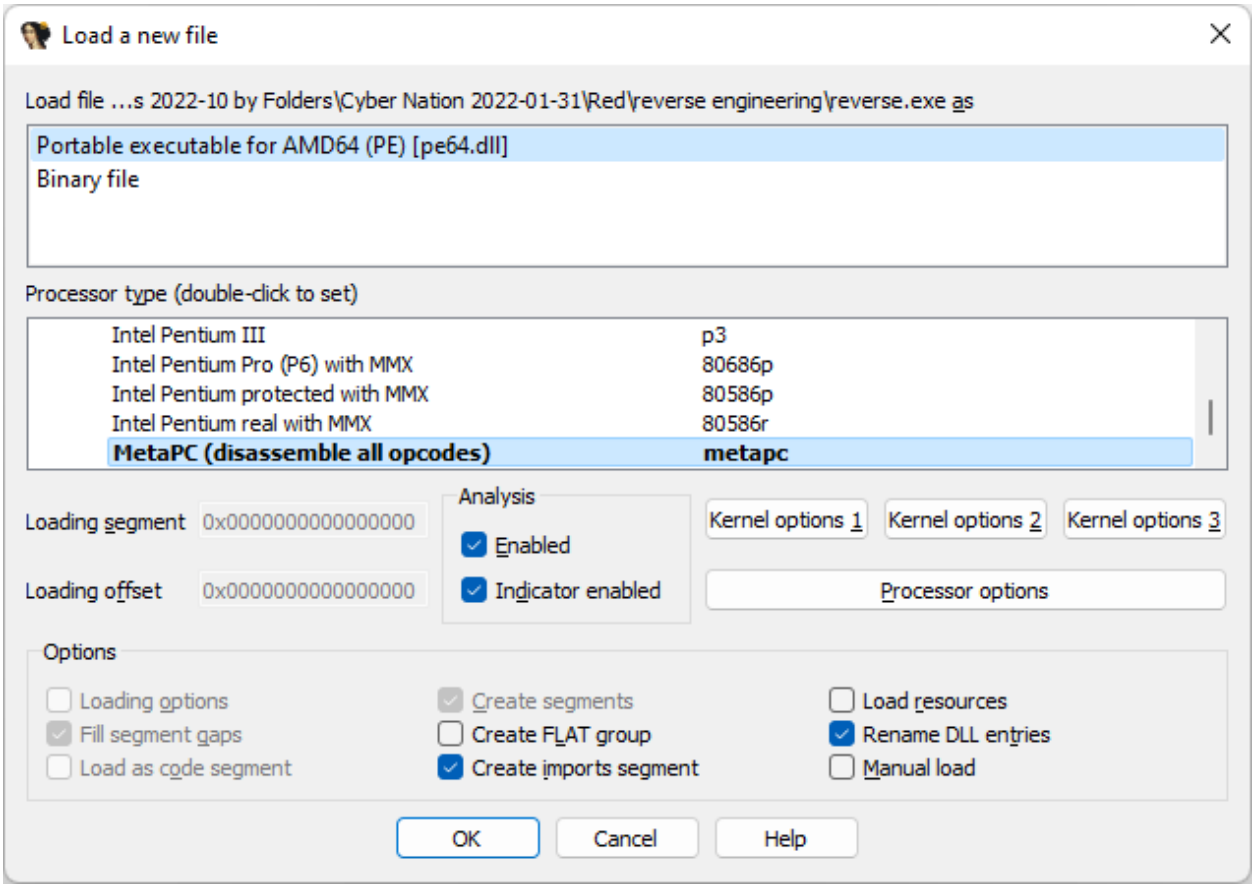
We'll open the file in IDA disassembly, going with the default options:

Load as:

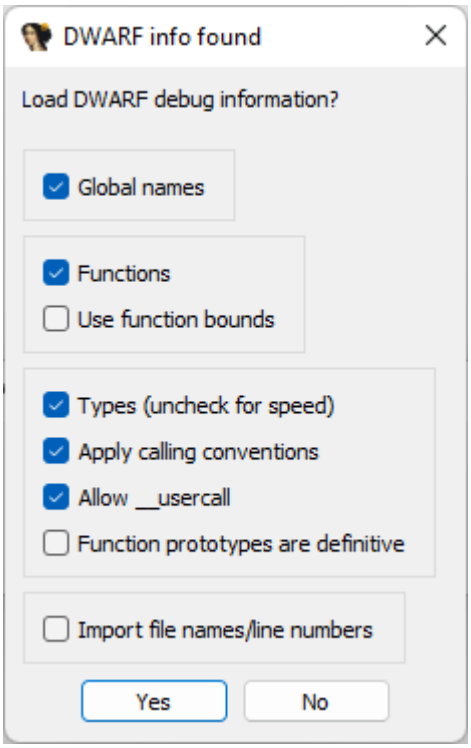
Portable executable for AMD64 (PE) [pe64.dll]

Processor type:

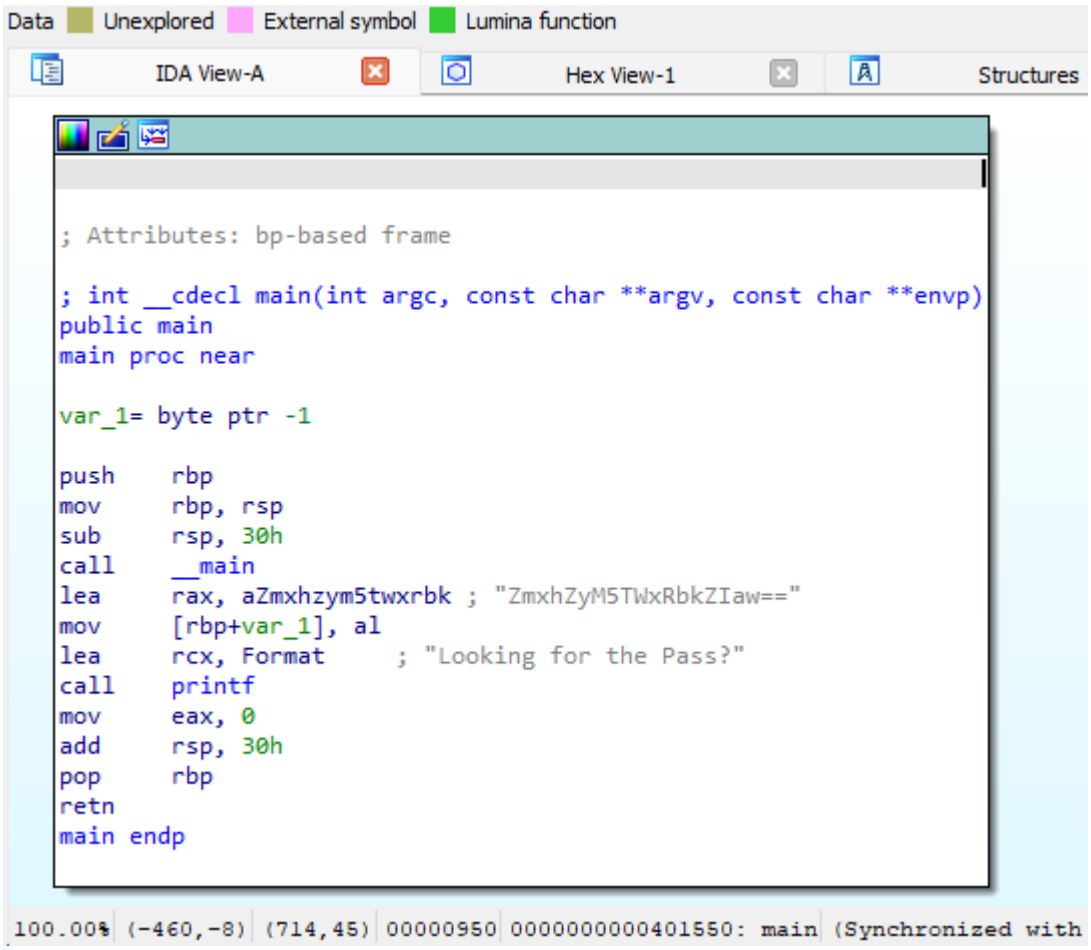
MetaPC



We select Yes in the next dialog box:



And the following screen appears:



This screen contains the file's machine code. We could try to look at its commands to understand what it does, but the biggest clue we get here is the string in the comment:

ZmxhZyM5TWxRbkZIaw==

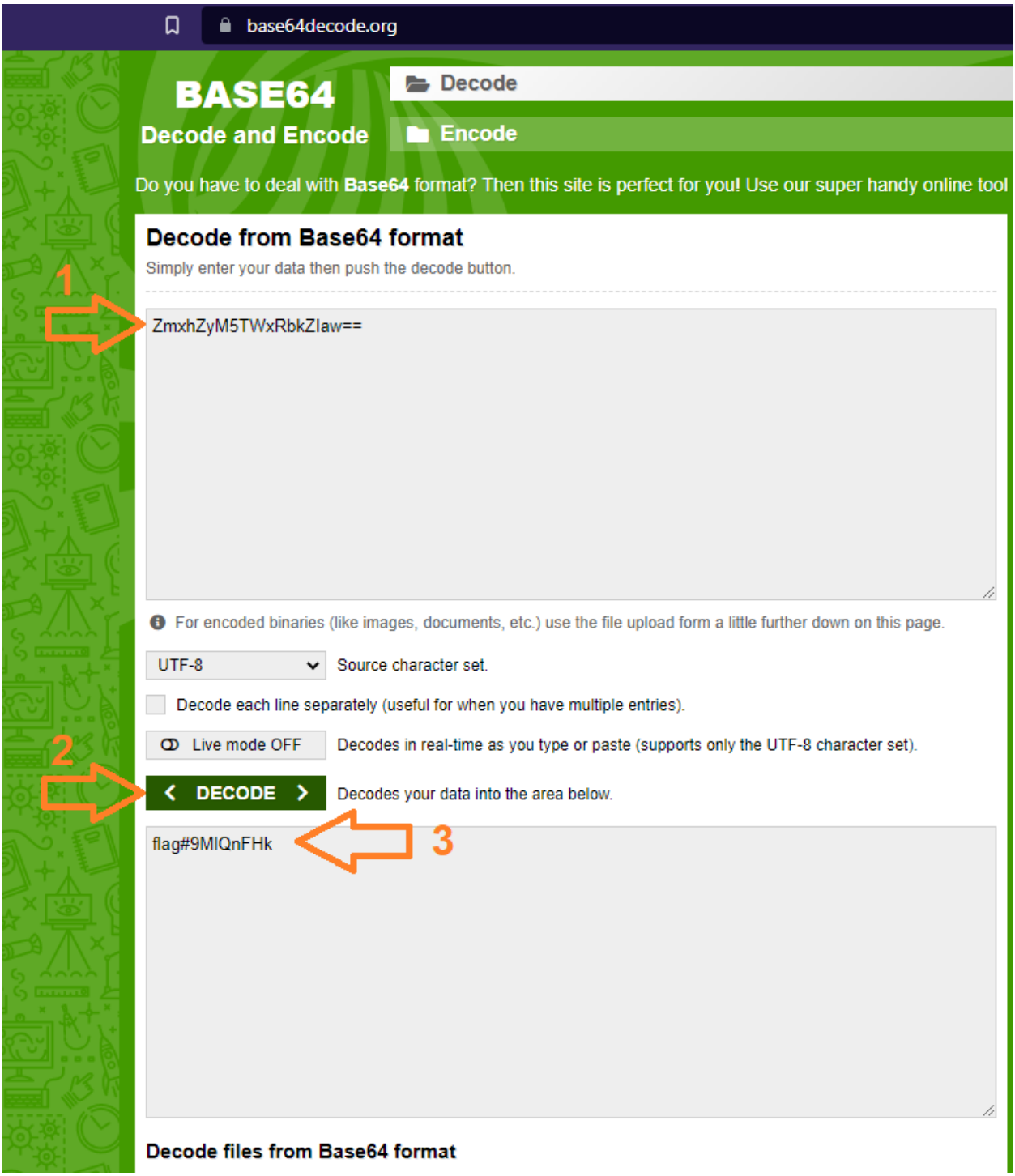
The two = characters in the end remind us of the padding suffix in Base64 encoding.

So, let's search Google for a Base64 decoding website.

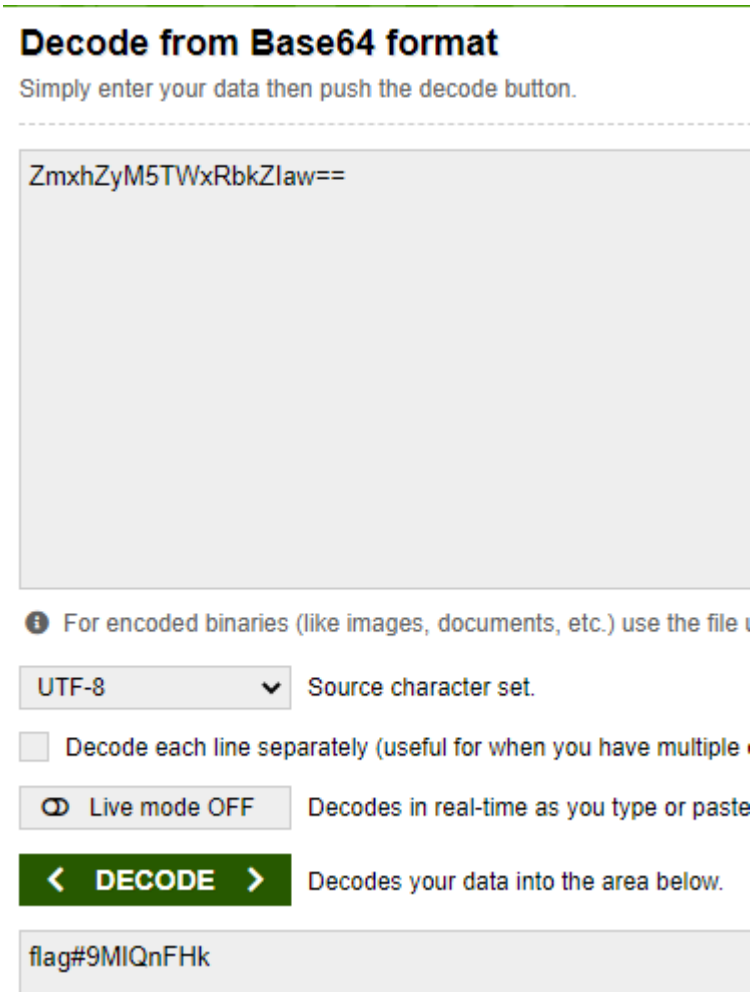
For example, we find:

https://www.base64decode.org/

Let's enter the string ZmxhZyM5TWxRbkZIaw== into the input box and press the green button captioned < Decode >:


base64decode.org
BASE64
Decode and Encode
Decode
Encode
Do you have to deal with Base64 format? Then this site is perfect for you! Use our super handy online tool
Decode from Base64 format
Simply enter your data then push the decode button.
1
ZmxhZyM5TWxRbkZIaw==
For encoded binaries (like images, documents, etc.) use the file upload form a little further down on this page.
UTF-8
Source character set.
Decode each line separately (useful for when you have multiple entries).
Live mode OFF
Decodes in real-time as you type or paste (supports only the UTF-8 character set).
2
DECODE
Decodes your data into the area below.
flag#9MlQnFHk
3
Decode files from Base64 format

Zooming in:



As output, we get a flag:

flag#9MlQnFHk

And that's our answer.

We can also do the decoding process manually if we have a Base64 table and an ASCII table.

A Base64 table can be found e.g., here:

https://en.wikipedia.org/wiki/Base64#Base64_table_from_RFC_4648

And an ASCII table can be found e.g., here:

https://en.wikipedia.org/wiki/ASCII#Printable_characters

According to the Base64 table, the string ZmxhZyM5TWxRbkZIaw== translates to (ignoring the padding characters):

Z	m	x	h	Z	y	M	5	T
25	38	49	33	25	50	12	57	19
011001	100110	110001	100001	011001	110010	001100	111001	010011

W	x	R	b	k	Z	I	a	w
22	49	17	27	36	25	8	26	22
010110	110001	010001	011011	100100	011001	001000	011010	110000

The overall binary string is 18×6=108 bits long:

011001100110110001100001011001110010001100111001010011

010110110001010001011011100100011001001000011010110000

Our flag, being in our standard format, should start with flag#, which in ASCII translates to:

Character	f	l	a	g	#
Hex	66	6C	61	67	23
7-bit ASCII	1100110	1101100	1100001	1100111	0100011
8-bit ASCII	01100110	01101100	01100001	01100111	00100011

Let's compare the 7-bit ASCII string the beginning of our Base64 string:

11001101101100110000111001110100011
011001100110110001100001011001110010001100111001010011

No match.

Let's compare the 8-bit ASCII string the first half of our Base64 string:

01100110011011000110000101100111001000 11
01100110011011000110000101100111001000 1100111001010011

And we have a match.

Moving forward with the decoding, breaking the Base64 bit into 8-bit chunks:

01100110 01101100 01100001 01100111 00100011 00111001

01001101 01101100 01010001 01101110 01000110 01001000

01101011 0000

Since 108 isn't divisible by 8, we'll only use 13×8=104 bits to make a 13-character long string (the last 4 bits are 0s anyway, also used for padding).

Binary	Dec	Hex	Text
0110 0110	102	66	f
0110 1100	108	6C	l
0110 0001	97	61	a
0110 0111	103	67	g
0010 0011	35	23	#
0011 1001	57	39	9
0100 1101	77	4D	M
0110 1100	108	6C	l
0101 0001	81	51	Q
0110 1110	110	6E	n
0100 0110	70	46	F
0100 1000	72	48	H

Binary	Dec	Hex	Text
0110 1011	107	6B	k

Resulting in our flag: flag#9MlQnFHk