Crypto - Beginner

Crypto - Beginner

It won't get much easier than this one....

There is evidence that cryptography has existed since at least 1900 BC. If you want to do a deep dive, there is tons of literature available. Here are a few examples you'll see in some form in all CTFs.

Base64 encoding

Base64 was created as a means to convert binary data to printable characters. At a high level, data is converted from its 8-bit representation to a 6-bit representation. Put another way, it changes them from 256 bits to 64 bits. The sixty-four characters are the upper and lowercase alphabet, the numbers 0-9, and the + and / characters. The equal sign is used as a padding character.

Here's an example from a Unix terminal. If you're unfamiliar with Unix, the echo command repeats what's in the quotes. The output is sent to the base64 program, which converts the input and prints it on the screen. The next line sends the encoded string to the base64 program, the -d tells the program to decode the input.

```
% echo "Base64 Magic" |base64
QmFzZTY0IE1hZ2ljCg==

% echo "QmFzZTY0IE1hZ2ljCg==" |base64 -d
Base64 Magic
```

ROT-13

Rot-13 sounds mysterious. It's also called a Caesar Cipher. It stands for "Rotate 13 characters". More simply put, A will become N and vice versa.

1	2	3	4	5	6	7	8	9	10	11	12	13
A	B	C	D	E	F	G	H	I	J	K	L	M
N	O	P	Q	R	S	T	U	V	W	X	Y	Z

"ROT13 Magic" becomes "EBG13 ZNTVP"

Sometimes the rotation will be 1, 2, 3...25. Fun fact: Rot26 is used to encode every book ever written. 😉

XOR ciphers

XOR is a boolean operator represented as a ^ in Python or a ⊕ . If you just said, "XOR is what what?" keep reading. A boolean can be one of two things. For example, True or False, 1 or 0, etc. In logic, an XOR loosely translates to "one or the other but not both." If I'm evaluating two binary digits (bits) with an XOR if the bits don't match the output is True, if they do match, the output is false.

True (1)	**True (1)**	**False (0)**
True(1)	False(0)	True (1)
False (0)	True (1)	True (1)
False (0)	False (0)	False (0)

So what does it all mean? Using XOR for a cipher takes binary, compares it to other binary data and gives the output. If you know the key it's easy to reverse the process and get the original data back. Let's go through a simple example.

Let's encrypt the name "Bob", and use "key" as the key. To do this we need to convert Bob and key to binary representations. The computer understands the letter 'B' to be equal to 66 in decimal, 'o' is 111, and 'b' to be 98. If you change them to binary numbers, they are:

Bob = 1000010 1101111 1100010

key = 1101011 1100101 1111001

For the example, let's encode "B" with "k" using XOR

B	**1**	**0**	**0**	**0**	**0**	**1**	**0**
k	1	1	0	1	0	1	1
Output	0	1	0	1	0	0	1

The output of the XOR operations equals 0101001 or 41 in decimal.

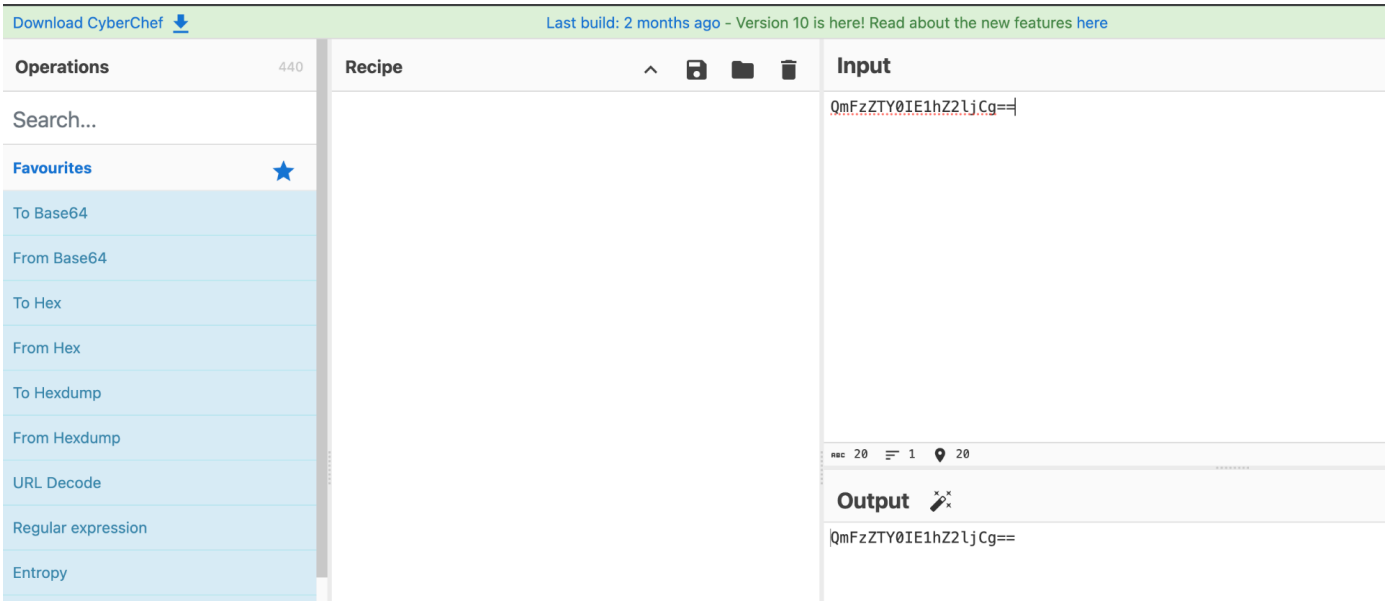
I'll leave the other two as an exercise using the next topic.

CyberChef

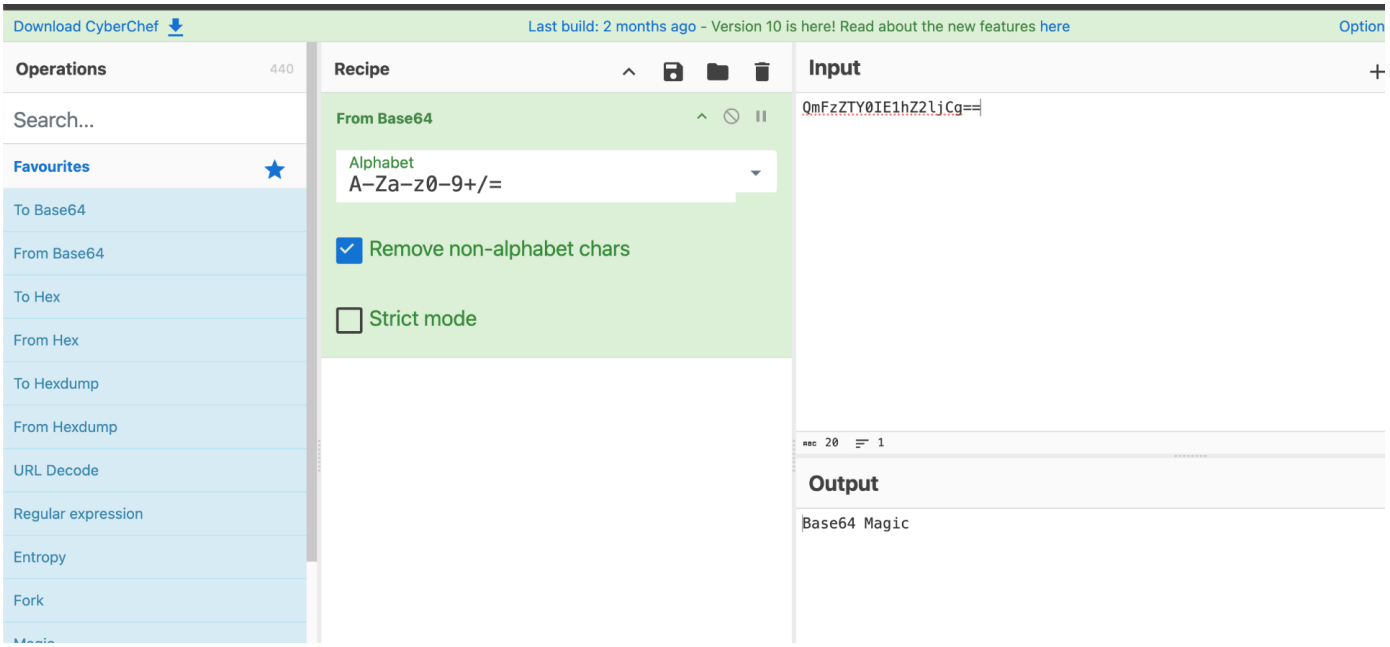
CyberChef is the Swiss Army Knife ™ for decoding, text formatting, etc. I used it to decode Base64, ROT13, and XOR (when I know the key). It's also great for extracting and defanging URLs. There's too much to go over, the best way to figure it all out is just to play with it.

Here are a few examples using Base64 and Rot13. You can access CyberChef at https://gchq.github.io/CyberChef/

When you log in you'll see this screen, just cut and paste what you want decoded in the Input screen:

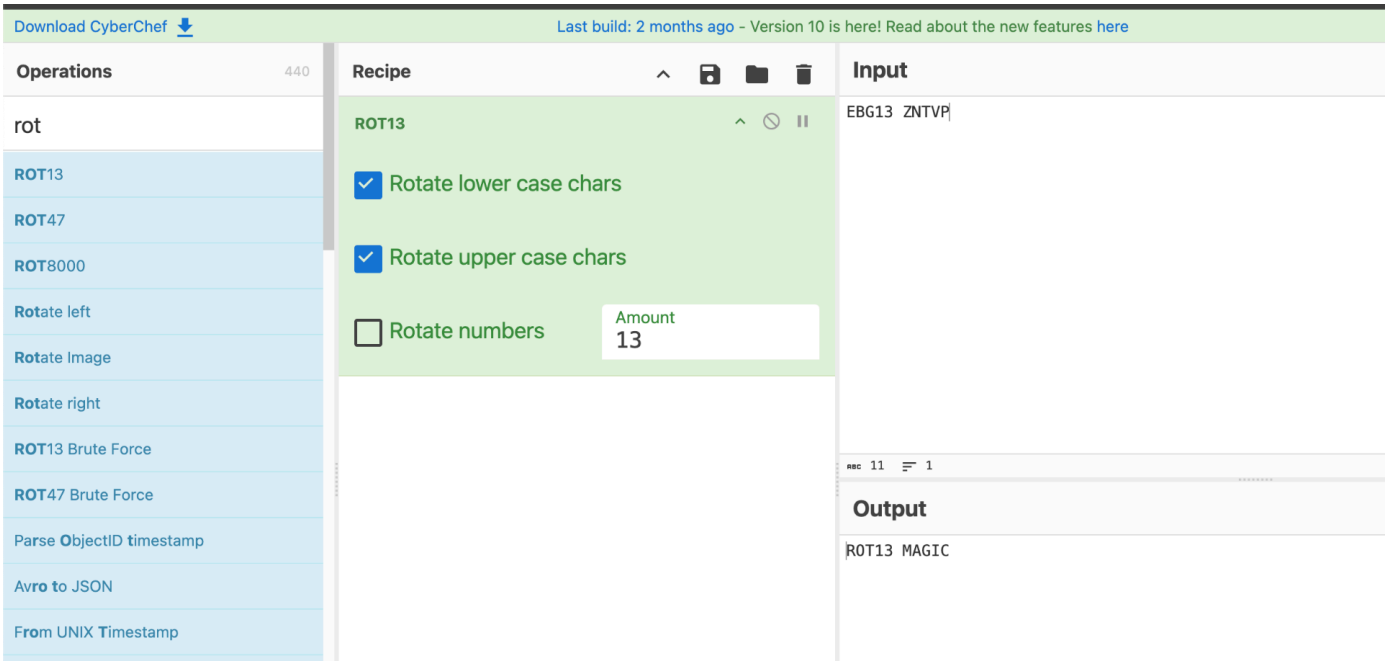


Notice From Base64 on the left side of the screen. Left-click and drag it into the Recipe section, and voila!

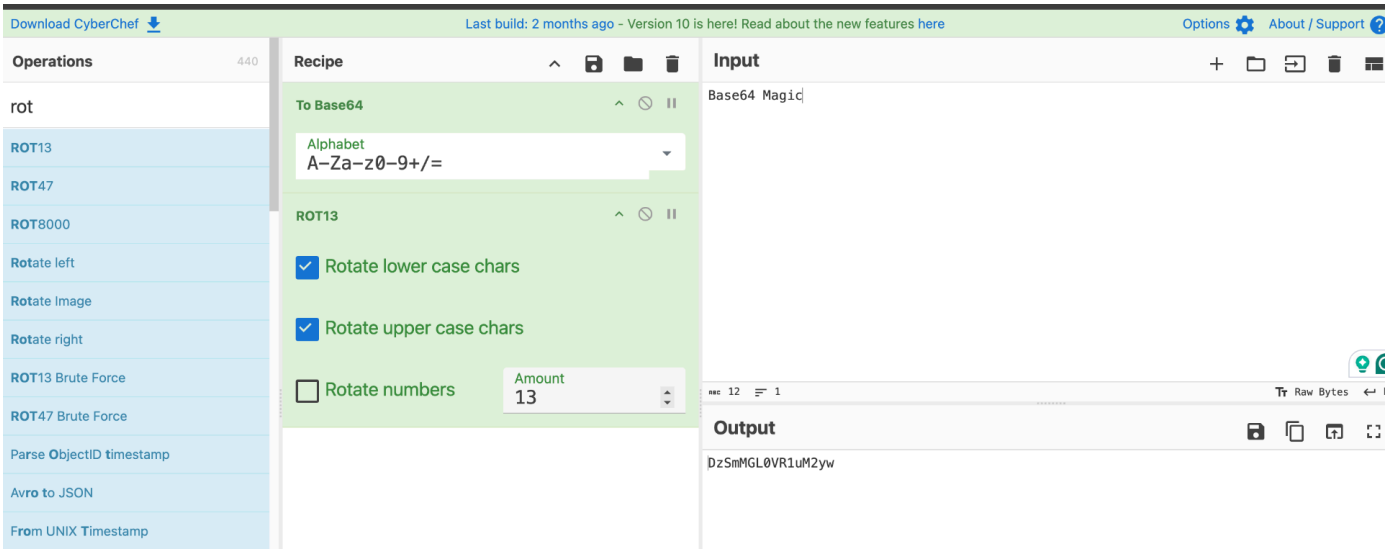


The output is given in the lower right.

For Rot13, click the trash cans to remove the recipes and the Input. Paste in the Rot13 to decode, search for Rot13 and then drag it into the Recipe section.



One final note: You can chain Recipes together just by dragging them into the Recipe section. This creates a Rot13 from the original Base64 encoding.



The Challenge

Use Cyberchef (or the Unix command line) to get the flag:

dGVyYWN0ZntXaDBfZDAzc250X3c0bnRfNF9DaDNmfQ==