This document contains the solutions to all the original public challenges found at <https://ctf.exr0n.com>.

### AB-0

The images suggest `document.cookie`, which is how we access cookies in javascript. Querying `document.cookie` in the console prints out the URL encoded cookie, which decodes to **CT24bb[yumch0c0ch1p]**

### AB-1

Inspecting the HTML source of the page reveals a check that requires a password that matches the regex `/^(s\s\SS[^@|{\B"A-\M"S~s\s`S'$S-Z+.6}\t&\0^\ss\*\\\-(?!0-9)#6%\w/]S){2,}?\S+\ss$/`, and that has a length of `16`. A sequence like ` s sSßSs sSßSss s` happens to match. The server then redirects us to a text file containing the flag: **CT24bb[g0773m81shfm]**

### AB-2

This challenge contains an implementation of a simple seeded PRNG, and the output of the program. The input can be reversed by tracing the value of the generator and subtracting each corresponding value from the corresponding output. For example, the first output is `94`, which we know is equal to `input + (random(1336)%64)`. Thus, the first letter of the input has an ascii value of 67, which corresponds to “C”. Repeating this process and noting the new value of `last` in each cycle reveals the flag: **CT24bb[y33tpyr3v3r51n]**

### AT-0

The qr code has been infested by lenny faces, and some quick research shows that the lenny faces have replaced 7x7 pixel alignment blocks than normally inhabit the corner. A touch up (ignoring the lenny face in the center) allows the QR code to be scanned, and the resulting string contains the flag: **CT24bb[qrc0d350r351113n7113kw7]**

### AT-1

Upon inspecting the HTML source of the page, we see that the password must meet an impossible condition. However, the redirection link upon success is revealed, and navigating to that page manually reveals the flag: **CT24bb[p3n73571n3Z8ruh]**

### AN-0

A commented version of the code found on the page:

|  |
| --- |
| function e(input) {  let a=[];  for (let i=0; i<input.length; i++) { // for each character of the input, front to back  let ord = input.charCodeAt(i); // ascii value of character   for (let j=0; j<8; j++) { // for each bit in the binary representation of the ascii value  if ((ord >> j) & 1) { // if that bit is 1  a.push(1 + j + (input.length - 1 - i) \* 8); // push the encoded position of that on bit in a way that can be decoded...  }  }  }  let b = ''; // our final output starts as an empty string  while (a.length) {  let t = (Math.random() \* a.length)|0; // get a random number 0 <= num <= remaning\_encoded\_one-bit\_indexes  // the above line essentially selects a random index from the list that we generated earlier.     b = b + '0'.repeat(a[t])+'O'; // take the number and add that many `0`s to b, and then tack a `O` on to the end to separate it from the next value.  a = a.slice(0, t).concat(a.slice(t+1)); // remove that index from the array so we don't encode the same bit twice.   }  return b; } |

This essentially takes each `1` bit in the ascii representation of the message, and encodes

the position of that message as `8\*(the position of the character in the message counting backwards) + (the position of the bit)`.

To decode, we seperate the input by `O`s, (remember, that is how they numbers are seperated,)

and count how many `0`s there are. The largest of ((values/8)|0)+1 represents the number of

characters in the original message, and each of these values represent an ON (1) bit at the

position -1\*((value-1-j)/8 +1-input.length). Using that formula on all the segments allows us to

construct the binary representation of the original message, and that reveals the flag: **CT24bb[1h80bfu5k473dj5d0n7u?]**

### CB-0

The page has a simple monoalphabetic cipher, which can be cracked by a website with a frequency analysis tool such as <https://quipquip.com> in “statistics” mode. It can also be solved by hand using frequency analysis and other standard monoalphabetic cipher analysis strategies, which can be found online. The resulting plaintext includes the string `cttwentyfourbb open bracket ikantusltrssohdtonotspell close bracket`, which is pretty clearly the flag: **CT24bb[ikantusltrssohdtonotspell]**

### CB-1

The seemingly random lower and uppercase letters along with the telltale trailing equals signs show that the string is base64 encoded. This can be easily decode through a website such as <https://www.base64decode.org>, and the resulting plaintext, when read backwards, reveals the flag: **CT24bb[0r81u34h34d1gu3551fur3411yw4n7]**

### CB-2

A quick reverse google image search reveals that the image is one of Norman Joseph Woodland, who invented the barcode. Upon closer inspection, we notice that there are a series of black and white pixels along the bottom of the image. When stretched vertically, these pixels do indeed form a valid barcode that can be scanned to yield the flag: **CT24bb[rvrsimgsrch1zgr347]**

### CT-0

The page says “the baby is in the crib”, which is a cryptological term suggesting that the word “baby” is in the plaintext. Using this information, and the failure of frequency analysis, one can infer that this is likely a vigenere cipher and use a corresponding decoder such as <https://www.dcode.fr/vigenere-cipher>. The plaintext says that the flag is the key in all caps, which means we can submit the flag: **CT24bb[GJBOI]**

### CT-1

The reference of “darken, darken” in the hint suggests some hidden pixels that are similar to but not the same color as the background. This color difference is indistinguishable to the human eye but a computer can easily tell the difference. Filling in an area with a paint bucket tool reveals these slight differences, and doing so in the space between the green brackets reveals the flag: **CT24bb[cXmJv31]**

### CN-0

The page explains that the flag is encrypted with a custom implementation of RSA, and provides the public key and modulus. Because the modulus is small, we can find its factorization with a website such as <https://www.wolframalpha.com>. From this we can compute the Carmichael and find the private key as described in the key generation section of the wikipedia page on RSA: <https://en.wikipedia.org/wiki/RSA_(cryptosystem)#Key_generation>. With this key we can decrypt the message and get the flag: **CT24bb[gUdm47h1n801]**

### SB-0

Searching for `etymxbtome` on google leads to a youtube channel with that name. The description of the only video contains the flag: **CT24bb[g00g11zact41fsm467416]**

### SB-1

The prompt contains white text on a white background that says “channel name twenty fifteen pfp green with lower case t”, which suggests YouTube. Indeed, such a channel can be found by filtering by channel, and the time code “19:15” matches the length of one of those videos. A comment on that video ([www.youtube.com/watch?v=8W3ajbSwJ3I](https://www.youtube.com/watch?v=8W3ajbSwJ3I)) contains the flag: **CT24bb[5up3r53cur3p455w0rd15]﻿**

### SB-2

The challenge mentions `AED-G121`, which suggests an AED box by G121—the GCC office. Indeed, on the side of the box is a QR code, which when scanned yields the flag: **CT24bb[b17pr073c710n1z1mp0r74n7]**

### ST-0

The prompt says to “read the black”, which can be interpreted as counting the black pixels on the image. Left to right, the pixel count is [67, 84, 50, 52, 98, 98, 91, 121, 117, 109, 89, 85, 77, 93]. When these numbers are treated as ASCII values, the letters form the flag: **CT24bb[yumYUM]**

### ST-1

This challenge alludes to a yearbook from “two decades ago”, and inspecting the page source reveals that the challenge was written in 2019. The library holds four copies of the 1999 yearbook, and the pocket on the front cover of the one labeled C.3 holds a QR code. When scanned, that code yields the flag: **CT24bb[w0wd4z01dbu7k0w1r173]**

### SN-0

The dots form valid braille, which decodes to K-100-handicpstikr. This suggests K-100 handicap sticker, or the handicap accessible stickers near the front doors of the cafe. A quick search around both reveals a small QR code on the underside of the metal beam above the left one (facing the cafe from the library). The QR code, when scanned, reveals the flag: **CT24bb[w3nu7ry8r41130n45cr33n1m40]**

After solving these challenges, a link to the bonus section is revealed. They can be accessed at <https://ctf.exr0n.com/public_static/CT24bonus/congrats.txt>. The write ups for those challenges can be found at /public\_static/writeups/bonus.txt