Bonus1

1. JSFuck
   1. It starts with `+[];` and then the actual thing, so people hopefully will notice the semicolon and look that up, which will make them realize that its all javascript
   2. The flag is CT24bonus[js1z7h3b35714ngu4g3101f173m3]
2. Words from teachers
   1. People should go to specific teachers and do a chicken dance for 15 seconds, then say “abra cordoba” to get words
   2. Words: Novice, Poncho, Rainbow, Style
      1. Lelia: poncho
      2. Abby: rainbow
      3. Tom: style
      4. Cynthia: novice
   3. Flag: CT24bonus[noviceponchorainbowstyle]
3. Session storage
   1. There will be a bitly link that transfers to an html page that puts something in session storage and then transfers to a text document. People will have to go into their history and figure out what happened, and then go into session storage and get their flag.
   2. Bitly link: <http://bit.ly/CT24bonus38bffce013e484413cabc103cc5e072df>
   3. We use that so that people are already expecting a redirect and the second one will be more likely to fool them
   4. Server side, I only send the flag if the referer is the transfer page which means that only the transfer page can get the flag. If they manually change the referer header then props they get to bypass that security.
   5. Flag: CT24bonus[s0m371m35d0n7c134rurh1570ry3h]
4. Really big file
   1. There will be a 32+ gb file which contains something like “../CT24bogus/chal\_4\_end\_<Hex>../CT24bogus/chal\_4\_end\_<hex>../CT24bonus/chal\_4\_end\_<realhex>../CT24bogus/chal\_4\_end\_<hex>” or something
   2. They will have to read the file from a stream to avoid crashing anything
   3. Something like python

|  |
| --- |
| length = len('public\_static/CT24bogus/stage4\_end\_f2f80fbd67842ad9')  with open('bigfile.txt', 'r') as f:  while True:  c = f.read(length)  if "CT24bonus" in c:  print(c)  break |

* 1. Flag: CT24bb[r3411yr3411y81gf11354r3cr444zy]

1. Jimmy Sombrero
   1. Jimmy sombrero remains in the music room corner tack board thingy
   2. QR: 
   3. Flag: CT24bb[g00d01dj1mm3y1z571117h3r31g]
2. Hash cracking
   1. MD5: puppycat4 = **0c8ba823cc1ae993d16778ae46559266**
   2. Flag: CT24bonus[s417urh45h35555r3333111]
3. Headers
   1. Link to fifa 2018 headers compilation
   2. Request headers, contains a header that says something like “endpath”, “<end path here>”
   3. Flag: CT24bonus[r3q357h34d3r54r3r3411y1mp0r74n720745]
4. XSS
   1. Input that leads directly to an eval so you can put “x.responseText” in the input to print out the response with flag
   2. Flag: CT24bonus[s4n171z3ur1npu75r333398734]
5. Scav
   1. Bench by tree between library and that curvey path on the underside of a concrete slab
   2. Qrcode: 
   3. Flag: CT24bonus[y0ukn0w7h35ch001700w311435978]
6. Coordinates and plus codes (google maps)
   1. Desmos graph with cordinate pairs:
      1. <https://www.desmos.com/calculator/0won6uqcqs>
   2. So each place has a plus code apparently
   3. It goes something like this: XXXX+XX City, country
   4. A regex that matches what we care about: /[\w\d]{4,}[+].{2}/
   5. So what we can say is that its ([\w\d]{4,}[+].{2})[-2:] # that last part is python for last two characters
   6. And we just get a bunch of them and boom
   7. So the solving goes like this:
      1. Figure out what that thing means
      2. For each coordinate pair
         1. Get the plus code
         2. Find the thing that it matches
         3. Add
      3. Probably 8-10 coordinate pairs for a harder to brute force code
   8. Pairs:
      1. 47.002060, 24.319743: RV
      2. 33.271457, 158.659171: HM
      3. 41.769503, -99.381546: R9
      4. 1.991574, 20.383128: J7
      5. -29.013849, 134.753254: F8
      6. 66.359692, -136.735640: VP
      7. 22.987539, -81.823382: 2J
      8. -8.835461, -48.509003: R9
      9. -25.966372, -56.189602: F5
      10. 21.709634, 78.772157: VV
   9. Path: bonus10/end\_RVHMR9J7F8VP2JR9F5VV.txt
   10. Flag: CT24bonus[y0uw111n3v3r35c4p37h3r3g3x43972]
7. Bitcoin blockchain exploration
   1. Bitcoin block number: 566254
   2. It was mined by 1Hz96kJKF2HLPGY15JWLB5m9qGNxvt8tHJ according to <https://blockexplorer.com/block/000000000000000000207d3d8929e91c3f8377ad1bfd9cd4dd6369d4cd87c12d>
   3. Prompt: Who mined 566254?
   4. Path: bonus11/end\_1Hz96kJKF2HLPGY15JWLB5m9qGNxvt8tHJ.txt
   5. Flag: CT24bonus[7h3m057v01171137h1ng73982]
8. Referer header bypass
   1. The prompt says “sent me to” which is analogous to “referred”. Making a request to the end path leads nowhere until a referrer header is tacked on. This gives the flag: CT24bonus[f1r571337h4x1ngf0r7h3r341w0r1d38274]