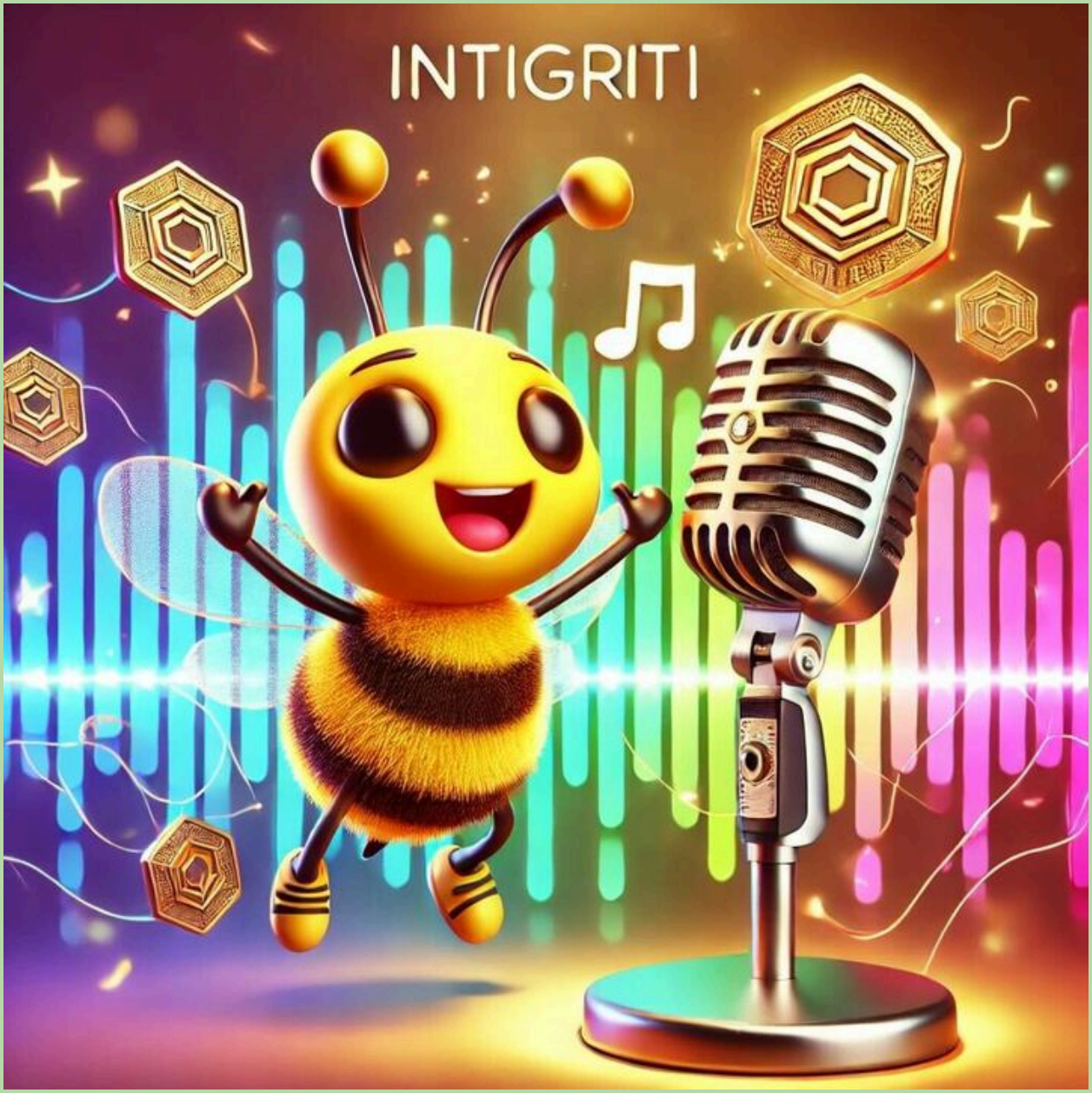
Dropping Bars at DEFCON 32

Challenge Name: DEFCON 32 - Coin Challenge Contest



Description

https://www.intigriti.com/landing/coin_challenge

Approach

Intro

There are two ways to start this challenge:

1. Receiving a physical challenge coin (very cool, you spotted some fun vulnerabilities on a challenge card laying around the conference floor)
2. Scanning the QR code on a sheet at Intigriti's desk (also cool, you still get to talk to Intigriti 😎)

Either way, you receive a similar amount of information. Most of the text on the coin isn't too noteworthy, but you'll notice that the edges contain an interesting pattern of rectangles. From here, there are a few observations you can make:

- If you are familiar with and have used Spotify, these kind of look like Spotify QR codes
- If you look at the web page, you'll notice more that some letters on the coin are bolded, and in fact directly spell out: **SPOTIFY**

Verse 1 - Identifying Spotify Codes

From here, it doesn't take too long to realize that the association with Spotify is that this is one of their media codes. If you try scanning the code, you'll run into trouble. First of all, the orientation isn't exactly right and there's not a great guarantee that the code is even valid. Either way, we can try doing the following:

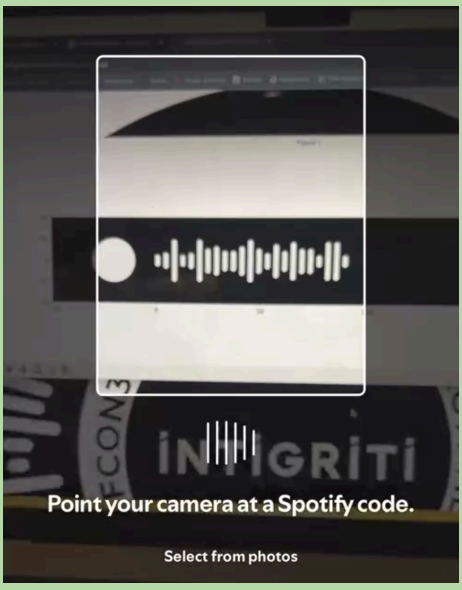
1. Re-orient the Spotify code and try to use their software
2. Figure out how their codes even work

The first step would be figuring out *how* to rearrange the bars;

- Using image editing software, you can clip some of the pieces on the web page and rearrange them into a Spotify code (we found this to be a bit tedious)
- You can measure the heights of the bars and come up with the octal representation of the code yourself.
    - You can do this by *relatively* ranking the order of the bars (0 being the shortest, 7 being the tallest)
    - You can also look at the pixel heights to derive this information (the heights differ by 10 pixels or so for each, so you can easily tell what the height of a rectangle is supposed to be)

There is some wiggle room in the heights as error correction is done on Spotify codes (discussed later). If you measure the heights (left image), you can turn them into a Spotify code using a library like matplotlib (right image):

```
heights = [0, 1, 7,
           0, 1, 7,
           3, 4, 2,
           2, 4, 7,
           3, 1, 5,
           2, 6, 3,
           3, 0, 6,
           6, 0]
```



Unfortunately, this doesn't work. The next thing that we can try to do is see how Spotify codes work and try to directly decode the rectangles.

Lead-in - How Spotify Codes Work

Peter Boone does a fantastic job walking through his reverse engineering of how Spotify codes work. I won't include all of the details here, but in essence:

1. Spotify represents its codes in octal via 23 rectangles with 8 varying heights.
2. From here, one can turn this octal into bytes, perform a cyclic redundancy check (CRC), apply forward error correction (FRC), and then do a convolutional decoding via the Viterbi algorithm
3. This provides a media ref (a decimal number) which can then be turned into a Spotify resource URI using an internal API that Spotify publicly exposes

Alternatively, if you are not a math major like me, you can just clone Peter Boone's GitHub and just run the decoder yourself :)

Using the heights depicted above, you can modify the decoder in the above cloned repository to look something like:

```
if __name__ == "__main__":
    token = "<token here>"
    heights = [0, 1, 7,
               0, 1, 7,
               3, 4, 2,
               2, 4, 7,
               3, 1, 5,
               2, 6, 3,
               3, 0, 6,
               6, 0]
    print(f"Heights: {heights}")
    # drop the fist, last, and 12th bar
    heights = heights[1:11] + heights[12:-1]
    decoded = spotify_bar_decode(heights)
    print(f"Media ref: {decoded}")
    uri = get_uri(decoded, token)
    print(f"URI: {uri['target']}")
    summary, full_response = get_info(uri["target"], token)
    print("Summary:")
    pprint.pprint(summary)
```

Running the above code will give us the following media ref: 575541171

The blog post outlines how to get a Spotify token (requires an account) to use in the media ref→URI lookup, however if you do this you'll see that you get a 404 (i.e., the media ref does not exist).

Outro - Cracking the Media Ref

Since the media ref isn't valid, we need to figure out something else to do with it. This amount of decimal is a bit too small to have information itself (wsq= in ASCII isn't a real flag). There are many different things we can try to do with a 9 digit code, but one of the more likely options is that it maps to an IP address.

Converting the media ref into an IP address gives: 57.55.41.171

Unfortunately, this goes nowhere. This would seem like a dead-end, but if we try putting the number into Google we get some ideas that this number is not a broken up IP address, but instead the *decimal representation* of an IP address. When we convert it properly into an Ipv4 address, we get: 34.78.15.179

Entering this address into a browser redirects us to:
https://app.intigriti.com/programs/intigriti/defcon32-coinchallengecontest/detail

At this point, we are done and can complete a submission to finish the challenge 🙂