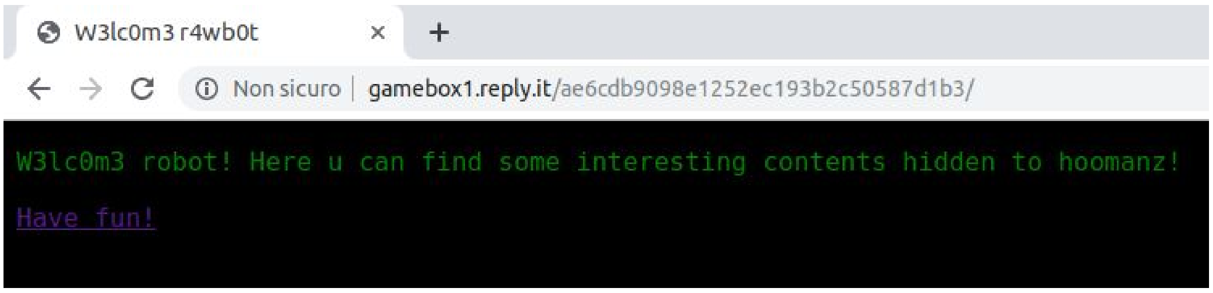
STEGANO>9000 - MISC 300

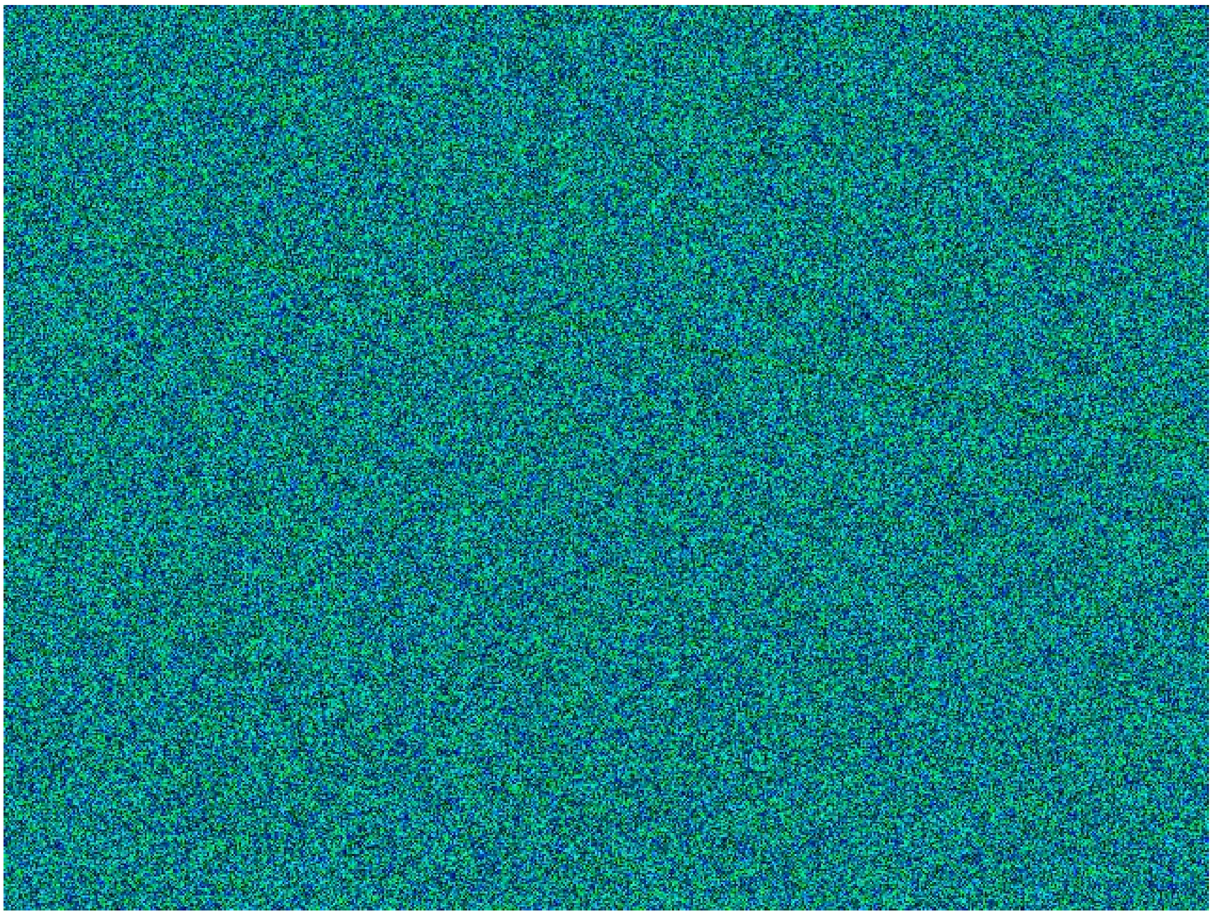
"Oh boy! It's clear Armstrong's been kidnapped. According to the clues, he was snatched by some kind of robotic human-hating aliens with a passion for weird internet memes. R-boy discovers a website that probably contains more useful information for rescuing Armstrong. However, it seems to be protected by some anti-human access mechanism. Can you bypass it?"

This challenge appears like a reverse-captcha, we must prove to be a robot.

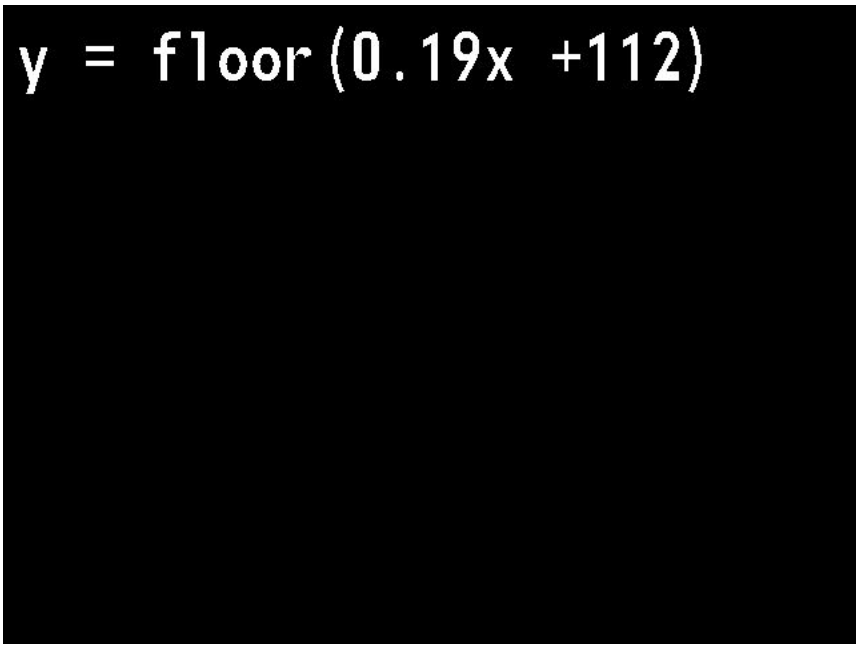


```
W3lc0m3 robot! Here u can find some interesting contents hidden to hoomanz!

Have fun!
```

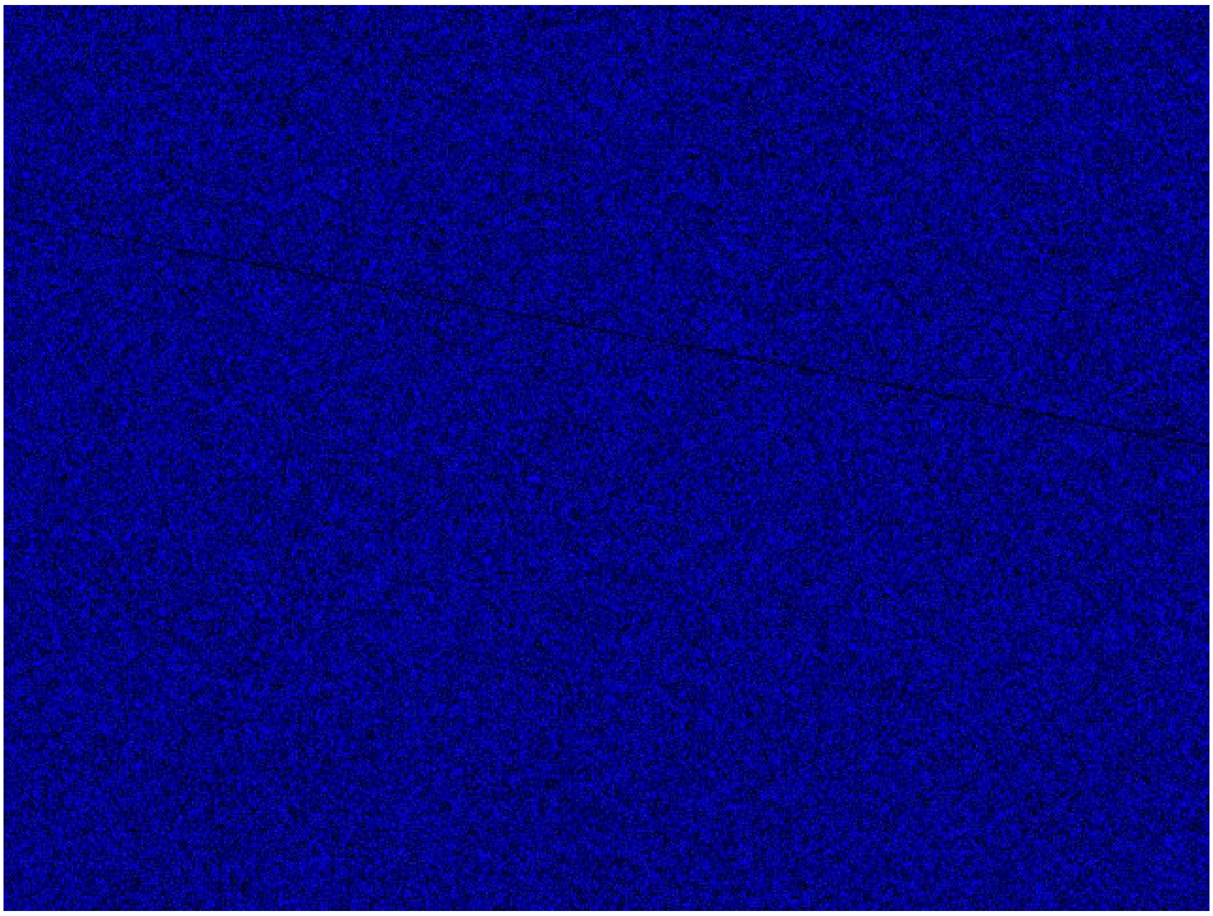
To do so, we have to automate a 5-step steganography problem, here is an example image:



Using StegSolve we found out that we had a straight line formula in the green pane:

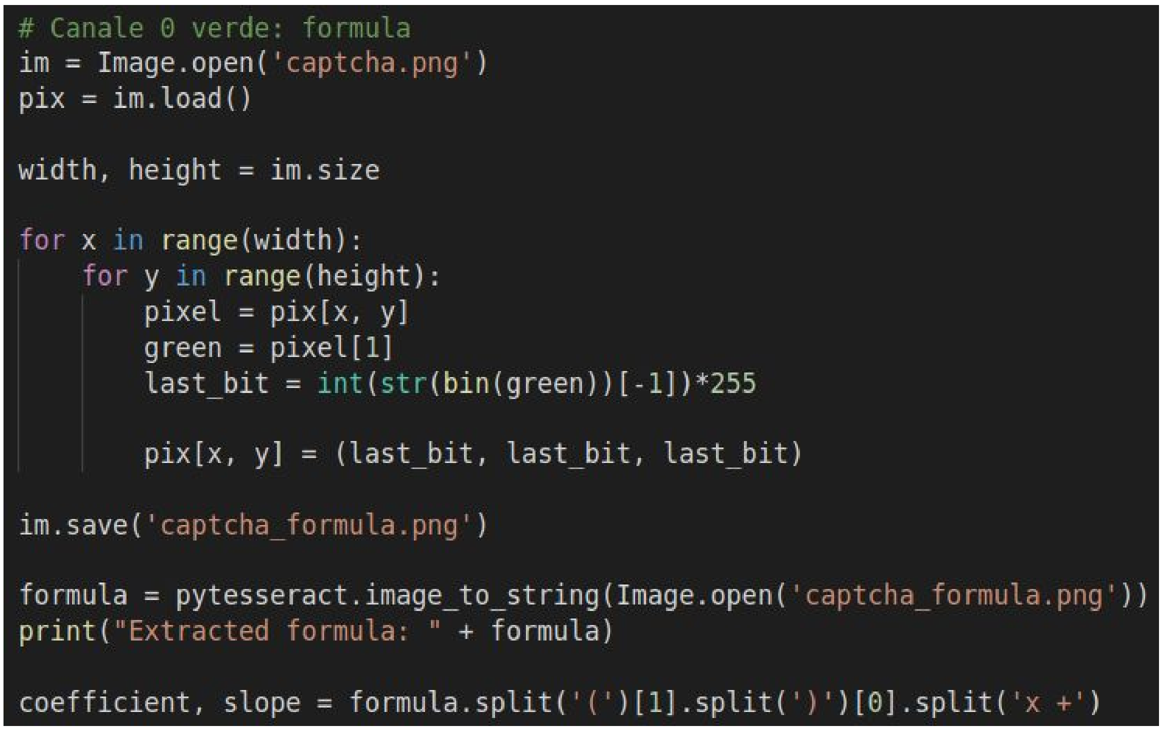

y = floor(0.19x +112)

And also the line itself, with some interesting strange pixels at the start in the blue pane:



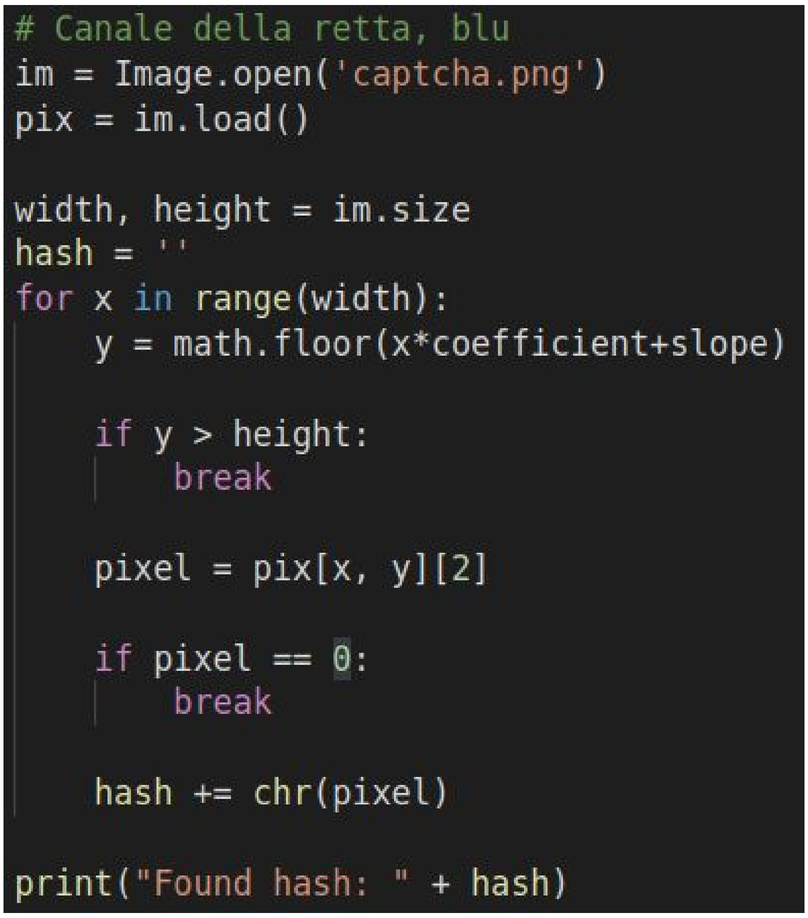
To find out what these pixels what about we used python, which also allow us to automate all the process.

Using PIL and Pytesseract we extracted the formula as coefficient and slope:



```python
# Canale 0 verde: formula
im = Image.open('captcha.png')
pix = im.load()

width, height = im.size

for x in range(width):
    for y in range(height):
        pixel = pix[x, y]
        green = pixel[1]
        last_bit = int(str(bin(green))[-1])*255

        pix[x, y] = (last_bit, last_bit, last_bit)

im.save('captcha_formula.png')

formula = pytesseract.image_to_string(Image.open('captcha_formula.png'))
print("Extracted formula: " + formula)

coefficient, slope = formula.split('(')[1].split(')')[0].split('x +')
```

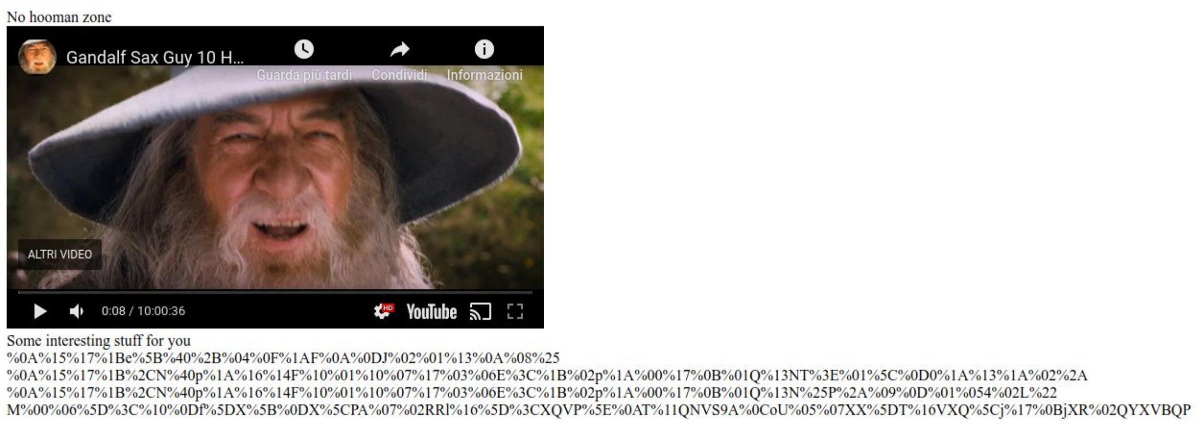
Then with the following script we extract the non-zero pixels at the start:



```python
# Canale della retta, blu
im = Image.open('captcha.png')
pix = im.load()

width, height = im.size
hash = ''
for x in range(width):
    y = math.floor(x*coefficient+slope)

    if y > height:
        break

    pixel = pix[x, y][2]

    if pixel == 0:
        break

    hash += chr(pixel)

print("Found hash: " + hash)
```

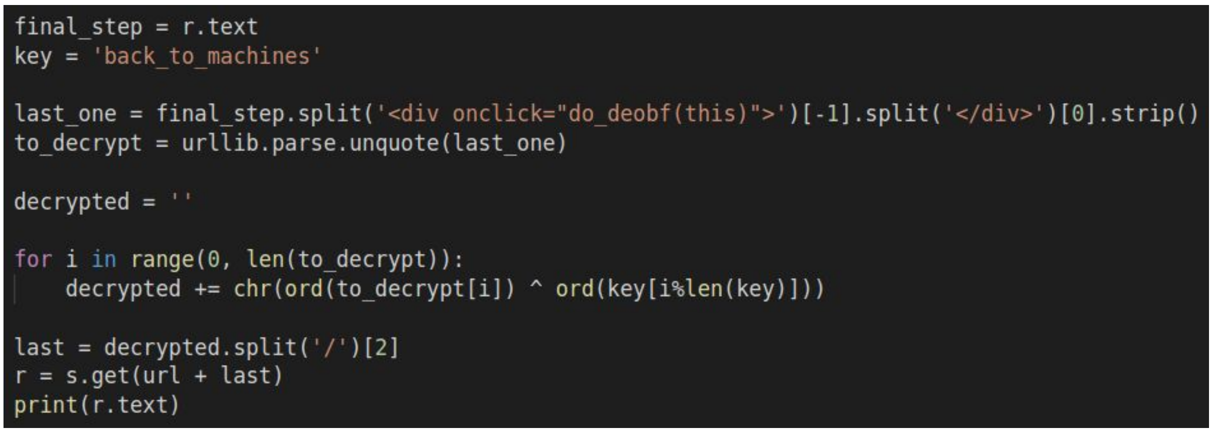
The string found was an md5 hash of a random number, but after some tries we found out that was to be sent as is, and so we did.

After the 5 steps however we was presented by the following page:


No hooman zone

Some interesting stuff for you
%0A%15%17%1Be%5B%40%2B%04%0F%1AF%0A%0DJ%02%01%13%0A%08%25
%0A%15%17%1B%2CN%40p%1A%16%14F%10%01%10%07%17%03%06E%3C%1B%02p%1A%00%17%0B%01Q%13NT%3E%01%5C%0D0%1A%13%1A%02%2A
%0A%15%17%1B%2CN%40p%1A%16%14F%10%01%10%07%17%03%06E%3C%1B%02p%1A%00%17%0B%01Q%13N%25P%2A%09%0D%01%054%02L%22
M%00%06%5D%3C%10%0Df%5DX%5B%0DX%5CPA%07%02RRl%16%5D%3CXQVP%5E%0AT%11QNVS9A%0CoU%05%07XX%5DT%16VXQ%5Cj%17%0BjXR%02QYXVBQP

Watching carefully all those clickbaits, we found out that the flag was in the last encrypted URL, so we scripted it:


```python
final_step = r.text
key = 'back_to_machines'

last_one = final_step.split('<div onclick="do_deobf(this)">')[-1].split('</div>')[0].strip()
to_decrypt = urllib.parse.unquote(last_one)

decrypted = ''

for i in range(0, len(to_decrypt)):
    decrypted += chr(ord(to_decrypt[i]) ^ ord(key[i%len(key)]))

last = decrypted.split('/')[2]
r = s.get(url + last)
print(r.text)
```

And got it:

{FLG:Oh_M4m4m14_M4m4m14_L3tM3Go}