The fact that the first 3 bytes of xored are 0 means that s1 and s2 start with the same 3 characters. We know the flag format is CTF{sha256}, so lets try to find s1 assuming s2 starts with either CTF of ctf.

```
xored = ['\x00', '\x00', '\x00', '\x18', 'C', '_', '\x05', 'E', 'V', 'T', 'F',
'U', 'R', 'B', '_', 'U', 'G', '_', 'V', '\x17', 'V', 'S', '@', '\x03', '[', 'C',
'\x02', '\x07', 'C', 'Q', 'S', 'M', '\x02', 'P', 'M', '_', 'S', '\x12', 'V',
'\x07', 'B', 'V', 'Q', '\x15', 'S', 'T', '\x11', '_', '\x05', 'A', 'P', '\x02',
'\x17', 'R', 'Q', 'L', '\x04', 'P', 'E', 'W', 'P', 'L', '\x04', '\x07', '\x15',
'T', 'V', 'L', '\x1b']
s2 = "ctf"
s2 = (s2 * ((len(xored) // len(s2)) + 1))[:len(xored)]
s1 = [chr(ord(a) ^ ord(b)) for a, b in zip(xored, s2)]
s1_str = "".join(s1)
print(s1_str)
```

After running that script, we see that s2 is actually ctf looped over and over, and that we got the flag.

Made with love by: AndreiCat