Copyright © 2017
Powered by Hugo and Hyde-X

Pwn2Win CTF 2016 Simple Cryptography (Crypto 60) Writeup

Mar 28, 2016 · 2 minute read · by rspkt · writeups

Problem

> Description: The club decided to evolve the security of communication for something more efficient and simple to be implemented … simply too much!

The technique was reproduced in the message encoded below:

GA3TCYZRGU2DKXZXG5PTENZTHAZDSXZQHEYTKMJUGBSV6MBXGFSF6MTDGIYTGYJSMMZTIM3FL4YDOMLCL42GENJRGU2TIOBVGQ2WIXZRGBPTAZQ=

Add CTF-BR{} to submit the resulting hash.

Solution

Figuring out the encoding scheme didn't take many minutes. Noticing that the encoding alphabet seems to be a subset of Base64, we gave Base32 a shot. After decoding it, we were given the string:

071c1545_77_273829_0915140e_071d_2c213a2c343e_071b_4b515548545d_10_0f

This is of course not just hex-encoded ASCII chars, but it's been processed in some way. When seeing these kinds of things we typically throw XOR on the hex, and hopefully we see something that looks like English.

We XORed the string with all possible byte values. No single value gave a full match, but we saw words like "always" and "simple". We could conclude that different XOR "keys" was used for different words. When puzzling together the actual printable words, we got:

sha1_(_why_this_is_always_so*simple?*_)

So we SHA1-hashed the text between the parenthesis and inserted it into the flag format.

Code

```
text = "071c1545_77_273829_0915140e_071d_2c213a2c343e_071b_4b515548545d_10_0f"
text = text.replace("_", "")

# XOR with all possible byte values
for i in range(255):
    r = ""
    for j in range(0, len(text), 2):
        d = int(text[j:j+2], 16)
        r += chr((d ^ i) % 256)

    # Re-insert underscores and print
    print i, ":", '_'.join([r[0:4], r[4:5], r[5:8], r[8:12], r[12:14],
                            r[14:20], r[20:22], r[22:28], r[28], r[29]])
```