Features Business Explore Marketplace Pricing
This repository Search
Sign in or Sign up

pogTeam / writeups
Watch 3 Star 8 Fork 1
Code Issues 0 Pull requests 0 Projects 0 Wiki Insights

Branch: master writeups / 2016 / seccon / Vigenere / README.md
Find file Copy path

marcosValle Minor fixes
1c3729e on Dec 13 2016
1 contributor

62 lines (44 sloc) 3.08 KB
Raw Blame History

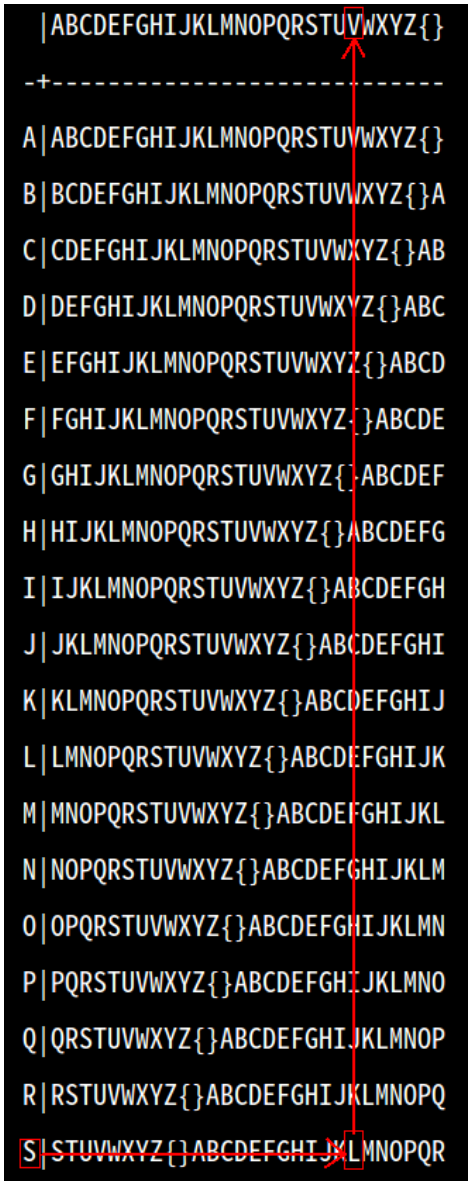
#Vigenere - Crypto 100

As the title for this chall claims, this is all about Vigenere cipher. Interestingly the alphabet used is not [A-Z], but also includes '{' and '}'. Besides giving us a full Vigenere table, the chall also provides some information about the key, the plaintext and the ciphertext.

```
k: ???????????
p: SECCON{??????????????????????????????????}
c: LMIG}RPEDOEEWKJIQIWKJWMNDTSR}TFVUFWYOCBAJBQ

k=key, p=plain, c=cipher, md5(p)=f528a6ab914c1ecf856a1d93103948fe
```

From now on we are going to assume *len(k)==12*. Our main goal is clearly to find *p*. Since we have the first 7 chars of *p* we could easily find the first 7 chars of *c*. We could even use the given table and do it manually. For the first char of *p*, *p[0]=='S'*, we would check the row of the table corresponding to *S*. Since *S* gets mapped to *c[0]=='L'*, we look for *L* in this row, which is in the column of *V*. The figure below ilustrates this process:




```
pt = "SECCON{"
ct = "LMIG}RP"

res = ""
for p,c in zip(pt, ct):
    res += chr( ord('A') + ( (ord(c) - ord(p)) % 28) )

print(res)
```

This simple procedure results in *VIGESEN* as the first part of the key. Evidently the chars '{' and '}' are not being correctly treated in positions 5 and 7. Either by correcting them manually or by guessing, we might deduce that *VIGENER* is indeed the first part. It is not hard to find that *VIGENERE* are the first 8 chars of the key.

We have the following result so far:

```
P: SECCON{A_ _ _ _BCDEDEFG_ _ _ _KLMNOPQR_ _ _ _VWXYYZ}
K: VIGENERE_ _ _ _VIGENERE_ _ _ _VIGENERE_ _ _ _VIGENER
C: LMIG}RPED O E EWKJIQIWKJ W M NDTSR}TFVU F W YOCBAJBQ
```

It seems the alphabet is part of *p*. After *G* there might be *HIJ_* or maybe *_HIJ*. The same goes for *STU_* or *_STU* right after *R*. Before going for a bruteforce solution we decided to test a few possibilities manually. We tried *H* in position 23 and *S* in position 32. Surprisingly, we got *C* as the result for the key in both cases. Certainly a good sign. Trying the other chars we got *VIGENERECOD_* for the key. Not hard to guess the answer should be *VIGENERECODE*, proving our first guess was correct!

With the key in hands all we had to do was decode the ciphertext in order to obtain **SECCON{ABABABCDEDEFGHIJJKLMNOPQRSTTUVWXYYZ}**.

Although we used a lot of guessing to make things quicker, our next approach would be bruteforcing the given md5 hash. In fact, we decided to confirm our guesses with a little coding:

```
import itertools
import hashlib
import binascii

hashChall = "f528a6ab914c1ecf856a1d93103948fe"
res = ""

# the range could include the whole alphabet for more extensive search
for a in itertools.product("AB",repeat=4):
    for b in itertools.product("HIJJ",repeat=4):
        for c in itertools.product("STTU",repeat=4):
            pt = "SECCON{A"+"".join(a)+"BCDEDEFG"+"".join(b)+"KLMNOPQR"+"".join(c)+"VWXYYZ}"
            hashTst = hashlib.md5(pt.encode('utf-8')).hexdigest()
            if hashTst == hashChall:
                res = pt
print(res)
```

© 2017 GitHub, Inc. Terms Privacy Security Status Help

Contact GitHub API Training Shop Blog About