The-Climb

发表于 2021-01-08 分类于 Challenge， 2020， CSICTF， Crypto
Challenge | 2020 | CSICTF | Crypto | The-Climb

点击此处获得更好的阅读体验

WriteUp来源

https://dunsp4rce.github.io/csictf-2020/crypto/2020/07/21/The-Climb.html

by `raghul-rajasekar`

题目描述

> *We are not lost, we're right here somewhere on this little blue line. Wait, why do I feel like I'm being watched?*

Files:

- *theclimb.txt*
- *theclimb.java*

题目考点

解题思路

Solution

`theclimb.java` contains the following code:

```
1  public class Main
2  {
3      int kmatrix[][];
4      int tmatrix[];
5      int rmatrix[];
6
7      public void div(String temp, int size)
8      {
9          while (temp.length() > size)
10         {
11             String substr = temp.substring(0, size);
12             temp = temp.substring(size, temp.length());
13             perf(substr);
14         }
15         if (temp.length() == size)
16             perf(temp);
17         else if (temp.length() < size)
18         {
19             for (int i = temp.length(); i < size; i++)
20                 temp = temp + 'x';
21             perf(temp);
22         }
23     }
24
25     public void perf(String text)
26     {
27         textconv(text);
28         multiply(text.length());
29         res(text.length());
30     }
31
32     public void keyconv(String key, int len)
33     {
34         kmatrix = new int[len][len];
35         int c = 0;
36         for (int i = 0; i < len; i++)
37         {
```

```
38              for (int j = 0; j < len; j++)
39              {
40                  kmatrix[i][j] = ((int) key.charAt(c)) - 97;
41                  c++;
42              }
43          }
44      }
45
46      public void textconv(String text)
47      {
48          tmatrix = new int[text.length()];
49          for (int i = 0; i < text.length(); i++)
50          {
51              tmatrix[i] = ((int) text.charAt(i)) - 97;
52          }
53      }
54
55      public void multiply(int len)
56      {
57          rmatrix = new int[len];
58          for (int i = 0; i < len; i++)
59          {
60              for (int j = 0; j < len; j++)
61              {
62                  rmatrix[i] += kmatrix[i][j] * tmatrix[j];
63              }
64              rmatrix[i] %= 26;
65          }
66      }
67
68      public void res(int len)
69      {
70          String res = "";
71          for (int i = 0; i < len; i++)
72          {
73              res += (char) (rmatrix[i] + 97);
74          }
75          System.out.print(res);
76      }
77
78
79      public static void main(String[] args)
80      {
81          Main obj = new Main();
82          System.out.println("Enter the plain text: ");
83          String text = "fakeflag";
84          System.out.println(text);
85          System.out.println("Enter the key: ");
86          String key = "gybnqkurp";
87          System.out.println(key);
88          double root = Math.sqrt(key.length());
89          if (root != (long) root)
90              System.out.println("Invalid key length.");
91          else
92          {
93              int size = (int) root;
94
95                  System.out.println("Encrypted text = ");
96                  obj.keyconv(key, size);
97                  obj.div(text, size);
98          }
99      }
100 }
```

To summarize, it takes the key `gybnqkurp`, converts it into a 3x3 matrix and encodes each block of three characters by taking the dot product of the block with each row of the key matrix modulo 26. To reverse this, the easiest idea I could come up with was to simply iterate through all possible blocks of size 3 (totalling $26^3 = 17576$) for each block of the ciphertext, encrypt it and see if it matches with the ciphertext block. While I didn't attempt to rigorously prove this, the idea was that the number of pre-images for each ciphertext block would be low, so I could manually pick out the correct plaintext blocks. Luckily (or not, I'm still not sure), each ciphertext block corresponded to exactly one plaintext block.

```
import string
cipher = 'lrzlhhombgichae'
key = 'gybnqkurp'
def encrypt(text, key):
    keylist = []
    for c in key:
        keylist.append(ord(c) - ord('a'))
    textlist = []
    for i in range(3):
        for c in text:
            textlist.append(ord(c) - ord('a'))
    ret = ''
    for i in range(3):
        temp = 0
        for j in range(3):
            ind = i*3 + j
            temp += keylist[ind]*textlist[ind]
        ret += chr(temp%26 + ord('a'))
    return ret
cipher = [cipher[i:i+3] for i in range(0, len(cipher), 3)]
for s in cipher:
    for a1 in string.ascii_lowercase:
        for a2 in string.ascii_lowercase:
            for a3 in string.ascii_lowercase:
                if encrypt(a1+a2+a3, key) == s:
                    print(a1+a2+a3, end = '')
```

The output was `hillshaveeyesxx`, where `x` is for padding. Removing the padding and wrapping the rest in the flag format gave the final flag.

Flag

```
csictf{hillshaveeyes}
```

- ***本文作者：*** *CTFHub*
- ***本文链接：*** *https://writeup.ctfhub.com/Challenge/2020/CSICTF/Crypto/x544ciQXKD9ztLtsfrNc2f.html*
-

 # Challenge # 2020 # Crypto # CSICTF
little RSA
Quick Math