

TQLCTF2022 WriteUP（自己做+收集整理+持续更新）

原创

[rickliuxiao](#) 已于 2022-02-22 11:54:45 修改 574 收藏

文章标签: [TQLCTF 2022](#)

于 2022-02-22 11:31:16 首次发布

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: <https://blog.csdn.net/rickliuxiao/article/details/123064574>

版权

题目下载链接 (<https://download.csdn.net/download/rickliuxiao/81862001>)

[Misc]1.签到

关注微信公众号后, 获得1个链接, 在源码中找到flag。

[Misc]2. Wizard

访问链接, 打开web页面显示的内容如下:

```
SHA256('TQLCTF' + ?) starts with 58011
Please input the string:
```

于是, 猜测这题需要通过命令行 (CLI) 界面进行交互。

题目是对字符串 `'TQLCTF' + ?` 进行SHA256哈希计算, 得到的哈希值以 `58011` 开头。

显然, 我们需要根据server端发来的回显信息, 提取到哈希值 (`58011` 并不是固定的, 而是一直会变化)。

随便写一个脚本, 爆破sha256哈希值, 再发给server端。脚本如下:

```

from hashlib import sha256
import string
import itertools
import socket
import re

HOST='120.79.12.160'
PORT=41985
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect((HOST, PORT))

def brute_force(pad, shav):
    for i in range(20):
        for str in itertools.product(string.ascii_letters + string.digits, repeat=i):
            t = ''.join(str)
            ts = pad + t
            h = sha256(ts.encode('utf-8')).hexdigest()
            if h.startswith(shav):
                print(ts, t, h)
                return t

content = sock.recv(1024).strip().decode()
print(content)
start1 = len("SHA256('TQLCTF' + ?) starts with ")
pad = 'TQLCTF'
h = content[start1:start1+5]
# print(h)
s2 = brute_force(pad, h).encode()
# print(s2)
sock.send(s2)
sock.send('\n'.encode())
content = sock.recv(1024).strip().decode()
# print(content)

```

server端再返回了以下信息:

```

If you want to know Zard's secret, you need to play a game with him.
Zard has an array of n distinct integers. You can ask no more than n questions.
Each question contains m distinct positions. Zard will take the corresponding m numbers from his array and sort
them. However, he will only tell you the k-th number among them, in ascending order.
Your task is to guess the size of k.
(1<=k<=m<n<=10000)

You can perform two operations:

1. Query. Your query starts with a capital 'Q', followed by m positions. You will get the k-th element of the in
tegers corresponding to these positions in the array in ascending order. (e.g., Q 2 3 4 5. The answer is 8.)

2. Guess. Your guess starts with a capital 'G', followed by a number, which is k. If your guess is correct, you'
ll know Zard's secret. (e.g., G 3.)

For example:
Array = [1, 0, 9, 8, 2]
n = 5, m = 4, k = 3

[Query]
Q 2 3 4 5
8

[Guess]
G 3
You are so smart! You will get Zard's secret!

Let's start!
n = 1578, m = 245

```

题目大意就是：

```

(1<=k<=m<n<=10000)
1. n: 有一个数组array, 其中有n个互不相同的整数, 但是乱序的。
2. 我们可以提出n次问题。每次提问时, 包括m个数字。每个数字对应的是在数组array中的序号。这m个数字就对应于在数组中的m个数字, 假设
为array2。
题目会将array2进行排序后的第k个数字返回。但这个k是未知的。
3. 我们需要将k的值猜出来, 发回给server端。这样就能得到flag。

```

以 `n = 1578, m = 245` 为题, 我们可以这样解题:

```

1. 直接向服务器取数组array中的前m个数据。这里m=245。假设得到array[1..245]。
对这个数组进行排序, 得到array2[1..245]。
2. 返回的数字为knumber。
根据题意可知: knumber是array2[1..245]中的第k个数据。
3. 再引入数组array中的第 (m+1) 个数据, 即第246个数据array[246]。
由于每次只能发送m个数字, 所以, 需要用前面[1..245]中的一个数字q来替换为246。
(1) 如果数组array[q]<knumber, 且array[246]<knumber, 第k个数字仍然是knumber;
(2) 如果数组array[q]>knumber, 且array[246]>knumber, 第k个数字仍然是knumber;
(3) 如果数组array[q]<knumber, 且array[246]>knumber, 第k个数字变为k2number, 其中, k2number>knumber;
(4) 如果数组array[q]>knumber, 且array[246]<knumber, 第k个数字变为k2number, 其中, k2number<knumber;
由于数组中的数字都互不相同, 故array[1..245]中有k-1个数字小于knumber。
如果循环m次 (m=245) 用前面[1..245]中的一个数字q来替换为246, 每次发送数组[1..246]给server后返回一个数字:
(1) 当array[246]<knumber时, 返回knumber的数量就是k。
(2) 当array[246]>knumber时, 返回k2number的数量就是k。
【不知道这样推导正确不正确。求大佬指正!】

```

按照上面的思路, 写脚本如下:

```

# 前部分的脚本仍保持不变

```

```

# 前部分的脚本保持不变。
from hashlib import sha256
import string
import itertools
import socket
import re

HOST='120.79.12.160'
PORT=41985
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect((HOST, PORT))

def brute_force(pad, shav):
    for i in range(20):
        for str in itertools.product(string.ascii_letters + string.digits, repeat=i):
            t = ''.join(str)
            ts = pad + t
            h = sha256(ts.encode('utf-8')).hexdigest()
            if h.startswith(shav):
                print(ts, t, h)
                return t

content = sock.recv(1024).strip().decode()
print(content)
start1 = len("SHA256('TQLCTF' + ?) starts with ")
pad = 'TQLCTF'
h = content[start1:start1+5]
# print(h)
s2 = brute_force(pad, h).encode()
# print(s2)
sock.send(s2)
sock.send('\n'.encode())
content = sock.recv(1024).strip().decode()
# print(content)

# =====以下是新增的部分脚本：=====
# 提取n和m。
last = content.split('\n')[-1]
print(last)
n = re.findall(r'n = (\d.*),', last)[0]
m = re.findall(r', m = (\d.*)', last)[0]
print(n,m)

# 先向server端发送m个数字[1..m]，得到knumber
lq = ["Q"]
for i in range(int(m)):
    lq.append(str(i+1))

q = ' '.join(lq)
# print(q)
sock.send(q.encode())
sock.send('\n'.encode())
content = sock.recv(1024).strip().decode()
print(content)
knumber = content

# 循环m次：每次循环时，将[1..m]中的一个数字替换为 (m+1)
tc = [content]
for i in range(int(m)):
    lq = ["Q"]

```


Original Text: (length: 559)

```
KGR/QRI 10646-1 zswtqgg d tnxcs tsdtofbrx osk ndnzhl gna
Ietygfviy Idoilfvsu Arz (QQJ) hkkqk maikaglvusv ubyyp cw
ekg krzyj'o kitwkbj alypsdd. Wjs rzvmebrwoa duwcuosu
pqecqgamo cw ekg IFA, uussmpu, ysum aup qfxschllyk swks
pcbb khxsee drdoqppwfyv cbg xeupctzou, oql gneg ylv nsg
bb zds upygzrxzkjh fq XVT-8, wpr uxxvnm qt wpyv isdz. XVT-
8 kif zds tsdtofbrxegktf qt szryafmtqi hkm sahz LD-DUQLQ
egjuv, auqjllvtc qfxschljvrehp hlrv iqyk omjehog, sieyafj
lqf cwprx ocwezcfh bugp fwvb qb XA-NYYWZ gdniha oap oip
wtoqacgnsee wq cwprx rocfnhu. HTTPZB{QFOLP6_KRZ1Q}
```

Hidden Text: (length: 0)

Steganography Text: (length: 559)

```
KGR/QRI 10646-1 zswtqgg d tnxcs tsdtofbrx osk ndnzhl gna
Ietygfviy Idoilfvsu Arz (QQJ) hkkqk maikaglvusv ubyyp cw
ekg krzyj'o kitwkbj alypsdd. Wjs rzvmebrwoa duwcuosu
pqecqgamo cw ekg IFA, uussmpu, ysum aup qfxschllyk swks
pcbb khxsee drdoqppwfyv cbg xeupctzou, oql gneg ylv nsg
bb zds upygzrxzkjh fq XVT-8, wpr uxxvnm qt wpyv isdz. XVT-
8 kif zds tsdtofbrxegktf qt szryafmtqi hkm sahz LD-DUQLQ
egjuv, auqjllvtc qfxschljvrehp hlrv iqyk omjehog, sieyafj
lqf cwprx ocwezcfh bugp fwvb qb XA-NYYWZ gdniha oap oip
wtoqacgnsee wq cwprx rocfnhu. HTTPZB{QFOLP6_KRZ1Q}
```

[Download Stego Text as File](#)

CSDN @rickliuxiao

在末尾发现字符串：HTTPZB{QFOLP6_KRZ1Q}

猜测其格式应为 TQLCTF{...}，估计是Vigenere编码。但密码未知。

在以下网站 (<https://www.guballa.de/vigenere-solver>) 进行爆破，得到flag。

Input

Cipher Text:

```
KGR/QRI 10646-1 zswtqgg d tnxcs tsatorbrx osk nanznl gna
Ietygfviy Idoilfvsu Arz (QQJ) hkkqk maikaglvusv ubyyp cw
ekg krzyj'o kitwkbj alypsdd. Wjs rzvmebrwoa duwcuosu pqecqgamo
cw ekg IFA, uussmpu, ysum aup qfxschllyk swks pcbb khxsee
drdoqppwfyv cbg xeupctzou, oql gneg ylv nsg bb zds
upygzrxzkjh fq XVT-8, wpr uxxvnm qt wpyv isdz. XVT-8 kif zds
tsdtofbrxegktf qt szryafmtqi hkm sahz LD-DUQLQ egjuv,
auqjllvtc qfxschljvrehp hlrv iqyk omjehog, sieyafj lqf cwprx
ocwezcfh bugp fwvb qb XA-NYYWZ gdniha oap oip wtoqacgnsee wq
cwprx rocfnhu. HTTPZB{QFOLP6_KRZ1Q}
```

Cipher Variant:

Language:

Key Length:
(e.g. 8 or a range e.g. 6-10)

Result

[Clear text \[hide\]](#)

Clear text using key "codingworld":

```
ISO/IEC 10646-1 defines a large character set called the Universal
Character Set (UCS) which encompasses most of the world's writing
systems. The originally proposed encodings of the UCS, however, were
not compatible with many current applications and protocols, and this
has led to the development of UTF-8, the object of this memo. UTF-8 has
the characteristic of preserving the full US-ASCII range, providing
compatibility with file systems, parsers and other software that rely
on US-ASCII values but are transparent to other values.
```

TQLCTF{CODIN6_WORLD}

[Details \[show\]](#)

Key length statistics [\[show\]](#)

Histogram [\[show\]](#)

CSDN @rickliuxiao