【WriteUp】全国大学生信息安全竞赛CISCN2021-东北赛区-Maple_root

原创

部分题目WP包含图片，因为CSDN无法获取外部图片所以无法加入，如想观看完整版本欢迎点此访问

> 参赛队员：
> x0r，b477eRy，f1oat

总结

> 最终成绩：3627
> 最终排名：13
> 一血数量：3

本次比赛前期一些顺利，后期感觉被py爆了，结果名次就拉了下来，整体题目全部都偏向MISC，打的很迷惑，但是算不上难（RE除外），希望下次国赛能进决赛看看…

MISC

MISC_签到

打开以后压缩包内有一个二维码文件，利用压缩包内的二维码扫描器扫描后即可得到flag

Sudoku

根据题目判断是一个数独解密，下载以后在压缩包内有一个flag压缩包和一个数独游戏的excel表格，打开以后是一个数独游戏，直接放入z3解密工具解出答案以后按照从左上到右下的对角线顺序依次输入以后解开flag压缩包得到最终结果

Vigenère

下载文件后使用binwalk解出一个`b.txt`文件，根据题目标题判断是维吉尼亚密码，但是没有找到key，用脚本爆破

```
import itertools
import string
import sys
import textwrap

"""
Run this script in a shell with the ciphertext to decode on STDIN
"""

############################################################################################################
# Vienere encryption and decryption functions
############################################################################################################

def vigenere(plaintext, key, a_is_zero=True):
    key = key.lower()
    if not all(k in string.ascii_lowercase for k in key):
        raise ValueError("Invalid key {!r}; the key can only consist of English letters".format(key))
    key_iter = itertools.cycle(map(ord, key))
    return "".join(
        chr(ord('a') + (
            (next(key_iter) - ord('a') + ord(letter) - ord('a'))    # Calculate shifted value
            + (0 if a_is_zero else 2)                               # Account for non-zero indexing
            ) % 26) if letter in string.ascii_lowercase             # Ignore non-alphabetic chars
        else letter
        for letter in plaintext.lower()
    )

def vigenere_decrypt(ciphertext, key, a_is_zero=True):
    # Decryption is encryption with the inverse key
    key_ind = [ord(k) - ord('a') for k in key.lower()]
    inverse = "".join(chr(ord('a') +
            ((26 if a_is_zero else 22) -
                (ord(k) - ord('a'))
            ) % 26) for k in key)
    return vigenere(ciphertext, inverse, a_is_zero)

def test_vigenere(text, key, a_is_zero=True):
    ciphertext = vigenere(text, key, a_is_zero)
    plaintext  = vigenere_decrypt(ciphertext, key, a_is_zero)
    assert plaintext == text, "{!r} -> {!r} -> {!r} (a {}= 0)".format(
        text, ciphertext, plaintext, "" if a_is_zero else "!")

# Test that the Vigenere encrypt and decrypt work (or are at least inverses)
for text in ["rewind", "text with spaces", "pun.ctuation", "numb3rs"]:
    for key in ["iepw", "aceaq", "safe", "pwa"]:
        test_vigenere(text, key, True)
        test_vigenere(text, key, False)

# Now that we're sure that all the vigenere stuff is working...

############################################################################################################
# Cipher solver
############################################################################################################
```

```
##########################################################################################################

# From http://code.activestate.com/recipes/142813-deciphering-caesar-code/
ENGLISH_FREQ = (0.0749, 0.0129, 0.0354, 0.0362, 0.1400, 0.0218, 0.0174, 0.0422, 0.0665, 0.0027, 0.0047,
                0.0357, 0.0339, 0.0674, 0.0737, 0.0243, 0.0026, 0.0614, 0.0695, 0.0985, 0.0300, 0.0116,
                0.0169, 0.0028, 0.0164, 0.0004)

def compare_freq(text):
    """
    Compare the letter distribution of the given text with normal English. Lower is closer.

    Performs a simple sum of absolute difference for each letter
    """
    if not text:
        return None
    text = [t for t in text.lower() if t in string.ascii_lowercase]
    freq = [0] * 26
    total = float(len(text))
    for l in text:
        freq[ord(l) - ord('a')] += 1
    return sum(abs(f / total - E) for f, E in zip(freq, ENGLISH_FREQ))


def solve_vigenere(text, key_min_size=None, key_max_size=None, a_is_zero=True):
    """
    Solve a Vigenere cipher by finding keys such that the plaintext resembles English

    Returns:
        the first and second best from the set of best keys for each length

    This is not a brute force solver; instead, it takes advantage of a weakness in the cipher to
    solve in O(n * K^2) where n is the length of the text to decrypt and K is the length of the
    longest key to try.

    The idea is that for any key length, the key is used repeatedly, so if the key is of length k
    and we take every k'th letter, those letters should have approximately the same distribution as
    the English language on a whole. Furthermore, since each letter in the key is independent, we
    can perform the analysis for each letter in the key by taking every k'th letter at different
    starting offsets. Then, since the letters in the key are independent, we can construct the best
    key for a given length by simply joining the best candidates for each position.
    """
    best_keys = []
    key_min_size = key_min_size or 1
    key_max_size = key_max_size or 20

    text_letters = [c for c in text.lower() if c in string.ascii_lowercase]

    for key_length in range(key_min_size, key_max_size):
        # Try all possible key lengths
        key = [None] * key_length
        for key_index in range(key_length):
            letters = "".join(itertools.islice(text_letters, key_index, None, key_length))
            shifts = []
            for key_char in string.ascii_lowercase:
                shifts.append(
                    (compare_freq(vigenere_decrypt(letters, key_char, a_is_zero)), key_char)
                )
            key[key_index] = min(shifts, key=lambda x: x[0])[1]
        best_keys.append("".join(key))
    best_keys.sort(key=lambda key: compare_freq(vigenere_decrypt(text, key, a_is_zero)))
```

```
    return best_keys[:2]

CIPHERTEXT = sys.stdin.read().strip()

print "Solving Vigenere cipher:"
print "*" * 80
print textwrap.fill(CIPHERTEXT, 80)
print "*" * 80
for key in reversed(solve_vigenere(CIPHERTEXT)):
    print ""
    print "Found key: {!r}".format(key)
    print "Solution:"
    print "=" * 80
    print textwrap.fill(vigenere_decrypt(CIPHERTEXT, key))
    print "=" * 80
```

得出了key`faisnigslk`并得到了明文，最后MD5哈希后提交即为flag

flagpng

png宽爆破，使用`crc32fix`工具修复后flag直接写脸上了。

easy_rsa

密码学的模板题...不知道怎么说，直接有现成的解密脚本

```
import ContinuedFractions, Arithmetic, RSAvulnerableKeyGenerator
import binascii

def hack_RSA(e,n):
    '''
    Finds d knowing (e,n)
    applying the Wiener continued fraction attack
    '''
    frac = ContinuedFractions.rational_to_contfrac(e, n)
    convergents = ContinuedFractions.convergents_from_contfrac(frac)

    for (k,d) in convergents:

        #check if d is actually the key
        if k!=0 and (e*d-1)%k == 0:
            phi = (e*d-1)//k
            s = n - phi + 1
            # check if the equation x^2 - s*x + n = 0
            # has integer roots
            discr = s*s - 4*n
            if(discr>=0):
                t = Arithmetic.is_perfect_square(discr)
                if t!=-1 and (s+t)%2==0:
                    print("Hacked!")
                    return d

# TEST functions

def test_hack_RSA():
    print("Testing Wiener Attack")

    while(times>0):
        e,n,d = RSAvulnerableKeyGenerator.generateKeys(1024)
```

```
        print("(e,n) is (", e, ", ", n, ")")
        print("d = ", d)

        hacked_d = hack_RSA(e, n)

        if d == hacked_d:
            print("Hack WORKED!")
        else:
            print("Hack FAILED")

        print("d = ", d, ", hacked_d = ", hacked_d)
        print("--------------------------")
        times -= 1

if __name__ == "__main__":
    #test_is_perfect_square()
    #print("--------------------------")
    # test_hack_RSA()
    e = 93233329287134031153658342577279978858147660880050161825720091363568871279795659501331245709194924178139
07072362183263242872600968722751009728047372771888563967063415867914583643875685579148368802107991838829017791501740605034519922617995768757427887742433903105607196347897202199929749468203148029395725803535
    n = 10831784196037194486387996324752026726447279717417499260785686734674917217298919391626641928852084345413
70193744078154888072589708037117486860213089624795029582525501783298026959443870222339003799747202112246474259375161019073230508249672271697738321500894559008261698558072028050806042318719109646040290668273
    c = 62967132169895897004578576202001003381484927788637734193032964531847340267617591251480081297436355598128
71298354543444896395148951193742778334307991495130689300556153305166624284798657245079812375827793536448004235134853577187239085545439152401179954644191658232147484965697358446855686878564958349009996822935
    d=hack_RSA(e, n)
    print('d=',d)
    m=pow(c, d,n)
    print('m=',m)
    b = hex(m)  #转换成相同的字符串即'0x665554'
    b = b[2:]   #截取掉'0x'
    c = binascii.a2b_hex(b) #转换成ASCii编码的字符串
    print(c)
```

huahua

下载以后发现压缩包受损，在`010editor`中把压缩包头改为`504b0304`解压后得到一张png图片，也是缺少png头，加上`89504e47`后工具中又提示crc报错，那就再把高改一下即可得到flag

WEB

简单的注入

发一个登录请求，burp存下来
用sqlmap跑出时间盲注，跑出库名表名，
然后在lostandfound.user表里`select * from user where username='admin';`
出密码`Admin@12999 ...`，登录拿到flag。

Flagin

先用burp抓个包，怀疑是xee
waf拦截后去掉`<?xml`可过
最终payload:

```
<!DOCTYPE xxe [
<!ELEMENT name ANY >
<!ENTITY xxe SYSTEM "php://read=convert.base64-encode/resource=/flag.txt" >]>
<user><username>&xxe;</username><password>admin</password></user>
```

Be_Careful

上来careful的index.php是个链接
点进去是?file=a.php来着？ 里面是there is nothing
然后改成php伪协议base64读index.php源码
看到过滤列表和结尾注释的real_flag.php
这里过滤了存flag的文件 但没过滤real_flag.php
再读real_flag.php的源码 构造?a获得flag

Crypto

Sign_me_up

下载打开文件以后发现是一大段的`BASE64`加密后的内容，疑似多次套娃，多次解密以后即得到密文内容`Welcome to ciscn.`
`flag{4b7fb332177a5df1b4aa6ba53e29d8fd}`

Super_Man

依然是`010editor`打开图片，文件头88改为89后图片变为超人图片，接着在文件尾找到salted，放入`cyperchef`解两层即可得到flag

凯撒大帝的Unicode

根据unicode提示，将unicode转化为数字后尝试偏移(25000-24000)(忽略负数)到ascii区域进行解码
偏移到24591时得到字符串`'k_\\Here&comes$the]un1c0de[f1Ag1s{3c3b832287a5578b3511fcd7486f4ef9}&%O##%&P!#TQR%"&$'`
脚本:

```
f = open("caesar", "r", encoding="utf-8").read()

f = f.split("\n")[1]
f = [int(c.encode("unicode-escape").decode("utf-8").replace("\\u", ""), 16) for c in f]
offset = 25000
while True:
    # 尝试过用unicode还原回去 没找到
    # print(''.join([hex(b-offset).replace("0x", "\\u0" if len(hex(b-offset).replace("-", "")) == 5 else "\\u").
encode().decode("unicode-escape") for b in f]))
    try:
        print(f"offset:{offset} result: " + ''.join([chr(b-offset) for b in f]))
    except:
        continue
    finally:
        offset -= 1
    print("-------")
```

可读出flag

cipertext

古典密码题...打开之后一段ook密码和一段BrainFuck以及一段jsfuck密码解密以后连起来就是flag

RE

RE_签到

本题拿入后在IDA7.5中多次查看无果，只能分析出是一个调用了官方库的哈希加密工具，后来在Linux环境下分别使用了`strings`+`greg`和`cat`查看，发现在cat后文件的最后有一段形如`flag{xxx}`的内容，即为最后答案，但是这里有一点很不解的就是为何使用strings无法正常找到flag段，这一点有待研究。