WRITE UP FINAL GEMASTIK XII

b333f

Hans Prama Setiawan

M Faqih Jihan Insani

Rafie Muhammad

Crypto

berAESnang-senang

Deskripsi : Seorang office boy yg bekerja di JYP Entertainment mengetahui bahwa para member TWICE menggunakan enkripsi tertentu untuk mengirimkan pesan rahasia mereka. Dengan penuh skill yang diajarkan mamank hekel, sang office boy menemukan kode untuk merahasiakan pesan mereka. Bantu sang office boy untuk memecahkan pesan rahasia "Senang-senang v0" dengan mengakses
nc min4tozaki.me 31000

Diberikan sebuah alamat service dan sebuah file python yang bernama s34cret.py. Saat melakukan nc ke service, terdapat sebuah string yang diberi keterangan “senang-senang v0” dan beberapa pilihan.

```
> nc min4tozaki.me 31000

            Selamat datang di Gemastik 12
              Selamat bersenang-senang!
               (https://min4tozaki.me)
                 Jangan aneh-aneh ya

Senang-senang v0 :
82c5b91665312dccaec616fc001fee0ac6822484e229d59afbdce6cf8cc9637143fabc9eb2047c7e
4bedd3b5c604ba65094fde0ab902603679f8df7337224bff3da8ff471ba4ba1f323b5f3d9ee91a6f
85d9be0daaf45474b176a9b70c13ecbd4e215720fd14585765a4b06f5cc462ce

Menu:

    1. Tebak Senang-senang v0
    2. Senang-senang v1
    3. Keluar

Pilih :
```

Saat menganalisis file python yang diberikan, terdapat beberapa fungsi diantaranya:

1. Fungsi pad. Fungsi ini akan melakukan padding terhadap pesan (msg) sepanjang ((16*n) – 1) char yang digunakan untuk melakukan padding adalah string dari angka v+5.
2. Fungsi cpadding. Fungsi ini mengembalikan char random antara \x00 sampai \xFF sepanjang 16 karakter.
3. Fungsi encryptSenang_senang. Fungsi ini akan melakukan enkripsi AES terhadap string di variabel plain yang sudah diconcat dengan string dari angka v dengan menggunakan kunci dan IV yang merupakan argumen fungsi tersebut. Namun, mode enkripsi AES tidak diberi tahu sama sekali. Selanjutnya, hasil enkripsi akan di-concat dengan cpadding dan IV. Dari hasil concat tersebut, hasil enkripsi berada di antara 48 char pertama dan 16 char terakhir.

```python
def pad(msg, v):
    if len(msg) % 16 != 0:
        msg = msg + str(v + 5) * ((16 - len(msg) % 16) - 1)
    else:
        msg = pad(msg + str(v + 5), v)
    return msg


def cpadding():
    padding = os.urandom(16)
    return padding


def encryptSenang_senang(iv, key, plain, v):
    cipher = AES.new(key, AES.MODE_***, iv)
    return (cpadding() + iv + cpadding() + cipher.encrypt(plain + str(v)) + cpadding()).encode('hex')
```

s

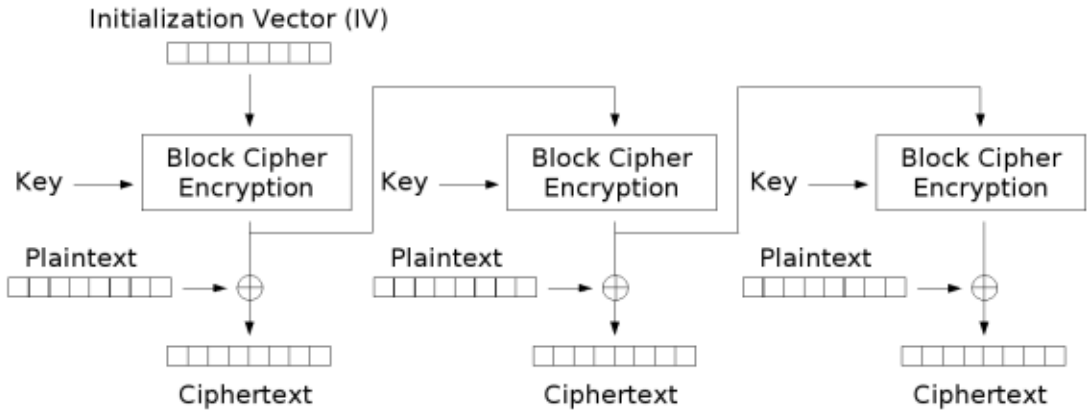
setelah dipelajari lebih lanjut, string yang diberi keterangan “senang-senang v0” merupakan hasil enkripsi dari variabel msg menggunakan fungsi encryptSenang_senang dan sudah dipadding.

```python
def encryptSenang_senang(iv, key, plain, v):
    cipher = AES.new(key, AES.MODE_***, iv)
    return (cpadding() + iv + cpadding() + cipher.encrypt(plain + str(v)) + cpadding()).encode('hex')


print """
            Selamat datang di Gemastik 12
            Selamat bersenang-senang!
            (https://min4tozaki.me)
            Jangan aneh-aneh ya
"""
print "Senang-senang v" + str(i), ":"
print encryptSenang_senang(iv, key, pad(msg, i), i)
i += 1
while 1:
    print"""
Menu:
```

Key dan IV merupakan string random sepanjang 16 karakter yang dihasilkan dari fungsi os.urandom.

```python
key = os.urandom(16)
iv = os.urandom(16)
msg = "-isi pesan member TWICE-"
i = 0
```

Karena mode enkripsi di-”sensor”, kami menduga bahwa mode enkripsinya merupakan kelemahan dari program ini. Kami lalu berasumsi bahwa mode enkripsi yang digunakan adalah Output Feedback (OFB)



Output Feedback (OFB) mode encryption

Mode ini bekerja dengan melakukan xor blok plaintext dengan hasil enkripsi dari IV menggunakan suatu kunci untuk menghasilkan ciphertext. Hasil enkripsi IV tadi akan dienkripsi lagi dengan menggunakan key yang sama untuk melakukan xor pada blok selanjutnya dari plaintext. Berarti, tiap-tiap blok hasil enkripsi IV yang digunakan untuk melakukan xor pada plaintext akan sama pada proses enkripsi yang berbeda. Jika kita mengetahui suatu plaintext dari ciphertext tertentu, maka kita dapat melakukan serangakn known plaintext pada enkripsi yang lain dengan menggunakan hasil xor dari ciphertext dan plaintext dan melakukan xor dari hasil tadi dengan ciphertext yang lain.

Pada program tersebut, IV dan Key digenerate pada awal program dan tidak diupdate sepanjang jalannya program, jadi blok hasil enkripsi IV akan selalu sama. Untuk menguji dugaan kami, kami mengumpulkan dua ciphertext yang sudah kami ketahui plaintextnya dengan menggunakan pilihan “2” pada program untuk menghasilkan ciphertext.

3sv.py

```
from pwn import *
r = remote('min4tozaki.me', 31000)

def tes(msg):
    r.sendlineafter('Pilih : ', '2')
    r.sendlineafter(' : ', msg)
    r.recvuntil(' :\n')
    return r.recvline()[:-1][96:-32]

a = tes('AAAA')
```

```
b = tes('BBBB')

print a
print b
```

```
> py 3sv.py
[+] Opening connection to min4tozaki.me on port 31000: Done
df90e717e6426adab8810c5673656351
dc93e414e7436bdbb9800d5772646252
[*] Closed connection to min4tozaki.me port 31000
```

string pertama merupakan ciphertext dari “AAAA” dan string kedua merupakan ciphertext dari “BBBB”. Pertama-tama, kita lakukan xor terhadap string pertama dengan “AAAA”

```
> py
Python 2.7.15+ (default, Feb  3 2019, 13:13:16)
[GCC 8.2.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> a = 'df90e717e6426adab8810c5673656351'.decode('hex')
>>> b = 'dc93e414e7436bdbb9800d5772646252'.decode('hex')
>>> k = ''
>>> for i in range(4):
...  k += chr(ord(a[i]) ^ ord('A'))
...
>>> k
'\x9e\xd1\xa6V'
```

Hasil xor tadi kemudian dixor dengan ciphertext kedua

```
>>> m = ''
>>> for i in range(4):
...  m += chr(ord(k[i]) ^ ord(b[i]))
...
>>> m
'BBBB'
>>>
```

Hasil yang didapatkan dari xor tersebut adalah plaintext ‘BBBB’. Berarti program tersebut menggunakan AES dengan mode OFB.

Untuk mendapatkan pesan dari string “senang-senang v0”, kami melakukan enkripsi string yang terdiri dari 100 char A. Selanjutnya setiap char hasil enkripsi dixor dengan char A. Kemudian,

hasil xor tersebut dixor dengan hasil enkripsi string “senang-senang v0”. Berikut script yang kami gunakan

```
from pwn import *
r = remote('min4tozaki.me', 31000)

def tes(msg):
    r.sendlineafter('Pilih : ', '2')
    r.sendlineafter(' : ', msg)
    r.recvuntil(' :\n')
    return r.recvline()[:-1][96:-32]

r.recvuntil(' :\n')
fl = r.recvline()[:-1][96:-32].decode('hex')

c = tes('A'*100).decode('hex')
k = ''

for i in c:
    k += chr(ord(i) ^ ord('A'))

flag = ''
for i in range(len(fl)):
    flag += chr(ord(fl[i]) ^ ord(k[i]))
print flag
```

saat dijalankan kami mendapatkan string berikut

```
> py sv.py
[+] Opening connection to min4tozaki.me on port 31000: Done
4E5_0fb_4lW4y5_m4kE_mE_fE3L_5p3c14L5555555555550
[*] Closed connection to min4tozaki.me port 31000
```

angka 5 dan 0 merupakan padding dari program. Jadi string dari “senang-senang v0” adalah 4E5_0fb_4lW4y5_m4kE_mE_fE3L_5p3c14L. Ketika dimasukkan ke pilihan satu, kami mendapatkan flag

```
> nc min4tozaki.me 31000

            Selamat datang di Gemastik 12
              Selamat bersenang-senang!
               (https://min4tozaki.me)
                 Jangan aneh-aneh ya

Senang-senang v0 :
19b20fd3b51e1bddae0708b5eb376b41c88f8269fc548e1a51c1e3c9928d41d91a125ed61a116552
9537a5444fcae60a78718f9d405168feddad26f85e6a001d444083a9c62885c5b631c050654612b5
7ebb47ff6a5a337bc8b492d59143731d0aa26522eb035c66a4ebe4cad3f2fcc7

Menu:

    1. Tebak Senang-senang v0
    2. Senang-senang v1
    3. Keluar

Pilih : 1
Input Senang-senang v0 : 4E5_0fb_4lW4y5_m4kE_mE_fE3L_5p3c14L

Selamat!!!
flag : gemastik12{4E5_0fb_4lW4y5_m4kE_mE_fE3L_5p3c14L}
```

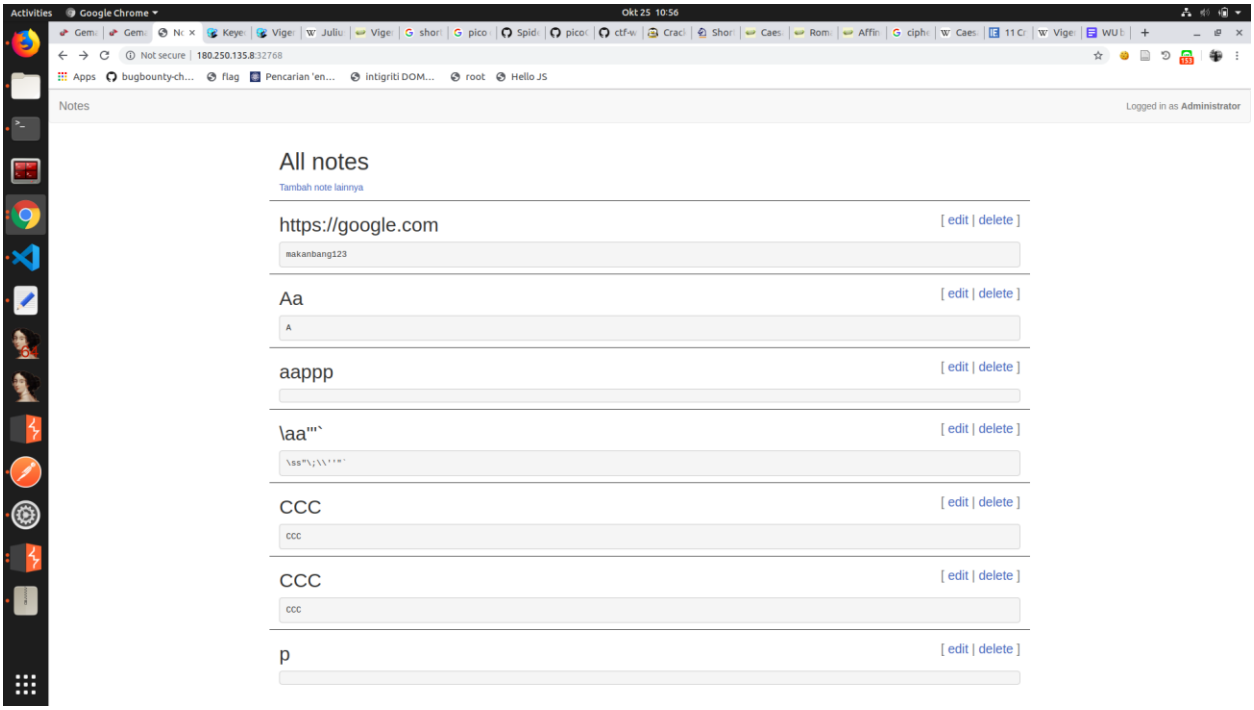
FLAG : gemastik12{4E5_0fb_4lW4y5_m4kE_mE_fE3L_5p3c14L}

Web App

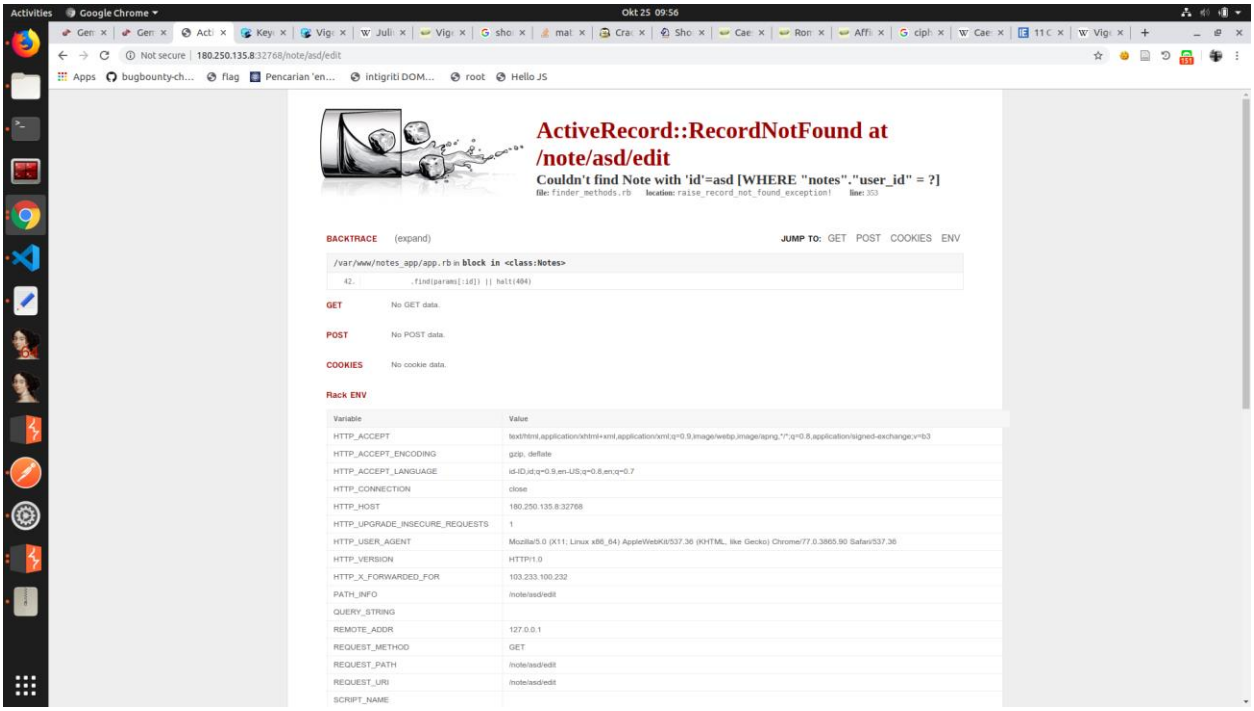
Aplikasi Note

Deskripsi : gali dan temukan kuncinya http://180.250.135.8:32768/

Diberikan sebuah service website dengan tampilan depan :



Setelah dilihat dan dicoba, service web memiliki fitur add, edit dan delete notes. Dan setelah di tes, ternyata kita dapat mentrigger XSS pada laman web pada konten body dari notes. Tetapi setelah, dicoba, kami tidak menemukan flag dari cara ini (cookie stealing). Setelah itu kami mencoba untuk melalukan testing pada parameter note_id, ternyata web menampilkan error apabila kita coba dengan id yang tidak numeric : http://180.250.135.8:32768/note/asd/edit



Dari error ini, kami lihat bahwa webservice menggunakan ruby dengan program mainnya terdapat pada : /var/www/notes_app/app.rb

Kami lalu mencoba untuk langsung mengakses file main-nya pada url : http://180.250.135.8:32768/app.rb dan ternyata kami bisa mendapatkan source code dari program tersebut. Berikut merupakan source codenya :

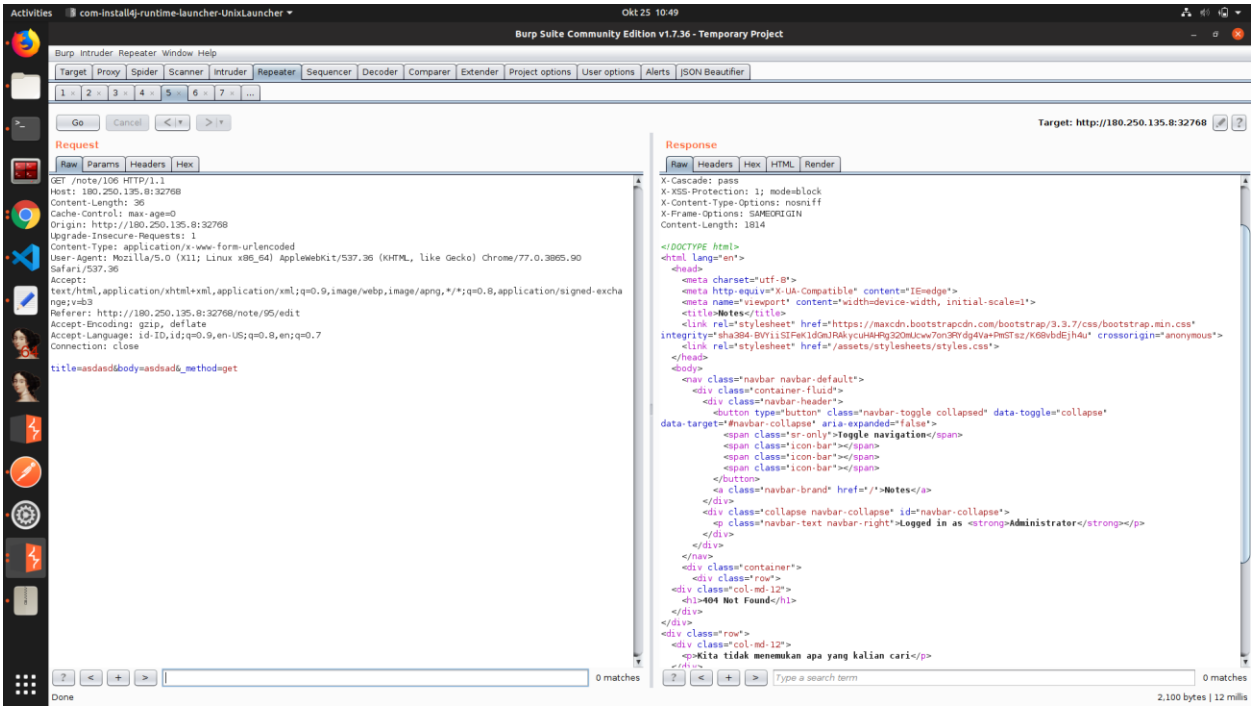
app.rb

```
require 'sinatra'
require 'active_record'
require_relative 'config/database'

# Logged in as "Administrator"
set :user_id, 1

class Notes < Sinatra::Application
  get '/' do
    if params[:filter]
      sql = "select #{params[:filter]} from notes where user_id = #{settings.user_id}"
      @notes = ActiveRecord::Base.connection.exec_query(sql)
    else
      @notes = Note.where(user_id: settings.user_id).all
    end
```

```
  erb :index
end

get '/note/add' do
  erb :add_note
end

post '/note/add' do
  begin
    @note = Note.new(
      title: params[:title],
      body: params[:body],
      user_id: settings.user_id
    )
    @note.save!
    redirect '/'
  rescue ActiveRecord::RecordInvalid => e
    @errors = e
    erb :add_note
  end
end

get '/note/:id/edit' do
  @note = Note
          .where(user_id: settings.user_id)
          .find(params[:id]) || halt(404)
  erb :edit_note
end

put '/note/:id/edit' do
  begin
    @note = Note
            .where(user_id: settings.user_id)
            .find(params[:id]) || halt(404)
    @note.update!(
      title: params[:title],
      body: params[:body]
    )
    redirect '/'
  rescue ActiveRecord::RecordInvalid => e
    @errors = e
    erb :edit_note
  end
end
```

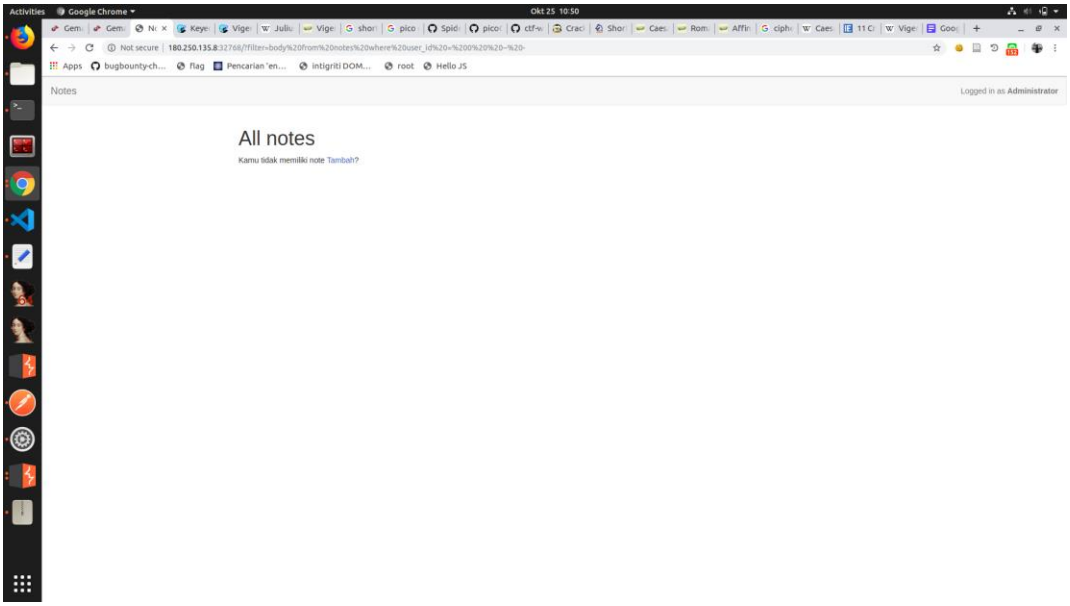
```
  get '/note/:id/delete' do
    @note = Note
          .where(user_id: settings.user_id)
          .find(params[:id]) || halt(404)
    erb :delete_note
  end

  delete '/note/:id/delete' do
    @note = Note
          .where(user_id: settings.user_id)
          .find(params[:id]) || halt(404)
    @note.delete
    redirect '/'
  end
end

# Show nice 404 pages
not_found do
  erb :not_found
end
```

Ternyata pada root url “/”, terdapat suatu proses query SQL apabila terdapat parameter “filter” pada url. Kita lihat pada source code juga bahwa user_id telah di set secara default sebagai 1. Kami lalu mencoba melakukan teknik SQL injection pada root endpoint dari web pada parameter “filter”. Kami mencoba untuk mendapatkan nilai dari title/body pada tabel notes. Nilai column ini kami asumsikan sama dengan nilai post parameter saat menambah notes :



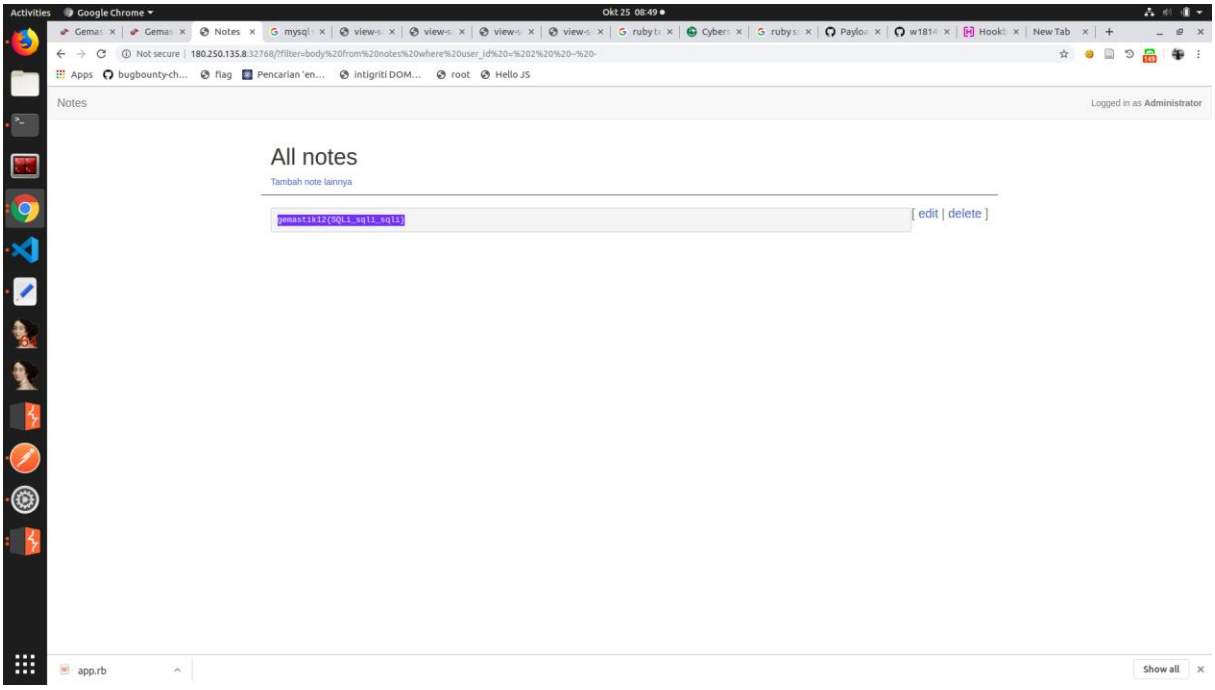
Setelah itu, kami coba mendapatkan semua nilai dari column “body” dengan user_id default yaitu 1, ternyata, tidak ditemukan flag. Lalu kami mencoba dengan user_id = 0, ternyata kosong :

```
?filter=body%20from%20notes%20where%20user_id%20=%200%20%20--%20-
```



Lalu kami mencoba dengan user_id = 2, dan kami dapatkan flagnya :

```
?filter=body%20from%20notes%20where%20user_id%20=%202%20%20--%20-
```



FLAG : gemastik12{SQLi_sqli_sqli}

Forensic

Disk Forensic (200 pts)

Seorang penyidik polisi menemukan image dari suatu file system. temukan flags dalam image filesystem ini
download image di http://10.251.251.194/forensic.img

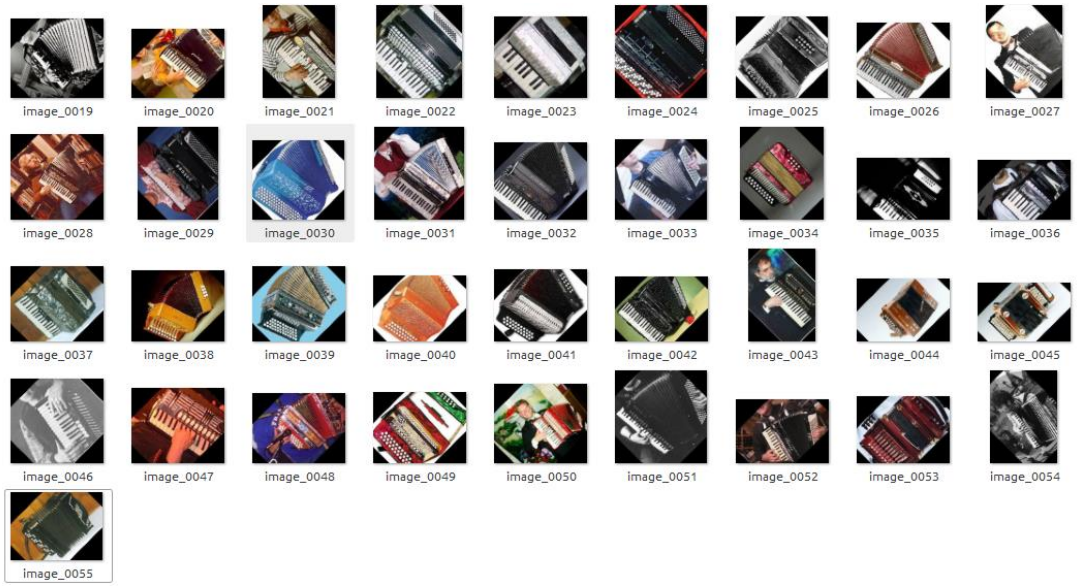
Diberikan **disk image** berupa **forensic.img**. Setelah beberapa saat dilakukan penelusuran, diketahui bahwa **diskimage** yang diberikan tidak valid. Dari sini kami berasumsi bahwa **diskimage** merupakan Ext2 partition mengingat terdapat directory **/home/tunky/forensic**. Adapun dilakukan proses file-carving dengan script sebagai berikut

```
$ xxd -p forensic.img | tr -d '\n' | sed
's/8dd7af5d0500ffff20ff010001000000/8dd7af5d0500ffff53ef010001000000/g' | xxd -r -
p > fixed.img

$ file fixed.img
fixed.img: Linux rev 1.0 ext2 filesystem data, UUID=5424215a-8103-4a3d-8687-
b4ed6f74fef2 (large files)

$ fls fixed.img
d/d 11: lost+found
r/r 12: wkwkwkw
d/d 1281:       accordion
V/V 2561:       $OrphanFiles
```

Setelah itu, Kami lakukan proses mount sedemikian hingga diperoleh flag dari image_0017.jpg pada directory **accordion/**

```
$ sudo mount fixed.img a/
$ ls a/accordion/
image_0001.jpg  image_0008.jpg  image_0015.jpg  image_0022.jpg  image_0029.jpg
image_0036.jpg  image_0043.jpg  image_0050.jpg
image_0002.jpg  image_0009.jpg  image_0016.jpg  image_0023.jpg  image_0030.jpg
image_0037.jpg  image_0044.jpg  image_0051.jpg
image_0003.jpg  image_0010.jpg  image_0017.jpg  image_0024.jpg  image_0031.jpg
image_0038.jpg  image_0045.jpg  image_0052.jpg
image_0004.jpg  image_0011.jpg  image_0018.jpg  image_0025.jpg  image_0032.jpg
image_0039.jpg  image_0046.jpg  image_0053.jpg
```

image_0005.jpg image_0012.jpg image_0019.jpg image_0026.jpg image_0033.jpg image_0040.jpg image_0047.jpg image_0054.jpg
image_0006.jpg image_0013.jpg image_0020.jpg image_0027.jpg image_0034.jpg image_0041.jpg image_0048.jpg image_0055.jpg
image_0007.jpg image_0014.jpg image_0021.jpg image_0028.jpg image_0035.jpg image_0042.jpg image_0049.jpg



FLAG : gemastik12{Disk_forensic_is Eazy}

ftp forensik

Bantu Badrun menemukan flag yang hilang dalam file dokumen miliknya

Diberikan berkas **paket data** bernama **trafik-gemastik12.pcapng**. Selanjutnya akan dilakukan enumerasi informasi terkait statistik paket data dengan bantuan **tshark**. Adapun hasilnya ialah sebagai berikut

```
$ tshark -r trafik-gemastik12.pcapng -q -z io,phs

===================================================================
Protocol Hierarchy Statistics
Filter:

frame                                    frames:1083 bytes:712553
  eth                                    frames:1083 bytes:712553
    ip                                   frames:1083 bytes:712553
      tcp                                frames:947 bytes:700667
        ftp                              frames:116 bytes:8951
          ftp.current-working-directory  frames:116 bytes:8951
        ftp-data                         frames:446 bytes:669338
          ftp-data.setup-frame           frames:446 bytes:669338
            ftp-data.setup-method        frames:446 bytes:669338
              ftp-data.command           frames:446 bytes:669338
                ftp-data.command-frame   frames:446 bytes:669338
                  ftp-data.current-working-directory frames:446 bytes:669338
                    data-text-lines      frames:3 bytes:953
      udp                                frames:136 bytes:11886
        dns                              frames:136 bytes:11886
===================================================================
```

Berdasarkan statistik di atas, dapat diketahui bahwa terdapat beberapa komunikasi yang berjalan pada **TCP** & **UDP** layer. Guna efisiensi penelusuran, akan dilakukan analisis pada **FTP streams** untuk mengetahui komunikasi antara client & FTP server. Adapun skema yang dilakukan ialah sebagai berikut

Identify TCP.stream & FTP-DATA

Sebelum mengekstraksi **FTP-DATA**, dilakukan penelusuran terhadap **tcp.stream** untuk mengetahui index dari **request** dan **response** dari eksekusi **FTP command**.

```
$ tshark -r trafik-gemastik12.pcapng -Y 'tcp.stream && ftp-data' -T fields -e tcp.stream| sort | uniq
14
19
26
9
```

Extract FTP-DATA

Dari sini, dapat dipahami bahwa **FTP requests** & **FTP response** masing-masing terdapat pada index **8,13,18,25** & **9,14,19,26**. Berdasarkan acuan tersebut, dilakukan proses ekstraksi sebagai berikut

```
$ for i in {8,13,18,25}; do echo "#TCP Stream $i\n REQ:\n\n" `tshark -r trafik-gemastik12.pcapng -z "follow,tcp,raw,$i" | awk 'f;/Node 1/,/=/' | grep -Ev '=|Node' | xxd -r -p`; done
#TCP Stream 8
 REQ:

 221 Goodbye.y send OK.ectory listing.,128,61,55).
#TCP Stream 13
 REQ:

 221 Goodbye.y send OK.ectory listing.,128,112,68).
#TCP Stream 18
 REQ:

 221 Goodbye.y send OK.ectory listing.,128,68,158).
#TCP Stream 25
 REQ:

 221 Goodbye. complete.e data connection for /home/gemastik12/files/file/data-gemastik.jpg (644463 bytes).
```

Terlihat bahwa terdapat proses **file-transfer** terhadap ImageFile **data-gemastik.jpg** yang berada pada tcp.streams ke-26. Langsung saja, dilakukan ekstraksi sebagai berikut:

```
$ tshark -r trafik-gemastik12.pcapng -z 'follow,tcp,raw,26' | awk 'f;/Node 1/,/=/' | grep -Ev '=|Node' | xxd -r -p > data.jpg

$ file data.jpg
data.jpg: JPEG image data, JFIF standard 1.01, resolution (DPI), density 300x300, segment length 16, Exif Standard: [TIFF image data, big-endian, direntries=4, manufacturer=Canon, model=Canon EOS 5D Mark III], baseline, precision 8, 510x340, frames 3
```

Sesaat setelah memperoleh ImageFile, dilakukan pengecekan terhadap *JPEG trailer* dengan bantuan **binwalk**. Adapun hasilnya, diperoleh *ZIP Archive* yang memuat tree.jpg, flag-diproteksi.pdf, dan data.png.

JPEG Trailer Exfiltration

```
$ binwalk data.jpg
DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             JPEG image data, JFIF standard 1.01
30            0x1E            TIFF image data, big-endian, offset of first image directory: 8
22337         0x5741          Zip archive data, at least v1.0 to extract, compressed size: 93051, uncompressed size: 93051, name: data.png
115426        0x1C2E2         Zip archive data, at least v2.0 to extract, compressed size: 206330, uncompressed size: 210266, name: flag-diproteksi.pdf
321805        0x4E90D         Zip archive data, at least v1.0 to extract, compressed size: 322317, uncompressed size: 322317, name: tree.jpg
644441        0x9D559         End of Zip archive, footer length: 22

$ foremost data.jpg
$ cd output/zip
$ 7z x 00000043.zip
```

Setelah beberapa saat, Kami berasumsi bahwa PDF-file dienkripsi dengan password yang bisa jadi diperoleh dari kedua ImageFile yang tersedia. Untuk itu, Kami berinisiatif untuk melakukan pengecekan dengan bantuan **Zsteg**.

```
$ zsteg data.png
meta Raw profile type APP1.. text: "\ngeneric profile\n202\n4578696600004d4d002a000000080004010e00020000001f0000003e0128000300000001\n000200000213000300000001000100008825000400000001000000 5e0000004e4f2043\n4f4445204558454355544 94f4e20414c4c4f5745442048455245 24500000004
```

```
000100020000\n00024e0000000020005000000030000009400030002000000024500
0000000400050000\n0003000000ac000000000000002b0000000100000036000000001
0001914000001d1b0000\n000900000010000003a0000000100004eb100001000\n"
b1,g,msb,xy        .. file: PGP\011Secret Key -
b1,rgb,lsb,xy                                                    ..          text:
"cGFzc3dvcmQgcGRmIGZpbGUgYWRhbGFoIGdhbGl0ZXJ1c3NhbXBhaWQ0cDR0a2
F3YW4="
b2,r,msb,xy        .. text: "@UUUUUUUUA"
b2,g,msb,xy        .. text: "TUUUUUUU"
b2,rgba,lsb,xy     .. text: "{k?g#s?{cs"
b2,abgr,msb,xy     .. text: "GCSSSSSSSSSSSSSS"
b3,rgba,lsb,xy     .. text: "?Wu?wwwWu?s"
b4,r,lsb,xy        .. text: "fUEEEU#2 "
b4,r,msb,xy        .. text: "DD\"\"33333333\"\"\"DD"
b4,g,lsb,xy        .. text: "3#\"3vwfgvffvvuB#"
b4,g,msb,xy        .. text: ";3;3;33w"
b4,b,lsb,xy        .. text: "gfwwgfgv"
b4,b,msb,xy        .. text: "ffff\"\"\"\"DTUU"
b4,rgb,lsb,xy      .. text: "c62b'2b'#c)3"
b4,bgr,lsb,xy      .. text: "3b6\"b7\"c'#c9"
b4,abgr,msb,xy     .. text: "h_`o(o(o"
```

Hasilnya diperoleh strings **base64-encoded text** pada LSB dari bit pixel RGB channel. Selanjutnya, Kami lalukan **base64-decode** sehingga diperoleh strings **password pdf file adalah galiterussampaid4p4tkawan**. Adapun diperoleh pesan dari PDF-file yang sekaligus memuat flag yang diminta

```
Ini budi sedang makan
gemastik12{a4e5e3b7ac929b5ec11726f7820cb1cd}
```

FLAG : gemastik12{a4e5e3b7ac929b5ec11726f7820cb1cd}

Steganography

digging deeper (200 pts)

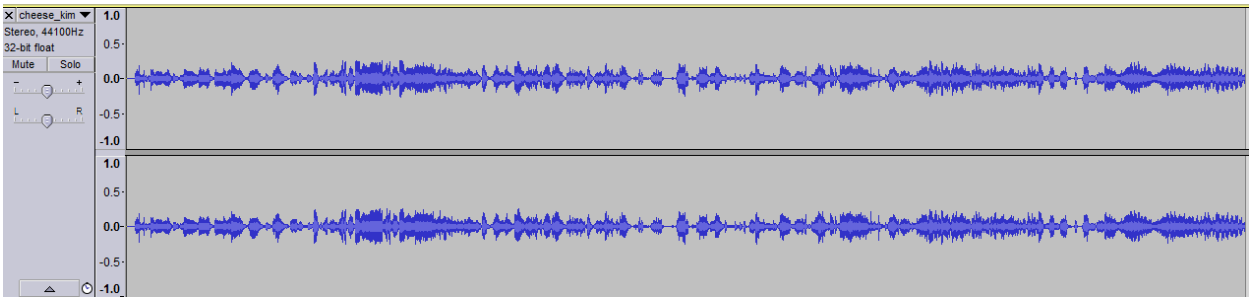
melodia menemukan sebuah file aneh, dia kesulitan untuk mencari informasi dalam file tersebut. bantu andi untuk menemukan informasi itu.

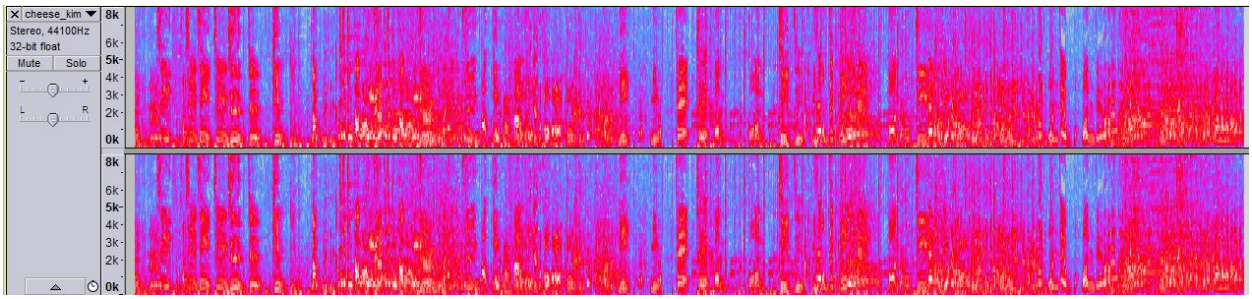
download file : http://180.250.135.20/minatozaki.bmp

Diberikan sebuah **binary file** bernama *minatozaki.bmp*. Selanjutnya akan dilakukan pengecekan terhadap validitas dari file. Setelah dilakukan pengecekan, diketahui bahwa binary file yang tersedia merupakan bzip archive, sehingga dilakukan proses ekstraksi sebagai berikut

```
$ file minatozaki.bmp
minatozaki.bmp: bzip2 compressed data, block size = 600k
$ 7z x -O1 minatozaki.bmp
$ file 1/minatozaki
minatozaki: POSIX tar archive
$ 7z x -O2 1/minatozaki

$ ls 2/
cheese_kimbap.wav
```

Berdasarkan hasil eksekusi tersebut, diperoleh AudioFile bernama **cheese_kimbap.wav**. Kami pun berinisiatif untuk melakukan observasi dengan bantuan *Audacity*.





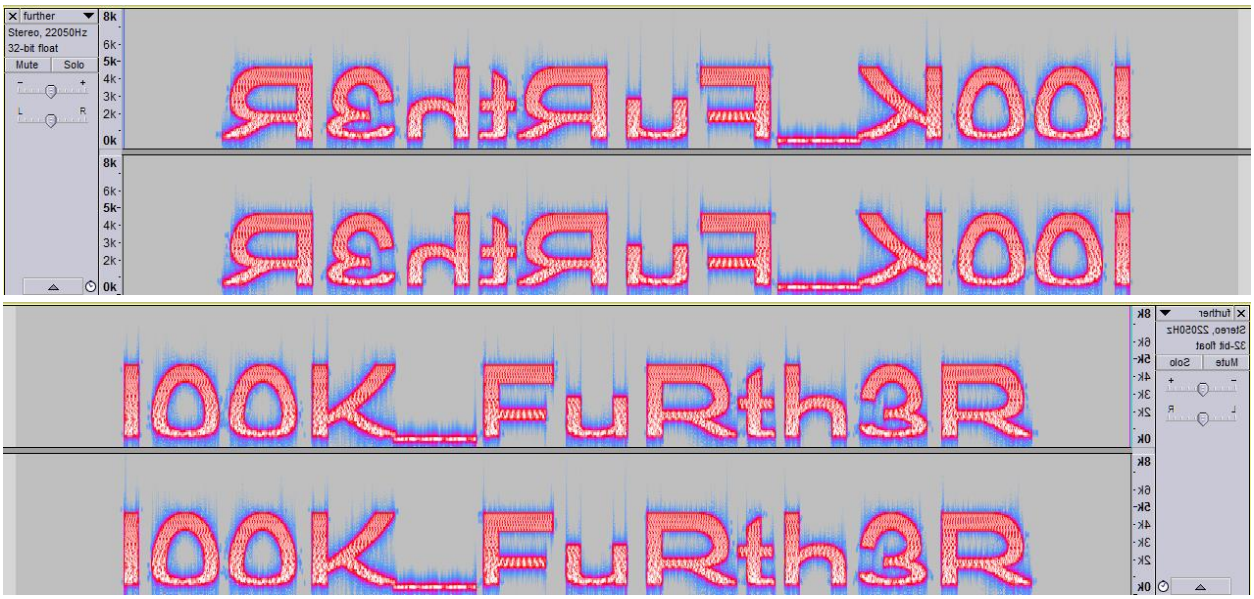
Namun, tidak diperoleh informasi yang relevan dengan flag. Setelah beberapa saat kemudian, muncul sebuah inisiasi untuk melakukan *reproduce* dari soal **what's flag?** yang merupakan soal penyisihan GEMASTIK XII lalu. Mengambil acuan tersebut, Kami juga mengambil **Indonesian Worldist** dari repository **Jim Geovedi** sebagai acuan dari proses *brute-force*. Adapun untuk menjalankan skema *brute-force*, Kami menggunakan 10 instance dari *GNU parallel* untuk menjalankan *steghide* dengan *passphrase* yang berbeda.

```
$ wget https://raw.githubusercontent.com/geovedi/indonesian-wordlist/master/00-indonesian-wordlist.lst
$ cat 00-indonesian-wordlist.lst | tr -d "'" | xargs -n 1 -I {} echo "echo {}; steghide extract -sf cheese_kimbap.wav -p {}" > commands.txt

$ parallel -j 10 < commands.txt &> result
```

Setelah beberapa saat, diperoleh passphrase yang valid beserta file *further.docx* hasil ektraksi steghide

```
$ grep –aB 1 wrote result
steghide: could not extract any data with that passphrase!
cowrote
--
patriot
wrote extracted data to "further.docx".

$ ls
cheese_kimbap.wav  further.docx
```

Kemudian, Kami lakukan dekompresi terhadap file sehingga diperoleh *further.wav* yang kemudian dianalisis kembali dengan bantuan *Audacity*. Hasilnya, diperoleh flag yang diminta dari hasil *spectrogram mode*



FLAG : gemastik12{l00K_FuRth3R}

Miscellaneous

Basic command

dalam direktori ini terdapat flags yang tersembunyi dalam suatu file. temukan flags tersebut.

Diberikan *gzip compressed data* bernama **dir.tar.gz**. Dari sini Kami lakukan proses ektraksi sedemikian hingga diperoleh binary file bernama "**find_me"** & **". "** yang dapat dirincikan sebagai berikut

```
$ 7z x -O1 dir.tar.gz
$ ls 1
dir.tar
$ 7z x -O2 1/dir.tar
$ ls -la  2/
$ ls -la 2/directory/
total 2
drwx------+ 1 shouko None  0 Oct 23 11:42  .
-rw----r--+ 1 shouko None 39 Oct 23 11:40 '. '
drwx------+ 1 shouko None  0 Oct 25 13:37  ..
-rw----r--+ 1 shouko None 37 Oct 23 11:42  find_me
```

Selanjutnya, Kami lakukan strings dump pada masing-masing file sehingga diperoleh flag yang diminta

```
$ strings 2/directory/find_me
temukan flags dalam direktori ini :)

$ strings 2/directory/.\
gemastik12{remember_your_unix_command}
```

FLAG : gemastik12{remember_your_unix_command}