Writeup IT RACE CTF lvl 3

Ditulis oleh petani\_hacker(s)



# Pwning The World #

Daftar Isi

[**Kategori Forensik**](#_6ag4er5h1y7j) **3**

[Lorem is not Ipsum (13 poin)](#_268ezhf7t5kk) 3

[Binary Typ0 (45 poin)](#_5fjncagtvgsc) 4

[**Kategori Web**](#_huku5drzcc2o) **6**

[AJAX XAJA (25 poin)](#_klfy0hh0nn71) 6

[Compare Us (40 poin)](#_3bs4k8974flx) 6

[Not Heart Bleed (43 poin)](#_tbidhqf08e2t) 8

[Square Them (60 poin)](#_2h2y99bok9uc) 10

[**Kategori Programming**](#_3la74nh2g9gu) **13**

[Pixel Racist (70 poin)](#_d2qse1s2z66b) 13

[Scazzy (88 poin)](#_m7malqyjgnos) 15

[**Kategori Reversing**](#_yx24t5fr6l4p) **19**

[Brute Self (35 poin)](#_2pfb1vtckv0v) 19

[module\_2 (90 poin)](#_ehtsfewgmyqh) 22

[**Kategori Recon**](#_q56svzltwivp) **29**

[Ping Me (50.1 poin)](#_d5f7ffy3dyci) 29

[Copy Pasta (100.2 poin)](#_9c1u2e3ql0zw) 29

[**Kategori Crypto**](#_wgz7w9mp75y9) **31**

[As Beautiful As Ruby (55 poin)](#_hms8lxnqi52c) 31

[Yarpchiever (78 poin)](#_12l8q7i61wxe) 34

[**Kategori Misc**](#_didz4tse2fo7) **39**

[Print The Flag (21 poin)](#_e0twc2wb0ha2) 39

# Kategori Forensik

#### **Lorem is not Ipsum (13 poin)**

**Soal:**

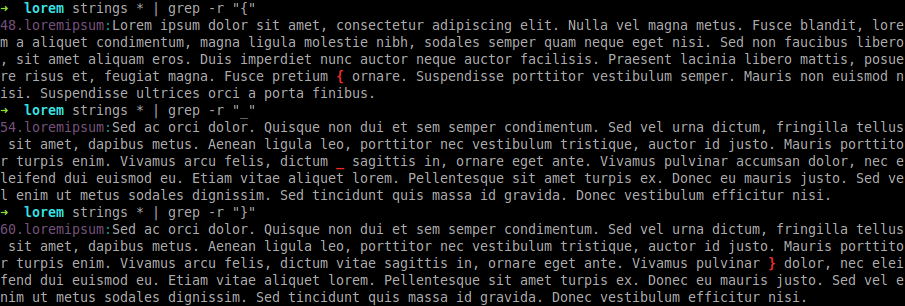
<http://task-00000100.itrace.systems/loremisnotipsum.tar.gz>

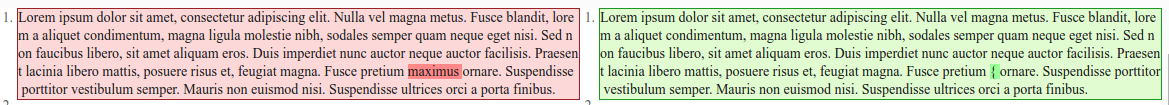
**Hint:**

-

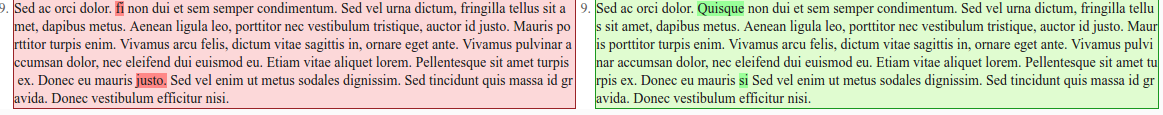
**Solusi:**

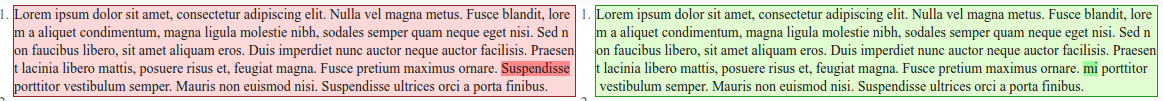
Mencari pattern yang digunakan untuk menulis flag dengan simbol “ { “, “\_”, “ } ”,



Dapat dilihat flag berada file lorem ipsum no 48 - 60. Lalu dicari perbedaan setiap file menggunakan website diffchecker.com, lalu bandingkan file no 1 dengan no 48-60. Contoh file no 1 dibandingkan dengan no 48, lihat di teks yang berwarna hijau ada simbol “{”

File no 1 dengan no 49, ada kata “si”

File no 1 dengan no 50, ada kata “mi”



Lalu lakukan sampai no 60, dan didapatkan flagnya,

Flag : ITRACE{similiriti\_similikiti}

#### **Binary Typ0 (45 poin)**

**Soal:**

<http://task-00000101.itrace.systems/raws.pcapng>

**Hint:**

-

**Solusi:**

Buka file raws.pcapng menggunakan wireshark, lalu follow TCP stream,

Pada bagian filter ubah menjadi “tcp.stream eq 6” lalu didapatkan source di bawah ini:

|  |
| --- |
| GET /t HTTP/1.1  Host: task-00000101.itrace.systems  User-Agent: curl/7.47.0  Accept: \*/\*  HTTP/1.1 200 OK  Date: Sun, 09 Oct 2016 14:02:14 GMT  Server: Apache  Upgrade: h2  Connection: Upgrade  Last-Modified: Sun, 09 Oct 2016 13:35:57 GMT  ETag: "5982d40-a11b-53e6eb9283528"  Accept-Ranges: bytes  Content-Length: 41243  Cache-Control: max-age=172800  Expires: Tue, 11 Oct 2016 14:02:14 GMT  Strict-Transport-Security: max-age=31536000  **ff d8 ff e0 0 10 fg hi jk lm** 0 1 1 1 0 48 0 48 0 0 ff fe 0 3a 46 6c 61 67 20 69 73 3a 20 49 54 52 41 43 45 7b 74 68 31 35 5f 69 35 5f 68 34 68 34 68 34 5f 6a 75 73 74 6b 69 64 64 69 6e 67 20 2e 20 6e 6f 74 20 74 68 69 73 20 3a 70 ff db 0 43 0 3 2 2 3 2 2 3 3 3 3 4 3 3 ....... |

Setelah dianalisis hex **ff d8 ff e0 0 10** merupakan header file gambar dengan format JPEG, dan **fg hi jk lm** merupakan hex yang typo, lalu ganti hex typo tersebut dengan byte yang seharusnya “4a 46 49 46” (JFIF) dan gabungkan semua source hex dari tcp.stream eq 6 sampai tcp.stream eq 13 didapatkan gambar di bawah ini:



Didapatkan flagnya

Flag : ITRACE{FR0M.0-F}

# Kategori Web

#### **AJAX XAJA (25 poin)**

**Soal:**

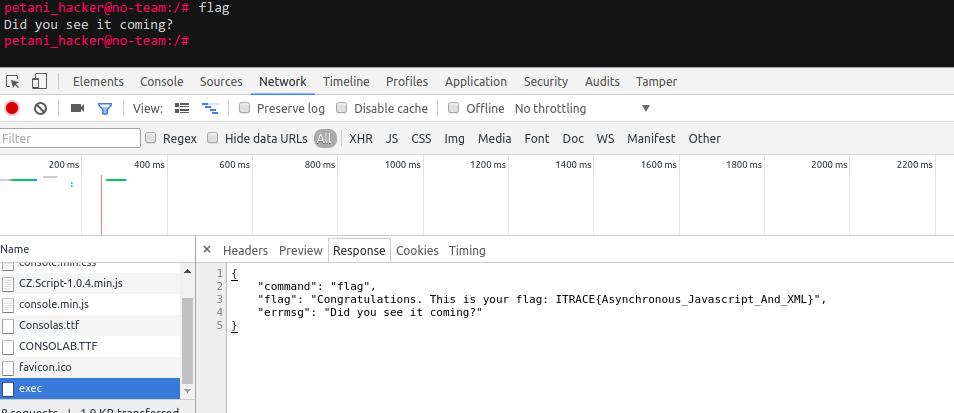
command: flag

**Hint:**

-

**Solusi:**

Ketik ‘flag’ di console platform CTF lalu lihat tab **network** di **developer tools**. Lihat respon dari request Ajax. Flag memang tidak ditampilkan di web, tapi data response Ajax-nya mengandung flag.



Dan didapatkan flagnya.

Flag: ITRACE{Asynchronous\_Javascript\_And\_XML}

#### **Compare Us (40 poin)**

**Soal:**

http://task-00000001.itrace.systems/compare-us.php

**Hint:**

-

**Solusi:**

|  |
| --- |
| <?php  error\_reporting(0);  include 'flag.php';  $check=thepassword();  parse\_str($\_SERVER['QUERY\_STRING']);  $A=$\_GET['key'];  if(ctype\_xdigit($A)){  $e=implode('',array\_map(function($i,$A){return chr(hexdec($A{$i+$i}.$A{$i+($i+1)}));},list($m,$n,$o)=range(0,2),array($A,$A,$A)));  if($e<1 && $e>0 && $e!==0){  if((int)(substr($A,strlen($e)\*2)+0) < -1){  if($check==$\_GET['password']){  echo flag();  } else {  echo 'Bad.';  }  } else {  echo 'Bad.';  }  } else {  echo 'Bad.';  }  } else {  echo 'Bad.';  }  echo "<pre>";  echo htmlentities(highlight\_string(file\_get\_contents(\_\_FILE\_\_)));  echo "</pre>"; |

Di web soal diberikan source codenya. Input parameter “key” yang akan jadi variabel $A harus memenuhi beberapa syarat. Syarat pertama ctype\_xdigit($A) yang artinya input harus merupakan digit hex (0-9 dan a-f). Syarat kedua, ada perhitungan variabel $e =

implode('',array\_map(function($i,$A){return chr(hexdec($A{$i+$i}.$A{$i+($i+1)}));},list($m,$n,$o)=range(0,2),array($A,$A,$A)));

Bisa dicek itu ngapain, dengan membuat script sederhana:

|  |
| --- |
| <?php  $A = "4142434445";  $e = implode('',array\_map(function($i,$A){return chr(hexdec($A{$i+$i}.$A{$i+($i+1)}));},list($m,$n,$o)=range(0,2),array($A,$A,$A)));  echo $e; |

Dan outputnya adalah “ABC”. Berarti dia melakukan decode dari hex namun hasilnya hanya sepanjang 3 karakter. Selanjutnya $e<1 && $e>0 && $e!==0, berarti input kalau di-decode harus berupa angka di antara 0 sampai 1, misalnya 0.5 (decode hexa = **302e35**).

Selanjutnya (int)(substr($A,strlen($e)\*2)+0) < -1, kita tahu strlen($e) pasti bernilai 3, sehingga itu sama saja (int)(substr($A,6)+0) < -1. Nilai $A sejauh ini adalah **302e35** (6 karakter), kalau di-substring 6 berarti dia mencari string lanjutannya, yang nanti dijumlahkan dengan nol (PHP otomatis mengubah tipenya jadi angka) dan di-cast jadi integer. Coba-coba:

* echo (int)(substr("302e3512345",6)+0); hasilnya 12345
* echo (int)(substr("302e3511111111111",6)+0); hasilnya 1111111111
* Sepertinya kita bisa overflow supaya melebihi nilai maksimum int sehingga jadi minus
* echo (int)(substr("302e351111111111111111111111111111111",6)+0); hasilnya -507921595474378752 ← sudah benar kurang dari -1, maka itulah nilai $A yang valid

Pengecekan terakhir $check==$\_GET['password'], sedangkan $check didapat dari $check=thepassword(); yang dirahasiakan passwordnya. Tapi tunggu, ada statement ini:

parse\_str($\_SERVER['QUERY\_STRING']);

Fungsi [parse\_str](http://php.net/manual/en/function.parse-str.php) ini termasuk vulnerability, karena kita bisa menimpa variabel apapun (mirip dengan fungsi [extract](http://php.net/manual/en/function.extract.php)). Jadi kita bisa masukkan parameter tambahan ke URL (check=aaa) untuk menimpa variabel $check, semua berkat parse\_str(). Jadi kita bisa menimpa isi variabel $check sesuai keinginan, sehingga waktu dibandingkan ($check==$\_GET['password']) akan selalu benar dan mencetak flag. Maka request-nya:

|  |
| --- |
| http://task-00000001.itrace.systems/compare-us.php?key=302e351111111111111111111111111111111&password=aaa&check=aaa |

Dapat flag: ITRACE{r1d1n9\_i5\_34513r\_th4n\_r34d1n9}

#### **Not Heart Bleed (43 poin)**

**Soal:**

<http://task-00001110.itrace.systems/request.php>

**Hint:**

-

**Solusi:**

#### 

Ketika dikunjungi laman soalnya terdapat hint yaitu

1. Method: Post
2. Param: request
3. Berhubungan dengan UTF-8

Sesuai nama soalnya, sepertinya behubungan dengan heart bleed. Target dicoba dengan mengirimkan request 10 karakter.

|  |
| --- |
| $ curl -s -X POST --data "request=`python -c "print 'A'\*10"`" 'http://task-00001110.itrace.systems/request.php'  <pre>{  "method": "POST",  "request": "AAAAAAAAAA",  "Accept-Charset": "UTF-8",  "Content-Length": 64,  "Respond-Text": "AAAAAAAAAA......................................................"  }% |

Respon dari server tetap 10 karakter. Akan dicoba lagi mengirim 64 karakter.

|  |
| --- |
| $ curl -s -X POST --data "request=`python -c "print 'A'\*64"`" 'http://task-00001110.itrace.systems/request.php'  <pre>{  "method": "POST",  "request": "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA",  "Accept-Charset": "UTF-8",  "Content-Length": 64,  "Respond-Text": "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"  }% |

Jika mengirim lebih dari 64 maka akan mengirim pesan error “Data is too long”

|  |
| --- |
| $ curl -s -X POST --data "request=`python -c "print 'A'\*65"`" 'http://task-00001110.itrace.systems/request.php'  Data is too long% |

Akhirnya dicoba mengirimkan karakter emoji 😁 ke server target.

|  |
| --- |
| $ curl -s -X POST --data "request=`python -c "print '😁'\*64"`" 'http://task-00001110.itrace.systems/request.php'  Data is too long% |

Hmm… data too long, artinya request yang dikirimkan lebih dari 64 byte. Setelah lihat list emoji dari konversinya ke UTF-8[[1]](#footnote-0) ternyata emoji karakter tersebut setara 4 byte. Sehingga, agar request yang dikirim pas 64 byte kita harus mengirim 64 / 4 yaitu 16 karakter emoji.

|  |
| --- |
| $ curl -s -X POST --data "request=`python -c "print '😁'\*16"`" 'http://task-00001110.itrace.systems/request.php'  <pre>{  "method": "POST",  "request": "\ud83d\ude01\ud83d\ude01\ud83d\ude01\ud83d\ude01\ud83d\ude01\ud83d\ude01\ud83d\ude01\ud83d\ude01\ud83d\ude01\ud83d\ude01\ud83d\ude01\ud83d\ude01\ud83d\ude01\ud83d\ude01\ud83d\ude01\ud83d\ude01",  "Accept-Charset": "UTF-8",  "Content-Length": 317,  "Respond-Text": "0x4c 0x4f 0x4f 0x4b 0x49 0x4e 0x47 0x20 0x46 0x4f 0x52 0x20 0x41 0x20 0x46 0x4c 0x41 0x47 0x3f 0x20 0x54 0x48 0x45 0x4e 0x20 0x54 0x48 0x49 0x53 0x20 0x49 0x53 0x20 0x46 0x4f 0x52 0x20 0x59 0x4f 0x55 0x3a 0x20 0x49 0x54 0x52 0x41 0x43 0x45 0x7b 0x59 0x34 0x59 0x5f 0x54 0x4f 0x5f 0x33 0x34 0x35 0x59 0x7d 0x0 0x0 0x0 "  }% |

Dan didapatkan sebuah respon kumpulan heksadesimal, agar proses decoding lebih mudah kami buat kode inline bash sederhana untuk menyelesaikan soal tersebut.

|  |
| --- |
| $ curl -s -X POST --data "request=`python -c "print '😁'\*16"`" 'http://task-00001110.itrace.systems/request.php' | grep -Eo "0x[a-f0-9]{2}" | xxd -p -r  LOOKING FOR A FLAG? THEN THIS IS FOR YOU: **ITRACE{Y4Y\_TO\_345Y}**% |

Flag: ITRACE{Y4Y\_TO\_345Y}

#### **Square Them (60 poin)**

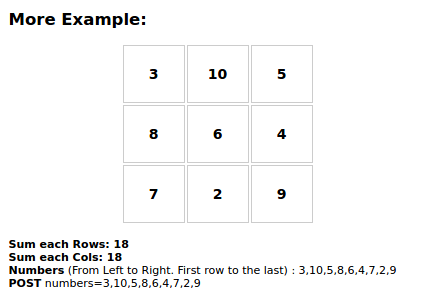
**Soal:**

http://task-00000010.itrace.systems/square.php

**Hint:**

-

**Solusi:**

Terdapat sebuah soal programming dimana kita diminta generate 9 angka yang jumlah kolom dan barisnya adalah N. Dimana N adalah nomer yang dikasih dari soal.

Dibuat sebuah solver-nya menggunakan library Z3 Constraint Solver[[2]](#footnote-1)

|  |
| --- |
| magicsq.py |
| #!/usr/bin/python  **from** z3 **import** \* **import** sys  **def** solve\_magic(sum):  c = sum / 3  a = Int('a')  b = Int('b')   s = Solver()  s.add(0 < a)  s.add(a < b)  s.add(b < c - a)  s.add(b != 2 \* a)   **if** s.check() == sat:  m = s.model()  a = int(str(m[a]))  b = int(str(m[b]))  square = [  c - b, c + (a + b), c - a,  c - (a - b), c, c + (a - b),  c + a, c - (a + b), c + b  ]  **assert** square[0] + square[1] + square[2] == sum  **assert** square[0] + square[3] + square[6] == sum  **assert** square[0] + square[4] + square[8] == sum  **return** square  **print** 'numbers='+','.join(str(x) **for** x **in** solve\_magic(int(sys.argv[1]))) |

Program tersebut menerima argumen sebuah angka yang diberikan dari soal, misal 18

|  |
| --- |
| $ python magicsq.py 18  numbers=3,10,5,8,6,4,7,2,9 |

Selanjutnya membuat script otomatis untuk submit ke soal.

|  |
| --- |
| square.sh |
| #!/bin/bash  SESSION='PHPSESSID=o9mqvjfv376h986in5edkn7u84' NUM=$(curl -s -X POST --cookie $SESSION 'http://task-00000010.itrace.systems/square.php' | tail -n 5 | head -n 1 | **grep** -Eo '[0-9]' | tr -d '\n') DATA=$(python magicsq**.**py $NUM)  **while** true; **do**  RSP=$(curl -s -X POST --data $DATA --cookie $SESSION 'http://task-00000010.itrace.systems/square.php')  SOLVD=$(**echo** $RSP | **grep** -Eo 'ITRACE{.+}')  **echo** $RSP  **if** [[ $SOLVD ]]; **then**   **break**  **fi**  NUM=$(**echo** $RSP | **grep** -Eo '[0-9]' | tr -d '\n')  DATA=$(python magicsq**.**py $NUM) **done** |

Lalu jalankan

|  |
| --- |
| $ ./square.sh  {"errmsg":"Good job","nextsum":261}  {"errmsg":"Good job","nextsum":222}  {"errmsg":"Good job","nextsum":273}  {"errmsg":"Good job","nextsum":291}  {"errmsg":"Good job","nextsum":222}  {"errmsg":"Good job","nextsum":288}  {"errmsg":"Good job","nextsum":210}  {"errmsg":"Good job","nextsum":246}  {"errmsg":"Good job","nextsum":351}  {"errmsg":"Good job","nextsum":252}  Congratulation. This is your Flag: **ITRACE{m4g1c\_squ4r3\_is\_s0\_m4th}** |

Flag: ITRACE{m4g1c\_squ4r3\_is\_s0\_m4th}

# Kategori Programming

#### **Pixel Racist (70 poin)**

**Soal:**

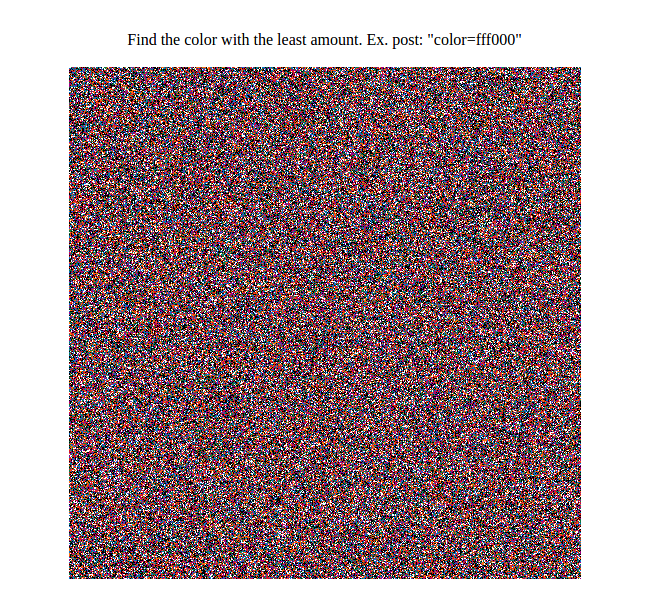
<http://task-00001001.itrace.systems/racist.php>

**Hint:**

-

**Solusi:**

Diberikan sebuah soal programming dimana kita harus mencari sebuah warna yang kemunculannya paling sedikit.



Submit dengan mengirim Post Request dengan parameter “color=(jawaban)”.

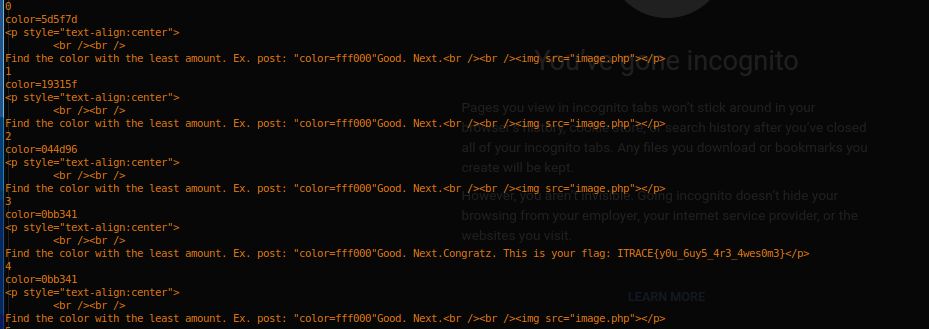
Buatlah sebuah solvernya dengan python

|  |
| --- |
| ppc.py |
| #!/usr/bin/python  **import** collections **from** PIL **import** Image  counter = collections.defaultdict(int) im = Image.open("target.png") pix = im.load() w, h = im.size tupl = []  **for** r **in** range(w):  **for** c **in** range(h):  tupl.append("".join([hex(x)[2:4].zfill(2) **for** x **in** pix[r,c]]))  **for** word **in** tupl:  counter[word] += 1  **print** "color="+sorted(counter.iteritems(), key=**lambda** x: x[::-1])[0][0] |

Buat sebuah script auto-submit dengan bash, sebelumnya ambil cookie terlebih dahulu dari browser.

|  |
| --- |
| ppc.sh |
| #!/bin/bash  i=0 SESSION="PHPSESSID=qedntnb92fll6g3k1ko17855a3"  **while** true; **do**  # counter **echo** $i; i=$(($i+1))  # download image curl 'http://task-00001001.itrace.systems/image.php' --cookie $SESSION -s > target**.**png  # get least amount color DATA=$(python ppc**.**py)  **echo** $DATA  curl -s --cookie $SESSION -X POST --data $DATA 'http://task-00001001.itrace.systems/racist.php'    **done** |

Didapatkan flagnya pada iterasi ke-3



Flag: ITRACE{y0u\_6uy5\_4r3\_4wes0m3}

#### 

#### **Scazzy (88 poin)**

**Soal:**

Scazzy, bot berjenis kelamin perempuan. Suka menggoda. Tapi kalo dirayu jual mahal. <https://web.telegram.org/#/im?p=@itrace>

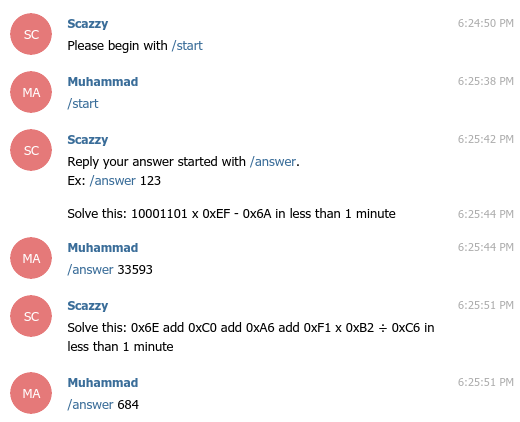
**Hint:**

-

**Solusi:**

Sewaktu kita buka web Telegram itu, ada bot “Scazzy” yang menyapa channel @itrace. Kita bisa PM ke bot Scazzy tersebut, dan dia akan menjelaskan instruksinya. Jadi via PM bukan public channel.

Kita bisa membalas dengan “/start”, dan dia akan memberi pertanyaan hitung-hitungan. Kita harus menjawab dengan “/answer <jawaban>”.



Kita diharuskan menjawab pertanyaannya dalam waktu kurang dari semenit (di tengah acara diupdate jadi kurang dari 20 detik). Jadi sepertinya kita harus bikin program buat mengotomatisasinya.

Butuh instal:

* Telegram CLI <https://github.com/vysheng/tg>
* Telegram Python client <https://github.com/luckydonald/pytg>

Instal dan baca dokumentasinya.

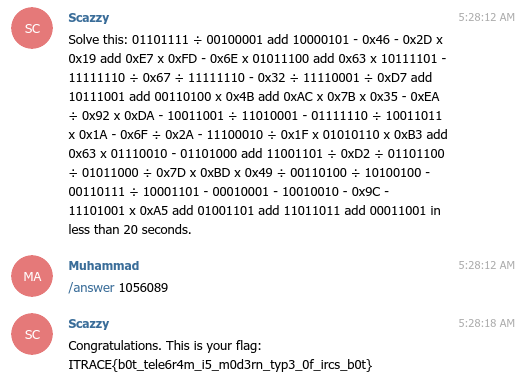
Buat fungsi untuk solve soal dari Scazzy:

|  |
| --- |
| *# -\*- coding: utf-8 -\*-*  **import** **re**  **def** solve(text):  text = re.search(r'Solve this: (.\*) in less than', text).group(1)  text = text.replace(u' x ', u' \* ') *# ubah x jadi \**  text = text.replace(u' - ', u' - ') *# ubah - jadi -*  text = text.replace(u' add ', u' + ') *# ubah add jadi +*  text = text.replace(u' ÷ ', u' / ') *# ubah ÷ jadi /*  text = re.sub(r'([01]{8})', r'0b\1', text) *# cari biner 8 digit & tambahkan 0b didepan*  **return** eval(text) *# let python calculate it unsafely* |

Sekaligus script untuk secara otomatis menerima dan mengirim pesan ke bot. Script ini dimodifikasi dari tutorial/dokumentasi library **pytg**:

|  |
| --- |
| **from** **pytg** **import** Telegram **from** **pytg.utils** **import** coroutine  **from** **pytg.sender** **import** Sender **from** **pytg.receiver** **import** Receiver receiver = Receiver(host="localhost", port=4458) sender = Sender(host="localhost", port=4458)  sender.send\_msg("Scazzy", u'/start')  @coroutine *#* **def** main\_loop():  **while** 1:  msg = (**yield**) *# it waits until it got a message, stored now in msg.*  **if** msg[u'event'] == u'message':  text = msg[u'text']  **if** 'Solve this' **in** text:  answer = solve(text)  **print** "Question =", text  **print** "Answer =", answer  sender.send\_msg("Scazzy", u'/answer ' + str(answer))  *#else:*  *# print text*  receiver.start() receiver.message(main\_loop()) |

Di akhir acara didapatkan flagnya (percakapan muncul otomatis di web Telegram waktu script berjalan):



Flag: ITRACE{b0t\_tele6r4m\_i5\_m0d3rn\_typ3\_0f\_ircs\_b0t}

# Kategori Reversing

#### **Brute Self (35 poin)**

**Soal:**

http://task-00000011.itrace.systems/password.tar.gz

**Hint:**

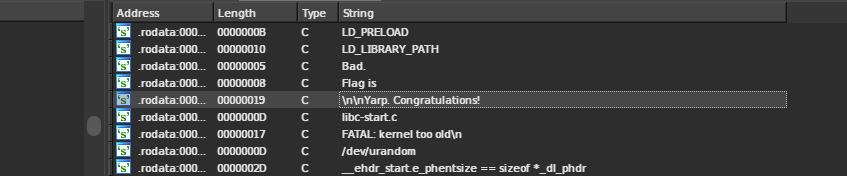
-

**Solusi:**

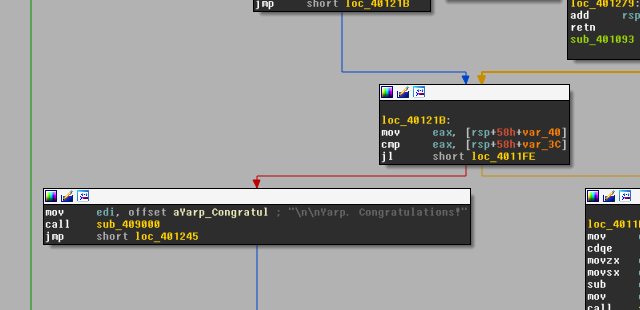
Pertama-tama dicek terlebih dahulu format file soal dengan command ‘file’.

|  |
| --- |
| $ file password  password: ELF **64-bit** LSB executable, x86-64, version 1 (SYSV), statically linked, for GNU/Linux 2.6.24, BuildID[sha1]=636e9ba2b787af74e02a1060fa0e599d0cd8844f, **stripped** |

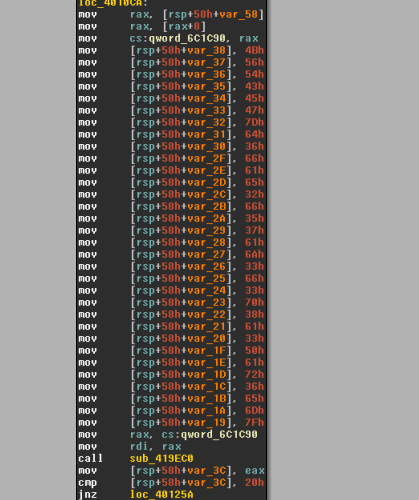
Analisis string dengan menggunakan IDA Pro 64 bit .



Kunjungi alamat yang memanggil string tersebut. Lalu lihat flow graph-nya

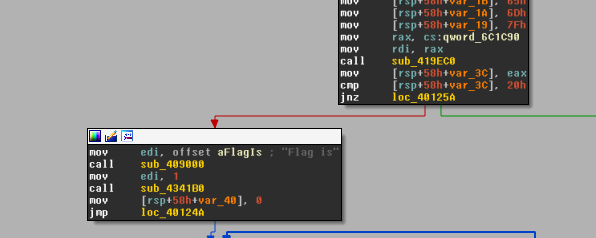


Setelah mundur-mundur terdapat sebuah array mencurigakan.



Berikut array-nya

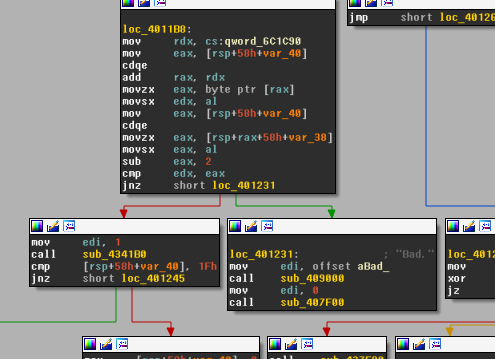
|  |
| --- |
| 0x4B 0x56 0x54 0x43 0x45 0x47 0x7D 0x64 0x36 0x66 0x61 0x65 0x32 0x66 0x35 0x37 0x61 0x6A 0x33 0x66 0x33 0x70 0x38 0x61 0x33 0x50 0x61 0x72 0x36 0x65 0x6D 0x7F |



Perhatikan instruksi ‘**cmp [rsp+58h+var\_3C], 20h**’ sepertinya dia melakukan perbandingan suatu variable dengan 20h (32) sepertinya variable tersebut adalah jumlah input.



Ternyata program melakukan print ‘Flag is’ namun masih salah. Lanjut ke instruksi lainnya.



var\_38 (array ‘mencurigakan’ tadi) disimpan di eax, lalu di operasikan ‘**sub eax 2**’ dan dibandingkan dengan **edx** jika salah maka masuk ke kondisi ‘Bad’. Tanpa berpikir panjang kami langsung buat inline script Python untuk melakukan pengurangan nilai 2 dari array tadi.

|  |
| --- |
| >>> "".join([chr(int(x, 16)-2) **for** x **in** "0x4B 0x56 0x54 0x43 0x45 0x47 0x7D 0x64 0x36 0x66 0x61 0x65 0x32 0x66 0x35 0x37 0x61 0x6A 0x33 0x66 0x33 0x70 0x38 0x61 0x33 0x50 0x61 0x72 0x36 0x65 0x6D 0x7F".split()])  **'ITRACE{b4d\_c0d35\_h1d1n6\_1N\_p4ck}'** |

Flag: ITRACE{b4d\_c0d35\_h1d1n6\_1N\_p4ck}

#### **module\_2 (90 poin)**

**Soal:**

Unlock [module\_2](http://task-00001010.itrace.systems/module_2.apk) | [module\_2\_dependencies\_optional.tar.gz](http://task-00001100.itrace.systems/module_2_dependencies_optional.tar.gz)

**Hint:**

-

**Solusi:**

Decompile APK yang dikasih dengan apktool atau yang online[[3]](#footnote-2), cari main activity, ternyata app dibuat dengan Cordova, berarti bentuknya halaman web.

|  |
| --- |
| package com.cafelabs.itrace;  import android.os.Bundle;  import org.apache.cordova.CordovaActivity;  public class MainActivity extends CordovaActivity {  public void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  loadUrl(this.launchUrl);  }  } |

Di folder assets/www, ini halaman utama aplikasinya (index.html), ini script di halaman itu:

|  |
| --- |
| **<script** type="text/javascript" charset="utf-8"**>**  register\_local();  document.addEventListener('deviceready',  **function**(){  register\_gkey();  getservertime();  **if**(localStorage.isstart=="zero"){  toast("didn't received registration\_id from Google yet");  } **else** {    }  FCMPlugin.onNotification(  **function**(data){  **if**(data.wasTapped){  alert(data.body);  } **else** {  **var** c={id:genrandom(),html:data.title+  "<br />"+data.body}  appendto(c);  alert("Flag received");  }  },  **function**(msg){  //register\_gkey();  },  **function**(err){  alert('Error registering onNotification callback: ' + err);  }  );  },**false**  ); **</script>** |

Pertama dia register\_local, mengeset localstorage “gkey” dan “isstart”.

|  |
| --- |
| **function** register\_local(){  **if**(!localStorage.gkey) localStorage.setItem("gkey","no");  **if**(!localStorage.isstart) localStorage.setItem("isstart","zero"); } |

Kemudian register\_gkey, sepertinya retrieve token FCM (Firebase Cloud Messaging[[4]](#footnote-3)) dan mengeset localstorage “gkey” dari token yang didapat:

|  |
| --- |
| **function** register\_gkey(){  FCMPlugin.getToken(  **function**(token){  **if**(!localStorage.firstUse){  localStorage.setItem("firstUse",1);  }  **if**(token=="null" || token==""){  **return** register\_gkey();  } **else** {  localStorage.gkey=token;  }  FCMPlugin.subscribeToTopic("contestant");  },  **function**(err){  alert('error retrieving token: ' + err);  }  ); } |

Kemudian getservertime, dia request ke \_app\_host http://cafelinux.info/sideprojects/itrace-lvl-3/time?gkey=GKEY&ver=APPVER, kemudian mengeset localstorage “cookie” dari cookie yang didapat, lalu memanggil syncapp(0)

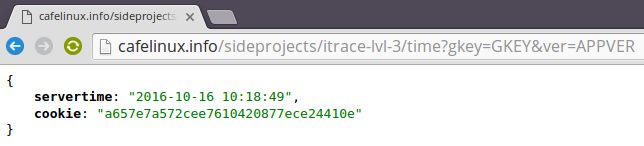
|  |
| --- |
| **function** getservertime(){  **var** a={id:"a",html:"Initializing..."};  appendto(a);  **var** obj={target:\_app\_host+"time?gkey="+localStorage.gkey+"&ver="+\_app\_version};  CZAjax(obj,  **function**(x){  **var** o=JSON.parse(x);  **var** a={id:0,html:"Server time is: <b>"+o.servertime+"</b><br />"};  appendto(a);  localStorage.setItem("cookie",o.cookie);  syncapp(0);  }  ); } |

Di syncapp, dia request ke http://cafelinux.info/sideprojects/itrace-lvl-3/sync?gkey=GKEY&ver=APPVER&cookie=COOKIE. COOKIE adalah cookie yang didapat dari getservertime tadi. Setelah dapat dia update cookie di localstorage (localStorage.cookie=o.cookie). Kalau “module” yang didapat dari request itu == 2 maka stop.

|  |
| --- |
| **function** syncapp(s){  **if**(s===0){\_r\_module=[];}  **var** a={id:genrandom(),html:"module\_"+s+" sending SYN at "+now()};  appendto(a);  **var** b={id:genrandom(),html:"module\_"+s+" waiting response..."};  appendto(b);  **var** ctime=**new** Date().getTime();  **var** obj={  target:\_app\_host+"sync?gkey="+localStorage.gkey+"&ver="+\_app\_version+"&cookie="+localStorage.cookie  };  CZAjax(obj,  **function**(x){  **var** o=JSON.parse(x);  **var** c={id:genrandom(),html:o.errmsg}  appendto(c);  localStorage.cookie=o.cookie;  **if**(o.module==2){  clog('stahp');  } **else** {  **if**(!pushtomodule(o.module)){  **var** a={id:genrandom(),html:"<span style='color:red'>session module\_"+(o.module+1)+" is invalid.</span><br />exit."};  appendto(a);  } **else** {  **var** a={id:genrandom(),html:"<span style='color:green'>module\_"+o.module+" synced.</span>"};  appendto(a);  syncapp(o.module+1);  }  }  }  );   } |

Mungkin bisa kita simulasikan request ke /time dan /sync itu lewat browser, dengan “gkey” ngasal saja siapa tahu berhasil. Kalau pakai browser kita nggak dapat gkey karena gkey di-push ke perangkat mobile.

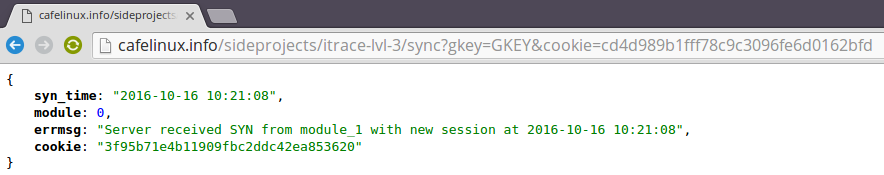
Pertama request ke /time



Dapat cookie, pakai buat request ke /sync seperti di alur programnya



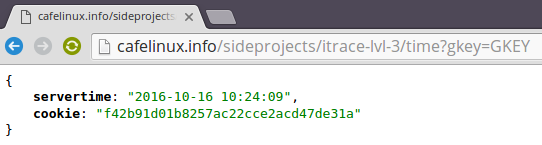
Dapat module 0 dan cookie baru, kita request lagi ke /sync pakai cookie baru



Kok tetap module 0? Harusnya module: 1. Kalau kita ulang terus tetap dapat module 0.

Kayaknya ada yang salah. Mungkin cookie-nya harusnya jangan diganti baru? Jadi pakai cookie yang didapat pertama kali saja.

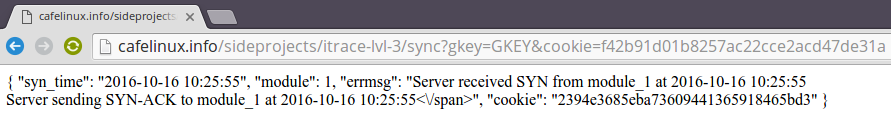
Kita ulang lagi dari awal, request ke /time



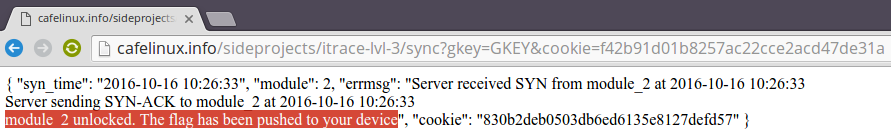
Dapat cookie, pakai buat request ke /sync



Kita request lagi, tapi kali ini **jangan diganti cookie-nya**. Berarti cukup refresh.



Nah sukses dapat module: 1. Refresh lagi untuk dapat module: 2.



Dapat module 2, dan flag has been pushed to your device. Tapi karena kita jalanin di browser ya tidak dapat push notificationnya. Berarti sampai sini kita punya dua pilihan:

1. Intercept request dari app ke server pakai Burp untuk mengedit request supaya cookie yang dikirim tetap
2. Edit aplikasinya (kode JS) supaya tidak merubah cookie setiap kali request ke /sync

Kita coba yang kedua, ngedit APK yang dikasih. File APK kan sebenernya file ZIP, bisa di-extract. Dan karena kodenya JS, bisa langsung diubah. Yang diubah di fungsi syncapp, baris yang mengubah cookie (localStorage.cookie=o.cookie) dihapus supaya waktu request ke /sync selanjutnya, cookienya tetap.

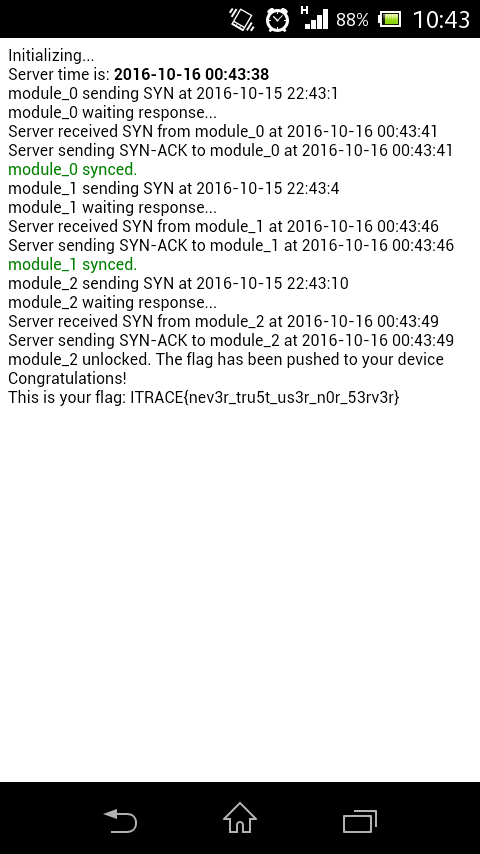
Setelah diubah, lalu signing. Ingat file APK kalau diubah, supaya bisa jalan harus di-sign. Untungnya di soal dikasih informasi signing-nya ([module\_2\_dependencies\_optional.tar.gz](http://task-00001100.itrace.systems/module_2_dependencies_optional.tar.gz)).

|  |
| --- |
| Itrace.config:  -- if needed for re-signing --  Pass:1tr4c3lvl3  alias:itrace-key |

Kita sign APK yang sudah diubah, dia minta password yang juga sudah diberikan di file itrace.config:

|  |
| --- |
| $ jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore itrace.keystore module\_2.apk itrace-key  Enter Passphrase for keystore: 1tr4c3lvl3 |

Selesai, kita install APK yang sudah diubah ke device, dan jalankan. Sekarang karena kita pakai device beneran, flag di-push ke device dan ditampilkan.



Flag: ITRACE{nev3r\_tru5t\_us3r\_n0r\_53rv3r}

# Kategori Recon

#### **Ping Me (50.1 poin)**

**Soal:**

Ping Me. Literally.

**Hint:**

are mine

**Solusi:**

Searching di google dengan keyword ‘Ping Me Cafelinux’, karena organizer CTF ini adalah Cafelinux.info. Kami menemukan <http://cafelinux.info/pingme>



Flag: ITRACE{http\_www.cafelinux.info\_ping.me}

#### **Copy Pasta (100.2 poin)**

**Soal:**

-

**Hint:**

-actually-heartbeat-opposite

-Do you know pastebin? because i know.

WARNING: Please use TOR Browser while diving in Deep Web

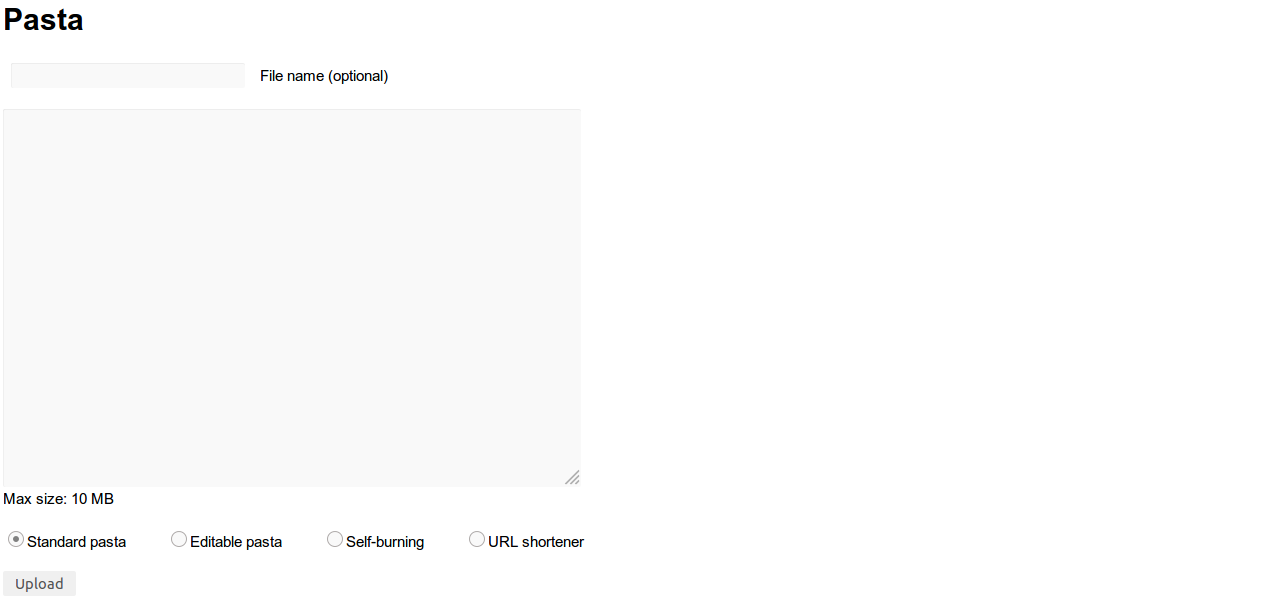
**Solusi:**

Pertama sekali Download TOR Browser <https://www.torproject.org/download/download>

Lalu search dengan search engine TOR di web <https://ahmia.fi/> dengan keyword “copy pasta”

Lihat pada index pertama ada sebuah link website seperti *pastebin* yang pada dasarnya bisa menyimpan string yaitu <http://pastagdsp33j7aoq.onion/>

Setelah terbuka , tidak ada form search untuk mencari keyword seperti “ITRACE” atau “heartbeat”



Lalu dengan skill **GUESSING** dan dibantu dengan Hint soal, kami ubah URL jadi seperti ini <https://pastagdsp33j7aoq.onion.to/actually-heartbeat-opposite>

Uwuw Finally We Got That Flag

|  |
| --- |
| https://pastagdsp33j7aoq.onion.to/Congratulations. This is your flag: ITRACE{n0t\_s0\_d33p\_but\_still\_d0nt\_g0\_d33per} |

Flag: ITRACE{n0t\_s0\_d33p\_but\_still\_d0nt\_g0\_d33per}

# Kategori Crypto

#### **As Beautiful As Ruby (55 poin)**

**Soal:**

Encryptor: <http://task-00010000.itrace.systems/beauty.rb>

Encrypted File: [flag](http://task-00010000.itrace.systems/flag)

**Hint:**

-

**Solusi:**

Diberi script Ruby yang kalau “dirapikan” jadi begini:

|  |
| --- |
| flag = "Find This Flag" z = "" i = [] $t = 0x00  1.times **do** *#joget*  2.times.map{i+=[Random.rand(0xAA..0xFF)]}   3.times **do**   z.concat(  ($t==0x01) ?   flag.unpack('B\*').map{|e|e}.join.split("").**rotate**(i[0x01]).join   :  flag.unpack('b\*').map{|e|e}.join.split("").**shuffle**.join  );  $t = $t+0x01;  **end**  **end**  $t=0x01; z.concat(  4.times.map{|f| (f==$t)? z.split("").join : z.split("").shuffle.join}.join )   p z |

Jadi string flagnya di-unpack jadi string binary (tergantung suatu variabel $t, jika $t = 1, **B\*** little endian, selainnya **b\*** big endian). Tiga kali, dan disambungkan dengan string z. Jika $t bernilai 1, string binary di-rotate sebanyak angka random, selainnya string binary diacak (shuffle). Kita harus mencari hasil rotate ini, karena hanya itu yang dapat dikembalikan (kalau yang hasil shuffle tidak bisa dikembalikan karena bit-bitnya diacak).

Ilustrasinya, misal string kita “ABCD”, binary-nya[[5]](#footnote-4): 01000001010000100100001101000100. Kalau di-rotate ke kiri 5 kali menjadi 00101000010010000110100010001000. Kita perlu menebak berapa kali string asli di-rotate. Bisa dicari kalau kita tahu string awalnya.

Jadi sampai sini panjang string z = 3 x (panjang flag dalam biner). Selanjutnya ada z.concat(4.times…), artinya string z disambung dengan string yang panjangnya 4 kali lipat dirinya. Jadi di akhir, panjang string binernya adalah 15x panjang flag dalam biner. Penjelasannya:

z = 3 \* flag (3.times do z.concat...)

z = z + 4 \* z (z.concat(4.times.map...)

Jadi len(z) = 3 \* len(flag) + 4 \* (3 \* len(flag)) = 15 \* len(flag)

Flag yang dienkrip:

|  |
| --- |
| 001100000001011001101101000001000110000010111001001101001100001001101001100111101110101100000100010010110010001000001000011110010110111111000011011100110111101001011101110101100100000010110101111101100110001011101100111001110000100110111011101101001010111001111100110111110000111010001001110010011101110101011011000010011101011101011111010100100101010100010100100100000101000011010001010111101101110010011101010010100000111000001010010111100100111010001100010011010100111010011100110011000000111010001110000011001100110100011110110111010101110100011010010110011001111101010110011110010100011111001111111010010100000111011000000010011100000001111101001110011010000101110011111101111000100000110010100011001110110100100101011111000010011000101111100010100001101100111110011111011000101010101111001010110000110110001110110101011000011100001110101010001000100001000110000001100100000101101001111010100001000001101010001000101111010111001000101011001101111010011101100011011011011110001111000011000101010000100000111100111000111011001000000011100100101010101101110011010000010000011100111011011111101110010110010001001011110011111011100000010110101111001001101010001111010011111111101000000011100000100001111110100011011111100011100110011101001011101111000111110110011000111100010000110010011110000001111010111100111011100111000101100011100101011001100011011110110010001101000011000000100110011111100001010110000111101100000110010001111010010010111011101110000000101111110010000001000010011000100011010001111111000110011011001110101011011010101111001110111001001100111010001001000100100101010101101111000000101010001000000111101001101100000101000101101001111011100100011001111011010101101000100100000000000101011101101010110111111101001100000001011001101101000001000110000010111001001101001100001001101001100111101110101100000100010010110010001000001000011110010110111111000011011100110111101001011101110101100100000010110101111101100110001011101100111001110000100110111011101101001010111001111100110111110000111010001001110010011101110101011011000010011101011101011111010100100101010100010100100100000101000011010001010111101101110010011101010010100000111000001010010111100100111010001100010011010100111010011100110011000000111010001110000011001100110100011110110111010101110100011010010110011001111101010110011110010100011111001111111010010100000111011000000010011100000001111101001110011010000101110011111101111000100000110010100011001110110100100101011111000010011000101111100010100001101100111110011111011000101010101111001010110000110110001110110101011000011100001110101010001000100001000110010001110100100000101101000011001001101101000101101001101000110101001001100000001000101110101010010001110101111111111000001000000001010010100011010111001100110101101111110100110110001000110001010100111011011000101100101000010010011010100101100000000101001001011100000010110101100111111100001111101001111101011100101011111111001000110011101001000101001000110011111011000011010011101110111111000001010000101100000001001111011001111111100101011110010101011001000011111011001000001100101111111110100101110010101010011111011001111010000110000000001111001000111010011111101101011001001110001101011001100010101111100001101100111110111011100011110011110010100010001000010100010100111010011100001010101000110000010001100010010000110011010011011101011110011110011011110110010111110101010010100110000001100001011001010100111100011010101111100101011100100110010010001111010000000011001001110110110010000001000100010001000001010000110010111000100011101011101101101100101011110011101110001010111111000000001010010100000000001001000100010100111010110011111011010100010101001101011000111011100111011010101001101111110111000101000000110111011110001100101101101010101111011000101100111100100100111000001111000100111110111110001010110100010011111000100101101011001011011110011100100000101110011100101111010000101011110001101000000110010110011100100100011111101101111111000011101011100111000101011000100011111001110000110111111101111001111110101010011101000100000111110110101101010010111101011111000101010100100100100000101001011000101011010101110110101010011100101111001110100001001000100111000011000110001010001011111111000010001011000000010110000101010010011010101000101000101111010111000001101111110001110000001000110010100100011011111000010010 |

Dipecah jadi 15 elemen sesuai perhitungan kita tadi.

|  |  |
| --- | --- |
| Index ($t) | Keterangan |
| 0 | Flag biner di-shuffle |
| 1 | Flag biner di-rotate |
| 2 | Flag biner di-shuffle |
| sisanya | Z yang diulang 4 kali |

Untuk mendapatkan flag yang di-rotate, ambil pecahan yang kedua (index 1):

|  |
| --- |
| 110010011101110101011011000010011101011101011111010100100101010100010100100100000101000011010001010111101101110010011101010010100000111000001010010111100100111010001100010011010100111010011100110011000000111010001110000011001100110100011110110111010101110100011010010110011001111101010110 |

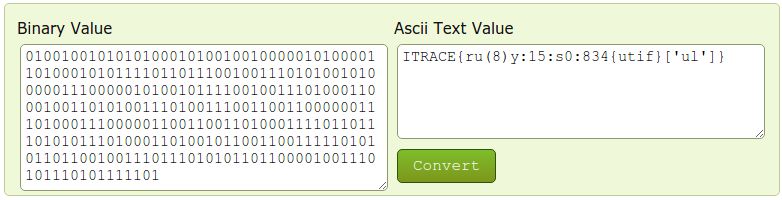
Asumsi string awal “ITRACE” (010010010101010001010010010000010100001101000101), ketemu di rangkaian binary itu:

|  |
| --- |
| 11001001110111010101101100001001110101110101111101**010010010101010001010010010000010100001101000101**0111101101110010011101010010100000111000001010010111100100111010001100010011010100111010011100110011000000111010001110000011001100110100011110110111010101110100011010010110011001111101010110 |

Karena itu hasil rotate, ambil bit-bit sebelumnya dan taruh di belakang, jadinya seperti ini:

|  |
| --- |
| **010010010101010001010010010000010100001101000101**0111101101110010011101010010100000111000001010010111100100111010001100010011010100111010011100110011000000111010001110000011001100110100011110110111010101110100011010010110011001111101010110**11001001110111010101101100001001110101110101111101** |

Decode[[6]](#footnote-5):



Flag: ITRACE{ru(8)y:15:s0:834{utif}['ul']}

#### 

#### **Yarpchiever (78 poin)**

**Soal:**

Yarpchiever, an Encryption File Method Using Binary Rotation and Compress it Using gzip at Once

Adalah makalah pertama yang pernah saya kirim di event CFP IDSECCON pada tahun 2015. Dan ditolak. :p

Karena metode Yarpchiever ini memang memiliki kelemahan, yakni ditulis menggunakan bahasa pemrograman yang tidak di compile. Sehingga jika user ingin menggunakan aplikasi ini, sudah pasti harus memiliki source codenya. Dan sudah pasti juga, sebuah enkripsi jika open source, maka data yang diencrypt sudah jelas tidak aman.

So, tugas kalian adalah mendapatkan flag didalam file yang sudah terenkripsi menggunakan Yarpchiever yang telah kehilangan "Token Key" nya dengan memanfaatkan kelemahan Yarpchiever.

Yaitu, Open Source. :)

Paper: [Yarpchiever.pdf](http://task-00001101.itrace.systems/Yarpchiever.pdf)

Source Code: [yarpchiever.tar.gz](http://task-00001101.itrace.systems/yarpchiever.tar.gz)

Encrypted File: [flag.txt.zit](http://task-00001101.itrace.systems/flag.txt.zit)

**Hint:**

-

**Solusi:**

Fungsi enkripsi:

|  |
| --- |
| **function** encrypt($filename,$contents){  **echo** "Encrypting...";  $hex=bin2hex($contents);  $salt=generateSalt($hex);  $rot=fakemd5\_decode($salt);  $rotated=yarpRotate($hex,$rot);  **echo** "**\t**done**\n**";  **echo** "Compressing done.**\n**";  **if**($this->param\_key\_index('-w')>0){  $fp=fopen(".token","w");  fwrite($fp,$salt);  fclose($fp);  **echo** "Token key saved to .token**\n**";  } **else** {  **echo** "Token key : ".$salt."**\n**";  **echo** $salt;  }  $this->param\_o($filename,$rotated);  } |

Pertama isi dari file akan dijadikan hexadecimal, kemudian dari situ dibuat “salt”, kemudian dari salt dicari nilai rotasinya, kemudian dilakukan “rotasi biner” seperti yang dijelaskan pada paper yang menyertai soal ini. Setelah dirotasi kemudian dilakukan kompresi gzip ($this->param\_o($filename,$rotated)) hasil finalnya.

Fungsi-fungsi penting lain:

|  |
| --- |
| **function** generateSalt($str){  **return** fakemd5\_encode(rand(1,1000),**true**); } **function** fakemd5\_encode($str,$bool=**false**){  $len = strlen($str);  $hex = **array**(1=>1,2,3,4,5,6,7,8,9,"a","b","c","d","e","f");  **if**($len>15){  **return** **false**;  } **else** {  $hash = **NULL**;  $rand = generateChars(32,$bool);  $obj = strrev($str);  $used = $len \* 2;  $cover = substr($rand,$used,31-$used);  **for**($i=0;$i<$len;$i++){  $hash .= substr($rand,$i \* 2 ,1).substr($obj,$i,1);  }  **return** $hash.$cover.$hex[$len];  } } **function** generateChars($limit,$specialchars=**false**,$type='all'){  $hexa=**NULL**;  $chars = ($specialchars===**false**)?**null**:"^-+";  $chrup='ABCDEFGHIJKLMNOPQRSTUVWXYZ';  $chrlow='abcdefghijklmnopqrstuvqxyz';  $chrnum='0123456789';  **switch**($type){  **case** 'up' : $chr=$chrup.$chars; **break**;  **case** 'low' : $chr=$chrlow.$chars; **break**;  **case** 'num' : $chr=$chrnum.$chars; **break**;  **case** 'uplow' : $chr=$chrup.$chrlow.$chars; **break**;  **case** 'upnum' : $chr=$chrup.$chrnum.$chars; **break**;  **case** 'lownum' : $chr=$chrlow.$chrnum.$chars; **break**;  **default** : $chr=$chrup.$chrlow.$chrnum.$chars; **break**;  }  **for**($i=1;$i<$limit+1;$i++){  $rIdx = rand(1,strlen($chr));  $hexa .=substr($chr,$rIdx,1);  }  **return** $hexa; } **function** fakemd5\_decode($str){  $hex = **array**(1=>1,2,3,4,5,6,7,8,9,"a","b","c","d","e","f");  $len = array\_search(substr($str,-1,1),$hex);  $hash = **NULL**;  **for**($i=0;$i<$len;$i++){  $hash .= substr($str,$i \* 2+1,1);  }  **return** strrev($hash); } **function** yarpRotate($bin,$n,$rot=1){  $n=$n\*$rot;  $output=**null**;  **for**($i=0;$i<strlen($bin);$i++){  $x=$i+$n;  **if**($x>strlen($bin)-1) $x=$x-strlen($bin);  $output .= substr($bin,$x,1);  }  **return** $output; } |

Analisis:

* generateSalt membuat string acak dari suatu angka random 1 sampai 1000 (rand(1,1000))
* String acak ini yang menjadi “token” (via fakemd5\_encode)
* Dari string token bisa didapatkan lagi angka acak yang tadi (via fakemd5\_decode)
* Angka acak itu digunakan untuk merotasi data biner dari isi file sehingga berubah

Bruteforce terhadap string tokennya impossible karena panjang dan kemungkinan karakternya sangat banyak. Tapi lihat bahwa jumlah rotasi dihitung dari token ini. Dan karena rentang angka acaknya ini hanya 1 sampai 1000, maka sangat bisa di-bruteforce jumlah rotasinya untuk mendapatkan data aslinya. Lihat rutin untuk dekripsinya:

|  |
| --- |
| **echo** "Reading file...**\n**";  **echo** "Validating token...";  $token=$this->is\_token();  **if**($token){  **echo** "**\t**done**\n**";  **echo** "Extracting file...";  $f=file\_get\_contents($filename);  $rest=substr($f,-4);  $unpack=unpack("V",$rest);  $fpsize=end($unpack);  $fp = gzopen($filename, "rb");  $fread = gzread($fp, $fpsize);  gzclose($fp);  $hex=bin2hex($fread);  $rotated=yarpRotate($hex,$token,-1);  $bin=hex2bin($rotated);  $ungzip=gzuncompress($bin);  $header=explode("|",$ungzip,3);  **if**($header[0]!=="yarp"){  **echo** "**\n**False token**\n**";  } **else** {  $this->param\_p($header[1],$header[2]);  }  } **else** {  **echo** "**\n**Token not valid**\n**";  } |

Nilai $token bisa kita bruteforce, rentangnya hanya 1-1000. Buat script singkat modifikasi dari fungsi dekripsi:

|  |
| --- |
| error\_reporting(0);  $filename = "flag.txt.zit";  **echo** "Extracting file..."; $f=file\_get\_contents($filename); $rest=substr($f,-4); $unpack=unpack("V",$rest); $fpsize=end($unpack); $fp = gzopen($filename, "rb"); $fread = gzread($fp, $fpsize); gzclose($fp); $hex=bin2hex($fread);  **for** ($token = 1; $token < 1000; $token++) {  $rotated=yarpRotate($hex,$token,-1);  $bin=hex2bin($rotated);  $ungzip=gzuncompress($bin);  **if** ($ungzip) {  $header=explode("|",$ungzip,3);  **if**($header[0]!=="yarp"){  **echo** "**\n**False token**\n**";  } **else** {  print\_r($header);  }  } } |

Secara singkat dapat hasilnya karena rentang bruteforce-nya pendek.

|  |
| --- |
| Extracting file...Array  (  [0] => yarp  [1] => flag.txt  [2] =>  ./DISCLAIMER  Yarpchiever is an Encryption File Method Using Binary Rotation and Compress it Using gzip at Once  i write this code to securing my files. But actually it's ridiculus  if you made an encryption using un-compile-able programming like PHP.  Now you see this code as open source, and i hope there is someone would  continue this encryption project using another programming language.  I'm just an ordinary programmer from Ternate, with low-level skills.  I'm just an Android Geek wannabe. Maybe sometimes :)  Btw, congratulations.  This is your flag: ITRACE{3ncrypt\_345y\_8ut\_d3crypt\_15nt}  ) |

Flag: ITRACE{3ncrypt\_345y\_8ut\_d3crypt\_15nt}

# Kategori Misc

#### **Print The Flag (21 poin)**

**Soal:**

Download [PrintTheFlag.class](http://task-00001111.itrace.systems/PrintTheFlag.class) and reupload to<http://45.64.99.71:5555/upload.php>

**Hint:**

-

**Solusi:**

Diberikan sebuah file dengan nama [PrintTheFlag.class](http://task-00001111.itrace.systems/PrintTheFlag.class), soal memberi tahu agar upload kembali file tersebut ke alamat yang telah disediakan. Namun setelah dicoba upload ulang menghasilkan pesan File do not match.



Setelah dilihat kode pada halaman upload ternyata konten dari file di hash kemudian dicocokkan dengan hash yang telah ditentukan, kita dapat menyimpulkan bahwa file yang baru saja diupload kontennya telah diubah / rusak.

Kita coba decompile file class dengan tools online, hasilnya diketahui bahwa signature pada file yang diberikan salah atau bukan signature file .class yang seharusnya (CA FE BA BE)

.

Kita coba perbaiki dengan EmEditor dengan melihat signature file .class lain yang masih baik. Coba upload kembali dan voila didapatkan flagnya



Didapatkan flagnya dari web:

Flag: ITRACE{s0m3t1m35\_j4v4\_is\_s0\_t3xty}

1. http://apps.timwhitlock.info/emoji/tables/unicode [↑](#footnote-ref-0)
2. <https://github.com/Z3Prover/z3> [↑](#footnote-ref-1)
3. <http://www.javadecompilers.com/apk> [↑](#footnote-ref-2)
4. <https://firebase.google.com/docs/cloud-messaging/> [↑](#footnote-ref-3)
5. <http://www.binaryhexconverter.com/ascii-text-to-binary-converter> [↑](#footnote-ref-4)
6. <http://www.binaryhexconverter.com/binary-to-ascii-text-converter> [↑](#footnote-ref-5)