WRITEUP CTF

GEMASTIK 13

by

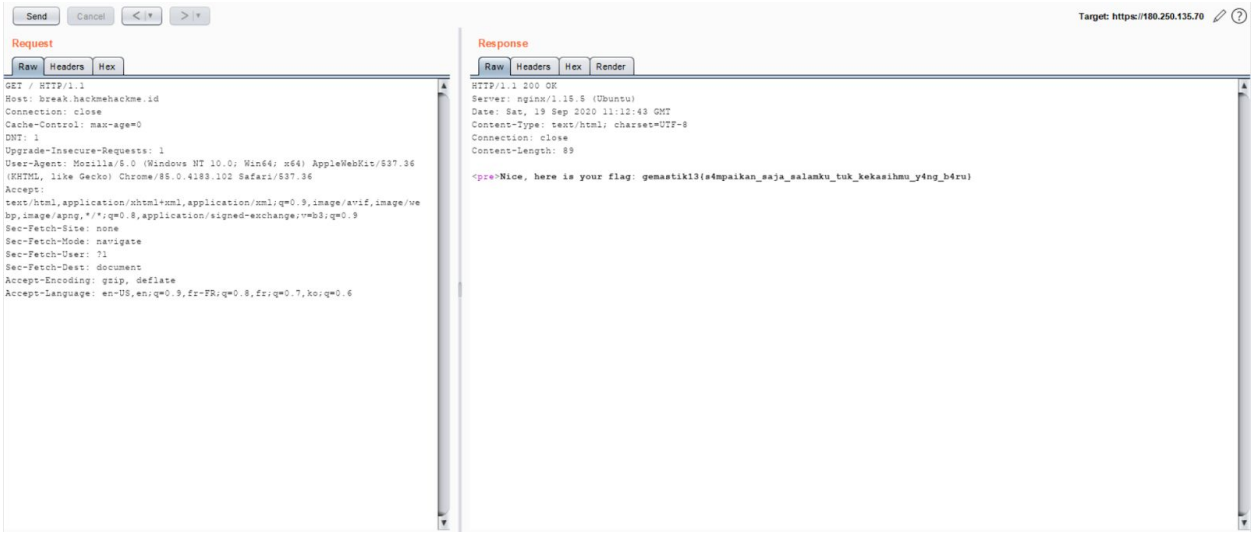
-BigBrainBois-

Institut Teknologi Bandung

Web

My information is leaking

Diberikan sebuah website dengan sertifikat yang mencurigakan, yang di-issue oleh break.hackmehackme.id. Kami coba menaruh sub domain tersebut pada header Host menggunakan Burps. Hasilnya adalah sebagai berikut:



Flag: gemastik13{s4mpaikan_saja_salamku_tuk_kekasihmu_y4ng_b4ru}

Please Hack Me

Diberikan sebuah website yang menampilkan kode php. Dari kode tersebut ada 2 hal yang unik. Pertama, kita harus membuat data[“hmac”] == data[“admin_hash”] sehingga diprint flagnya, namun data[“admin_hash”] diset dengan nilai yang tidak kita ketahui. Kedua, kita bisa melakukan object injection melalui unserialize. Dengan demikian, kita harus menggunakan unserialize untuk membuat data[“hmac”] senilai dengan data[“admin_hash”]. Payloadnya adalah sebagai berikut:

```
a:2:{s:10:"admin_hash";N;s:4:"hmac";R:2;}
```

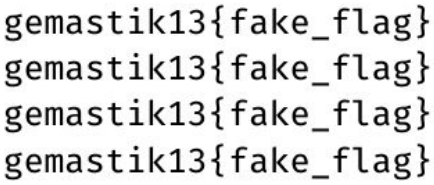
Encode menjadi base64 dan diperolehlah flagnya.

Flag: gemastik13{B4ckr3ferenc3}

Forensics

Congratulations

Diberikan sebuah image png yang rusak. Jika dianalisis dengan pngcheck, mengeluarkan hasil kerusakan setelah chunk IEND. Setelah dilihat, terdapat 2 chunk IDAT berlebih di akhir. Kami menduga bahwa kita harus menyisipkan chunk IDAT tersebut. Setelah dicoba-coba secara manual, kedua chunk tersebut diletakkan sebagai chunk ketiga dan keempat. Gambar flagnya adalah sebagai berikut:



gemastik13{1d4t_s1z3_4lw4ys_s4m3_3xc3pt_l45t_0N3}

Flag: gemastik13{1d4t_s1z3_4lw4ys_s4m3_3xc3pt_l45t_0N3}

Steganography

Missing Something ?

Diberikan sebuah gambar. Pada awalnya kami mencoba berbagai hal, tetapi tidak ada yang berhasil. Setelah bertapa beberapa lama, kami menyadari bahwa gambar tersebut bisa jadi merupakan sebuah stereogram.

Kami pun langsung membuka stegsolve dan gunakan stereogram solver. Setelah mencoba-coba offset, didapat suatu tulisan flag.

Berikut adalah gambar *full*-nya.

Flag: gemastik13{P4nd3m1C_m4k3s_m3_stay_@_H0m4}

AH↓HA↑HA↑HA↑HA↑

Diberikan sebuah file suara yang jika dilihat channel kiri dan kanan berbeda jauh. Extract bagian kanan saja, lalu dicoba dimasukkan ke qsstv. Mengikuti guide dari https://ourcodeworld.com/articles/read/956/how-to-convert-decode-a-slow-scan-television-transmissions-sstv-audio-file-to-images-using-qsstv-in-ubuntu-18-04, dengan file audio yang sudah dimodifikasi didapatkan gambar dengan flag.

Flag: gemastik13{yougotme_peko}

Cryptography

:(

Reversing

REVERSE WORLD

Diberikan sebuah file binary. Saat dibuka menggunakan IDA terdapat proses enkripsi dan dibandingkan dengan suatu value. Panjang string input harus 31.

```
v5 = strlen(s);
for ( i = 0LL; i < v5; ++i )
  s[i] = get_tbl_entry((unsigned int)s[i]);
if ( v5 == 31 )
{
  if ( !strncmp(s, s2, 0x20uLL) )
  {
    puts("YESSS You Change The World");
    result = 0;
  }
  else
  {
    puts("You Not Change The World");
    result = 1;
  }
```

Dengan dynamic analysis menggunakan gdb, diketahui bahwa mapping dari karakter input terhadap enkripsi 1 to 1 sehingga bisa dimapping saja. Digunakan break pada bagian strncmp dan diexamine memory untuk input yang telah dienkripsi. Lakukan step ini beberapa kali untuk berbagai input yang bervariasi sehingga didapatkan full mappingnnya.

Berikut script yang digunakan untuk mapping.

```
import textwrap

mapping = {}

s = textwrap.wrap('803a3c3b9a39f8b9', 2)[::-1]
s += textwrap.wrap('969238dfafcad67a', 2)[::-1]
s += textwrap.wrap('9e713dfeed620a4f', 2)[::-1]
s += textwrap.wrap('b05c6e7bcfc83f', 2)[::-1]

s = [int(x, 16) for x in s]

inp = 'abcdefghijklmnopqrstuvwxyz01234'

for i, c in enumerate(inp):
```

```
      mapping[s[i]] = c

s = textwrap.wrap('1ed51b7f4aa5232b', 2)[::-1]
s += textwrap.wrap('048d60eca1335dba', 2)[::-1]
s += textwrap.wrap('ef41f22d981aeabf', 2)[::-1]
s += textwrap.wrap('66a07cdec15915', 2)[::-1]

s = [int(x, 16) for x in s]

inp = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ56789'

for i, c in enumerate(inp):
      mapping[s[i]] = c

s = textwrap.wrap('fca9ccf406b7aa99', 2)[::-1]
s += textwrap.wrap('40a8e8451c50086a', 2)[::-1]
s += textwrap.wrap('2cf6b877e943ad19', 2)[::-1]
s += textwrap.wrap('e1422a485b1d95', 2)[::-1]

s = [int(x, 16) for x in s]

inp = """!"#$%&'()*+,-./:;<=>?@[]^_`{|}~"""

for i, c in enumerate(inp):
      mapping[s[i]] = c
```

Setelah didapatkan mappingnya, selanjutnya lihat memory yang dibandingkan, dan reverse terhadap mapping yang sebelumnya dibuat.

Berikut scriptnya:

```
s = textwrap.wrap('ca7aed62b9df3b3a', 2)[::-1]
s += textwrap.wrap('3a38b980a5485c7b', 2)[::-1]
s += textwrap.wrap('1d5c80981d3a387a', 2)[::-1]
s += textwrap.wrap('429aaf6e7b9241', 2)[::-1]
s = [int(x, 16) for x in s]

fl = ''
for x in s:
      fl += mapping[x]

print fl
```

Didapatlah flag berikut.

Flag: gemastik13{Changing_Th3_Wo12ld}

Mr. Simple

Diberikan sebuah file binary. Saat dibuka menggunakan IDA didapat bahwa terdapat proses enkripsi berupa AES CBC. Key dan IV-nya juga dapat dilihat.

```
    | puts("input must be <= 32 char");
      result = 1;
    }
    else
    {
      base64_decode((__int64)&v12, 64, (__int64)"InilahKuncinyaUntukEnkripsiPesan");
      AES_init_ctx((__int64)&v11, (__int64)&v12);
      base64_decode((__int64)&v12, 64, (__int64)"DanIniAdalahIVuntukEnkripsiPesan");
      AES_ctx_set_iv((__int64)&v11, (const __m128i *)&v12);
      v7 = 8LL;
      v8 = &v10;
      while ( v7 )
      {
        *(_DWORD *)v8 = 0;
        v8 = (__int64 *)((char *)v8 + 4);
        --v7;
      }
      strcpy((char *)&v10, v3);
      printf("INP: '%s'\n", &v10);
      AES_CBC_encrypt_buffer((__int64)&v11, (signed __int64)&v10, 0x20u);
      printf("INP: %s\n", &v10);
      base64_encode((__int64)&v12, 64, (__int64)&v10, 32);
      printf("Buffer: %s\n", &v12);
      result = 0;
    }
  }
  return result;
```

Langsung saja, kita buat script untuk melakukan proses dekripsi. IV dan key dipotong menjadi sepanjang ukuran bloknya, yaitu 16.

```
from Crypto.Cipher import AES
import base64


key = base64.b64decode(b'InilahKuncinyaUntukEnkripsiPesan')[:16]
iv = base64.b64decode(b'DanIniAdalahIVuntukEnkripsiPesan')[:16]
ct = base64.b64decode(b'zOHmai4ZLj2j50vYcWZhGdftB9ICmGlNOiKtjKID+Cc=')

if __name__ == '__main__':
    cipher = AES.new(key, mode=AES.MODE_CBC, IV=iv)
    print(cipher.decrypt(ct))
```

Didapatlah flag berikut.

Flag: gemastik13{AeS-Sederhana}

Gemastik Premium

Diberikan sebuah file apk, kita bisa menggunakan tools online untuk mendecompilenya, dilihat terdapat main file di MainActivity.java, setelah dibaca, file tersebut ternyata mengenkripsikan input lalu di ada constraint yang harus dipenuhi.

Berikut fungsi enkripsi:

```
public final String encrypt(String keyfactor) {
    Intrinsics.checkParameterIsNotNull(keyfactor, "keyfactor");
    StringBuilder fixed = new StringBuilder();
    int length = keyfactor.length();
    for (int x = 0; x < length; x++) {
        if (x == 0) {
            fixed.append((char) (keyfactor.charAt(x) ^ 30));
        } else if (x % 2 == 0) {
            fixed.append((char) ((keyfactor.charAt(x) + 30) ^ 16));
        } else if ((x & 3) == 3) {
            fixed.append((char) ((keyfactor.charAt(x) & 18) ^ 12));
        } else if (x % 3 == 0) {
            fixed.append((char) ((keyfactor.charAt(x) ^ 18) ^ 10));
        } else if (x / 4 == 0) {
            fixed.append((char) ((keyfactor.charAt(x) & 2) ^ 12));
        } else {
            fixed.append((char) (keyfactor.charAt(x) ^ 'Z'));
        }
    }
    String sb = fixed.toString();
    Intrinsics.checkExpressionValueIsNotNull(sb, "fixed.toString()");
    return sb;
}
```

Dan fungsi constraint validasi.

```
public final boolean validation(String keyfactor) {
    Intrinsics.checkParameterIsNotNull(keyfactor, "keyfactor");
    if (keyfactor.length() == 25 && keyfactor.charAt(0) == (keyfactor.length()
        return true;
    }
    return false;
}
```

Kita dapat mereverse fungsi validasi sehingga tahu setelah dienkripsi input harus seperti apa. Berikut script yang digunakan:

```
keyfactor = [0 for i in range(25)]

keyfactor[0] = (25 * 4) - 11
keyfactor[1] = 25 / 2
keyfactor[2] = (25 * 5) - 2
keyfactor[3] = 25 / 2
keyfactor[4] = (25 * 4) - 3
keyfactor[5] = (25 / 2) + 2
keyfactor[6] = (25 * 4) + 19
```

```
keyfactor[7] = (25 / 2) + 2
keyfactor[8] = (25 * 4) - 5
keyfactor[9] = (25 * 2) - 7
keyfactor[10] = (25 * 5) + 12
keyfactor[11] = 25 / 2
keyfactor[12] = (25 * 2) + 15
keyfactor[13] = (25 * 2) + 11
keyfactor[14] = (25 * 6) - 1
keyfactor[15] = 25 + 3
keyfactor[16] = (25 * 5) + 5
keyfactor[17] = 25 / 5
keyfactor[18] = (25 * 6) - 6
keyfactor[19] = 25 + 3
keyfactor[20] = (25 * 6) + 3
keyfactor[21] = (25 * 2) - 6
keyfactor[22] = (25 * 6) + 6
keyfactor[23] = 25 + 5
keyfactor[24] = (25 * 5) + 14
```

Setelah mengetahui keyfactor, kita dapat mengetahui juga input awal dengan mereverse bagian encrypt. Berikut fungsi yang digunakan:

```
r = ''

for i, n in enumerate(keyfactor):
      if i == 0:
            a = n^30
      elif i%2 == 0:
            a = (n^16)-30
      elif i&3 == 3:
            a = (n^12)&18
            a = ord('X')
            print n
      elif i%3 == 0:
            a = (n^10)^18
      elif i//4 == 0:
            # a = (n^12)&2
            a = ord('E')
      else:
            a = n^ord('Z')

      r += chr(a%256)
```

Pada bagian ord('E') dan ord('X') sebenarnya fungsi tidak bisa direverse secara langsung karena menggunakan &, namun pembatasan karakter bisa dilakukan. Flag yang didapat sampai step ini adalah G?M?STIK13{?3gg?t_b?k4n?}.

Berikut script yang digunakan untuk membangkitkan constraintnya:

```
import string

s = [12, 14, 12, 28, 28, 30]

def genpos(a):
      pos = []
      for i in range(256):
            if (i&18)^12 == a and (chr(i) in string.printable):
                  pos.append(chr(i))

      return pos

for el in s:
      print str(el) + ' = ' + ', '.join(genpos(el))
```

Berikut output dari program tersebut yang membatasi karakter:

```
12 = !, $, %, (, ), ,, -, @, A, D, E, H, I, L, M, `, a, d, e, h, i, l, m
14 = ", #, &, ', *, +, ., /, B, C, F, G, J, K, N, O, b, c, f, g, j, k, n, o
12 = !, $, %, (, ), ,, -, @, A, D, E, H, I, L, M, `, a, d, e, h, i, l, m
28 = 0, 1, 4, 5, 8, 9, <, =, P, Q, T, U, X, Y, \, ], p, q, t, u, x, y, |, }
28 = 0, 1, 4, 5, 8, 9, <, =, P, Q, T, U, X, Y, \, ], p, q, t, u, x, y, |, }
30 = 2, 3, 6, 7, :, ;, >, ?, R, S, V, W, Z, [, ^, _, r, s, v, w, z, {, ~
```

Dari pembatasan karakter, penebakan beberapa huruf, dan dari format flag didapatkanlah flagnya.

Flag: GEMASTIK13{l3gg1t_buk4n?}

Pwn

Repeat After Me

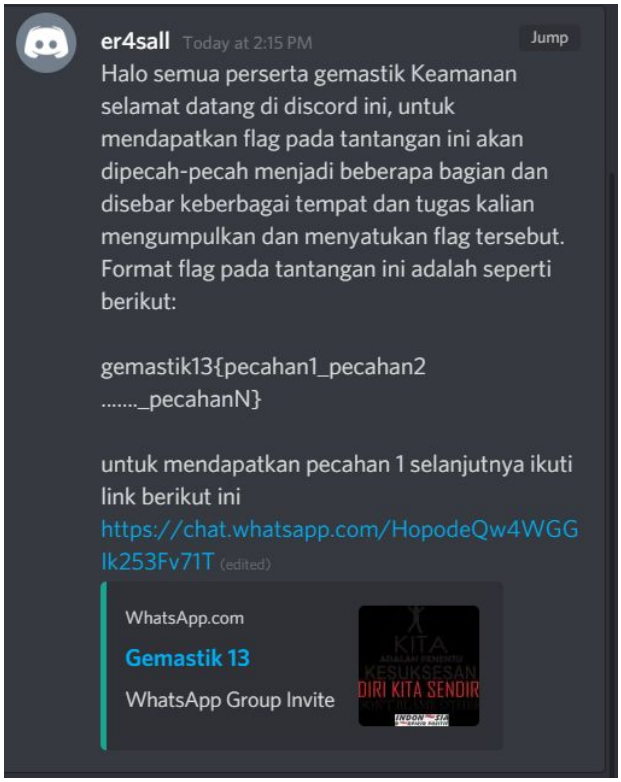
Diberikan koneksi ke 180.250.135.6:9090, jika dibuffer overflow didapatkan flagnya. Input menggunakan ‘A’ yang banyak.

Flag: lupa flagnya apa servisnya udah mati gabisa ngecek lagi :(

Mix

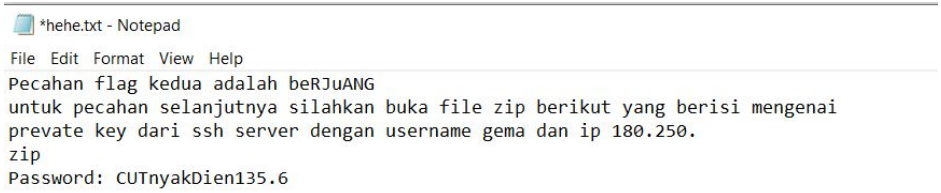
Around The World

Dari deskripsi soal, kami mendapat link server discord. Didapat format flag.



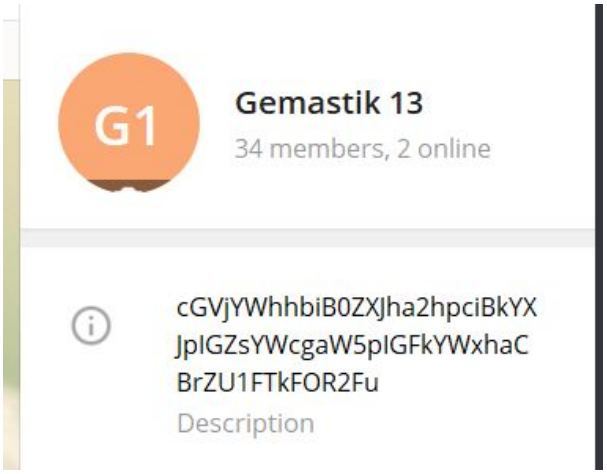
Kemudian, kami masuk ke grup whatsapp memakai link yang diberikan. Didapat potongan flag pertama, **BeRsamA**.

Kemudian, kami download file yang pada link drive yang diberikan. Didapat potongan flag kedua, **beRJuANG** (ditulis memakai font putih, bisa dibaca jika dipaste ke notepad atau semacamnya).



Kemudian, kami buka file zip **myPrivate.7z** yang juga didapat dari drive sebelumnya. Didapat suatu key .ppk untuk konek ssh ke suatu server (gema@180.250.135.6). IP pada pdf agak tergeser, sehingga harus dikoreksi manual.

Kami pun mengakses ssh menggunakan PuTTY. Akan tetapi, kami hanya menemukan link ke grup telegram di dalamnya. Kami pun masuk ke grup telegram tersebut. Pada awalnya bahkan di grup telegramnya tidak ada potongan flag. Baru beberapa menit kemudian, kami menyadari bahwa potongan flagnya telah ditambahkan.



Decode base64 didapat potongan flag terakhir adalah **keMENANGan**. Akan tetapi, karena potongan flag di server dihapus entah oleh siapa, kami stuck di titik ini cukup lama.

Baru akhirnya dishare sebuah file **kunci.db** di grup wa. Kami langsung buka file db tersebut menggunakan python dan modul sqlite3. Eksekusi query ke tabel kunci, didapat beberapa string base64 yang saat didecode salah satunya menjadi kata **Raih** yang merupakan potongan flag ketiga.

Gabungkan potongan flag didapatlah flag utuh.

Flag: gemastik13{BeRsamA_beRJuANG_Raih_keMENANGan}

F