Logo

Description automatically generated

[Capture The Flag]

**NAMA TIM : [John The Kipper]**

**Institusi : SMK Telkom Malang, SMKN 7 Makassar**

Selasa, 24 November 2020

**Ketua Tim**

1. **MUHAMAD FATHONI**

**Member**

1. **BILL ELIM**
2. **FAHRIZAL AKBAR**

**Table of Content**

├── Crypto

│ ├── Aha

│ ├── Rox

│ ├── Kracken

│ ├── Basic

│ └── Asr

│

├── Forensic

│ ├── Hardwired

│ ├── audit 101

│ └── InspectUs

│

├── Reverse & Pwn

│ ├── Serial

│ ├── Basic

│ └── Apakah Perlu Patching?

│

└── Web

├── Rrrrrrrrrrrrrrrrrrr

├── HLA Basic

└── Easy

### Crypto

#### Aha

Abstraksi

diberikan sebuah file berupa sebuah python script seperti berikut

|  |
| --- |
| """ just learned python and i can't find builtin hashing functions - so, i tried to make one myself. not sure whether it is a hashing or not because it produces diff- erent length for different inpu- ts. but, who cares? """   def my\_hash\_function(c):  chunks = [c[i:i + 2] for i in range(0, len(c), 2)]  rv = [hex(len(c))[2:]]  for ck in chunks:  if len(ck) < 2:  rv.append(hex(ck[0]\*\*2)[2:])  break  rv.append(hex((ck[0]\*\*3 - ck[1]\*\*3) \* ck[1])[2:])  return 'g'.join(rv)   flag\_hash = ('1dg0gx1acf64ga8d80ga8d80g63082b6g2b76d77gx1fe00e6g169b0fcg78c092gx3ab29dcgx347d21g1767ba7g1080168gx24090e8g3d09')  if \_\_name\_\_ == '\_\_main\_\_' \  and my\_hash\_function(input('Validate your Flag = ').encode()) == flag\_hash:  print('congrats!') |

Script tersebut memvalidasi flag kita dengan cara membandingkan fungsi hash custom miliknya pada input kita dengan hash flag yang asli

Pembahasan

bisa dilihat bahwa di my\_hash\_function plaintext kita dibagi bagi menjadi beberapa chunk dimana tiap chunk terdiri atas 2 karakter. Lalu di hash dengan rumus (x\*\*3 - y\*\*3) \* y, dengan asumsi semua karakter pada flag ini printable semua, maka hanya ada paling banyak 128\*128 kemungkinan hash di setiap chunk, sehingga relatif cepat untuk di bruteforce per chunk nya, saya menghilangkan 2 chunk pertama yaitu info jumlah chunk dan 2 huruf pertama yaitu KK karena tidak bisa di bruteforce. chunk terakhir juga tidak ter brute karena rumusnya beda jadi saya tambahkan aja di akhir dengan asumsi karakter terakhir adalah “}”

|  |
| --- |
| flag\_hash = ('x1acf64ga8d80ga8d80g63082b6g2b76d77gx1fe00e6g169b0fcg78c092gx3ab29dcgx347d21g1767ba7g1080168gx24090e8g3d09').split('g')  c = "KK" found = False for i in flag\_hash:  for ckzero in range(0,128):  if found:  found = False  break  for ckone in range(0,128):  if (hex((ckzero\*\*3 - ckone\*\*3) \* ckone)[2:]) == i:  c += chr(ckzero)  c += chr(ckone)  found = True  break c += "}" print(c) |

**Flag : KKST2020{Break\_Something\_Wow}**

#### 

#### 

#### 

#### 

#### 

#### 

#### 

#### Rox

Abstraksi

diberikan file berupa python script seperti berikut

|  |
| --- |
| import string, random, base64  def gen\_key():  k = ''.join([random.choice(string.ascii\_letters) for x in range(0, 3)])  print(k)  return k  def \_cipher(ky, pl):  r = random.randint(0, 10)  print(r)  random.seed(r)  cp = []  for p in pl:  cp.append(hex(ord(p) ^ ord(random.choice(ky))))  return cp  if \_\_name\_\_ == "\_\_main\_\_":  print(base64.b64encode(str(\_cipher(gen\_key(), input("Cipher : "))).encode("utf-8")).decode("utf-8")) |

fungsi gen\_key menciptakan sebuah key yang terdiri dari 3 karakter random

fungsi \_cipher memilih 1 dari 3 karakter itu secara acak untuk kemudian di xor dengan plaintext.

Pembahasan

kelemahannya disini adalah :

1. Kita dapat mengetahui key key yang memungkinkan berdasarkan beberapa karakter awal yang sudah kita ketahui dari flag formatnya, yaitu KKST2020, jadi tinggal di xor ulang untuk mendapatkan karakter key nya (tidak berurutan)
2. kemungkinan seed yang digunakan untuk random.choice hanya sedikit sehingga bisa di bruteforce

karena untuk mendapatkan dekripsi yang pas kita butuh key dengan urutan yang benar, maka tinggal di permute semua kemungkinan key nya, karena hanya ada 3 elemen berarti ada 3\*2\*1=6 kombinasi key, digabungkan dengan 10 kemungkinan seed sehingga hanya butuh 60 loop bruteforce

|  |
| --- |
| # ini python3 ya kalau pake python2 ngga bakal work soalnya random nya beda  from random import seed,randint,choice a = ['0x29', '0x29', '0x31', '0x11', '0x50', '0x5b', '0x59', '0x75', '0x3e', '0x2a', '0x20', '0x28', '0x26', '0x2e', '0x2d', '0x1f'] ky = [[0x62,0x6b,0x45],[0x62,0x45,0x6b],[0x6b,0x62,0x45],[0x6b,0x45,0x62],[0x45,0x6b,0x62],[0x45,0x62,0x6b]] plain = "" for key in ky:  for i in range (0,11):  plain = ""  seed(i)  for k in a:  plain += chr(int(k[2:],16) ^ choice(key))  if "KKST" in plain:  print (plain) |

**Flag : KKST2020{ABCDEF}**

#### 

#### 

#### 

#### 

#### 

#### 

#### 

#### 

#### 

#### 

#### 

#### Kracken

Abstraksi

di deskripsi terdapat sebuah ciphertext dan diberikan sebuah file berupa python script sebagai berikut

|  |
| --- |
| from Crypto.Util.Padding import pad, unpad  from Crypto.Cipher import AES  BLOCK\_SIZE = 32    key = '6b?dadcd478f76?0' # i think i lost my key :( cipher = AES.new(key.encode('utf8'), AES.MODE\_ECB) msg = cipher.encrypt(pad(b'hello\_world', BLOCK\_SIZE)) print(msg.encode('hex')) |

script ini hanya mengimplimentasikan AES ECB dan mengenkripsi kalimat hello\_world menggunakan sebuah key yang sepertinya kehilangan 2 karakter

Pembahasan

karena hanya ada 2 tanda tanya, hingga asumsi saya adalah cukup bruteforce key nya dan decrypt ciphertext di deskripsi dan ternyata berhasil.

|  |
| --- |
| from Crypto.Util.Padding import pad, unpad from Crypto.Cipher import AES BLOCK\_SIZE = 32  for i in "0123456789abcdef":  for j in "0123456789abcdef":  key = '6b'+ i + 'dadcd478f76' + j + '0' # i think i lost my key :(  cipher = AES.new(key.encode('utf8'), AES.MODE\_ECB)  msg = "5ada0e30fd3c562e3db448f17bbd2169a7ba768c8492798698c3acc8446f1486".decode("hex")  result = (cipher.decrypt(msg))  if "KKST" in result:  print result |

**Flag : KKST2020{Gigantic\_Sea\_Monster}**

#### 

#### 

#### Basic

Abstraksi

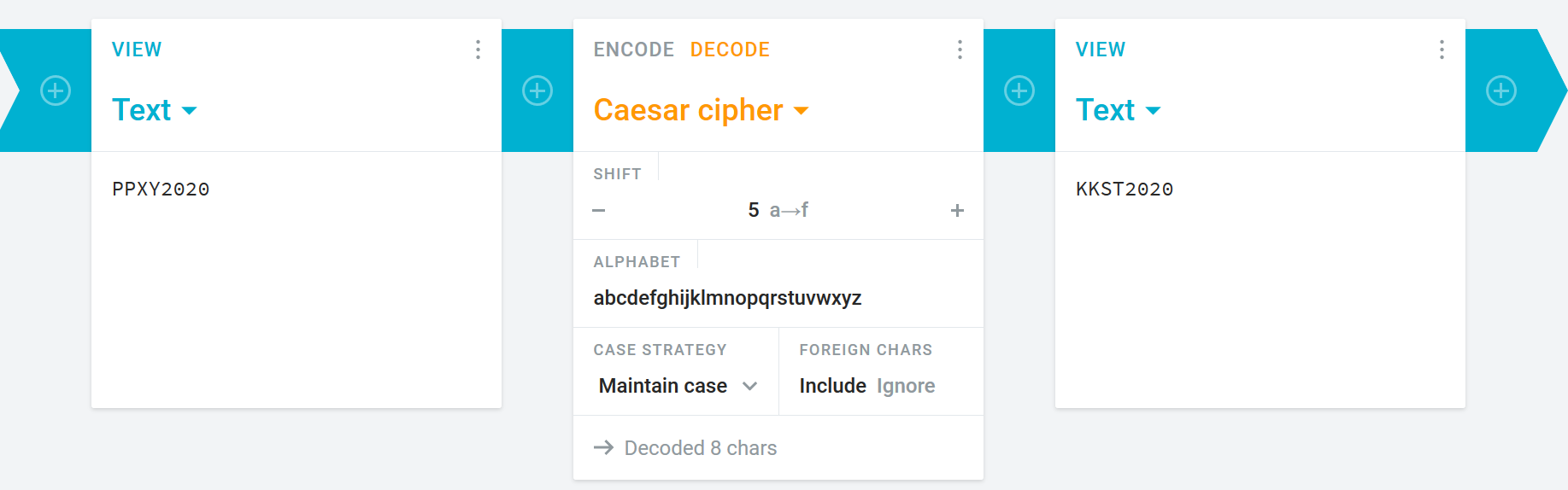
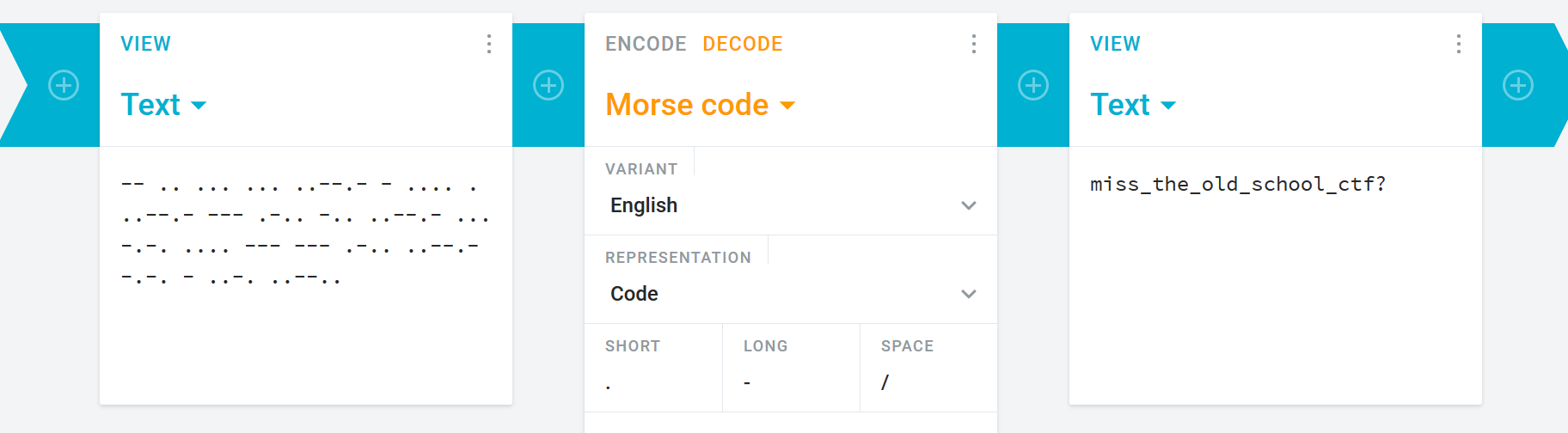
Terdapat soal   
“**PPXY2020{-- .. ... ... ..--.- - .... . ..--.- --- .-.. -.. ..--.- ... -.-. .... --- --- .-.. ..--.- -.-. - ..-. ..--.. }”**

Pembahasan

2. kami berasumsi bahwa

PPXY2020 adalah **Caesar** , sedangkan yang belakang adalah **Morse**.

3. Kami setelah di decode melalui web <https://cryptii.com/>   
hasil nya adalah **KKST2020{miss\_the\_old\_school\_ctf?}**



4. Kami sempat *Stuck* disini selama **> 4 Jam ,** karena submit flag nya gagal terus.

kami mencoba INPUT semua nya mulai dari   
 KKST2020{misstheoldschoolctf?}

KKST2020{MissTheOldSchoolCtf?}

… selama 4 jam kami berfikir dan mencoba

dan Flag yang benar adalah  
**KKST2020{MISSTHEOLDSCHOOLCTF?}**

#### 

#### 

#### Asr

Abstraksi

erdapat sebuah service, ketika kita connect, diberikan 3 angka dengan label “d, n dan c”

|  |
| --- |
| $ nc 207.148.78.100 40001 ---------------------------------------------------------------- d = 95978139776700614937893129799430319466823005747004859783901802545903136026434777023158481332967723496599900078155445907621739016820618426161523758173628060346774815756456124374618660213048443423826534541640999481429005598908334361865804856132709138459291188420529409779696483835002062420310681909332081108711 n = 104888185234423734165056030284556056576859030071913122179412059459189795519252808381451430846532148798136636282534681125186162254925553917148471427075255809267103422294316025998332002892262570195372879288006737978537721370472998598498014922922955760471783311857654515644547878224129150471467234792787390932047 c = 6886662742066684282243034510751751275434685417511688878116374487196733333656199560490798795999700812699701499236876536535502636478664324783259769120576667903967061416337932892403921091497840442848987378637618658397243563216526543079929989528151392855407012880708159979391434568835554730110734871928540239749 |

Pembahasan

Bila menggunakan rumus RSA biasa, tinggal menghitung c\*\*d % n, sayangnya flagnya tidak ketemu, berarti ada salah pelabelan disini. dengan asumsi tidak ada value disini yang dibuat buat hanya untuk mengecoh, maka saya berpikir bahwa yang diberikan sebenarnya merupakan public key, yaitu e,n dan c, dan karena itu value e merupakan value yang terlalu besar untuk value normalnya (3 atau 65537), maka Wiener’s Attack ( <https://en.wikipedia.org/wiki/Wiener%27s_attack> ) bisa dilakukan.

Saya menggunakan implementasi Wiener’s attack dari <https://github.com/pablocelayes/rsa-wiener-attack> , hanya saya modif kodenya sedikit berdasarkan value e dan n yang didapatkan, berikut kode di RSAwienerHacker.py setelah saya modif

|  |
| --- |
| ''' Created on Dec 14, 2011  @author: pablocelayes '''  import ContinuedFractions, Arithmetic, RSAvulnerableKeyGenerator  def hack\_RSA(e,n):  '''  Finds d knowing (e,n)  applying the Wiener continued fraction attack  '''  frac = ContinuedFractions.rational\_to\_contfrac(e, n)  convergents = ContinuedFractions.convergents\_from\_contfrac(frac)    for (k,d) in convergents:    #check if d is actually the key  if k!=0 and (e\*d-1)%k == 0:  phi = (e\*d-1)//k  s = n - phi + 1  # check if the equation x^2 - s\*x + n = 0  # has integer roots  discr = s\*s - 4\*n  if(discr>=0):  t = Arithmetic.is\_perfect\_square(discr)  if t!=-1 and (s+t)%2==0:  print("Hacked!")  return d  # TEST functions  def test\_hack\_RSA():  print("Testing Wiener Attack")  times = 1    while(times>0):  # e,n,d = RSAvulnerableKeyGenerator.generateKeys(1024)  n = 104888185234423734165056030284556056576859030071913122179412059459189795519252808381451430846532148798136636282534681125186162254925553917148471427075255809267103422294316025998332002892262570195372879288006737978537721370472998598498014922922955760471783311857654515644547878224129150471467234792787390932047  e = 95978139776700614937893129799430319466823005747004859783901802545903136026434777023158481332967723496599900078155445907621739016820618426161523758173628060346774815756456124374618660213048443423826534541640999481429005598908334361865804856132709138459291188420529409779696483835002062420310681909332081108711  print("(e,n) is (", e, ", ", n, ")")  # print("d = ", d)    hacked\_d = hack\_RSA(e, n)    # if d == hacked\_d:  # print("Hack WORKED!")  # else:  # print("Hack FAILED")    print(", hacked\_d = ", hacked\_d)  print("-------------------------")  times -= 1   if \_\_name\_\_ == "\_\_main\_\_":  #test\_is\_perfect\_square()  #print("-------------------------")  test\_hack\_RSA() |

output :

|  |
| --- |
| $ python RSAwienerHacker.py Testing Wiener Attack (*'(e,n) is ('*, 95978139776700614937893129799430319466823005747004859783901802545903136026434777023158481332967723496599900078155445907621739016820618426161523758173628060346774815756456124374618660213048443423826534541640999481429005598908334361865804856132709138459291188420529409779696483835002062420310681909332081108711L, *', '*, 104888185234423734165056030284556056576859030071913122179412059459189795519252808381451430846532148798136636282534681125186162254925553917148471427075255809267103422294316025998332002892262570195372879288006737978537721370472998598498014922922955760471783311857654515644547878224129150471467234792787390932047L, *')'*) Hacked! (', hacked\_d = ', 16789392877914656119L) ------------------------- |

dekripsi c:

|  |
| --- |
| >>> d = 16789392877914656119L >>> n = 104888185234423734165056030284556056576859030071913122179412059459189795519252808381451430846532148798136636282534681125186162254925553917148471427075255809267103422294316025998332002892262570195372879288006737978537721370472998598498014922922955760471783311857654515644547878224129150471467234792787390932047 >>> c = 6886662742066684282243034510751751275434685417511688878116374487196733333656199560490798795999700812699701499236876536535502636478664324783259769120576667903967061416337932892403921091497840442848987378637618658397243563216526543079929989528151392855407012880708159979391434568835554730110734871928540239749 >>> hex(pow(c,d,n))[2:-1].decode("hex") 'KKST2020{/LONG/}' |

verifikasi di servicenya

|  |
| --- |
| $ nc 207.148.78.100 40001 ---------------------------------------------------------------- d = 95978139776700614937893129799430319466823005747004859783901802545903136026434777023158481332967723496599900078155445907621739016820618426161523758173628060346774815756456124374618660213048443423826534541640999481429005598908334361865804856132709138459291188420529409779696483835002062420310681909332081108711 n = 104888185234423734165056030284556056576859030071913122179412059459189795519252808381451430846532148798136636282534681125186162254925553917148471427075255809267103422294316025998332002892262570195372879288006737978537721370472998598498014922922955760471783311857654515644547878224129150471467234792787390932047 c = 6886662742066684282243034510751751275434685417511688878116374487196733333656199560490798795999700812699701499236876536535502636478664324783259769120576667903967061416337932892403921091497840442848987378637618658397243563216526543079929989528151392855407012880708159979391434568835554730110734871928540239749  [Guess Flag] > KKST2020{/LONG/}  Correct! |

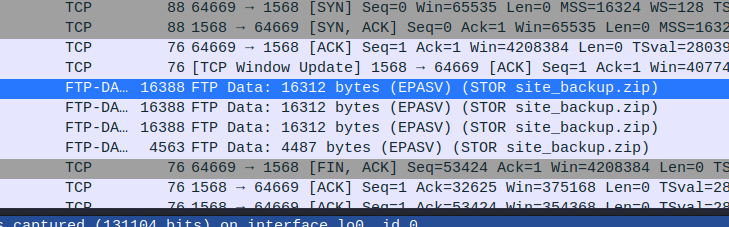
**Flag : KKST2020{/LONG/}**

### Forensic

#### Hardwired

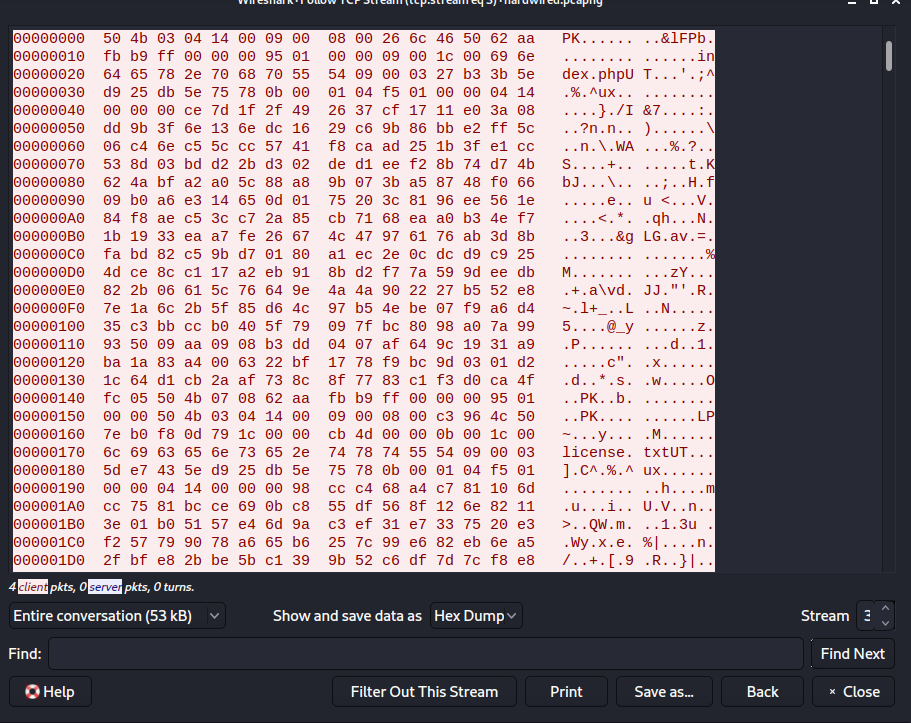
Abstraksi

diberikan sebuah file pcapng, saat diinspeksi ternyata terdapat sebuah file site\_backup.zip yang dipecah pecah menjadi beberapa bagian.



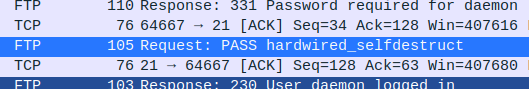
Pembahasan

tinggal follow tcp stream terus save as hexdump dan decode dari hexdump ke text pakai cyberchef (<https://gchq.github.io/CyberChef/#recipe=From_Hexdump()>) terus di unzip deh.



harusnya sih bisa save langsung jadi zip dari wireshark tapi entah kenapa ada sedikit error jadi terpaksa pakai cara ini daripada membuang buang waktu fix errornya.

Sayangnya ketika diunzip file tersebut membutuhkan password, jadi inspeksi file network capturenya lagi dan ternyata ada sebuah password disana yaitu hardwired\_selfdestruct



Setelah di unzip menggunakan password tersebut terdapat file wp\_secret.txt dan disitulah flagnya.

**Flag : KKST2020{wireshark\_is\_not\_hardwired\_to\_self\_destruct}**

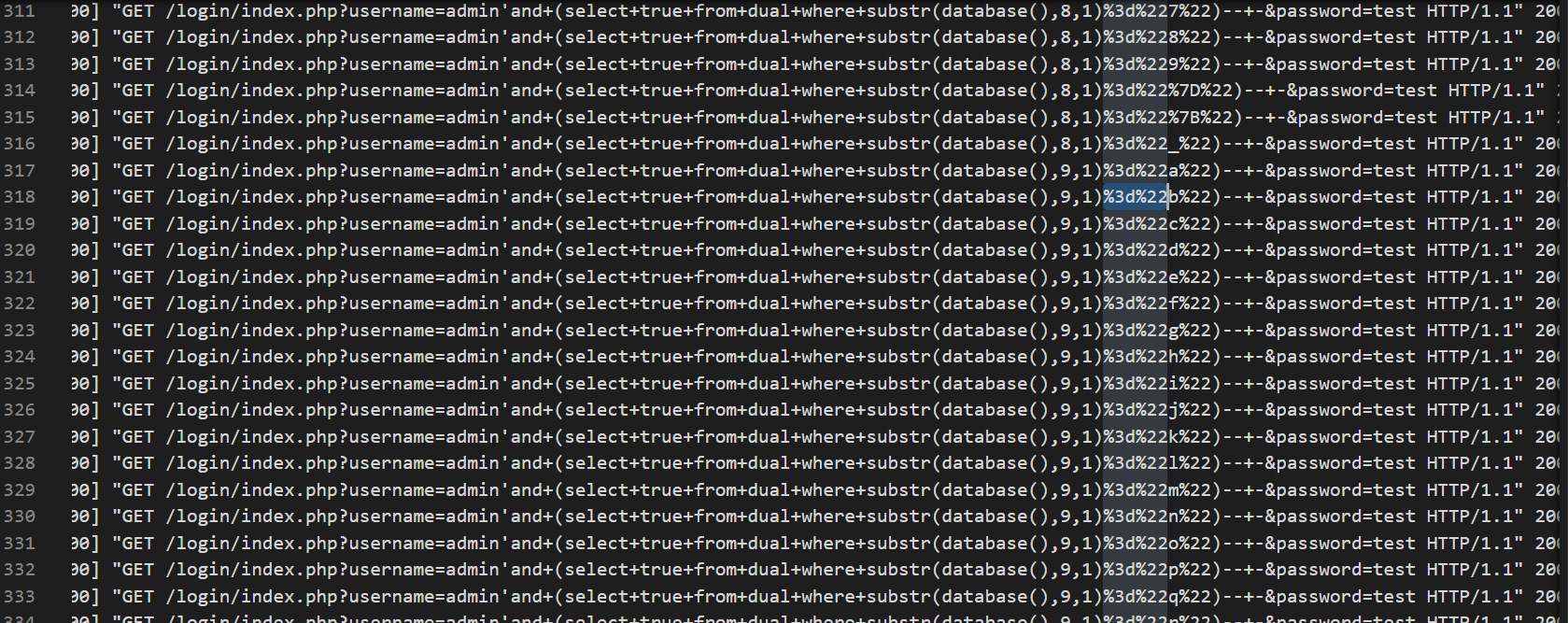
### 

#### 

#### audit 101

Abstraksi

Diberikan file yang bernama access\_log , yang dimana file tersebut kami asumsikan log tersebut adalah log web / log http banyak , serta flag nya tersembunyi pada status angka 364.



Pembahasan

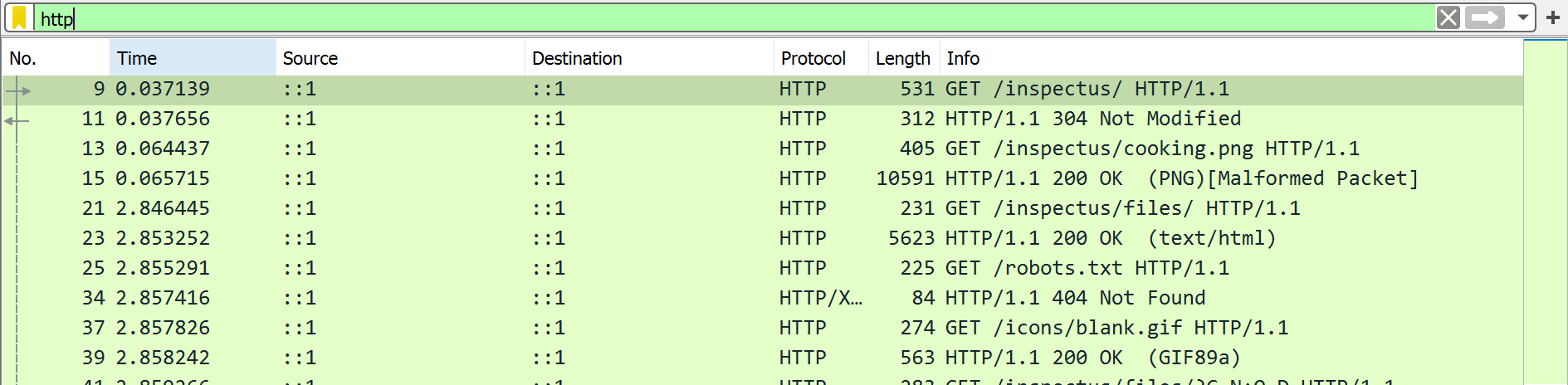
Kami melakukan solve dengan mengambil suatu *line* / baris yang dimana di baris tersebut terdapat angka 364 dengan menggunakan Javascript NodeJS.  
 Setelah menjalankan script tersebut , hasil nya tersimpan pada file res.txt. Kami melakukan *filter* pada file res.txt dengan menghapus beberapa huruf seperti “%3d%22” menggunakan VSCode dan tersusun sebuah flag yaitu **kkst2020{s1mple\_http\_l0g\_aud1t\_101}.**

#### 

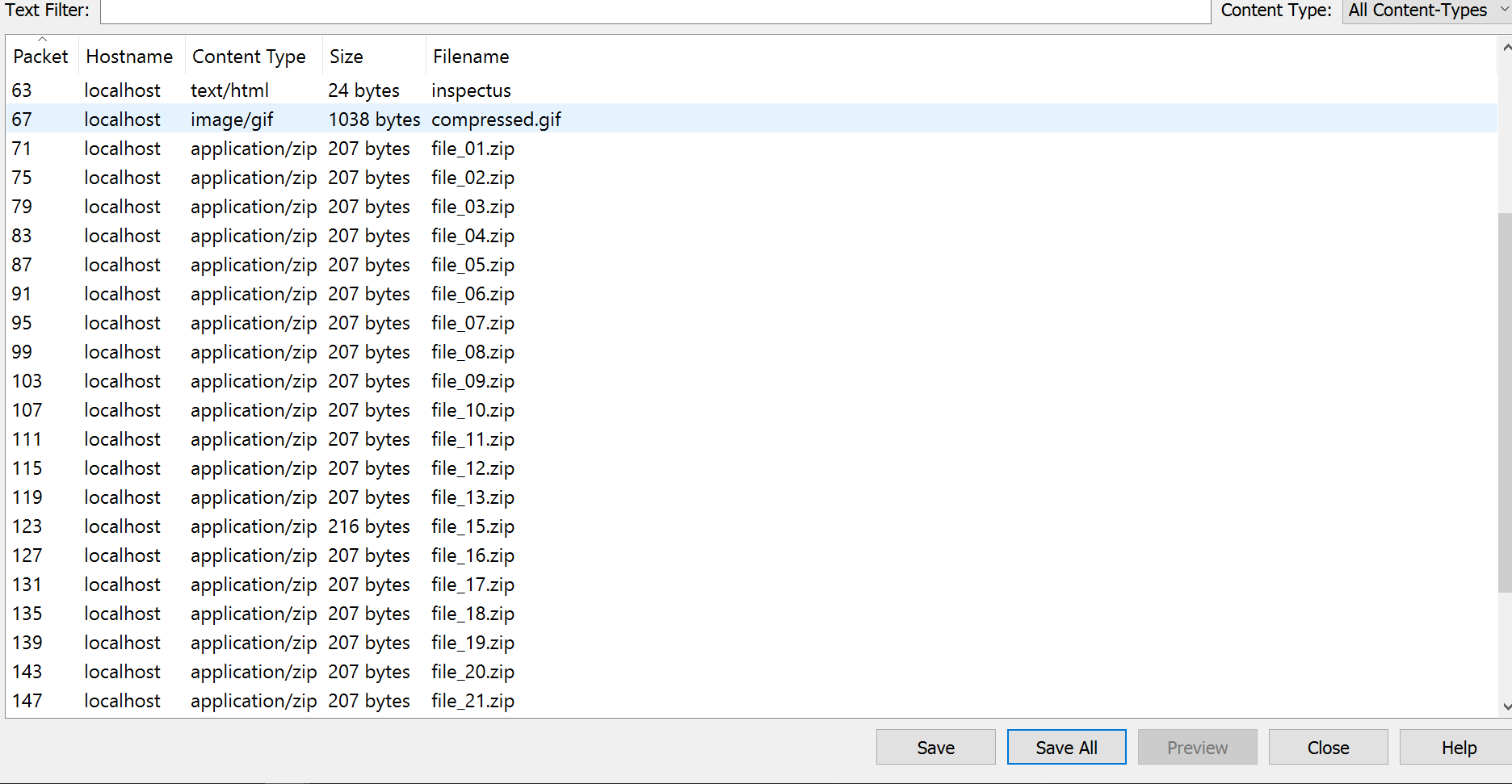
#### 

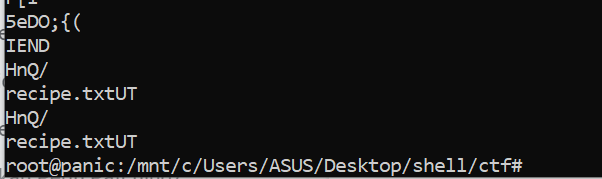
#### InspectUs

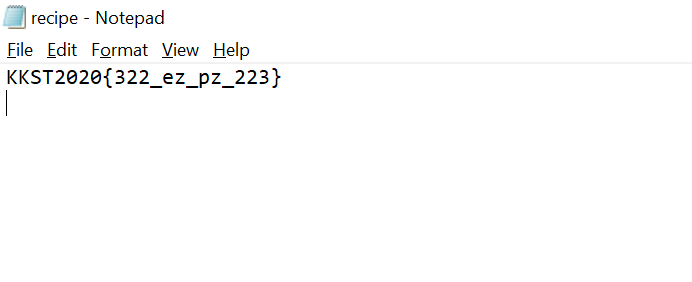
Abstraksi

Diberikan sebuah file Inspect.pcapng yang dimana isi tersebut berisikan paket. kami melakukan analisa dan menyimpulkan bahwa flag tersimpan pada paket protokol http .

Pembahasan

Kami melakukan export http menggunakan tools wireshark dan menyimpan semua paket tersebut. 

Setelah itu kami melakukan perintah ***strings*** pada gambar cooking.png dan kami menemukan bahwa digambar tersebut masih bisa diextrak karena terdapat tulisan recipe.txt.

kami lakukan ***binwalk***dan ternyata file zip nya di password. langkah kami selanjutnya adalah melakukan teknik **Bruteforce** password zip menggunakan tools **John The Ripper** pada setiap file zip dan berasil menemukan sebuah file yang kami asumsikan adalah password untuk membuka recipe.txt yaitu pada ***file\_15.zip***. password file tersebut adalah *999999* dan kami mendapatkan key.txt yang berisikan *youknowwhatscooking?boom* , dan saat kami masukan key tersebut pada recipe.txt untuk membuka password kami berhasil mendapatkan flag. 

**Flag : KKST2020{322\_ez\_pz\_223}**

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### Reversing & PWN

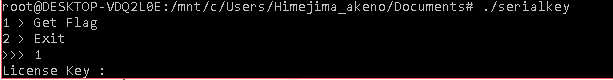
### 

#### Serial

Abstraksi

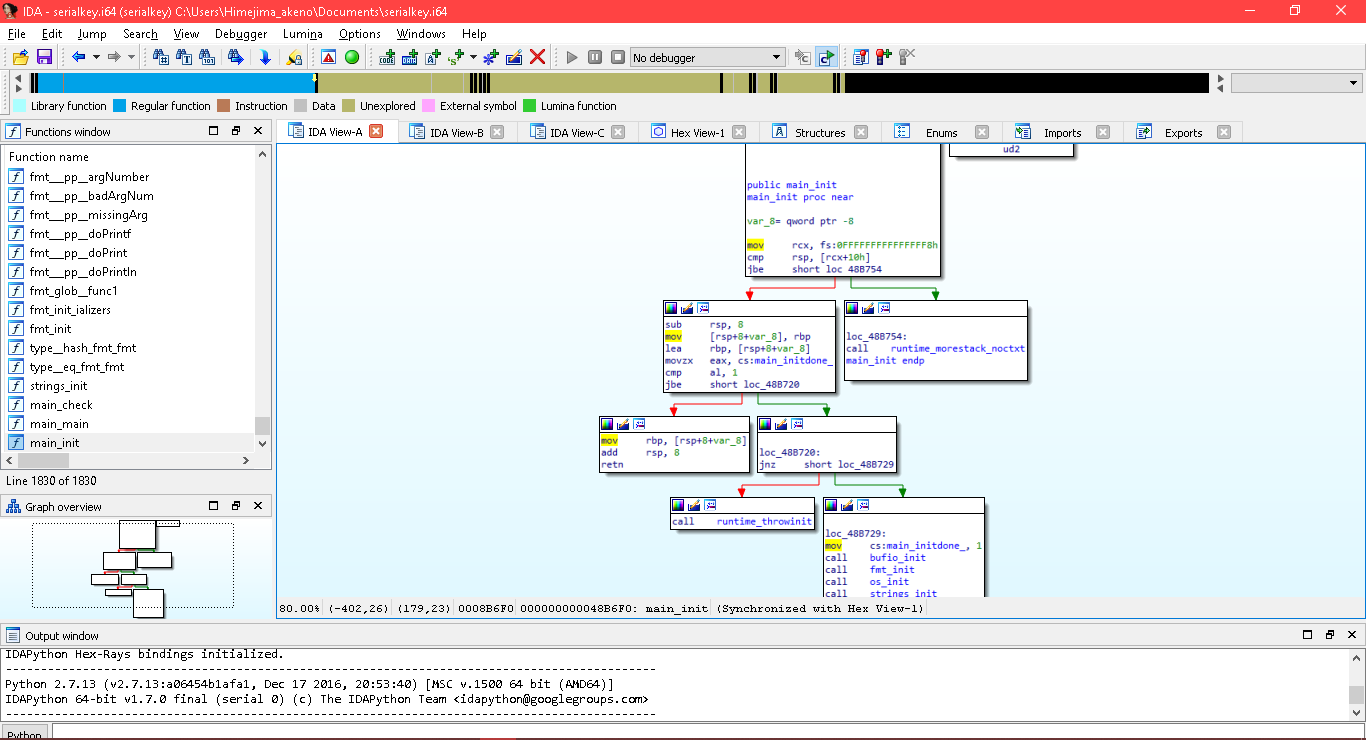


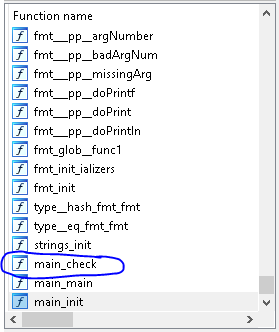
file berbetuk executable dan ketika di run meminta serial key



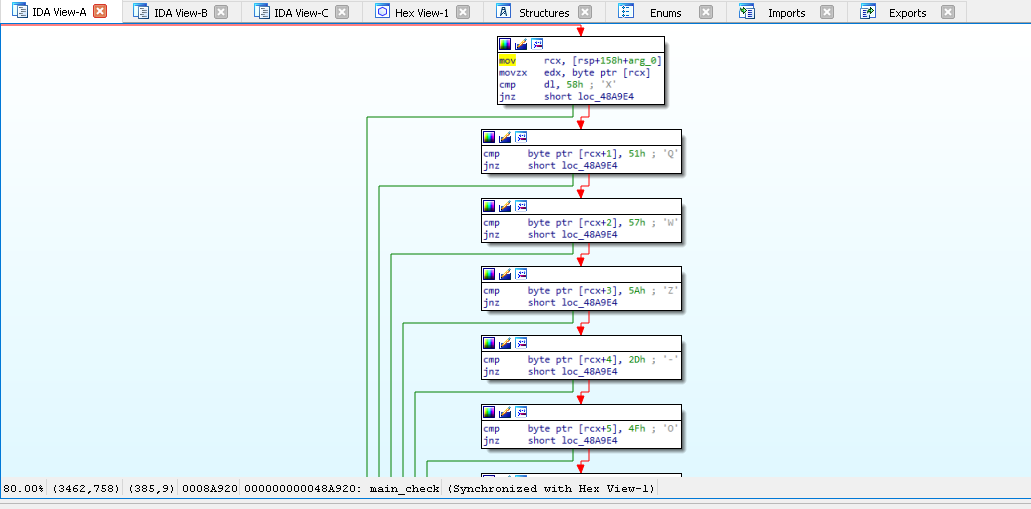
Pembahasan

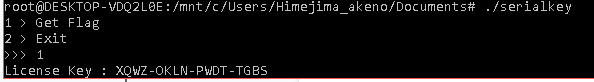
saya mencoba memasukan file nya ke IDA pro 64 saya menemukan Fucntion yang menurut saya aneh yaitu main\_check

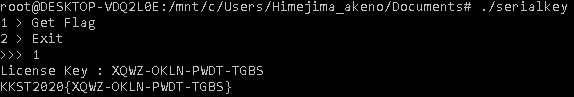




setelah di cek terdapat string yang menurut kami itu adalah License key, String tersebut membentuk XQWZ-OKLN-PWDT-TGBS



kemudian saya masukan ke program Dan muncul Flag nya 



**Flag : KKST2020{XQWZ-OKLN-PWDT-TGBS}**

#### Basic

Abstraksi

Diberikan sebuah file binary, berikut kode main nya dalam assembly (saya menggunakan GDB GEF)

|  |
| --- |
| gef➤ disas main Dump of assembler code for function main:  0x00000000000006fa <+0>: push rbp  0x00000000000006fb <+1>: mov rbp,rsp  0x00000000000006fe <+4>: sub rsp,0x160  0x0000000000000705 <+11>: mov DWORD PTR [rbp-0x154],edi  0x000000000000070b <+17>: mov QWORD PTR [rbp-0x160],rsi  0x0000000000000712 <+24>: mov rax,QWORD PTR fs:0x28  0x000000000000071b <+33>: mov QWORD PTR [rbp-0x8],rax  0x000000000000071f <+37>: xor eax,eax  0x0000000000000721 <+39>: movabs rax,0x735f6362696c5f5f  0x000000000000072b <+49>: movabs rdx,0x69616d5f74726174  0x0000000000000735 <+59>: mov QWORD PTR [rbp-0x150],rax  0x000000000000073c <+66>: mov QWORD PTR [rbp-0x148],rdx  0x0000000000000743 <+73>: mov QWORD PTR [rbp-0x140],0x6e  0x000000000000074e <+84>: mov QWORD PTR [rbp-0x138],0x0  0x0000000000000759 <+95>: mov QWORD PTR [rbp-0x130],0x0  0x0000000000000764 <+106>: mov QWORD PTR [rbp-0x128],0x0  0x000000000000076f <+117>: mov QWORD PTR [rbp-0x120],0x0  0x000000000000077a <+128>: mov QWORD PTR [rbp-0x118],0x0  0x0000000000000785 <+139>: mov QWORD PTR [rbp-0x110],0x0  0x0000000000000790 <+150>: mov QWORD PTR [rbp-0x108],0x0  0x000000000000079b <+161>: mov QWORD PTR [rbp-0x100],0x0  0x00000000000007a6 <+172>: mov QWORD PTR [rbp-0xf8],0x0  0x00000000000007b1 <+183>: mov DWORD PTR [rbp-0xf0],0x0  0x00000000000007bb <+193>: movabs rax,0x5f6b636174735f5f  0x00000000000007c5 <+203>: movabs rdx,0x6c6961665f6b6863  0x00000000000007cf <+213>: mov QWORD PTR [rbp-0xe0],rax  0x00000000000007d6 <+220>: mov QWORD PTR [rbp-0xd8],rdx  0x00000000000007dd <+227>: mov QWORD PTR [rbp-0xd0],0x0  0x00000000000007e8 <+238>: mov QWORD PTR [rbp-0xc8],0x0  0x00000000000007f3 <+249>: mov QWORD PTR [rbp-0xc0],0x0  0x00000000000007fe <+260>: mov QWORD PTR [rbp-0xb8],0x0  0x0000000000000809 <+271>: mov QWORD PTR [rbp-0xb0],0x0  0x0000000000000814 <+282>: mov QWORD PTR [rbp-0xa8],0x0  0x000000000000081f <+293>: mov QWORD PTR [rbp-0xa0],0x0  0x000000000000082a <+304>: mov QWORD PTR [rbp-0x98],0x0  0x0000000000000835 <+315>: mov QWORD PTR [rbp-0x90],0x0  0x0000000000000840 <+326>: mov QWORD PTR [rbp-0x88],0x0  0x000000000000084b <+337>: mov DWORD PTR [rbp-0x80],0x0  0x0000000000000852 <+344>: movabs rax,0x2e6f732e6362696c  0x000000000000085c <+354>: mov edx,0x36  0x0000000000000861 <+359>: mov QWORD PTR [rbp-0x70],rax  0x0000000000000865 <+363>: mov QWORD PTR [rbp-0x68],rdx  0x0000000000000869 <+367>: lea rdx,[rbp-0x60]  0x000000000000086d <+371>: mov eax,0x0  0x0000000000000872 <+376>: mov ecx,0xa  0x0000000000000877 <+381>: mov rdi,rdx  0x000000000000087a <+384>: rep stos QWORD PTR es:[rdi],rax  0x000000000000087d <+387>: mov rdx,rdi  0x0000000000000880 <+390>: mov DWORD PTR [rdx],eax  0x0000000000000882 <+392>: add rdx,0x4  0x0000000000000886 <+396>: mov rax,QWORD PTR [rbp-0x160]  0x000000000000088d <+403>: add rax,0x8  0x0000000000000891 <+407>: mov rdx,QWORD PTR [rax]  0x0000000000000894 <+410>: lea rax,[rbp-0x150]  0x000000000000089b <+417>: mov rsi,rdx  0x000000000000089e <+420>: mov rdi,rax  0x00000000000008a1 <+423>: call 0x5d0 <strcmp@plt>  0x00000000000008a6 <+428>: test eax,eax  0x00000000000008a8 <+430>: jne 0x931 <main+567>  0x00000000000008ae <+436>: mov rax,QWORD PTR [rbp-0x160]  0x00000000000008b5 <+443>: add rax,0x10  0x00000000000008b9 <+447>: mov rdx,QWORD PTR [rax]  0x00000000000008bc <+450>: lea rax,[rbp-0xe0]  0x00000000000008c3 <+457>: mov rsi,rdx  0x00000000000008c6 <+460>: mov rdi,rax  0x00000000000008c9 <+463>: call 0x5d0 <strcmp@plt>  0x00000000000008ce <+468>: test eax,eax  0x00000000000008d0 <+470>: jne 0x931 <main+567>  0x00000000000008d2 <+472>: mov rax,QWORD PTR [rbp-0x160]  0x00000000000008d9 <+479>: add rax,0x18  0x00000000000008dd <+483>: mov rdx,QWORD PTR [rax]  0x00000000000008e0 <+486>: lea rax,[rbp-0x70]  0x00000000000008e4 <+490>: mov rsi,rdx  0x00000000000008e7 <+493>: mov rdi,rax  0x00000000000008ea <+496>: call 0x5d0 <strcmp@plt>  0x00000000000008ef <+501>: test eax,eax  0x00000000000008f1 <+503>: jne 0x931 <main+567>  0x00000000000008f3 <+505>: mov rax,QWORD PTR [rbp-0x160]  0x00000000000008fa <+512>: add rax,0x18  0x00000000000008fe <+516>: mov rcx,QWORD PTR [rax]  0x0000000000000901 <+519>: mov rax,QWORD PTR [rbp-0x160]  0x0000000000000908 <+526>: add rax,0x10  0x000000000000090c <+530>: mov rdx,QWORD PTR [rax]  0x000000000000090f <+533>: mov rax,QWORD PTR [rbp-0x160]  0x0000000000000916 <+540>: add rax,0x8  0x000000000000091a <+544>: mov rax,QWORD PTR [rax]  0x000000000000091d <+547>: mov rsi,rax  0x0000000000000920 <+550>: lea rdi,[rip+0xad] # 0x9d4  0x0000000000000927 <+557>: mov eax,0x0  0x000000000000092c <+562>: call 0x5c0 <printf@plt>  0x0000000000000931 <+567>: mov eax,0x0  0x0000000000000936 <+572>: mov rcx,QWORD PTR [rbp-0x8]  0x000000000000093a <+576>: xor rcx,QWORD PTR fs:0x28  0x0000000000000943 <+585>: je 0x94a <main+592>  0x0000000000000945 <+587>: call 0x5b0 <\_\_stack\_chk\_fail@plt>  0x000000000000094a <+592>: leave   0x000000000000094b <+593>: ret  End of assembler dump. |

Pembahasan

Sederhananya ada 3 strcmp yang dilakukan, lalu ada beberapa movabs yang bisa jadi merupakan string yang akan di compare dengan input kita. kita decode movabs nya satu satu.

|  |
| --- |
| 0x0000000000000721 <+39>: movabs rax,0x735f6362696c5f5f > \_\_libc\_s 0x000000000000072b <+49>: movabs rdx,0x69616d5f74726174 > tart\_mai  0x0000000000000735 <+59>: mov QWORD PTR [rbp-0x150],rax > \_\_libc\_s 0x000000000000073c <+66>: mov QWORD PTR [rbp-0x148],rdx > tart\_mai 0x0000000000000743 <+73>: mov QWORD PTR [rbp-0x140],0x6e > n  \_\_libc\_start\_main  0x00000000000007bb <+193>: movabs rax,0x5f6b636174735f5f > \_\_stack\_ 0x00000000000007c5 <+203>: movabs rdx,0x6c6961665f6b6863 > chk\_fail  0x00000000000007cf <+213>: mov QWORD PTR [rbp-0xe0],rax > \_\_stack\_ 0x00000000000007d6 <+220>: mov QWORD PTR [rbp-0xd8],rdx > chk\_fail 0x00000000000007dd <+227>: mov QWORD PTR [rbp-0xd0],0x0  \_\_stack\_chk\_fail  0x0000000000000852 <+344>: movabs rax,0x2e6f732e6362696c > libc.so. 0x000000000000085c <+354>: mov edx,0x36 > 6  0x0000000000000861 <+359>: mov QWORD PTR [rbp-0x70],rax > libc.so. 0x0000000000000865 <+363>: mov QWORD PTR [rbp-0x68],rdx > 6  libc.so.6 |

ketemu deh 3 string yang kita perlu input

|  |
| --- |
| gef➤ r \_\_libc\_start\_main \_\_stack\_chk\_fail libc.so.6 Starting program: /home/wrth/CTF/kks-tni-ad/rev/s \_\_libc\_start\_main \_\_stack\_chk\_fail libc.so.6 KKST2020{\_\_libc\_start\_main\_\_\_stack\_chk\_fail\_libc.so.6} [Inferior 1 (process 9241) exited normally] |

**Flag : KKST2020{\_\_libc\_start\_main\_\_\_stack\_chk\_fail\_libc.so.6}**

#### Apakah Perlu Patching?

Abstraksi

Diberikan sebuah file binary, berikut kode main nya dalam assembly (saya menggunakan GDB GEF)

|  |
| --- |
| gef➤ disas main Dump of assembler code for function main:  0x000055555555482a <+0>: push rbp  0x000055555555482b <+1>: mov rbp,rsp  0x000055555555482e <+4>: lea rdi,[rip+0xc7] # 0x5555555548fc  0x0000555555554835 <+11>: call 0x5555555545b0 <puts@plt>  0x000055555555483a <+16>: mov eax,0x0  0x000055555555483f <+21>: pop rbp  0x0000555555554840 <+22>: ret  End of assembler dump. |

Sepertinya tidak ada apa apa, bila di run juga yang di puts hanyalah “Welcome To KKST2020”,

Pembahasan

coba kita baca dari entrypoint nya saja

|  |
| --- |
| gef➤ info files Symbols from "/home/wrth/CTF/kks-tni-ad/rev/cmad". Local exec file:  `/home/wrth/CTF/kks-tni-ad/rev/cmad', file type elf64-x86-64.  Entry point: 0x5555555545f0  0x0000555555554238 - 0x0000555555554254 is .interp ... CUT...  0x00005555555545a0 - 0x00005555555545e0 is .plt  0x00005555555545e0 - 0x00005555555545e8 is .plt.got  0x00005555555545f0 - 0x00005555555548c2 is .text  0x00005555555548c4 - 0x00005555555548cd is .fini ...CUT… |

|  |
| --- |
| gef➤ x/150i 0x00005555555545f0  0x5555555545f0 <\_start>: xor ebp,ebp  0x5555555545f2 <\_start+2>: mov r9,rdx  0x5555555545f5 <\_start+5>: pop rsi  ...CUT...  0x5555555546fa <initiateflag>: push rbp  0x5555555546fb <initiateflag+1>: mov rbp,rsp  0x5555555546fe <initiateflag+4>: movabs rax,0x5372616a616c6550  0x555555554708 <initiateflag+14>: movabs rdx,0x736f6c6f42616b75  0x555555554712 <initiateflag+24>: mov QWORD PTR [rip+0x200927],rax # 0x555555755040 <flag>   0x555555554719 <initiateflag+31>: mov QWORD PTR [rip+0x200928],rdx # 0x555555755048 <flag+8>   0x555555554720 <initiateflag+38>: mov DWORD PTR [rip+0x200926],0x6e614b # 0x555555755050 <flag+16>  0x55555555472a <initiateflag+48>: nop  0x55555555472b <initiateflag+49>: pop rbp  0x55555555472c <initiateflag+50>: ret   0x55555555472d <validator>: push rbp  0x55555555472e <validator+1>: mov rbp,rsp  0x555555554731 <validator+4>: add rsp,0xffffffffffffff80  0x555555554735 <validator+8>: mov rax,QWORD PTR fs:0x28  0x55555555473e <validator+17>: mov QWORD PTR [rbp-0x8],rax  0x555555554742 <validator+21>: xor eax,eax  0x555555554744 <validator+23>: lea rdx,[rbp-0x70]  0x555555554748 <validator+27>: mov eax,0x0  0x55555555474d <validator+32>: mov ecx,0xc  0x555555554752 <validator+37>: mov rdi,rdx  0x555555554755 <validator+40>: rep stos QWORD PTR es:[rdi],rax  0x555555554758 <validator+43>: mov rdx,rdi  0x55555555475b <validator+46>: mov DWORD PTR [rdx],eax  0x55555555475d <validator+48>: add rdx,0x4  0x555555554761 <validator+52>: mov BYTE PTR [rbp-0x70],0x50  0x555555554765 <validator+56>: mov BYTE PTR [rbp-0x6f],0x65  0x555555554769 <validator+60>: mov BYTE PTR [rbp-0x6e],0x6c  0x55555555476d <validator+64>: mov BYTE PTR [rbp-0x6d],0x61  0x555555554771 <validator+68>: mov BYTE PTR [rbp-0x6c],0x6a  0x555555554775 <validator+72>: mov BYTE PTR [rbp-0x6b],0x61  0x555555554779 <validator+76>: mov BYTE PTR [rbp-0x6a],0x72  0x55555555477d <validator+80>: mov BYTE PTR [rbp-0x69],0x53  0x555555554781 <validator+84>: mov BYTE PTR [rbp-0x68],0x75  0x555555554785 <validator+88>: mov BYTE PTR [rbp-0x67],0x6b  0x555555554789 <validator+92>: mov BYTE PTR [rbp-0x66],0x61  0x55555555478d <validator+96>: mov BYTE PTR [rbp-0x65],0x42  0x555555554791 <validator+100>: mov BYTE PTR [rbp-0x64],0x65  0x555555554795 <validator+104>: mov BYTE PTR [rbp-0x63],0x6c  0x555555554799 <validator+108>: mov BYTE PTR [rbp-0x62],0x61  0x55555555479d <validator+112>: mov BYTE PTR [rbp-0x61],0x6a  0x5555555547a1 <validator+116>: mov BYTE PTR [rbp-0x60],0x61  0x5555555547a5 <validator+120>: mov BYTE PTR [rbp-0x5f],0x72  0x5555555547a9 <validator+124>: mov BYTE PTR [rbp-0x5e],0x3f  0x5555555547ad <validator+128>: mov DWORD PTR [rbp-0x74],0x0  0x5555555547b4 <validator+135>: jmp 0x5555555547f1 <validator+196>  0x5555555547b6 <validator+137>: mov eax,DWORD PTR [rbp-0x74]  0x5555555547b9 <validator+140>: movsxd rdx,eax  0x5555555547bc <validator+143>: lea rax,[rip+0x20087d] # 0x555555755040 <flag>   0x5555555547c3 <validator+150>: movzx edx,BYTE PTR [rdx+rax\*1]  0x5555555547c7 <validator+154>: mov eax,DWORD PTR [rbp-0x74]  0x5555555547ca <validator+157>: cdqe   0x5555555547cc <validator+159>: movzx eax,BYTE PTR [rbp+rax\*1-0x70]  0x5555555547d1 <validator+164>: cmp dl,al  0x5555555547d3 <validator+166>: je 0x5555555547ed <validator+192>  0x5555555547d5 <validator+168>: lea rdi,[rip+0xf8] # 0x5555555548d4  0x5555555547dc <validator+175>: mov eax,0x0  0x5555555547e1 <validator+180>: call 0x5555555545d0 <printf@plt>  0x5555555547e6 <validator+185>: mov eax,0x0  0x5555555547eb <validator+190>: jmp 0x555555554814 <validator+231>  0x5555555547ed <validator+192>: add DWORD PTR [rbp-0x74],0x1  0x5555555547f1 <validator+196>: cmp DWORD PTR [rbp-0x74],0x12  0x5555555547f5 <validator+200>: jle 0x5555555547b6 <validator+137>  0x5555555547f7 <validator+202>: lea rsi,[rip+0x200842] # 0x555555755040 <flag>   0x5555555547fe <validator+209>: lea rdi,[rip+0xe9] # 0x5555555548ee  0x555555554805 <validator+216>: mov eax,0x0  0x55555555480a <validator+221>: call 0x5555555545d0 <printf@plt>  0x55555555480f <validator+226>: mov eax,0x0  0x555555554814 <validator+231>: mov rsi,QWORD PTR [rbp-0x8]  0x555555554818 <validator+235>: xor rsi,QWORD PTR fs:0x28  0x555555554821 <validator+244>: je 0x555555554828 <validator+251>  0x555555554823 <validator+246>: call 0x5555555545c0 <\_\_stack\_chk\_fail@plt>  0x555555554828 <validator+251>: leave   0x555555554829 <validator+252>: ret   0x55555555482a <main>: push rbp  0x55555555482b <main+1>: mov rbp,rsp  0x55555555482e <main+4>: lea rdi,[rip+0xc7] # 0x5555555548fc  0x555555554835 <main+11>: call 0x5555555545b0 <puts@plt>  0x55555555483a <main+16>: mov eax,0x0  0x55555555483f <main+21>: pop rbp  0x555555554840 <main+22>: ret |

Ada 2 fungsi yang menarik, yaitu initiateflag dan validator, di initiateflag terdapat movabs rax,0x5372616a616c6550 dan movabs rdx,0x736f6c6f42616b75, yang bila didecode menjadi PelajarSukaBolos, bila kita mencoba submit itu sebagai flag, ternyata incorrect, jadi kita harus melihat fungsi kedua yaitu validator, terdapat beberapa replacement character di validator.

|  |
| --- |
| 0x555555554761 <validator+52>: mov BYTE PTR [rbp-0x70],0x50  0x555555554765 <validator+56>: mov BYTE PTR [rbp-0x6f],0x65  0x555555554769 <validator+60>: mov BYTE PTR [rbp-0x6e],0x6c  0x55555555476d <validator+64>: mov BYTE PTR [rbp-0x6d],0x61  0x555555554771 <validator+68>: mov BYTE PTR [rbp-0x6c],0x6a  0x555555554775 <validator+72>: mov BYTE PTR [rbp-0x6b],0x61  0x555555554779 <validator+76>: mov BYTE PTR [rbp-0x6a],0x72  0x55555555477d <validator+80>: mov BYTE PTR [rbp-0x69],0x53  0x555555554781 <validator+84>: mov BYTE PTR [rbp-0x68],0x75  0x555555554785 <validator+88>: mov BYTE PTR [rbp-0x67],0x6b  0x555555554789 <validator+92>: mov BYTE PTR [rbp-0x66],0x61  0x55555555478d <validator+96>: mov BYTE PTR [rbp-0x65],0x42  0x555555554791 <validator+100>: mov BYTE PTR [rbp-0x64],0x65  0x555555554795 <validator+104>: mov BYTE PTR [rbp-0x63],0x6c  0x555555554799 <validator+108>: mov BYTE PTR [rbp-0x62],0x61  0x55555555479d <validator+112>: mov BYTE PTR [rbp-0x61],0x6a  0x5555555547a1 <validator+116>: mov BYTE PTR [rbp-0x60],0x61  0x5555555547a5 <validator+120>: mov BYTE PTR [rbp-0x5f],0x72  0x5555555547a9 <validator+124>: mov BYTE PTR [rbp-0x5e],0x3f |

mari kita coba untuk breakpoint di instruction setelah instruction ini dan memeriksa bagian memorynya untuk mengetahui apa yang berubah

|  |
| --- |
| gef➤ b \*0x5555555547b4 Breakpoint 1 at 0x5555555547b4 gef➤ b main Breakpoint 2 at 0x55555555482e gef➤ r Starting program: /home/wrth/CTF/kks-tni-ad/rev/cmad   Breakpoint 2, 0x000055555555482e in main ()  gef➤ jump validator Continuing at 0x555555554731.  Breakpoint 1, 0x00005555555547b4 in validator () [ Legend: Modified register | Code | Heap | Stack | String ] ─────────────────────────────────────────────────────────────────────── registers ──── $rax : 0x0  $rbx : 0x0  $rcx : 0x0  $rdx : 0x00007fffffffe064 → 0xf251db0000007fff $rsp : 0x00007fffffffdff0 → 0x0000000000000000 $rbp : 0x00007fffffffe070 → 0x0000555555554850 → <\_\_libc\_csu\_init+0> push r15 $rsi : 0x00007fffffffe158 → 0x00007fffffffe468 → "/home/wrth/CTF/kks-tni-ad/rev/cmad" $rdi : 0x00007fffffffe060 → 0x00007fff00000000 $rip : 0x00005555555547b4 → <validator+135> jmp 0x5555555547f1 <validator+196> $r8 : 0x0  $r9 : 0x00007ffff7fe3530 → <\_dl\_fini+0> push rbp $r10 : 0x00007ffff7f73a83 → "\_\_vdso\_getcpu" $r11 : 0x206  $r12 : 0x00005555555545f0 → <\_start+0> xor ebp, ebp $r13 : 0x00007fffffffe150 → 0x0000000000000001 $r14 : 0x0  $r15 : 0x0  $eflags: [zero carry parity adjust sign trap INTERRUPT direction overflow resume virtualx86 identification] $cs: 0x0033 $ss: 0x002b $ds: 0x0000 $es: 0x0000 $fs: 0x0000 $gs: 0x0000  ─────────────────────────────────────────────────────────────────────────── stack ──── 0x00007fffffffdff0│+0x0000: 0x0000000000000000 ← $rsp 0x00007fffffffdff8│+0x0008: 0x0000000000000000 0x00007fffffffe000│+0x0010: "PelajarSukaBelajar?" 0x00007fffffffe008│+0x0018: "ukaBelajar?" 0x00007fffffffe010│+0x0020: 0x00000000003f7261 ("ar?"?) 0x00007fffffffe018│+0x0028: 0x0000000000000000 0x00007fffffffe020│+0x0030: 0x0000000000000000 0x00007fffffffe028│+0x0038: 0x0000000000000000 ───────────────────────────────────────────────────────────────────── code:x86:64 ────  0x5555555547a2 <validator+117> rex.RB movabs al, ds:0xa245c672a145c661  0x5555555547ac <validator+127> (bad)   0x5555555547ad <validator+128> mov DWORD PTR [rbp-0x74], 0x0  → 0x5555555547b4 <validator+135> jmp 0x5555555547f1 <validator+196>  0x5555555547b6 <validator+137> mov eax, DWORD PTR [rbp-0x74]  0x5555555547b9 <validator+140> movsxd rdx, eax  0x5555555547bc <validator+143> lea rax, [rip+0x20087d] # 0x555555755040 <flag>  0x5555555547c3 <validator+150> movzx edx, BYTE PTR [rdx+rax\*1]  0x5555555547c7 <validator+154> mov eax, DWORD PTR [rbp-0x74] ───────────────────────────────────────────────────────────────────────── threads ──── [#0] Id 1, Name: "cmad", stopped 0x5555555547b4 in validator (), reason: BREAKPOINT ─────────────────────────────────────────────────────────────────────────── trace ──── [#0] 0x5555555547b4 → validator() [#1] 0x7ffff7e12e0b → \_\_libc\_start\_main(main=0x55555555482a <main>, argc=0x1, argv=0x7fffffffe158, init=<optimized out>, fini=<optimized out>, rtld\_fini=<optimized out>, stack\_end=0x7fffffffe148)  [#2] 0x55555555461a → \_start() |

.

Ternyata kata “bolos” diganti menjadi “belajar?”, dan ketika kita coba submit sebagai flag ternyata benar.

**Flag : KKST2020{PelajarSukaBelajar?}**

### Web

#### Rrrrrrrrrrrrrrrrrrr

Abstraksi

Diberikan sebuah link, dimana, ketika link tersebut dibuka di browser biasa, akan memunculkan error karena website tersebut terlalu banyak melakukan redirect

Pembahasan

Isi konten dari web tersebut dapat kita lihat dengan menggunakan curl:

|  |
| --- |
| $ curl -v --path-as-is http://207.148.78.100:20002/ \* Trying 207.148.78.100... \* TCP\_NODELAY set \* Connected to 207.148.78.100 (207.148.78.100) port 20002 (#0) > GET / HTTP/1.1 > Host: 207.148.78.100:20002 > User-Agent: curl/7.55.1 > Accept: \*/\* > < HTTP/1.1 302 Found < Date: Tue, 24 Nov 2020 15:36:12 GMT < Server: Apache/2.4.38 (Debian) < X-Powered-By: PHP/7.2.34 < location: r1.php < Content-Length: 0 < Content-Type: text/html; charset=UTF-8 < \* Connection #0 to host 207.148.78.100 left intact |

Dapat dilihat bahwa kita ter redirect ke r1.php, bila kita curl ke sana kita dapat melihat lagi bahwa kita akan ter redirect ke r3.php, dan begitu seterusnya. Saat nge curl satu satu, ternyata ada flag tersembunyi di r3.php

|  |
| --- |
| $ curl -v --path-as-is http://207.148.78.100:20002/r3.php \* Trying 207.148.78.100... \* TCP\_NODELAY set \* Connected to 207.148.78.100 (207.148.78.100) port 20002 (#0) > GET /r3.php HTTP/1.1 > Host: 207.148.78.100:20002 > User-Agent: curl/7.55.1 > Accept: \*/\* > < HTTP/1.1 302 Found < Date: Tue, 24 Nov 2020 15:36:22 GMT < Server: Apache/2.4.38 (Debian) < X-Powered-By: PHP/7.2.34 < location: r4.php < Content-Length: 26 < Content-Type: text/html; charset=UTF-8 < KKST2020{TooMany\_Red1r3ct}\* Connection #0 to host 207.148.78.100 left intact |

**Flag : KKST2020{TooMany\_Red1r3ct}**

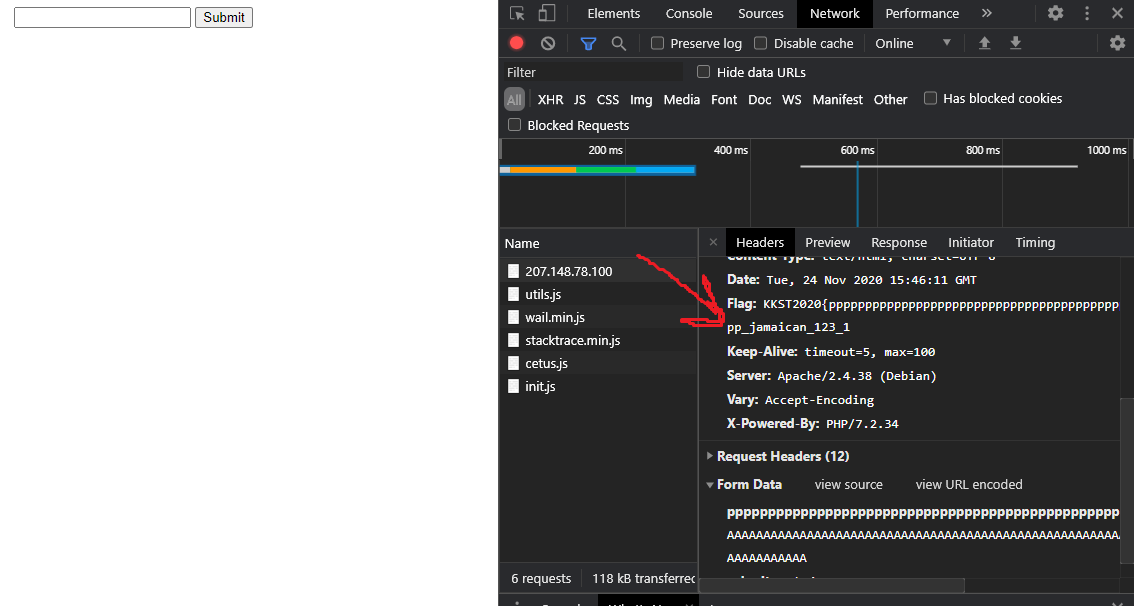
#### HLA basic

Abstraksi

Diberikan sebuah link dan pada link tersebut terdapat sebuah form kecil untuk submit sesuatu, sekilas tidak ada output apa apa ketika kita menyubmit sesuatu

Pembahasan

Ketika melihat response header, dapat dilihat bahwa terdapat respon bernama flag



Dengan mengirimkan jumlah karakter yang tepat, yaitu 104 karakter, maka kita dapat mendapatkan flagnya secara penuh. Bila terlalu banyak nanti flagnya tidak akan keluar.

Flag: **KKST2020{pppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppppp\_jamaican\_123\_123@}**

#### 

#### Easy

Abstraksi

Diberikan sebuah link yang berisi semacam blog sederhana.

Pembahasan

Cukup mengakses /flag.txt dan kita akan mendapatkan flagnya <http://207.148.78.100:20003/flag.txt> .

Sepertinya ada sedikit kesalahan di challenge yang ini, mungkin seharusnya flag.txt ini tidak publik dan hanya dapat diakses berdasarkan celah SQL injection yang berada di index.php?id= , contohnya seperti ini ( <http://207.148.78.100:20003/index.php?id=%27%29%20UNION%20SELECT%20%27XXX%27%2c%27YYYY%27%2c%27ZZZZ%27%2c%28%27z> ). kami memberitahu untuk sekedar untuk berjaga jaga saja agar poinnya tidak invalid

**Flag : KKST2020{FLAGGGGGG}**